Finanzperspektiven der IV

Technische Dokumentation

BSV MAS (sekretariat.mas@bsv.admin.ch)

06.11.2024

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Implementation des Finanzperspektivenmodells der IV in der Programmiersprache R. Eine nicht-technische Zusammenfassung des Modellansatzes findet sich im Dokument „Modellbeschrieb_IV“. Um diese technische Dokumentation zu verstehen, ist es hilfreich, vorgängig den nicht-technischen Modellbeschrieb anzuschauen.

Das Finanzperspektivenmodell der IV besteht aus zwei Teilen. In einem ersten Teil werden die Daten, welche in Kapitel 2 beschrieben sind, aufbereitet. Das heisst, die Daten werden eingelesen, bei Bedarf in ein Format gemäss den tidy data Prinzipien umgewandelt, und danach in einem zentralen Ordner abgelegt.[1] Die Module hierzu sind in Kapitel 3 beschrieben. In einem zweiten Teil wird dann, basierend auf den im ersten Teil aufbereiteten Daten, das eigentliche Finanzperspektivenmodell berechnet. Die Module hierzu sind in Kapitel 4 beschrieben.

Das Programm ist so aufgebaut, dass es aus verschachtelten Modulen besteht. Im Falle des Teils zur Berechnung des Finanzperspektivenmodells (Kapitel 4) bedeutet dies, dass ein Hauptmodul zuerst auf ein Untermodul zugreift, das die aufbereiteten Daten einliest, und dann auf ein Untermodul, das die eigentlichen Berechnungen durchführt. Das Untermodul, das die Berechnungen durchführt ist wiederum aufgeteilt in ein Modul, das die Einnahmen der IV berechnet, und ein Modul, das die Ausgaben der IV berechnet. Diese Module sind wiederum aufgeteilt in Untermodulen nach Einnahmen- und Ausgabenkategorien, bis schlussendlich das Untermodul erreicht wird, das dem jeweiligen Einnahme- oder Ausgabeposten der Erfolgsrechnung respektive der Bilanz der IV-Abrechnung entspricht.[2]

1.1 Aufsetzen der Entwicklungsumgebung

Wir nehmen für die folgenden Erläuterungen an, dass der Ordner mit den Programmcodes mit *delfinverse* benennt und direkt auf dem Laufwerk C abgespeichert wird. Natürlich kann unter Anpassung des Grundpfades jeder beliebige Ordnername und Speicherort gewählt werden.

Um die Entwicklungsumgebung aufzusetzten genügt das folgende kurze Skript, das die Pfade definiert und die im Finanzperspektivenmodell verwendeten R-Packete lädt:

```
setwd("C:/delfinverse")

devtools::load_all("dinfra")
devtools::load_all("dinput")
devtools::load_all("delfin")
devtools::load_all("dmeasures")
devtools::load_all("doutput")
```

Die Packete enthalten durch das BSV entwickelte Programme für die folgenden Zwecke:

- `dinfra`: Grundprogramme, welche in allen Berechnungsschritten des Finanzperspektivenmodells immer wieder aufgerufen werden.
- `dinput`: Programme zur Aufbereitung der Input-Daten (vlg. Kapitel 3).
- `delfin`: Programme, welche den Kern des Finanzperspektivenmodells, also die Projektionen für die einzelnen Einnahmen- und Ausgabenpositionen berechnen (vgl. Kapitel 4).
- `dmeasures`: Programme zur Berechnung der Auswirkungen von Politikmassnahmen (bspw. enthält dieses Packet ein Modul zur Abschätzung der Kosten einer 13. IV-Rente).
- `doutput`: Programme zur optischen Aufbereitung des Outputs (bspw. die Finanzperspektiven-Übersichtstabellen, die auf der BSV-Website veröffentlicht werden).

[1] https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

[2] Der Aufbau der Bilanz und Erfolgsrechnung ist hier ersichtlich: https://ar.compenswiss.ch/de_CH/konten/iv

1.2 Spezifikation der zu verwendenden Daten und Modellparameter

Die Inputdaten, respektive die Pfade zu diesen, sowie die zu verwendenden Modellparameter werden nicht direkt im R-Code, sondern „extern “ in einer .csv-Datei spezifiziert. Dies dient dazu, eine möglichst grosse Flexibilität bei der Modellierung zu erhalten, und zu verhindern, dass zum Testen von alternativen Parameterspezifikationen der Modellcode angepasst werden muss.

1.2.1 Spezifikation der zu verwendenden Daten

Bei der Ausführung des Codes zum aufbereiten der Input-Daten (Kapitel 3) wird die Datei `PARAM_INPUTS.csv` aufgerufen, welche angibt, welche Rohndaten eingelesen und aufbereitet werden sollen, und unter welchem Pfad diese Rohdaten abgelegt sind. Der Zweck von `PARAM_INPUTS` ist also einerseits, dass die Speicherorte der Rohdaten in einer zentralen Datei ersichtlich sind, und andererseits, dass Änderungen im Speicherpfad der Input-Daten einfach, d.h. ohne Anpassungen am Programmcode, vorgenommen werden können. Nachfolgend ein Auszug aus `PARAM_INPUTS.csv` (Stand September 2024) als Beispiel:

key	value
path_iv_abrechnung	P:/Projekte/MASS_db/00_sv/02_iv/
file_iv_abrechnung_fin	sv_iv_fin.xlsx
sheet_iv_abrechnung_def	daten
sheet_iv_abrechnung_prov	daten_prov

Wir sehen in der Tabelle den Pfad zur IV-Abrechnung, sowie die Spezifikation in welchem Blatt des entsprechenden .xlsx die provisorische respektive die definitive IV-Abrechnung abgelegt sind.

1.2.2 Spezifikation der Modellparameter

Bei der Ausführung des Codes für die Berechnung der Finanzperspektiven (Kapitel 4) wird die Datei `PARAM_GLOBAL.csv` aufgerufen, welche angibt, mit welchen Parametern die Berechnungen des Finanzperspektivenmodells durchgeführt werden sollen. `PARAM_GLOBAL.csv` erlaubt es also, auf einen Blick nachzuvollziehen unter welchen Parameterannahmen das Finanzperspektivenmodell berechnet wird, und diese Parameterannahmen ohne Änderungen am Code anzupassen. Zusätzlich zu den Modellparametern kann in `PARAM_GLOBAL.csv` auch festgelegt werden, welche Grundlagedaten für die Berechnung der Finanzperspektiven verwendet werden sollen (bspw. welche Projektion für die Lohn- und Preisentwicklung oder welches BFS-Bevölkerungsszenario). Nachfolgend ein Auszug aus `PARAM_GLOBAL.csv` (Stand September 2024) als Beispiel:

key	value
jahr_ende	2070
MA_years	3
szenariolag	5

Wir sehen in der Tabelle in der Zeile `jahr_ende` die Angabe des Jahres, bis zu welchem das Finanzperspektivenmodell berechnet werden soll (je länger der Projektionshorizont desto länger die Rechenzeit). Die Zeile `MA_years` zeigt, dass drei Jahre für die Berechnung des gleitenden Durchschnitts bei der Invalidisierungsrate und der Mutationsrate im Rentenmodell verwendet werden sollen, und die Zeile `szenariolag` zeigt, das fünf Jahre für die Schätzung der IV-Szenarien „hoch“ und „tief“ berücksichtigt werden sollen (der Modellteil, in welchem diese Parameter verwendet werden, ist in Kapitel 4.2 beschrieben).

Bei der Ausführung des Finanzperspektivenmodells kann optional auch eine Auswahl an Politikmassnahmen spezifiziert werden, welche bei den Berechnungen berücksichtigt werden sollen (der Modellteil zu den Politikmassnahmen ist in Kapitel 4.1.11 beschrieben). Diese werden in der Datei `PARAM_MASSNAHMEN_BASE.csv` spezifiziert. Diese Datei sieht Stand September 2024 wie folgt aus:

key	value
verwendete_massnahmen	intern, ext_sv
aktivierte_massnahmen_int	iv_tabellenlohn_nv_rentiers, iv_tabellenlohn_ac_rentiers iv_furh_int_autismus
felder_ext_iv	
felder_ext_sv	btr_v_R65, btr_v_ausg, hm_R65, assb_R65
mass_f_1	iv_tabellenlohn_nv_rentiers, iv_tabellenlohn_ac_rentiers iv_furh_int_autismus
mass_f_2	group1, group2
group1	btr_v_R65, btr_v_ausg, EIN_AVS21
group2	assb_R65, hm_R65, AUSG_AVS21

Wir sehen in der Tabelle in der ersten Zeile die Angabe der Massnahmen, die bei den Berechnungen berücksichtigt werden sollen. Es werden sowohl IV-interne Massnahmen (`intern`) als auch für die IV relevante Massnahmen von anderen Sozialversicherungen (`ext_sv`) berücksichtigt. In der Zeile `aktivierte_massnahmen_int` werden die zu verwendenden IV-internen Massnahmen spezifiziert, während in der Zeile `felder_ext_sv` die Massnahmen von anderen Sozialversicherungen erwähnt sind. Bei letzterem handelt es sich hier um Auswirkungen der AHV-21 Reform, welche durch die Erhöhung des Rentenalters der Frauen für die IV Mehrkosten bewirkte. Die Zeilen `mass_f_1` bis `group2` dienen lediglich der Strukturierung der Massnahmen bei der Darstellung in den Output-Dateien.

2 Beschrieb der verwendeten Input-Daten

Dokumentation zuletzt aktualisiert am 24.10.2024

In diesem Kapitel werden die Input-Daten beschrieben, auf welchen die Berechnungen des Finanzperspektivenmodell IV beruhen.

2.1 Daten zur Wohn- und Erwerbsbevölkerung

Wir verwenden Daten zur ständigen Wohnbevölkerung nach Alter und Geschlecht aus der Statistik der Bevölkerung und der Haushalte (STATPOP)[3] des BFS. Um eine komplette Zeitreihe der historischen Entwicklungen zu erhalten ergänzen wir die STATPOP zudem mit Daten zur Synthesestatistik von Stand und Struktur der Bevölkerung (ESOPOP)[4] des BFS.}. Diese Daten erlauben uns, den IST-Bestand der Wohnbevölkerung nachzuvollziehen, und werden im Finanzperspektivenmodell unter anderem zur Abschätzung des Anteils der Wohnbevölkerung, die in einem Jahr neu eine Invalidenrente erhält, verwendet.

Unsere Projektionen basieren auf den durch das BFS in den Wohnbevölkerungsszenarien[5] sowie den Szenarien der Erwerbsbevölkerung in Vollzeitäquivalenten[6] projizierten Entwicklungen. Diese Szenarien werden alle fünf Jahre aktualisiert, wobei Stand 2024 die letzte Aktualisierung 2020 stattgefunden hat. Beide Szenarien basieren auf dem Inländer-Konzept, das heisst, dass sie die Entwicklung der ständigen Wohnbevölkerung respektive die Erwerbsbevölkerung in Vollzeitäquivalenten innerhalb der ständigen Wohnbevölkerung abdecken. Wir basieren uns im Finanzperspektivenmodell durchgehend auf das Referenzszenario für die Wohnbevölkerung respektive die Erwerbsbevölkerung in Vollzeitäquivalenten.

[3] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/espop.html
[4] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/espop.html
[5] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/szenarien.html
[6] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erwerbstaetigkeit-arbeitszeit/erwerbsbevoelkerung/kuenftige-entwicklung-erwerbsbevoelkerung.html

2.2 Daten zur Lohn- und Preisentwicklung

Wir verwenden Daten zur historischen Lohn- und Preisentwicklung des BFS. Für die Lohnentwicklung basieren wir uns auf Daten zum nominalen Schweizerischen Lohnindex (SLI) mit Basis 1939=100[7]. Für die Preisentwicklung basieren wir uns auf den Landesindex der Konsumentenpreise (LIK) mit Basis 1977=100[8], wobei wir bis 2017 den LIK Stand Dezember, respektive die jährliche Veränderung des LIK Stand Dezember, und ab 2017 das Jahresmittel des LIK, respektive die jährliche Verändungsrate des Jahresmittels des LIK, für die Berechnung der Teuerung verwenden.[9]

Unsere Projektionen für die Lohn- und Preisentwicklung gemäss SLI respektive LIK basieren auf den durch die Eidgenössische Finanzverwaltung (EFV) projizierten Eckwerte für die Finanzplanung.[10] Zusätzlich zu den publizierten Eckwerten für die Finanzplanungsperiode (aktuelles Jahr und die kommenden 4 Jahre) stellt uns die EFV Projektionen für die SLI und LIK Entwicklung für die Mittelfristperspektive, also die 5 Jahre nach den Finanzplanungsjahren, zur Verfügung. Für die Erstellung ihrer Projektionen stütz sich die ESTV einerseits auf die Prognosen der Expertengruppe Konjunkturprognose des Bundes, und andererseits auf die Mittelfristprognosen des Staatssekretariats für Wirtschaft SECO. Detaillierte Dokumentationen sind bei der EFV unter https://www.efv.admin.ch/efv/de/home/finanzberichterstattung/daten/eckwerte-finanzplanung.html, respektive auf Anfrage direkt bei der EFV, erhältlich.

2.3 Daten aus den zentralen Registern der 1. Säule

Wir verwenden die folgenden Daten aus den zentralen Registern der 1. Säule:

2.3.1 Daten aus den individuellen Konten (AHV-IK)

Die Daten der Individuelle Konten (IK) enthalten individuelle Daten zur Erwerbstätigkeit jeder in der Schweiz beitragspflichtigen Person.[11] Diese Konten werden von der Zentralen Ausgleichsstelle (ZAS) und dem Bundesamt für Sozialversicherungen (BSV) geführt und sind entscheidend für die Berechnung der individuellen Rentenansprüche. Auf den individuellen Konten wird der Erwerbsverlauf, die Art der Beschäftigung und die entsprechenden Jahreseinkommen erfasst. Aus diesen Informationen in den IK können die geleisteten Beiträge berechnet werden.

Als Input für das Finanzperspektivenmodell verwenden wir Angaben zur ausbezahlten Lohnsumme, respektive der einbezahlten Lohnbeiträge, aggregiert nach Jahr, Alter, und Geschlecht.

2.3.2 Daten aus dem Rentenregister

Aus dem Rentenregister verwenden wir Daten zu den volljährigen Beziehenden von Invalidenrenten und Hilflosenentschädigungen. Als Input für das Finanzperspektivenmodell dienen uns Angaben zur Anzahl Beziehende, der Anzahl Neubeziehenden (d.h. Personen die im Dezember eines Jahres eine Leistung beziehen, diese aber im Dezember des Vorjahres nicht bezogen haben), der Summe der im Dezember total ausbezahlten Leistungen (Renten und Hilflosenentschädigungen), sowie der Summe der im Dezember an Neubeziehende ausbezahlten Leistungen, aggregiert nach Jahr, Alter und Geschlecht.

2.3.3 Daten aus SUMEX

Das SUMEX-Register ist eine zentrale Datenbank in der Schweiz, die detaillierte Informationen zu Leistungen und Kosten im Bereich der Sozialversicherungen erfasst, insbesondere im Kontext der Invalidenversicherung (IV). Wir verwenden aus SUMEX Daten zur Hilflosenentschädigung bei Kindern sowie den Intensivpflegezuschlag, sowie Daten zu den Rechnungen für medizinische Massnahmen.

[7] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/loehne-erwerbseinkommen-arbeitskosten/lohnindex.html

[8] https://www.bfs.admin.ch/bfs/de/home/statistiken/preise/landesindex-konsumentenpreise/indexierung.assetdetail.32767557.html

[9] Die Verwendung der Basisjahre folgt den gesetztlichen Anforderungen für die Berechnung des Mischindexes für die Bestimmung der Minimalrenten.

[10] https://www.efv.admin.ch/efv/de/home/finanzberichterstattung/daten/eckwerte-finanzplanung.html

[11] https://www.ahv-iv.ch/p/1.04.d

Für die Hilflosenentschädigung bei Kindern sowie den Intensivpflegezuschlag verwenden wir Daten zur Anzahl Beziehende, der Anzahl Neubeziehenden, der Summe der im Dezember total ausbezahlten Leistungen (Hilflosenentschädigungen und Intensivpflegezuschläge), sowie der Summe der im Dezember an Neubeziehende ausbezahlten Leistungen, aggregiert nach Jahr, Alter und Geschlecht.

Für die medizinische Massnahmen verwenden wir Daten zur Anzahl Beziehenden und den Rechnungssummen, aggregiert nach Jahr und Alter.

2.4 Daten aus der IV Abrechnung

Wir verwenden die Abrechnungsdaten der IV gemäss der von Compenswiss publizierten Jahresrechnung (Bilanz und der Erfolgsrechnung der IV.[12]

2.5 Daten zu den MWST-Einnahmen

Die ESTV liefert dem BSV mehrmals im Jahr zwei Zeitreihen zu den projizierten MWST-Einnahmen pro MWST-Prozentpunkt über die kommenden 5-8 Jahren (der Projektionshorizont variiert je nach Lieferung). Die Erste Zeitreihe ist die Haupt-MWST-Projektion, welche die tatsächlich erwarteten MWST-Einnahmen pro MWST-Prozent abbildet. Im Kontext der IV Finanzperspektiven ist diese MWST-Projektion lediglich relevant, um die Einnahmen einer allfälligen IV-Zusatzfinanzierung über die MWST abzuschätzen.

Die zweite von der ESTV gelieferte Zeitreihe zu den projizierten MWST-Einnahmen beinhaltet die für die Berechnung des Bundesbeitrages an die IV relevante Entwicklung der MWST-Einnahmen. Diese Zeitreihe unterscheidet sich leicht von der Ersten, da für die MWST-Entwicklung, welche für die Berechnung des Bundesbeitrages zu verwenden ist, die folgenden spezielle gestzlichen Vorgaben gelten: 1. Es werden immer die Werte für die Steuersätze 7,6%, 2,4% und 3.6% verwendet, wobei die entsprechenden Erträge aus der Staatsrechnung ersichtlich sind. 2. Es werden Bereinigungen vorgenommen, sofern es eine nicht zweckgebundene Erhöhung gibt, oder bei einer Änderung der Bemessensgrundlage (Einführung einer neuen Steuerausnahme oder Aufhebung einer Steuerausnahme).[13]

In seltenen Fällen liefert die ESTV verschiedene Szenarien für die MWST-Projektionen, beispielsweise zuletzt 2021, um die Unsicherheit der zukünftigen Einnahmen im Zuge der Covid-Pandemie abzubilden.

2.6 Ausgabeprojektionen aus BSV-internen Umfragen

Für verschiedene Ausgabepositionen der IV, insbesondere im Bereich der individuellen Massnahmen, werden die Ausgabeprojektionen für die kommenden 5 Jahre ab dem aktuellsten Abrechnungsjahr von den BSV-internen IV-Fachbereichen erstellt. Da diese Ausgabeprojektionen ausserhalb des in Kapitel 4 erläuterten Finanzperspektivenmodells berechnet werden, werden diese Daten als externe Inpudaten behandelt und hier beschrieben. Nachfolgend wird kurz erläutert, welche Hintergrunddaten und Überlegungen bei der Erstellung der Projektionen einfliessen:

Als Grundlage für die Erstellung der Ausgabenprojektionen für die kommenden 5 Jahre werden den BSV-internen IV-Fachbereiche die folgenden Informationen zur Verfügung gestellt:

- Finanzzahlen der IV-Betriebsrechnung
- Daten zur Anzahl Leistungsbezüger pro Massnahmegruppe
- Anzahl Rechnungen pro Jahr
- Durchschnittliche Kosten pro Massnahmegruppe und pro IV-Stelle
- Grafische Darstellung der jährlichen Entwicklung Ist gegenüber bisheriger Budgetplanung

In einem nächsten Schritt erstellen die Fachbereiche eine Schätzung, wobei die zugrunde-liegenden Annahmen mit einem begleitenden Kommentar dokumentiert werden müssen. Meistens bilden die Planungsgrundlage

[12] https://ar.compenswiss.ch/de_CH/konten/iv

[13] Die für die Berechnung des Bundesbeitrages zu verwendende MWST-Grundlage folgt Art. 78 IVG, Absätze 1-3. Ein detaillierter Beschrieb der verwendenden MWST-Grundlage, und der Bereinigungen, findet sich im Brief vom 21.3.2014 von der EFV an das BFS, die ESTV, sowie die Zentrale Ausgleichsstelle. Dieser ist auf Anfrage beim Bereich MATH des BSV erhältlich.

die Betrachtung der bisherigen Entwicklung der Anzahl Beziehenden respektive Leistungen/Rechnungen und der durchschnittlichen Ausgaben pro Beziehende respektive Leistung/Rechnung. Im Weiteren werden von den Fachbereichen exogene Faktoren (Preisteuerungen, neue Tarifverträge/-Preise, Anmeldeentwicklung) in der Planung berücksichtigt.

3 Module zur Aufbereitung der Input-Daten

In diesem Kapitel werden die Module beschrieben, mithilfe welcher die Input-Daten für das Finanzperspektivenmodell IV eingelesen und aufbereitet werden. Die Datenaufbereitung für die Finanzperspektivenmodelle sämtlicher durch das BSV modellierten Sozialversicherungen (AHV, IV, EO, EL) erfolgt über ein gemeinsames Modul `prepare_input.R`. Dies ist dadurch begründet, dass vielfach die gleichen Input-Daten von allen Sozialversicherungen genutzt werden (bspw. Bevölkerungsstatistiken und -szenarien des BFS).[14]

3.1 Modul prepare_input.R

Dokumentation zuletzt aktualisiert am 16.09.2024

Das Hauptmodul zur Aufbereitung der Input-Daten für das Finanzperspektivenmodell der IV wird wie folgt aufgerufen:

```
path_input = "C:/delfinverse/data/PARAM_INPUTS.csv"
prepare_input(path = path_input)
```

`path_input` ist der Pfad zur `PARAM_INPUTS.csv` Datei, welche für die Berechnungen verwendet werden soll. `PARAM_INPUTS.csv` enthält Dateipfade zu den zu verwendenden Rohdaten (vgl. Kapitel 1.2.1). Bevor der obige Code ausgeführt werden kann, muss natürlich sichergestellt werden, dass der Ordner `.../data` existiert, und die Datei `PARAM_INPUTS.csv` enthält. Zudem darf der Ordner `.../data` ausser `PARAM_INPUTS.csv` vor dem Ausführen von `prepare_input` keine anderen Ordner oder Dateien enthalten.

Das Modul `prepare_input` enthält die folgenden Elemente:

```
function(path, path_out = file.path(dirname(path)),
                          overwrite = FALSE) {
```

Das Modul nimmt den vorangehend spezifizierten Pfad `path` entgegen, und spezifiziert den Ordner auf welchen `path` verweist als `path_out`, also den Ordner, in welchen die aufbereiteten Daten am Ende des Moduls ausgelesen werden sollen. Zudem wird die Variable `overwrite` auf `FALSE` gesetzt, um zu verhindern, dass falls der Ordner `.../data` bereits aufbereitete Daten enthält, diese überschrieben werden.

Danach werden die Input-Parameter aus `PARAM_INPUTS.csv` eingelesen, und im Dataframe `PARAM_INPUTS` abgelegt:

```
# Parameters
# ------------------------------------------------------------------

PARAM_INPUTS <- read_param(path)
```

Als nächstes werden die Pfade spezifiziert, unter welchen die aufbereiteten Input-Daten später im Modul abgelegt werden sollen:

```
# Paths
# ------------------------------------------------------------------------

inp_path_ahv_go <- file.path(path_out, "ahv", "go")
```

[14] In diesem Kapitel werden nur Untermodule von `prepare_input.R` beschrieben, welche Daten einlesen, die auch im Finanzperspektivenmodell IV verwendet werden. Die Module, welche Daten einlesen, die ausschliesslich von anderen Sozialversicherungen als der IV verwendet werden, werden ausgeklammert.

```
inp_path_ahv_massnahmen <- file.path(path_out, "ahv", "massnahmen")

inp_path_iv_go <- file.path(path_out, "iv", "go")
inp_path_iv_massnahmen <- file.path(path_out, "iv", "massnahmen")

inp_path_eo_go <- file.path(path_out, "eo", "go")
inp_path_eo_massnahmen <- file.path(path_out, "eo", "massnahmen")

inp_path_go_el <- file.path(path_out, "el", "go")
inp_path_massnahmen_el <- file.path(path_out, "el", "massnahmen")

inp_path_rententab <- file.path(path_out, "rententab")

inp_path_beitragstab <- file.path(path_out, "beitragstab")

inp_path_eotab <- file.path(path_out, "eotab")
```

Es wird später für jede Sozialversicherung ein separater Ordner für die aufbereiteten Input-Daten erstellt. `inp_path_ahv_go` enthält beispielsweise den Pfad zum Ordner, in welchem Input-Daten für die Berechnungen der AHV-Finanzperspektiven später abgespeichert werden sollen, und `inp_path_ahv_massnahmen` den Pfad zum Ordner, in welchem die Input-Daten für die Berechnung der Politikmassnahmen für die AHV abgespeichert werden sollen. Es gibt Input-Daten, welche für mehrere Finanzperspektivenmodelle verwendet werden, beispielsweise die Bevölkerungszahlen und -szenarien, oder die Proejektionen zur Lohn- und Preisentwicklung. Um Duplikate zu vermeiden gilt hier die Konvention, dass diese Input-Daten ausschliesslich im Ordner `inp_path_ahv_go` abegelegt werden, was widerum bedeutet, dass auch das Modell zur Berechnung der IV-Finanzperspektiven später auf einzelne Dateien im Ordner `inp_path_ahv_go` zugreiffen wird.

Der Nachfolgende Code-Block stellt sicher, dass die Verschiedenen Ordner mit den Input-Daten nicht schon im Ordner `.../data` enthalten sind. Es handelt sich hierbei um ein Sicherheitscheck, der verhindern soll, dass bestehende Input-Daten versehentlich überschrieben werden:

```
ensure_path <- function(path) {
    # do not allow overwriting if overwrite == FALSE
    if (!overwrite && file.exists(path))
        stop(path, "already exists")
    if (!file.exists(path)) {
        dir.create(path, recursive = TRUE)
    }
    file.remove(list.files(path, full.names = TRUE))
}

ensure_path(inp_path_ahv_go)
ensure_path(inp_path_ahv_massnahmen)

ensure_path(inp_path_iv_go)
ensure_path(inp_path_iv_massnahmen)

ensure_path(inp_path_eo_go)
ensure_path(inp_path_eo_massnahmen)

ensure_path(inp_path_go_el)
ensure_path(inp_path_massnahmen_el)

ensure_path(inp_path_rententab)
```

```
ensure_path(inp_path_beitragstab)

ensure_path(inp_path_eotab)
```

Als nächstes werden die verschiedenen Rohdatenfiles eingelesen und aufbereitet. Die Funktionsweise dieser repetitiven Code-Elemente wird nachfolgend am Beispiel der Input-Daten zur ständigen Wohnbevölkerung erläutert:

```
# Input Data
# ------------------------------------------------------------------

# Input Ständige Wohnbevölkerung / Population suisse
# résidante permanente

tl_input_ofs_pop_res <- mod_input_ofs_dwh_pop_res(PARAM_INPUTS = PARAM_INPUTS)

BEV_POP <- tl_input_ofs_pop_res$BEV_POP

POP_SCENARIO_BEV <- tl_input_ofs_pop_res$POP_SCENARIO_BEV

POP_SCENARIO_EPT <- tl_input_ofs_pop_res$POP_SCENARIO_EPT

POP_SCENARIO_EPT_FULL <- tl_input_ofs_pop_res$POP_SCENARIO_EPT_FULL
```

Es wird das Modul `mod_input_ofs_dwh_pop_res` (vgl. Kapitel 3.2) aufgerufen, welches die Bevölkerungsdaten des BFS einliest und aufbereitet. Der Output des Moduls wird in der tidy list `tl_input_ofs_pop_res` abgespeichert. Danach werden die einzelnen Teile der Bevölkerungsdaten des BFS in separaten Dataframes abgelegt, beispielsweise werden die Daten zur ständigen Wohnbevölkerung in das Dataframe `BEV_POP` abgelegt, und die Wohnbevölkerungsszenarien in das Dataframe `POP_SCENARIO_BEV`.

Am Ende des Moduls `prepare_input.R` werden die eingelesenen und aufbereiteten Daten in den oben spezifizierten Ordnern abgelegt. Am Beispiel der Input-Daten der IV sieht der betreffende Codeteil wie folgt aus:

```
  # Geltende Ordnung IV
  tidy list_write(iv_go, inp_path_iv_go)

  # Massnahmen IV
  tidy list_write(iv_massnahmen, inp_path_iv_massnahmen)
```

Die Funktion `tidy list_write` nimmt die in der tidy list `iv_go` spezifizierten Dataframes, und schreibt diese im .csv-Format in den Ordner `inp_path_iv_go` (siehe oben). Dasselbe geschieht mit den in `iv_massnahmen` spezifizierten Dataframes.

3.2 Modul mod_input_ofs_dwh_pop_res.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_ofs_dwh_pop_res.R` liest die Bevölkerungsdaten des BFS ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_ofs_pop_res <- mod_input_ofs_dwh_pop_res(PARAM_INPUTS = PARAM_INPUTS)
```

Als erstes werden in`mod_input_ofs_dwh_pop_res.R` die verschiedenen Rohdateien mit den Bevölkerungsdaten eingelesen:

```
# Function to give the name to the data loaded
loadRData <- function(fileName) {
    # loads an RData file, and returns it
    load(fileName)
    get(ls()[ls() != "fileName"])
}

# Load ESPOP
ESPOP <- loadRData(file.path(PARAM_INPUTS$path_ofs_dwh, PARAM_INPUTS$name_espop))
# Load STATPOP
STATPOP <- loadRData(file.path(PARAM_INPUTS$path_ofs_dwh, PARAM_INPUTS$name_statpop))

# Load Scenarios population
SCENARIO_POP <- loadRData(file.path(PARAM_INPUTS$path_ofs_dwh,
    PARAM_INPUTS$name_scenario_pop))

# Load Scenarios ept
SCENARIO_EPT <- loadRData(file.path(PARAM_INPUTS$path_ofs_dwh,
    PARAM_INPUTS$name_scenario_ept))

# Old scenarios
lazyLoad(file.path(PARAM_INPUTS$path_SASData, PARAM_INPUTS$path_bevdata,
    PARAM_INPUTS$file_bevdata))
lazyLoad(file.path(PARAM_INPUTS$path_SASData, PARAM_INPUTS$path_bevdata,
    PARAM_INPUTS$file_eptdata))
```

Die Funktion `loadRData` stellt sicher, dass die Daten im R-Format korrekt eingelesen werden. Darauffolgend werden nacheinander die Daten zur Synthesestatistik von Stand und Struktur der Bevölkerung (ESOPOP)[15], zur Statistik der Bevölkerung und der Haushalte (STATPOP)[16], die Bevölkerungsszenarien (SCENARIO_POP)[17], sowie die Erwerbsbevölkerungsszenarien (SCENARIO_EPT)[18] eingelesen. Die letzten zwei Codeblöcke dienen dazu, vergangene (Erwerbs-)Bevölkerungsszenarien einzulesen (bspw. zur späteren Verwendung in einem Backtesting).

Als nächstes werden die Bevölkerungsdaten aus ESPOP, auf welchen die Schweizerische Bevölkerungsstatistik bis 2009 basierte, und die Bevölkerungsdaten aus STATPOP, welche ab 2010 verfügbar sind, zusammengefügt. Hierbei werden alle Alter über 98 Jahre in der Altersgruppe 99 zusammengefasst, und die Schweizerische Wohnbevölkerung wird nach den Zellen Jahr, Geschlecht (Mann oder Frau), Nationalität (Schweizer oder Ausländer), und Alter aggregiert:

```
# Import Wohnbevoelkerung gemaess Beobachtungen
# ----------------------------#

# ESPOP: 1971 - 2009

BEV_ESPOP <- ESPOP %>%
    dplyr::select(jahr, sex, nat, alt, bevendejahr) %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(jahr, sex, nat, alt) %>%
    summarize(bevendejahr = sum(bevendejahr)) %>%
    filter(jahr < 2010) %>%
```

[15] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/espop.html
[16] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/statpop.html
[17] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/szenarien.html
[18] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erwerbstaetigkeit-arbeitszeit/erwerbsbevoelkerung/kuenftige-entwicklung-erwerbsbevoelkerung.html

```
    ungroup()

# STATPOP Definitiv (bis PARAM_INPUTS$lastyear_statpop)

BEV_STATPOP <- STATPOP %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(jahr, sex, nat, alt) %>%
    summarize(bevendejahr = sum(bevendejahr)) %>%
    ungroup()

# Staendige Wohnbevoelkerung: definitive
BEV_POP <- bind_rows(BEV_ESPOP, filter(BEV_STATPOP, jahr <=
↪  PARAM_INPUTS$lastyear_statpop),
    .id = "source") %>%
    mutate(source = case_when(source == "1" ~ "ESPOP", TRUE ~
        paste0("STATPOP", PARAM_INPUTS$lastyear_statpop)))
```

Der nachfolgende Code dient lediglich dazu, die Formatierung der Spalten Nationalität und Geschlecht in den alten Bevölkerungsszenarien an die Formatierung in den neuen Szenarien anzugleichen :

```
# Convert factor columns into character
bev_A_00_2000[] <- lapply(bev_A_00_2000, function(x) {
    if (is.factor(x)) {
        as.character(x)
    } else {
        x
    }
})
bev_A_00_2005[] <- lapply(bev_A_00_2005, function(x) {
    if (is.factor(x)) {
        as.character(x)
    } else {
        x
    }
})
bev_A_00_2010[] <- lapply(bev_A_00_2010, function(x) {
    if (is.factor(x)) {
        as.character(x)
    } else {
        x
    }
})
bev_A_17_2010[] <- lapply(bev_A_17_2010, function(x) {
    if (is.factor(x)) {
        as.character(x)
    } else {
        x
    }
})
```

Als nächstes werden die alten Bevölkerungsszenarien in einem Dataframe zusammengefasst, und es wird sichergestellt, dass die Formatierung mit der Formatierung der neuen Bevölkerungsszenarien übereinstimmt:

```
# Bind all old scenarios
POP_SCENARIO_BEV <- bind_rows(bev_A_00_2000, bev_A_00_2005, bev_A_00_2010,
    bev_A_17_2010, .id = "scenario") %>%
    mutate(scenario = case_when(scenario == "1" ~ "A_00_2000",
        scenario == "2" ~ "A_00_2005", scenario == "3" ~ "A_00_2010",
        scenario == "4" ~ "A_17_2010")) %>%
    mutate(sex = recode(sex, Mann = "m", Frau = "f"), nat = recode(nat,
        CH = "ch", AU = "au")) %>%
    rename(alt = alter) %>%
    mutate(alt = as.integer(alt)) %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(scenario, jahr, sex, nat, alt) %>%
    summarize(bevendejahr = sum(bevendejahr)) %>%
    ungroup() %>%
    # JOIN the new version of the scenarios
bind_rows(SCENARIO_POP %>%
    mutate(scenario = case_when(grepl("-", scenario) ~ gsub("-",
        "_", scenario), TRUE ~ scenario)) %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(scenario, jahr, sex, nat, alt) %>%
    summarise(bevendejahr = sum(bevendejahr)))
```

Als nächstes werden die Erwerbsbevölkerungsszenarien aufbereitet. Hierbei werden wiederum die gleichen Zellen gebildet wie weiter oben bei den Wohnbevölkerungsszenarien, und die Alter 98 werden wiederum im Alter 99 zusammengefasst. Das Dataframe `POP_SCENARIO_EPT` enthält sämtliche Erwerbsbevölkerungsszenarien, insbesondere auch das Szenario 2010 (`ept_A_17_2010`). Da für `ept_A_17_2010` lediglich die ein Erwerbsbevölkerungsszenario in Vollzeitäquivalenten (fr. équivalent temps plein, EPT) verfügbar ist, enthält `POP_SCENARIO_EPT` lediglich diese Variable, welche in `POP_SCENARIO_EPT` mit `bevendejahr` benannt ist. Demgegenüber enthält `POP_SCENARIO_EPT_FULL` lediglich die Erwerbsbevölkerungsszenarien ab 2015, wobei hier sowohl die Erwerbsbevölkerung in Anzahl Personen als auch in Vollzeitäquivalenten verfügbar ist. Diese Variablen werden in `POP_SCENARIO_EPT_FULL` `erwbev` respektive `erwbevept` genannt:

```
POP_SCENARIO_EPT <- bind_rows(ept_A_17_2010 %>%
    dplyr::select(jahr, sex, nat, alt = alter, bevendejahr = erwbevept) %>%
    mutate(scenario = "A_17_2010", sex = recode(sex, Mann = "m",
        Frau = "f"), nat = recode(nat, CH = "ch", AU = "au")),
    dplyr::select(left_join(id, SCENARIO_EPT %>%
        mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
        group_by(jahr, sex, nat, alt, scenario) %>%
        summarize(erwbevept = sum(erwbevept)) %>%
        mutate(scenario = case_when(grepl("-", scenario) ~ gsub("-",
            "_", scenario), TRUE ~ scenario)), by = c("jahr",
        "sex", "nat", "alt", "scenario")), jahr, sex, nat, alt,
        bevendejahr = erwbevept, scenario)) %>%
    mutate(bevendejahr = ifelse(is.na(bevendejahr), 0, bevendejahr))

# POPULATION ACTIVE EN PERSONNES (erwbev) ET EN ETP
# (erwbevept)
POP_SCENARIO_EPT_FULL <- dplyr::select(left_join(id, SCENARIO_EPT %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(jahr, sex, nat, alt, scenario) %>%
    summarize(erwbevept = sum(erwbevept), erwbev = sum(erwbev)) %>%
    mutate(scenario = case_when(grepl("-", scenario) ~ gsub("-",
        "_", scenario), TRUE ~ scenario)), by = c("jahr", "sex",
```

```
    "nat", "alt", "scenario")), jahr, sex, nat, alt, erwbev = erwbev,
    erwbevept = erwbevept, scenario) %>%
    mutate(erwbev = ifelse(is.na(erwbev), 0, erwbev), erwbevept =
    ↪  ifelse(is.na(erwbevept),
        0, erwbevept))
```

Zum Abschluss werden die eingelesenen Daten wie folgt an `prepare_input.R` zurückgegeben:

```
# Return
# ------------------------------------------------------------------------#

mod_return(BEV_POP, POP_SCENARIO_BEV, POP_SCENARIO_EPT, POP_SCENARIO_EPT_FULL)
```

3.3 Modul mod_input_indices.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_indices.R` liest die Indizes Lohnindex (gemäss nominalem Schweizerischen Lohnindex (SLI)), Preisindex Dezember, Jahresmittel Preisindex und die jährliche Veränderungsrate dieser Indizes, sowie den Strukturfaktors ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_indices <- mod_input_indices(PARAM_INPUTS = PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die in den ersten 8 Spalten des betreffenden Excel-Sheets enthaltenen Indizes und Veränderungsraten eingelesen, und danach an `prepare_input.R` zurückgegeben werden:

```
# --- Read file with indices
# ------------------------------------------------

INDICES <- read_excel(paste0(PARAM_INPUTS$path_indices, PARAM_INPUTS$file_indices),
    sheet = PARAM_INPUTS$sheet_indices, range = readxl::cell_cols(1:8))

# --- Return tidy df with indices
# ------------------------------------------

mod_return(INDICES)
```

3.4 Modul mod_input_eckwerte.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_eckwerte.R` liest die von der ESTV gelieferten Projektionen zur Entwicklung der Löhne und Preise ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_eckwerte <- mod_input_eckwerte(PARAM_INPUTS = PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Eckwerte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
# --- Read file with eckwerte
# ------------------------------------------------

ECKWERTE <- read_excel(paste0(PARAM_INPUTS$path_eckwerte, PARAM_INPUTS$file_eckwerte),
    sheet = PARAM_INPUTS$sheet_eckwerte, range = readxl::cell_cols(1:12))
```

```
# --- Return tidy df with eckwerte
# ----------------------------------------

mod_return(ECKWERTE)
```

3.5 Modul mod_input_minimalrente.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_minimalrente.R` liest die historische Zeitreihe zur Höhe der Minimalrente, deren Wachstumsrate, sowie zum Wert des Rentenindexes ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_minimalrente <- mod_input_minimalrente(PARAM_INPUTS = PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
# Einlesen der Parameter
# ----------------------------------------------------#

sheet_minimalrente <- PARAM_INPUTS$sheet_minimalrente

# Einlesen der Minimalrenten
# ----------------------------------------------#

MINIMALRENTE <- read_excel(paste0(PARAM_INPUTS$path_minimalrente,
    PARAM_INPUTS$file_minimalrente), sheet = PARAM_INPUTS$sheet_minimalrente)

colnames(MINIMALRENTE) <- c("jahr", "minimalrente", "dminimalrente",
    "rentenindex")

# Return tidy list mit Minimalrenten
# ----------------------------------------#

mod_return(MINIMALRENTE)
```

3.6 Modul mod_input_estv.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_estv.R` liest die MWST-Projektionen der ESTV ein. Diese ist im Finanzperspektivenmodell der IV relevant für den Fall, dass die Auswirkungen einer MWST-Zusatzfinanzierung abgeschätzt werden sollten. Es handelt sich hier nicht um die MWST-Projektion der ESTV für die Berechnung des Bundesbeitrages, welche nach eigenen gesetzlichen Vorgaben berechnet wird, und welche mit dem Code in Kapitel 3.12 importiert wird. `mod_input_estv.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_estv <- mod_input_estv(PARAM_INPUTS = PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
ESTV <- read_excel(paste0(PARAM_INPUTS$path_estv, PARAM_INPUTS$file_estv),
    sheet = PARAM_INPUTS$sheet_estv, range = readxl::cell_cols(1:8))

# --- Output
```

```
# ------------------------------------------------------------------

mod_return(ESTV)
```

3.7 Modul mod_input_ikregister.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_estv.R` liest die Daten der individuellen Konten der ZAS (IK) ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_ikregister <- mod_input_ikregister(PARAM_INPUTS = PARAM_INPUTS)
```

Als Erstes wird die in `PARAM_INPUTS` spezifizierte Zeitspanne eingelesen, über welche die IK-Daten verfügbar sind:

```
# --- Loop Jahre 1997 bis zum mit jahr_ik gewuenschten Jahr
# ------------------#

all_years <- PARAM_INPUTS$jahr_beginn:PARAM_INPUTS$jahr_ik
```

Als nächstes werden die IK-Daten eingelesen:

```
# --- Eingebettete Funktion, welche 1 Jahr IK Daten
# einliest -----------------#

read_one_year <- function(l_j) {
    file_base <- paste0(PARAM_INPUTS$path_to_structure, "/ik_",
        l_j, "_stat_", PARAM_INPUTS$aggregation)
    if (PARAM_INPUTS$file_source == "sas") {
        z <- sas7bdat::read.sas7bdat(paste0(file_base, "agg.sas7bdat"))
    } else {
        z <- read.csv(paste0(file_base, "agg.csv"), header = TRUE,
            sep = ";", stringsAsFactors = FALSE)
    }
    as_tibble(z) %>%
        mutate(jahr = as.integer(l_j))
}

# Einlesen IK Daten
# ------------------------------------------------------------#

BEITRAG_IK <- bind_rows(lapply(all_years, read_one_year))
```

Hierfür wird zuerst die Funktion `read_one_year`, welche für ein gegebenes Jahr entweder die SAS-Datei oder CSV-Datei mit den IK-Daten dieses Jahres einliest (je nachdem wie `PARAM_INPUTS$file_source` spezifiziert ist). Danach wird die Funktion für jedes Jahr in `all_years` ausgeführt, und die Daten werden in `BEITRAG_IK` abgelegt.

Als nächstes wird die Kodierung und Benennung einiger Variablen so angepasst, dass sie mit den anderen verwendeten Daten im Finanzperspektivenmodell (bspw. den Bevölkerungsdaten und -szenarien) konsistent ist:

```
# --- Recodierung einzelner Variabler
# ------------------------------------#
```

```
IK_TMP <- BEITRAG_IK %>%
    mutate(sex = recode(csex, `1` = "m", `2` = "f"), nat = recode(nation,
        `1` = "ch", `2` = "au")) %>%
    dplyr::select(-csex, -nation) %>%
    rename(alt = alterP, personen = Anzahl, personen_gew = anzahlGEW,
        einkommen_ik = mrevsom, beitrag_ahvzas = mcotsomZAS,
        beitrag_ahvbsv = AHVCOT_BSV, beitrag_ivbsv = IVCOT_BSV,
        beitrag_eobsv = EOCOT_BSV)

# --- Return tidy df mit IK Daten
# ----------------------------------------#

IK <- IK_TMP %>%
    pivot_longer(c(beitrag_ahvbsv, beitrag_ivbsv, beitrag_eobsv),
        names_to = "vz", values_to = "beitrag_bsv") %>%
    mutate(vz = recode(vz, beitrag_ahvbsv = "ahv", beitrag_ivbsv = "iv",
        beitrag_eobsv = "eo")) %>%
    arrange(match(vz, c("ahv", "iv", "eo")))
```

Zum Abschluss werden die eingelesenen Daten wie folgt an `prepare_input.R` zurückgegeben:

```
mod_return(IK)
```

3.8 Modul mod_input_iv_abrechnung.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_iv_abrechnung.R` liest die IV-Abrechnung ein.[19] Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_input_iv_abrechnung <- mod_input_iv_abrechnung(PARAM_INPUTS = PARAM_INPUTS)
```

Als Erstes werden die Daten aus der monatlichen IV-Abrechnung eingelesen und in ein geeignetes Format gebracht:

```
# Monatsdaten ab dem mass_db General open rectangle Upper
# left = A11, everything else unspecified

ABRECHNUNG_M <- read_excel(path = paste0(PARAM_INPUTS$path_iv_abrechnung,
    PARAM_INPUTS$file_iv_abrechnung_m), sheet = PARAM_INPUTS$sheet_iv_abrechnung_def,
    range = readxl::cell_limits(c(11, 1), c(NA, NA)))

# Identify empty lines column 1 : jahr column 2 : monat

ABRECHNUNG_M <- ABRECHNUNG_M %>%
    mutate(id = apply(.[, -c(1, 2)], MARGIN = 1, FUN = sum, na.rm = T)) %>%
    mutate(id = if_else(id == 0, 0, 1))
```

Als nächstes wird sichergestellt, dass nur Abrechnungsdaten von Jahren, für welche sämtliche Monatsdaten vorliegen, verwendet werden:

```
# Identify full years

id_full_years <- ABRECHNUNG_M %>%
```

[19] https://ar.compenswiss.ch/de_CH/konten/iv

```
    dplyr::select(jahr, monat, id) %>%
    group_by(jahr) %>%
    summarise(id_y = sum(id)) %>%
    ungroup()

ABRECHNUNG_M <- full_join(ABRECHNUNG_M, id_full_years, by = "jahr")

ABRECHNUNG_M_DEF <- ABRECHNUNG_M %>%
    filter(id_y == 12) %>%
    dplyr::select(-id, -id_y)
```

In einem nächsten Schritt werden die Provisorischen Dezemberdaten eingelesen. Diese werden für die Erstellung der provisorischen IV-Abrechnung verwendet. Zu beachten ist, dass das Blatt `PARAM_INPUTS$sheet_iv_abrechnung_prov`, welches im ersten Codeblock eingelesen wird, lediglich Abrechnungsdaten für den Monat Dezember enthält:

```
ABRECHNUNG_M_PROV <- read_excel(paste0(PARAM_INPUTS$path_iv_abrechnung,
    PARAM_INPUTS$file_iv_abrechnung_m), sheet = PARAM_INPUTS$sheet_iv_abrechnung_prov,
    range = readxl::cell_limits(c(11, 1), c(NA, NA)))

year_prov <- unique(ABRECHNUNG_M_PROV$jahr)

ABRECHNUNG_M_OHNE_DEZ <- ABRECHNUNG_M %>%
    filter(id_y >= 11) %>%
    filter((!(jahr %in% year_prov & monat == 12))) %>%
    dplyr::select(-id, -id_y)

ABRECHNUNG_M_PROV <- bind_rows(ABRECHNUNG_M_OHNE_DEZ, ABRECHNUNG_M_PROV) %>%
    arrange(jahr, monat)

ABRECHNUNG_M_ALL <- bind_rows(ABRECHNUNG_M_DEF, ABRECHNUNG_M_PROV,
    .id = "id")
```

Es werden also zuerst die provisorischen Dezember-Daten eingelesen und danach mit den definitiven Monatsdaten ohne Dezemberdaten im Dataframe `ABRECHNUNG_M_PROV` kombiniert. Zum Schluss werden die provisorischen Monatsdaten `ABRECHNUNG_M_PROV` mit den definitiven Monatsdaten `ABRECHNUNG_M_DEF` im Dataframe `ABRECHNUNG_M_ALL` kombiniert, wobei die Variable `id` angibt, ob es sich um die definitiven Daten (`id==1`) oder die provisorischen Daten (`id==2`) handelt.

Als nächstes werden die Daten von Millionen CHF in CHF umgerechnet, und nach Jahr aggregiert:

```
ABRECHNUNG_M_ALL <- ABRECHNUNG_M_ALL %>%
    mutate_at(.vars = colnames(ABRECHNUNG_M_ALL)[!(colnames(ABRECHNUNG_M_ALL) %in%
        c("id", "jahr", "monat"))], list(~. * 1e+06)) %>%
    mutate(jahr = as.integer(jahr), monat = as.integer(monat))

# --- Variables specific to iv
# ------------------------------------------------

ABRECHNUNG_M_ALL <- ABRECHNUNG_M_ALL

#-------------------------------------------------------------------------------

# --- Jahresdaten = Summe Monatsdaten
# ----------------------------------------
```

```
ABRECHNUNG <- ABRECHNUNG_M_ALL %>%
    dplyr::select(-monat) %>%
    mutate_all(list(~as.numeric(.))) %>%
    mutate_all(list(~if_else(is.na(.), 0, .))) %>%
    group_by(id, jahr) %>%
    summarise_all(sum) %>%
    ungroup()
```

Als nächstes werden die provisorischen und definitven Daten der IV-Jahresabrechnung eingelesen. Aus diesen Daten wird ausschliesslich der Stand der Schulden, des IV-Fonds, sowie der flüssigen Mittel verwendet. Für die restlichen Positionen der IV-Abrechnung werden die aus der Summe der monatlichen Abrechnungen errechneten Jahreswerte verwendet. Wir verwenden die monatlichen Abrechnung anstatt der jährlichen, da diese eine detailliertere Gruppierung der Ausgabepositionen enthalten:

```
# Provisorische Jahresdaten ab dem mass_db General open
# rectangle Upper left = A11, everything else unspecified

ABRECHNUNG_Y_PROV <- read_excel(paste0(PARAM_INPUTS$path_iv_abrechnung,
    PARAM_INPUTS$file_iv_abrechnung_fin), sheet = PARAM_INPUTS$sheet_iv_abrechnung_prov,
    range = readxl::cell_limits(c(11, 1), c(NA, NA)))

year_prov <- unique(ABRECHNUNG_Y_PROV$jahr)

ABRECHNUNG_Y_OHNE_PROV <- ABRECHNUNG_Y %>%
    filter((!(jahr %in% year_prov)))

ABRECHNUNG_Y_PROV <- bind_rows(ABRECHNUNG_Y_OHNE_PROV, ABRECHNUNG_Y_PROV)

ABRECHNUNG_Y_ALL <- bind_rows(ABRECHNUNG_Y, ABRECHNUNG_Y_PROV,
    .id = "id")

# Beiträge in Millionen Franken

ABRECHNUNG_Y_ALL <- ABRECHNUNG_Y_ALL %>%
    mutate_at(.vars = colnames(ABRECHNUNG_Y_ALL)[!(colnames(ABRECHNUNG_Y_ALL) %in%
        c("id", "jahr"))], list(~. * 1e+06)) %>%
    mutate(jahr = as.integer(jahr))

# --- Select subset of variables from yearly data
# ----------------------------

ABR_Y_SEL <- ABRECHNUNG_Y_ALL %>%
    dplyr::select(id, jahr, kap, fonds, fl_mtl) %>%
    filter(min(ABRECHNUNG_M_ALL$jahr) <= jahr & jahr <= max(ABRECHNUNG_M_ALL$jahr))

ABR_Y_SEL <- ABR_Y_SEL %>%
    mutate(id = as.numeric(id))

ABRECHNUNG <- left_join(ABRECHNUNG, ABR_Y_SEL, by = c("id", "jahr"))
```

Zum Abschluss werden die Abrechnungsdaten in ein Dataframe für die provisorischen Abrechnungssdaten und in eines für die definitiven Abrechnungsdaten aufgeteilt, und an `prepare_input.R` zurückgegeben:

```
# Definitive IV Abrechnungen

IV_ABRECHNUNG_DEF <- ABRECHNUNG %>%
    filter(id == 1) %>%
    dplyr::select(-id)

# Provisorische IV Abrechnungen Bis 2016: definitive
# Jahresdaten Ab 2017: provisorische Jahresdaten

IV_ABRECHNUNG_PROV <- ABRECHNUNG %>%
    filter(id == 2) %>%
    dplyr::select(-id)

# --- Output
# ------------------------------------------------------------------

mod_return(IV_ABRECHNUNG_DEF, IV_ABRECHNUNG_PROV)
```

3.9 Modul mod_input_zins_scen.R

Dokumentation zuletzt aktualisiert am 26.09.2024

Das Modul `mod_input_zins_scen.R` liest die Szenarien für die Realzinse auf die Vermögen und Schulden der IV ein (vgl. Kapitel 4.30). `mod_input_zins_scen.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
ZINS_SCEN <- mod_input_zins_scen(PARAM_INPUTS = PARAM_INPUTS)$ZINS_SCEN
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
tfile <- paste0(PARAM_INPUTS$path_zins_scen, PARAM_INPUTS$file_zins_scen)

ZINS_SCEN <- read_excel(tfile, sheet = PARAM_INPUTS$sheet_zins_scen,
    skip = 10)

mod_return(ZINS_SCEN)
```

3.10 Modul mod_input_ivrenten.R

Dokumentation zuletzt aktualisiert am 26.09.2024

Das Modul `mod_input_ivrenten.R` liest die Daten zu den IV-Renten und den Hilflosenentschädigungen ein. `mod_input_ivrenten.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_ivrenten <- mod_input_ivrenten(PARAM_INPUTS)
```

In einem ersten Schritt werden die Daten zu den Flüssen und Beständen bei den IV-Renten eingelesen:

```
#------ Lesen Inhalt Excel - sheet ------------------------------------#
IV_RENTEN_NEW <- read_excel(path = paste0(PARAM_INPUTS$path_iv_renten_new,
    PARAM_INPUTS$file_iv_renten_new), sheet = PARAM_INPUTS$sheet_iv_renten_new,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
) %>%
    select(-pop, -cinf_grp_bez, -cinf_grp, -sex_bez, -wohnort_ch,
        -wohnort_ch_bez) %>%
```

```
    rename(alt = alter) %>%
    mutate_if(is.numeric, ~coalesce(., 0))
```

Der nächste Schritt korrigiert einen Fehler in der Variablenbennenung bei der Datenlieferung 2024. Diese Korrektur wurde bewusst so implementiert, dass falls dieser Fehler in der Variablenbennennung in der Datenlieferung 2025 korrigiert wird, der Code angehalten wird und ein Fehler ausgegeben, und falls der Fehler 2025 nicht schon in der Datenlieferung korrigiert wird, dieser an dieser Stelle korrigiert wird:[20]

```
# Dieser Block wird implementiert da in der derzeitigen
# Version der Rohdaten mpr_nr fälschlicherweise als
# mpr_hr_nr klassifiziert ist. Falls dies korrigiert wurde
# kann der folgende Block gelöscht werden.
if ("mpr_hr_nr" %in% colnames(IV_RENTEN_NEW)) {
    stop("WARNUNG: IV_RENTEN_NEW kann nicht generiert werden, da mpr_hr_nr bereits
    ↪  existiert. Code in mod_input_ivrenten anpassen.")
} else {
    IV_RENTEN_NEW <- IV_RENTEN_NEW %>%
        mutate(mpr_hr_nr = mpr_nr) %>%
        mutate(mpr_nr = mpr_hr_nr + mpr_kr_nr)
}
```

Als nächstes wird die Variable für das Geschlecht gemäss dem Finanzperspektivenmodell umcodiert, und es werden Variablen für den gewichteten Rentenbestand (`N_g`) sowie die gewichteten Neurenten (`n_g`) generiert. Danach werden alle Werte nach Jahr, Geschlecht, und Alter aggregiert (in den Rohdaten sind die Werte zusätzlich nach Rententeil (`pfrt`) desaggregiert):

```
IV_RENTEN_NEW <- IV_RENTEN_NEW %>%
    mutate(sex = ifelse(sex == 1, "m", "f"), N_g = pfrt * N_tot,
        n_g = pfrt * n_nr) %>%
    group_by(jahr, sex, alt) %>%
    summarize(mpr_tot = sum(mpr_tot, na.rm = TRUE), mpr_hr_tot = sum(mpr_hr_tot,
        na.rm = TRUE), mpr_kr_tot = sum(mpr_kr_tot, na.rm = TRUE),
        mpr_nr = sum(mpr_nr, na.rm = TRUE), mpr_hr_nr = sum(mpr_hr_nr,
            na.rm = TRUE), mpr_kr_nr = sum(mpr_kr_nr, na.rm = TRUE),
        bestand_pfrt = sum(N_g, na.rm = TRUE), neurenten_pfrt = sum(n_g,
            na.rm = TRUE), bestand_n = sum(N_tot, na.rm = TRUE),
        neurenten_n = sum(n_nr, na.rm = TRUE)) %>%
    ungroup()
```

Als nächstes werden diverse Dataframes eingelesen, die im aktuellen Finanzperspektivenmodell nicht mehr verwendet werden (das Einlesen erfolgt nur zwecks der Ermöglichung von Backtestings) und daher hier nicht erläutert werden:

```
IV_RENTEN_DWH <- read_excel(path = paste0(PARAM_INPUTS$path_iv_renten,
    PARAM_INPUTS$file_iv_renten), sheet = PARAM_INPUTS$sheet_iv_renten,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
)

IV_NEURENTEN <- read_excel(path = paste0(PARAM_INPUTS$path_iv_renten,
    PARAM_INPUTS$file_iv_renten), sheet = PARAM_INPUTS$sheet_iv_nr,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:5)
)
```

[20] Einfacher wäre natürlich die direkte Korrektur des Fehlers in den Rohdaten gewesen. Jedoch bestünde dann die Gefahr, dass der Fehler, falls auch in der Datenlieferung 2025 vorhanden, dann unentdeckt bliebe.

```
IV_ABGAENGE <- read_excel(path = paste0(PARAM_INPUTS$path_iv_renten,
    PARAM_INPUTS$file_iv_renten), sheet = PARAM_INPUTS$sheet_iv_abgaenge,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:5)
)

IV_HE_DWH <- read_excel(path = paste0(PARAM_INPUTS$path_iv_renten,
    PARAM_INPUTS$file_iv_renten), sheet = PARAM_INPUTS$sheet_iv_he,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
)
```

Als nächstes werden die Daten zu zu den Flüssen und Beständen bei den Hilflosenentschädigungen bei Erwachsenen eingelesen. Das Vorgehen ist, unter Beachtung der leicht anderen Rohdatenstruktur, grundsätzlich analog zum oben beschriebenen Vorgehen beim Einlesen der Daten zu den Renten. Der zweite Codeblock dient dazu, Zeilen mit 0-Werten für fehlende Geschlecht-Alter-Jahr Kombinationen einzufügen:

```
IV_HE_NEW <- read_excel(path = paste0(PARAM_INPUTS$path_iv_he_new,
    PARAM_INPUTS$file_iv_he_new), sheet = PARAM_INPUTS$sheet_iv_he_new,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
) %>%
    mutate(sex = ifelse(sex == 1, "m", "f"), alt = alter, mpr_n =
    ↪  ifelse(he_inzidenz_grad_bez ==
        "Neue HE", mpr, 0), n_he_n = ifelse(he_inzidenz_grad_bez ==
        "Neue HE", n_he, 0)) %>%
    group_by(jahr, sex, alt) %>%
    summarize(mpr = sum(mpr), bestand_n = sum(n_he), mpr_n = sum(mpr_n),
        neurenten_n = sum(n_he_n)) %>%
    ungroup()

IV_HE_NEW <- IV_HE_NEW %>%
    # Alle Kombinationen von jahr, sex, alt erstellen
complete(jahr = unique(IV_HE_NEW$jahr), nesting(sex, alt)) %>%
    # Fehlende Werte mit 0 füllen
mutate(mpr = ifelse(is.na(mpr), 0, mpr), mpr_n = ifelse(is.na(mpr_n),
    0, mpr_n), bestand_n = ifelse(is.na(bestand_n), 0, bestand_n),
    neurenten_n = ifelse(is.na(neurenten_n), 0, neurenten_n)) %>%
    arrange(jahr, sex, alt)
```

Danach werden die Daten zu zu den Flüssen und Beständen bei den Hilflosenentschädigungen und dem Intensivpflegezuschlag bei Kindern eingelesen (da dieses Register separat vom Erwachsenen Hilflosenentschädigungsregister geführt wird, müssen die Daten separat eingelesen werden). Hierbeit wird genau gleich wie bei der Erwachsenen Hilflosenentschädigung vorgegangen:

```
IV_HE_KINDER <- read_excel(path = paste0(PARAM_INPUTS$path_iv_he_kinder,
    PARAM_INPUTS$file_iv_he_kinder), sheet = PARAM_INPUTS$sheet_iv_he_kinder,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
) %>%
    mutate(sex = ifelse(sex == 1, "m", "f"), alt = age, alt = ifelse(alt >
        18, 18, alt), mpr = mpr_jahr/12, mpr_n = nr_he/n_dist_he *
        mpr_jahr/12) %>%
    group_by(jahr, sex, alt) %>%
    summarize(mpr = sum(mpr), bestand_n = sum(n_dist_he), mpr_n = sum(mpr_n),
        neurenten_n = sum(nr_he)) %>%
    ungroup()
```

```
IV_HE_KINDER <- IV_HE_KINDER %>%
    # Alle Kombinationen von jahr, sex, alt erstellen
complete(jahr = unique(IV_HE_KINDER$jahr), nesting(sex, alt)) %>%
    # Fehlende Werte mit 0 füllen
mutate(mpr = ifelse(is.na(mpr), 0, mpr), mpr_n = ifelse(is.na(mpr_n),
    0, mpr_n), bestand_n = ifelse(is.na(bestand_n), 0, bestand_n),
    neurenten_n = ifelse(is.na(neurenten_n), 0, neurenten_n)) %>%
    arrange(jahr, sex, alt)
```

Danach werden die Individualdaten aus dem Rentenregister für das letzte verfügbare Jahr (typischerweise das Vorjahr) eingelesen:

```
IV_RR_RAM <- readr::read_delim(file = paste0(PARAM_INPUTS$path_iv_rr_ram,
    PARAM_INPUTS$file_iv_rr_ram), delim = ";", show_col_types = FALSE,
    trim_ws = TRUE)
```

Zum Schluss werden die eingelesenen Daten an `prepare_input.R` zurückgegeben:

```
#------ Output --------------------------------------------------------#
mod_return(IV_RENTEN_NEW, IV_RENTEN_DWH, IV_NEURENTEN, IV_ABGAENGE,
    IV_HE_DWH, IV_HE_NEW, IV_HE_KINDER, IV_RR_RAM)
```

3.11 Modul mod_input_umfragen.R

Dokumentation zuletzt aktualisiert am 26.09.2024

Das Modul `mod_input_umfragen.R` liest die Daten zu den verschiedenen Ausgabepositionen der IV, für welche die Projektionen für die 5 Jahre ab der aktuellen Abrechnung von den BSV-internen Fachbereichen erstellt werden, ein. `mod_input_umfragen.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_umfragen <- mod_input_umfragen(PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
UMFRAGEN <- read_excel(path = paste0(PARAM_INPUTS$path_umfragen,
    PARAM_INPUTS$file_umfragen), sheet = PARAM_INPUTS$sheet_umfragen,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:8)
)
#------ Output --------------------------------------------------------#
mod_return(UMFRAGEN)
```

3.12 Modul mod_input_estv_iv.R

Dokumentation zuletzt aktualisiert am 25.09.2024

Das Modul `mod_input_estv.R` liest MWST-Projektion der ESTV für die Berechnung des Bundesbeitrages, welche nach eigenen gesetzlichen Vorgaben berechnet wird, ein. Es handelt sich hier nicht um die generelle MWST-Projektion der ESTV, welche Fall, dass die Auswirkungen einer MWST-Zusatzfinanzierung abgeschätzt werden sollte, verwendet wird und mit dem Code in Kapitel 3.6 importiert wird. `mod_input_estv.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_estv_iv <- mod_input_estv_iv(PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
#------ Lesen Inhalt Excel - sheet ------------------------------------#
ESTV_IV <- read_excel(path = paste0(PARAM_INPUTS$path_estv_iv,
    PARAM_INPUTS$file_estv_iv), sheet = PARAM_INPUTS$sheet_estv_iv,
    range = cell_limits(c(1, 1), c(NA, 9))  # cell_cols(1:9)
,
    col_types = c(rep("guess", 8), "numeric"))
#------ Output -------------------------------------------------------#
mod_return(ESTV_IV)
```

3.13 Modul mod_input_mwst_satz_iv.R

Dokumentation zuletzt aktualisiert am 26.09.2024

Das Modul `mod_input_mwst_satz_iv.R` liest den MWST-Satz ein, der bei einer MWST-Zusatzfinanzierung zugunsten der IV angewendet wird. Momentan (Stand 2024) enthalten die hier eingelesenen Daten nur für die Jahre 2011-2018, also für die Zeitspanne für welche die IV durch die MWST zusatzfinanziert wurde, von 0 abweichende Werte. `mod_input_mwst_satz_iv.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_mwst_satz_iv <- mod_input_mwst_satz_iv(PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
MWST_SATZ_IV <- read_excel(path = paste0(PARAM_INPUTS$path_mwst_satz_iv,
    PARAM_INPUTS$file_mwst_satz_iv), sheet = PARAM_INPUTS$sheet_mwst_satz_iv,
    range = cell_limits(c(1, 1), c(NA, NA)))
#------ Output -------------------------------------------------------#
mod_return(MWST_SATZ_IV)
```

3.14 Modul mod_input_sv_beitragssatz.R

Dokumentation zuletzt aktualisiert am 26.09.2024

Das Modul `mod_input_sv_beitragssatz.R` liest die vollständige Historie der gültigen Sozialversicherungsbeitragssätze für beitragspflichtige Arbeitnehmer, Arbeitgeber, Selbständigerwerbende und Nichterwerbstätige ein. `mod_input_sv_beitragssatz.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_sv_beitragssatz <- mod_input_sv_beitragssatz(PARAM_INPUTS)
```

Das Modul besteht lediglich aus dem folgenden Code, mit welchem die Werte eingelesen und danach an `prepare_input.R` zurückgegeben werden:

```
SV_BEITRAGSSATZ <- read_excel(path = paste0(PARAM_INPUTS$path_sv_beitrag,
    PARAM_INPUTS$file_sv_beitrag), sheet = PARAM_INPUTS$sheet_sv_beitrag,
    range = cell_limits(c(11, 1), c(NA, 20)), col_types = "numeric")
#------ Output -------------------------------------------------------#
mod_return(SV_BEITRAGSSATZ)
```

3.15 Modul mod_input_iv_med_massnahmen.R

Dokumentation zuletzt aktualisiert am 29.08.2024

Das Vorgehen für das jährliche Datenupdate ist direkt im File mit den Input-daten beschrieben (sheet *Jährliches Datenupdate*): /05_data/IV/iv_medizinische_massnahmen.xlsx

Das Modul `mod_input_iv_med_massnahmen.R`, liest die Registerdaten zu den Rechnungen und der Anzahl Beziehende von medizinischen Massnahmen, sowie Daten zu allfälligen Sondereffekten und Parameterwechseln (siehe Auch Kapitel 4.5.2 und 4.5.6) ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
tl_iv_med_massnahmen <- mod_input_iv_med_massnahmen(PARAM_INPUTS)
```

Die Elemente von `mod_input_iv_med_massnahmen.R` werden nachfolgend erläutert.

3.15.1 Einlesen von Registerdaten zu Rechnungen und Anzahl Beziehende

```
#------ Lesen Inhalt Excel - sheet ------------------------------------#
# Funktion zum Importieren eines Excel-Blatts mit MM-Daten aus DWH und Bereitstellen
# eines aufgeräumten Dataframes
process_sheet <- function(sheet_name) {

  suppressMessages({
  # Das spezifische Blatt aus der Excel-Datei lesen
  df <- read_excel(path = paste0(PARAM_INPUTS$path_iv_med_massnahmen,
                                 PARAM_INPUTS$file_iv_med_massnahmen),
                   sheet = sheet_name)

  df <- df %>%
    slice(-1:-5) %>%  # Entferne die ersten 5 Zeilen
    select(-ncol(.))  # Entferne die letzte Spalte

  colnames(df) <- as.character(unlist(df[1, ]))  # Ersetze Spaltennamen
  colnames(df)[1] <- "alt"  # Setze den Namen der ersten Spalte auf "alt"

  df <- df %>%
    slice(-1) %>%  # Entferne die erste Zeile
    slice(-n())  # Entferne die letzte Zeile

  # Konvertiere in das Long-Format
  df <- df %>%
    pivot_longer(
      cols = -alt,  # Behalte "alt" als separate Spalte
      names_to = "jahr",  # Setze den Namen der neuen Spalte auf "leistungscode"
      values_to = "wert"  # Setze den Namen der Spalte für die Werte
    )

  # Die Daten in ein aufgeräumtes Format umformen und die Anzahl aggregieren
  df <- df %>%
    mutate(
      jahr = as.numeric(jahr),
      alt = ifelse(alt == "111 +", "21", alt),
      alt = as.numeric(alt),
      alt = ifelse(alt >= 21, 21, alt),
      wert = ifelse(wert == "-", "0", wert),
      wert = as.numeric(wert)
    ) %>%
    arrange(jahr, alt)

  df <- df %>%
    group_by(jahr, alt) %>%  # Gruppiere nach "jahr", "alt" und "leistungscode"
```

```
      summarise(wert = sum(wert, na.rm = TRUE))  # Summiere "wert" für jede Gruppe

    if (str_detect(sheet_name, "Rechnungssumme")) {
      df <- df %>%
        rename(ausgaben_nom = wert)
    }
    if (str_detect(sheet_name, "Bezueger")) {
      df <- df %>%
        rename(invalide = wert)
    }

    return(df)
    })
  }
```

Diese Funktion hat zum Ziel, die Daten in den *Rentensumme* und *Bezueger* sheets im Rohdatenfile (`\05_data\IV\iv_medizinische_massnahmen.xlsx`) einzulesen. Sie wird durch die folgenden Befehle aufgerufen:

```
# Ein leeres Dataframe initialisieren, um alle Ergebnisse
# zu speichern
IV_MED_MASSN_REGISTER <- data.frame()

BEZUEGER1 <- process_sheet("Rechnungssumme2012-2017")
RECHNUNGSSUMME1 <- process_sheet("Bezueger2012-2017")
BEZUEGER2 <- process_sheet("Rechnungssumme2018-2021")
RECHNUNGSSUMME2 <- process_sheet("Bezueger2018-2021")

IV_MED_MASSN_REGISTER <- bind_rows(full_join(BEZUEGER1, RECHNUNGSSUMME1,
    by = c("jahr", "alt")), full_join(BEZUEGER2, RECHNUNGSSUMME2,
    by = c("jahr", "alt")))
rm(BEZUEGER1, RECHNUNGSSUMME1, BEZUEGER2, RECHNUNGSSUMME2)


# Initialisiere das Jahr
i <- 2022
suppressMessages({
    while (TRUE) {
        # Versuche, das Blatt 'Bezueger' für das aktuelle
        # Jahr zu lesen
        existsBezueger <- tryCatch({
            read_excel(path = paste0(PARAM_INPUTS$path_iv_med_massnahmen,
                PARAM_INPUTS$file_iv_med_massnahmen), sheet = paste0("Bezueger",
                i))
            TRUE  # Wenn erfolgreich, gibt TRUE zurück
        }, error = function(e) {
            FALSE  # Wenn ein Fehler auftritt (Blatt existiert nicht), gibt FALSE zurück
        })

        # Beende die Schleife, wenn das Blatt nicht
        # existiert
        if (!existsBezueger)
            break
```

```
        # Jedes Blatt für das gegebene Jahr verarbeiten
        BEZUEGER <- process_sheet(paste0("Bezueger", i))
        RECHNUNGSSUMME <- process_sheet(paste0("Rechnungssumme",
            i))

        # Mit den Daten aus allen anderen Jahren
        # kombinieren
        IV_MED_MASSN_REGISTER <- bind_rows(IV_MED_MASSN_REGISTER,
            full_join(BEZUEGER, RECHNUNGSSUMME, by = c("jahr",
                "alt")))
        rm(BEZUEGER, RECHNUNGSSUMME)

        i <- i + 1  # Erhöhe das Jahr für den nächsten Durchlauf
    }
})
```

In einem ersten Schritt wird ein leeres Dataframe `IV_MED_MASSN_REGISTER` initialisiert, in welches dann die Daten geschrieben werden. Danach wird die oben dargestellte Funktion `process_sheet` aufgerufen, um die Rechnungssummen sowie die Beziehendezahlen für 2012 bis 2021 einzulesen.

Für die Jahre ab 2022 sind die Rechnungssummen und Beziehendezahlen für jedes Jahr in einem separaten Sheet enthalten. Daher wird ab 2022 eine While-Schleife gestartet, die solange die Excel-sheets einliest, bis für ein Jahr kein Excel-sheet mehr existiert. Die Excel-sheets werden also automatisch und ohne weitere Parametereingabe bis zum aktuellsten Jahr eingelesen.

3.15.2 Einlesen von Daten zu Sondereffekten

Die nachfolgenden Codezeilen dienen dazu, die Daten zu den Sondereffekten aus den Rohdaten einzulesen (mehr Details zu den Sondereffekten in Kapitel 4.5.2):

```
#------ Lesen Inhalt Excel - sheet Sondereffekte ------------------------------------#

# Excel-Datei einlesen
suppressMessages({
    IV_MED_MASSN_SONDEREFFEKTE <- read_excel(path =
↪   paste0(PARAM_INPUTS$path_iv_med_massnahmen,
        PARAM_INPUTS$file_iv_med_massnahmen), sheet = "Sondereffekte",
        col_types = "text") %>%
        rename(Jahr = ...1, Alter = ...2, `Permanent Invalidisierung` = `Permanente
        ↪  Veränderung`,
            `Permanent Ausgaben` = ...4, `Temporär Invalidisierung` = `Temporäre
            ↪  Veränderung`,
            `Temporär Ausgaben` = ...6) %>%
        slice(-1)
})
# Hilfsfunktionen definieren
expand_alter <- function(alter) {
    if (str_detect(alter, ";")) {
        # Liste von Zahlen
        return(as.numeric(unlist(str_split(alter, ";"))))
    } else if (str_detect(alter, "-")) {
        # Bereich von Zahlen
        range <- as.numeric(unlist(str_split(alter, "-")))
        return(seq(range[1], range[2]))
    } else {
```

```
        # Einzelne Zahl
        return(as.numeric(alter))
    }
}

# Dataframe erweitern und transformieren
IV_MED_MASSN_SONDEREFFEKTE <- IV_MED_MASSN_SONDEREFFEKTE %>%
    mutate(sondereffektjahr = as.numeric(Jahr), sondereffektalter = map(Alter,
        expand_alter), permanent_invalidisierung = ifelse(`Permanent Invalidisierung` ==
        "x", "permanent-invalidisierung", NA), permanent_ausgaben = ifelse(`Permanent
        ↪  Ausgaben` ==
        "x", "permanent-ausgaben", NA), temporaer_invalidisierung = ifelse(`Temporär
        ↪  Invalidisierung` ==
        "x", "temporaer-invalidisierung", NA), temporaer_ausgaben = ifelse(`Temporär
        ↪  Ausgaben` ==
        "x", "temporaer-ausgaben", NA)) %>%
    select(sondereffektjahr, sondereffektalter, permanent_invalidisierung,
        permanent_ausgaben, temporaer_invalidisierung, temporaer_ausgaben) %>%
    unnest(sondereffektalter) %>%
    pivot_longer(cols = starts_with("permanent_") | starts_with("temporaer_"),
        names_to = "sondereffekt", values_drop_na = TRUE) %>%
    select(-value)
```

3.15.3 Einlesen von Daten zu Parameterkorrekturen

Die nachfolgende Codezeile dient dazu, die Daten zu den manuellen Parameterkorrekturen aus den Rohdaten einzulesen (mehr Details zu den manuellen Parameterkorrekturen in Kapitel 4.5.6):

```
#------ Lesen Inhalt Excel - sheet Parameteranpassungen -----------------------------#
IV_MED_MASSN_PARAMCHANGES <- read_excel(path =
↪  paste0(PARAM_INPUTS$path_iv_med_massnahmen,
    PARAM_INPUTS$file_iv_med_massnahmen), sheet = "Parameteranpassungen")
```

3.15.4 Übermitteln der Inputdaten zu medizinischen Massnahmen

Die drei Dataframes zu den medizinischen Massnahmen werden von `mod_input_iv_med_massnahmen.R` wie folgt an `prepare_input.R` zurückgegeben:

```
#------ Lesen Inhalt Excel - sheet Parameteranpassungen ------------------#
mod_return(IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES)
```

`prepare_input.R` fügt diese drei Dataframes dann der `iv_go` liste hinzu, wodurch diese dann im jeweiligen Input-Container im Ordner iv/go abgespeichert werden:

```
tl_iv_med_massnahmen <- mod_input_iv_med_massnahmen(PARAM_INPUTS)


#------ geltende Ordnung IV ----------------------------------------#
iv_go <- c(tl_ivrenten, tl_ivgrundlagen, tl_umfragen, tl_estv_iv,
    tl_mwst_satz_iv, tl_sv_beitragssatz, tl_iv_med_massnahmen)
```

4 Module zur Berechnung der Finanzperspektiven

In diesem Kapitel werden die Module beschrieben, mithilfe welcher das Finanzperspektivenmodell IV berechnet wird. Die hier beschriebenen Module lesen die in Kapitel 3 aufbereiteten Input-Daten sowie die Datei `PARAM_GLOBAL.csv` ein, führen die Berechnungen des Finanzperspektivenmodells durch, und lesen verschiedene Tabellen mit Resultaten aus. Hierzu zählen insbesondere auch die auf der Internetseite des BSV veröffentlichten Tabellen zu den finanziellen Perspektiven der IV.[21]

4.1 Modul run_iv.R

Dokumentation zuletzt aktualisiert am 10.10.2024

Das Hauptmodul zur Berechnung des Finanzperspektivenmodells der IV wird wie folgt aufgerufen:

```
run_iv(path = path_parametercontainer, path_out = path_output)
```

`path_parametercontainer` ist der Pfad zum Ordner, welcher die zu verwendenden Modellparameter enthält, insbesondere auch die Datei `PARAM_GLOBAL.csv`. `path_output` ist der Pfad, wo die Resultate der Modellberechnungen schlussendlich abgelegt werden sollen.

Das Modul `run_iv` enthält die folgenden Elemente:

```
run_iv <- function(path, path_out, param_replace = NULL) {
```

Das Modul nimmt den vorangehend spezifizierten Pfade `path` und `path_out` entgegegen. Zudem wird die Variable `param_replace` auf `NULL` gesetzt.

Der folgende Codeteil ist für den Fall, dass unter `path_parametercontainer` eine Liste von Pfade spezifiziert wird, was dazu führt, dass die Finanzperspektivenberechnungen nacheinander für jede in `path_paramcontainer` spezifizierte `PARAM_GLOBAL` Datei ausgeführt werden:

```
if (length(path) > 1) {
    return(multi_run(path, run_iv, path_out))
}
```

Danach werden die Input-Daten eingelesen:

```
# Allgemeine Input-Daten einlesen (sind gemäss Konvention
# im AHV-Inputordner abgelegt):
tl_inp <- mod_inp(path, vz = "ahv") %>%
    param_replace(param_replace = param_replace)

# IV-spezifische Input-Daten einlesen:
tl_inp_iv <- mod_inp(path, vz = "iv") %>%
    param_replace(param_replace = param_replace)

# Allgemeine und IV-spezifische Input-Daten zusammenfügen:
tl_inp <- merge_tl(tl_inp_iv, tl_inp)
rm(tl_inp_iv)
```

Da das Finanzperspektivenmodell der IV auch auf Inputdaten zugreift, die unter den Inputdaten für die AHV abgespeichert sind (bspw. Bevölkerungsdaten, siehe auch Kapitel 3.1), wird `mod_inp` auch für den Versicherungszweig AHV (vz = "ahv") ausgeführt. Zum Schluss werden die Input-Daten für das AHV-Finanzperspektivenmodell mit den IV-spezifischen Inputdaten zusammengefügt und im dataframe tl_inp gespeichert (`tl_inp <- merge_tl(tl_inp_iv, tl_inp)`).

Als nächstes werden die Paramterwerte überprüft und default-Parameterwerte gesetzt:

[21] https://www.bsv.admin.ch/bsv/de/home/sozialversicherungen/iv/finanzen-iv.html

```
# Modul zum überprüfen von Parameterwerten und setzten von
# default-Parameterwerten:
tl_iv_param_global <- mod_iv_param_global(tl_inp)

# Ersetze PARAM_GLOBAL in tl_inp durch PARAM_GLOBAL in
# tl_iv_param_global
tl_inp$PARAM_GLOBAL <- tl_iv_param_global$PARAM_GLOBAL

# Sicherstellen, dass alle hier hinzugefügten Parameter
# auch in PARAM_ALL enthalten sind
tl_inp <- tl_inp %>%
    param_replace(param_replace = param_replace)

# Massnahmen-Parameterliste erstellen und zu PARAM_ALL
# hinzufügen
if (tl_inp$PARAM_GLOBAL$flag_param_massn) {
    tl_inp$PARAM_ALL <- tl_inp$PARAM_ALL %>%
        bind_cols(tl_inp$PARAM_MASSNAHMEN_BASE)
}
```

Das Modul `mod_iv_param_global` dient dazu, default-Parameterwerte zu setzten, falls diese nicht in `PARAM_GLOBAL` spezifiziert sind (vgl. Kapitel 4.1.3).[22]

In einem nächsten Schritt werden die vorbereitenden Berechnungen für das IV-Finanzperspektivenmodell durchgeführt:

```
tl_vorb_ber <- wrap_vorb_berechn(tl_inp = tl_inp)
tl_inp <- merge_tl(tl_inp, tl_vorb_ber)
rm(tl_vorb_ber)
```

Diese Berechnungen dienen dazu, Inputvariablen zu berechnen, die an mehreren Orten im Finanzperspektivenmodell verwendet werden. Das Modul `wrap_vorb_berechn` ist in Kapitel 4.1.1 erläutert.

Dadurch ist die Datenaufbereitung abgeschlossen, und es wird somit das Modul `wrap_iv` (vgl. Kapitel 4.1.2) aufgerufen, in welchem die einzelnen Ausgaben- und Einnahmenpositionen des Finanzperspektivenmodells berechnet werden:

```
tl_out_iv <- wrap_iv(tl_inp)
```

Zum Schluss werden die in `wrap_iv` berechneten Ausgaben und Einnahmen zusammengefügt und die Output-Dateien werden aufbereitet und in den vorangehend spezifizierten Output-Ordner geschrieben:

```
# Identifikationsnummer erstellen (um Output-Ordner zu
# benennen)
PARAM_OUT_IV <- tibble(identifier_number = ffh_identifier_number(vz = "iv",
    path))
tl_inp <- merge_tl(tl_inp, tidylist(PARAM_OUT_IV))

# Output-Dateien in den Output-Ordner schreiben
path_out_iv <- mod_out_iv(path, path_out, tl_inp, tl_out_iv)
return(path_out_iv)
```

Das Modul `ffh_identifier_number` dient dazu, eine Identifikationsnummer zu generieren, mit welcher der Ordner mit den Output-Dateien benannt wird. Danach werden die Dateien mit dem Modul `mod_out_iv` in

[22] Dieses Modul wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ implementiert.

den Output-Ordner geschrieben, und der Pfad zum Output-Ordner wird in der R-Konsole ausgelesen.

4.1.1 Modul wrap_vorb_berechn.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Modul `wrap_vorb_berechn.R` führt Vorberechnungen für Projektionsvariablen durch, die in allen Finanzperspektivenmodellen verwendet werden (bspw. Lohn- und Preisentwicklung). Das Modul `wrap_vorb_berechn.R` wird also nicht nur in den Finanzperspektiven der IV verwendet, sondern wird bspw. auch im Finanzperspektivenmodell der AHV aufgerufen. Dies stellt die Abgrenzung zum weiter unten erläuterten Modul `wrap_iv_vorb_berechn.R` (vgl. Kapitel 4.1.4) dar, in welchem IV-spezifische Projektionsvariablem berechnet werden.

`wrap_vorb_berechn.R` wird in `run_iv.R` (vgl. Kapitel 4.1) wie folgt aufgerufen:

```
tl_vorb_ber <- wrap_vorb_berechn(tl_inp = tl_inp)
```

In einem ersten Schritt werden darin die Daten der IV-Abrechnung eingelesen, wobei im Modul `mod_abrechnung` beachtet wird, ob die provisorische oder die definitive Abrechnung verwendet werden soll:

```
# Abrechnung
tl_abrechnung <- mod_abrechnung(list = tl_inp)
```

Als nächstes werden die Bevölkerungsdaten eingelesen, wobei im Modul `mod_population` (vgl. Kapitel 4.24) die Bevölkerungsprognosen der BFS-Bevölkerungsszenarien, welche nur alle 5 Jahre aktualisiert werden, auf die letzten verfügbaren Daten aus `STAT_POP` rebasiert werden:

```
  # Backward compatibility (to ensure compatibility with older containers)
  if (is.null(tl_inp$PARAM_GO_BASE)) {
    tl_bevoelkerung <- mod_population(
      list = tl_inp
    )
  } else {
```

Auf den Teil nach `else` wird hier nicht eingegangen, da `is.null(tl_inp$PARAM_GO_BASE)` in der Standardmässigen Parameterkonfiguration `TRUE` ist.

Als erstes werden die Eckwerte zur wirtschaftlichen Entwicklung aufbereitet:

```
# Berechnung Eckwerte

tl_eckwerte <- mod_eckwerte(list = tl_inp)
```

Das Modul `mod_eckwerte` (vgl. Kapitel 4.25) fügt historische Daten zur Lohn- und Preisentwicklung mit den Projektionsdaten zusammen und erstellt so eine Zeitreihe für die Entwicklung dieser wirtschaftlichen Eckwerte.

In einem nächsten Schritt wird die Entwicklung des Durchschnittslohnes, was in der Modellterminologie als Einkommensentwicklung bezeichnet wird, berechnet:

```
# Berechnung Eckwerte
tl_eink_entwicklung <- mod_eink_entwicklung(list = c(tl_inp,
    tl_eckwerte))
```

Hierzu wird im Modul `mod_eink_entwicklung` (vgl. Kapitel 4.26) das Produkt aus der (projizierten) Entwicklung des Schweizer Lohnindexes (SLI) und dem Sturkturfaktor (für welchen Stand 2024 angenommen wird, dass er 0.3% beträgt) gebildet.

Danach wird der Deflator relativ zum aktuellen Jahr, was in der Modellterminologie als Diskontfaktor bezeichnet wird, berechnet:

```
# Berechnung Diskontfaktor

tl_diskontfaktor <- mod_diskontfaktor(list = c(tl_inp, tl_eckwerte))
```

Das Modul `mod_diskontfaktor` (vgl. Kapitel 4.27) berechnet den Diskontfaktor so, dass sämtliche nominellen Grössen durch multiplikation mit dem Diskontfaktor zu Preisen vom Preisbasis-Jahr (typischerweise das Jahr der letzten Abrechnung) umgerechnet werden (dh. der Diskontfaktor ist 1 im Preisbasis-Jahr, und bei positiver Inflation < 1 in der Zukunft und > 1 in der Vergangenheit).

Als nächstes wird der Strukturfaktor berechnet:

```
tl_strukturfaktor <- mod_strukturfaktor(list = c(tl_inp, tl_bevoelkerung,
    tl_eink_entwicklung, tl_diskontfaktor, tl_abrechnung))
```

Das Modul `mod_strukturfaktor` (vgl. Kapitel 4.28) wählt den Strukturfaktor so, dass dieser in den Ersten 2 Jahren ab der aktuellen Abrechnung zusammen mit der Entwicklung der Erwerbsbevölkerung gemäss BFS-Szenarion dem BESTA-Erwerbsbevölkerungswachstum +0.3% entspricht. Für die Jahre danach entspricht der Strukturfaktor genau 0.3%. Die Anpassung in den ersten zwei Jahren nach der aktuellen Abrechnung stellt sicher, dass das Wachstum der Lohnsumme, und daher das Wachstum der Lohnbeiträgen, mit den Beschäftigungsprojektionen der BESTA (welche im Gegensatz zu den BFS-Erwerbsbevölkerungsszenarien jährlich aktualisiert werden) konsistent ist.

Da das Modul für diese Korrektur die Entwicklung des Durchschnittslohnes unter der Annahme, dass der Strukturfaktor auch in den Ersten 2 Jahren ab der aktuellen Abrechnung 0.3 beträgt, verwendet, welche vorangehend im Modul `mod_eink_entwicklung` berechnet und in der Liste `tl_eink_entwicklung` gespeichert wurde, muss diese Berechnung nach einer ersten Berechnung von `tl_eink_entwicklung` erfolgen. Das heisst auch, dass `tl_eckwerte` und `tl_eink_entwicklung` jetzt mit dem aktualisierten Strukturfaktor noch einmal berechnet werden müssen, was mit dem folgenden Codeteil geschieht:

```
tl_inp$ECKWERTE <- tl_strukturfaktor$ECKWERTE

# Berechnung Eckwerte

tl_eckwerte <- mod_eckwerte(list = tl_inp)

# Berechnung Einkommensentwicklung

tl_eink_entwicklung <- mod_eink_entwicklung(list = c(tl_inp,
    tl_eckwerte))
```

D.h., die Eckwerte im Input-Dataframe werden durch den Befehl `tl_inp$ECKWERTE<- tl_strukturfaktor $ECKWERTE` mit den im Modul `mod_strukturfaktor` berechneten neuen Eckwerten ersetzt, wobei sich nur der Strukturfaktor für die ersten zwei Jahre der aktuellsten Eckwerte verändert, und alle anderen Werte identisch sind. danach werden die Module `mod_eckwerte` und `mod_eink_entwicklung` nochmals ausgeführt.

Als nächstes wird die Entwicklung der Minimalrente berechnet:

```
# Berechnung Rentenentwicklung gemaess Rentenanpassungen

tl_rentenentwicklung <- mod_rentenentwicklung(list = c(tl_inp,
    tl_eckwerte))
```

Hierbei wird im Modul `mod_rentenentwicklung` (vgl. Kapitel 4.29) zuerst die für die Berechnung der Minimalrente massgebliche Entwicklung des Mischindexes gemäss der Entwicklung des nominalen Schweizerischen

Lohnindexes (SLI) und der Preisentwicklung berechnet, und danach die daraus folgende zukünftige Entwicklung der Minimalrente berechnet (vgl. Kapitel 4.29).

Als nächstes wird die erwartete Rendite der nicht flüssigen Fondsanalagen gemäss dem ausgewählten Zinsszenario eingelesen (vgl. Kapitel 4.30:

```
# Berechnung der nominellen Rendite auf den nichtfluessigen
# Fondsanlagen

tl_zins <- mod_zins_scen(list = c(tl_inp, tl_eckwerte))
```

Es folgen Berechnungen, die für das Finanzperspektivenmodell der IV nicht relevant sind. Diese werden daher hier nicht erläutert. Am Ende von `wrap_vorb_berechn` werden die Dataframes mit den Vorberechnungen an das Finanzperspektivenmodell (respektive das Modul `run_iv.R`) zurückgegeben. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
else {
    c(
      tl_abrechnung,
      tl_bevoelkerung,
      tl_eckwerte,
      tl_eink_entwicklung,
      tl_rentenentwicklung,
      tl_zins,
      tl_scenario_erstrenten,
      tl_diskontfaktor,
      tl_bundesanteilpzt_vekt,
      tl_beitragssatz_vekt,
      tl_demografiepzt_vekt,
      tl_indices_futurs,
      tl_ivschuld,
      tl_uebr_einn_scen
    )
  }
```

4.1.2 Modul wrap_iv.R

Dokumentation zuletzt aktualisiert am 07.10.2024

Das Modul zur Berechnung des Finanzperspektivenmodells der IV wird in `run_iv.R` wie folgt aufgerufen:

```
tl_out_iv <- wrap_iv(tl_inp)
```

Am Anfang des Moduls werden die IV-spezifischen Vorberechnungen durchgeführt.

```
#------ Vorbereitende Berechnungen iv ---------------------------------#
tl_vorb_iv <- wrap_iv_vorb_berechn(tl_inp = tl_inp)
tl_inp <- merge_tl(tl_inp, tl_vorb_iv)
rm(tl_vorb_iv)
```

Das Modul `wrap_iv_vorb_berechn` ist in Kapitel 4.1.4 beschrieben.

Als nächstes wird das Modul aufgerufen, dass die Hauptberechnungen zum Finanzperspektivenmodell, also die Berechnung der einzelnen Einnahmen- und Ausgabepositionen, durchführt:

```
#------ Hauptberechnung: Einnahmen/Ausgaben g.O. ----------------------#
tl_iv_hauptberechnung <- wrap_iv_hauptberechnung(tl_inp = tl_inp)
```

Das Modul `wrap_iv_hauptberechnung` ist in Kapitel 4.1.5 beschrieben.

Danach werden die folgenden Codezeilen ausgeführt:

```
if (tl_inp$PARAM_GLOBAL$rentenmodell != "OLD") {
    tl_inp$BESTAND_IV <- tl_iv_hauptberechnung$BESTAND_IV
    tl_inp$RENTENBESTAND_IV <- tl_iv_hauptberechnung$RENTENBESTAND_IV
    tl_iv_hauptberechnung$BESTAND_IV <- NULL
    tl_iv_hauptberechnung$RENTENBESTAND_IV <- NULL
}
```

Dieser Code dient dazu, im Falle dass nicht das alte Rentenmodell in `PARAM_GLOBAL` ausgewählt wurde (im ab Herbst 2024 gültigen Modell sollte `tl_inp$PARAM_GLOBAL$rentenmodell != `"OLD "immer `=TRUE` sein), die Projektionen zum IV-Rentenbestand mit den im Hauptberechnungsmodul projizierten Dataframes zu ersetzen (im alten Rentenmodell wurden die Dataframes BESTAND_IV und RENTENBESTAND_IV in `wrap_iv_vorb_berechn` berechnet).

Danach werden die folgenden Codezeilen ausgeführt:

```
tl_inp <- append(tl_inp, list(IV_MED_MASSN_FORECAST_AVERAGES =
↪  tl_iv_hauptberechnung$IV_MED_MASSN_FORECAST_AVERAGES))  #
↪  IV_MED_MASSN_FORECAST_AVERAGES zu tl_inp hinzufügen, da
↪  IV_MED_MASSN_FORECAST_AVERAGES Modell-input-parameter enthält
tl_iv_hauptberechnung$IV_MED_MASSN_FORECAST_AVERAGES <- NULL  # Entferne das Element
```

Dieser Codeteil dient dazu, das Dataframe `IV_MED_MASSN_FORECAST_AVERAGES` (vgl. Kapitel 4.5) an `tl_inp` anzufügen. Dies wird gemacht, um sicherzustellen, dass `IV_MED_MASSN_FORECAST_AVERAGES` zu den Output-Files am Ende hinzugefügt wird, und so nach den Modellberechnungen überprüft und plausibilisiert werden kann.

Mit dem nachfolgenden Codeblock werden die Auswirkungen der im Parameter-Ordner ausgewählten Massnahmen berechnet:

```
#------ Massnahmen auswerten ----------------------------------------#
tl_iv_mass <- wrap_iv_massnahmen(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung)
```

Das Modul `wrap_iv_massnahmen` ist in Kapitel 4.1.11 beschrieben.

Der darauffolgende Codeblock dient dazu, aus den einzelnen Ausgaben- und Einnahmenprojektionen der Hauptberechnungen sowie der Massnahmenberechnungen die Bilanz und Erfolgsrechnung der IV zu berechnen:

```
#------ Berechnungen Erfolgsrechnung und Bilanz ----------------------#
tl_iv_ergebnisse <- wrap_iv_ergebnisse(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung,
    tl_iv_mass = tl_iv_mass)
```

Das Modul `wrap_iv_ergebnisse` ist in Kapitel 4.1.12 beschrieben.

Darauffolgend werden für das Finanzperspektivenmodell nicht direkt relevante Nebenergebnisse berechnet:

```
#------ Berechnungen Varia (Anzahl Rentner etc.) (Platzhalter)---------#
tl_iv_varia <- wrap_iv_varia(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung,
    tl_iv_mass = tl_iv_mass, tl_iv_ergebnisse = tl_iv_ergebnisse)
```

Das Modul `wrap_iv_varia` ist in Kapitel 4.1.13 beschrieben.

Der Nachfolgende Codeblock dient dazu, die Berechnungen des Finanzperspektivenmodells, welche nominal sind, in reale Grössen (dh. zu Preisen im ausgewählten Jahr, typischerweise dem Jahr der letzten IV-Abrechnung) umzurechnen:

```
#------ post-processing ------------------------------------------------#
tl_iv_postprocessing <- mod_iv_postprocessing(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_BILANZ = tl_iv_ergebnisse$BILANZ_IV, IV_MASSNAHMEN = tl_iv_mass$IV_MASSNAHMEN,
    IV_INDICES = tl_iv_varia$IV_INDICES, DISKONTFAKTOR = tl_inp$DISKONTFAKTOR)
```

Das Modul `mod_iv_postprocessing` ist in Kapitel 4.1.14 beschrieben.

Der nachfolgende Codeblock dient der Aufbereitung der Einnahmen- und Ausgabeprojektionen für die Finanzplan-Periode (dh. das aktuelle Jahr und die 4 darauffolgenden Jahre). Er ist für die Berechnung der Finanzperspektiven nicht relevant, sondern bereitet lediglich die berechneten Einnahmen- und Ausgabenvektore für einen beschränkten Zeitraum anders auf (im breiten anstatt im langen Format):

```
#------ vafp -----------------------------------------------#
tl_iv_vafp <- mod_iv_vafp(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_ABRECHNUNG_DEF = tl_inp$IV_ABRECHNUNG_DEF, IV_ABRECHNUNG_PROV =
    ↪  tl_inp$IV_ABRECHNUNG_PROV,
    IV_FHH_NOM = tl_iv_postprocessing$IV_FHH_NOM)
```

Somit können die Berechnungen an das Finanzperspektivenmodell (respektive das Modul `run_iv.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
#------ Output ---------------------------------------------------------#
return(c(tl_inp, tl_iv_hauptberechnung, tl_iv_mass, tl_iv_ergebnisse,
    tl_iv_varia, tl_iv_postprocessing, tl_iv_vafp))
```

4.1.3 Modul mod_iv_param_global.R

Dokumentation zuletzt aktualisiert am 10.10.2024

Das Modul zu den Ausgaben wird in `run_iv.R` wie folgt aufgerufen:[23]

```
tl_iv_param_global <- mod_iv_param_global(tl_inp)
```

`mod_iv_param_global.R` setzt default-Parameterwerte in PARAM_GLOBAL. Es gibt hier drei Arten von Parameterwerten. 1. Parameterwerte, die zur Ausführung des IV-Finanzperspektivenmodells spezifiziert sein müssen, aber auf die Berechnungen keinen Einfluss haben. Hierbei handelt es sich beispielsweise um Parameterwerte, die in Modulen verwendet werden, die sowohl vom IV-Finanzperspektivenmodell als auch vom AHV-Finanzperspektivenmodell aufgerufen werden, jedoch Modulteile betreffen, die nur für die Berechnungen des AHV-Finanzperspektivenmodells relevant sind.

2. Parameterwerte, für welche grundsätzlich ein Standardwert ("Default") existiert, für welche es aber in gewissen Fällen (bspw. Zwecks Backtestings oder Reproduktion von Berechnungen in der Vergangenheit) möglich sein soll, diese in der Datei `PARAM_GLOBAL.csv` anders zu wählen.

Als Erstes werden die Parameterwerte der 1. Art festgelegt:

```
# Versicherung (vz) auf IV setzten (für Module, die mit
# AHV, EO, und EL geteilt werden)
tl_inp$PARAM_GLOBAL$vz <- "iv"
# Folgende Parameter sind für IV nicht relevant, müssen
# aber gesetzt werden, da sie in den Vorberechnungen
# evaluiert werden
tl_inp$PARAM_GLOBAL$vz_iv <- FALSE
```

[23] Dieses Modul wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ implementiert.

```
tl_inp$PARAM_GLOBAL$vz_eo <- FALSE
tl_inp$PARAM_GLOBAL$popu_suisse <- FALSE
tl_inp$PARAM_GLOBAL$emigres <- FALSE
tl_inp$PARAM_GLOBAL$assures_facultatifs <- FALSE
tl_inp$PARAM_GLOBAL$frontaliers <- FALSE
tl_inp$PARAM_GLOBAL$saisonniers <- FALSE
tl_inp$PARAM_GLOBAL$bs_go_staf_jahr <- 2020
tl_inp$PARAM_GLOBAL$andere_einn_scen <- "scen_go"
tl_inp$PARAM_GLOBAL$param_chi_carre_prod <- FALSE
tl_inp$PARAM_GLOBAL$degrees_of_freedom <- 3
tl_inp$PARAM_GLOBAL$a <- 1
tl_inp$PARAM_GLOBAL$b <- 0.01
tl_inp$PARAM_GLOBAL$c <- 0.00035
tl_inp$PARAM_GLOBAL$ivschuld_scen <- "ivschuld_2024_08"  # Schuldenstands- und
↪  Rückzahlungsvektor für IV-Schuld bei der AHV. Wird nur für die Finanzperspektiven
↪  AHV verwendet.
```

Als nächstes werden die Parameterwerte 2. Art festgelegt. Hierbei erfolgt zuerst immer eine Prüfung, ob der entsprechende Parameterwert nicht schon in `tl_inp$PARAM_GLOBAL` vorhanden ist (was der Fall ist, wenn der entsprechende Parameter in `PARAM_GLOBAL.csv` spezifiziert wurde):

```
# letztes Abrechnungsjahr festlegen, falls nicht festgelegt
if (!"jahr_abr" %in% names(tl_inp$PARAM_GLOBAL)) {
    if (tl_inp$PARAM_GLOBAL$abr_prov == TRUE) {
        tl_inp$PARAM_GLOBAL$jahr_abr <- max(tl_inp$EO_ABRECHNUNG_PROV$jahr)
    } else {
        tl_inp$PARAM_GLOBAL$jahr_abr <- max(tl_inp$EO_ABRECHNUNG_DEF$jahr)
    }
}
# Preisbasis auf Abrechnungsjahr legen, falls nicht manuell
# gesetzt
if (!"jahr_preisbasis" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_preisbasis <- tl_inp$PARAM_GLOBAL$jahr_abr
}
# laufendes Jahr jahr_lj festlegen, falls nicht manuell
# gesetzt
if (!"jahr_lj" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_lj <- tl_inp$PARAM_GLOBAL$jahr_abr +
        1
}
# erstes Jahr, ab welchem Umfraagewerte fortgeschrieben
# werden sollen jahr_umfragen_fs festlegen, falls nicht
# manuell gesetzt
if (!"jahr_umfragen_fs" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_umfragen_fs <- max((tl_inp$UMFRAGEN %>%
        filter(umfrageid == tl_inp$PARAM_GLOBAL$id_umfragen))$jahr) +
        1
}
# sollen Werte aus all_felder_massdb verwendet werden? Nur
# für Backtesting relevant, da hier auf FALSE gesetztz,
# falls nicht manuell gesetzt
if (!"all_felder_massdb" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$all_felder_massdb <- FALSE
}
```

```
# Gewichtung der Preiskomponente im Mischindex auf 1
# setzten, falls nicht manuell gesetzt
if (!"gew_pi_comp" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$gew_pi_comp <- 1
}
# Gewichtung der Lohnkomponente im Mischindex auf 1
# setzten, falls nicht manuell gesetzt
if (!"gew_li_comp" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$gew_li_comp <- 1
}
# Rundung für die Minimalrente auf 5 Rappen setzen, falls
# nicht manuell gesetzt
if (!"minrente_rd" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$minrente_rd <- 5
}
# Jahr des Beginns des Resultatevektors auf 1997 setzten,
# falls nicht manuell gesetzt
if (!"jahr_beginn" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_beginn <- 1997
}
# Jahr des IK setzten (falls nicht manuell gesetzt)
if (!"jahr_ik" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_ik <- max(tl_inp$IK$jahr)
}
# Jahr des Rentenregisters setzten (falls nicht manuell
# gesetzt)
if (!"jahr_rr" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_rr <- max(tl_inp$IV_RR_RAM$an_rr)
}
# Minimumalter der Bevölkerung in den Daten setzten (falls
# nicht manuell gesetzt)
if (!"min_age" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$min_age <- 0
}
# Minimumalter der Bevölkerung in den Daten setzten (falls
# nicht manuell gesetzt)
if (!"min_age" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$min_age <- 0
}
# Maximumalter der Bevölkerung in den Daten setzten (falls
# nicht manuell gesetzt)
if (!"max_age" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$max_age <- 99
}
# Letztes beobachtetes Jahr für die Bevölkerung setzten
# (falls nicht manuell gesetzt)
if (!"jahr_bev" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_bev <- max(tl_inp$BEV_POP$jahr)
}
# Letztes beobachtetes Jahr für die Bevölkerung setzten
# (falls nicht manuell gesetzt)
if (!"jahr_bev" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_bev <- max(tl_inp$BEV_POP$jahr)
```

```
}
# Basisjahr für Bevölkerung auf letztes Beobachtungsjahr
# setzten (falls nicht manuell gesetzt)
if (!"jahr_bev_base" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_bev_base <- max(tl_inp$BEV_POP$jahr)
}
# Überprüfe, ob bev_ept angegeben wird (Wenn TRUE würde die
# EPT-Erwerbsbevölkerung als Bevölkerung verwendet)
if (!"bev_ept" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$bev_ept <- FALSE
}
# Überprüfe, ob ept_ept angegeben wird (Wenn FALSE würde
# die Wohnbevölkerung als Erwerbsbevölkerung verwendet)
if (!"ept_ept" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$ept_ept <- TRUE
}
# Der Parameter pi_eckwerte (Auswahl des Preisindexes)
# sollte 1 sein, jedoch könnte er (bspw. für Backtestings)
# auf 0 gesetzt werden.
if (!"pi_eckwerte" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$pi_eckwerte <- 1
}
# Überprüfe, ob feature switch parameter für 'mod_iv_mm'
# existiert.
if (!"mod_iv_mm" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$mod_iv_mm <- "NEW"
    warning("Parameter 'mod_iv_mm' ist in PARAM_GLOBAL nicht spezifiziert, 'NEW'
    ↪  angenommen")
}
# justierung=0 setzten für Modul 'justierung.R'
if (!"justierung" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$justierung <- 0
}
# Der Parameter bev_nurCH wird in mod_population.R
# verwendet. Grundsätzlich FALSE für IV-FHH.
if (!"bev_nurCH" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$bev_nurCH <- FALSE
}
# Der Parameter jahr_demo, also das erste Jahr, für welche
# Demografiedaten vorhanden sind, sollte 1971 sein.
if (!"jahr_demo" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$jahr_demo <- 1971
}
# Der Parameter min_age_cot, ist für den IV-FHH momentan
# irrelevant (wird aber frei gelassen, falls er bei
# Massnahmen verwendet werden soll)
if (!"min_age_cot" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$min_age_cot <- 21
}
# Der Parameter min_age_retraite, ist für den IV-FHH
# momentan irrelevant (wird aber frei gelassen, falls er
# bei Massnahmen verwendet werden soll)
if (!"min_age_retraite" %in% names(tl_inp$PARAM_GLOBAL)) {
```

```
    tl_inp$PARAM_GLOBAL$min_age_retraite <- 62
}
# Der Parameter max_age_retraite, ist für den IV-FHH
# momentan irrelevant (wird aber frei gelassen, falls er
# bei Massnahmen verwendet werden soll)
if (!"max_age_retraite" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$max_age_retraite <- 70
}
if ("jahr_mwst_fs" %in% names(tl_inp$PARAM_GLOBAL)) {
    warning(paste0("Parameter 'jahr_mwst_fs' wird nicht verwendet für MWSt-Fortschreibung
    ↪  beim Bundesbeitrag. Fortschreibung automatisch nach letztem Jahr von id_estv_iv
    ↪  ",
        tl_inp$PARAM_GLOBAL$id_estv_iv, " MWSt-Prognose."))
}
if (!"rentenmodell" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$rentenmodell <- "NEW"
    warning("Parameter 'rentenmodell' ist in PARAM_GLOBAL nicht spezifiziert, 'NEW'
    ↪  angenommen.")
}
if (!"he_modell" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_modell <- "NEW"
    warning("Parameter 'he_modell' ist in PARAM_GLOBAL nicht spezifiziert, 'NEW'
    ↪  angenommen.")
}
if (!"invalidisierung_szenario" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$invalidisierung_szenario <- "mittel"
    warning("Parameter 'invalidisierung_szenario' ist in PARAM_GLOBAL nicht spezifiziert,
    ↪  'mittel' angenommen.")
}
if (!"MA_years" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$MA_years <- 3
    warning("Parameter 'MA_years' ist in PARAM_GLOBAL nicht spezifiziert, 3 angenommen.")
}
if (!"fort_entw" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$fort_entw <- FALSE
    warning("Parameter 'fort_entw' ist in PARAM_GLOBAL nicht spezifiziert, FALSE
    ↪  angenommen.")
}
if (!"wachstum_lagyears" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$wachstum_lagyears <- 5
    warning("Parameter 'wachstum_lagyears' ist in PARAM_GLOBAL nicht spezifiziert, 5
    ↪  angenommen (nur relevant, falls fort_entw==TRUE ist).")
}
if (!"fort_entw_years" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$fort_entw_years <- 3
    warning("Parameter 'fort_entw_years' ist in PARAM_GLOBAL nicht spezifiziert, 3
    ↪  angenommen (nur relevant, falls fort_entw==TRUE ist).")
}
if (!"mitAHV21" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$mitAHV21 <- TRUE
    warning("Parameter 'mitAHV21' ist in PARAM_GLOBAL nicht spezifiziert, TRUE
    ↪  angenommen.")
}
```

```
if (!"szenariolag" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$szenariolag <- 5
    warning("Parameter 'szenariolag' ist in PARAM_GLOBAL nicht spezifiziert, 5
    ↪  angenommen.")
}
if (!"he_MA_years" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_MA_years <- 3
    warning("Parameter 'he_MA_years' ist in PARAM_GLOBAL nicht spezifiziert, 3
    ↪  angenommen.")
}
if (!"he_fort_entw" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_fort_entw <- FALSE
    warning("Parameter 'he_fort_entw' ist in PARAM_GLOBAL nicht spezifiziert, FALSE
    ↪  angenommen.")
}
if (!"he_wachstum_lagyears" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_wachstum_lagyears <- 5
    warning("Parameter 'he_wachstum_lagyears' ist in PARAM_GLOBAL nicht spezifiziert, 5
    ↪  angenommen (nur relevant, falls he_fort_entw==TRUE ist).")
}
if (!"he_fort_entw_years" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_fort_entw_years <- 3
    warning("Parameter 'he_fort_entw_years' ist in PARAM_GLOBAL nicht spezifiziert, 3
    ↪  angenommen (nur relevant, falls he_fort_entw==TRUE ist).")
}
if (!"he_mitAHV21" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_mitAHV21 <- TRUE
    warning("Parameter 'he_mitAHV21' ist in PARAM_GLOBAL nicht spezifiziert, TRUE
    ↪  angenommen.")
}
if (!"he_szenario" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_szenario <- FALSE
    warning("Parameter 'he_szenario' ist in PARAM_GLOBAL nicht spezifiziert, 'FALSE'
    ↪  angenommen, d.h., Szenarien werden nicht auf HE angewendet.")
}
if (!"he_szenariolag" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$he_szenariolag <- 5
    warning("Parameter 'he_szenariolag' ist in PARAM_GLOBAL nicht spezifiziert, 5
    ↪  angenommen.")
}
if (!"beitragsmodell" %in% names(tl_inp$PARAM_GLOBAL)) {
    tl_inp$PARAM_GLOBAL$beitragsmodell <- "OLD"
    warning("Parameter 'beitragsmodell' ist in PARAM_GLOBAL nicht spezifiziert, OLD
    ↪  angenommen.")
}

# Überprüfe den Wert von invalidisierung_szenario und setze
# die Untertitel, sowie renten_szenario (letzteres
# entfernen, sobald altes Rentenmodell ausgepflegt wird)
if (tl_inp$PARAM_GLOBAL$invalidisierung_szenario == "mittel") {
    tl_inp$PARAM_GLOBAL$renten_szenario <- 0
    tl_inp$PARAM_GLOBAL$subtitle_d <- "Mittleres Szenario"
    tl_inp$PARAM_GLOBAL$subtitle_f <- "Scénario moyen"
```

```
    tl_inp$PARAM_GLOBAL$subtitle_i <- "Scenario medio"

} else if (tl_inp$PARAM_GLOBAL$invalidisierung_szenario == "hoch") {
    tl_inp$PARAM_GLOBAL$renten_szenario <- 1
    tl_inp$PARAM_GLOBAL$subtitle_d <- "Hohes Szenario"
    tl_inp$PARAM_GLOBAL$subtitle_f <- "Scénario haut"
    tl_inp$PARAM_GLOBAL$subtitle_i <- "Scenario elevato"
} else if (tl_inp$PARAM_GLOBAL$invalidisierung_szenario == "tief") {
    tl_inp$PARAM_GLOBAL$renten_szenario <- 2
    tl_inp$PARAM_GLOBAL$subtitle_d <- "Tiefes Szenario"
    tl_inp$PARAM_GLOBAL$subtitle_f <- "Scénario bas"
    tl_inp$PARAM_GLOBAL$subtitle_i <- "Scenario basso"
}

# Massnahmen-Parameterliste erstellen und zu PARAM_ALL
# hinzufügen
if (tl_inp$PARAM_GLOBAL$flag_param_massn) {
    if (!is.na(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)) {
        if (length(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int) >
            0) {
            mass_name <- strsplit(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int,
                ",")
            if ("iv_remb_dette_res_exploit" %in% mass_name[[1]]) {
                tl_inp$PARAM_GLOBAL$reb_res = TRUE
            }
        }
    }
}
```

Der letzte Codeblock dient dazu, den Parameter `reb_res` auf `TRUE` zu setzten, falls die Massnahme `iv_remb_dette_res_exploit` (also die Massnahme, dass alle allfälligen Zukünftigen Einnahmeüberschüsse zur Schuldenrückzahlung verwendet werden) spezifiziert wurde. Dies ist notwendig, da in dem Fall bei der Berechnung der IV-Bilanz Änderungen notwendig sind. Konkret wird in einem ersten Schritt mit `if(tl_inp$PARAM_GLOBAL$flag_param_massn)` überprüft, ob Massnahmen überhaupt berücksichtigt werden sollen. Danach wird überprüft, ob Massnahmen spezifiziert sind, und wenn dies der Fall ist (`!is.na(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)`, und mehr als eine Massnahme spezifiziert ist (`length(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)>0`), werden die Massnahmen als Liste in `mass_name` abgespeichert. Die letzte Zeile überprüft, ob die Massnahme `iv_remb_dette_res_exploit` spezifiziert ist, und setzt den Parameter `tl_inp$PARAM_GLOBAL$reb_res = TRUE`, falls dies der Fall ist.

Damit kann das ergänzte `PARAM_GLOBAL`-Dataframe an `run_iv.R` zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
PARAM_GLOBAL <- tl_inp$PARAM_GLOBAL

#------ Output --------------------------------------------------------#
mod_return(PARAM_GLOBAL)
```

4.1.4 Modul wrap_iv_vorb_berechn.R

Dokumentation zuletzt aktualisiert am 07.10.2024

Das Modul `wrap_iv_vorb_berechn.R` führt Vorberechnungen für Projektionsvariablen durch, die ausschliesslich im IV-Finanzperspektivenmodell verwendet werden. Dies stellt die Abgrenzung zum weiter oben

erläuterten Modul `wrap_vorb_berechn.R` (vgl. Kapitel 4.1.1) dar, welches nicht nur in den Finanzperspektiven der IV verwendet wird, sondern bspw. auch im Finanzperspektivenmodell der AHV aufgerufen wird.

Das Modul zu den IV-spezifischen Vorberechnungen wird in `wrap_iv.R` wie folgt aufgerufen:

```
#------ Vorbereitende Berechnungen iv ---------------------------------#
tl_vorb_iv <- wrap_iv_vorb_berechn(tl_inp = tl_inp)
```

Zu Beginn wird die folgende Überprüfung durchgeführt:

```
#------ Zusätzliche Überprüfung während Einführung neues Rentemodells (kann entfernt
↪  werden, wenn alle input-container die neuen Rentemodell das IV_ABGAENGE dataframe
↪  enthalten) ---#
# Stelle sicher, dass tl_inp das IV_ABGAENGE dataframe
# enthält, falls nicht vorhanden
required_dfs <- c("IV_ABGAENGE", "IV_RENTEN_NEW")

for (df_name in required_dfs) {
    if (!df_name %in% names(tl_inp)) {
        # Erstelle einen leeren Dataframe
        tl_inp[[df_name]] <- data.frame()
    }
}
```

Das nachfolgende Modul diente der Datenaufbereitung für gewisse Massnahmenarten. Es wird im Finanzperspektivenmodell 2024 nicht verwendet, weshalb hier nicht weiter darauf eingegangen wird:

```
#--- Inputs Handling ------------------------------------------------#
tl_iv_rr_ram_handling <- mod_iv_rr_ram_handling(IV_RR_RAM = tl_inp$IV_RR_RAM)
```

Die nachfolgenden zwei Module dienen der Berechnung der Invalidisierung und Abgangsraten im alten Rentenmodell. Sie sind zur Reproduzierbarkeit von Vorjahresmresultaten noch im Code enthalten, werden aber im Finanzperspektivenmodell 2024 nicht verwendet, weshalb hier nicht weiter darauf eingegangen wird:

```
#------ Invalidisierungen OLD-----------------------------------------------#
tl_iv_ixy <- mod_iv_ixy(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL, IV_GRUNDL_FAKTOR =
↪  tl_inp$IV_GRUNDL_FAKTOR,
    IV_GRUNDLAGEN_IX = tl_inp$IV_GRUNDLAGEN_IX)

#------ Invalidisierungen und Abgänge (NEW Modell) ------------------#
tl_iv_raten_invalidisierung_abgaenge <-
↪  mod_iv_raten_invalidisierung_abgaenge(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_NEURENTEN = tl_inp$IV_NEURENTEN, IV_ABGAENGE = tl_inp$IV_ABGAENGE,
    IV_RENTEN_DWH = tl_inp$IV_RENTEN_DWH, BEVOELKERUNG = tl_inp$BEVOELKERUNG)
```

Das nachfolgende Modul berechnet den Sozialversicherungs-Beitragssatz für sämtliche Lohnbeiträge (inklusive Arbeitslosenversicherung). Dieser Satz wird verwendet, um später bei den Ausgaben den Arbeitgeberanteil bei den Lohnbeiträgen, welcher bei den Taggeldern von der IV bezahlt wird, zu berechnen:

```
#------ Beitragssätze der Sozialversicherungen ------------------------#
tl_sv_satz <- mod_sv_satz(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    SV_BEITRAGSSATZ = tl_inp$SV_BEITRAGSSATZ)
```

Das Modul `mod_sv_satz` ist in Kapitel 4.31 beschrieben.

Die nachfolgenden zwei Module sind wiederum nur für das alte Rentenmodell relevant. Sie sind zur Reproduzierbarkeit von Vorjahresmresultaten noch im Code enthalten, werden aber im Finanzperspektivenmodell

2024 nicht verwendet, weshalb hier nicht weiter darauf eingegangen wird:

```
#------ Wachsum der Renten ----------------------------------------#

  if(tl_inp$PARAM_GLOBAL$rentenmodell == "OLD"){
      tl_iv_rentenentwicklung <- mod_iv_rentenbestand(
          PARAM_GLOBAL       = tl_inp$PARAM_GLOBAL
          , BEVOELKERUNG       = tl_inp$BEVOELKERUNG
          , IV_RENTEN_DWH      = tl_inp$IV_RENTEN_DWH
          , IV_NEURENTEN       = tl_inp$IV_NEURENTEN
          , IXY                = tl_iv_ixy$IXY
      )
  }
# ---- nachfolgendes Modul. inklusive rentenmodell_horizont wird nicht gebraucht. Bei
↪  Cleanup löschen (beachten dass IV_RATEN_IVALIDISIERUNG_ABGAENGE auch gelöscht werden
↪  muss!)
  else{
      tl_inp$PARAM_GLOBAL$rentenmodell_horizont <- tl_inp$PARAM_GLOBAL$MA_years

      tl_iv_rentenentwicklung <- mod_iv_rentenbestand_NEW(
          PARAM_GLOBAL       = tl_inp$PARAM_GLOBAL
          , BEVOELKERUNG       = tl_inp$BEVOELKERUNG
          , IV_RENTEN_DWH      = tl_inp$IV_RENTEN_DWH
          , IV_NEURENTEN       = tl_inp$IV_NEURENTEN
          , IV_RATEN_INVALIDISIERUNG_ABGAENGE                =
↪  tl_iv_raten_invalidisierung_abgaenge$IV_RATEN_INVALIDISIERUNG_ABGAENGE
      )
  }
```

Das nachfolgende Modul wählt die in `PARAM_GLOBAL` spezifizierten Varianten für die MwSt-Einnahmeprojektion (welche später für die Berechnung des Bundesbeitrages benötigt wird), sowie die Variante für die Umfrage-basierten Ausgabeprojektionen aus:

```
#------ Filtern der Szenarien ----------------------------------------#
tl_iv_input <- mod_iv_filter_inp(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    UMFRAGEN = tl_inp$UMFRAGEN, ESTV_IV = tl_inp$ESTV_IV)
```

Das Modul `mod_iv_filter_inp` ist in Kapitel 4.32 beschrieben.

Das nächste Modul berechnet die Fortschreibungs-Vektore, also die Projektionen für die Preisentwicklung, Lohnentwicklung, die Entwicklung der Lohnsumme, sowie die Entwicklung des Bundesbeitrages:

```
#------ Fortschreibungswerte ----------------------------------------#
tl_iv_fortschreibung <- mod_iv_fortschreibung(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_inp$ECKWERTE_EXTENDED, EINK_ENTWICKLUNG =
    ↪  tl_inp$EINK_ENTWICKLUNG,
    AKTIVE_BEV = tl_inp$AKTIVE_BEV, IK = tl_inp$IK, IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG,
    DISKONTFAKTOR = tl_inp$DISKONTFAKTOR, ESTV_IV_SCEN = tl_iv_input$ESTV_IV_SCEN,
    RENTENINDEX_ZR = tl_inp$RENTENINDEX_ZR)
```

Das Modul `mod_iv_fortschreibung` ist in Kapitel 4.33 beschrieben.

Zum Schluss werden die Berechnungen an das Finanzperspektivenmodell (respektive das Modul `wrap_iv.R`) zurückgegeben. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
#------ Output ----------------------------------------------------------#
```

```
return(c(tl_iv_rr_ram_handling, tl_iv_ixy, tl_iv_raten_invalidisierung_abgaenge,
    tl_sv_satz, tl_iv_rentenentwicklung, tl_iv_input, tl_iv_fortschreibung))
```

4.1.5 Modul wrap_iv_hauptberechnung.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Modul zu den Hauptberechnungen wird in `wrap_iv.R` wie folgt aufgerufen:

```
tl_iv_hauptberechnung <- wrap_iv_hauptberechnung(tl_inp = tl_inp)
```

Zu Beginn wird die folgende Überprüfung durchgeführt:

```
#------ Zusätzliche Überprüfung während Einführung neues Modell für medizinische
↪  Massnahmen (kann entfernt werden, wenn alle input-container die drei MM-dataframes
↪  unter 'required_dfs' enthalten) ---#

# Stelle sicher, dass tl_inp die erforderlichen Dataframes
# als leere Dataframes enthält, falls nicht vorhanden
required_dfs <- c("IV_MED_MASSN_REGISTER", "IV_MED_MASSN_SONDEREFFEKTE",
    "IV_MED_MASSN_PARAMCHANGES")

for (df_name in required_dfs) {
    if (!df_name %in% names(tl_inp)) {
        # Erstelle einen leeren Dataframe
        tl_inp[[df_name]] <- data.frame()
    }
}
```

Danach wird das Modul zur Berechnung der Ausgabeprojektionen aufgerufen (vgl. Kapitel 4.1.6):

```
tl_iv_ausgaben <- mod_iv_ausgaben(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEVOELKERUNG = tl_inp$BEVOELKERUNG, IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG,
    BESTAND_IV = tl_inp$BESTAND_IV, IV_HE_DWH = tl_inp$IV_HE_DWH,
    FORTSCHREIBUNG_IV = tl_inp$FORTSCHREIBUNG_IV, RENTENENTWICKLUNG =
    ↪  tl_inp$RENTENENTWICKLUNG,
    SV_SATZ = tl_inp$SV_SATZ, UMFRAGE_IV = tl_inp$UMFRAGE_IV,
    IV_MED_MASSN_REGISTER = tl_inp$IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
    ↪  tl_inp$IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = tl_inp$IV_MED_MASSN_PARAMCHANGES,
    DISKONTFAKTOR = tl_inp$DISKONTFAKTOR, IV_RENTEN_NEW = tl_inp$IV_RENTEN_NEW,
    IV_HE_NEW = tl_inp$IV_HE_NEW, IV_HE_KINDER = tl_inp$IV_HE_KINDER)
```

Darauffolgend wird das Modul zur Berechnung der Einnahmeprojektionen aufgerufen (vgl. Kapitel 4.1.10):

```
tl_iv_einnahmen <- mod_iv_einnahmen(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    AKTIVE_BEV = tl_inp$AKTIVE_BEV, IK = tl_inp$IK, ESTV = tl_inp$ESTV,
    ESTV_IV_SCEN = tl_inp$ESTV_IV_SCEN, EINK_ENTWICKLUNG = tl_inp$EINK_ENTWICKLUNG,
    IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, AUSGABEN_IV = tl_iv_ausgaben$AUSGABEN_IV,
    RENTENINDEX_ZR = tl_inp$RENTENINDEX_ZR, MWST_SATZ_IV = tl_inp$MWST_SATZ_IV,
    ZINS = tl_inp$ZINS, DISKONTFAKTOR = tl_inp$DISKONTFAKTOR,
    POP_SCENARIO_EPT_FULL = tl_inp$POP_SCENARIO_EPT_FULL, INDICES_FUTURS =
    ↪  tl_inp$INDICES_FUTURS,
    ECKWERTE = tl_inp$ECKWERTE, AHV_ABREECHNUNG_DEF = tl_inp$AHV_ABRECHNUNG_DEF)
```

Zum Schluss werden die Ausgaben- und Einnahmeprojektionen an das Finanzperspektivenmodell (respek-

tive das Modul `wrap_iv.R`) zurückgegeben. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
#------ Output ------------------------------------------------------------#
return(c(tl_iv_ausgaben, tl_iv_einnahmen))
```

4.1.6 Modul mod_iv_ausgaben.R

Dokumentation zuletzt aktualisiert am 30.08.2024

Das Modul zu den Ausgaben wird in `wrap_iv_hauptberechnung.R` wie folgt aufgerufen:

```
tl_iv_ausgaben <- mod_iv_ausgaben(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEVOELKERUNG = tl_inp$BEVOELKERUNG, IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG,
    BESTAND_IV = tl_inp$BESTAND_IV, IV_HE_DWH = tl_inp$IV_HE_DWH,
    FORTSCHREIBUNG_IV = tl_inp$FORTSCHREIBUNG_IV, RENTENENTWICKLUNG =
    ↪  tl_inp$RENTENENTWICKLUNG,
    SV_SATZ = tl_inp$SV_SATZ, UMFRAGE_IV = tl_inp$UMFRAGE_IV,
    , IV_MED_MASSN_REGISTER = tl_inp$IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
↪   tl_inp$IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = tl_inp$IV_MED_MASSN_PARAMCHANGES,
    DISKONTFAKTOR = tl_inp$DISKONTFAKTOR, IV_RENTEN_NEW = tl_inp$IV_RENTEN_NEW,
    IV_HE_NEW = tl_inp$IV_HE_NEW, IV_HE_KINDER = tl_inp$IV_HE_KINDER)
```

Als erstes wird das Modul zur Berechnung der Geldleistungen der IV aufgerufen (vgl. Kapitel 4.1.7):

```
#------ Geldleistungen / Prestations en espèces ------------------------#
tl_iv_geldleistungen <- mod_iv_geldleistungen(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BESTAND_IV = BESTAND_IV, IV_HE_DWH = IV_HE_DWH, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV,
    RENTENENTWICKLUNG = RENTENENTWICKLUNG, SV_SATZ = SV_SATZ,
    UMFRAGE_IV = UMFRAGE_IV, IV_RENTEN_NEW = IV_RENTEN_NEW, IV_HE_NEW = IV_HE_NEW,
    IV_HE_KINDER = IV_HE_KINDER)
```

Als nächstes wird das Modul zur Berechnung der Sachleistungen der IV aufgerufen (vgl. Kapitel 4.1.8):

```
#------ Kosten für ind. Massnahmen / Frais pour mesures individuelles -#
tl_iv_sachleistungen <- mod_iv_sachleistung(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV,
    IV_MED_MASSN_REGISTER = IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
    ↪  IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = IV_MED_MASSN_PARAMCHANGES, DISKONTFAKTOR = DISKONTFAKTOR,
    BEVOELKERUNG = BEVOELKERUNG, IV_GELDLEISTUNG = tl_iv_geldleistungen$IV_GELDLEISTUNG)
```

Danach wird das Modul aufgerufen, das alle anderen verbleibenden Ausgaben berechnet (vgl. Kapitel 4.1.9):

```
#------ Übrige Ausgaben ------------------------------------------------#
tl_iv_uebrigeausgaben <- mod_iv_uebrigeausgaben(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_GELDLEISTUNG = tl_iv_geldleistungen$IV_GELDLEISTUNG, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV,
    DISKONTFAKTOR = DISKONTFAKTOR)
```

Als nächstes wird das Dataframe `AUSGABEN_IV` berechnet, welches sämtliche Ausgabenpositionen ausser den Schuldzinsen enthält:

```
#------ Projektion Ausgaben total (ohne Schuldzins) -------------------#
AUSGABEN_IV <- tl_iv_geldleistungen$IV_GELDLEISTUNG %>%
    inner_join(tl_iv_sachleistungen$SACHLEISTUNG_IV, by = c("jahr")) %>%
    inner_join(tl_iv_uebrigeausgaben$UEBRIGE_AUSGABEN_IV, by = c("jahr")) %>%
    mutate(across(.cols = everything(), ~if_else(is.na(.), 0,
        .), .names = "{.col}")) %>%
    mutate(aus_tot_iv = geld_leist_iv + ind_mass_tot_iv + btr_inst_org_iv +
        df_kost_iv + verw_aufw_iv + a_aufw)
```

Der nächste Codeteil wird hier nicht weiter erläutert, da Stand 2024 im geltenden Finanzperspektivenmodell `PARAM_GLOBAL$all_felder_massdb==FALSE` ist:

```
#------ nicht mehr gebuchte Kosten -----------------------------------#
#------ Zinsen / Intérêts débiteurs du capital
#------ Überweisung Betriebsrech-nung AHV / Transfert du compte d'exploitation AVS #
#------ Zinsbelastung IV, Anteil EO / Intérêts à charge de l'AI, part APG #
#------ Zinsen auf Kapitalhilfe  / Intérêts d'aides en capital --------#
# vor Einführung eines eigenständigen IV-Fonds wurden das
# Negativsaldo der IV innerhalb des gemeinsamen Fonds
# geführt. Ab 2011 wurde `zins` durch `schzins` ersetzt.
# Eine Zinszahlung an EO entfällt, da die Verbindlichkeit
# neu explizit gegenüber der AHV besteht.
if (PARAM_GLOBAL$all_felder_massdb) {
    ABR_MASSDB <- IV_ABRECHNUNG %>%
        select(jahr, zins, ueberw_btr_ahv, zins_bel_ant_eo, zins_ein_kap_hilf) %>%
        rbind(data.frame(jahr = c((PARAM_GLOBAL$jahr_abr + 1):PARAM_GLOBAL$jahr_ende))
        ↪  %>%
            mutate(zins = 0, ueberw_btr_ahv = 0, zins_bel_ant_eo = 0,
                zins_ein_kap_hilf = 0)) %>%
        mutate(zins = ueberw_btr_ahv + zins_bel_ant_eo + zins_ein_kap_hilf)

    AUSGABEN_IV <- AUSGABEN_IV %>%
        inner_join(ABR_MASSDB, by = c("jahr")) %>%
        mutate(aus_tot_iv = aus_tot_iv + zins)
}
```

Zum Schluss werden die Berechnungen an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_hauptberechnung.R`) zurückgegeben. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
#------ Modellparameter von Modellierung medizinische Massnahmen ------#
IV_MED_MASSN_FORECAST_AVERAGES <- tl_iv_sachleistungen$IV_MED_MASSN_FORECAST_AVERAGES

#------ Output -------------------------------------------------------#
if (PARAM_GLOBAL$rentenmodell == "OLD") {
    mod_return(AUSGABEN_IV, IV_MED_MASSN_FORECAST_AVERAGES)
} else {
    BESTAND_IV <- tl_iv_geldleistungen$BESTAND_IV
    RENTENBESTAND_IV <- tl_iv_geldleistungen$RENTENBESTAND_IV
    mod_return(AUSGABEN_IV, IV_MED_MASSN_FORECAST_AVERAGES, BESTAND_IV,
        RENTENBESTAND_IV)
}
```

Der obere Codeteil liest das Dataframe `IV_MED_MASSN_FORECAST_AVERAGES` aus. Die if-Bedingung im unte-

ren Codeteil dient dazu zu unterscheiden, ob die Dataframes `BESTAND_IV` und `RENTENBESTAND_IV` mit den im Geldleistungen-Modul berechneten Dataframe ersetzt werden sollen oder nicht. Dies ist nur der Fall, wenn nicht das alte Rentenmodell (`PARAM_GLOBAL$rentenmodell ==` "OLD ") ausgewählt wurde. Im alten Rentenmodell werden die Dataframes `BESTAND_IV` und `RENTENBESTAND_IV` in den vorbereitenden Berechnungen berechnet (vgl. Kapitel 4.1.4), während die Berechnung dieser Dataframes im neuen Rentenmodell im Modul `mod_iv_geldleistungen` erfolgt (vgl. Kapitel 4.1.7).

4.1.7 Modul mod_iv_geldleistungen.R

Dokumentation zuletzt aktualisiert am 30.08.2024

Das Modul zur Berechnung der Ausgaben für Geldleistungen wird in `mod_iv_ausgaben.R` wie folgt aufgerufen:

```
tl_iv_geldleistungen <- mod_iv_geldleistungen(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BESTAND_IV = BESTAND_IV, IV_HE_DWH = IV_HE_DWH, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV,
    RENTENENTWICKLUNG = RENTENENTWICKLUNG, SV_SATZ = SV_SATZ,
    UMFRAGE_IV = UMFRAGE_IV, IV_RENTEN_NEW = IV_RENTEN_NEW, IV_HE_NEW = IV_HE_NEW,
    IV_HE_KINDER = IV_HE_KINDER)
```

Als erstes wird das Modul zur Berechnung der Rentenausgaben aufgerufen, wobei auch die Möglichkeit besteht, das alte Rentenmodell auszuführen. Stand 2024 wird jedoch das neue Rentenmodell verwendet (vgl. Kapitel 4.2):

```
#------ Renten ordentliche und ao / Rentes (ordinaires et exo.) -------#
if (PARAM_GLOBAL$rentenmodell == "OLD") {
    tl_iv_rentensumme <- mod_iv_rentensumme(PARAM_GLOBAL = PARAM_GLOBAL,
        IV_ABRECHNUNG = IV_ABRECHNUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
        BESTAND_IV = BESTAND_IV)
} else {
    tl_iv_rentensumme <- mod_iv_rentensumme_NEW(PARAM_GLOBAL = PARAM_GLOBAL,
        IV_ABRECHNUNG = IV_ABRECHNUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
        BEVOELKERUNG = BEVOELKERUNG, IV_RENTEN_NEW = IV_RENTEN_NEW)

}
```

Als nächstes wird das Modul zur Berechnung der Hilflosenentschädigung aufgerufen, wobei auch hier die Möglichkeit besteht, das alte Hilflosenentschädigungsmodell auszuführen. Stand 2024 wird jedoch das neue Hilflosenentschädigungsmodell verwendet (vgl. Kapitel 4.4):

```
#------ Hilflosenentschädigungen / Allocations pour impotents ---------#
if (PARAM_GLOBAL$he_modell == "OLD") {
    tl_iv_he <- mod_iv_he(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
        IV_HE_DWH = IV_HE_DWH, BEVOELKERUNG = BEVOELKERUNG, RENTENENTWICKLUNG =
        ↪  RENTENENTWICKLUNG)
} else {
    tl_iv_he <- mod_iv_he_NEW(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
        BEVOELKERUNG = BEVOELKERUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
        IV_HE_NEW = IV_HE_NEW, IV_HE_KINDER = IV_HE_KINDER)
}
```

Als nächstes wird das Modell für Berechnung der Ausgaben für Taggelder und des Arbeitgeberanteils der Lohnbeiträge (welche bei den Taggelder anfallen und von der IV bezahlt werden) aufgerufen (vgl. Kapitel 4.3):

```
#------ Taggelder / Indemnités journalières ---------------------------#
tl_iv_taggelder <- mod_iv_taggelder(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)

#------ Beitragsanteil der IV / Part de cotisations à charge de l'AI --#
tl_iv_beitragsanteil_iv <- mod_iv_beitragsanteil_iv(PARAM_GLOBAL = PARAM_GLOBAL,
    TAGGELDER_IV = tl_iv_taggelder$TAGGELDER_IV, IV_ABRECHNUNG = IV_ABRECHNUNG,
    SV_SATZ = SV_SATZ)
```

Zum Schluss wird ein Dataframe gebildet, das die projizierten Ausgabenvektoren für alle Geldleistungen enthält. Dieser Ausgabenvektor wird dann an das Finanzperspektivenmodell, respektive das Modul `mod_iv_ausgaben`, zurückgegeben. Falls das neue Rentenmodell ausgewählt wurde werden zudem noch die Dataframes `BESTAND_IV` und `RENTENBESTAND_IV` hinzugefügt:

```
#------ Alle Geldleistungen ------------------------------------------#
IV_GELDLEISTUNG <- tl_iv_rentensumme$RENTENSUMME_IV %>%
    inner_join(tl_iv_he$HE_IV, by = c("jahr")) %>%
    inner_join(tl_iv_taggelder$TAGGELDER_IV, by = c("jahr")) %>%
    inner_join(tl_iv_beitragsanteil_iv$IV_BEITRAGSANTEIL, by = c("jahr")) %>%
    mutate(geld_leist_iv = rentensumme_iv + he_iv + tg_geld +
        btr_ant_iv)

#------ Output -------------------------------------------------------#
if (PARAM_GLOBAL$rentenmodell == "OLD") {
    mod_return(IV_GELDLEISTUNG)
} else {
    BESTAND_IV <- tl_iv_rentensumme$BESTAND_IV
    RENTENBESTAND_IV <- tl_iv_rentensumme$RENTENBESTAND_IV
    mod_return(IV_GELDLEISTUNG, BESTAND_IV, RENTENBESTAND_IV)
}
```

4.1.8 Modul mod_iv_sachleistung.R

Dokumentation zuletzt aktualisiert am 30.08.2024

Das Modul zur Berechnung der Ausgaben für Sachleistungen wird in `mod_iv_ausgaben.R` wie folgt aufgerufen:

```
#------ Kosten für ind. Massnahmen / Frais pour mesures individuelles -#
tl_iv_sachleistungen <- mod_iv_sachleistung(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV,
    IV_MED_MASSN_REGISTER = IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
    ↪  IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = IV_MED_MASSN_PARAMCHANGES, DISKONTFAKTOR = DISKONTFAKTOR,
    BEVOELKERUNG = BEVOELKERUNG, IV_GELDLEISTUNG = tl_iv_geldleistungen$IV_GELDLEISTUNG)
```

Als erstes wird das Modul zur Berechnung der Ausgaben für medizinische Massnahmen aufgerufen, wobei auch die Möglichkeit besteht, das alte Modell auszuführen. Stand 2024 wird jedoch das neue Modell für medizinische Massnahmen verwendet (vgl. Kapitel 4.5):

```
#------ Medizinische Massnahmen / Mesures médicales ------------------#

if (PARAM_GLOBAL$mod_iv_mm == "NEW") {
    tl_iv_mm <- mod_iv_mm_new(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
```

```
        IV_MED_MASSN_REGISTER = IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
        ↪  IV_MED_MASSN_SONDEREFFEKTE,
        IV_MED_MASSN_PARAMCHANGES = IV_MED_MASSN_PARAMCHANGES,
        DISKONTFAKTOR = DISKONTFAKTOR, BEVOELKERUNG = BEVOELKERUNG)
} else {
    tl_iv_mm <- mod_iv_mm(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
        IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
    # Leeres dataframe mit Modell-Vorhersagedurchschnitten
    # erstellen, damit Rest des Codes konsistent ist.
    tl_iv_mm[["IV_MED_MASSN_FORECAST_AVERAGES"]] <- data.frame()

}
```

Danach werden nacheinander die Module, die alle anderen Ausgaben für Sachleistungen berechnen, aufgerufen:

```
#------ Frühinterventionsmassnahmen / Mesures d'intervention précoce --#
tl_iv_fi <- mod_iv_fi(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)

#------ Beratung und Begleitung / Conseils et suivi --------------------#
tl_iv_ber_begl <- mod_iv_ber_begl(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ Integrationsmassnahmen / Mesures de réinsertion ----------------#
tl_iv_im <- mod_iv_im(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)

#------ Massnahmen beruflicher Art / Mesures d'ordre professionnel ----#
tl_iv_mba <- mod_iv_mba(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)

#------ And. Kosten beruf. Einglied. / Autres coûts de réadap. profes.-#
tl_iv_aus_uebr <- mod_iv_aus_uebr(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ Hilfsmittel / Moyens auxiliaires -------------------------------#
tl_iv_hm <- mod_iv_hm(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, BEVOELKERUNG = BEVOELKERUNG,
    DISKONTFAKTOR = DISKONTFAKTOR)

#------ Reisekosten / Frais de voyage ----------------------------------#
tl_iv_rk <- mod_iv_rk(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)

#------ Assistenzbeitrag / Contribution d'assistance -------------------#
tl_iv_assb <- mod_iv_assb(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, IV_GELDLEISTUNG = IV_GELDLEISTUNG)

#------ Rück. für individ. Massnahmen / Mesures individ. restituer ----#
tl_iv_rueck_im <- mod_iv_rueck_im(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Die obigen Module werden in Kapitel 4.6 bis 4.14 genauer beschrieben.

Zum Schluss werden alle Sachleistungen in einem Dataframe zusammengefasst:

```
# Bemerkung für die jahre vor 2012 fehlt der Betrag von
# btr_sond_s (Beiträge Sonderschulung und hilflose
# Minderjährige)
SACHLEISTUNG_IV <- tl_iv_mm$IV_MM %>%
    inner_join(tl_iv_fi$IV_FI, by = c("jahr")) %>%
    inner_join(tl_iv_im$IV_IM, by = c("jahr")) %>%
    inner_join(tl_iv_ber_begl$IV_BER_BEGL, by = c("jahr")) %>%
    inner_join(tl_iv_mba$IV_MBA, by = c("jahr")) %>%
    inner_join(tl_iv_aus_uebr$IV_AUS_UEBR, by = c("jahr")) %>%
    inner_join(tl_iv_hm$IV_HM, by = c("jahr")) %>%
    inner_join(tl_iv_rk$IV_RK, by = c("jahr")) %>%
    inner_join(tl_iv_assb$IV_ASSB, by = c("jahr")) %>%
    inner_join(tl_iv_rueck_im$IV_RUECK_IM, by = c("jahr")) %>%
    mutate(ind_mass_tot_iv = mm + fi + im + ber_begl + mba +
        aus_uebr + hm + reise_kost + btr_ass + rueck_erstattungsf_ind)
```

Der nächste Codeteil wird hier nicht weiter erläutert, da Stand 2024 im geltenden Finanzperspektivenmodell `PARAM_GLOBAL$all_felder_massdb==FALSE` ist:

```
#------ nicht mehr gebuchte Kosten ------------------------------------#
#------ Beiträge Sonderschulung / Subsides formation scolaire ---------#

if (PARAM_GLOBAL$all_felder_massdb) {
    ABR_MASSDB <- IV_ABRECHNUNG %>%
        select(jahr, btr_sond_s) %>%
        rbind(data.frame(jahr = c((PARAM_GLOBAL$jahr_abr + 1):PARAM_GLOBAL$jahr_ende))
        ↪  %>%
            mutate(btr_sond_s = 0))

    SACHLEISTUNG_IV <- SACHLEISTUNG_IV %>%
        inner_join(ABR_MASSDB, by = c("jahr")) %>%
        mutate(ind_mass_tot_iv = ind_mass_tot_iv + btr_sond_s)
}
```

Zum Schluss wird noch das Dataframe `IV_MED_MASSN_FORECAST_AVERAGES` ausgelesen, bevor es zusammen mit dem Dataframe `SACHLEISTUNG_IV` and das Finanzperspektivenmodell, respektive das Modul `mod_iv_ausgaben`, zurückgegeben wird:

```
-#------ Modellparameter von Modellierung medizinische Massnahmen ------#
  IV_MED_MASSN_FORECAST_AVERAGES <- tl_iv_mm$IV_MED_MASSN_FORECAST_AVERAGES


#------ Output --------------------------------------------------------#
  mod_return(SACHLEISTUNG_IV, IV_MED_MASSN_FORECAST_AVERAGES)
```

4.1.9 Modul mod_iv_uebrigeausgaben.R

Dokumentation zuletzt aktualisiert am 30.08.2024

Das Modul zur Berechnung der übrigen Ausgaben wird in `mod_iv_ausgaben.R` wie folgt aufgerufen:

```
#------ Übrige Ausgaben ------------------------------------------------#
tl_iv_uebrigeausgaben <- mod_iv_uebrigeausgaben(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_GELDLEISTUNG = tl_iv_geldleistungen$IV_GELDLEISTUNG, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV,
    DISKONTFAKTOR = DISKONTFAKTOR)
```

In diesem Modul werden nacheinander alle Module aufgerufen, die Ausgaben berechnen die weder zu den Geld- noch zu den Sachleistungen zählen:

```
#------ Beiträge an Institutionen / Subventions aux institutions ------#
tl_iv_institutionen <- mod_iv_institutionen(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR =
    ↪  DISKONTFAKTOR)

#------ Durchführungskosten / Frais d'instruction ----------------------#
tl_iv_durchfuehrungskosten <- mod_iv_durchfuehrungskosten(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
#------ Verwaltungsaufwand / Frais d'administration --------------------#
tl_iv_verwaltungsaufwand <- mod_iv_verwaltungsaufwand(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ Andere Aufwände (Total) / Autres charges (total) --------------#
tl_iv_andere_aufw <- mod_iv_andere_aufw(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG)
```

Die obigen Module werden, mit Ausnahme von `mod_iv_andere_aufw.R`, weleches ein Vektor von 0-Projektionen zurückgibt, in den Kapiteln 4.15, 4.16 und 4.17 erläutert.

Nach den Berechnungen werden alle übrigen Ausgaben in einem Dataframe zusammengefasst:

```
#------ Total alle übrigen Ausgaben ------------------------------------#
UEBRIGE_AUSGABEN_IV <- tl_iv_institutionen$IV_INSTITUTIONEN %>%
    inner_join(tl_iv_durchfuehrungskosten$IV_DURCHFUEHRUNG, by = c("jahr")) %>%
    inner_join(tl_iv_verwaltungsaufwand$IV_VERWALTUNGSAUFWAND,
        by = c("jahr")) %>%
    inner_join(tl_iv_andere_aufw$IV_ANDERE_AUFW, by = c("jahr")) %>%
    mutate(uebrige_ausgaben_iv = btr_inst_org_iv + verw_aufw_iv +
        a_aufw)
```

Zum Schluss werden die berechneten übrigen Ausgaben an das Finanzperspektivenmodell, respektive das Modul `mod_iv_ausgaben`, zurückgegeben:

```
#------ Output ---------------------------------------------------------#
mod_return(UEBRIGE_AUSGABEN_IV)
```

4.1.10 Modul mod_iv_einnahmen.R

Dokumentation zuletzt aktualisiert am 07.10.2024

Das Modul zu den Einnahmen wird in `wrap_iv_hauptberechnung.R` wie folgt aufgerufen:

```
tl_iv_einnahmen <- mod_iv_einnahmen(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    AKTIVE_BEV = tl_inp$AKTIVE_BEV, IK = tl_inp$IK, ESTV = tl_inp$ESTV,
    ESTV_IV_SCEN = tl_inp$ESTV_IV_SCEN, EINK_ENTWICKLUNG = tl_inp$EINK_ENTWICKLUNG,
```

```
    IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, AUSGABEN_IV = tl_iv_ausgaben$AUSGABEN_IV,
    RENTENINDEX_ZR = tl_inp$RENTENINDEX_ZR, MWST_SATZ_IV = tl_inp$MWST_SATZ_IV,
    ZINS = tl_inp$ZINS, DISKONTFAKTOR = tl_inp$DISKONTFAKTOR,
    POP_SCENARIO_EPT_FULL = tl_inp$POP_SCENARIO_EPT_FULL, INDICES_FUTURS =
    ↪  tl_inp$INDICES_FUTURS,
    ECKWERTE = tl_inp$ECKWERTE, AHV_ABREECHNUNG_DEF = tl_inp$AHV_ABRECHNUNG_DEF)
```

Als erstes wird in `mod_iv_einnahmen` das Modul zur Berechnung der Einnahmen aus Beiträgen von Arbeitgebern und Versicherten aufgerufen:

```
#------ Beiträge / Contributions ---------------------------------------#
if (PARAM_GLOBAL$beitragsmodell == "OLD") {
    tl_iv_beitrag <- mod_iv_beitrag(PARAM_GLOBAL = PARAM_GLOBAL,
        AKTIVE_BEV = AKTIVE_BEV, IK = IK, EINK_ENTWICKLUNG = EINK_ENTWICKLUNG,
        IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR)
} else {
    tl_iv_beitrag <- mod_iv_beitrag_NEW(PARAM_GLOBAL = PARAM_GLOBAL,
        POP_SCENARIO_EPT_FULL = POP_SCENARIO_EPT_FULL, INDICES = INDICES_FUTURS,
        ECKWERTE = ECKWERTE, AKTIVE_BEV = AKTIVE_BEV, IK = IK,
        EINK_ENTWICKLUNG = EINK_ENTWICKLUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
        DISKONTFAKTOR = DISKONTFAKTOR)
}
```

Hierbei wird berücksichtigt, ob das bestehende Beitragsmodell (`PARAM_GLOBAL$beitragsmodell == "OLD "`) ausgewählt werden soll, oder das neue Beitragsmodell. Stand 2024 wird das bestehende Beitragsmodell verwendet, da das neue Beitragsmodell noch in Entwicklung ist. `mod_iv_beitrag` wird in Kapitel 4.18 beschrieben.

Als nächstes wird die Tidylist erstellt, welche das Dataframe `BUND_IV` enthält. `BUND_IV` enthält die Werte aus der Abrechnung für den Bundesbeitrag, sowie 0 als Platzhalter für alle Jahre nach der letzten Abrechnung. Der Grund, weshalb hier nur die Abrechnungsdaten für den Bundesbeitrag angefügt werden, ist, dass der Bundesbeitrag mindestens 37.7% und maximal 50% der Gesamtausgaben der IV betragen muss, und die Gesamtausgaben in diesem Stand des Modells noch nicht berechnet sind, da sowohl die Ausgaben für Politikmassnahmen als auch der Zinsaufwand erst später berechnet werden können. Aus diesem Grund werden die Einnahmen aus dem Bundesbeitrag für die Zukunft erst im Modul `mod_bilanz_iv_rekursiv.R` (vgl. Kapitel 4.23) berechnet, und hier provisorisch auf 0 gesetzt.[24]

```
tl_iv_bund <- list(BUND_IV = data.frame(jahr =
↪  c(PARAM_GLOBAL$jahr_beginn:PARAM_GLOBAL$jahr_ende)) %>%
    left_join(IV_ABRECHNUNG %>%
        select(jahr, btr_bund), by = "jahr") %>%
    mutate(btr_bund = coalesce(btr_bund, 0)))
```

Als nächstes wird das Modul zur Berechnung der übrigen Einnahmen aufgerufen (vgl. Kapitel 4.19):

```
#------ Übrige Einnahmen ------------------------------------------------#
tl_iv_uebrigeeinnahmen <- mod_iv_uebrigeeinnahmen(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, AUSGABEN_IV = AUSGABEN_IV)
```

Als nächstes wird das Modul zur Berechnung der allfälligen Einnahmen aus einer MWST-Zusatzfinanzierung für die IV ausgeführt. Da momentan keine solche Zusatzfinanzierung besteht, wird auf dieses Modul nicht weiter eingegangen:

[24] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
#------ Zusatzfinanzierung ------------------------------------------#
tl_iv_zusatzfinanzierung <- mod_iv_zusatzfinanzierung(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, ESTV = ESTV, MWST_SATZ_IV = MWST_SATZ_IV,
    IV_LOHNSUMME = tl_iv_beitrag$IV_LOHNSUMME)
```

Zum Schluss werden alle Einnahmen in einem Dataframe zusammengefasst:

```
#------ Total der Einnahmen -----------------------------------------#
EINNAHMEN_IV <- tl_iv_beitrag$IV_BEITRAGSSUMME %>%
    inner_join(tl_iv_bund$BUND_IV, by = c("jahr")) %>%
    inner_join(tl_iv_zusatzfinanzierung$ZUSATZFINANZIERUNG_IV,
        by = c("jahr")) %>%
    inner_join(tl_iv_uebrigeeinnahmen$UEBRIGEEINNAHMEN_IV, by = c("jahr")) %>%
    mutate(across(where(is.numeric), ~if_else(is.na(.), 0, .),
        .names = "{.col}")) %>%
    mutate(btr_oeffh = btr_bund + btr_mwst + btr_zinsiv) %>%
    mutate(ein_tot_iv = 0)

for (i in 1:nrow(EINNAHMEN_IV)) {
    if (!is.na(IV_ABRECHNUNG$btr_kant[i]) & IV_ABRECHNUNG$btr_kant[i] !=
        0) {
        EINNAHMEN_IV$ein_tot_iv[i] <- IV_ABRECHNUNG$ein_tot[i]
    } else {
        EINNAHMEN_IV$ein_tot_iv[i] <- EINNAHMEN_IV$btr_vs_ag[i] +
            EINNAHMEN_IV$btr_bund[i] + EINNAHMEN_IV$btr_mwst[i] +
            EINNAHMEN_IV$btr_zinsiv[i] + EINNAHMEN_IV$regr_ein_tot[i] +
            EINNAHMEN_IV$ein_uebr_tot[i]
    }
}
```

Während der erste Codeblock einfach alle Einnahmen-Dataframes zusammenfügt, so berechnet der zweite Codeblock die totalen Einnahmen, wobei unterschieden wird zwischen den Perioden, wo `btr_kant>0` war (bis und mit 2007), also wo die IV zu einem Teil durch Kantonsbeiträgen finanziert wurde, und der Periode danach, um sicherzustellen, dass alle Einnahmen berücksichtigt werden.

Zum Schluss wird die projizierte Lohnsumme ausgelesen, und danach zusammen mit dem Einnahmen-Dataframe an das Finanzperspektivenmodell, respektive das Modul `wrap_iv_hauptberechnung`, zurückgegeben:

```
#------ Output ------------------------------------------------------#
IV_LOHNSUMME <- tl_iv_beitrag$IV_LOHNSUMME

mod_return(EINNAHMEN_IV, IV_LOHNSUMME)
```

4.1.11 Modul wrap_iv_massnahmen.R

Dokumentation zuletzt aktualisiert am 20.09.2024

Das Modul zu den Massnahmen dient dazu, die Auswirkungen von Politikmassnahmen auf die IV abzuschätzen (vgl. auch Kapitel 1.2.2). Es wird in `wrap_iv.R` wie folgt aufgerufen:

```
tl_iv_mass <- wrap_iv_massnahmen(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung)
```

Am Anfang des Moduls steht die Prüfung, ob überhaupt auszuführende Massnahmen spezifiziert wurden:

```
if (tl_inp$PARAM_GLOBAL$flag_param_massn) {
```

Wenn dies nicht der Fall ist (also `tl_inp$PARAM_GLOBAL$flag_param_massn==FALSE` ist), werden direkt leere Dataframes für die Auswirkungen der Massnahmen erstellt und das Modul wird abgeschlossen:

```
else {
    IV_MASSNAHMEN <- data.frame(jahr =
↪  c(tl_inp$PARAM_GLOBAL$jahr_beginn:tl_inp$PARAM_GLOBAL$jahr_ende))
    IV_EINAUSGABEN <- data.frame(jahr = as.numeric(), variable = as.character(), wert =
↪  as.numeric())
    tl_iv_massnahmen <- list(IV_MASSNAHMEN = as_tibble(IV_MASSNAHMEN), IV_EINAUSGABEN =
↪  as_tibble(IV_MASSNAHMEN))
  }
```

Falls Massnahmen spezifiziert wurden (also `tl_inp$PARAM_GLOBAL$flag_param_massn==TRUE` ist), werden zuerst die Arten von Massnahmen (intern und/oder extern) ausgelesen und in der Liste `aktiviert` abgelegt:

```
#------ Berechnungen Massnahmen ----------------------------------------#
aktiviert <- strsplit(tl_inp$PARAM_MASSNAHMEN_BASE$verwendete_massnahmen,
    ", ")[[1]]
```

Als nächstes wird überprüft, ob eine der Massnahmen `iv_remb_dette_res_exploit`, `iv_modification_bundesbeitrag`, oder `iv_remb_hors_recette` enthalten ist, da für diese eine andere Berechnungsweise angewendet werden müsste. Konkret würden diese Massnahmen, falls spezifiziert, aus dem Dataframe `tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int` entfernt, und für die spätere Bearbeitung im Dataframe `PARAM_MASSNAHMEN_BASE_TK` zwischengespeichert. Da Stand 2024 keine der drei Massnahmen `iv_remb_dette_res_exploit`, `iv_modification_bundesbeitrag`, und `iv_remb_hors_recette` aktiviert ist wird hier nicht weiter auf diesen Codeteil eingegangen:

```
PARAM_MASSNAHMEN_BASE_TK <- tl_inp$PARAM_MASSNAHMEN_BASE
if (length(separate_at_comma(PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) >
    1) {

    intern_mass <-
↪  separate_at_comma(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)

    if (grepl(",iv_remb_dette_res_exploit",
    ↪  PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {
        tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int <-
↪  gsub(",iv_remb_dette_res_exploit",
            PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int,
            replacement = "")
    }
    if (grepl(",iv_modification_bundesbeitrag",
    ↪  PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {
        tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int <-
↪  gsub(",iv_modification_bundesbeitrag",
            PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int,
            replacement = "")
    }
    if (grepl(",iv_remb_hors_recette",
    ↪  PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {
        tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int <-
↪  gsub(",iv_remb_hors_recette",
```

```
            PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int,
            replacement = "")
    }
    if (!is.na(match("intern", aktiviert))) {
        tl_iv_massnahmen_int <- wrap_iv_massnahmen_int(tl_inp = tl_inp,
            tl_iv_hauptberechnung = tl_iv_hauptberechnung)
    } else {

        tl_iv_massnahmen_int <- NULL

    }

}
```

Als nächstes wird geprüft, ob Massnahmen von externen Sozialversicherungen aktiviert sind, ohne dass gleichzeitig extern spezifizierte IV-Massnahmen aktiviert sind (was spätestens seit 2024 nicht mehr der Fall ist) und wenn dies der Fall ist, wird die Berechnung der Auswirkungen dieser Massnahmen über das Modul `mod_iv_massnahmen_ext.R` (vgl. Kapitel 4.20) eingeleitet, und die Resultate werden im Dataframe `IV_EINAUSGABEN` abgelegt:

```
if (!(is.na(match("ext_sv", aktiviert)) & is.na(match("ext_iv",
    aktiviert)))) {
    tl_iv_massnahmen_ext <- mod_iv_massnahmen_ext(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
        PARAM_MASSNAHMEN = tl_inp$PARAM_MASSNAHMEN_BASE, IV_ABRECHNUNG =
        ↪  tl_inp$IV_ABRECHNUNG,
        MASSNAHMEN = tl_inp$MASSNAHMEN, MASSNAHMEN_DESC = tl_inp$MASSNAHMEN_DESC,
        MASSNAHMEN_SV = tl_inp$MASSNAHMEN_SV, MASSNAHMEN_SV_DESC =
        ↪  tl_inp$MASSNAHMEN_SV_DESC)
}
IV_EINAUSGABEN <- tl_iv_massnahmen_ext$EXT_EINAUSGABEN
IV_EINAUSGABEN <- IV_EINAUSGABEN %>%
    group_by(jahr, Konto) %>%
    summarize(wert = sum(wert)) %>%
    pivot_wider(names_from = Konto, values_from = wert)
tl_inp$IV_EINAUSGABEN <- IV_EINAUSGABEN
```

Danach wird auch geprüft, ob IV-interne Massnahmen aktiviert sind, und wenn dies der Fall ist, wird die Berechnung der Auswirkungen dieser Massnahmen über das Modul `wrap_iv_massnahmen_int.R` (vgl. Kapitel 4.21) eingeleitet:

```
if (!is.na(match("intern", aktiviert))) {
    tl_iv_massnahmen_int <- wrap_iv_massnahmen_int(tl_inp = tl_inp,
        tl_iv_hauptberechnung = tl_iv_hauptberechnung)
    tl_inp$PARAM_GLOBAL$reb_res = TRUE
} else {

    tl_iv_massnahmen_int <- NULL

}
```

Als nächstes werden nochmals die externen Massnahmen berechnet (vorausgesetzt diese sind spezifiziert):

```
#------ Extern berechnete Massnahmen ---------------------------------#
if (!(is.na(match("ext_sv", aktiviert)) & is.na(match("ext_iv",
    aktiviert)))) {
```

```
    tl_iv_massnahmen_ext <- mod_iv_massnahmen_ext(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
        PARAM_MASSNAHMEN = tl_inp$PARAM_MASSNAHMEN_BASE, IV_ABRECHNUNG =
        ↪  tl_inp$IV_ABRECHNUNG,
        MASSNAHMEN = tl_inp$MASSNAHMEN, MASSNAHMEN_DESC = tl_inp$MASSNAHMEN_DESC,
        MASSNAHMEN_SV = tl_inp$MASSNAHMEN_SV, MASSNAHMEN_SV_DESC =
        ↪  tl_inp$MASSNAHMEN_SV_DESC)
}
```

In einem nächsten Schritt werden die Projektionen für die Ausgaben- und Einnahmeneffekte aller Massnahmen zusammengefasst und in die tidy-Liste `tl_iv_massnahmen` abgelegt, wobei die verschiedenen Schritte nur ausgeführt werden, wenn die entsprechenden Massnahmentypen aktiviert sind, was die diversen `if` statements begründet:

```
#------ Zusammenfassen der Massnahmen ----------------------------------#
if (exists("tl_iv_massnahmen_int") & !is.null(tl_iv_massnahmen_int$INT_MASSNAHMEN)) {
    if (exists("tl_iv_massnahmen_ext")) {
        IV_MASSNAHMEN <- tl_iv_massnahmen_int$INT_MASSNAHMEN %>%
            full_join(tl_iv_massnahmen_ext$EXT_MASSNAHMEN, by = c("jahr"))
        IV_EINAUSGABEN <- rbind(tl_iv_massnahmen_int$INT_EINAUSGABEN,
            tl_iv_massnahmen_ext$EXT_EINAUSGABEN)
    } else {
        IV_MASSNAHMEN <- tl_iv_massnahmen_int$INT_MASSNAHMEN
        IV_EINAUSGABEN <- tl_iv_massnahmen_int$INT_EINAUSGABEN
    }
} else {
    IV_MASSNAHMEN <- tl_iv_massnahmen_ext$EXT_MASSNAHMEN
    IV_EINAUSGABEN <- tl_iv_massnahmen_ext$EXT_EINAUSGABEN
}

IV_MASSNAHMEN <- IV_MASSNAHMEN %>%
    mutate(across(.cols = everything(), as.numeric, .names = "{.col}")) %>%
    mutate(across(.cols = everything(), list(~if_else(is.na(.),
        0, .)), .names = "{.col}"))
IV_EINAUSGABEN <- IV_EINAUSGABEN %>%
    group_by(jahr, Konto) %>%
    summarize(wert = sum(wert)) %>%
    pivot_wider(names_from = Konto, values_from = wert)
suppressWarnings(if (!is.null(IV_EINAUSGABEN)) {
    IV_EINAUSGABEN[is.na(IV_EINAUSGABEN)] <- 0
})

tl_iv_massnahmen <- list(IV_MASSNAHMEN = as_tibble(IV_MASSNAHMEN),
    IV_EINAUSGABEN = as_tibble(IV_EINAUSGABEN))
```

Als nächstes wird widerum überprüft, ob eine der Massnahmen `iv_remb_dette_res_exploit`, `iv_modification_bundesbeitrag`, und `iv_remb_hors_recette` enthalten ist, und im gegebenen Fall werden die Berechnungen für diese Massnahmen durchgeführt. Da Stand 2024 keine der drei Massnahmen `iv_remb_dette_res_exploit`, `iv_modification_bundesbeitrag`, und `iv_remb_hors_recette` aktiviert ist wird hier nicht weiter auf diesen Codeteil eingegangen:

```
if ("iv_remb_dette_res_exploit" %in%
↪  separate_at_comma(PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {
    tl_res_expl <- mod_opt_iv_remb_dette_res_exploit(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
        PARAM_IV_REG_DETTE = tl_inp$PARAM_IV_REG_DETTE, AUSGABEN_IV =
        ↪  tl_iv_hauptberechnung$AUSGABEN_IV,
```

```
        EINNAHMEN_IV = tl_iv_hauptberechnung$EINNAHMEN_IV, PARAM_MASSNAHMEN_BASE =
        ↪  tl_inp$PARAM_MASSNAHMEN_BASE,
        IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, MASSNAHMEN = tl_inp$MASSNAHMEN,
        MASSNAHMEN_DESC = tl_inp$MASSNAHMEN_DESC, MASSNAHMEN_SV = tl_inp$MASSNAHMEN_SV,
        MASSNAHMEN_SV_DESC = tl_inp$MASSNAHMEN_SV_DESC, ZINS = tl_inp$ZINS,
        IV_LOHNSUMME = tl_iv_hauptberechnung$IV_LOHNSUMME, ECKWERTE_EXTENDED =
        ↪  tl_inp$ECKWERTE_EXTENDED,
        IV_EINAUSGABEN = IV_EINAUSGABEN)
    OPT_AUSGABEN = tl_res_expl$OPT_AUSGABEN
    OPT_EINNAHMEN = tl_res_expl$OPT_EINNAHMEN
    OPT_BILANZ = tl_res_expl$OPT_BILANZ

    tl_inp$PARAM_GLOBAL$reb_res = TRUE

    tl_iv_massnahmen_int$OPT_AUSGABEN = OPT_AUSGABEN
    tl_iv_massnahmen_int$OPT_EINNAHMEN = OPT_EINNAHMEN
    tl_iv_massnahmen_int$OPT_BILANZ = OPT_BILANZ
}

if ("iv_modification_bundesbeitrag" %in%
↪  separate_at_comma(PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {
    tl_bundesbeitrag <- mod_opt_iv_modification_bundesbeitrag(PARAM_GLOBAL =
↪  tl_inp$PARAM_GLOBAL,
        PARAM_IV_BUNDESBEITRAG_MOD = tl_inp$PARAM_IV_BUNDESBEITRAG_MOD,
        AUSGABEN_IV = tl_iv_hauptberechnung$AUSGABEN_IV, EINNAHMEN_IV =
        ↪  tl_iv_hauptberechnung$EINNAHMEN_IV,
        PARAM_MASSNAHMEN_BASE = tl_inp$PARAM_MASSNAHMEN_BASE,
        IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, MASSNAHMEN = tl_inp$MASSNAHMEN,
        MASSNAHMEN_DESC = tl_inp$MASSNAHMEN_DESC, MASSNAHMEN_SV = tl_inp$MASSNAHMEN_SV,
        MASSNAHMEN_SV_DESC = tl_inp$MASSNAHMEN_SV_DESC, ZINS = tl_inp$ZINS,
        IV_LOHNSUMME = tl_iv_hauptberechnung$IV_LOHNSUMME, ECKWERTE_EXTENDED =
        ↪  tl_inp$ECKWERTE_EXTENDED,
        IV_EINAUSGABEN = IV_EINAUSGABEN)

    OPT_AUSGABEN = tl_bundesbeitrag$OPT_AUSGABEN
    OPT_EINNAHMEN = tl_bundesbeitrag$OPT_EINNAHMEN
    OPT_BILANZ = tl_bundesbeitrag$OPT_BILANZ
    DELTA_EINN_BTR_BUND..iv_modification_bundesbeitrag =
↪  tl_bundesbeitrag$DELTA_EINN_BTR_BUND

    tl_inp$PARAM_GLOBAL$reb_res = TRUE

    tl_iv_massnahmen_int$OPT_AUSGABEN = OPT_AUSGABEN
    tl_iv_massnahmen_int$OPT_EINNAHMEN = OPT_EINNAHMEN
    tl_iv_massnahmen_int$OPT_BILANZ = OPT_BILANZ
    tl_iv_massnahmen_int$DELTA_EINN_BTR_BUND..iv_modification_bundesbeitrag <-
↪  DELTA_EINN_BTR_BUND..iv_modification_bundesbeitrag

    tl_iv_massnahmen$IV_MASSNAHMEN <- tl_iv_massnahmen$IV_MASSNAHMEN %>%
        left_join(DELTA_EINN_BTR_BUND..iv_modification_bundesbeitrag,
            by = "jahr") %>%
        rename(einnahmen..iv_modification_bundesbeitrag = wert)
}
```

```
if ("iv_remb_hors_recette" %in%
↪  separate_at_comma(PARAM_MASSNAHMEN_BASE_TK$aktivierte_massnahmen_int)) {

    OPT_AUSGABEN = tl_iv_massnahmen_int$OPT_AUSGABEN..iv_remb_hors_recette
    OPT_EINNAHMEN = tl_iv_massnahmen_int$OPT_EINNAHMEN..iv_remb_hors_recette
    OPT_BILANZ = tl_iv_massnahmen_int$OPT_BILANZ..iv_remb_hors_recette
    OPT_REMB_HORS_RECETTE =
↪  tl_iv_massnahmen_int$OPT_REMB_HORS_RECETTE..iv_remb_hors_recette

    # tl_inp$PARAM_GLOBAL$hors_recette = TRUE

    tl_iv_massnahmen_int$OPT_AUSGABEN..iv_remb_hors_recette = OPT_AUSGABEN
    tl_iv_massnahmen_int$OPT_EINNAHMEN..iv_remb_hors_recette = OPT_EINNAHMEN
    tl_iv_massnahmen_int$OPT_BILANZ..iv_remb_hors_recette = OPT_BILANZ
    tl_iv_massnahmen_int$OPT_REMB_HORS_RECETTE = OPT_REMB_HORS_RECETTE

    IV_MASSNAHMEN <- IV_MASSNAHMEN %>%
        mutate(iv_remb_hors_recette = case_when(jahr >=
        ↪  tl_inp$PARAM_IV_REMB_HORS_RECETTE$jahr_beginn &
            jahr <= tl_inp$PARAM_IV_REMB_HORS_RECETTE$jahr_end ~
            tl_inp$PARAM_IV_REMB_HORS_RECETTE$zuschlag_mio *
                1e+06, TRUE ~ 0))

    tl_iv_massnahmen$IV_MASSNAHMEN <- IV_MASSNAHMEN
}
```

Zum Schluss werden die entsprechenden Tidy-Listen an das Finanzperspektivenmodell, repsektive das Modul `wrap_iv.R` zurückgegeben:

```
#------ Output --------------------------------------------------------#
if (exists("tl_iv_massnahmen_int")) {
    return(c(tl_iv_massnahmen, tl_iv_massnahmen_int))
} else {
    return(c(tl_iv_massnahmen))
}
```

4.1.12 Modul wrap_iv_ergebnisse.R

Dokumentation zuletzt aktualisiert am 07.10.2024

Das Modul `wrap_iv_ergebnisse.R` dient dazu, die Erfolgsrechnung inklusive Umlageergebnis, und die Bilanz inklusive Kapital und Schuldenstand der IV zu berechnen. Es wird in `wrap_iv.R` wie folgt aufgerufen:

```
tl_iv_ergebnisse <- wrap_iv_ergebnisse(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung,
    tl_iv_mass = tl_iv_mass)
```

Da sowohl `sum(!is.na(tl_iv_mass$OPT_BILANZ..iv_remb_dette_res_exploit))>1` als auch `sum(!is.na(tl_iv_mass$OPT_BILANZ))>1` Stand 2024 `=FALSE` sind, wird hier nur auf den `else`-Teil der ersten `if`-Schleife eingegangen. Dieser sieht wie folt aus:

```
else{
#------ Massnahmen zum Total aufsummieren -----------------------------#
  tl_ein_aus            <- mod_iv_einausgaben(
     PARAM_GLOBAL       = tl_inp$PARAM_GLOBAL
```

```
    , IV_ABRECHNUNG       = tl_inp$IV_ABRECHNUNG
    , AUSGABEN_IV         = tl_iv_hauptberechnung$AUSGABEN_IV
    , EINNAHMEN_IV        = tl_iv_hauptberechnung$EINNAHMEN_IV
    , IV_EINAUSGABEN      = tl_iv_mass$IV_EINAUSGABEN
  )

#------ Berechnungen Umlage und Bilanz ---------------------------------#
  tl_bilanz_iv           <- mod_bilanz_iv_rekursiv(
     PARAM_GLOBAL        = tl_inp$PARAM_GLOBAL
    , IV_ABRECHNUNG      = tl_inp$IV_ABRECHNUNG
    , AUSGABEN_IV        = tl_ein_aus$AUSGABEN
    , EINNAHMEN_IV       = tl_ein_aus$EINNAHMEN
    , ZINS               = tl_inp$ZINS
    , IV_LOHNSUMME       = tl_iv_hauptberechnung$IV_LOHNSUMME
    , ECKWERTE_EXTENDED  = tl_inp$ECKWERTE_EXTENDED
    , FORTSCHREIBUNG_IV  = tl_inp$FORTSCHREIBUNG_IV
  )
}
```

Das heisst, es wird zuerst das Modul aufgerufen, das die Einnahmen- und Ausgaben aus den Hauptberechnungen und den Massnahmen-Berechnungen aggregiert (vgl. Kapitel 4.22), und danach das Modul, das die Bilanz der IV für jedes zukünftige Jahr berechnet (vgl. Kapitel 4.23).

Sobald diese Module ausgeführt sind wird der resultierende Output an das Finanzperspektivenmodell, respektive das Modul `wrap_iv.R` zurückgegeben:

```
#------ Output ---------------------------------------------------------#
return(c(tl_bilanz_iv, tl_ein_aus))  # tl_umlage_iv,
## Modul mod_iv_rentensumme_new.R \label{renten}
```

4.1.13 Modul wrap_iv_varia.R und mod_iv_indices.R

Dokumentation zuletzt aktualisiert am 23.09.2024

Das Modul `wrap_iv_varia.R` dient lediglich dazu, Indizes, die im veröffentlichten Finanzhaushalt der IV angegeben sind, auszulesen. Es wird wie folgt in `wrap_iv.R` aufgerufen:

```
tl_iv_varia <- wrap_iv_varia(tl_inp = tl_inp, tl_iv_hauptberechnung =
↪  tl_iv_hauptberechnung,
    tl_iv_mass = tl_iv_mass, tl_iv_ergebnisse = tl_iv_ergebnisse)
```

Stand 2024 besteht es lediglich aus den folgenden Codezeilen, welche selbsterklärend sind:

```
#------ Berechnen der Indices ------------------------------------------#
tl_iv_indices <- mod_iv_indices(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BILANZ_IV = tl_iv_ergebnisse$BILANZ_IV)

#------ Output ---------------------------------------------------------#
return(c(tl_iv_indices))
```

4.1.14 Modul mod_iv_postprocessing.R

Dokumentation zuletzt aktualisiert am 23.09.2024

Das Modul `mod_iv_postprocessing.R` dient lediglich dazu, die im Finanzperspektivenmodell berechneten Werte, welche nominal sind, in Werte zu konstanten Preisen umzurechnen. Es wird wie folgt in `wrap_iv.R`

aufgerufen:

```
#------ post-processing --------------------------------------------------#
tl_iv_postprocessing <- mod_iv_postprocessing(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_BILANZ = tl_iv_ergebnisse$BILANZ_IV, IV_MASSNAHMEN = tl_iv_mass$IV_MASSNAHMEN,
    IV_INDICES = tl_iv_varia$IV_INDICES, DISKONTFAKTOR = tl_inp$DISKONTFAKTOR)
```

Es wird der folgende Code aufgerufen, wobei erkennbar ist, dass der `if` und der `else`-Teil identisch sind, also die Bedingung keinen Unterschied macht. Die Bedingung ist lediglich noch im Code enthalten, falls manuell Zwecks Analysen eine Rebasierung der Berechnungen eingefügt werden sollte:

```
#------ realer FH --------------------------------------------------------#
if ("reb_res" %in% colnames(PARAM_GLOBAL)) {
    IV_FHH <- IV_BILANZ %>%
        diskontierung(DISKONTFAKTOR) %>%
        # # corrected kap_etr and erg_betr mutate(
        # rueckzahlung_b = if_else(rueckzahlung == 0,
        # rueckzahlung, kap - lag(kap)), anl_erg =
        # if_else(is.na(fonds - lag(fonds) - erg_umlag_iv +
        # rueckzahlung), anl_erg, fonds - lag(fonds) -
        # erg_umlag_iv + rueckzahlung), erg_betr =
        # erg_umlag_iv + anl_erg ) %>% # end corrected
        # kap_etr and erg_betr
    left_join(IV_INDICES %>%
        select(jahr, indx_bund, indx_fl_mtl_ausg), by = "jahr")
    #
} else {
    IV_FHH <- IV_BILANZ %>%
        diskontierung(DISKONTFAKTOR) %>%
        # # corrected kap_etr and erg_betr mutate(
        # rueckzahlung = if_else(rueckzahlung == 0,
        # rueckzahlung, kap - lag(kap)), anl_erg =
        # if_else(is.na(fonds - lag(fonds) - erg_umlag_iv +
        # rueckzahlung), anl_erg, fonds - lag(fonds) -
        # erg_umlag_iv + rueckzahlung), erg_betr =
        # erg_umlag_iv + anl_erg ) %>% # end corrected
        # kap_etr and erg_betr
    left_join(IV_INDICES %>%
        select(jahr, indx_bund, indx_fl_mtl_ausg), by = "jahr")
}
```

Die Funktion `diskontierung(DISKONTFAKTOR)` nimmt lediglich sämtliche Werte in `IV_BILANZ` ausser der Spalte `jahr`, und dividiert diese durch den in `DISKONTFAKTOR` angegebenen Deflator für das entsprechende Jahr:

```
diskontierung <- function(DATA, DISKONTFAKTOR, askontierung = FALSE) {
    colnames(DISKONTFAKTOR) <- c("jahr", "diskontfaktor")

    DATA_DISKONTIERT <- DATA %>%
        left_join(DISKONTFAKTOR, by = "jahr") %>%
        mutate(diskontfaktor = if (askontierung) {
            1/diskontfaktor
        } else {
            diskontfaktor
        }) %>%
```

```
        mutate_at(vars(-jahr, -diskontfaktor), list(~. * diskontfaktor)) %>%
        dplyr::select(-diskontfaktor)

    return(DATA_DISKONTIERT)
}
```

Als nächstes werden für den Fall, dass Politikmassnahmen aktiviert sind (`PARAM_GLOBAL$flag_param_massn==TRUE`) auch die Ausgabenvektoren der Politikmassnahmen deflationiert, und es wird die Liste `tl_out` erstellt:

```
#------ Output --------------------------------------------------------#
  if (PARAM_GLOBAL$flag_param_massn) {

    IV_MASSNAHMEN_NOM  <- IV_MASSNAHMEN
    IV_MASSNAHMEN_REAL <- IV_MASSNAHMEN %>% diskontierung(DISKONTFAKTOR)
    tl_out      <- tidy list(
      IV_MASSNAHMEN_REAL,
      IV_FHH,
      IV_MASSNAHMEN_NOM,
      IV_FHH_NOM
    )
  } else {
    tl_out      <- tidy list(IV_FHH, IV_FHH_NOM)
  }
```

Sobald diese Module ausgeführt sind werden die in der Liste `tl_out` enthaltenen Dataframes an das Finanzperspektivenmodell, respektive das Modul `wrap_iv.R`, zurückgegeben:

```
mod_return(tl_out)
```

4.2 Modul mod_iv_rentensumme_new.R

Dokumentation zuletzt aktualisiert am 10.10.2024

Notiz Übergangsphase: Im Moment wird das neue Modul nur aufgerufen, falls im Input-Conatiner in `PARAM_GLOBAL` dem Parameter *rentenmodell* der Wert *NEW* zugewiesen wird.

Das Prognosemodell zu den Rentenausgaben ist im Skript `mod_iv_rentensumme_new.R` enthalten, welches wie folgt im Skript `mod_iv_geldleistungen.R` aufgerufen wird:

```
tl_iv_rentensumme <- mod_iv_rentensumme_NEW(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEVOELKERUNG = BEVOELKERUNG, IV_RENTEN_NEW = IV_RENTEN_NEW)
```

Das Modul `mod_iv_rentensumme_new.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `RENTENENTWICKLUNG`: Dataframe mit der Entwicklung der Minimalrente.
- `BEVOELKERUNG`: Die Bevölkerungsdaten und -szenarien weren dazu genutzt, um die Anzahl Invaliden nach Alter in der Zukunft zu schätzen.
- `IV_RENTEN_NEW`: Dataframe mit Details zum Retenbestand (gemäss Rentenregister) sowie der Neurenten nach Jahr, Alter und Geschlecht.

In einem ersten Schritt werden in `mod_iv_rentensumme_NEW` Parameterwerte für die weiteren Berechnungen

gesetzt:[25]

```
jahr_projektion <- PARAM_GLOBAL$jahr_abr + 1

# Folgende Parameter könnten verwendet werden, um die
# Invalidisierung bei Jungen bis Alter X anders zu
# berechnen
junge_anders <- FALSE
junge_alter_bis <- 30
junge_anders_MA_lastyear <- 2017
```

4.2.1 Berechnung von Rentenbestand, Neurenten, und Mutationen

Die Berechnung des Invalidenbestandes sowie der Neurenten in Anzahl Minimalrenten erfolgt mit dem folgenden Code:

```
BESTAND <- IV_RENTEN_NEW %>%
    filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(bestand = mpr_tot/minimalrente, neurenten = mpr_nr/minimalrente) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten"))
```

Dieser Codeteil dividiert die total im Dezember eines Jahres ausbezahlten Renten (Hauptrenten+Kinderrenten) `mpr_tot` mit der in diesem Jahr gültigen Minimalrente (bspw. 1225 für das Jahr 2023), um den Rentenbestand in Anzahl Minimalrenten zu erhalten. Gleichsam dividiert es die neu ausbezahlte Renten `mpr_nr`, also die Haupt- und Kinderrenten, die im Dezember dieses Jahres ausbezahlt wurden aber nicht im Dezember des Vorjahres, mit der in diesem Jahr gültigen Minimalrente, um die Neurenten in Anzahl Minimalrenten zu erhalten.

Die Berechnung der Bestandesmutationen erfolgt mit den folgenden zwei Codeteilen:

```
    BESTAND <- BESTAND %>%
        filter(jahr!=min(BESTAND$jahr)) %>%
        full_join(BESTAND %>%
                      filter(jahr!=max(BESTAND$jahr)) %>%
                      mutate(alt=alt+1, jahr=jahr+1) %>%
                      rename(
                          bestand_vj=bestand,
                          bestand_vj_n=bestand_n
                      ) %>%
                      select(jahr, sex, alt, contains("bestand_vj"))
        ) %>%
        mutate_if(is.numeric, ~ coalesce(., 0)) %>%
```

Dieser Codeteil fügt dem Rentenbestand aus dem Vorjahr an das Dataframe mit dem aktuellen Rentenbestand an. Da dieselben Personen ein Jahr vorher ein Jahr jünger waren, wird zum Joinen nicht nur ein Jahr addiert, sonder auch ein Alter (bspw. die gleichen Frauen, die in 2023 30 Jahre alt sind waren im 2022 29 Jahre alt). Es wird sowohl der Bestand in Minimalrenten des Vorjahres (`bestand_vj`) als auch der Bestand in Anzahl RentenBeziehende des Vorjahres (`bestand_vj_n`) angehängt. Die letzte Zeile stellt sicher, dass Nullbestände als 0 in den Daten erscheinen (und nicht als NA).

Die Bestandesmutationen werden dann wie folgt berechnet:

[25] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
mutate(bestandesmutationen = bestand - bestand_vj - neurenten,
    bestandesmutationen_n = bestand_n - bestand_vj_n - neurenten_n) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten"),
        contains("bestandesmutationen")) %>%
    arrange(jahr, sex, alt)
```

D.h. die Bestandesmutationen (in Minimalrenten `bestandesmutationen` und in Anzahl Beziehende `bestandesmutationen_n`) ergeben sich aus dem Bestand im Dezember, abzüglich des Bestandes, der auf Neurenten zurückzuführen ist, und abzüglich des Bestandes des Vorjahres. Bestandesmutationen wiederspiegeln daher Veränderungen des Rentenbestandes, die nicht auf Neurenten zurückzuführen sind. Beispiel: Wenn der Bestand an 30 Jährigen Frauen in diesem Jahr 120 beträgt, davon 40 Neurenten sind, und der Bestand dieser Frauen im Vorjahr 100 betrug, so heisst dies, dass 20 aus dem Bestand abgegangen sind, und somit betragen die Bestandesmutationen -20.

4.2.2 Berechnung der Rate der Invalidisierung und der Mutationsrate

Die Invalidisierungs- und Mutationsraten werden wie folgt berechnet:

```
RATEN_INV_MUT <- BESTAND %>%
    left_join(BEVOELKERUNG %>%
        mutate(jahr = jahr + 1, alt = alt + 1) %>%
        group_by(jahr, sex, alt) %>%
        summarize(bevendejahr = sum(bevendejahr, na.rm = TRUE))) %>%
    mutate(invalidisierung = neurenten/(bevendejahr - bestand_vj_n),
        invalidisierung_n = neurenten_n/(bevendejahr - bestand_vj_n),
        mutationsrate = case_when(bestand_vj == 0 ~ 0, TRUE ~
            bestandesmutationen/bestand_vj), mutationsrate_n = case_when(bestand_vj_n ==
            0 ~ 0, TRUE ~ bestandesmutationen_n/bestand_vj_n)) %>%
    select(jahr, sex, alt, contains("invalidisierung"), contains("mutationsrate"))
```

Die Invalidisierung im Jahr t berechnet sich als der Anteil der Wohnbevölkerung, der Anfang des Jahres t noch keine Invalidenrente bezieht aber Ende des Jahres t eine Invalidenrente bezieht. Daher wird in einem ersten Schritt die Bevölkerung am Ende des Vorjahres (=Bevölkerung am Anfang des aktuellen Jahres), die am Ende des aktuellen Jahres ein Jahr älter sein wird, mit dem Befehl `left_join(BEVOELKERUNG` angefügt.

Die Invalidisierung im Jahr t ergibt sich dann aus den Neurenten dividiert durch die Bevölkerung am Ende des Vorjahres abzüglich der Anzahl Personen, die bereits am Ende des Vorjahres eine Rente bezogen haben. Die Bestandesmutationen ergeben sich aus dem Quotienten der Bestandsmutationen im Jahr t und des Bestandes am Ende des Vorjahres.

4.2.3 Invalidisierungs- und Mutationsraten für die Projektionen

Die durchschnittlichen Invalidisierungs- und Mutationsraten, die danach für die Projektionen verwendet werden, werden wie folgt berechnet:

```
    IV_RATEN_INV_MUT_RAW <- RATEN_INV_MUT %>%
        filter(
            if (junge_anders) {
                (alt <= junge_alter_bis & jahr >= junge_anders_MA_lastyear -
↪   (PARAM_GLOBAL$MA_years-1) & jahr <= junge_anders_MA_lastyear) |
                    (alt > junge_alter_bis & jahr >= PARAM_GLOBAL$jahr_abr -
↪   (PARAM_GLOBAL$MA_years-1) & jahr <= PARAM_GLOBAL$jahr_abr)
            } else {
                jahr >= PARAM_GLOBAL$jahr_abr - (PARAM_GLOBAL$MA_years-1) & jahr <=
↪   PARAM_GLOBAL$jahr_abr # MA_years-1 beschreibt die Anzahl Jahre vor jahr_abr, die
↪   verwendet werden sollen
```

```
            }
        ) %>%
        group_by(sex, alt) %>%
        summarize(
            invalidisierung = mean(invalidisierung, na.rm = TRUE),
            mutationsrate = mean(mutationsrate, na.rm = TRUE),
            invalidisierung_n = mean(invalidisierung_n, na.rm = TRUE),
            mutationsrate_n = mean(mutationsrate_n, na.rm = TRUE),
        ) %>%
        ungroup() %>%
```

Der `if`-Teil würde dem Fall dienen, dass für Junginvalide ein anderer Berechnungshorizont für die Invalidisierung angewendet werde soll. Da `junge_anders` Stand 2024 `==FALSE` ist, wird nicht auf diesen Teil eingegangen. Der Teil nach `else` wählt für die Berechnung der Invalidisierung und Mutationsrate die spezifizierte Anzahl Jahre vor dem letzten Abrechnungsjahr aus (Stand 2024 nutzen wir 3 Jahre, also die Jahre 2021, 2022, und 2023).

Danach wird die Invalidisierung/Mutationsrate aus der durchschnittlichen Invalidisierung/Mutationsrate der ausgewählten Jahre gebildet.

Der folge Codeteil dient zur Berechnung der Wachstumsrate der Invalidsierung. Diese Wachstumsrate wird ausschliesslich für die Szenarien „hoch“ und „tief“ „hoch“ und „tief“ verwendet:

```
        left_join(
            RATEN_INV_MUT %>%
                filter(
                    if (junge_anders) {
                        (alt <= junge_alter_bis & jahr >= junge_anders_MA_lastyear -
↪   (PARAM_GLOBAL$szenariolag-1) & jahr <= junge_anders_MA_lastyear) |
                            (alt > junge_alter_bis & jahr >= PARAM_GLOBAL$jahr_abr -
↪   (PARAM_GLOBAL$szenariolag-1) & jahr <= PARAM_GLOBAL$jahr_abr)
                    } else {
                        jahr >= PARAM_GLOBAL$jahr_abr - (PARAM_GLOBAL$szenariolag-1) &
↪   jahr <= PARAM_GLOBAL$jahr_abr # szenariolag-1 beschreibt die Anzahl Jahre vor
↪   jahr_abr, die verwendet werden sollen
                    }
                ) %>%
                arrange(sex, alt, jahr) %>%
                group_by(sex, alt) %>%
                mutate(
                    invalidisierung_diff = c(NA, abs(diff(invalidisierung))),
                    # mutationsrate_diff = c(NA, abs(diff(mutationsrate))) ,
                    invalidisierung_diff_n = c(NA, abs(diff(invalidisierung_n)))
                ) %>%
                summarise(
                    mean_abs_invalidisierung_diff = mean(invalidisierung_diff, na.rm =
                    ↪  TRUE),
                    # mean_abs_mutationsrate_diff = mean(mutationsrate_diff, na.rm =
                    ↪  TRUE) ,
                    mean_abs_invalidisierung_diff_n = mean(invalidisierung_diff_n, na.rm
                    ↪  = TRUE)
                )  %>%
                ungroup(),
            by = c("sex", "alt")
        ) %>%
```

Der `if`-Teil würde wiederum dem Fall dienen, dass für Junginvalide ein anderer Berechnungshorizont für die Invalidisierung angewendet werde soll. Da `junge_anders` Stand 2024 `==FALSE` ist, wird nicht auf diesen Teil eingegangen. Der Teil nach `else` wählt für die Berechnung der Szenarien die spezifizierte Anzahl Jahre vor dem letzten Abrechnungsjahr aus (Stand 2024 nutzen wir 5 Jahre, also die Jahre 2019, 2020, 2021, 2022, und 2023).

Danach bilden wir die absolute Jahr-zu-Jahr Veränderung der Invalidisierung über den ausgewählten Zeitraum (die Mutationsrate ist auskommentiert, da Stand 2024 die Szenarien nur auf die Invalidisierungsrate, nicht aber auf die Mutationsrate, abstützten). Zum Schluss bilden wir den Mittelwert über die absolute Jahr-zu-Jahr Veränderung der Invalidisierung.

Der folgende Codeteil wäre für den Fall, dass das vergangene Wachstum der Invalidisierung für weitere Jahre fortgeschrieben werden soll. Da diese Option Stand 2024 nicht aktiviert ist (`PARAM_GLOBAL$fort_entw==FALSE`) wird nicht weiter auf diesen Codeteil eingegangen.

```
left_join(RATEN_INV_MUT %>%
    filter(if (junge_anders) {
        (alt <= junge_alter_bis & (jahr == junge_anders_MA_lastyear |
            jahr == junge_anders_MA_lastyear - PARAM_GLOBAL$wachstum_lagyears)) |
            (alt > junge_alter_bis & (jahr == PARAM_GLOBAL$jahr_abr |
                jahr == PARAM_GLOBAL$jahr_abr - PARAM_GLOBAL$wachstum_lagyears))
    } else {
        jahr == PARAM_GLOBAL$jahr_abr | jahr == PARAM_GLOBAL$jahr_abr -
            PARAM_GLOBAL$wachstum_lagyears
    }) %>%
    arrange(sex, alt, jahr) %>%
    mutate(rank = min_rank(jahr)) %>%
    mutate(period = ifelse(rank == 1, 0, 1)) %>%
    group_by(sex, alt) %>%
    mutate(invalidisierung_growth = c(NA,
    ↪  diff(invalidisierung))/PARAM_GLOBAL$wachstum_lagyears,
        mutationsrate_growth = c(NA, diff(mutationsrate))/PARAM_GLOBAL$wachstum_lagyears,
        invalidisierung_growth_n = c(NA,
        ↪  diff(invalidisierung_n))/PARAM_GLOBAL$wachstum_lagyears,
        mutationsrate_growth_n = c(NA,
        ↪  diff(mutationsrate_n))/PARAM_GLOBAL$wachstum_lagyears,
        ) %>%
    ungroup() %>%
    filter(period == 1) %>%
    select(sex, alt, contains("_growth")), by = c("sex", "alt"))
```

Mit dem folgenden Codeteil werden die Projektionsjahr-spezifischen Invalidisierungsraten gebildet. Dieser Codeteil ist relevant, da die Szenarien „tief“ und „hoch“ über die kommenden 20 Jahre ansteigende/abfallende Invalidisierungsraten verwenden.

```
    # Initialisiere ein leeres Dataframe vor der Schleife
    IV_RATEN_INV_MUT <- tibble()

    for(lJahr in (jahr_projektion):(jahr_projektion+20)) {

        IV_RATEN_INV_MUT_JAHR <-
            IV_RATEN_INV_MUT_RAW %>%
            mutate(
                invalidisierung = invalidisierung + PARAM_GLOBAL$fort_entw *
                ↪  invalidisierung_growth *((ifelse(lJahr - jahr_projektion <
                ↪  PARAM_GLOBAL$fort_entw_years, lJahr - jahr_projektion + 1,
                ↪  PARAM_GLOBAL$fort_entw_years))),
```

```
            invalidisierung_n = invalidisierung_n + PARAM_GLOBAL$fort_entw *
            ↪  invalidisierung_growth_n *((ifelse(lJahr - jahr_projektion <
            ↪  PARAM_GLOBAL$fort_entw_years, lJahr - jahr_projektion + 1,
            ↪  PARAM_GLOBAL$fort_entw_years))),
            mutationsrate = mutationsrate + PARAM_GLOBAL$fort_entw *
            ↪  mutationsrate_growth * ((ifelse(lJahr - jahr_projektion <
            ↪  PARAM_GLOBAL$fort_entw_years, lJahr - jahr_projektion + 1,
            ↪  PARAM_GLOBAL$fort_entw_years))),
            mutationsrate_n = mutationsrate_n + PARAM_GLOBAL$fort_entw *
            ↪  mutationsrate_growth_n * ((ifelse(lJahr - jahr_projektion <
            ↪  PARAM_GLOBAL$fort_entw_years, lJahr - jahr_projektion + 1,
            ↪  PARAM_GLOBAL$fort_entw_years))),
            invalidisierung_tief = invalidisierung + mean_abs_invalidisierung_diff *
            ↪  (1 + (lJahr - jahr_projektion) * (1/20)),
            invalidisierung_hoch = invalidisierung - mean_abs_invalidisierung_diff *
            ↪  (1 + (lJahr -  jahr_projektion) * (1/20)),
            invalidisierung_tief_n = invalidisierung_n +
            ↪  mean_abs_invalidisierung_diff_n * (1 + (lJahr - jahr_projektion) *
            ↪  (1/20)),
            invalidisierung_hoch_n = invalidisierung_n -
            ↪  mean_abs_invalidisierung_diff_n * (1 + (lJahr -  jahr_projektion) *
            ↪  (1/20))
        ) %>%
        select(alt, sex, contains("mutationsrate"), contains("invalidisierung"),
        ↪  contains("invalidisierung_tief"), contains("invalidisierung_hoch")) %>%
        select(-contains("_diff"), -contains("growth")) %>%
        mutate(jahr = lJahr) # Füge eine Spalte mit dem aktuellen Jahr hinzu
```

Es wird eine Schleife über die nächsten 20 Jahre ab dem aktuellen Projektionsjahr geschrieben (Für die Finanzperspektiven 2024 entspricht dies einer Schleife von 2024 bis 2044). Danach werden die Invalidisierungs- und Mutationsrate für das entsprechende Jahr berechnet. `invalidisierung` berechnet die Invalidisierung im mittleren Szenario. Der Parameter `fort_entw` ist Stand 2024 = 0 respektive `FALSE`, Term der mit `fort_entw` multipliziert wird wegfällt. Daher gehen wir nicht weiter auf diesen Term ein. `invalidisierung = invalidisierung + fort_entw *...` zeigt, dass die Invalidisierung im mittleren Szenario konstant und gleich der vorangehend berechneten durchschnittlichen Invalidisierung ist. Das gleiche trifft auf die anderen drei Variablen im mittleren Szenario zu (`invalidisierung_n`, `mutationsrate`, und `mutationsrate_n`). Die Invalidisierung im tiefen Szenarion `invalidisierung_tief` ergibt sich aus der Invalidisierung im mittleren Szenario zuzüglich `mean_abs_invalidisierung_diff` multipliziert mit einem 20zigstel der Anzahl Jahre, die das entsprechende Jahr vom aktuellen Projektionsjahr entfernt ist. Die Berechnung der Invalidisierung im hohen Szenario erfolgt analog.

Der Nachfolgende Codeteil dient dazu, den Effekt der AHV-21 bedingte Rentenaltererhöhung von 64 auf 65 bei den Frauen auf die IV abzubilden.[26]

```
    if (PARAM_GLOBAL$mitAHV21 & lJahr>=2025 & (PARAM_GLOBAL$jahr_abr -
    ↪  (PARAM_GLOBAL$MA_years-1))<2028) { # Ab 2028 ist die AHV-21 voll in Kraft,
    ↪  das heisst, dass ab dann die Invalidisierungs- und Mutationsraten der
    ↪  64-jährigen Frauen nicht mehr von der AHV21-Übergangsregelung betroffen sind,
    ↪  und für die Schätzung verwendet werden können. Da die dem letzten MA-Jahr
    ↪  vorangehenden PARAM_GLOBAL$MA_years auch für die Schätzung verwendet werden,
    ↪  muss sicher gestellt werden, dass diese Jahre auch mindestens 2028 sind.

        IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
```

[26] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
            filter(!((alt == 65 | alt==64) & sex == "f")) %>%  # Entferne alle Zeilen
            ↪  mit (alt == 64 | alt ==65) und sex == "f" (solche werden ab 2025
            ↪  enthalten sein, da ab dann auf Grund der AHV21 auch Frauen, die Ende
            ↪  Jahr 65 sein werden, in der IV sind)
            add_row(alt = 65,
                    sex = "f",
                    mutationsrate = -1,
                    mutationsrate_n = -1,
                    invalidisierung = 0,
                    invalidisierung_n = 0,
                    invalidisierung_tief = 0,
                    invalidisierung_hoch = 0,
                    invalidisierung_tief_n = 0,
                    invalidisierung_hoch_n = 0,
                    jahr = lJahr)  # Füge neue Zeile für die 65-jährigen Frauen mit
                    ↪  den richtigen mutationsraten (-1) und invalidisierungen (0)
                    ↪  hinzu

        # Dupliziere Zeilen mit alt == 63 und sex == "f" und setze alt = 64 in der
        ↪  duplizierten Version
        duplicated_rows <- IV_RATEN_INV_MUT_JAHR %>%
            filter(alt == 63 & sex == "f") %>%
            mutate(alt = 64)

        IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
            bind_rows(duplicated_rows)

        # Ändere die Zeilen mit sex == "f" und alt == 64
        IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
            mutate(across(contains("invalidisierung"), ~ ifelse(sex == "f" & alt ==
            ↪  64, . * min(4, (lJahr - 2024)) / 4, .))) %>%
            mutate(across(contains("mutationsrate"), ~ ifelse(sex == "f" & alt == 64,
            ↪  . * min(4, (lJahr - 2024)) / 4 + min(0, (lJahr - 2028)) / 4, .))) %>%
            arrange(sex, alt)
    }

    # Aktualisiere das Dataframe mit neuen Daten
    IV_RATEN_INV_MUT <- bind_rows(IV_RATEN_INV_MUT, IV_RATEN_INV_MUT_JAHR) # Füge die
↪  Ergebnisse der Iteration zum Dataframe hinzu

}
```

In einem ersten Schritt wird die Invalidisierung und Mutationsrate bei Frauen mit Alter 64 und 65 entfernt. Danach wird eine Zeile hinzugefügt, die die Invalidisierung und Mutationsrate für 65-jährige Frauen so setzt, wie sie nach Einführung der AHV21 zu erwarten ist (d.h., es gehen alle Frauen 65-jährige Frauen in die AHV über (Mutationsrate von -1) und es kommen keine neuen 65-jährige Frauen dazu (Invalidisierung von 0)).

Danach wird das dataframe `duplicated_rows` gebildet, welches die Invaldisierungs- und Mutationsraten der 63-jährigen Frauen auf die 64-jährigen Frauen überträgt. Der Grund ist, dass wir in der Übergangsphase zur AHV-21 keine Daten zur Invalidisierung und den Mutationen bei den 64-jährigen Frauen haben (da vor der AHV-21 alle Frauen mit 64 schon in der AHV sind). Daher nehmen wir an, dass die Invalidisierung/Mutationsrate bei den 64-jährigen der Invalidisierung/Mutationsrate bei den 63-jährigen entspricht.

Der letzte Codeteil nach dem Kommentar `# Ändere die Zeilen mit sex == f" und alt == 64` fängt

ab, dass die Rentenaltererhöhung bei den Frauen über die Jahre 2025 bis 2028 gestaffelt um jeweils ein vierteljahr pro Jahr erfolgt. Wir passen also die Invalidisierung und Mutationsrate an die im jeweiligen Jahr total aufgelaufene Rentenaltererhöhung an.

4.2.4 Rentenbestand projizieren

Mit dem nachfolgenden Code werden die Bestände an Renten und Neurenten gemäss den Registerdaten ins Dataframe `RENTENBESTAND_IV` eingelesen:

```
RENTENBESTAND_IV <- BESTAND %>%
    mutate(bestand_tief = bestand, bestand_hoch = bestand, neurenten_tief = neurenten,
        neurenten_hoch = neurenten, bestandesmutationen_tief = bestandesmutationen,
        bestandesmutationen_hoch = bestandesmutationen, bestand_n_tief = bestand_n,
        bestand_n_hoch = bestand_n, neurenten_n_tief = neurenten_n,
        neurenten_n_hoch = neurenten_n, bestandesmutationen_n_tief =
        ↪  bestandesmutationen_n,
        bestandesmutationen_n_hoch = bestandesmutationen_n, quelle = "Register") %>%
    select(-contains("_hr"), -contains("_kr"), -contains("_vj"),
        -contains("_pfrt"))
```

Mit der nachfolgenden Code-Schleife werden rekursiv die Rentenbestände für die kommenden Jahre projiziert:

```
    #------ Projektionen für Rentenbestände berechnen ----------------------#
    for(lJahr in (jahr_projektion):PARAM_GLOBAL$jahr_ende) {

            RENTENBESTAND_NEU <- RENTENBESTAND_IV %>%
                filter(jahr == lJahr - 1,
                       quelle == ifelse(lJahr == jahr_projektion, "Register",
↪  "Projektion"),
                       (sex == "m" & alt <= 64) |
                           (sex == "f" & PARAM_GLOBAL$mitAHV21 == FALSE & alt <= 63) |
                           (sex == "f" & PARAM_GLOBAL$mitAHV21 == TRUE & lJahr < 2025 &
↪  alt <= 63) |
                           (sex == "f" & PARAM_GLOBAL$mitAHV21 == TRUE & lJahr >= 2025 &
↪  alt <= 64)
                ) %>%
                select(jahr, sex, alt, starts_with("bestand")) %>%
                full_join(BEVOELKERUNG %>%
                              group_by(jahr, sex, alt) %>%
                              summarize(bevendejahr = sum(bevendejahr)) %>%
                              ungroup() %>%
                              filter(alt>=17,
                                     (sex == "m" & alt <= 64) |
                                         (sex == "f" & PARAM_GLOBAL$mitAHV21 == FALSE &
↪  alt <= 63) |
                                         (sex == "f" & PARAM_GLOBAL$mitAHV21 == TRUE &
↪  lJahr < 2025 & alt <= 63) |
                                         (sex == "f" & PARAM_GLOBAL$mitAHV21 == TRUE &
↪  lJahr >= 2025 & alt <= 64),
                                     jahr==lJahr-1)
                ) %>%
                arrange(sex, alt) %>%
                mutate(jahr=jahr+1,
                       alt=alt+1
                ) %>%
```

```
        mutate(jahrX=jahr,
        ↪  jahr=ifelse(jahr<=(jahr_projektion+20),jahr,(jahr_projektion+20)))
        ↪  %>%
        left_join(IV_RATEN_INV_MUT) %>%
        select(-jahr) %>%
        rename(jahr=jahrX)
```

Die Schleife läuft vom ersten Projektionsjahr bis zum Ende des Projektionshorizont (in 2024 heisst das von 2024 bis 2070). Zuerst wird das Dataframe `RENTENBESTAND_NEU` aus dem Rentenbestand des Vorjahres gebildet (welcher im Projektionsjahr aus dem Register entnommen wird und in allen folgenden Jahren aus der Projektion für das vorangehende Jahr, d.h., der vorangehenden Iteration der Schleife). Danach werden nur relevante Bestände ausgewählt, das heisst, nur Männer bis 64 (da Männer, die Ende Jahr 65 sind per definition in die AHV übergegangen sind), und Frauen bis 2024 bis 63, und danach auch bis 64 (sofern `mitAHV21==TRUE` ist, was Stand 2024 der Fall ist).

Danach wird mit dem `full_join(BEVOELKERUNG` Befehl die relevante Bevölkerung am Ende des Jahres vor `lJahr` hinzugefügt.

Mit dem Befehl `mutate(jahrX=jahr` wird das eigentliche Projektionsjahr (also das `lJahr`) in der Variable `jahrX` gespeichert, während die Variable `jahr` so spezifiziert wird, dass sie mit der im Projektionsjahr gültigen Invalidisierung und Mutationsrate aus dem Dataframe `IV_RATEN_INV_MUT` (siehe Abschnitt 4.2.2) gejoined werden kann.

Danach werden mit dem folgenden, zweiten Teil der Schleife, die Bestände an Renten und Neurenten im `lJahr` berechnet:

```
        # converting NA to zero
        RENTENBESTAND_NEU[is.na(RENTENBESTAND_NEU)] <- 0

        RENTENBESTAND_NEU <- RENTENBESTAND_NEU %>%
            mutate(
                neurenten= (bevendejahr-bestand_n) * invalidisierung,
                bestandesmutationen = bestand * mutationsrate,
                bestand = bestand * (1 + mutationsrate) + neurenten,
                neurenten_tief= (bevendejahr-bestand_n_tief) * invalidisierung_tief,
                bestandesmutationen_tief = bestand_tief * mutationsrate,
                bestand_tief = bestand_tief * (1 + mutationsrate) + neurenten_tief,
                neurenten_hoch= (bevendejahr-bestand_n_hoch) * invalidisierung_hoch,
                bestandesmutationen_hoch = bestand_hoch * mutationsrate,
                bestand_hoch = bestand_hoch * (1 + mutationsrate) + neurenten_hoch,
                neurenten_n= (bevendejahr-bestand_n) * invalidisierung_n,
                bestandesmutationen_n = bestand_n * mutationsrate_n,
                bestand_n = bestand_n * (1 + mutationsrate_n) + neurenten_n,
                neurenten_n_tief = (bevendejahr-bestand_n_tief) *
                ↪  invalidisierung_tief_n,
                bestandesmutationen_n_tief = bestand_n_tief * mutationsrate_n,
                bestand_n_tief = bestand_n_tief * (1 + mutationsrate_n) +
                ↪  neurenten_n_tief,
                neurenten_n_hoch = (bevendejahr-bestand_n_hoch) *
                ↪  invalidisierung_hoch_n,
                bestandesmutationen_n_hoch = bestand_n_hoch * mutationsrate_n,
                bestand_n_hoch = bestand_n_hoch * (1 + mutationsrate_n) +
                ↪  neurenten_n_hoch
            ) %>%
            ungroup()
```

```
        RENTENBESTAND_NEU <- RENTENBESTAND_NEU %>%
            select(-bevendejahr, -contains("mutationsrate"),
            ↪  -contains("invalidisierung"), -contains("invalidisierung_tief"),
            ↪  -contains("invalidisierung_hoch")) %>%
            mutate(quelle="Projektion")

        #------ Step 3b: an Ergebnistabelle anfügen ---------------------------#
        RENTENBESTAND_IV   <- RENTENBESTAND_IV %>%
            rbind(., RENTENBESTAND_NEU)
    }
```

Die Berechnung der Bestände an Renten und Neurenten in den Projektionsjahren ist analog zur in Abschnitt 4.2.1 erläuterten Berechnung der Bestände an Renten und Neurenten aus den Registerdaten.

Zum Schluss werden mit dem folgenden Code Rentenbestände und Neurenten gemäss dem gewählten Szenario ausgewählt:

```
RENTENBESTAND_IV <- RENTENBESTAND_IV %>%
    mutate(bestand = case_when(PARAM_GLOBAL$invalidisierung_szenario ==
        "mittel" ~ bestand, PARAM_GLOBAL$invalidisierung_szenario ==
        "tief" ~ bestand_tief, PARAM_GLOBAL$invalidisierung_szenario ==
        "hoch" ~ bestand_hoch, TRUE ~ bestand  # default to existing iv_renten if needed
),
        neurenten = case_when(PARAM_GLOBAL$invalidisierung_szenario ==
            "mittel" ~ neurenten, PARAM_GLOBAL$invalidisierung_szenario ==
            "tief" ~ neurenten_tief, PARAM_GLOBAL$invalidisierung_szenario ==
            "hoch" ~ neurenten_hoch, TRUE ~ neurenten  # default to existing neurenten if
            ↪  needed
)) %>%
    select(jahr, sex, alt, bestand, neurenten)
```

4.2.5 Rentenausgaben projizieren

Unser Modell projiziert die gesamten Rentenausgaben (exklusive Nachzahlungen). Da unser Finanzperspektivenmodell aber die gleiche Struktur hat wie die IV-Jahresrechnung, benötigen wir separate Projektionen für gewisse Unterkategorien der Rentenausgaben, bspw. die Aufteilung der Rentenaufgaben auf Ausgaben für orderntliche und ausserordentliche Renten. Die Anteile der verschiedenen Rentenausgaben-Unterkategorien werden mit dem folgenden Code berechnet:

```
ANTEILE <- IV_ABRECHNUNG %>%
    filter(jahr >= jahr_projektion - 3, jahr < jahr_projektion) %>%
    left_join(RENTENBESTAND_IV %>%
        group_by(jahr) %>%
        summarize(bestand = sum(bestand), neurenten = sum(neurenten)) %>%
        ungroup() %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(renten = bestand * minimalrente * 12, neurenten = neurenten *
            minimalrente * 12) %>%
        filter(jahr >= jahr_projektion - 3, jahr < jahr_projektion)) %>%
    mutate(frac_rent_ord = rent_ord/renten, frac_rent_aord = rent_aord/renten,
        frac_rent_ord_nachz = rent_ord_nachz/neurenten, frac_rent_aord_nachz =
        ↪  rent_aord_nachz/neurenten,
```

```
        frac_ausl_rueck = ausl_rueck/renten, frac_ausl_fs_leist = ausl_fs_leist/renten,
        frac_rueck_erstattungsf = rueck_erstattungsf/renten,
        frac_abschr_rueck_erstattungsf = abschr_rueck_erstattungsf/renten) %>%
    summarize(frac_rent_ord = mean(frac_rent_ord), frac_rent_aord = mean(frac_rent_aord),
        frac_rent_ord_nachz = mean(frac_rent_ord_nachz), frac_rent_aord_nachz =
        ↪  mean(frac_rent_aord_nachz),
        frac_ausl_rueck = mean(frac_ausl_rueck), frac_ausl_fs_leist =
        ↪  mean(frac_ausl_fs_leist),
        frac_rueck_erstattungsf = mean(frac_rueck_erstattungsf),
        frac_abschr_rueck_erstattungsf = mean(frac_abschr_rueck_erstattungsf)) %>%
    ungroup()
```

D.h. wir berechnen für die letzten drei Jahre, welchen Anteil an den Rentenausgaben gemäss den Registerdaten die verschiedenen Rentenausgabenpositionen der IV-Abrechnung ausgemacht haben. Konkret nehmen wir in obigem Code zuerst die IV-Abrechnungen der letzten 3 Jahre, und fügen den gesamten Bestand an Renten und Neurenten (d.h. aggregiert über alle Alter und Geschlechter) an, wobei wir den Rentenbestand und die Neurenten in Jahresrenten in CHF umrechnen (mit den Formeln `renten=bestand*minimalrente*12` respektive `neurenten=neurenten*minimalrente*12`). Danach berechnen wir für die verschiedenen Rentenausgabenpositionen (`rent_ord`, `rent_aord`, ...), welchen Anteil an den Rentenausgaben gemäss den Registerdaten diese in den jeweiligen Jahren ausgemacht haben. Eine Ausnahme bilden die Rentennachzahlungen. Rentennachzahlungen werden zum allergrössten Teil bei Neurentnern geleistet. Aus diesem Grund bilden wir die Anteile für die Rentennachzahlungen (`frac_rent_ord_nachz` und `frac_rent_aord_nachz`) als Anteil der Ausgaben für Neurenten in einem Jahr.[27] Zum Schluss bilden wir den Mittelwert der Anteile über die letzten drei Jahre.

Die Rentenausgaben gemäss den verschiedenen Positionen werden dann schlussendlich mit dem folgenden Code berechnet:

```
RENTENSUMME_IV <- RENTENBESTAND_IV %>%
    group_by(jahr) %>%
    summarize(bestand = sum(bestand), neurenten = sum(neurenten)) %>%
    ungroup() %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(renten = bestand * minimalrente * 12, neurenten = neurenten *
        minimalrente * 12) %>%
    mutate(rent_ord = renten * ANTEILE$frac_rent_ord, rent_aord = renten *
        ANTEILE$frac_rent_aord, rent_ord_nachz = neurenten *
        ANTEILE$frac_rent_ord_nachz, rent_aord_nachz = neurenten *
        ANTEILE$frac_rent_aord_nachz, ausl_rueck = renten * ANTEILE$frac_ausl_rueck,
        ausl_fs_leist = renten * ANTEILE$frac_ausl_fs_leist,
        rueck_erstattungsf = renten * ANTEILE$frac_rueck_erstattungsf,
        abschr_rueck_erstattungsf = renten * ANTEILE$frac_abschr_rueck_erstattungsf) %>%
    select(jahr, rent_ord, rent_ord_nachz, rent_aord, rent_aord_nachz,
        ausl_rueck, ausl_fs_leist, rueck_erstattungsf, abschr_rueck_erstattungsf) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    rbind(IV_ABRECHNUNG %>%
        select(jahr, rent_ord, rent_ord_nachz, rent_aord, rent_aord_nachz,
            ausl_rueck, ausl_fs_leist, rueck_erstattungsf, abschr_rueck_erstattungsf))
↪   %>%
    mutate(rentensumme_iv = rent_ord + rent_ord_nachz + rent_aord +
        rent_aord_nachz + ausl_rueck + ausl_fs_leist + rueck_erstattungsf +
```

[27] Die Korrelation zwischen den Ausgaben für Neurenten und den Rentennachzahlungen betrug über die Jahre 2014-2023 0.96 (Nachzahlungen für ordentliche Renten) und 0.94 (Nachzahlungen für ausserordentliche Renten).

```
        abschr_rueck_erstattungsf) %>%
    arrange(jahr)
```

Wir aggregieren in einem ersten Schritt den gesamten Rentenbestand respektive die gesamten Neurenten pro Jahr (`summarize(bestand...)`, und rechnen dann wiederum den Rentenbestand in jährliche Rentenausgaben um (`mutate(renten)`. Danach teilen wir die Rentenausgaben, sowie die Neurentenausgaben auf die einzelnen Rentenausgabenpositionen auf. Der restliche Code dient zur Anordnung der Daten, sowie zur Berechnung der Rentensumme als Summe aller Rentenausgabenpositionen.

Der nachfolgende Code ist nicht zentral für die Berechnung des Finanzhaushalts, da das dadurch generierte Dataframe `BESTAND_IV` nicht für die Berechnung des Finanzperspektivenmodells genutzt wird (das Dataframe dient nur zur Kostenabschätzung der Tabellenlöhne). Er wird daher nachfolgend aufgeführt aber nicht näher erläutert.

```
BESTAND_IV <- RENTENBESTAND_IV %>%
    mutate(iv_r_malt = bestand * alt, iv_nr_alt = neurenten *
        alt) %>%
    group_by(jahr, sex) %>%
    summarize(iv_renten = sum(bestand), iv_neurenten = sum(neurenten),
        iv_renten_alt = sum(iv_r_malt), iv_neurenten_alt = sum(iv_nr_alt)) %>%
    ungroup() %>%
    rbind(., RENTENBESTAND_IV %>%
        mutate(iv_r_malt = bestand * alt, iv_nr_alt = neurenten *
            alt) %>%
        mutate(sex = "total") %>%
        group_by(jahr, sex) %>%
        summarize(iv_renten = sum(bestand), iv_neurenten = sum(neurenten),
            iv_renten_alt = sum(iv_r_malt), iv_neurenten_alt = sum(iv_nr_alt)) %>%
        ungroup()) %>%
    mutate(iv_renten_alt = iv_renten_alt/iv_renten, iv_neurenten_alt =
    ↪  iv_neurenten_alt/iv_neurenten)

#------ Wachstumsaten ---------------------------------------------------#
BESTAND_IV <- BESTAND_IV %>%
    inner_join(BESTAND_IV %>%
        mutate(jahr = jahr + 1) %>%
        select(jahr, sex, iv_renten, iv_neurenten), by = c("jahr",
        "sex"), suffix = c("", "_vj")) %>%
    mutate(iv_renten_wr = iv_renten/iv_renten_vj, iv_neurenten_wr =
    ↪  iv_neurenten/iv_neurenten_vj) %>%
    select(-iv_renten_vj, -iv_neurenten_vj)
```

Somit können die Rentenausgabeprojektionen und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_geldleistungen.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
mod_return(RENTENSUMME_IV, RENTENBESTAND_IV, BESTAND_IV)
```

4.3 Modul mod_iv_taggelder.R

Dokumentation zuletzt aktualisiert am 24.09.2024

Das Prognosemodell zu den Taggeldern ist im Skript `mod_iv_taggelder.R` enthalten, welches wie folgt im Skript `mod_iv_geldleistungen.R` aufgerufen wird:

```
tl_iv_taggelder <- mod_iv_taggelder(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_taggelder.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

In einem ersten Schritt wird das Modul `mod_init()` ausgeführt:

```
#------ Initialisierung / testen der Input Parameter -----------------#
mod_init()
```

Das Hilfsmodul `mod_init()` nimmt keine Berechnungen vor, sondern überprüft lediglich die Struktur der verwendeten Dataframes und gibt bei Fehlern in den Input-Daten eine Fehlermeldung aus respektive stoppt die Ausführung des Codes. `mod_init()` wird in mehreren Modulen am Anfang aufgerufen. Da `mod_init()` auf die Berechnungen keinen Einfluss hat wird es hier nicht näher erläutert, und im Rest dieser Dokumentation nicht mehr erwähnt.

Im Modul wird in einem ersten Schritt der Taggelder-Vektor gemäss IV-Abrechnung ausgelesen, sowie die Taggelder-Summe im letzten Abrechnungsjahr:

```
#------ Startwerte aus dem Abrechnungsjahr ---------------------------#
TAGGELDER_IV_ABR <- IV_ABRECHNUNG %>%
    select(jahr, tg_geld)

iv_tag_init <- IV_ABRECHNUNG %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
    select(tg_geld) %>%
    as.numeric()
```

Danach werden die Taggelder für die Jahre bis 2026 mit einem in `PARAM_GLOBAL`spezifizierten Vektor (Stand 2024), und ab 2027 mit dem Lohnsummen-wachstum fortgeschrieben. Die gesonderte Fortschreibung der Taggelder 2022-2026 ist durch einen einmaligen Efekt der Weiterentwicklung der IV begründet: Der Grund sind Kürzungen im Zuge der Weiterentwicklung der IV im Jahr 2022, welche erst um das Jahr 2026 ihre volle Wirkung entfalten.[28] Im Code sieht das dann wie folgt aus:

```
TAGGELDER_IV <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsumme, preis) %>%
    left_join(data.frame(cbind(jahr =
    ↪  c(1:length(as.numeric(strsplit(PARAM_GLOBAL$wr_taggeld_fp,
        ", ")[[1]]))) + 2021, erh_tg = as.numeric(strsplit(PARAM_GLOBAL$wr_taggeld_fp,
        ", ")[[1]]))), by = "jahr") %>%
    mutate(lohnsumme = if_else(!is.na(erh_tg) | is.na(lohnsumme),
        0, lohnsumme), erh_tg = if_else(is.na(erh_tg), 1, erh_tg *
        (1 + preis/100))) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    mutate(lohnsummeidx = cumprod(1 + lohnsumme/100), tgidx = cumprod(erh_tg),
        totidx = cumprod((1 + lohnsumme/100) * erh_tg)) %>%
    mutate(tg_geld = totidx * iv_tag_init) %>%
    select(jahr, tg_geld)
```

[28] Zu beachten ist, dass Taggelder zum Teil über längere Zeit ausbezahlt werden, insbesondere auch die von der Weiterentwicklung der IV betroffenen Taggelder bei ehemaliger beruflicher Ausbildung. Dies er-klärt, weshalb der Effekt der Weiterentwicklung der IV auch längere Zeit nach deren Einführung anhält.

```
TAGGELDER_IV <- rbind(TAGGELDER_IV_ABR, TAGGELDER_IV)
```

Der Befehl `left_join(data.frame` dient dazu, den Vektor für die reale Entwicklung der Taggelder zwischen 2022 und 2026 gemäss `PARAM_GLOBAL` anzufügen. In `PARAM_GLOBAL` wurde die Variable `wr_taggeld_fp` Stand 2024 so definiert, dass sie für 2022 und 2023 1 + der beobachteten Wachstumsrate der Taggelder-Ausgaben entspricht, und für 2024 bis 2026 der projizierten Wachstumsrate der Taggelder. Für 2022 und 2023 entspricht das beobachtete Wachstum der Taggelder -8.6% respektive -5.0%. Stand 2024 stellen wir fest, dass der Rückgang immer geringer wird, was auch zu erwarten ist, da der Effekt der Weiterentwicklung der IV mit der Zeit nachlässt. Aus diesem Grund schreiben wir den Rückgang wie folgt fort: Für 2024 als $\frac{-5.0\%}{-8.6\%} * -5.0\%$, für 2025 als $(\frac{-5.0\%}{-8.6\%})^2 * -5.0\%$, und für 2026 als $(\frac{-5.0\%}{-8.6\%})^3 * -5.0\%$. Diese Berechnungen sind im Code oben nicht ersichtlich, da die Zahlen für die Fortschreibung bis 2026 direkt aus `PARAM_GLOBAL` eingelesen werden.

Für die Zeit ab 2027 wird die Lohnsumme, also die projizierte Summe der insgesamt in der Schweiz ausbezahlten Löhne, für die Fortschreibung verwendet. Dies ist im unteren Teil des Codes implementiert. Siehe Kapitel 4.18 für Informationen zur Berechnung der projizierten Lohnsumme.

Somit können die Taggelderprojektionen an das Finanzperspektivenmodell (respektive das Modul `mod_iv_geldleistungen.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
mod_return(TAGGELDER_IV)
```

4.4 Modul mod_iv_he_NEW.R

Dokumentation zuletzt aktualisiert am 10.10.2024

Das Modul zu den Ausgaben für Hilflosenentschädigungen und den Intensivpflegezuschlag `mod_iv_he_NEW.R` ist analog zum Modul `mod_iv_rentensumme_new.R` aufgebaut. Spezifisch ist, dass das Modell für die Kinder-Hilflosenentschädigung und den Intensivpflegezuschlag und das Modell für die Erwachsenen-Hilflosenentschädigung separat berechnet werden müssen, da die Daten für diese beiden Leistungen nicht verknüpft werden können. Das füht dazu, dass der Code in `mod_iv_he_NEW.R` die gleiche Struktur hat wie der Code in `mod_iv_rentensumme_new.R`, mit dem Unterschied dass jede Berechnung zweimal durchgeführt wird, je einmal für das Modell für die Kinder-Hilflosenentschädigung und den Intensivpflegezuschlag und einmal für das Modell für die Erwachsenen-Hilflosenentschädigung.

Das Prognosemodell zu den Ausgaben für Hilflosenentschädigungen (HE) wird wie folgt im Skript `mod_iv_geldleistungen.R` aufgerufen wird:

```
tl_iv_he <- mod_iv_he_NEW(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEVOELKERUNG = BEVOELKERUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    IV_HE_NEW = IV_HE_NEW, IV_HE_KINDER = IV_HE_KINDER)
```

Das Modul `mod_iv_he_NEW.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `BEVOELKERUNG`: Die Bevölkerungsdaten und -szenarien weren dazu genutzt, um die Anzahl Invaliden nach Alter in der Zukunft zu schätzen.
- `RENTENENTWICKLUNG`: Dataframe mit der Entwicklung der Minimalrente.
- `IV_RENTEN_NEW`: Dataframe mit Details zum Bestand an HE-Beziehenden sowie der neuen HE-Beziehenden nach Jahr, Alter und Geschlecht.
- `IV_HE_KINDER`: Dataframe mit Details zum Bestand an Beziehenden von Kinder-HE und Intensivpflegezuschlag sowie der neuen Beziehenden nach Jahr, Alter und Geschlecht.

In einem ersten Schritt werden in `mod_iv_he_NEW` Parameterwerte für die weiteren Berechnungen gesetzt:[29]

```
jahr_projektion <- PARAM_GLOBAL$jahr_abr + 1

# Folgende Parameter könnten verwendet werden, um die
# Invalidisierung bei Jungen bis Alter X anders zu
# berechnen
junge_anders <- FALSE
junge_alter_bis <- 30
junge_anders_MA_lastyear <- 2017
```

4.4.1 Berechnung von HE-Bestand, neuen HE-Beziehenden, und Mutationen

Die Berechnung des HE-Bestandes sowie der neuen HE-Beziehenden in Anzahl Minimalrenten erfolgt mit dem folgenden Code:

```
BESTAND_HE_KINDER <- IV_HE_KINDER %>%
    filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(bestand = mpr/minimalrente, neurenten = mpr_n/minimalrente) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten")) %>%
    arrange(jahr, sex, alt)

BESTAND_HE <- IV_HE_NEW %>%
    filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(bestand = mpr/minimalrente, neurenten = mpr_n/minimalrente) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten")) %>%
    arrange(jahr, sex, alt)
```

Dieser Codeteil dividiert die total im Dezember eines Jahres ausbezahlten HE-Leistungen (bei der Kinder-HE schliesst dies hier und in allen folgenden Berechnungen den Intensivpflegezuschlag mit ein) `mpr` mit der in diesem Jahr gültigen Minimalrente (bspw. 1225 für das Jahr 2023), um den HE-Bestand in Anzahl Minimalrenten zu erhalten. Gleichsam dividiert es die neu ausbezahlte HEs `mpr_n`, also die HEs, die im Dezember dieses Jahres ausbezahlt wurden aber nicht im Dezember des Vorjahres, mit der in diesem Jahr gültigen Minimalrente, um die Neu-HEs in Anzahl Minimalrenten zu erhalten.

Die Berechnung der Bestandesmutationen erfolgt mit den folgenden zwei Codeteilen:

```
BESTAND_HE_KINDER <- BESTAND_HE_KINDER %>%
    filter(jahr != min(BESTAND_HE_KINDER$jahr)) %>%
    full_join(BESTAND_HE_KINDER %>%
        filter(jahr != max(BESTAND_HE_KINDER$jahr)) %>%
        mutate(alt = alt + 1, jahr = jahr + 1) %>%
        rename(bestand_vj = bestand, bestand_vj_n = bestand_n) %>%
        select(jahr, sex, alt, contains("bestand_vj"))) %>%
    mutate_if(is.numeric, ~coalesce(., 0)) %>%
    mutate(bestandesmutationen = bestand - bestand_vj - neurenten,
        bestandesmutationen_n = bestand_n - bestand_vj_n - neurenten_n) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten"),
        contains("bestandesmutationen")) %>%
    arrange(jahr, sex, alt)
```

[29] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
BESTAND_HE <- BESTAND_HE %>%
    filter(jahr != min(BESTAND_HE$jahr)) %>%
    full_join(BESTAND_HE %>%
        filter(jahr != max(BESTAND_HE$jahr)) %>%
        mutate(alt = alt + 1, jahr = jahr + 1) %>%
        rename(bestand_vj = bestand, bestand_vj_n = bestand_n) %>%
        select(jahr, sex, alt, contains("bestand_vj"))) %>%
    mutate_if(is.numeric, ~coalesce(., 0)) %>%
    mutate(bestandesmutationen = bestand - bestand_vj - neurenten,
        bestandesmutationen_n = bestand_n - bestand_vj_n - neurenten_n) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten"),
        contains("bestandesmutationen")) %>%
    arrange(jahr, sex, alt)
```

Dieser Codeteil fügt dem HE-Bestand aus dem Vorjahr an das Dataframe mit dem aktuellen HE-Bestand an. Da dieselben Personen ein Jahr vorher ein Jahr jünger waren, wird zum Joinen nicht nur ein Jahr addiert, sonder auch ein Alter (bspw. die gleichen Frauen, die in 2023 30 Jahre alt sind waren im 2022 29 Jahre alt). Es wird sowohl der Bestand in Minimalrenten des Vorjahres (`bestand_vj`) als auch der Bestand in Anzahl RentenBeziehende des Vorjahres (`bestand_vj_n`) angehängt. Die Zeile `mutate_if(is.numeric, ~ coalesce(., 0))` stellt sicher, dass Nullbestände als 0 in den Daten erscheinen (und nicht als NA).

Die Bestandesmutationen werden oben mit dem letzten Codeteil

```
mutate(bestandesmutationen = bestand - bestand_vj - neurenten,
    bestandesmutationen_n = bestand_n - bestand_vj_n - neurenten_n) %>%
    select(jahr, sex, alt, contains("bestand"), contains("neurenten"),
        contains("bestandesmutationen")) %>%
    arrange(jahr, sex, alt)
```

berechnet. D.h. die Bestandesmutationen (in Minimalrenten `bestandesmutationen` und in Anzahl Beziehende `bestandesmutationen_n`) ergeben sich aus dem Bestand im Dezember, abzüglich des Bestandes, der auf Neu-Beziehende zurückzuführen ist, und abzüglich des Bestandes des Vorjahres. Bestandesmutationen widerspiegeln daher Veränderungen des HE-Bestandes, die nicht auf Neu-Beziehende zurückzuführen sind. Beispiel: Wenn der Bestand an 30 Jährigen Frauen in diesem Jahr 120 beträgt, davon 40 Neu-Beziehende sind, und der Bestand dieser Frauen im Vorjahr 100 betrug, so heisst dies, dass 20 aus dem Bestand abgegangen sind, und somit betragen die Bestandesmutationen -20.

4.4.2 Berechnung der Rate der HE-Invalidisierung und der HE-Mutationsrate

Die HE-Invalidisierungs- und HE-Mutationsraten werden wie folgt berechnet:

```
RATEN_INV_MUT_HE_KINDER <- BESTAND_HE_KINDER %>%
    left_join(bind_rows(BEVOELKERUNG %>%
        mutate(bevendejahr = bevendejahr/2) %>%
        filter(alt == 0), BEVOELKERUNG %>%
        mutate(jahr = jahr + 1, alt = alt + 1)) %>%
        group_by(jahr, sex, alt) %>%
        summarize(bevendejahr = sum(bevendejahr, na.rm = TRUE))) %>%
    mutate(invalidisierung = neurenten/(bevendejahr - bestand_vj_n),
        invalidisierung_n = neurenten_n/(bevendejahr - bestand_vj_n),
        mutationsrate = case_when(bestand_vj == 0 ~ 0, TRUE ~
            bestandesmutationen/bestand_vj), mutationsrate_n = case_when(bestand_vj_n ==
            0 ~ 0, TRUE ~ bestandesmutationen_n/bestand_vj_n)) %>%
    select(jahr, sex, alt, contains("invalidisierung"), contains("mutationsrate"))
```

```
RATEN_INV_MUT_HE <- BESTAND_HE %>%
    left_join(BEVOELKERUNG %>%
        mutate(jahr = jahr + 1, alt = alt + 1) %>%
        group_by(jahr, sex, alt) %>%
        summarize(bevendejahr = sum(bevendejahr, na.rm = TRUE))) %>%
    mutate(invalidisierung = neurenten/(bevendejahr - bestand_vj_n),
        invalidisierung_n = neurenten_n/(bevendejahr - bestand_vj_n),
        mutationsrate = case_when(bestand_vj == 0 ~ 0, TRUE ~
            bestandesmutationen/bestand_vj), mutationsrate_n = case_when(bestand_vj_n ==
            0 ~ 0, TRUE ~ bestandesmutationen_n/bestand_vj_n)) %>%
    select(jahr, sex, alt, contains("invalidisierung"), contains("mutationsrate"))
```

Die HE-Invalidisierung im Jahr t berechnet sich als der Anteil der Wohnbevölkerung, der Anfang des Jahres t noch keine HE bezieht aber Ende des Jahres t eine HE bezieht. Daher wird in einem ersten Schritt die Bevölkerung am Ende des Vorjahres (=Bevölkerung am Anfang des aktuellen Jahres), die am Ende des aktuellen Jahres ein Jahr älter sein wird, mit dem Befehl `left_join(BEVOELKERUNG` angefügt.

Die HE-Invalidisierung im Jahr t ergibt sich dann aus den Neu-Beziehenden dividiert durch die Bevölkerung am Ende des Vorjahres abzüglich der Anzahl Personen, die bereits am Ende des Vorjahres eine HE bezogen haben. Die Bestandesmutationen ergeben sich aus dem Quotienten der Bestandsmutationen im Jahr t und des Bestandes am Ende des Vorjahres.

4.4.3 HE-Invalidisierungs- und HE-Mutationsraten für die Projektionen

Die durchschnittlichen HE-Invalidisierungs- und HE-Mutationsraten, die danach für die Projektionen verwendet werden, werden wie folgt berechnet. Die Berechnungen werden nachfolgend anhand der Erwachsenen-HE erläuter. Die Berechnungen für die Kinder-HE sind identisch:

```
    IV_RATEN_INV_MUT_RAW <- RATEN_INV_MUT_HE %>%
        filter(
            if (junge_anders) {
                (alt <= junge_alter_bis & jahr >= junge_anders_MA_lastyear -
↪   (PARAM_GLOBAL$he_MA_years-1) & jahr <= junge_anders_MA_lastyear) |
                    (alt > junge_alter_bis & jahr >= PARAM_GLOBAL$jahr_abr -
↪   (PARAM_GLOBAL$he_MA_years-1) & jahr <= PARAM_GLOBAL$jahr_abr)
            } else {
                jahr >= PARAM_GLOBAL$jahr_abr - (PARAM_GLOBAL$he_MA_years-1) & jahr <=
↪   PARAM_GLOBAL$jahr_abr # MA_years-1 beschreibt die Anzahl Jahre vor jahr_abr, die
↪   verwendet werden sollen
            }
        ) %>%
        group_by(sex, alt) %>%
        summarize(
            invalidisierung = mean(invalidisierung, na.rm = TRUE),
            mutationsrate = mean(mutationsrate, na.rm = TRUE),
            invalidisierung_n = mean(invalidisierung_n, na.rm = TRUE),
            mutationsrate_n = mean(mutationsrate_n, na.rm = TRUE),
        ) %>%
        ungroup() %>%
```

Der `if`-Teil würde dem Fall dienen, dass für Junge HE-Beziehende ein anderer Berechnungshorizont für die HE-Invalidisierung angewendet werde soll. Da `junge_anders` Stand 2024 `==FALSE` ist, wird nicht auf diesen Teil eingegangen. Der Teil nach `else` wählt für die Berechnung der HE-Invalidisierung und HE-Mutationsrate die spezifizierte Anzahl Jahre vor dem letzten Abrechnungsjahr aus (Stand 2024 nutzen wir 3 Jahre, also die

Jahre 2021, 2022, und 2023).

Danach wird die HE-Invalidisierung/HE-Mutationsrate aus der durchschnittlichen HE-Invalidisierung/HE-Mutationsrate der ausgewählten Jahre gebildet.

Der folge Codeteil dient zur Berechnung der Wachstumsrate der Invalidsierung. Diese Wachstumsrate wird ausschliesslich für die Szenarien „hoch“ und „tief“ gebildet und wird nur verwendet, falls die HE-Invalidisierung in den Szenarien auch variiert werden soll (D.h., falls der Parameter `PARAM_GLOBAL$he_szenario==TRUE` ist, was Stand 2024 nicht der Fall ist):

```
        left_join(
            RATEN_INV_MUT_HE%>%
                filter(
                    if (junge_anders) {
                        (alt <= junge_alter_bis & jahr >= junge_anders_MA_lastyear -
↪  (PARAM_GLOBAL$he_szenariolag-1) & jahr <= junge_anders_MA_lastyear) |
                            (alt > junge_alter_bis & jahr >= PARAM_GLOBAL$jahr_abr -
↪  (PARAM_GLOBAL$he_szenariolag-1) & jahr <= PARAM_GLOBAL$jahr_abr)
                    } else {
                        jahr >= PARAM_GLOBAL$jahr_abr - (PARAM_GLOBAL$he_szenariolag-1) &
↪  jahr <= PARAM_GLOBAL$jahr_abr # szenariolag-1 beschreibt die Anzahl Jahre vor
↪  jahr_abr, die verwendet werden sollen
                    }
                ) %>%
                arrange(sex, alt, jahr) %>%
                group_by(sex, alt) %>%
                mutate(
                    invalidisierung_diff = c(NA, abs(diff(invalidisierung))),
                    # mutationsrate_diff = c(NA, abs(diff(mutationsrate))) ,
                    invalidisierung_diff_n = c(NA, abs(diff(invalidisierung_n)))
                ) %>%
                summarise(
                    mean_abs_invalidisierung_diff = mean(invalidisierung_diff, na.rm =
                     ↪  TRUE),
                    # mean_abs_mutationsrate_diff = mean(mutationsrate_diff, na.rm =
                     ↪  TRUE) ,
                    mean_abs_invalidisierung_diff_n = mean(invalidisierung_diff_n, na.rm
                     ↪  = TRUE)
                )  %>%
                mutate(
                    mean_abs_invalidisierung_diff = mean_abs_invalidisierung_diff/3,
                    mean_abs_invalidisierung_diff_n = mean_abs_invalidisierung_diff_n/3
                )  %>%
                ungroup(),
            by = c("sex", "alt")
        ) %>%
```

Der `if`-Teil würde wiederum dem Fall dienen, dass für Junginvalide ein anderer Berechnungshorizont für die HE-Invalidisierung angewendet werde soll. Da `junge_anders` Stand 2024 `==FALSE` ist, wird nicht auf diesen Teil eingegangen. Der Teil nach `else` wählt für die Berechnung der Szenarien die spezifizierte Anzahl Jahre vor dem letzten Abrechnungsjahr aus (Stand 2024 nutzen wir 5 Jahre, also die Jahre 2019, 2020, 2021, 2022, und 2023).

Danach bilden wir die absolute Jahr-zu-Jahr Veränderung der HE-Invalidisierung über den ausgewählten Zeitraum (die HE-Mutationsrate ist auskommentiert, da Stand 2024 die Szenarien nur auf die Invalidisierungsrate, nicht aber auf die Mutationsrate, abstützten). Zum Schluss bilden wir den Mittelwert über die

absolute Jahr-zu-Jahr Veränderung der HE-Invalidisierung.

Der folgende Codeteil wäre für den Fall, dass das vergangene Wachstum der HE-Invalidisierung für weitere Jahre fortgeschrieben werden soll. Da diese Option Stand 2024 nicht aktiviert ist (d.h. `PARAM_GLOBAL$fort_entw==FALSE`) wird nicht weiter auf diesen Codeteil eingegangen.

```
left_join(RATEN_INV_MUT_HE %>%
    filter(if (junge_anders) {
        (alt <= junge_alter_bis & (jahr == junge_anders_MA_lastyear |
            jahr == junge_anders_MA_lastyear - PARAM_GLOBAL$he_wachstum_lagyears)) |
            (alt > junge_alter_bis & (jahr == PARAM_GLOBAL$jahr_abr |
                jahr == PARAM_GLOBAL$jahr_abr - PARAM_GLOBAL$he_wachstum_lagyears))
    } else {
        jahr == PARAM_GLOBAL$jahr_abr | jahr == PARAM_GLOBAL$jahr_abr -
            PARAM_GLOBAL$he_wachstum_lagyears
    }) %>%
    arrange(sex, alt, jahr) %>%
    mutate(rank = min_rank(jahr)) %>%
    mutate(period = ifelse(rank == 1, 0, 1)) %>%
    group_by(sex, alt) %>%
    mutate(invalidisierung_growth = c(NA,
    ↪  diff(invalidisierung))/PARAM_GLOBAL$he_wachstum_lagyears,
        mutationsrate_growth = c(NA,
        ↪  diff(mutationsrate))/PARAM_GLOBAL$he_wachstum_lagyears,
        invalidisierung_growth_n = c(NA,
        ↪  diff(invalidisierung_n))/PARAM_GLOBAL$he_wachstum_lagyears,
        mutationsrate_growth_n = c(NA,
        ↪  diff(mutationsrate_n))/PARAM_GLOBAL$he_wachstum_lagyears,
        ) %>%
    ungroup() %>%
    filter(period == 1) %>%
    select(sex, alt, contains("_growth")), by = c("sex", "alt"))
```

Wie oben schon erwähnt werden die exakt gleichen Berechnungen für die Kinder-HE durchgeführt, d.h., unter Verwendung von `RATEN_INV_MUT_HE_KINDER` statt `RATEN_INV_MUT_HE`.

Mit dem folgenden Codeteil werden die Projektionsjahr-spezifischen HE-Invalidisierungsraten gebildet. Dieser Codeteil ist relevant, da die Szenarien „tief“ und „hoch“ über die kommenden 20 Jahre ansteigende/abfallende HE-Invalidisierungsraten verwenden.

```
    # Initialisiere ein leeres Dataframe vor der Schleife
    IV_RATEN_INV_MUT_HE<- tibble()

    for(lJahr in (jahr_projektion):(jahr_projektion+20)) {

        IV_RATEN_INV_MUT_JAHR <-
            IV_RATEN_INV_MUT_RAW %>%
            mutate(
                invalidisierung = invalidisierung + PARAM_GLOBAL$he_fort_entw *
                ↪  invalidisierung_growth *((ifelse(lJahr - jahr_projektion <
                ↪  PARAM_GLOBAL$he_fort_entw_years, lJahr - jahr_projektion + 1,
                ↪  PARAM_GLOBAL$he_fort_entw_years))),
                invalidisierung_n = invalidisierung_n + PARAM_GLOBAL$he_fort_entw *
                ↪  invalidisierung_growth_n *((ifelse(lJahr - jahr_projektion <
                ↪  PARAM_GLOBAL$he_fort_entw_years, lJahr - jahr_projektion + 1,
                ↪  PARAM_GLOBAL$he_fort_entw_years))),
```

```
            mutationsrate = mutationsrate + PARAM_GLOBAL$he_fort_entw *
            ↪  mutationsrate_growth * ((ifelse(lJahr - jahr_projektion <
            ↪  PARAM_GLOBAL$he_fort_entw_years, lJahr - jahr_projektion + 1,
            ↪  PARAM_GLOBAL$he_fort_entw_years))),
            mutationsrate_n = mutationsrate_n + PARAM_GLOBAL$he_fort_entw *
            ↪  mutationsrate_growth_n * ((ifelse(lJahr - jahr_projektion <
            ↪  PARAM_GLOBAL$he_fort_entw_years, lJahr - jahr_projektion + 1,
            ↪  PARAM_GLOBAL$he_fort_entw_years))),
            invalidisierung_tief = invalidisierung + mean_abs_invalidisierung_diff *
            ↪  (1 + (lJahr - jahr_projektion) * (1/20)),
            invalidisierung_hoch = invalidisierung - mean_abs_invalidisierung_diff *
            ↪  (1 + (lJahr -  jahr_projektion) * (1/20)),
            invalidisierung_tief_n = invalidisierung_n +
            ↪  mean_abs_invalidisierung_diff_n * (1 + (lJahr - jahr_projektion) *
            ↪  (1/20)),
            invalidisierung_hoch_n = invalidisierung_n -
            ↪  mean_abs_invalidisierung_diff_n * (1 + (lJahr -  jahr_projektion) *
            ↪  (1/20))
        ) %>%
        select(alt, sex, contains("mutationsrate"), contains("invalidisierung"),
        ↪  contains("invalidisierung_tief"), contains("invalidisierung_hoch")) %>%
        select(-contains("_diff"), -contains("growth")) %>%
        mutate(jahr = lJahr) # Füge eine Spalte mit dem aktuellen Jahr hinzu
```

Es wird eine Schleife über die nächsten 20 Jahre ab dem aktuellen Projektionsjahr geschrieben (Für die Finanzperspektiven 2024 entspricht dies einer Schleife von 2024 bis 2044). Danach werden die HE-Invalidisierungs- und HE-Mutationsrate für das entsprechende Jahr berechnet. `invalidisierung` berechnet die Invalidisierung im mittleren Szenario. Der Parameter `fort_entw` ist Stand 2024 = 0 respektive `FALSE`, Term der mit `fort_entw` multipliziert wird wegfällt. Daher gehen wir nicht weiter auf diesen Term ein. `invalidisierung = invalidisierung + fort_entw *...` zeigt, dass die HE-Invalidisierung im mittleren Szenario konstant und gleich der vorangehend berechneten durchschnittlichen HE-Invalidisierung ist. Das gleiche trifft auf die anderen drei Variablen im mittleren Szenario zu (`invalidisierung_n`, `mutationsrate`, und `mutationsrate_n`). Die HE-Invalidisierung im tiefen Szenarion `invalidisierung_tief` ergibt sich aus der HE-Invalidisierung im mittleren Szenario zuzüglich `mean_abs_invalidisierung_diff` multipliziert mit einem 20zigstel der Anzahl Jahre, die das entsprechende Jahr vom aktuellen Projektionsjahr entfernt ist. Die Berechnung der HE-Invalidisierung im hohen Szenario erfolgt analog.

Der Nachfolgende Codeteil dient dazu, den Effekt der AHV-21 bedingte Rentenaltererhöhung von 64 auf 65 bei den Frauen auf die IV abzubilden.[30]

```
    if (PARAM_GLOBAL$he_mitAHV21 & lJahr>=2025 & (PARAM_GLOBAL$jahr_abr -
    ↪  (PARAM_GLOBAL$MA_years-1))<2028) { # Ab 2028 ist die AHV-21 voll in Kraft,
    ↪  das heisst, dass ab dann die Invalidisierungs- und Mutationsraten der
    ↪  64-jährigen Frauen nicht mehr von der AHV21-Übergangsregelung betroffen sind,
    ↪  und für die Schätzung verwendet werden können. Da die dem letzten MA-Jahr
    ↪  vorangehenden PARAM_GLOBAL$MA_years auch für die Schätzung verwendet werden,
    ↪  muss sicher gestellt werden, dass diese Jahre auch mindestens 2028 sind.

        IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
        filter(!((alt == 65 | alt==64) & sex == "f")) %>%  # Entferne alle Zeilen
        ↪  mit (alt == 64 | alt ==65) und sex == "f" (solche werden ab 2025
        ↪  enthalten sein, da ab dann auf Grund der AHV21 auch Frauen, die Ende
        ↪  Jahr 65 sein werden, in der IV sind)
```

[30] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
                add_row(alt = 65,
                        sex = "f",
                        mutationsrate = -1,
                        mutationsrate_n = -1,
                        invalidisierung = 0,
                        invalidisierung_n = 0,
                        invalidisierung_tief = 0,
                        invalidisierung_hoch = 0,
                        invalidisierung_tief_n = 0,
                        invalidisierung_hoch_n = 0,
                        jahr = lJahr)  # Füge neue Zeile für die 65-jährigen Frauen mit
                        ↪  den richtigen mutationsraten (-1) und invalidisierungen (0)
                        ↪  hinzu

            # Dupliziere Zeilen mit alt == 63 und sex == "f" und setze alt = 64 in der
            ↪  duplizierten Version
            duplicated_rows <- IV_RATEN_INV_MUT_JAHR %>%
                filter(alt == 63 & sex == "f") %>%
                mutate(alt = 64)

            IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
                bind_rows(duplicated_rows)

            # Ändere die Zeilen mit sex == "f" und alt == 64
            IV_RATEN_INV_MUT_JAHR <- IV_RATEN_INV_MUT_JAHR %>%
                mutate(across(contains("invalidisierung"), ~ ifelse(sex == "f" & alt ==
                ↪  64, . * min(4, (lJahr - 2024)) / 4, .))) %>%
                mutate(across(contains("mutationsrate"), ~ ifelse(sex == "f" & alt == 64,
                ↪  . * min(4, (lJahr - 2024)) / 4 + min(0, (lJahr - 2028)) / 4, .))) %>%
                arrange(sex, alt)
        }

        # Aktualisiere das Dataframe mit neuen Daten
        IV_RATEN_INV_MUT_HE <- bind_rows(IV_RATEN_INV_MUT_HE, IV_RATEN_INV_MUT_JAHR) #
↪  Füge die Ergebnisse der Iteration zum Dataframe hinzu

    }
```

In einem ersten Schritt wird die HE-Invalidisierung und HE-Mutationsrate bei Frauen mit Alter 64 und 65 entfernt. Danach wird eine Zeile hinzugefügt, die die HE-Invalidisierung und HE-Mutationsrate für 65-jährige Frauen so setzt, wie sie nach Einführung der AHV21 zu erwarten ist (d.h., es gehen alle Frauen 65-jährige Frauen in die AHV über (HE-Mutationsrate von -1) und es kommen keine neuen 65-jährige Frauen dazu (HE-Invalidisierung von 0)).

Danach wird das Dataframe `duplicated_rows` gebildet, welches die HE-Invaldisierungs- und HE-Mutationsraten der 63-jährigen Frauen auf die 64-jährigen Frauen überträgt. Der Grund ist, dass wir in der Übergangsphase zur AHV-21 keine Daten zur HE-Invalidisierung und den HE-Mutationen bei den 64-jährigen Frauen haben (da vor der AHV-21 alle Frauen mit 64 schon in der AHV sind). Daher nehmen wir an, dass die HE-Invalidisierung/HE-Mutationsrate bei den 64-jährigen der HE-Invalidisierung/HE-Mutationsrate bei den 63-jährigen entspricht.

Der letzte Codeteil nach dem Kommentar `# Ändere die Zeilen mit sex == f" und alt == 64` fängt ab, dass die Rentenaltererhöhung bei den Frauen über die Jahre 2025 bis 2028 gestaffelt um jeweils ein vierteljahr pro Jahr erfolgt. Wir passen also die HE-Invalidisierung und HE-Mutationsrate an die im jeweiligen Jahr

total aufgelaufene Rentenaltererhöhung an.

Die obigen Berechnungen werden wiederum genau gleich auch für die Kinder-HE durchgeführt, wobei bei den Kinder-HE der Teil zur AHV21/Frauenrentenaltererhöhung (logischerweise) entfällt.

4.4.4 HE-Bestand projizieren

Mit dem nachfolgenden Code werden die Bestände an HE-Beziehenden und Neu-HE-Beziehenden gemäss den Registerdaten ins Dataframe `RENTENBESTAND_IV` eingelesen:

```
RENTENBESTAND_IV <- BESTAND_HE %>%
    mutate(bestand_tief = bestand, bestand_hoch = bestand, neurenten_tief = neurenten,
        neurenten_hoch = neurenten, bestandesmutationen_tief = bestandesmutationen,
        bestandesmutationen_hoch = bestandesmutationen, bestand_n_tief = bestand_n,
        bestand_n_hoch = bestand_n, neurenten_n_tief = neurenten_n,
        neurenten_n_hoch = neurenten_n, bestandesmutationen_n_tief =
        ↪  bestandesmutationen_n,
        bestandesmutationen_n_hoch = bestandesmutationen_n, quelle = "Register") %>%
    select(-contains("_vj"))
```

Wie oben wird auch hier die genau gleiche Berechnung mit dem Dataframe `BESTAND_HE_KINDER` durchgeführt.

Mit der nachfolgenden Code-Schleife werden dann rekursiv die HE-Bestände für die kommenden Jahre projiziert:

```
    for(lJahr in (jahr_projektion):PARAM_GLOBAL$jahr_ende) {
        suppressMessages({

            RENTENBESTAND_NEU <- RENTENBESTAND_IV %>%
                filter(jahr == lJahr - 1,
                       quelle == ifelse(lJahr == jahr_projektion, "Register",
↪  "Projektion"),
                       (sex == "m" & alt <= 64) |
                           (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == FALSE & alt <= 63) |
                           (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == TRUE & lJahr < 2025
↪  & alt <= 63) |
                           (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == TRUE & lJahr >= 2025
↪  & alt <= 64)
                ) %>%
                select(jahr, sex, alt, starts_with("bestand")) %>%
                full_join(BEVOELKERUNG %>%
                              group_by(jahr, sex, alt) %>%
                              summarize(bevendejahr = sum(bevendejahr)) %>%
                              ungroup() %>%
                              filter(alt>=17,
                                     (sex == "m" & alt <= 64) |
                                         (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == FALSE
↪  & alt <= 63) |
                                         (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == TRUE &
↪  lJahr < 2025 & alt <= 63) |
                                         (sex == "f" & PARAM_GLOBAL$he_mitAHV21 == TRUE &
↪  lJahr >= 2025 & alt <= 64),
                                     jahr==lJahr-1)
                ) %>%
                arrange(sex, alt) %>%
```

```
        mutate(jahr=jahr+1,
               alt=alt+1
        ) %>%
        mutate(jahrX=jahr,
        ↪  jahr=ifelse(jahr<=(jahr_projektion+20),jahr,(jahr_projektion+20)))
        ↪  %>%
        left_join(IV_RATEN_INV_MUT_HE) %>%
        select(-jahr) %>%
        rename(jahr=jahrX)
```

Die Schleife läuft vom ersten Projektionsjahr bis zum Ende des Projektionshorizont (in 2024 heisst das von 2024 bis 2070). Zuerst wird das Dataframe `RENTENBESTAND_NEU` aus dem HE-Bestand des Vorjahres gebildet (welcher im Projektionsjahr aus dem Register entnommen wird und in allen folgenden Jahren aus der Projektion für das vorangehende Jahr, d.h., der vorangehenden Iteration der Schleife). Danach werden nur relevante Bestände ausgewählt, das heisst, nur Männer bis 64 (da Männer, die Ende Jahr 65 sind per definition in die AHV übergegangen sind), und Frauen bis 2024 bis 63, und danach auch bis 64 (sofern `mitAHV21==TRUE` ist, was Stand 2024 der Fall ist).

Danach wird mit dem `full_join(BEVOELKERUNG` Befehl die relevante Bevölkerung am Ende des Jahres vor `lJahr` hinzugefügt.

Mit dem Befehl `mutate(jahrX=jahr` wird das eigentliche Projektionsjahr (also das `lJahr`) in der Variable `jahrX` gespeichert, während die Variable `jahr` so spezifiziert wird, dass sie mit der im Projektionsjahr gültigen HE-Invalidisierung und HE-Mutationsrate aus dem Dataframe `IV_RATEN_INV_MUT` (vgl. Kapitel 4.4.2) gejoined werden kann.

Danach werden mit dem folgenden, zweiten Teil der Schleife, die Bestände an HE-Beziehenden und Neu-Beziehenden im `lJahr` berechnet:

```
        # converting NA to zero
        RENTENBESTAND_NEU[is.na(RENTENBESTAND_NEU)] <- 0

        RENTENBESTAND_NEU <- RENTENBESTAND_NEU %>%
            mutate(
                neurenten= (bevendejahr-bestand_n) * invalidisierung,
                bestandesmutationen = bestand * mutationsrate,
                bestand = bestand * (1 + mutationsrate) + neurenten,
                neurenten_tief= (bevendejahr-bestand_n_tief) * invalidisierung_tief,
                bestandesmutationen_tief = bestand_tief * mutationsrate,
                bestand_tief = bestand_tief * (1 + mutationsrate) + neurenten_tief,
                neurenten_hoch= (bevendejahr-bestand_n_hoch) * invalidisierung_hoch,
                bestandesmutationen_hoch = bestand_hoch * mutationsrate,
                bestand_hoch = bestand_hoch * (1 + mutationsrate) + neurenten_hoch,
                neurenten_n= (bevendejahr-bestand_n) * invalidisierung_n,
                bestandesmutationen_n = bestand_n * mutationsrate_n,
                bestand_n = bestand_n * (1 + mutationsrate_n) + neurenten_n,
                neurenten_n_tief = (bevendejahr-bestand_n_tief) *
                ↪  invalidisierung_tief_n,
                bestandesmutationen_n_tief = bestand_n_tief * mutationsrate_n,
                bestand_n_tief = bestand_n_tief * (1 + mutationsrate_n) +
                ↪  neurenten_n_tief,
                neurenten_n_hoch = (bevendejahr-bestand_n_hoch) *
                ↪  invalidisierung_hoch_n,
                bestandesmutationen_n_hoch = bestand_n_hoch * mutationsrate_n,
                bestand_n_hoch = bestand_n_hoch * (1 + mutationsrate_n) +
                ↪  neurenten_n_hoch
```

```
                ) %>%
                ungroup()


            RENTENBESTAND_NEU <- RENTENBESTAND_NEU %>%
                select(-bevendejahr, -contains("mutationsrate"),
                ↪  -contains("invalidisierung"), -contains("invalidisierung_tief"),
                ↪  -contains("invalidisierung_hoch")) %>%
                mutate(quelle="Projektion")

            #------ Step 3b: an Ergebnistabelle anfügen ---------------------------#
            RENTENBESTAND_IV   <- RENTENBESTAND_IV %>%
                rbind(., RENTENBESTAND_NEU)
        })
    }
```

Die Berechnung der Bestände an HE-Beziehenden und Neu-Beziehenden in den Projektionsjahren ist analog zur in Abschnitt 4.4.1 erläuterten Berechnung der Bestände an HE-Beziehenden und Neu-Beziehenden aus den Registerdaten.

Wiederum werden hier die genau gleichen Berechnungen für die Kinder-HE durchgeführt, wobei auch hier wiederum natürlich der Teil zur AHV-21 entfällt.

4.4.5 HE-Ausgaben projizieren

Um die Projektionen für die gesamten HE-Ausgaben zu erstellen, werden zuerst die Bestandes- und Flussgrössen für die Erwachsenen-HE und die Kinder-HE separat nach Jahr aggregiert:

```
BESTAND_IV <- RENTENBESTAND_IV %>%
    group_by(jahr, quelle) %>%
    summarize(bestand = sum(bestand), neurenten = sum(neurenten),
        bestandesmutationen = sum(bestandesmutationen), bestand_hoch = sum(bestand_hoch),
        bestand_tief = sum(bestand_tief), neurenten_hoch = sum(neurenten_hoch),
        neurenten_tief = sum(neurenten_tief), bestandesmutationen_hoch =
        ↪  sum(bestandesmutationen_hoch),
        bestandesmutationen_tief = sum(bestandesmutationen_tief),
        bestand_n = sum(bestand_n), bestandesmutationen_n = sum(bestandesmutationen_n),
        neurenten_n = sum(neurenten_n), bestand_n_hoch = sum(bestand_n_hoch),
        bestand_n_tief = sum(bestand_n_tief), neurenten_n_hoch = sum(neurenten_n_hoch),
        neurenten_n_tief = sum(neurenten_n_tief), bestandesmutationen_n_hoch =
        ↪  sum(bestandesmutationen_n_hoch),
        bestandesmutationen_n_tief = sum(bestandesmutationen_n_tief))


BESTAND_IV_KINDER <- RENTENBESTAND_IV_KINDER %>%
    group_by(jahr, quelle) %>%
    summarize(bestand = sum(bestand), neurenten = sum(neurenten),
        bestandesmutationen = sum(bestandesmutationen), bestand_hoch = sum(bestand_hoch),
        bestand_tief = sum(bestand_tief), neurenten_hoch = sum(neurenten_hoch),
        neurenten_tief = sum(neurenten_tief), bestandesmutationen_hoch =
        ↪  sum(bestandesmutationen_hoch),
        bestandesmutationen_tief = sum(bestandesmutationen_tief),
        bestand_n = sum(bestand_n), bestandesmutationen_n = sum(bestandesmutationen_n),
        neurenten_n = sum(neurenten_n), bestand_n_hoch = sum(bestand_n_hoch),
        bestand_n_tief = sum(bestand_n_tief), neurenten_n_hoch = sum(neurenten_n_hoch),
```

```
        neurenten_n_tief = sum(neurenten_n_tief), bestandesmutationen_n_hoch =
        ↪  sum(bestandesmutationen_n_hoch),
        bestandesmutationen_n_tief = sum(bestandesmutationen_n_tief))
```

Danach werden die Registerdaten mit den Abrechnungsdaten aus der IV-Abrechnung verglichen:

```
ANTEILE <- IV_ABRECHNUNG %>%
    filter(jahr >= jahr_projektion - 3, jahr < jahr_projektion) %>%
    left_join(BESTAND_IV %>%
        select(jahr, bestand) %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12) %>%
        filter(jahr >= jahr_projektion - 3, jahr < jahr_projektion)) %>%
    left_join(BESTAND_IV_KINDER %>%
        select(jahr, bestand) %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12) %>%
        rename(bestand_kinder = bestand) %>%
        filter(jahr >= jahr_projektion - 3, jahr < jahr_projektion)) %>%
    mutate(frac_he_nachz = (he - bestand - bestand_kinder)/(bestand +
        bestand_kinder)) %>%
    summarize(frac_he_nachz = mean(frac_he_nachz)) %>%
    ungroup()
```

D.h. wir berechnen in der drittletzten Zeile des obigen Codes die Abweichung zwischen den Totalausgaben für HE von der Summe der Anhand der Dezemberbestände hochgerechneten Ausgaben für Erwachsenen-HE und Kinder-HE `he-bestand-bestand_kinder` relativ zur Summe der Anhand der Dezemberbestände hochgerechneten Ausgaben für Erwachsenen-HE und Kinder-HE `bestand+bestand_kinder`. Wir benennen diese Abweichung `frac_he_nachz` nehmen also an, dass das Ausmass mit welchem die in der IV-Abrechnung ausgewiesenen Totalausgaben für HE von den Anhand der Dezemberbestände der Registerdaten hochgerechneten Ausgaben abweichen Nachzahlungen geschuldet ist. Der Grund für die Abweichung ist irrelevant, es geht schlussendlich lediglich um die Korrektur der Abweichung der Registerdaten von den Abrechnungsdaten, wobei implizit angenommen wird, dass diese Abweichung in Zukunft der mittleren Abweichung der letzten 3 Jahre entspricht. Tatsächlich ist die Abweichung gering und fast konstant über die Zeit. So beträgt `frac_he_nachz` beispielsweise in den Jahren 2021, 2022, und 2023 0.062, 0.060 respektive 0.050.

Somit können wir die Gesamtausgaben für HE projizieren:

```
    RENTEN_IV <- BESTAND_IV %>%
        select(jahr, quelle, bestand, bestand_hoch, bestand_tief, neurenten,
        ↪  neurenten_hoch, neurenten_tief) %>%
        left_join(RENTENENTWICKLUNG %>% select(jahr, minimalrente)) %>%
        mutate(
            renten = bestand * minimalrente * 12, # Berechne renten
            renten_hoch = bestand_hoch * minimalrente * 12, # Berechne renten_hoch
            renten_tief = bestand_tief * minimalrente * 12 # Berechne renten_tief
        ) %>%
        select(jahr, quelle, contains("renten"), minimalrente, -contains("neurenten"))
        ↪  %>%
        left_join(
            BESTAND_IV_KINDER %>%
                select(jahr, quelle, bestand, bestand_hoch, bestand_tief, neurenten,
                ↪  neurenten_hoch, neurenten_tief) %>%
```

```
                left_join(RENTENENTWICKLUNG %>% select(jahr, minimalrente)) %>%
                mutate(
                    renten = bestand * minimalrente * 12, # Berechne renten für Kinder
                    renten_hoch = bestand_hoch * minimalrente * 12, # Berechne
                    ↪  renten_hoch für Kinder
                    renten_tief = bestand_tief * minimalrente * 12 # Berechne renten_tief
                    ↪  für Kinder
                ) %>%
                rename(
                    renten_kinder = renten, # Umbenennen von renten zu renten_kinder
                    renten_hoch_kinder = renten_hoch, # Umbenennen von renten_hoch zu
                    ↪  renten_hoch_kinder
                    renten_tief_kinder = renten_tief # Umbenennen von renten_tief zu
                    ↪  renten_tief_kinder
                ) %>%
                select(jahr, quelle, contains("renten"), -contains("neurenten"))
        ) %>%
        mutate(
            nachzahlungen = (renten + renten_kinder) * ANTEILE$frac_he_nachz, # Berechne
            ↪  nachzahlungen
            nachzahlungen_hoch = (renten_hoch + renten_hoch_kinder) *
            ↪  ANTEILE$frac_he_nachz, # Berechne nachzahlungen_hoch
            nachzahlungen_tief = (renten_tief + renten_tief_kinder) *
            ↪  ANTEILE$frac_he_nachz # Berechne nachzahlungen_tief
        ) %>%
        mutate(
            renten_total = renten + renten_kinder + nachzahlungen, # Berechne
            ↪  renten_total
            renten_total_hoch = renten_hoch + renten_hoch_kinder + nachzahlungen_hoch, #
            ↪  Berechne renten_total_hoch
            renten_total_tief = renten_tief + renten_tief_kinder + nachzahlungen_tief #
            ↪  Berechne renten_total_tief
        )
```

Hierzu berechnen wir die Summe der Anhand der Dezemberbestände hochgerechneten Ausgaben für Erwachsenen-HE und Kinder-HE, und rechnen dann die Nachzahlungen unter Verwendung des oben berechneten Faktors `ANTEILE$frac_he_nachz` dazu. Danach ergibt sich aus `renten + renten_kinder + nachzahlungen` die Variable `renten_total`, welche die totalen Ausgaben für HE darstellt.

Der nachfolgende Codeteil dient dazu, für den Fall dass `PARAM_GLOBAL$he_szenario == TRUE` wäre, was Stand 2024 nicht der Fall ist, die entsprechende Szenarioprojektion für `renten_total` auszuwählen. Wenn `PARAM_GLOBAL$he_szenario == FALSE` ist (Stand 2024 der Fall) heisst das, dass in jedem Invalidisierungs-Szeanrio für die HE-Ausgaben die Projektion des Referenzszenarios (oder mittleren Szeanrios) gewählt wird.

```
if (PARAM_GLOBAL$he_szenario == TRUE) {
    RENTEN_IV <- RENTEN_IV %>%
        mutate(renten_total = case_when(PARAM_GLOBAL$invalidisierung_szenario ==
            "mittel" ~ renten_total, PARAM_GLOBAL$invalidisierung_szenario ==
            "tief" ~ renten_total_tief, PARAM_GLOBAL$invalidisierung_szenario ==
            "hoch" ~ renten_total_hoch, TRUE ~ renten_total  # default to existing
            ↪  iv_renten if needed
))
}
```

Somit können die HE-Ausgabenprojektionen mit den Werten aus der IV-Abrechnung zusammengefügt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_geldleistungen.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
    HE_IV         <- rbind(IV_ABRECHNUNG %>%
                           select(jahr, he) %>%
                           rename(he_iv = he), RENTEN_IV %>%
                           filter(jahr>PARAM_GLOBAL$jahr_abr) %>%
                           group_by(jahr) %>%
                           summarize(he_iv=sum(renten_total, na.rm = TRUE)) %>%
                           ungroup()
                       )
    })
    #------ Output ------------------------------------------------------------#
    mod_return(HE_IV)
```

4.5 Modul mod_iv_mm_new.R

Dokumentation zuletzt aktualisiert am 29.08.2024

Notiz Übergangsphase: Im Moment wird das neue Modul nur aufgerufen, falls im Input-Conatiner in `PARAM_GLOBAL` dem Parameter *mod_iv_mm* der Wert *NEW* zugewiesen wird.

Das Prognosemodell zu den medizinischen Massnahmen ist im Skript `mod_iv_mm_new.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_mm <- mod_iv_mm_new(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
    IV_MED_MASSN_REGISTER = IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
    ↪  IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = IV_MED_MASSN_PARAMCHANGES, DISKONTFAKTOR = DISKONTFAKTOR,
    BEVOELKERUNG = BEVOELKERUNG)
```

Das Modul `mod_iv_mm_new.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `IV_MED_MASSN_REGISTER`: Die Registerdaten werden für die Berechnung der Modellparameter Invalidisierung, Kosten pro Invalide, und Kostenwachstum pro Invalide verwendet.
- `IV_MED_MASSN_SONDEREFFEKTE`: Die Sondereffekte werden verwendet, um für Sondereffekte in der Invalidisierung, der Ausgaben pro Invalide, oder dem Wachstum der Ausgaben pro Invalide zu korrigieren.
- `IV_MED_MASSN_PARAMCHANGES`: Das Dataframe mit Parameterkorrekturen dient dazu, allfällig eingegebene manuelle Veränderungen der Modellparameter für die Prognose miteinzubeziehen.
- `DISKONTFAKTOR`: Da das Modell in realen grössen gerechnet wird, wird der Diskontfaktor benötigt, um nominale grössen vor der Modellberechnung in reale Grössen umzuwandeln, respektive um die realen Modellprognosen am Ende in nominale Grössen umzuwandeln.
- `BEVOELKERUNG`: Die Bevölkerungsdaten und -szenarien weren dazu genutzt, um die Anzahl Invaliden nach Alter in der Zukunft zu schätzen.

Im Modul werden in `mod_iv_mm_new.R` werden in einem ersten Schritt sämtliche nominalen Grössen in reale Grössen umgerechnet:

```
#------ Monetäre Grössen zu konstanten Preisen umwandel ---------------#

MM_ABRECHNUNG_REAL <- IV_ABRECHNUNG %>%
    left_join(DISKONTFAKTOR) %>%
    mutate(mm_real = mm * diskontfaktor) %>%
    select(jahr, mm_real)
```

```
IV_MED_MASSN_REGISTER_REAL <- IV_MED_MASSN_REGISTER %>%
    left_join(DISKONTFAKTOR) %>%
    mutate(ausgaben_real = ausgaben_nom * diskontfaktor) %>%
    select(-ausgaben_nom, -diskontfaktor)
```

4.5.1 Invalidisierung berechnen

Als erstes wird die Invalidisierung nach Alter und Jahr berechnet:

```
REGISTERDATEN_INVALIDISIERUNG <- IV_MED_MASSN_REGISTER_REAL %>%
    left_join(BEVOELKERUNG %>%
        group_by(jahr, alt) %>%
        summarise(bevendejahr = sum(bevendejahr), .groups = "drop")) %>%
    arrange(alt, jahr) %>%
    mutate(invalidisierung = invalide/bevendejahr, ) %>%
    select(jahr, alt, invalidisierung)
```

Die im Modell verwendete Schätzung der Invalidisierung entspricht dem Durchschnitt der Invalidisierung zwischen $t = T - 10$ und $t = T - 1$, wobei T das Jahr des aktuellen Finanzhaushalts darstellt (und $T - 1$ somit das letzte Jahr, für welches die Registerdaten verfügbar sind).

Diese einfache Schätzung kann aber nur verwendet werden, wenn im Dataframe `SONDEREFFEKTE` für die Jahre $t = T - 10$ bis $t = T - 1$ keine Sondereffekte für die Invalidisierung ausgewiesen sind. Sind Sondereffekte vorhanden, so sind zusätzliche Berechnung notwendig, welche nachfolgend erläutert werden.

4.5.2 Invalidisierung mit Sondereffekten

Bei Vorhandensein von Sondereffekten ist eine spezifischere Berechnung der Invalidisierung nötig. Als erstes werden permanente Sondereffekte und temporäre Sondereffekte separat extrahiert:

```
# Schritt 1: Identifizierung der relevanten Jahre für
# permanent_invalidisierung
PERMANENT_INVALIDISIERUNG_YEARS <- IV_MED_MASSN_SONDEREFFEKTE %>%
    filter(sondereffekt == "permanent_invalidisierung") %>%
    select(sondereffektjahr, sondereffektalter)

# Identifizierung der Jahre für temporaer_invalidisierung
TEMPORAER_INVALIDISIERUNG_YEARS <- IV_MED_MASSN_SONDEREFFEKTE %>%
    filter(sondereffekt == "temporaer_invalidisierung") %>%
    select(sondereffektjahr, sondereffektalter)
```

Die Unterscheidung zwischen permanenten und temporären Sondereffekten ist wichtig, da diese verschiedene Interpretationen haben und daher die Modellberechnung unterschiedlich beeinflussen:

- Temporäre Sondereffekte: Sondereffekte, die nur in einem bestimmten Jahr einen Einfluss auf die Invalidisierung bei einer Altersgruppe haben (bspw. eine Pilotversuch-bedingte vorübergehende Vergütung von Leistungen). In diesem Fall wird das entsprechende Jahr für die entsprechende Altersgruppe bei der Berechnung der durchschnittlichen Invalidisierung ausgeschlossen.[31]
- Permanente Sondereffekte: Sondereffekte, die dazu führen, dass die Invalidisierung in einer Altersgruppe ab dem entsprechenden Jahr permanent höher ist (bspw. die Aufnahme neuer Gebrechen in den Leistungskatalog im Rahmen einer Reform). In diesem Fall dient der Durchschnitt der Invalidisierung in der entsprechenden Altersgruppe in den Jahren vor dem Sondereffektjahr als Schätzer für die Invalidisierung in den Jahren vor dem Sondereffektjahr, und der Durchschnitt der Invalidisierung in den

[31] Für das Jahr, in welchem der temporäre Sondereffekt vermerkt ist, wird die Invalidisierung im Sondereffektjahr verwendet.

Jahren ab dem Sondereffektjahr als Schätzer für die Invalidisierung in den Jahren ab dem Sondereffektjahr.[32]

Nachfolgend ein Beispiel zur Illustration: Im Januar 2022 trat die Weiterentwicklung der IV in Kraft. Dies führte zu Verschiebungen im Leistungskatalog bei den medizinischen Massnahmen. Als konsequenz davon gehen wir davon aus, dass die Jahre vor 2022 bezüglich Invalidisierung und Kosten pro Invalide nicht representativ sind für die Zeit ab 2022. Auf Grund von Rechnungsverzögerungen kann aber auch nicht davon ausgegangen werden, dass das Jahr 2022 representativ ist für die Entwicklung nach 2022. Wir betrachten daher die Invaldisierung und die Kosten pro Invalide im Jahr 2022 als von einem temporären Sondereffekt betroffen, und gehen davon aus, dass dieser Sondereffekt im Jahr 2023 permanent ist (ab 2023 befinden wir uns vollständig im post-Weiterentwicklung IV System). Im Modell heisst das, dass in den Jahren ab 2023 nur die Jahre ab 2023 für die Berechnung der durchschnittlichen Invalidisierung verwendet werden. Für das Jahr 2022 entspricht die Invalidisierung der Invalidisierung im Jahr 2022, und für die Jahre vor 2022 wird der Durchschnitt der Invalidisierung als der Mittelwert der beobachteten Invalidisierung in den Jahren 2014 bis 2021 gebildet. Der Durchschnitt der Invalidisierung für Jahre in der Vergangenheit wird gebildet, da die Modellschätzung für diese Jahre später bei der Justierung auf die IV-Abrechnung benötigt wird (details in Kapitel 4.5.8).

Nachfolgend der R-Code, der diese Logik implementiert:

```
REGISTERDATEN_INVALIDISIERUNG_PERMANENT <- REGISTERDATEN_INVALIDISIERUNG %>%
    anti_join(TEMPORAER_INVALIDISIERUNG_YEARS,
              by = c("jahr" = "sondereffektjahr", "alt" = "sondereffektalter"))

 suppressWarnings( # warnings von erstem mutate() Befehl unterdrücken
    MEAN_INVALIDISIERUNG_YEARS <- REGISTERDATEN_INVALIDISIERUNG_PERMANENT %>%
        filter(jahr > letztes_registerjahr - 10 & jahr <= letztes_registerjahr) %>%
        rowwise() %>%
        mutate(
            first_jahr = pmax(letztes_registerjahr-9,
                              max(PERMANENT_INVALIDISIERUNG_YEARS$sondereffektjahr[
                                PERMANENT_INVALIDISIERUNG_YEARS$sondereffektjahr <=
                                  jahr &
                                  PERMANENT_INVALIDISIERUNG_YEARS$sondereffektalter
                                == alt], na.rm = TRUE), na.rm = TRUE),
            last_jahr = pmin(letztes_registerjahr,
                             min(PERMANENT_INVALIDISIERUNG_YEARS$sondereffektjahr[
                               PERMANENT_INVALIDISIERUNG_YEARS$sondereffektjahr >
                                 jahr &
                                 PERMANENT_INVALIDISIERUNG_YEARS$sondereffektalter
                               == alt] - 1, na.rm = TRUE), na.rm = TRUE)
        ) %>%
        mutate(
            mean_invalidisierung =
              if_else(is.na(first_jahr)
                      | is.na(last_jahr),
                      NA_real_,
                      mean(REGISTERDATEN_INVALIDISIERUNG_PERMANENT$invalidisierung[
                        REGISTERDATEN_INVALIDISIERUNG_PERMANENT$jahr >=
                          first_jahr & REGISTERDATEN_INVALIDISIERUNG_PERMANENT$jahr
                        <= last_jahr & REGISTERDATEN_INVALIDISIERUNG_PERMANENT$alt
                        == alt], na.rm = TRUE))
        ) %>%
```

[32]Sind mehrere permanente Sondereffekte vorhanden, so wird die durchschnittliche Invalidisierung in den Jahren zwischen den Sondereffektjahren als der Durchschnitt der Invalidisierung in den Jahren zwischen den Sondereffektjahren gebildet.

```
            select(jahr, alt, mean_invalidisierung) %>%
            rename(invalidisierung = mean_invalidisierung) %>%
            ungroup() %>%
            bind_rows(., REGISTERDATEN_INVALIDISIERUNG %>%
                          right_join(TEMPORAER_INVALIDISIERUNG_YEARS,
                                     by = c("jahr" = "sondereffektjahr",
                                            "alt" = "sondereffektalter")) %>%
                          filter(jahr > letztes_registerjahr- 10 &
                                   jahr <= letztes_registerjahr) %>%
                          mutate(
                              mean_invalidisierung = invalidisierung
                          ) %>%
                          select(jahr, alt, mean_invalidisierung) %>%
                          rename(invalidisierung = mean_invalidisierung)) %>%
            arrange(alt, jahr)
    )
```

4.5.3 Ausgaben pro Invalide berechnen

Die Berechnung der durchschnittlichen Ausgaben pro Invalide nach Alter, sowie der Umgang mit Sondereffekten, sind analog zu den Berechnungen für die Invalidisierung im vorangehenden Abschnitt. Die durchschnittlichen Ausgaben pro Invalide nach Alter und Jahr werden anhand des folgenden Codes berechnet:

```
#------ Vorbereitung Modell: Ausgaben --------------------------------#
    REGISTERDATEN_AUSGABEN <-
        IV_MED_MASSN_REGISTER_REAL %>%
        arrange(alt, jahr) %>%
        mutate(
            ausgaben_invalide = ausgaben_real / invalide,
        ) %>%
        select(jahr, alt, ausgaben_invalide)

    # Schritt 1: Identifizierung der relevanten Jahre für permanent_invalidisierung
    PERMANENT_AUSGABEN_YEARS <- IV_MED_MASSN_SONDEREFFEKTE %>%
        filter(sondereffekt == "permanent_ausgaben") %>%
        select(sondereffektjahr, sondereffektalter)

    # Identifizierung der Jahre für temporaer_invalidisierung
    TEMPORAER_AUSGABEN_YEARS <- IV_MED_MASSN_SONDEREFFEKTE %>%
        filter(sondereffekt == "temporaer_ausgaben") %>%
        select(sondereffektjahr, sondereffektalter)

    REGISTERDATEN_AUSGABEN_PERMANENT <- REGISTERDATEN_AUSGABEN %>%
        anti_join(TEMPORAER_AUSGABEN_YEARS, by = c("jahr" = "sondereffektjahr",
                                                   "alt" = "sondereffektalter"))

    suppressWarnings( # warnings von erstem mutate() Befehl unterdrücken
        MEAN_AUSGABEN_YEARS <- REGISTERDATEN_AUSGABEN_PERMANENT %>%
            filter(jahr > letztes_registerjahr - 10 & jahr <= letztes_registerjahr) %>%
            rowwise() %>%
            mutate(
                first_jahr = pmax(letztes_registerjahr-9,
                                  max(PERMANENT_AUSGABEN_YEARS$sondereffektjahr[
                                    PERMANENT_AUSGABEN_YEARS$sondereffektjahr <=
```

```
                                    jahr & PERMANENT_AUSGABEN_YEARS$sondereffektalter
                                  == alt], na.rm = TRUE), na.rm = TRUE),
            last_jahr = pmin(letztes_registerjahr,
                             min(PERMANENT_AUSGABEN_YEARS$sondereffektjahr[
                               PERMANENT_AUSGABEN_YEARS$sondereffektjahr >
                                 jahr & PERMANENT_AUSGABEN_YEARS$sondereffektalter
                               == alt] - 1, na.rm = TRUE), na.rm = TRUE)
        ) %>%
        mutate(
            mean_ausgaben_invalide =
              if_else(is.na(first_jahr)
                      | is.na(last_jahr),
                      NA_real_,
                      mean(REGISTERDATEN_AUSGABEN_PERMANENT$ausgaben_invalide[
                        REGISTERDATEN_AUSGABEN_PERMANENT$jahr >= first_jahr &
                          REGISTERDATEN_AUSGABEN_PERMANENT$jahr <= last_jahr &
                          REGISTERDATEN_AUSGABEN_PERMANENT$alt == alt],
                        na.rm = TRUE)),
            mean_ausgaben_jahr =
              if_else(is.na(first_jahr) | is.na(last_jahr), NA_real_,
                      mean(REGISTERDATEN_AUSGABEN_PERMANENT$jahr[
                        REGISTERDATEN_AUSGABEN_PERMANENT$jahr >= first_jahr &
                          REGISTERDATEN_AUSGABEN_PERMANENT$jahr <= last_jahr &
                          REGISTERDATEN_AUSGABEN_PERMANENT$alt == alt],
                        na.rm = TRUE))
        ) %>%
        select(jahr, alt, mean_ausgaben_invalide, mean_ausgaben_jahr) %>%
        rename(ausgaben_invalide = mean_ausgaben_invalide) %>%
        ungroup() %>%
        bind_rows(., REGISTERDATEN_AUSGABEN %>%
                      right_join(TEMPORAER_AUSGABEN_YEARS,
                                 by = c("jahr" = "sondereffektjahr",
                                        "alt" = "sondereffektalter")) %>%
                      filter(jahr > letztes_registerjahr - 10 & jahr <=
                               letztes_registerjahr) %>%
                      mutate(
                          mean_ausgaben_invalide = ausgaben_invalide,
                          mean_ausgaben_jahr = jahr
                      ) %>%
                      select(jahr,
                             alt,
                             mean_ausgaben_invalide,
                             mean_ausgaben_jahr) %>%
                      rename(ausgaben_invalide = mean_ausgaben_invalide)) %>%
        arrange(alt, jahr)
)
```

Der einzige Unterschied zu den Berechnungen für die Invalidisierung liegt darin, dass zusätzlich die Variable `mean_ausgaben_jahr` ausgelesen wird. Diese Variable beschreibt das mittlere Jahr des Zeitraumes, über welchen die durchschnittlichen Ausgaben pro Invalide in einem entsprechenden Jahr berechnet wurden. Nachfolgend ein Beispiel zur Illustration, nochmals Anhand der Weiterentwicklung der IV 2022 (siehe 4.5.2): In den Jahren ab 2023 werden nur die Jahre ab 2023 für die Berechnung der durchschnittlichen Ausgaben pro Invalide verwendet. Nehmen wir nun an, dass wir den Finanzhaushalt im Jahr 2024 erstellen.

Da wir über Registerdaten bis und mit 2023 verfügen, entspricht für die Jahre ab 2023 der Durchschnitt der Ausgaben pro Invalide (`mean_ausgaben_invalide`) den Ausgaben pro Invalide im Jahr 2023. Somit entspricht `mean_ausgaben_jahr` in 2023 2023. Für das Jahr 2022 entspricht `mean_ausgaben_invalide` den Ausgaben pro Invalide im Jahr 2022, und `mean_ausgaben_jahr` 2022. Für die Jahre vor 2022 entspricht `mean_ausgaben_invalide` dem Mittelwert der Ausgaben zwischen 2014 und 2021, und `mean_ausgaben_jahr` $(2014 + 2015 + 2016 + 2017 + 2018 + 2019 + 2020 + 2021)/8 = 2017.5$ dem Mittelwert der verbleibenden 8 Jahren vor 2022. `mean_ausgaben_jahr` wird hier berechnet, weil es später bei der Schätzung des Ausgabenwachstums bei den Ausgaben pro Invalide benötigt wird.

4.5.4 Wachstum der Ausgaben pro Invalide

Das Wachstum der Ausgaben pro Invalide nach Alter und Jahr wird anhand der folgenden Codezeilen berechnet:

```
#------ Geometrisches Mittel des Ausgabenwachstums berechnen ----------#
MEAN_WACHSTUM_AUSGABEN <- IV_MED_MASSN_REGISTER_REAL %>%
    arrange(alt, jahr) %>%
    mutate(ausgaben_invalide = ausgaben_real/invalide) %>%
    filter(jahr > letztes_registerjahr - 10 & jahr <= letztes_registerjahr) %>%
    group_by(alt) %>%
    mutate(delta_ausgaben_invalide = ausgaben_invalide/lag(ausgaben_invalide) -
        1) %>%
    ungroup() %>%
    select(jahr, alt, delta_ausgaben_invalide) %>%
    anti_join(IV_MED_MASSN_SONDEREFFEKTE %>%
        filter(sondereffekt %in% c("permanent_ausgaben", "temporaer_ausgaben")) %>%
        mutate(jahr = if_else(sondereffekt == "temporaer_ausgaben",
            sondereffektjahr + 0:1, sondereffektjahr)), by = c(jahr = "sondereffektjahr",
        alt = "sondereffektalter")) %>%
    group_by(alt) %>%
    summarise(delta_ausgaben_invalide = exp(mean(log(delta_ausgaben_invalide +
        1), na.rm = TRUE)) - 1, .groups = "drop")
```

Genau wie bei der Berechnung der durchschnittlichen Ausgaben pro Invalide werden zuerst die Ausgaben pro Invalide berechnet. Danach wird das Jahr-zu-Jahr-Wachstum der Ausgaben pro Invalide berechnet.

Die Sondereffektjahre werden nach den folgenden Kriterien ausgeschlossen:

- Temporäre Sondereffekte: Es werden sowohl das Ausgabenwachstum für das Jahr, in welchem der temporäre Ausgaben-Sondereffekt in der entsprechenden Altersgruppe präsent war, als auch das Jahr nach dem Sondereffekt, von der Berechnung des Durchschnittswachstums der Ausgaben pro Invalide ausgeschlossen. Der Grund liegt darin, dass ein temporärer Sondereffekt sowohl für das Sondereffektjahr als auch für das daraffolgende Jahr zu verzerrten Wachstumsraten führt (da ja das Wachstum im Folgejahr relativ zum Wert im Sondereffektjahr berechnet wird).
- Permanente Sondereffekte: Es wird nur das Ausgabenwachstum für das Jahr, in welchem der permanente Ausgaben-Sondereffekt in der entsprechenden Altersgruppe präsent war von der Berechnung des Durchschnittswachstums der Ausgaben pro Invalide ausgeschlossen.

Nach Ausschluss der Sondereffekte wird das Durchschnittswachstum der Ausgaben pro Invalide nach Altersgruppe berechnet. Das Durchschnittswachstum der Ausgaben pro Invalide ist daher, im Gegensatz zur Invalidisierung und den durchschnittlichen Ausgaben pro Invalide, nicht jahr-spezifisch.

4.5.5 Parameterwerte für die Projektion extrahieren

Die vorangehend berechneten altersspezifischen Durchschnitte für die Invalidisierung, die Ausgaben pro Invalide, und das Wachstum der Ausgaben pro Invalide bilden die Grundlage für die Projektionen. Für die

durchschnittliche Invalidisierung und die durchschnittlichen Ausgaben pro Invalide, welche sich über die Jahre verändern können (siehe oben), verwenden wir grundsätzlich den Wert des letzten verfügbaren Registerjahres. Einzige Ausnahme ist, wenn das letzte verfügbare Registerjahr von einem temporären Sondereffekt betroffen ist. In diesem Fall wird das vorangehende Jahr verwenden.

Die für die Prognose massgeblichen Mittelwerte werden somit wie folgt in einem Dataframe zusammengefügt:

```
AVERAGES <- MEAN_INVALIDISIERUNG_YEARS %>%
    full_join(MEAN_AUSGABEN_YEARS) %>%
    full_join(MEAN_WACHSTUM_AUSGABEN)

FORECAST_AUSGABEN <- AVERAGES %>%
    anti_join(IV_MED_MASSN_SONDEREFFEKTE %>%
        filter(sondereffekt == "temporaer_ausgaben") %>%
        rename(alt = sondereffektalter, jahr = sondereffektjahr)) %>%
    group_by(alt) %>%
    filter(jahr == max(jahr)) %>%
    ungroup() %>%
    select(alt, ausgaben_invalide, mean_ausgaben_jahr)

FORECAST_INVALIDISIERUNG <- AVERAGES %>%
    anti_join(IV_MED_MASSN_SONDEREFFEKTE %>%
        filter(sondereffekt == "temporaer_invalidisierung") %>%
        rename(alt = sondereffektalter, jahr = sondereffektjahr)) %>%
    group_by(alt) %>%
    filter(jahr == max(jahr)) %>%
    ungroup() %>%
    select(alt, invalidisierung)

IV_MED_MASSN_FORECAST_AVERAGES <- AVERAGES %>%
    filter(jahr == letztes_registerjahr) %>%
    select(alt, delta_ausgaben_invalide) %>%
    left_join(FORECAST_AUSGABEN) %>%
    left_join(FORECAST_INVALIDISIERUNG)
```

4.5.6 Manuelle Parameterkorrekturen einbinden

Nachdem die für die Prognose massgeblichen Mittelwerte in einem Dataframe zusammengefügt wurden, werde die manuellen Parameterkorrekturen wie folgt einbezogen:

```
# Korrekturen aus PARAMCHANGES einbeziehen Schritt 1: Führe
# einen left_join durch, um die dataframes zu verbinden
IV_MED_MASSN_FORECAST_AVERAGES <- left_join(IV_MED_MASSN_FORECAST_AVERAGES,
    IV_MED_MASSN_PARAMCHANGES, by = "alt", suffix = c("_forecast",
        "_paramchanges"))

# Schritt 2: Ersetze Werte in
# IV_MED_MASSN_FORECAST_AVERAGES durch nicht-NA-Werte aus
# IV_MED_MASSN_PARAMCHANGES wenn vorhanden
IV_MED_MASSN_FORECAST_AVERAGES <- IV_MED_MASSN_FORECAST_AVERAGES %>%
    mutate(delta_ausgaben_invalide = coalesce(delta_ausgaben_invalide_paramchanges,
        delta_ausgaben_invalide_forecast), ausgaben_invalide =
↪   coalesce(ausgaben_invalide_paramchanges,
        ausgaben_invalide_forecast), mean_ausgaben_jahr =
↪   coalesce(mean_ausgaben_jahr_paramchanges,
```

```
        mean_ausgaben_jahr_forecast), invalidisierung =
↪   coalesce(invalidisierung_paramchanges,
        invalidisierung_forecast)) %>%
    select(alt, delta_ausgaben_invalide, ausgaben_invalide, mean_ausgaben_jahr,
        invalidisierung)
```

Sowohl das Dataframe `IV_MED_MASSN_FORECAST_AVERAGES` als auch `IV_MED_MASSN_FORECAST_AVERAGES` enthalten die Spalten `delta_ausgaben_invalide`, `ausgaben_invalide`, `mean_ausgaben_jahr` und `invalidisierung`, sowie 27 Zeilen für jedes Alter von 0-21 Jahre. Das Skript prüft nun, ob im File `IV_MED_MASSN_PARAMCHANGES` bei gewissen Altern für diese Felder ein Parameterchange spezifiziert wurde (per Default entsprechen alle Felder im Dataframe `IV_MED_MASSN_PARAMCHANGES` dem Wert `NA`). Ist dies der Fall, so wird durch die Funktion `coalesce()` der Wert aus `IV_MED_MASSN_PARAMCHANGES` anstelle des Wertes aus `IV_MED_MASSN_FORECAST_AVERAGES` verwendet, d.h. der entsprechende Parameter wird ersetzt.

Zweck des Paramchanges-Files ist, dass damit nachdem der Finanzhaushalt einmal durchgelaufen ist, Nachjustierungen bei den Prognoseparametern des Modells vorgenommen werden können. So kann beispielsweise im Fall, dass aufgrund einer zukünftigen Reform der IV eine Veränderung der durchschnittlichen Ausgaben pro Invalide in einer bestimmten Altersgruppe erwartet wird, dies bereits anhand des Paramchanges-Files miteinbezogen werden. Die für die Prognose verwendeten Parameterwerte werden im Finanzhaushalt-Outputfile IV_MED_MASSN_FORECAST_AVERAGES.csv abgespeichert.

4.5.7 Modellprojektionen berechnen

Die Projektionen für die Gesamtausgaben für medizinische Massnahmen werden gebildet, indem pro Alter zuerst die Ausgaben pro invalide geschätzt werden (`ausgaben_invalide_hat`), diese danach mit der Anzahl Invalide (gegeben durch `invalidisierung*bevendejahr`) multipliziert werden, und am Ende über alle Alter aufsummiert wird. Dies geschieht im nachfolgenden Code im unteren Teil:

```
PAST_ROWS <- AVERAGES %>%
    left_join(BEVOELKERUNG %>%
        group_by(jahr, alt) %>%
        summarise(bevendejahr = sum(bevendejahr), .groups = "drop"))

FUTURE_ROWS <- BEVOELKERUNG %>%
    group_by(jahr, alt) %>%
    summarise(bevendejahr = sum(bevendejahr), .groups = "drop") %>%
    filter(alt <= 26, jahr > letztes_registerjahr) %>%
    left_join(IV_MED_MASSN_FORECAST_AVERAGES)

AUSGABEN_DWH_HAT <- bind_rows(PAST_ROWS, FUTURE_ROWS) %>%
    arrange(alt, jahr) %>%
    mutate(ausgaben_invalide_hat = ausgaben_invalide * (1 +
    ↪   delta_ausgaben_invalide)^(jahr -
        mean_ausgaben_jahr)) %>%
    mutate(ausgaben_mm_dwh_hat = invalidisierung * bevendejahr *
        ausgaben_invalide_hat) %>%
    group_by(jahr) %>%
    summarize(ausgaben_mm_dwh_hat = sum(ausgaben_mm_dwh_hat,
        na.rm = TRUE)) %>%
    ungroup() %>%
    left_join(MM_ABRECHNUNG_REAL)
```

Vor dieser Berechnung werden die Daten für die Projektionen separat für die Vergangenheit (`PAST_ROWS`) und die Zukunft (`FUTURE_ROWS`) berechnet. Für die Vergangenheit werden die Jahres-spezifischen Werte verwendet, welche im Dataframe `AVERAGES` enthalten sind. Für die Zukunft werden die Parameterwerte

aus dem Dataframe `IV_MED_MASSN_FORECAST_AVERAGES` verwendet, welches manuelle Parameterkorrekturen miteinbezieht (siehe vorangehender Abschnitt).

4.5.8 Justierung Registerdaten Abrechnung

Die Projektionen in `AUSGABEN_DWH_HAT` basieren ausschliesslich auf Rechnungsdaten aus dem Datawarehouse-Register. Es hat sich gezeigt, dass die Summe der Rechnungen aus dem Datawarehouse-Register in der jüngeren Vergangenheit 1-2% unter den in der IV-Abrechnung ausgagebenen Gesamtausgaben für medizinische Massnahmen lagen.[33] Wir korrigieren daher mit dem nachfolgenden Code für die Abweichung zwischen den Modell-Rechnungssummen und den Ausgaben gemäss IV-Abrechnung:

```
# Registerdaten auf Abrechnung justieren
REG_MODEL <- stats::lm(mm_real ~ ausgaben_mm_dwh_hat - 1, data = AUSGABEN_DWH_HAT %>%
    filter(jahr <= letztes_registerjahr & jahr > letztes_registerjahr -
        10))

# Add predictions as a new column to the dataset
IV_MM_PROJ <- AUSGABEN_DWH_HAT %>%
    mutate(predicted_mm_real = predict(REG_MODEL, newdata = AUSGABEN_DWH_HAT)) %>%
    left_join(DISKONTFAKTOR) %>%
    mutate(mm = predicted_mm_real/diskontfaktor) %>%
    select(jahr, mm) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr)
```

Konkret regressieren wir die Ausgaben für medizinische Massnahmen aus der IV-Abrechnung (`mm_real`) auf die durch unser Modell geschätzten Ausgaben (`ausgaben_mm_dwh_hat`), ohne einen Intercept in die Regression einzubeziehen. Dadurch entspricht der Koeffizient von `ausgaben_mm_dwh_hat` dem Faktor, mit welchem `ausgaben_mm_dwh_hat` die tatsächlichen Ausgaben `mm_real` in den letzten 10 Jahren durchschnittlich unterschätzt hat (also bspw. 1.01 bei einer Unterschätzung von 1%).

Die Justierung erfolgt dann im unteren Teil von obigem Code, indem alle Werte (also auch zukünftige) von `ausgaben_mm_dwh_hat` mit dem in der Regression geschätzten Koeffizienten multipliziert werden (via die `predict()`-Funktion).

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden. Dies geschieht mit den folgenden abschliessenden Codezeilen des Moduls:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_MM_ABR <- IV_ABRECHNUNG %>%
    select(jahr, mm)

#------ Total Medizinische Massnahmen -------------------------------#
IV_MM <- rbind(IV_MM_ABR, IV_MM_PROJ)

#------ Output ------------------------------------------------------#
mod_return(IV_MED_MASSN_FORECAST_AVERAGES, IV_MM)
```

4.6 Modul mod_iv_fi.R

Dokumentation zuletzt aktualisiert am 17.09.2024

[33] Gründe hierfür sind, unter anderen, dass gewisse Spezialbehandlungen nicht über das Sumex-System (auf welchem die Datawarehouse-Registerdaten basieren) verbucht wurden, sondern die Verbuchung manuell von der ZAS direkt in der IV-Betriebsrechnung vorgenommen wurde.

Das Prognosemodell zu den Frühinterventionsmassnahmen ist im Skript `mod_iv_fi.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_fi <- mod_iv_fi(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_fi.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_FI_ABR <- IV_ABRECHNUNG %>%
    select(jahr, fi)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Ausgaben für Frühinterventionsmassnahmen ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("fi")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Frühinterventionsmassnahmen im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_FI_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, fi), by = c("jahr")) %>%
    mutate(fi = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        fi, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Frühinterventionsmassnahmen ----------------------------#
IV_FI <- rbind(IV_FI_ABR, IV_FI_PROJ)
```

```
#------ Output -------------------------------------------------------#
mod_return(IV_FI)
```

4.7 Modul mod_iv_ber_begl.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den Ausgaben für Beratung und Begleitung ist im Skript `mod_iv_ber_begl.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_ber_begl <- mod_iv_ber_begl(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_ber_begl.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung -----------------------------------------#
IV_BER_BEGL_ABR <- IV_ABRECHNUNG %>%
    select(jahr, ber_begl)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Ausgaben für Beratung und Begleitung ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("ber_begl")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Beratung und Begleitung im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_BER_BEGL_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, ber_begl), by = c("jahr")) %>%
    mutate(ber_begl = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        ber_begl, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr

`PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Beratung und Begleitung ---------------------------------#
IV_BER_BEGL <- rbind(IV_BER_BEGL_ABR, IV_BER_BEGL_PROJ)

#------ Output --------------------------------------------------------#
mod_return(IV_BER_BEGL)
```

4.8 Modul mod_iv_im.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den Integrationsmassnahmen ist im Skript `mod_iv_im.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_im <- mod_iv_im(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_im.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ------------------------------------------#
IV_IM_ABR <- IV_ABRECHNUNG %>%
    select(jahr, im)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Ausgaben für Integrationsmassnahmen ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("im")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Integrationsmassnahmen im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_IM_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, im), by = c("jahr")) %>%
```

```
    mutate(im = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        im, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Integrationsmassnahmen ---------------------------------#
IV_IM <- rbind(IV_IM_ABR, IV_IM_PROJ)

#------ Output -------------------------------------------------------#
mod_return(IV_IM)
```

4.9 Modul mod_iv_mba.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den Massnahmen beruflicher Art ist im Skript `mod_iv_mba.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_mba <- mod_iv_mba(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_mba.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung -----------------------------------------#
IV_MBA_ABR <- IV_ABRECHNUNG %>%
    select(jahr, mba)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Ausgaben für Massnahmen beruflicher Art ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("mba")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Massnahmen der beruflichen Art im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_MBA_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, mba), by = c("jahr")) %>%
    mutate(mba = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        mba, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Massnahmen der beruflichen Art --------------------------#
IV_MBA <- rbind(IV_MBA_ABR, IV_MBA_PROJ)

#------ Output --------------------------------------------------------#
mod_return(IV_MBA)
```

4.10 Modul mod_iv_aus_uebr.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den übrigen Kosten der beruflichen Eingliederung ist im Skript `mod_iv_aus_uebr.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_aus_uebr <- mod_iv_aus_uebr(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_aus_uebr.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ------------------------------------------#
IV_AUS_UEBR_ABR <- IV_ABRECHNUNG %>%
    select(jahr, aus_uebr)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die übrigen Kosten der beruflichen Eingliederung ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
```

```
    1), c("aus_uebr")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für übrige Ausgaben (MBA) im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_AUS_UEBR_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, aus_uebr), by = c("jahr")) %>%
    mutate(aus_uebr = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        aus_uebr, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total übrige Ausgaen Massnahmen beruflicher Art ---------------#
IV_AUS_UEBR <- rbind(IV_AUS_UEBR_ABR, IV_AUS_UEBR_PROJ)

#------ Output ------------------------------------------------------------#
mod_return(IV_AUS_UEBR)
```

4.11 Modul mod_iv_hm.R

Dokumentation zuletzt aktualisiert am 06.09.2024

Das Prognosemodell zu den Hilfsmitteln ist im Skript `mod_iv_hm.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_hm <- mod_iv_hm(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, BEVOELKERUNG = BEVOELKERUNG,
    DISKONTFAKTOR = DISKONTFAKTOR)
```

Das Modul `mod_iv_hm.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `BEVOELKERUNG`: Die Fortschreibung der Hilfsmittel basiert auf dem Wachstum der Wohnbevölkerung.
- `DISKONTFAKTOR`: Der Diskontfaktor benötigt, um die Fortschreibung um das Wachstum des Landesindexes der Konsumentenpreise zu skalieren.

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_HM_ABR <- IV_ABRECHNUNG %>%
    select(jahr, hm)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Hilfsmittelausgaben ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit den Rentenausgaben --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("hm")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Hilfsmittel im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Als nächstes wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet:

```
  # Idee: Im 2023 fiel genau (bis auf Rundungen) die Hälfte der IV-Hilfsmittelkosten für
  ↪  Personen unter 50 an, und die andere Hälfte für Personen über 50.
  # Das reale Wachstum der Hilfsmittelausgaben für die Fortschreibung nach dem
  ↪  Umfragehorizont soll deshalb aus dem Mittelwert des Bevölkerungswachstums dieser
  ↪  zwei Personengruppen bestehen.
  # Erstelle die zwei Gruppen unter50 und 50bis65 und berechne die aggregierten Werte
  HM_WACHSTUMSFAKTOR <- BEVOELKERUNG %>%
      filter(alt<65) %>%
      mutate(gruppe = case_when(
          alt < 50 ~ "unter50",
          alt >= 50 & alt < 65 ~ "50bis65"
      )) %>%
      group_by(gruppe, jahr) %>%
      summarise(bevendejahr = sum(bevendejahr, na.rm = TRUE), .groups = 'drop') %>%
      ungroup() %>%
      group_by(gruppe) %>% # Berechne den Bevölkerungswachstumsfaktor basierend auf dem
      ↪  Jahr PARAM_GLOBAL$jahr_umfragen_fs
      mutate(hm_wachstumsfaktor_real = bevendejahr / bevendejahr[jahr ==
      ↪  PARAM_GLOBAL$jahr_umfragen_fs]) %>%
      ungroup() %>%
      group_by(jahr) %>%
      summarize(hm_wachstumsfaktor_real=mean(hm_wachstumsfaktor_real)) %>%
      left_join(DISKONTFAKTOR) %>%
      mutate(diskontfaktor = diskontfaktor / diskontfaktor[jahr ==
      ↪  PARAM_GLOBAL$jahr_umfragen_fs]) %>% # Diskontfaktor einstellen, dass er im Jahr
      ↪  PARAM_GLOBAL$jahr_umfragen_fs==1 ist
      mutate(hm_wachstumsfaktor_nom=hm_wachstumsfaktor_real/diskontfaktor) # Durch die
      ↪  division durch den Diskontfaktor wird die kummulierte Inflation ab dem Jahr
      ↪  PARAM_GLOBAL$jahr_umfragen_fs berechnet.
```

Das Wachstum der Ausgaben für Hilfsmittel wird mit dem gewichteten Bevölkerungswachstum multipliziert mit dem Wachstum des Landesindexes der Konsumentenpreise fortgeschrieben. Die genaue Umsetztung ist anhand der Code-Kommentare ersichtlich.

Als nächstes wird der Projektionsvektor gebildet, wobei dieser bis zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs` (was Stand 2024 dem Abrechnungsjahr +5 entspricht) den Werten aus der Umfrage entspricht, und danach

der Fortschreibung des letzten Umfragewertes mit dem oben berechneten `hm_wachstumsfaktor_nom`:

```
IV_HM_PROJ <- HM_WACHSTUMSFAKTOR %>%
    select(jahr, hm_wachstumsfaktor_nom) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, hm), by = c("jahr")) %>%
    mutate(hm = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        hm, last_umfr_val * hm_wachstumsfaktor_nom)) %>%
    select(-hm_wachstumsfaktor_nom)
```

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Hilfsmittel ------------------------------------------------#
IV_HM <- rbind(IV_HM_ABR, IV_HM_PROJ)

#------ Output -----------------------------------------------------------#
mod_return(IV_HM)
```

4.12 Modul mod_iv_rk.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den Reisekosten ist im Skript `mod_iv_rk.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_aus_uebr <- mod_iv_aus_uebr(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_rk.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ---------------------------------------------#
IV_RK_ABR <- IV_ABRECHNUNG %>%
    select(jahr, reise_kost)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Reisekosten ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit dem Lohnindex --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("reise_kost")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Reisekosten im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
```

```
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_RK_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, reise_kost), by = c("jahr")) %>%
    mutate(reise_kost = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        reise_kost, last_umfr_val * lohnidx)) %>%
    select(-lohnidx)
```

dh., es wird der Lohnindex (nominale Schweizerische Lohnindex) für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieser ist in der Variable `lohnidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Reisekosten ------------------------------------------------#
IV_RK <- rbind(IV_RK_ABR, IV_RK_PROJ)

#------ Output -----------------------------------------------------------#
mod_return(IV_RK)
```

4.13 Modul mod_iv_assb.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zum Assistenzbeitrag ist im Skript `mod_iv_assb.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_assb <- mod_iv_assb(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, IV_GELDLEISTUNG = IV_GELDLEISTUNG)
```

Das Modul `mod_iv_assb.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `IV_GELDLEISTUNG`: Das prjizierte Ausgabenwachstum für Hiflosenentschädigungen wird für die Fortschreibung der Ausgaben für den Asssistenzbeitrag verwendet. Die projizierten Ausgaben für Hiflosenentschädigungen sind im Dataframe `IV_GELDLEISTUNG` enthalten.

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ---------------------------------------------#
IV_ASSB_ABR <- IV_ABRECHNUNG %>%
    select(jahr, btr_ass)
```

Als nächstes wird der letzte Wert der Umfrageprojektion zum Assistenzbeitrag ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Ausgaben für Hilflosenentschädigung
↪  --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("btr_ass")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für den Assistenzbeitrag im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
startwert <- as.numeric(IV_GELDLEISTUNG[IV_GELDLEISTUNG$jahr ==
    (PARAM_GLOBAL$jahr_umfragen_fs - 1), c("he_iv")])
IV_ASSB_PROJ <- IV_GELDLEISTUNG %>%
    select(jahr, he_iv) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, btr_ass), by = c("jahr")) %>%
    mutate(btr_ass = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        btr_ass, last_umfr_val * he_iv/startwert)) %>%
    select(-he_iv)
```

dh., es wird zuerst der Wert der Ausgaben für Hilflosenentschädigungen im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` in der Variable `startwert` gespeichert. Dieser Startwert wird dann verwendet, um das Wachstum der Ausgaben für Hilflosenentschädigungen in einem Jahr relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` zu berechnen, was in der zweitletzten Zeile des obigen Codes geschieht.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Assistenzbeitrag ----------------------------------------#
IV_ASSB <- rbind(IV_ASSB_ABR, IV_ASSB_PROJ)

#------ Output --------------------------------------------------------#
mod_return(IV_ASSB)
```

4.14 Modul mod_iv_rueck_im.R

Dokumentation zuletzt aktualisiert am 17.09.2024

Das Prognosemodell zu den Rückerstattungsforderungen für individuelle Massnahmen ist im Skript `mod_iv_rueck_im.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_rueck_im <- mod_iv_rueck_im(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_rueck_im.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.

- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_RUECK_IM_ABR <- IV_ABRECHNUNG %>%
    select(jahr, rueck_erstattungsf_ind, abschr_rueck_ind_mass) %>%
    mutate(rueck_erstattungsf_ind = rueck_erstattungsf_ind +
        abschr_rueck_ind_mass) %>%
    select(-abschr_rueck_ind_mass)
```

Als nächstes wird der letzte Wert der Umfrageprojektion zu den Rückerstattungsforderungen für individuelle Massnahmen ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Letztes Jahr der Abrechnung  ---------------------------------#
last_abr_val <- as.numeric(IV_RUECK_IM_ABR[IV_RUECK_IM_ABR$jahr ==
    PARAM_GLOBAL$jahr_abr, c("rueck_erstattungsf_ind")])
if (is.na(last_abr_val)) {
    warnings(paste("fehlender Wert für Rückforderungen bei individuellen Massnahmen im
    ↪  Jahr",
        PARAM_GLOBAL$jahr_abr))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
# Fortschreibung mit Lohnsumme
IV_RUECK_IM_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsumme) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    mutate(lohnsummeidx = cumprod(1 + lohnsumme/100)) %>%
    mutate(rueck_erstattungsf_ind = last_abr_val * lohnsummeidx) %>%
    select(-lohnsummeidx, -lohnsumme)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_sachleistungen.R`) zurückgegeben werden:

```
#------ Total Rückforderungen individuelle Massnahmen ----------------#
IV_RUECK_IM <- rbind(IV_RUECK_IM_ABR, IV_RUECK_IM_PROJ)

#------ Output -------------------------------------------------------#
mod_return(IV_RUECK_IM)
```

4.15 Modul mod_iv_institutionen.R

Dokumentation zuletzt aktualisiert am 18.09.2024

Die Prognosemodelle für die Beiträge an Organisationen sowie die Beiträge Pro Infirmis sind im Skript `mod_iv_institutionen.R` enthalten, welches wie folgt im Skript `mod_iv_sachleistungen.R` aufgerufen wird:

```
tl_iv_institutionen <- mod_iv_institutionen(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR =
    ↪  DISKONTFAKTOR)
```

Das Modul `mod_iv_institutionen.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `DISKONTFAKTOR`: Zur Fortschreibung der nominalen Beiträge an Organisationen mit dem Landesindex der Konsumentenpreise wird der (inverse) Diskontfaktor verwendet.

Innerhalb des Moduls werden nacheinander die Module für die Projektion der Beiträge an Organisationen respektive der Beiträg an Pro Infirmis aufgerufen:

```
#------ Beiträge an Organisationen / Subventions aux organisations ----#
tl_iv_btr_org <- mod_iv_btr_org(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR =
    ↪  DISKONTFAKTOR)

#------ Beiträge an Pro Infirmis / Subventions à Pro Infirmis ---------#
tl_iv_btr_proinf <- mod_iv_btr_proinf(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR =
    ↪  DISKONTFAKTOR)
```

Da die beiden Module identisch aufgebaut sind wird hier nur das Modul `mod_iv_btr_org.R` erläutert:

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_BTR_ORG_ABR <- IV_ABRECHNUNG %>%
    select(jahr, btr_beh) %>%
    rename(btr_org = btr_beh)
```

Als nächstes wird der letzte Wert der Umfrageprojektion für die Beiträge an Organisationen ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit dem Lohnindex --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("btr_org")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Beiträge an Organisationen im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_BTR_ORG_PROJ <- DISKONTFAKTOR %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, btr_org), by = c("jahr")) %>%
    mutate(btr_org = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
```

```
        btr_org, last_umfr_val/diskontfaktor)) %>%
    select(-diskontfaktor)
```

dh., es wird die Inflation für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Modul `mod_iv_institutionen.R` zurückgegeben werden:

```
#------ Total Integrationsmassnahmen ---------------------------------#
IV_BTR_ORG <- rbind(IV_BTR_ORG_ABR, IV_BTR_ORG_PROJ)

#------ Output -------------------------------------------------------#
mod_return(IV_BTR_ORG)
```

In `mod_iv_institutionen.R` wird dann die Summe aus den Beiträgen an Organisationen und den Beiträgen an Pro Infirmis gebildet:

```
#------ Total Institutionen ------------------------------------------#
IV_INSTITUTIONEN <- tl_iv_btr_org$IV_BTR_ORG %>%
    inner_join(tl_iv_btr_proinf$IV_PROINF, by = c("jahr")) %>%
    mutate(btr_inst_org_iv = btr_org + btr_proinf)
```

Auf den nachfolgenden Codeteil wird nicht näher eingegangen, da der Parameter `all_felder_massdb` Stand 2024 `=FALSE` ist:

```
#------ nicht mehr gebuchte Kosten -----------------------------------#
if (PARAM_GLOBAL$all_felder_massdb) {
    #------ Beiträge an Institutionen / Subventions aux institutions ------#
    #------ Baubeiträge / Subventions aux constructions -------------------#
    #------ Betriebsbeiträge / Subventions frais d'exploitation -----------#
    #------ Arbeitsämter, Berufsberatung und Spezialstellen ---------------#
    ABR_MASSDB <- IV_ABRECHNUNG %>%
        select(jahr, btr_bau, btr_betr, btr_berat) %>%
        rbind(data.frame(jahr = c((PARAM_GLOBAL$jahr_abr + 1):PARAM_GLOBAL$jahr_ende))
        ↪  %>%
            mutate(btr_bau = 0, btr_betr = 0, btr_berat = 0))

    IV_INSTITUTIONEN <- IV_INSTITUTIONEN %>%
        inner_join(ABR_MASSDB, by = c("jahr")) %>%
        mutate(btr_beh = btr_bau + btr_betr + btr_org)
}
```

Somit können die Durchführungskosten an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output -------------------------------------------------------#
mod_return(IV_INSTITUTIONEN)
```

4.16 Modul mod_iv_durchfuehrungskosten.R

Dokumentation zuletzt aktualisiert am 20.09.2024

Das Prognosemodell zu den Durchführungskosten ist im Skript `mod_iv_durchfuehrungskosten.R` enthalten,

welches wie folgt im Skript `mod_iv_uebrigeausgaben.R` aufgerufen wird:

```
tl_iv_durchfuehrungskosten <- mod_iv_durchfuehrungskosten(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_durchfuehrungskosten.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne (Variable `lohnsumme`).

Zu Beginn des Moduls wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ----------------------------------------#
IV_DURCHF_ABR <- IV_ABRECHNUNG %>%
    select(jahr, df_kost, abklaer_mass, parteient_gericht)
```

Als nächstes werden die Anteile der Abklärungsmassnahmen und der Parteienentschädigung an den Durchführungskosten im Abrechnungsjahr gebildet. Dies dient später dazu, die gesamten Durchführungskosten auf diese zwei Positionen aufzuteilen:

```
IV_DFA <- IV_DURCHF_ABR %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
    mutate(abklaer_mass = abklaer_mass/df_kost, parteient_gericht =
    ↪  parteient_gericht/df_kost) %>%
    select(abklaer_mass, parteient_gericht)
```

Als nächstes wird der letzte Wert der Umfrageprojektion zu den Durchführungskosten ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Lohnsumme --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("df_kost")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für die Durchführungskosten im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_DURCHF_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, df_kost), by = c("jahr")) %>%
    mutate(df_kost = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        df_kost, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx) %>%
    #------ Anteile aus dem Abrechnungsjahr applizieren -------------------#
#------ Abklärungsmassnahmen / Mesures d'instruction ------------------#
```

```
#------ Kosten und Parteientschädigungen / Frais et dépens ------------#
merge(., IV_DFA) %>%
    mutate(abklaer_mass = abklaer_mass * df_kost, parteient_gericht = parteient_gericht *
        df_kost)
```

dh., es wird das Lohnsummenwachstum für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieses ist in der Variable `lonsummeidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt. Zum Schluss werden die Durchführungskosten wieder auf die zwei Abrechnungspositionen aufgeteilt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt werden:

```
#------ Total Durchführungskosten -------------------------------------#
IV_DURCHFUEHRUNG <- rbind(IV_DURCHF_ABR, IV_DURCHF_PROJ) %>%
    rename(df_kost_iv = df_kost)
```

Auf den nachfolgenden Codeteil wird nicht näher eingegangen, da der Parameter `all_felder_massdb` Stand 2024 `=FALSE` ist:

```
#------ nicht mehr gebuchte Kosten ------------------------------------#
#------ Sekretariat IV-Kommissionen / Secrétariats des commissions AI -#
#------ IV-Kommissionen / Commissions AI ------------------------------#
#------ IV-Regionalstel-len / Offices régionaux AI --------------------#
#------ Spezialstellen / Services spéciaux ----------------------------#
#------ Abklärungs-massnahmen / Frais d'examens -----------------------#
#------ IV-Stellen / IV-Stellen ---------------------------------------#
if (PARAM_GLOBAL$all_felder_massdb) {
    ABR_MASSDB <- IV_ABRECHNUNG %>%
        select(jahr, sekr_iv_kom, iv_kom, iv_reg_ste, spez_st,
            iv_ste) %>%
        rbind(data.frame(jahr = c((PARAM_GLOBAL$jahr_abr + 1):PARAM_GLOBAL$jahr_ende))
        ↪  %>%
            mutate(sekr_iv_kom = 0, iv_kom = 0, iv_reg_ste = 0,
                spez_st = 0, iv_ste = 0))

    IV_DURCHFUEHRUNG <- IV_DURCHFUEHRUNG %>%
        inner_join(ABR_MASSDB, by = c("jahr")) %>%
        mutate(df_kost_iv = df_kost_iv + sekr_iv_kom + iv_kom +
            iv_reg_ste + spez_st + iv_ste)
}
```

Somit können die Durchführungskosten an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output --------------------------------------------------------#
mod_return(IV_DURCHFUEHRUNG)
```

4.17 Modul mod_iv_verwaltungsaufwand.R

Dokumentation zuletzt aktualisiert am 20.09.2024

Das Prognosemodell zu den Durchführungskosten ist im Skript `mod_iv_verwaltungsaufwand.R` enthalten, welches wie folgt im Skript `mod_iv_uebrigeausgaben.R` aufgerufen wird:

```
tl_iv_durchfuehrungskosten <- mod_iv_durchfuehrungskosten(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

Das Modul `mod_iv_verwaltungsaufwand.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `UMFRAGE_IV`: Die ersten fünf Jahre der Fortschreibung basieren auf den vom zuständigen Fachbereich im BSV gelieferten Umfragewerten.
- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der projizierten Summe der in der Schweiz ausbezahlten Löhne, sowie des projizierten Lohnindexes.

Im Modul `mod_iv_verwaltungsaufwand.R` werden nacheinander die folgenden Untermodule aufgerufen:

```
#------ Posttaxen / Taxes postales ------------------------------------#
tl_iv_posttax <- mod_iv_posttax(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ Verwaltungskosten / Frais de gestion administrative -----------#
tl_iv_verwaltungskosten <- mod_iv_verwaltungskosten(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ Abschreibungen Imm. IV-Stellen / Amort. immeubles Offices AI --#
tl_iv_imm_absch <- mod_iv_imm_absch(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)

#------ IV-Stellen (inkl. RAD) / Offices AI (y compris SMR) -----------#
tl_iv_iv_ste <- mod_iv_iv_ste(PARAM_GLOBAL = PARAM_GLOBAL, UMFRAGE_IV = UMFRAGE_IV,
    IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```

Da alle Module gleich aufgebaut sind, wird hier die Struktur dieser Module nur am Beispiel des Moduls `mod_iv_posttax.R` erläutert:

Zu Beginn des Moduls `mod_iv_posttax` wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung -------------------------------------------#
IV_POSTTAX_ABR <- IV_ABRECHNUNG %>%
    select(jahr, post_taxen)
```

Als nächstes wird der letzte Wert der Umfrageprojektion zu den Durchführungskosten ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit dem Lohnindex --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("post_taxen")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für Posttaxen im Jahr", (PARAM_GLOBAL$jahr_umfragen_fs
    ↪  -
        1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
IV_POSTTAX_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, post_taxen), by = c("jahr")) %>%
    mutate(post_taxen = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        post_taxen, last_umfr_val * lohnidx)) %>%
    select(-lohnidx)
```

dh., es wird der Lohnindex (nominale Schweizerische Lohnindex) für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieser ist in der Variable `lohnidx` so abgebildet, dass diese Variable im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` 1 entspricht und danach das kummulierte Wachstum relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` auffängt.

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt werden:

```
#------ Total Posttaxen ------------------------------------------------#
IV_POSTTAX <- rbind(IV_POSTTAX_ABR, IV_POSTTAX_PROJ)

#------ Output ---------------------------------------------------------#
mod_return(IV_POSTTAX)
```

Die anderen oben erwähnten Module sind identisch aufgebaut, wobei bei `mod_iv_verwaltungskosten.R` und bei `mod_iv_iv_ste.R` anstatt des Lohnindexes der Lohnsummenindex für die Fortschreibung verwendet wird.

In der Fortsetzung des Codes im Modul `mod_iv_verwaltungsaufwand.R` werden die folgenden zwei Module aufgerufen:

```
#------ Kostenrückerstattungen / Remboursements de frais --------------#
tl_iv_rueck_vk <- mod_iv_rueck_vk(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG)

#------ Kosten Fondsverwaltung / Frais de gestion des Fonds -----------#
tl_iv_kost_fonds <- mod_iv_kost_fonds(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV)
```

In `mod_iv_rueck_vk` wird der Wert 0 projiziert, da dieser Ausgabeposten seit 2018 0 ist. In `mod_iv_kost_fonds`, dem Modul zur Schätzung der Fondsverwaltungskosten, wobei projiziert wird dass die dass die Fondsverwaltungskosten über die ersten 5 Jahre ab der aktuellsten Abrechnung um 3% p.a. wachsen, und danach mit dem Lohnindex.

Als nächstes wird der totale Verwaltungsaufwand aus der Summe der einzelnen Positionen berechnet:

```
#------ Total Verwaltungsaufwand ---------------------------------------#
IV_VERWALTUNGSAUFWAND <- tl_iv_posttax$IV_POSTTAX %>%
    inner_join(tl_iv_verwaltungskosten$IV_VERWALTUNGSKOSTEN,
        by = c("jahr")) %>%
    inner_join(tl_iv_imm_absch$IV_IMM_ABSCH, by = c("jahr")) %>%
    inner_join(tl_iv_iv_ste$IV_IV_STE, by = c("jahr")) %>%
    inner_join(tl_iv_rueck_vk$IV_RUECK_VK, by = c("jahr")) %>%
    inner_join(tl_iv_kost_fonds$IV_KOST_FONDS, by = c("jahr")) %>%
    mutate(verw_aufw_iv = post_taxen + verw_kost + abschr_iv_ste +
```

```
        iv_ste_rad + kost_rueck + kost_fonds_ant)
```

Auf den nachfolgenden Codeteil wird nicht näher eingegangen, da der Parameter `all_felder_massdb` Stand 2024 `=FALSE` ist:

```
#------ nicht mehr gebuchte Kosten ------------------------------------#
#------ Durchführungskosten gemäss IVG / Frais d'application, selon LAI
#------ Erlös aus Verkäufen und Arbeiten für Dritte (versch. Einnahmen) / Productions de
↪  ventes et de travaux pour tiers
# Die Erlöse wurden als Aufwandminderung bei der
# Verwaltungsaufwendungen gebucht.
if (PARAM_GLOBAL$all_felder_massdb) {
    ABR_MASSDB <- IV_ABRECHNUNG %>%
        select(jahr, df_kost_ivg, erl_verk_arb_dr) %>%
        rbind(data.frame(jahr = c((PARAM_GLOBAL$jahr_abr + 1):PARAM_GLOBAL$jahr_ende))
        ↪  %>%
            mutate(df_kost_ivg = 0, erl_verk_arb_dr = 0))

    IV_VERWALTUNGSAUFWAND <- IV_VERWALTUNGSAUFWAND %>%
        inner_join(ABR_MASSDB, by = c("jahr")) %>%
        mutate(verw_aufw_iv = verw_aufw_iv + df_kost_ivg + erl_verk_arb_dr)
}
```

Somit kann der Verwaltungsaufwand an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output --------------------------------------------------------#
mod_return(IV_VERWALTUNGSAUFWAND)
```

4.18 Module mod_iv_beitrag.R und mod_beitragssumme.R

Dokumentation zuletzt aktualisiert am 30.08.2024

Notiz Übergangsphase: Im Moment wird das neue Modul nur aufgerufen, falls im Input-Conatiner in `PARAM_GLOBAL` dem Parameter *beitragsmodell* der Wert *OLD* zugewiesen wird (eine neue Version des Moduls ist in Erarbeitung, kann aber noch nicht produktiv verwendet werden).

Das Prognosemodell zu den Beiträgen von Versicherten und Arbeitnehmern ist im Skript `mod_iv_beitrag.R` enthalten, welches wie folgt im Skript `mod_iv_einnahmen.R` aufgerufen wird:

```
tl_iv_beitrag <- mod_iv_beitrag(PARAM_GLOBAL = PARAM_GLOBAL,
    AKTIVE_BEV = AKTIVE_BEV, IK = IK, EINK_ENTWICKLUNG = EINK_ENTWICKLUNG,
    IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR)
```

Das Modul `mod_iv_mm_new.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `AKTIVE_BEV`: Die Wohnbevölkerung gemäss Statistik fortgeschrieben mit den Erwerbsbevölkerungsszenarien des BFS. Es werden ausschliesslich die Wachstumsraten über die Zeit innerhalb der Zellen für die Projektion verwendet werden.
- `IK`: Die Daten gemäss der individuellen Konten der ZAS.
- `EINK_ENTWICKLUNG`: Zeitreihe mit der Entwicklung des Durchschnittslohnes $eink\_entwicklung = \frac{SLI_t^{nominal} * (1 + Sturkturfaktor)}{SLI_{Abrechnungsjahr}^{nominal}}$. Wichtig ist hier der Spezialfall für die ersten zwei Jahre ab dem aktuellen Abrechnungsjahr (also Stand 2024 2024 und 2025). Für diese Periode enthält `eink_entwicklung` zusätzlich noch das Beschäftigungswachstum gemäss der Projektionen der BESTA, oder genauer gesagt die Abweichung des Beschäftigungswachstum gemäss BESTA von dem Erwerbsbevölkerungs-

wachstum gemäss Erwerbsbevölkerungsszenario des BFS. Die genaue Berechnung ist im Modul `mod_eink_entwicklung.R` (vgl. Kapitel 4.26) ersichtlich.
- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `DISKONTFAKTOR`: Da das Modell in realen grössen gerechnet wird, wird der Diskontfaktor benötigt, um nominale grössen vor der Modellberechnung in reale Grössen umzuwandeln, respektive um die realen Modellprognosen am Ende in nominale Grössen umzuwandeln.

4.18.1 Modul mod_beitragssumme.R

Als erstes wird das Modul `mod_beitragssumme.R` aufgerufen, wobei zuerst mit `PARAM_GLOBAL$vz <- 'iv'` sichergestellt wird, dass die Beitragssumme für die IV berechnet wird.[34]

```
PARAM_GLOBAL$vz <- "iv"

tl_iv_beitrag <- mod_beitragssumme(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = AKTIVE_BEV, IK = IK, EINK_ENTWICKLUNG = EINK_ENTWICKLUNG,
    ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR,
    BEITRAGSSATZ = as_tibble(data.frame()))
```

Auffällig ist hier insbesondere die Linie `BEVOELKERUNG = AKTIVE_BEV`. Es wird also bewusst angegeben, dass die Referenzbevölkerung für die Berechnung der Beitragssumme die aktive Bevölkerung sein soll. Dies illustriert den modularen Aufbau des Modells: Es wäre hier auch möglich, irgendeine andere Referenzbevölkerung anzugeben, bspw. die Wohnbevölkerung. Da wir die Entwicklung der aktiven Bevölkerung als für am relevantesten für die Beitragsentwicklung halten, verwenden wir diese für die Projektion der Beitragssumme.

Das Modul `mod_beitragssumme.R` beginnt mit dem folgenden Code:

```
IK <- IK %>%
    filter(vz == PARAM_GLOBAL$vz)

# --- IK Daten fuer alle Jahre
# ------------------------------------------------

D_IK_YEARS <- IK %>%
    na.omit() %>%
    mutate(alt = as.integer(alt)) %>%
    mutate(alt = if_else(alt <= 98L, alt, 99L)) %>%
    group_by(jahr, sex, nat, alt) %>%
    summarize(spersonen = sum(personen), slohnsumme = sum(einkommen_ik),
        sbeitrag_bsv = sum(beitrag_bsv)) %>%
    # Die Beitraegerquote wird mit derselben Bevoelkerung
    # berechnet, die auch fuer die Projektion verwendet
    # wird.
left_join(BEVOELKERUNG, by = c("jahr", "sex", "nat", "alt")) %>%
    mutate(beitraegerquote = spersonen/bevendejahr) %>%
    mutate(beitragssatz = sbeitrag_bsv/slohnsumme) %>%
    mutate(durchschnittseinkommen = slohnsumme/spersonen) %>%
    dplyr::select(c(jahr, alt, sex, nat, spersonen, slohnsumme,
        sbeitrag_bsv, beitraegerquote, beitragssatz, durchschnittseinkommen))

# --- IK Daten fuer Projektion
```

[34] Kleine Abweichungen in der Beitragssummenentwicklung zischen den Sozialversicherungen (bspw. AHV und IV) sind möglich, da die relativen Unterschiede in den Beitragssätzen für Selbständige und nichterwerbstätige nicht den relativen Unterschiede in den Beitragssätze für unselbständige Arbeitnehmer entsprechen.

```
# ------------------------------------------------

D_IK <- D_IK_YEARS %>%
    filter(jahr == PARAM_GLOBAL$jahr_ik) %>%
    ungroup() %>%
    dplyr::select(-jahr, -spersonen, -slohnsumme, -sbeitrag_bsv)
```

Es werden zuerst aus dem IK-Datensatz die Einträge für die IV ausgewählt (da `PARAM_GLOBAL$vz='iv'`). Danach wird die Summe der Personen `spersonen`, Löhne `slohnsumme`, und der Beiträge `sbeitrag_bsv` nach Jahr, Geschlecht, Nationalität (Schweizer oder Ausländer), und Alter gebildet. Mit Hilfe der (Erwerbs-)Bevölkerung, welche an das Dataframe angehängt wird, werden dann Quoten und Sätze gebildet. Zum Schluss wird das letzte Jahr, für welches die IK-Daten verfügbar sind, für die Projektionen ausgewählt und in das Dataframe `D_IK` gespeichert.

Wir nutzen diese Dataframes, um eine „statische Projektion“ der Beitragssumme zu berechnen. Die Projektion ist in dem Sinn statisch als dass sie einen konstanten Durchschnittslohn annimmt. Zuerst berechnen wir die „Projektion“ für die vergangenen Jahre:

```
# --- Statische Lösung
# ------------------------------------------------------

# Historisch

STATISCH_BS_TMP <- D_IK_YEARS %>%
    filter(jahr <= PARAM_GLOBAL$jahr_ik) %>%
    left_join(BEVOELKERUNG, by = c("jahr", "alt", "sex", "nat")) %>%
    # Statische Beitragssumme wird in IK geliefert
mutate(bsstatisch = bevendejahr * beitraegerquote * beitragssatz *
    durchschnittseinkommen) %>%
    # Statische Lohnsumme wird in IK geliefert
mutate(lsstatisch = bevendejahr * beitraegerquote * durchschnittseinkommen) %>%
    dplyr::select(alt, jahr, sex, nat, bsstatisch, lsstatisch,
        bevendejahr) %>%
    group_by(jahr) %>%
    filter(!is.na(bsstatisch)) %>%
    summarize(bsstatisch = sum(bsstatisch), lsstatisch = sum(lsstatisch),
        bevoelkerung = sum(bevendejahr))
```

Und danach die Projektion für die kommenden Jahre:

```
# Projektion

STATISCH_BS_PR <- BEVOELKERUNG %>%
    filter(jahr > PARAM_GLOBAL$jahr_ik) %>%
    # Die Projektion erfolgt mit den Kenngrössen aus dem IK
    # eines Jahres
left_join(D_IK, by = c("alt", "sex", "nat")) %>%
    mutate(bsstatisch = bevendejahr * beitraegerquote * beitragssatz *
        durchschnittseinkommen) %>%
    mutate(lsstatisch = bevendejahr * beitraegerquote * durchschnittseinkommen) %>%
    dplyr::select(alt, jahr, sex, nat, bsstatisch, lsstatisch,
        bevendejahr) %>%
    group_by(jahr) %>%
    filter(!is.na(bsstatisch)) %>%
    summarize(bsstatisch = sum(bsstatisch), lsstatisch = sum(lsstatisch),
```

```
        bevoelkerung = sum(bevendejahr))

# Historisch und Projektion

STATISCH_BS <- rbind(STATISCH_BS_TMP, STATISCH_BS_PR) %>%
    mutate(wr_bev = 100 * (bevoelkerung/lag(bevoelkerung) - 1)) %>%
    mutate(wr_bs = 100 * (bsstatisch/lag(bsstatisch) - 1))
```

Als nächsten werden die beiden Dataframes zusammengefügt, und die Wachstumnsraten der Bevölkerung der Beitragssumme berechnet:

```
# Historisch und Projektion

STATISCH_BS <- rbind(STATISCH_BS_TMP, STATISCH_BS_PR) %>%
    mutate(wr_bev = 100 * (bevoelkerung/lag(bevoelkerung) - 1)) %>%
    mutate(wr_bs = 100 * (bsstatisch/lag(bsstatisch) - 1))
```

Danach wird die „dynamische Projektion“ berechnet, welche sich von der „statischen Projektion“ dahingehend unterscheidet, dass sie die Entwicklung des Durchschnittslohnes berücksichtigt:

```
# --- Dynamische Loesung
# ------------------------------------------------------

DYNAMISCH_BS <- STATISCH_BS %>%
    left_join(EINK_ENTWICKLUNG, by = "jahr") %>%
    mutate(bsdynamisch = bsstatisch * eink_entwicklung) %>%
    mutate(lsdynamisch = lsstatisch * eink_entwicklung) %>%
    dplyr::select(jahr, bsdynamisch, lsdynamisch)
```

In einem nächsten Schritt werden die oben projizierten Beitragssumen auf die Abrechnung „justiert“:

```
# --- Dynamische Loesung
# ------------------------------------------------------

ABR_BEITRAGSSUMME <- ABRECHNUNG %>%
    dplyr::select(jahr, btr_vs_ag)

just_val <- justierung(PARAM_GLOBAL = PARAM_GLOBAL, ABRECHNUNG = ABR_BEITRAGSSUMME,
    MODELL = DYNAMISCH_BS)

DYNAMISCH_BS_ABR <- DYNAMISCH_BS %>%
    left_join(ABR_BEITRAGSSUMME, by = c("jahr")) %>%
    mutate(just_tmp = btr_vs_ag/bsdynamisch)

JUST_DYNAMISCH_BS <- DYNAMISCH_BS %>%
    dplyr::select(jahr) %>%
    left_join(DYNAMISCH_BS_ABR, by = c("jahr")) %>%
    mutate(justierung = if_else(jahr <= PARAM_GLOBAL$jahr_abr,
        just_tmp, just_val))
```

D.h., anhand der Höhe der Abweichung (im letzten Abrechnungsjahr) der Registerbasierten Projektion der Beitragssume, die im Dataframe `DYNAMISCH_BS` berechnet wurde, von der Beitragssumme gemäss IV-Abrechnung wird ein Faktor bestimmt, mit welchem alle Registerbasierten Projektionen angehoben werden. Diese Berechnung wird durch das Modul `justierung.R` duchgeführt (d.h. `justierung.R` berechnet `ABR_BEITRAGSSUMME$btr_vs_ag[ABR_BEITRAGSSUMME$jahr == PARAM_GLOBAL$jahr_abr] /`

`DYNAMISCH_BS$bsdynamisch[DYNAMISCH_BS$jahr == PARAM_GLOBAL$jahr_abr]`.

Für die Jahre vor dem aktuellsten Abrechnungsjahr wird der Justierungsfaktor als der Quotient `just_tmp=btr_vs_ag / bsdynamisch` berechnet, also genau so, dass `bsdynamisch` multipliziert mit dem Justierungsfaktor den tatsächlichen Wert gemäss der Abrechnung ergibt.

Danach wird das Dataframe `JUST_DYNAMISCH_BS` gebildet, welches für die Zukunft `just_val` als justierungsfaktor auswählt, und für die Veragangenheit `just_tmp`.

Nun wird das Dataframe `BEITRAGSSUMME_1` gebildet:

```
BEITRAGSSUMME_1 <- JUST_DYNAMISCH_BS %>%
    mutate(bsjustiert = bsdynamisch * justierung) %>%
    dplyr::select(jahr, btr_vs_ag = bsjustiert)
```

Dieses Dataframe enthält schon die Beitragssumme, welche dann auch für die Projektion der Beiträge von Versicherten und Arbeitgebern in den Finanzperspektiven verwendet wird.

Der folgende Code ist nur relevant wenn `PARAM_GLOBAL$vz == 'ahv'`, und wird daher hier nicht erläutert. Danach wird noch das Dataframe `BEITRAGSSUMME_1` in `BEITRAGSSUMME` geschrieben.

```
else {
      BEITRAGSSUMME <- BEITRAGSSUMME_1
    }
```

Der nächste Codeteil, der mit `PARAM_GLOBAL$vz == 'iv'` relevant ist, dient zur Projektion der Lohnsumme, und basiert auf der genau gleichen Logik wie der Code zur Projektion der Beiträge von Versicherten und Arbeitgebern:

```
# Lohnsumme IV
if (PARAM_GLOBAL$vz == "iv") {
    LOHNSUMME <- JUST_DYNAMISCH_BS %>%
        mutate(lsjustiert = lsdynamisch * justierung) %>%
        dplyr::select(jahr, ahv_lohnsumme = lsjustiert)
}
```

Der nachfolgende Codeteil dient lediglich zur Aufbereitung der Daten für die grafische Darstellung, und wird daher hier nicht weiter erläutert.

```
BEITRAGSSUMME_GRAFIK <- STATISCH_BS %>%
    left_join(JUST_DYNAMISCH_BS, by = "jahr") %>%
    rename(abrechnung = btr_vs_ag) %>%
    left_join(BEITRAGSSUMME, by = "jahr") %>%
    rename(bsjustiert = btr_vs_ag) %>%
    mutate(wr_bs_justiert = 100 * (bsjustiert/lag(bsjustiert) -
        1)) %>%
    left_join(LOHNSUMME, by = "jahr") %>%
    rename(lsjustiert = ahv_lohnsumme) %>%
    mutate(wr_ls_justiert = 100 * (lsjustiert/lag(lsjustiert) -
        1)) %>%
    left_join(DISKONTFAKTOR, by = "jahr") %>%
    mutate(bsjustiert_diskont = bsjustiert * diskontfaktor) %>%
    mutate(wr_bs_justiert_diskont = 100 * (bsjustiert_diskont/lag(bsjustiert_diskont) -
        1)) %>%
    mutate(lsjustiert_diskont = lsjustiert * diskontfaktor) %>%
    mutate(wr_ls_justiert_diskont = 100 * (lsjustiert_diskont/lag(lsjustiert_diskont) -
        1))
```

Zum schluss werden die berechneten Dataframes an das Modul `mod_iv_beitrag.R` zurückgegeben:

```
# --- Output
# ------------------------------------------------------------------

mod_return(BEITRAGSSUMME, LOHNSUMME, BEITRAGSSUMME_GRAFIK, D_IK_YEARS,
    D_IK)
```

4.18.2 Fortsetzung Modul mod_iv_beitrag.R

Die einzige verbleibende Berechnung im Modul `mod_iv_beitrag.R` ist, die in `mod_beitragssumme.R` berechnete Summe der Beiträge auf die einzelnen Positionen gemäss IV-Abrechnung aufzuteilen. Dies geschieht mit dem folgenden Code:

```
#------ Proportionale Aufteilung gemäss Abrechnungsjahr ---------------#
# Anteil im Abrechnungsjahr:
teil_cols <- c("btr_pers", "btr_lohn", "btr_lohn_ALV", "schad_ersatzf",
    "herabs_erl", "abschr_pers", "abschr_lohn", "nachz_abschr_lohn",
    "ruecks_btr_verlust", "verzugs_zins", "verguetungs_zins")

beitrag_abr_jahr <- as.numeric(IV_ABRECHNUNG[IV_ABRECHNUNG$jahr ==
    PARAM_GLOBAL$jahr_abr, "btr_vs_ag"])

BEITRAG_ANT <- IV_ABRECHNUNG %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
    select(match(teil_cols, colnames(IV_ABRECHNUNG))) %>%
    mutate(across(.cols = everything(), ~./beitrag_abr_jahr,
        .names = "{.col}"))

IV_LOHNSUMME <- tl_iv_beitrag$LOHNSUMME

#------ Abrechnung ----------------------------------------------------#
IV_BEITRAGSSUMME <- rbind(IV_ABRECHNUNG %>%
    filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
    select(c(jahr, "btr_vs_ag", teil_cols))) %>%
    #------ Prognose ------------------------------------------------------#
rbind(merge(tl_iv_beitrag$BEITRAGSSUMME %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr), BEITRAG_ANT) %>%
    mutate(across(teil_cols, ~. * btr_vs_ag, .names = "{.col}")))
```

D.h., es werden in `teil_cols` die Detailpositionen der IV-Abrechnung eingelesen, welche zu den Beiträgen von Versicherten und Arbeitgebern zählen. Danach wird in `beitrag_abr_jahr` das Total der Beiträge im Abrechnungsjahr gespeichert, und mit `BEITRAG_ANT` der Anteil der jeweiligen Positionen in `teil_cols` am Total der Beiträge im Abrechnungsjahr berechnet. Die Projektion für die verschiedenen Positionen wird dann gebildet, indem die projizierte totale Beitragssumme gemäss den Anteilen im Abrechnungsjahr auf die verschienden Positionen aufgeteilt wird. Dies geschieht ganz am Ende des Codes oben.

Somit können die Projektion für die Beiträge und die Lohnsumme an das Finanzperspektivenmodell (respektive das Modul `mod_iv_einnahmen.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
mod_return(IV_BEITRAGSSUMME, IV_LOHNSUMME)
```

4.19 Module mod_iv_uebrigeeinnahmen.R und mod_iv_regress.R

Dokumentation zuletzt aktualisiert am 20.09.2024

Das Prognosemodell zum Assistenzbeitrag ist im Skript `mod_iv_uebrigeeinnahmen.R` enthalten, welches wie folgt im Skript `mod_iv_einnahmen.R` aufgerufen wird:

```
tl_iv_uebrigeeinnahmen <- mod_iv_uebrigeeinnahmen(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, AUSGABEN_IV = AUSGABEN_IV)
```

Das Modul `mod_iv_uebrigeeinnahmen.R` verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Der Ausgabenvektor der Abrechnung wird am Ende des Moduls mit dem projizierten Ausgabenvektor zusammengefügt.
- `AUSGABEN_IV`: Für die im Untermodul `mod_iv_regress.R` projizierten Regresseinnahmen wird angenommen, dass diese eine Funktion der Ausgaben sind, weshalb hier der (projizierte) Ausgabenvektor eingelesen wird.

In `mod_iv_uebrigeeinnahmen.R` werden die folgenden zwei Untermodule aufgerufen:

```
#------ Regress -------------------------------------------------------#
tl_iv_regress <- mod_iv_regress(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, AUSGABEN_IV = AUSGABEN_IV)
#------ Andere Erträge ------------------------------------------------#
tl_iv_anderee <- mod_iv_andere_ertraege(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG)
```

`mod_iv_regress.R` beginnt wie folgt:

```
#------ Werte der Abrechnung ------------------------------------------#
IV_REGRESS_ABR <- IV_ABRECHNUNG %>%
    select(jahr, regr_ein_tot, regr_ein_zhpf, regr_ein_kost)

#------ Anteile im Abrechnungsjahr ------------------------------------#
IV_REA <- IV_REGRESS_ABR %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
    mutate(regr_ein_zhpf = regr_ein_zhpf/regr_ein_tot, regr_ein_kost =
    ↪  regr_ein_kost/regr_ein_tot) %>%
    select(regr_ein_zhpf, regr_ein_kost)
```

d.h. es werden die Werte aus dem Abrechnungsjahr ausgewählt, und der Anteil der zwei Regress-Unterpositionen im Abrechnungsjahr wird berechnet.

Danach werden die Regresseinnahmen mit der Wachstumsrate der Summe aus Rentenausgaben und Ausgaben für Hilflosenentschädigung fortgeschrieben, und am Ende werden die Regresseinnahmen widerum gemäss den Anteilen im Abrechnungsjahr auf die zwei Regress-Unterpositionen aufgeteilt:

```
#------ aktueller Abrechnungswert -------------------------------------#
last_abr_val <- data.frame(IV_REGRESS_ABR)[IV_REGRESS_ABR$jahr ==
    PARAM_GLOBAL$jahr_abr, c("regr_ein_tot")]

#------ Fortschreibung der Folgejahre der Entwicklung Renten / Hilo ---#
# falls eine Massname die Rentesumme/Hilo ändert, müsste
# dies neu berechnet werden.
IV_REGRESS_ENTW <- AUSGABEN_IV %>%
```

```
    select(jahr, rentensumme_iv, he_iv) %>%
    mutate(leist = rentensumme_iv + he_iv) %>%
    mutate(wr_leist = leist/lag(leist)) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    mutate(regr_ein_tot = cumprod(wr_leist) * last_abr_val) %>%
    select(jahr, regr_ein_tot) %>%
    merge(., IV_REA) %>%
    mutate(regr_ein_zhpf = regr_ein_zhpf * regr_ein_tot, regr_ein_kost = regr_ein_kost *
        regr_ein_tot)
```

Somit können die Ausgabenprojektionen mit den IV-Abrechnungsdaten zusammengeführt und an das Modul `mod_iv_uebrigeeinnahmen.R` zurückgegeben werden:

```
#------ Total Regress ---------------------------------------------------#
IV_REGRESS <- rbind(IV_REGRESS_ABR, IV_REGRESS_ENTW)

#------ Output ----------------------------------------------------------#
mod_return(IV_REGRESS)
```

Das Modul `mod_iv_andere_ertraege.R` wird hier nicht weiter erläutert, da es Stand 2024 lediglich einen Vektor von 0-Erträgen projiziert.

Als nächstes wird der letzte Wert der Umfrageprojektion zum Assistenzbeitrag ausgewählt, respektive eine Warnung angezeigt, falls dieser Wert nicht vorhanden ist:

```
#------ Werte der Umfrage und Fortschreibung mit der Ausgaben für Hilflosenentschädigung
↪  --------#
last_umfr_val <- as.numeric(UMFRAGE_IV[UMFRAGE_IV$jahr == (PARAM_GLOBAL$jahr_umfragen_fs
↪  -
    1), c("btr_ass")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für den Assistenzbeitrag im Jahr",
        (PARAM_GLOBAL$jahr_umfragen_fs - 1)))
}
```

Danach wird der Fortschreibungsvektor für die Jahre nach der letzten Umfrageprojektion (=Abrechnungsjahr+6) gebildet, und mit den Umfrageprojektionen zusammengefügt:

```
startwert <- as.numeric(IV_GELDLEISTUNG[IV_GELDLEISTUNG$jahr ==
    (PARAM_GLOBAL$jahr_umfragen_fs - 1), c("he_iv")])
IV_ASSB_PROJ <- IV_GELDLEISTUNG %>%
    select(jahr, he_iv) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, btr_ass), by = c("jahr")) %>%
    mutate(btr_ass = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        btr_ass, last_umfr_val * he_iv/startwert)) %>%
    select(-he_iv)
```

dh., es wird zuerst der Wert der Ausgaben für Hilflosenentschädigungen im Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` in der Variable `startwert` gespeichert. Dieser Startwert wird dann verwendet, um das Wachstum der Ausgaben für Hilflosenentschädigungen in einem Jahr relativ zum Jahr `PARAM_GLOBAL$jahr_umfragen_fs - 1` zu berechnen, was in der zweitletzten Zeile des obigen Codes geschieht.

Zum Schluss werden die totalen übrigen einnahmen gebildet, und an das Finanzperspektivenmodell (respektive das Modul `mod_iv_einnahmen.R`) zurückgegeben:

```
#------ Total alle übrigen Einnahmen ---------------------------------#
UEBRIGEEINNAHMEN_IV <- tl_iv_regress$IV_REGRESS %>%
    inner_join(tl_iv_anderee$IV_ANDERE_ERTRAEGE, by = c("jahr")) %>%
    mutate(ein_uebr_tot = ein_uebr)

#------ Output -------------------------------------------------------#
mod_return(UEBRIGEEINNAHMEN_IV)
```

4.20 Modul mod_iv_massnahmen_ext.R

Dokumentation zuletzt aktualisiert am 24.09.2024

`mod_iv_massnahmen_ext.R` wird im Skript `wrap_iv_massnahmen.R` wie folgt aufgerufen:

```
tl_iv_massnahmen_ext <- mod_iv_massnahmen_ext(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    PARAM_MASSNAHMEN = tl_inp$PARAM_MASSNAHMEN_BASE, IV_ABRECHNUNG =
    ↪  tl_inp$IV_ABRECHNUNG,
    MASSNAHMEN = tl_inp$MASSNAHMEN, MASSNAHMEN_DESC = tl_inp$MASSNAHMEN_DESC,
    MASSNAHMEN_SV = tl_inp$MASSNAHMEN_SV, MASSNAHMEN_SV_DESC = tl_inp$MASSNAHMEN_SV_DESC)
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die vergangenen Einnahmen und Ausgaben gemäss IV-Abrechnung.
- `PARAM_MASSNAHMEN = tl_inp$PARAM_MASSNAHMEN_BASE`: Liste der Massnahmen, deren Auswirkungen berechnet werden soll.
- `MASSNAHMEN`: Einnahme- respektive Ausgabenprojektion für Massnahmen, welche direkt für die IV spezifiziert wurden, deren Auswirkungen aber ausserhalb des IV-Finanzperspektivenmodells berechnet wurde.
- `MASSNAHMEN_DESC`: Beschrieb der Massnahmen im Dataframe `MASSNAHMEN`, insbesondere auch, auf welche Einnahme- oder Ausgabeposition die jeweilige Massnahme einen Einfluss hat.
- `MASSNAHMEN_SV`: Einnahme- respektive Ausgabenprojektion für Massnahmen, welche für andere Sozialversicherungen spezifiziert wurden, aber indirekt einen Einfluss auf die IV-Finanzen haben (bspw. Rentenaltererhöhung bei den Frauen im Zuge der AHV-21)
- `MASSNAHMEN_SV_DESC`: Beschrieb der Massnahmen im Dataframe `MASSNAHMEN_SV`, insbesondere auch, auf welche Einnahme- oder Ausgabeposition die jeweilige Massnahme einen Einfluss hat.

Als Erstes wird für die gemäss `PARAM_MASSNAHMEN` spezifizierten IV-internen Massnahmen überprüft, ob tatsächlich auch eine Ein- oder Ausgabenprojektion im Dataframe `MASSNAHMEN` vorliegt. Wenn dies der Fall ist wird die entsprechende Projektion ausgelesen und der entsprechenden Einnahme- oder Ausgabenposition zugewiesen:

```
#------ Massnahmen IV ------------------------------------------------#
if (length(iv_fld) > 0) {
    #------ Prüfen, ob die Variablen vorhanden sind ----------------------#
    nofld <- setdiff(iv_fld, colnames(MASSNAHMEN))
    if (length(nofld) > 0) {
        iv_fld <- intersect(iv_fld, colnames(MASSNAHMEN))
        warnings(paste(nofld, "nicht in MASSNAHMEN berechnet"))
    }

    EXT_MASSNAHMEN_IV <- MASSNAHMEN[, c("jahr", iv_fld)]

    #------ Werte den Konti zuordnen -------------------------------------#
    KTO <- data.frame(t(MASSNAHMEN_DESC[, iv_fld]))
    colnames(KTO) <- as.character(MASSNAHMEN_DESC$Variable)
```

```
    KTO$variable <- rownames(KTO)

    EXT_EINAUSGABEN_IV <- EXT_MASSNAHMEN_IV %>%
        pivot_longer(-jahr, names_to = "variable", values_to = "wert") %>%
        inner_join(KTO %>%
            select(variable, Konto), by = "variable")
}
```

Als nächstes wird der genau gleiche Schritt für die IV-Externen Massnahmen durchgeführt, das heisst, es wird für die gemäss `PARAM_MASSNAHMEN` spezifizierten IV-externe Massnahmen überprüft, ob tatsächlich auch eine Ein- oder Ausgabenprojektion im Dataframe `MASSNAHMEN_SV` vorliegt. Wenn dies der Fall ist wird die entsprechende Projektion ausgelesen und der entsprechenden Einnahme- oder Ausgabenposition zugewiesen. Danach werden die Ein- und Ausgabeprojektionen der IV-Externen Massnahmen mit denen von den IV-internen Massnahmen zusammengefügt:

```
#------ Massnahmen andere Sozialversicherungen ------------------------#
if (length(sv_fld) > 0) {
    #------ Prüfen, ob die Variablen vorhanden sind ----------------------#
    nofld <- setdiff(sv_fld, colnames(MASSNAHMEN_SV))
    if (length(nofld) > 0) {
        sv_fld <- intersect(sv_fld, colnames(MASSNAHMEN_SV))
        warnings(paste(nofld, "nicht in MASSNAHMEN_SV berechnet"))
    }

    EXT_MASSNAHMEN_SV <- MASSNAHMEN_SV[, c("jahr", sv_fld)]

    #------ Werte den Konti zuordnen -------------------------------------#
    KTO <- data.frame(t(MASSNAHMEN_SV_DESC[, sv_fld]))
    colnames(KTO) <- as.character(MASSNAHMEN_SV_DESC$Variable)
    KTO$variable <- rownames(KTO)

    EXT_EINAUSGABEN_SV <- EXT_MASSNAHMEN_SV %>%
        pivot_longer(-jahr, names_to = "variable", values_to = "wert") %>%
        inner_join(KTO %>%
            select(variable, Konto), by = "variable")

    #------ Zusammenfügen der IV/SV Daten --------------------------------#
    if (exists("EXT_MASSNAHMEN_IV")) {
        EXT_MASSNAHMEN <- EXT_MASSNAHMEN_IV %>%
            full_join(EXT_MASSNAHMEN_SV, by = c("jahr"))
        if (length(iv_fld) > 0) {
            EXT_EINAUSGABEN <- rbind(EXT_EINAUSGABEN_IV, EXT_EINAUSGABEN_SV)
        } else {
            EXT_EINAUSGABEN <- EXT_EINAUSGABEN_SV
        }
    } else {
        EXT_MASSNAHMEN <- EXT_MASSNAHMEN_SV
        EXT_EINAUSGABEN <- EXT_EINAUSGABEN_SV
    }
} else {
    EXT_MASSNAHMEN <- EXT_MASSNAHMEN_IV
    EXT_EINAUSGABEN <- EXT_EINAUSGABEN_IV
}
```

Zum Schluss werden die Liste der Externen Massnahmen (`EXT_MASSNAHMEN`) sowie deren Ausgabe- und Einnahmeprojektionen (`EXT_EINAUSGABEN`), und an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_massnahmen.R`) zurückgegeben:

```
#------ Output ------------------------------------------------------------#
mod_return(EXT_MASSNAHMEN, EXT_EINAUSGABEN)
```

4.21 Modul wrap_iv_massnahmen_int.R

Dokumentation zuletzt aktualisiert am 24.09.2024

`wrap_iv_massnahmen_int.R` wird im Skript `wrap_iv_massnahmen.R` wie folgt aufgerufen:

```
tl_iv_massnahmen_int <- wrap_iv_massnahmen_int(tl_inp = tl_inp,
    tl_iv_hauptberechnung = tl_iv_hauptberechnung)
```

Die Dataframes, welche in `wrap_iv_massnahmen_int.R` verwendet werden, sind abhängig von den spezifizierten Massnahmen. So nutzen beispielsweise Massnahmen, die Auswirkungen von Veränderungen bei der Rentenhöhe modellieren, die Projektionen aus dem Rentenmodell, oder Massnahmen, welche Auswirkungen von Veränderungen bei den medizinischen Massnahmen modellieren, Projektionen aus dem Modell für medizinische Massnahmen. Aus diesem Grund werden der Funktion alle Input-Daten sowie alle Resultate der Hauptberechnungen übergeben. Diese werde in einem ersten Schritt in einer gemeinsamen Tidy-Liste zusammengefasst:

```
# Concatenate lists of inputs to use in the measures
tl_inp_massn <- c(tl_inp, tl_iv_hauptberechnung)
```

Als nächstes werden die aktivierten Massnahmen in der Liste `opt_massn` ausgelesen, und es werden leere Vektoren erstellt, in welchen danach die Einnahme- und Ausgabenprojektionen für die Massnahmen abgelegt werden:

```
#------ Liste der aktivieren Massnahmen ---------------------------------#
if (is.na(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)) {
    opt_massn <- as.character()
} else {
    opt_massn <-
↪   separate_at_comma(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)
}

#------ Leere Frames erstellen / Aufruf optionaler Element hinzufügen -#
INT_MASSNAHMEN <- as_tibble(data.frame(jahr =
↪   c(tl_inp$PARAM_GLOBAL$jahr_beginn:tl_inp$PARAM_GLOBAL$jahr_ende)))

INT_EINAUSGABEN <- as_tibble(data.frame(jahr = as.numeric(),
    variable = as.character(), wert = as.numeric()))
```

Als nächstes werden die Auswirkungen der Massnahmen in `opt_massn` berechnet:

```
#------ Run list of measures ----------------------------------------------#
# Run opt_massn
. <- run_opt_vectorized(opt_massn, data = tl_inp_massn)
```

Hierfür wird über die Funktion `run_opt_vectorized` für jede in `opt_massn` spezifizierte Massnahme die Funktion `run_opt` aufgerufen. Diese Funktionen stellen sicher, dass die R-Skripte für die Berechnung der Auswirkungen der Massnahmen ausgeführt werden, und geben die Resultate der Massnahmenberechnungen zurück:

```
  # run_opt_vectorized
run_opt_vectorized <- function(massnahmen, data) {
  z0 <- lapply(massnahmen, run_opt, data = data) %>%
    unlist(recursive = FALSE) # Défaire la liste
  if (length(z0) > 0) {
    z <- tidy list_ensure(z0)
  } else {
    z <- z0
  }
  z
}
  # run_opt
run_opt <- function(opt.id, data, use.OPT = FALSE) {
  if (use.OPT) {
    data <- c(data, collect_objects("^OPT_"))
  }
  data <- tidy list_ensure(data)

  calling.env <- sys.frame(-1)

  fun.name <- paste0("mod_opt_", opt.id)

  if (!exists(fun.name)) {
    stop("optional module does not exist: ", fun.name)
  }

  argnames <- setdiff(names(formals(get(fun.name, envir = calling.env))), "list")

  dff <- setdiff(argnames, names(data))
  if (length(dff) > 0) {
    stop("data frames not found: ", paste(dff, collapse = ", "))
  }
  ll <- tidy list_ensure(data[argnames])

  z <- eval(bquote(.(as.name(fun.name))(list = ll)))

  xnames <- names(z)
  xnames <- paste0(xnames, "..", opt.id)

  Map(function(x, value) assign(x, value, envir = calling.env),
    x = xnames, value = z
  )

  names(z) <- xnames
  return(invisible(z))
}
```

Als nächstes werden die Auswirkungen der Massnahmen auf die einzelnen Einnahme- und Ausgabepositionen, in der Modellterminologie `Deltas` genannt, ausgelesen, und nach Massnahme aufgeteilt:

```
  # Dataframes der Deltas
  tl_delta <- c(
    tl_delta_stufe_opt_massn
  )
```

```
  # Dataframes optionnels
  tl_opt <- c(
    tl_opt_stufe_opt_massn
  )

  # Tous les deltas
  if(length(tl_delta)>0){
  ALLE <-
    tl_delta %>%
    bind_rows(.id = "type") %>%
    mutate(type = tolower(type)) %>%
    mutate(wert = if_else(is.na(wert), 0, wert)) %>%
    separate(type, c("delta_type", "massnahme"), sep = "\\.\\.")
```

Als nächstes werden die Auswirkungen jeder Massnahmen nach Einnahme- und Ausgabenposition aggregiert:

```
# Sommes des deltas par mesure 1) Dépenses
AUSG_MASSNAHME <- ALLE %>%
    filter(grepl("^delta_ausg", delta_type)) %>%
    group_by(jahr, massnahme) %>%
    summarize(wert = sum(wert, na.rm = TRUE)) %>%
    ungroup()

# 2) Recettes
EINN_MASSNAHME <- ALLE %>%
    filter(grepl("^delta_einn", delta_type)) %>%
    group_by(jahr, massnahme) %>%
    summarize(wert = sum(wert, na.rm = TRUE)) %>%
    ungroup()

# 3) Recette et dépenses
EINN_AUSG_MASSNAHME <- rename(AUSG_MASSNAHME, ausgaben = wert) %>%
    left_join(rename(EINN_MASSNAHME, einnahmen = wert), by = c("jahr",
        "massnahme")) %>%
    mutate(jahr = as.numeric(jahr))
```

Die folgende Funktion dient dazu, die Massnahmenauswirkungen so darzustellen, dass jede Projektionsjahr eine Zeile ist und die Spalten die Auswirkungen der jeweiligen Massnahme anzeigen. Es handelt sich also in eine Umformung vom langen in das breite Datenformat, wobei gleichzeitig die Titel so gesetzt werden, dass klar ist, ob es sich um eine Mehrausgabe oder eine Mehreinnahme handelt, und Projektionsjahre mit 0-Werten entfernt werden:

```
# Fonction pour renommer les colonnes par 'ausgaben' et
# 'einnahmen' et pour ne garder que les non-nulles
tab_pub <- function(massnahmen, keep.cols = character()) {
    z <- tibble(jahr = tl_inp$PARAM_GLOBAL$jahr_abr:tl_inp$PARAM_GLOBAL$jahr_ende)
    for (.massnahme in massnahmen) {
        TAB_MASS_i <- EINN_AUSG_MASSNAHME %>%
            filter(massnahme == .massnahme) %>%
            rename_at(c("ausgaben", "einnahmen"), list(~paste0(.,
                "..", .massnahme))) %>%
            dplyr::select(-massnahme)
```

```
        z <- z %>%
            left_join(TAB_MASS_i, by = "jahr") %>%
            mutate_all(list(~if_else(is.na(.), 0, .)))
    }
    # spalten mit ausschliesslich 0 entfernen
    nicht.null <- abs(colSums(z, na.rm = TRUE)) > 1e-05
    nicht.null[names(nicht.null) %in% keep.cols] <- TRUE
    z[, nicht.null]
}

# Tableau par genre de mesure
TAB_OPT_MASSN <- tab_pub(opt_massn)
```

Als nächstes wird die Tabelle `EINN_AUSG_DELTA_TYPE`, welche die Auswirkungen aller Massnahmen nach Einnahme- und Ausgabenposition aggregiert (d.h. wenn mehrere Massnahmen Auswirkungen auf dieselbe Einnahme- oder Ausgabenposition haben, werden die Auswirkungen dieser Massnahmen aufsummiert):

```
#-----------Construction du tableau INT_EINAUSGABEN-------------------
# Dépenses
AUSG_DELTA_TYPE <- ALLE %>%
    filter(grepl("^delta_ausg", delta_type)) %>%
    group_by(jahr, delta_type) %>%
    summarize(wert = sum(wert, na.rm = TRUE)) %>%
    ungroup() %>%
    transmute(jahr, Konto = gsub("delta_ausg_", "", delta_type),
        wert)

# Recettes
EINN_DELTA_TYPE <- ALLE %>%
    filter(grepl("^delta_einn", delta_type)) %>%
    group_by(jahr, delta_type) %>%
    summarize(wert = sum(wert, na.rm = TRUE)) %>%
    ungroup() %>%
    transmute(jahr, Konto = gsub("delta_einn_", "", delta_type),
        wert)

# Dépenses et recettes
EINN_AUSG_DELTA_TYPE <- AUSG_DELTA_TYPE %>%
    full_join(EINN_DELTA_TYPE, by = c("jahr", "Konto", "wert"))

# KTO <- tl_inp$MASSNAHMEN_DESC %>% filter(Variable ==
# 'Konto') %>% pivot_longer(cols = everything(), names_to =
# 'Variable', values_to = 'Konto')


INT_EINAUSGABEN <- INT_EINAUSGABEN %>%
    full_join(EINN_AUSG_DELTA_TYPE, by = c("jahr", "wert"))
```

Zum Schluss werden die Listen bestimmt, die an das Finanzperspektivenmodell, respektive das Modul `wrap_iv_massnahmen` zurückgegeben werden. Der `else` betrifft den Fall, dass `if(length(tl_delta)>0)` weiter oben `FALSE` ist, d.h., dass die berechneten Massnahmen keine Auswirkungen auf die Einnahmen oder Ausgaben haben. In diesem Fall wird nur die Optionale Liste `tl_opt`, welche beliebige nicht-finanzielle Auswirkungen der Massnahmen enthält, beispielsweise wie viele Rentenbeziehende von einer Massnahme betroffen sind:

```
  #------ Output -------------------------------------------------------#
  c(
    tidy list(
      INT_MASSNAHMEN,
      INT_EINAUSGABEN
    ),
    tl_opt,
    tl_delta
  )
  }else{
    c(tl_opt)
  }
```

4.22 Modul mod_iv_einausgaben.R

Dokumentation zuletzt aktualisiert am 24.09.2024

`mod_iv_einausgaben.R` wird im Skript `wrap_iv_ergebnisse.R` wie folgt aufgerufen:

```
tl_ein_aus <- mod_iv_einausgaben(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, AUSGABEN_IV =
    ↪  tl_iv_hauptberechnung$AUSGABEN_IV,
    EINNAHMEN_IV = tl_iv_hauptberechnung$EINNAHMEN_IV, IV_EINAUSGABEN =
    ↪  tl_iv_mass$IV_EINAUSGABEN)
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die vergangenen Einnahmen und Ausgaben gemäss IV-Abrechnung.
- `AUSGABEN_IV`: Sämtliche Ausgabeprojektionen, mit Ausnahme des Zinsaufwandes (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.1.6) und OHNE die Ausgaben auf Grund von Politikmassnahmen.
- `EINNAHMEN_IV`: Sämtliche Einnahmeprojektionen, mit Ausnahme des Zinsertrages (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.1.10) und OHNE die Einnahmen auf Grund von Politikmassnahmen.
- `IV_EINAUSGABEN`: Die Einnahme- und Ausgabeauswirkungen der verschiedenen Politikmassnahmen, die berechnet wurden (vgl. Kapitel 4.1.11).

Als Erstes prüft das Modul, ob in den Politikmassnahmen, also im Dataframe `IV_EINAUSGABEN`, eine MWST-Zusatzfinanzierung (`btr_mwst`) oder eine Zusatzfinanzierung durch den Bund (`extern_bund`) spezifiziert ist, und falls dies der Fall ist, wird eine zusätzliche Einnahme-Spalte hinzugefügt:

```
## si TVA pour financer dette AI ajouter col pour FHH
if ("btr_mwst" %in% colnames(IV_EINAUSGABEN)) {
    EINNAHMEN_IV <- EINNAHMEN_IV %>%
        mutate(btr_mwst = 0)
}  ######

## si financement conf extern pour financer dette AI
## ajouter col pour FHH
if ("extern_bund" %in% colnames(IV_EINAUSGABEN)) {
    EINNAHMEN_IV <- EINNAHMEN_IV %>%
        mutate(extern_bund = 0)
}  ######
```

Als nächstes werden die Einnahme- und Ausgabepositionen ausgelesen, welche von Politikmassnahmen betroffen sind, und es wird überprüft, ob die Einnahme- und Ausgabepositionen der Politikmassnahmen in den

Einnahme- und Ausgabepositionen nach geltender Ordnung enthalten sind:

```
fld_einnahmen <- intersect(colnames(EINNAHMEN_IV), colnames(IV_EINAUSGABEN))
fld_ausgaben <- intersect(colnames(AUSGABEN_IV), colnames(IV_EINAUSGABEN))
fld_null <- setdiff(colnames(IV_EINAUSGABEN), c(fld_einnahmen,
    fld_ausgaben))
if (length(fld_null) > 0) {
    warnings(paste(fld_null, " : nicht definierte Konti"))
}
```

Als nächstes werden für die Ausgabepositionen, für welche Politikmassnahmen spezifiziert sind (also die Ausgabepositionen in `fld_ausgaben`), die Ausgaben für die Politikmassnahmen zu den Ausgaben in `AUSGABEN_IV` dazugezählt:

```
#------ summieren der Ausgaben ----------------------------------------#
if (length(fld_ausgaben) > 1) {
    AUSGABEN <- AUSGABEN_IV %>%
        pivot_longer(-jahr, names_to = "variable", values_to = "wert") %>%
        rbind(IV_EINAUSGABEN[, c(fld_ausgaben)] %>%
            pivot_longer(-jahr, names_to = "variable", values_to = "wert")) %>%
        group_by(jahr, variable) %>%
        summarize(wert = sum(wert)) %>%
        pivot_wider(names_from = variable, values_from = wert) %>%
        ungroup()

    AUSGABEN <- AUSGABEN[, colnames(AUSGABEN_IV)]

    #------ Anpassung bei den Renten --------------------------------------#
    # TODO Anpassung den Durchführungskosten (df_kost)
    if (!is.na(match("rentensumme_iv", fld_ausgaben))) {

        #------ Proportionale Aufteilung im Abrechnungsjahr -----------------#
        teil_cols <- c("rent_ord", "rent_ord_nachz", "rent_aord",
            "rent_aord_nachz", "ausl_rueck", "ausl_fs_leist",
            "rueck_erstattungsf", "abschr_rueck_erstattungsf")

        renten_start <- AUSGABEN %>%
            filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
            select(rentensumme_iv) %>%
            as.numeric()

        RENTEN_ANT <- AUSGABEN %>%
            filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
            select(match(teil_cols, colnames(AUSGABEN))) %>%
            mutate(across(where(is.numeric), ~./renten_start,
                .names = "{.col}"))

        #------ Anpassung der Werte im Prognosezeitraum -----------------------#
        AUSGABEN_TMP <- AUSGABEN %>%
            filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
            rbind(AUSGABEN %>%
                filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
                select(-match(teil_cols, colnames(AUSGABEN))) %>%
                merge(., RENTEN_ANT) %>%
                mutate(across(teil_cols, ~. * rentensumme_iv,
```

```
                .names = "{.col}")))

        AUSGABEN <- AUSGABEN_TMP %>%
            mutate(geld_leist_iv = rentensumme_iv + he_iv + tg_geld +
                btr_ant_iv, ind_mass_tot_iv = mm + fi + im +
                ber_begl + mba + aus_uebr + hm + reise_kost +
                btr_ass + rueck_erstattungsf_ind, btr_inst_org_iv = btr_org +
                btr_proinf, verw_aufw_iv = post_taxen + verw_kost +
                abschr_iv_ste + iv_ste_rad + kost_rueck + kost_fonds_ant) %>%
            mutate(uebrige_ausgaben_iv = btr_inst_org_iv + verw_aufw_iv +
                a_aufw, aus_tot_iv = geld_leist_iv + ind_mass_tot_iv +
                btr_inst_org_iv + df_kost_iv + verw_aufw_iv +
                a_aufw)
    }
} else {
    AUSGABEN <- AUSGABEN_IV
}
```

Die zweite `if`-Bedingung deckt den Fall ab, dass Massnahmen, welche die Rentensumme betreffen, vorhanden sind (`!is.na(match('rentensumme_iv', fld_ausgaben))`). In diesem Fall werden diese Ausgaben, analog zu den Berechnungen im Rentenmodell (vgl. Kapitel 4.2.5), auf die einzelnen Rentenausgabenpositionen aufgeteilt.

Als nächstes werden genau die gleichen Berechnungen für die Einnahmepositionen durchgeführt. D.h., es werden für Einnahmepositionen, für welche Politikmassnahmen spezifiziert sind (also die Einnahmepositionen in `fld_einnahmen`), die Einnahmen aus Politikmassnahmen zu den Einnahme in `EINNAHMEN_IV` dazugezählt

```
#------ summieren der Einnahmen----------------------------------------#
if (length(fld_einnahmen) > 1) {
    EINNAHMEN <- EINNAHMEN_IV %>%
        pivot_longer(-jahr, names_to = "variable", values_to = "wert") %>%
        rbind(IV_EINAUSGABEN[, c(fld_einnahmen)] %>%
            pivot_longer(-jahr, names_to = "variable", values_to = "wert")) %>%
        group_by(jahr, variable) %>%
        summarize(wert = sum(wert)) %>%
        pivot_wider(names_from = variable, values_from = wert) %>%
        ungroup()

    EINNAHMEN <- EINNAHMEN[, colnames(EINNAHMEN_IV)]

    #------ Anpassung bei Beiträgen ----------------------------------------#
    # TODO Anpassung bei Regress (regr_ein_tot)
    if (!is.na(match("btr_vs_ag", fld_einnahmen))) {

        #------ Proportionale Aufteilung gemäss Abrechnungsjahr ---------------#
        ## si TVA ou contribution extern pour financer
        ## dette AI ajouter col pour FHH
        ## if(!is.null(IV_EINAUSGABEN$btr_mwst) &
        ## !is.null(IV_EINAUSGABEN$extern_bund))
        if ("btr_mwst" %in% colnames(IV_EINAUSGABEN)) {
            teil_cols <- c("btr_pers", "btr_lohn", "btr_lohn_ALV",
                "schad_ersatzf", "herabs_erl", "abschr_pers",
                "abschr_lohn", "nachz_abschr_lohn", "ruecks_btr_verlust",
                "verzugs_zins", "verguetungs_zins")
```

```
} else {
    teil_cols <- c("btr_pers", "btr_lohn", "btr_lohn_ALV",
        "schad_ersatzf", "herabs_erl", "abschr_pers",
        "abschr_lohn", "nachz_abschr_lohn", "ruecks_btr_verlust",
        "verzugs_zins", "verguetungs_zins")
}  ######

beitrag_abr_jahr <- as.numeric(EINNAHMEN[EINNAHMEN$jahr ==
    PARAM_GLOBAL$jahr_abr, "btr_vs_ag"])

BEITRAG_ANT <- EINNAHMEN %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr) %>%
    select(match(teil_cols, colnames(EINNAHMEN))) %>%
    mutate(across(where(is.numeric), ~./beitrag_abr_jahr,
        .names = "{.col}"))

#------ Anpassung der Werte im Prognosezeitraum ----------------------#
EINNAHMEN_TMP <- EINNAHMEN %>%
    filter(jahr <= PARAM_GLOBAL$jahr_abr) %>%
    rbind(EINNAHMEN %>%
        filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
        select(-match(teil_cols, colnames(EINNAHMEN))) %>%
        merge(., BEITRAG_ANT) %>%
        mutate(across(teil_cols, ~. * btr_vs_ag, .names = "{.col}")))

## si TVA pour financer dette AI ajouter col pour
## FHH
if ("btr_mwst" %in% colnames(IV_EINAUSGABEN) | "extern_bund" %in%
    colnames(IV_EINAUSGABEN)) {
    if ("btr_mwst" %in% colnames(IV_EINAUSGABEN) & "extern_bund" %in%
        colnames(IV_EINAUSGABEN)) {
        EINNAHMEN <- EINNAHMEN_TMP %>%
          mutate(btr_oeffh = btr_bund + btr_mwst + btr_zinsiv,
            ein_tot_iv = btr_vs_ag + btr_bund + btr_mwst +
              btr_zinsiv + regr_ein_tot + ein_uebr_tot +
              btr_mwst + extern_bund)
    }
    if ("btr_mwst" %in% colnames(IV_EINAUSGABEN)) {
        EINNAHMEN <- EINNAHMEN_TMP %>%
          mutate(btr_oeffh = btr_bund + btr_mwst + btr_zinsiv,
            ein_tot_iv = btr_vs_ag + btr_bund + btr_mwst +
              btr_zinsiv + regr_ein_tot + ein_uebr_tot +
              btr_mwst)
    }
    if ("extern_bund" %in% colnames(IV_EINAUSGABEN)) {
        EINNAHMEN <- EINNAHMEN_TMP %>%
          mutate(btr_oeffh = btr_bund + btr_mwst + btr_zinsiv,
            ein_tot_iv = btr_vs_ag + btr_bund + btr_mwst +
              btr_zinsiv + regr_ein_tot + ein_uebr_tot +
              extern_bund)
    }
} else {
```

```
            EINNAHMEN <- EINNAHMEN_TMP %>%
                mutate(btr_oeffh = btr_bund + btr_mwst + btr_zinsiv,
                  ein_tot_iv = btr_vs_ag + btr_bund + btr_mwst +
                    btr_zinsiv + regr_ein_tot + ein_uebr_tot)
        }  ######

    }
} else {
    EINNAHMEN <- EINNAHMEN_IV
}
```

Zum Schluss werden die Ausgaben und Einnahmen an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_ergebnisse.R`) zurückgegeben:

```
#------ Output --------------------------------------------------------#
mod_return(AUSGABEN, EINNAHMEN)
```

4.23 Modul mod_bilanz_iv_rekursiv.R

Dokumentation zuletzt aktualisiert am 07.10.2024

`mod_bilanz_iv_rekursiv.R` wird im Skript `wrap_iv_ergebnisse.R` wie folgt aufgerufen:

```
tl_bilanz_iv <- mod_bilanz_iv_rekursiv(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG, AUSGABEN_IV = tl_ein_aus$AUSGABEN,
    EINNAHMEN_IV = tl_ein_aus$EINNAHMEN, ZINS = tl_inp$ZINS,
    IV_LOHNSUMME = tl_iv_hauptberechnung$IV_LOHNSUMME, ECKWERTE_EXTENDED =
    ↪  tl_inp$ECKWERTE_EXTENDED,
    FORTSCHREIBUNG_IV)
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die vergangenen Einnahmen und Ausgaben gemäss IV-Abrechnung.
- `tl_ein_aus$AUSGABEN`: Sämtliche Ausgabeprojektionen, mit Ausnahme des Zinsaufwandes (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.1.6), mit den Ausgaben auf Grund von Politikmassnahmen.
- `tl_ein_aus$EINNAHMEN`: Sämtliche Einnahmeprojektionen, mit Ausnahme des Zinsertrages (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.1.10), mit den Einnahmen auf Grund von Politikmassnahmen
- `ZINS`: Die für die Schulden und das Anlagekapital angenommenen Zinssätze.
- `IV_LOHNSUMME` und `ECKWERTE_EXTENDED`: Diese Dataframes werden zwar an das Modul übergeben, aber in der momentanen Programmierung in diesem nicht verwendet.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der Wachstumsrate des Bundesbeitrages (Variable `bundesbeitrag`).

In einem ersten Schritt werden die Einnahmen- und Ausgabenprojektionen mit der IV-Abrechnung zusammengefügt:

```
#------ Bereitstellung der Basisdaten ---------------------------------#
BILANZ_IV <- data.frame(jahr = c(PARAM_GLOBAL$jahr_beginn:PARAM_GLOBAL$jahr_ende)) %>%
    left_join(AUSGABEN_IV, by = "jahr") %>%
    left_join(EINNAHMEN_IV, by = "jahr") %>%
    left_join(IV_ABRECHNUNG %>%
        select(jahr, zins, schzins, kap, fonds, fl_mtl, anl_erg,
            erg_betr) %>%
        mutate(schzins = schzins + zins) %>%
```

```
        select(-zins), by = "jahr") %>%
    # Rechnungslegung 2011 - 2025 (Zins im Umlageergebnis)
mutate(aus_tot_iv = aus_tot_iv + schzins) %>%
    left_join(ZINS, by = "jahr") %>%
    mutate(erg_umlag_iv = ein_tot_iv - aus_tot_iv, zinstraeger = 0,
        rueckzahlung = 0) %>%
    mutate(across(c(-jahr), as.numeric, .names = "{.col}")) %>%
    mutate(across(where(is.numeric), list(~if_else(is.na(.),
        0, .)), .names = "{.col}"))
```

Hierzu wird bei der IV-Abrechnung die Summe aus den Spalten `schzins` und `zins` gebildet. Grund ist dass die Zinsausgaben bis 2010 unter dem Posten `zins` verbucht wurden, und ab 2011 unter dem Posten `schzins`, jedoch beide Positionen gleichsam den gesamten Zinsaufwand im jeweiligen Jahr umfassen. Zusätzlich wird das Dataframe `ZINS` angefügt, welches für jedes Projektionsjahr den im jeweiligen Jahr gültigen Zinssatz für die Schulden der IV bei der AHV umfasst.

Als nächstes folgt eine umfangreiche `for`-Schleife, welche zum Ziel hat, anhand der jeweiligen Erträge und Aufwände in einem Jahr durch rekursive Fortschreibung das Umlageergebnis, den Zinsaufwand und -ertrag, sowie den Stand des IV-Kapitals (Fondsstand) zu berechnen. Zur Erläuterung wird die Schleife nachfolgend Schritt-für-Schritt beschrieben:

```
#------ loop ------------------------------------------------------------#
  startjahr     <- which(BILANZ_IV$jahr == PARAM_GLOBAL$jahr_abr) + 1

   for(l_j in startjahr:NROW(BILANZ_IV)) {
#------ Zinsen zu Gunst. AHV / Int. en faveur de l'AVS -----------------#
    BILANZ_IV[l_j, 'schzins'] <- - BILANZ_IV[l_j - 1, 'kap'] * BILANZ_IV[l_j, 'iv_zins']
#------ Ausgaben neu berechnen, da Schuldzins 2011-2025 enthalten ist -#
# (gilt für die Periode ab Einführung eigener IV-Fonds bis zur Koorektur
# gemäss der überarbeiteten Rechnungslegung)
    BILANZ_IV[l_j, 'aus_tot_iv']  <-
       BILANZ_IV[l_j, 'geld_leist_iv'] +
       BILANZ_IV[l_j, 'ind_mass_tot_iv'] +
       BILANZ_IV[l_j, 'btr_inst_org_iv'] +
       BILANZ_IV[l_j, 'df_kost_iv'] +
       BILANZ_IV[l_j, 'verw_aufw_iv'] +
       BILANZ_IV[l_j, 'a_aufw'] +
       BILANZ_IV[l_j, 'schzins']

    # btr_zinsiv ist eine nicht mehr verwendete Position der IV-Bilanz, die seit 2018 =0
    ↪  ist.
    BILANZ_IV[l_j, 'btr_zinsiv'] <- 0
```

In einem ersten Schritt wird die Zeilennummer mit dem Startjahr für die Fortschreibung, welches dem Ersten Jahr ab der aktuellsten Abrechnung (Stand 2024 ist dies 2023+1=2024) entspricht, in die Variable `startjahr` ausgelesen. Danach wird die `for`-Schleife gestartet, welche als Erstes basierend auf dem Stand der Schulden der IV bei der AHV (Spalte `kap`) des Vorjahres die Höhe der Zinszahlung der IV an die AHV bestimmt. Danach werden die Totalausgaben unter Einbezug der Schuldzinszahlungen berechnet.

Als nächstes wird die Höhe des Bundesbeitrages berechnet:[35]

```
# Bundesbeitrag berechnen (unter Berücksichtigung der
# Untergrenze in PARAM_GLOBAL$bundesanteil_iv (37.7% in
# 2024)) Berechnung des Bundesbeitrags mit den angegebenen
```

[35] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.

```
# Bedingungen
berechneter_bund <- BILANZ_IV[l_j - 1, "btr_bund"] * (1 +
↪  FORTSCHREIBUNG_IV$bundesbeitrag[FORTSCHREIBUNG_IV$jahr ==
    BILANZ_IV[l_j, "jahr"]]/100)

# Untergrenze des Bundesanteils
untergrenze_bund <- PARAM_GLOBAL$bundesanteil_iv * BILANZ_IV[l_j,
    "aus_tot_iv"]

# Obergrenze 0.5 des Ausgaben
obergrenze_bund <- 0.5 * BILANZ_IV[l_j, "aus_tot_iv"]

# Überprüfe, ob berechneter_bund innerhalb der Grenzen
# liegt
if (berechneter_bund < untergrenze_bund) {
    BILANZ_IV[l_j, "btr_bund"] <- untergrenze_bund
} else if (berechneter_bund > obergrenze_bund) {
    BILANZ_IV[l_j, "btr_bund"] <- obergrenze_bund
} else {
    BILANZ_IV[l_j, "btr_bund"] <- berechneter_bund
}
```

Hierzu wird zuerst die Variable `berechneter_bund` gebildet, welche dem Bundesbeitrag des letzten Jahres multipliziert mit der in `FORTSCHREIBUNG_IV` (vgl. Kapitel 4.33) berechneten Wachstumsrate des Bundesbeitrages entspricht.

Danach wird überprüft, ob `berechneter_bund` innerhalb der gesetztlichen Grenze von 37.7% und 50% der Ausgaben liegt, und falls dies der Fall ist wird `berechneter_bund` als Bundesbeitragseinnahme für das entsprechende Jahr verwendet, und ansonsten der Wert von 37.7% respektive 50% der Ausgabe, falls die jeweilige Unter- respektive Obergrenze von `berechneter_bund` unter- respektive überschritten wird.

In einem nächsten Schritt wird für den Fall, dass in den Politikmassnahmen eine MWST-Zusatzfinanzierung oder eine Bundesbeitrags-Zusatzfinanzierung für die IV spezifiziert ist, der Wert dieser Zusatzfinanzierung zu den Einnahmen hinzugezählt. Stand 2024 ist keine dieser Zusatzfinanzierungen spezifiziert, sodass beide `if`-Bedingungne `FALSE` sind:

```
if ("btr_mwst" %in% colnames(EINNAHMEN_IV)) {
    BILANZ_IV[l_j, "ein_tot_iv"] <- BILANZ_IV[l_j, "btr_vs_ag"] +
        BILANZ_IV[l_j, "btr_bund"] + BILANZ_IV[l_j, "btr_zinsiv"] +
        BILANZ_IV[l_j, "btr_mwst"] + BILANZ_IV[l_j, "regr_ein_tot"]
}

if ("extern_bund" %in% colnames(EINNAHMEN_IV)) {
    BILANZ_IV[l_j, "ein_tot_iv"] <- BILANZ_IV[l_j, "btr_vs_ag"] +
        BILANZ_IV[l_j, "btr_bund"] + BILANZ_IV[l_j, "btr_zinsiv"] +
        BILANZ_IV[l_j, "btr_mwst"] + BILANZ_IV[l_j, "regr_ein_tot"] +
        BILANZ_IV[l_j, "extern_bund"]
}
```

Somit kann für das laufende Jahr das Umlageergebnis berechnet werden:

```
#------ neu berechnetes Umlageergebnis --------------------------------#
BILANZ_IV[l_j, "erg_umlag_iv"] <- BILANZ_IV[l_j, "ein_tot_iv"] -
    BILANZ_IV[l_j, "aus_tot_iv"]
```

Als nächstes wird das Anlagergebnis berechnet, also die in einem Projektionsjahr erwartete Anlagerendite, sowie das resultierende Betriebsergebnis:

```
#------ Zinsträger-------------------------------------------------------#
BILANZ_IV[l_j, "zinstraeger"] <- BILANZ_IV[l_j - 1, "fonds"] -
    PARAM_GLOBAL$fl_mittel_iv * BILANZ_IV[l_j - 1, "aus_tot_iv"] +
    0.5 * (BILANZ_IV[l_j, "btr_vs_ag"] + BILANZ_IV[l_j, "regr_ein_tot"] +
        BILANZ_IV[l_j, "btr_mwst"] - (1 - PARAM_GLOBAL$bundesanteil_iv) *
        BILANZ_IV[l_j, "aus_tot_iv"]) - PARAM_GLOBAL$fl_anteil_iv/12 *
    (BILANZ_IV[l_j, "rentensumme_iv"] + BILANZ_IV[l_j, "he_iv"])

#------ Zinsergebnis und BR ----------------------------------------------#
# Annahme Zins ist Zinsintenistät.
BILANZ_IV[l_j, "anl_erg"] <- BILANZ_IV[l_j, "zinstraeger"] *
    BILANZ_IV[l_j, "ahv_zins"]/(1 - BILANZ_IV[l_j, "ahv_zins"]/2)
BILANZ_IV[l_j, "erg_betr"] <- BILANZ_IV[l_j, "erg_umlag_iv"] +
    BILANZ_IV[l_j, "anl_erg"]
```

Wir nehmen an, dass der Zinsträger, also der Anteil des IV-Fonds (des Vermögens der IV ohne die Schulden bei der AHV), welcher verzinst wird, dem Fondsstand der Vorperiode abzüglich einem Anteil der laufenden Ausgaben zuzüglich einem Anteil der laufenden Einnahmen gemäss obiger Formel entspricht. Dieser Zinsträger wird dann mit dem Faktor `BILANZ_IV[l_j, 'ahv_zins']/(1 - BILANZ_IV[l_j, 'ahv_zins']/2)` verzinst, was das Anlageergebnis ergibt. Damit wird dann das Betriebsergebnis ermittelt.

In einem letzten Schritt wird nun der Stand des IV-Fonds, sowie der Schulden der IV bei der AHV, ermittelt:

```
#------ Fondstand --------------------------------------------------------#
#------ Schuld noch nicht zurückbezahlt: Regel für Rückzahlung: -------#
    if(BILANZ_IV[l_j - 1, 'kap'] < 0) {
      if(PARAM_GLOBAL$jahr_abr <= 2018) {
                kapital_soll <- 5E9
      } else {  kapital_soll <- (0.5 + PARAM_GLOBAL$fl_mittel_iv) * BILANZ_IV[l_j,
↪  'aus_tot_iv'] }
                fuerfonds    <- kapital_soll - BILANZ_IV[l_j - 1, 'fonds']
      BILANZ_IV[l_j, 'rueckzahlung'] <- BILANZ_IV[l_j, 'erg_betr'] - fuerfonds

      if("erg_rueckzahlung" %in% colnames(BILANZ_IV)){
        BILANZ_IV[l_j, 'rueckzahlung'] <- BILANZ_IV[l_j, 'erg_rueckzahlung']
      }

      if("extern_rueckzahlung" %in% colnames(BILANZ_IV)){
        BILANZ_IV[l_j, 'rueckzahlung'] <- BILANZ_IV[l_j, 'extern_rueckzahlung']
      }

#------ Keine Rückzahlung ------------------------------------------------#
      if((BILANZ_IV[l_j, 'rueckzahlung'] < 0) | (BILANZ_IV[l_j, 'erg_betr'] < 0) ) {
        BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j - 1, 'fonds'] + BILANZ_IV[l_j,
↪  'erg_betr']
        BILANZ_IV[l_j, 'kap']   <- BILANZ_IV[l_j - 1, 'kap']
        BILANZ_IV[l_j, 'rueckzahlung'] <- 0
#------ Amortisation -----------------------------------------------------#
      } else {
        BILANZ_IV[l_j, 'kap']   <- BILANZ_IV[l_j - 1, 'kap'] + BILANZ_IV[l_j,
↪  'rueckzahlung']
        if("erg_rueckzahlung" %in% colnames(BILANZ_IV)){
```

```
          if(BILANZ_IV[l_j, 'erg_rueckzahlung']==0){
            BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j - 1, 'fonds'] + fuerfonds
          }else{
            BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j - 1, 'fonds']
          }
        }else{
        BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j - 1, 'fonds'] + fuerfonds}
      }
#------ Rückzahlung darf Schuld nicht übersteigen ---------------------#
      if(BILANZ_IV[l_j, 'kap'] > 0) { # teilweise rückzahlung, falls schuld == 0
        BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j, 'fonds'] + BILANZ_IV[l_j, 'kap']
        BILANZ_IV[l_j, 'rueckzahlung'] <- BILANZ_IV[l_j, 'rueckzahlung'] - BILANZ_IV[l_j,
↪  'kap']
        BILANZ_IV[l_j, 'kap'] <- 0
      }
#pas de remb de la dette via les bénéfices de l'AI si externe rueckzahlung
      if("extern_rueckzahlung" %in% colnames(BILANZ_IV)){
        BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j-1, 'fonds'] + BILANZ_IV[l_j, 'erg_betr']
        }

#------ Schuld ist zurückbezahlt, Ertrag geht in den Fonds ------------#
    } else {
      BILANZ_IV[l_j, 'fonds'] <- BILANZ_IV[l_j-1, 'fonds'] + BILANZ_IV[l_j, 'erg_betr']
      BILANZ_IV[l_j, 'rueckzahlung'] <- 0
      BILANZ_IV[l_j, 'kap'] <- BILANZ_IV[l_j-1, 'kap']
    }
  }
```

Hierbei müssen verschiedene gesetzliche und technische Regeln für die Schuldenrückzahlung berücksichtigt werden. Als Erstes wird berücksichtigt, dass seit 2018 nur eine Schuldenrückzahlung erfolgt, falls die nicht-flüssigen Anlagen der IV 50% der Jahresausgaben überschreiten. Die `if`-Bedingungen für `erg_rueckzahlung` und `extern_rueckzahlung` decken die Fälle ab, dass in den Politikmassnahmen eine andere Schuldenrückzahlungsquelle spezifiziert wurde, was Stand 2024 nicht der Fall ist. Als nächstes werden die verschiedenen Fälle abgedeckt, unter welchen die nichtflüssigen Anlagen der IV 50% der Jahresausgaben überschreiten/nicht überschreiten, und dadurch Schulden teilweise oder vollständig zurückbezahlt werden können.

Zum Schluss werden noch die Ausgaben ohne Schuldzinszahlungen, sowie der Stand der nicht-flüssigen Anlagen berechnet:

```
BILANZ_IV <- BILANZ_IV %>%
    mutate(aus_tot_ivoz = aus_tot_iv - schzins, fl_mtl = ifelse(jahr >
        PARAM_GLOBAL$jahr_abr, fonds - PARAM_GLOBAL$fl_mittel_iv *
        aus_tot_iv, fl_mtl), nicht_fl_mtl = PARAM_GLOBAL$fl_mittel_iv *
        aus_tot_iv) %>%
    select(-ahv_zins, -iv_zins)  #, -zinstraeger
```

Zum Schluss wird die resultierende IV-Bilanz an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_ergebnisse.R`) zurückgegeben:

```
#------ Output ------------------------------------------------------#
mod_return(BILANZ_IV)
```

4.24 Module mod_population.R und mod_bevoelkerung.R

`mod_population.R` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_bevoelkerung <- mod_population(list = tl_inp)
```

Die Funktion verlangt die folgenden Argumente:

```
mod_population <- function(PARAM_GLOBAL,
                           BEV_POP,
                           POP_SCENARIO_BEV,
                           POP_SCENARIO_EPT,
                           EMIGRATION_POP,
                           EMIGRATION_SCENARIO_ALT,
                           EMIGRATION_SCENARIO_NEU,
                           ASSURES_FACULTATIFS,
                           FRONTALIERS_OBS,
                           FRONTALIERS_SCEN,
                           SAISONNIERS,
                           list = NULL) {
```

Wovon die folgenden für die in den IV-Finanzperspektiven genutzt werden:

- `BEV_POP`: Beobachtete Wohnbevölkerung.
- `POP_SCENARIO_BEV`: Wohnbevölkerungsszenarien
- `POP_SCENARIO_EPT`: Erwerbsbevölkerungsszenarien

Zu beginn des Skripts werden die Grundlageparameter aus `PARAM_GLOBAL` eingelesen:

```
# Einlesen der Grundparameter
# ------------------------------------------------

bev_scenario_neu <- PARAM_GLOBAL$bev_szenario_neu
bev_scenario_alt <- PARAM_GLOBAL$bev_szenario_alt
bev_scenario_ept_neu <- PARAM_GLOBAL$bev_szenario_ept_neu
bev_scenario_ept_alt <- PARAM_GLOBAL$bev_szenario_ept_alt
bev_ept <- PARAM_GLOBAL$bev_ept
ept_ept <- PARAM_GLOBAL$ept_ept
bev_nurCH <- PARAM_GLOBAL$bev_nurCH
```

Danach wird das Dataframe mit der beobachteten Wohnbevölkerung so bereinigt, dass sichergestellt ist, dass darin nur die ständige CHE Wohnbevölkerung enthalten ist:

```
# Remove some unwanted columns
if ("source" %in% names(BEV_POP)) {
    BEV_POP <- BEV_POP %>%
        dplyr::select(-source)
}

# Berechnung staendige Wohnbevoelkerung

BEV_POP <- BEV_POP %>%
    mutate(dom = "ch")
```

Als nächstes wird abhängig vom Parameter `bev_ept` das Szenario (entweder Wohnbevölkerungs- oder Erwerbsbevölkerungsszenario) für die Berechnung des `BEVOELKERUNG` Dataframe ausgewählt. Da wir im IV Finanzperpektivenmodell das `BEVOELKERUNG` Dataframe als Wohnbevölkerung verwenden, ist der Parameter `bev_ept==FALSE` (Stand 2024) und es wird daher das Wohnbevölkerungsszenario ausgewählt:

```
if (bev_ept) {
    # Population is EPT
    BEV_SCENARIO_ALT <- POP_SCENARIO_EPT %>%
        filter(scenario == bev_scenario_ept_alt) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)

    BEV_SCENARIO_NEU <- POP_SCENARIO_EPT %>%
        filter(scenario == bev_scenario_ept_neu) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)
} else {
    # Population is Wohnbevölkerung
    BEV_SCENARIO_ALT <- POP_SCENARIO_BEV %>%
        filter(scenario == bev_scenario_alt) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)

    BEV_SCENARIO_NEU <- POP_SCENARIO_BEV %>%
        filter(scenario == bev_scenario_neu) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)
}
```

Als nächstes wird mit `mod_bevoelkerung` (vgl. Kapitel 4.24.1 weiter unten) das `BEVOELKERUNG` Dataframe berechnet:

```
tl_bevoelkerung <- mod_bevoelkerung(PARAM_GLOBAL, BEV_POP, BEV_SCENARIO_ALT,
    BEV_SCENARIO_NEU)

BEVOELKERUNG <- tl_bevoelkerung$BEVOELKERUNG
```

Als nächstes folgt die Berechnung des `AKTIVE_BEV` Dataframes, wobei die genau gleiche Logik gilt wie oben. Da wir im IV Finanzperpektivenmodell das `AKTIVE_BEV` Dataframe als Erwerbsbevölkerung verwenden, ist der Parameter `ept_ept==TRUE` (Stand 2024) und es wird daher das Erwerbsbevölkerungsszenario in Vollzeitäquivalenten ausgewählt:

```
# Berechnung aktive Wohnbevoelkerung

if (ept_ept) {
    BEV_SCENARIO_ALT <- POP_SCENARIO_EPT %>%
        filter(scenario == bev_scenario_ept_alt) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)

    BEV_SCENARIO_NEU <- POP_SCENARIO_EPT %>%
        filter(scenario == bev_scenario_ept_neu) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)
} else {
    BEV_SCENARIO_ALT <- POP_SCENARIO_BEV %>%
        filter(scenario == bev_scenario_alt) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)
```

```
    BEV_SCENARIO_NEU <- POP_SCENARIO_BEV %>%
        filter(scenario == bev_scenario_neu) %>%
        mutate(dom = "ch") %>%
        dplyr::select(-scenario)
}

tl_aktive_bev <- mod_bevoelkerung(PARAM_GLOBAL, BEV_POP, BEV_SCENARIO_ALT,
    BEV_SCENARIO_NEU)

AKTIVE_BEV <- tl_aktive_bev$BEVOELKERUNG
```

Die restlichen Berechnungen in diesem Modul sind für das IV-Finanzperspektivenmodell nicht relevant.

4.24.1 Modul mod_bevoelkerung.R

`mod_bevoelkerung.R` wird im Skript `mod_population.R` wie folgt aufgerufen:

```
tl_bevoelkerung <- mod_bevoelkerung(PARAM_GLOBAL, BEV_POP, BEV_SCENARIO_ALT,
    BEV_SCENARIO_NEU)
```

Die folgenden Dataframes werden übergeben

- `BEV_POP`: Beobachtete Wohnbevölkerung (STATPOP).
- `BEV_SCENARIO_ALT`: Letztes (Wohn- oder Erwerbs-)Bevölkerungsszenario.
- `BEV_SCENARIO_NEU`: Aktuelles (Wohn- oder Erwerbs-)Bevölkerungsszenario.

Zu beginn des Skripts werden die Grundlageparameter aus `PARAM_GLOBAL` eingelesen:

```
# Einlesen der Grundparameter
# ------------------------------------------------

jahr_bev_base <- PARAM_GLOBAL$jahr_bev_base
jahr_ende <- PARAM_GLOBAL$jahr_ende
```

Als nächstes werden die Szenariodaten bereinigt, sodass sichergestellt ist, dass nur Jahre für welche tatsächlich Szenariodaten existieren behalten werden:

```
# Trimmen der Szenariodaten
# ------------------------------------------------

min_jahr_alt <- BEV_SCENARIO_ALT %>%
    group_by(jahr) %>%
    summarize(pop = sum(bevendejahr)) %>%
    ungroup() %>%
    filter(pop > 0)

BEV_SCENARIO_ALT <- BEV_SCENARIO_ALT %>%
    filter(jahr >= min(min_jahr_alt$jahr))

min_jahr_neu <- BEV_SCENARIO_NEU %>%
    group_by(jahr) %>%
    summarize(pop = sum(bevendejahr)) %>%
    ungroup() %>%
    filter(pop > 0)
```

```
BEV_SCENARIO_NEU <- BEV_SCENARIO_NEU %>%
    filter(jahr >= min(min_jahr_neu$jahr))

# Falls die beobachtete Wohnbevölkerung nicht bis zum
# vorgesehenen Justierungsjahr vorhanden ist, wird das
# Justierungsjahr auf das jüngste Beobachtungsjahr gesetzt.

if (max(BEV_POP$jahr) < jahr_bev_base) {
    jahr_bev_base <- max(BEV_POP$jahr)
}
```

Der nächste Schritt ist das zusammenführen der beobachteten Bevölkerung (`BEV_POP`) mit dem Bevölkerungsszenario:

```
if (min(BEV_SCENARIO_NEU$jahr) <= jahr_bev_base) {

    # Falls Justierungsjahr innerhalb des neuen Szenarios
    # ----------------------

    # Berechnen des Skalenfaktors im Justierungsjahr

    BEV_ABR_JAHR <- BEV_SCENARIO_NEU %>%
        filter(jahr == jahr_bev_base) %>%
        rename(bev_scen = bevendejahr) %>%
        left_join(dplyr::select(filter(BEV_BEO, jahr == jahr_bev_base),
            -jahr), by = c("sex", "nat", "alt", "dom")) %>%
        rename(bev_beo = bevendejahr) %>%
        mutate(scale_faktor = if_else(bev_scen != 0, bev_beo/bev_scen,
            1)) %>%
        dplyr::select(-jahr)

    # Skalieren der Scenariodaten auf das Niveau der
    # Beobachtungsdaten

    BEV_PR <- BEV_SCENARIO_NEU %>%
        filter(jahr >= jahr_bev_base) %>%
        rename(bev = bevendejahr) %>%
        left_join(BEV_ABR_JAHR, c("sex", "nat", "alt", "dom")) %>%
        mutate(bevendejahr = bev * scale_faktor) %>%
        dplyr::select(jahr, sex, nat, alt, dom, bevendejahr) %>%
        filter(jahr > jahr_bev_base) %>%
        filter(jahr <= jahr_ende)

    BEVOELKERUNG <- rbind(BEV_BEO, BEV_PR)
}
```

Oben abgedeckt ist der Fall, dass das erste Jahr des neuen Bevölkerungsszenarios tiefer ist als das Basisjahr der beobachteten Wohnbevölkerung (Stand 2024 ist dies der Fall, da das Bevölkerungsszenario 2020 Projektionen ab 2019 enthält, und das letzte Jahr von `BEV_POP` 2022 ist). Der Code berechnet die Abweichung des Bevölkerungsszeanrios von der beobachteten Wohnbevölkerung im Basisjahr (nach Alter, Geschlecht und Nationalität Zelle), und hebt dann sämtliche zukünftigen Szenariodaten entsprechend dieser Abweichung an.

Der `else`-Teil unten deckt den Fall ab, dass das Erste Jahr des neuen Bevölkerungsszenarios weiter zurück liegt als das Basisjahr der beobachteten Wohnbevölkerung. In dem Fall muss das alte Bevölkerungsszenario

für die Berechnung der Abweichung im Basisjahr verwendet werden, und dann nochmals die Abweichung zwischen dem alten und dem neuen Szenario:

```
else {

    # Falls Justierungsjahr innerhalb des alten Szenarios ----------------------

    # Berechnen des Skalenfaktors im Justierungsjahr

    BEV_ABR_JAHR <- BEV_SCENARIO_ALT %>%
      filter(jahr == jahr_bev_base) %>%
      rename(bev_scen = bevendejahr) %>%
      left_join(dplyr::select(filter(BEV_BEO, jahr == jahr_bev_base), -jahr),
        by = c("sex", "nat", "alt", "dom")
      ) %>%
      rename(bev_beo = bevendejahr) %>%
      mutate(scale_faktor = if_else(bev_scen != 0, bev_beo / bev_scen, 1)) %>%
      dplyr::select(-jahr)

    # Skalieren der alten Scenariodaten auf das Niveau der Beobachtungsdaten

    BEV_PR <- BEV_SCENARIO_ALT %>%
      filter(jahr >= jahr_bev_base) %>%
      rename(bev = bevendejahr) %>%
      left_join(BEV_ABR_JAHR, c("sex", "nat", "alt", "dom")) %>%
      mutate(bevendejahr = bev * scale_faktor) %>%
      dplyr::select(jahr, sex, nat, alt, dom, bevendejahr) %>%
      filter(jahr > jahr_bev_base) %>%
      filter(jahr <= jahr_ende)

    BEVOELKERUNG <- rbind(BEV_BEO, BEV_PR)

    # Ergänzen der beobachteten Bevölkerung mit den justierten Daten des
    # alten Szenarios bis zum letzten Jahr vor dem neuen Szenario

    BEV_BEO_NEU <- BEVOELKERUNG %>%
      filter(jahr <= min(BEV_SCENARIO_NEU$jahr))

    # Berechnen des Skalenfaktors im letzten Jahr vor dem neuen Szenario

    BEV_ABR_JAHR <- BEV_SCENARIO_NEU %>%
      filter(jahr == min(BEV_SCENARIO_NEU$jahr)) %>%
      rename(bev_scen = bevendejahr) %>%
      left_join(dplyr::select(
        filter(BEV_BEO_NEU, jahr == min(BEV_SCENARIO_NEU$jahr)),
        -jahr
      ),
      by = c("sex", "nat", "alt", "dom")
      ) %>%
      rename(bev_beo = bevendejahr) %>%
      mutate(scale_faktor = if_else(bev_scen != 0, bev_beo / bev_scen, 1)) %>%
      dplyr::select(-jahr)

    # Skalieren der neuen Szenariodaten auf das Niveau der projizierten
```

```
    # Beobachtungsdaten im letzten Jahr vor dem neuen Szenario

    BEV_PR <- BEV_SCENARIO_NEU %>%
      filter(jahr >= min(BEV_SCENARIO_NEU$jahr)) %>%
      rename(bev = bevendejahr) %>%
      left_join(BEV_ABR_JAHR, c("sex", "nat", "alt", "dom")) %>%
      mutate(bevendejahr = bev * scale_faktor) %>%
      dplyr::select(jahr, sex, nat, alt, dom, bevendejahr) %>%
      filter(jahr > min(BEV_SCENARIO_NEU$jahr)) %>%
      filter(jahr <= jahr_ende)

    BEVOELKERUNG <- rbind(BEV_BEO_NEU, BEV_PR)
  }
```

Somit ist die Berechnung des `BEVOELKERUNG` Dataframes abgeschlossen, und ebendieses kann an das Modul `mod_population.R` zurückgegeben werden:

```
  # Output ------------------------------------------------------------------

  mod_return(BEVOELKERUNG)
}
```

4.25 Modul mod_eckwerte.R

`mod_eckwerte.R` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_eckwerte <- mod_eckwerte(list = tl_inp)
```

Die Funktion verlangt die folgenden Argumente:

```
mod_eckwerte <- function(PARAM_GLOBAL,
                         ECKWERTE,
                         INDICES,
                         list = NULL) {
```

- `ECKWERTE`: Datafrme mit allen Projektionen zu den Eckwerten zur wirtschaftlichen Entwicklung (insbesondere der Lohn- und Preisentwicklung) gemäss ESTV.
- `INDICES`: Historische Lohn- und Preisindizes.

Zu beginn werden die gemäss `PARAM_GLOBAL` spezifizierten Eckwerte extrahiert:

```
ECKWERTE_SCENARIO <- ECKWERTE %>%
    filter(id == PARAM_GLOBAL$id_eckwerte) %>%
    filter(jahr <= PARAM_GLOBAL$jahr_ende)
```

Danach werden zwei Sicherheitsüberprüfungen durchgeführt, die sicherstellen, dass die Eckwerte korrekt spezifiziert sind:

```
if (PARAM_GLOBAL$jahr_abr < (min(ECKWERTE_SCENARIO$jahr) - 1)) {
    stop(paste0("ECKWERTE_SCENARIO empty for year ", PARAM_GLOBAL$jahr_abr))
}

if (min(ECKWERTE_SCENARIO$jahr) < PARAM_GLOBAL$jahr_abr) {
    stop(paste0("Eckwerte scenario PARAM_GLOBAL$id_eckwerte = ",
        PARAM_GLOBAL$id_eckwerte, " does not match PARAM_GLOBAL$jahr_abr = ",
```

```
        PARAM_GLOBAL$jahr_abr))
}
```

Als nächstes werden die historischen Eckwerte extrahiert, wobei sichergestellt wird, dass nur Eckwerte bis zum ersten verfügbaren projizierten Eckwert ausgewählt werden (was, ausser im Falle eines Backtestings bedeutet, dass alle verfügbaren historischen Eckwerte ausgewählt werden):

```
INDICES_FILTERED <- INDICES %>%
    filter(jahr < min(ECKWERTE_SCENARIO$jahr))
```

In einem nächsten Schritt erfolgt die Berechnung des jährlichen Lohn- und Preiswachstums. Für die Berechnung des Preiswachstums wird im Falle dass `PARAM_GLOBAL$pi_eckwerte==1` ist (was standardmässig der Fall ist) berücksichtigt, dass der für die Preisentwicklung relevante Index bis 2016 `pidez` (Preisindex im Dezember) ist, und ab 2017 `pimyr` (Jahresmittel des Preisindexes):

```
ECKWERTE_HIST <- INDICES_FILTERED %>%
    mutate(pi = case_when(PARAM_GLOBAL$pi_eckwerte == 0 ~ INDICES_FILTERED$pidez,
        PARAM_GLOBAL$pi_eckwerte == 1 & INDICES_FILTERED$jahr <
            2017 ~ INDICES_FILTERED$pidez, PARAM_GLOBAL$pi_eckwerte ==
            1 & INDICES_FILTERED$jahr >= 2017 ~ INDICES_FILTERED$pimyr,
        TRUE ~ INDICES_FILTERED$pimyr)) %>%
    mutate(lohn = 100 * (li - lag(li))/lag(li)) %>%
    mutate(preis = 100 * (pi - lag(pi))/lag(pi)) %>%
    dplyr::select(jahr, lohn, struktur, preis)
```

Als nächstes wird die vorangehend ausgewählte Eckwerte-Projektion ausgewählt:

```
ECKWERTE_GO <- ECKWERTE_SCENARIO %>%
    dplyr::select(jahr, lohn, struktur, preis)
```

Nun wird die Eckwerte Projektion für die fehlenden Jahre bis zum Ende des Projektionshorizonts erstellt:

```
  # Konstruktion der Eckwerte ab dem ausgewählten Szenario ----------------------

  if (last(ECKWERTE_GO$jahr) < PARAM_GLOBAL$jahr_ende) {
    ECKWERTE_PR <- crossing(
      jahr = (last(ECKWERTE_GO$jahr) + 1):PARAM_GLOBAL$jahr_ende,
      tail(
        ECKWERTE_GO %>%
          dplyr::select(-jahr),
        1
      )
    )
```

Hierbei wird zuerst geprüft, ob das letzte Jahr mit einer verfügbaren Eckwerte Projektion `last(ECKWERTE_GO$jahr)` tiefer ist als das letzte Jahr des Projektionshorizonts `PARAM_GLOBAL$jahr_ende`. Wenn dies der Fall ist, wird die Eckwerte-Projektion vom Jahr `last(ECKWERTE_GO$jahr)` für alle Jahre bis PARAM_GLOBAL$jahr_ende‘ verwendet. Stand 2024 sind Eckwerteprojektionen bis 2034 verfügbar, und das letzte Jahr des Projektionshorizonts ist 2070, d.h. die Eckwerte-Projektion für das Jahr 2034 wird für alle Jahre bis 2070 verwendet.

Als nächstes wird das Dataframe `ECKWERTE_EXTENDED` aus den historischen un den projizierten Eckwerten gebildet, wobei das Dataframe `ECKWERTE_PR` natürlich nur verwendet wird, wenn obige `if`-bedingung `TRUE` ist:

```
    ECKWERTE_EXTENDED <- bind_rows(
      ECKWERTE_HIST,
      ECKWERTE_GO,
      ECKWERTE_PR
    )
  } else {
    ECKWERTE_EXTENDED <- bind_rows(
      ECKWERTE_HIST,
      ECKWERTE_GO
    )
  }
```

Der nachfolgende Code deckt den Fall ab, dass ein Projektionshorizont über das Jahr 2090 hinaus ausgewählt wird. Da der Parameter `PARAM_GLOBAL$jahr_ende > 2090` Stand 2024 `FALSE` ist wird hier nicht näher auf diesen Codeteil eingegangen:

```
# Adaptation pour le scénario 3000
if (PARAM_GLOBAL$jahr_ende > 2090) {
    if (PARAM_GLOBAL$modif_scenario_3000_on_eckwerte == TRUE) {
        EXT_ECKWERTE_EXTENDED <- ECKWERTE_EXTENDED %>%
            filter(jahr == PARAM_GLOBAL$new_anneeseuil_eckwerte) %>%
            dplyr::select(-jahr) %>%
            crossing(jahr = (PARAM_GLOBAL$new_anneeseuil_eckwerte +
                1):PARAM_GLOBAL$jahr_ende) %>%
            mutate(lohn = PARAM_GLOBAL$new_lohn, preis = PARAM_GLOBAL$new_preis,
                struktur = PARAM_GLOBAL$new_struktur)

        ECKWERTE_EXTENDED <- ECKWERTE_EXTENDED %>%
            filter(jahr <= PARAM_GLOBAL$new_anneeseuil_eckwerte) %>%
            bind_rows(EXT_ECKWERTE_EXTENDED)
    } else {
        ECKWERTE_EXTENDED <- ECKWERTE_EXTENDED
    }
}
```

Somit ist die Berechnung des `ECKWERTE_SCENARIO` sowie des `ECKWERTE_EXTENDED`Dataframes abgeschlossen, wodurch diese an das Modul `wrap_vorb_berechn.R` zurückgegeben werden können:

```
mod_return(ECKWERTE_SCENARIO, ECKWERTE_EXTENDED)
```

4.26 Modul mod_eink_entwicklung.R

Dokumentation zuletzt aktualisiert am 24.09.2024

`mod_eink_entwicklung.R` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_eink_entwicklung <- mod_eink_entwicklung(list = c(tl_inp,
    tl_eckwerte))
```

Die Funktion verlangt die folgenden Argumente:

```
mod_eink_entwicklung <- function(PARAM_GLOBAL,
                                 ECKWERTE_EXTENDED,
                                 list = NULL) {
```

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.25)

Unter Nutzung der projizierten Wachstumsrate von `lohn` (gemäss dem nominalen Schweizerischen Lohnindex) und des Strukturfaktors wird daraus die Entwicklung des Durchschnittslohnes, in der Modellterminologie als `eink_entwicklung` bezeichnet, ab dem letzten Jahr, für welches der Auszug aus den individuellen Konten verfügbar ist und verwendet wird (`PARAM_GLOBAL$jahr_ik`), fortgeschrieben. Es gilt die Annahme, dass sich der Durchschnittslohn mit dem Produkt der Wachstumsraten von `lohn` und `struktur` entwickelt:

```
EINK_ENTWICKLUNG <- ECKWERTE_EXTENDED %>%
    filter(jahr > PARAM_GLOBAL$jahr_ik) %>%
    mutate(lohnindexentwicklung = cumprod(1 + lohn/100), strukturentwicklung = cumprod(1
    ↪  +
        struktur/100), eink_entwicklung = lohnindexentwicklung *
        strukturentwicklung) %>%
    dplyr::select(jahr, eink_entwicklung) %>%
    bind_rows(tibble(jahr = c(PARAM_GLOBAL$jahr_beginn:PARAM_GLOBAL$jahr_ik),
        eink_entwicklung = 1)) %>%
    arrange(jahr)
```

Somit ist die Berechnung des `EINK_ENTWICKLUNG` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_vorb_berechn.R` zurückgegeben werden kann:

```
# Output
# ------------------------------------------------------------------------

mod_return(EINK_ENTWICKLUNG)
```

4.27 Modul mod_diskontfaktor.R

Dokumentation zuletzt aktualisiert am 23.09.2024

`mod_diskontfaktor.R` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_diskontfaktor <- mod_diskontfaktor(list = c(tl_inp, tl_eckwerte))
```

Die Funktion verlangt die folgenden Argumente:

```
mod_diskontfaktor <- function(PARAM_GLOBAL,
                              ECKWERTE_EXTENDED,
                              list = NULL) {
```

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.25)

Zu Beginn des Moduls wird überprüft, ob das Jahr, welches als Preisbasis gewählt wurde, innerhalb der Jahre liegt, für welche in `ECKWERTE_EXTENDED` Werte für die Preisentwicklung vorhanden sind. Falls das nicht der Fall ist wird die Ausführung angehalten und eine Fehlermeldung ausgegeben:

```
# check if jahr_preisbasis exists
if (!"jahr_preisbasis" %in% names(PARAM_GLOBAL)) {
    stop("A value must be provided for jahr_preisbasis in PARAM_GLOBAL")
}

# check for NA values
ECKWERTE_EXTENDED$preis[1] <- 0
if (any(is.na(ECKWERTE_EXTENDED$jahr))) {
```

```
    stop("There are NA values in ECKWERTE_EXTENDED$jahr")
}
if (any(is.na(ECKWERTE_EXTENDED$preis))) {
    stop("There are NA values in ECKWERTE_EXTENDED$preis")
}

# check if jahr_preisbasis is in the range provided by
# ECKWERTE_EXTENDED
jahr_preisbasis <- PARAM_GLOBAL$jahr_preisbasis
min_jahr <- min(ECKWERTE_EXTENDED$jahr)
max_jahr <- max(ECKWERTE_EXTENDED$jahr)
if (!PARAM_GLOBAL$jahr_preisbasis %in% min_jahr:max_jahr) {
    msg <- paste0("The value for jahr_preisbasis must be in the interval ",
        "[", min_jahr, ", ", max_jahr, "].")
    stop(msg)
}
```

Somit erfolgt die Berechnung des Preisdeflators, welcher in der Modellterminologie als Diskontfaktor bezeichnet wird:

```
# calculate DISKONTFAKTOR
DISKONTFAKTOR <- ECKWERTE_EXTENDED |>
    dplyr::mutate(preisindex = cumprod(1 + preis/100)) |>
    dplyr::mutate(diskontfaktor = preisindex[jahr == jahr_preisbasis]/preisindex) |>
    dplyr::select(jahr, diskontfaktor)
```

Somit ist die Berechnung des `DISKONTFAKTOR` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_vorb_berechn.R` zurückgegeben werden kann:

```
mod_return(DISKONTFAKTOR)
```

4.28 Modul mod_strukturfaktor.R

Dokumentation zuletzt aktualisiert am 20.09.2024

`mod_strukturfaktor` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_strukturfaktor <- mod_strukturfaktor(list = c(tl_inp, tl_bevoelkerung,
    tl_eink_entwicklung, tl_diskontfaktor, tl_abrechnung))
```

Die Funktion verlangt die folgenden Argumente:

```
mod_strukturfaktor <- function(PARAM_GLOBAL,
                       AKTIVE_BEV,
                       IK,
                       EINK_ENTWICKLUNG,
                       AHV_ABRECHNUNG,
                       DISKONTFAKTOR,
                       ECKWERTE,
                       list = NULL) {
```

- `AKTIVE_BEV`: Die Wohnbevölkerung gemäss Statistik fortgeschrieben mit den Erwerbsbevölkerungsszenarien des BFS. Es werden ausschliesslich die Wachstumsraten über die Zeit innerhalb der Zellen für die Projektion verwendet werden.
- `IK`: Die Daten gemäss der individuellen Konten der ZAS.

- `EINK_ENTWICKLUNG`: Zeitreihe mit der Entwicklung des Durchschnittslohnes $eink\_entwicklung = \frac{SLI_t^{nominal} * (1 + Sturkturfaktor)}{SLI_{Abrechnungsjahr}^{nominal}}$. Hier handelt es sich um die Einkommensentwicklung VOR dem Ausführen von mod_strukturfaktor.R. Das heisst, die Einkommensentwicklung, wenn der Strukturfaktor nicht die Abweichung der BESTA von den BFS-Erwerbsbevölkerungsszenarien enthält.
- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `DISKONTFAKTOR`: Da das Modell in realen grössen gerechnet wird, wird der Diskontfaktor benötigt, um nominale grössen vor der Modellberechnung in reale Grössen umzuwandeln, respektive um die realen Modellprognosen am Ende in nominale Grössen umzuwandeln.

Am Anfang des Moduls wird das Wachstum der Lohnbeiträge ohne Anpassung des Strukturfaktors an die BESTA gerechnet, also ein hypothetisches Lohnbeitragswachstum wenn wir auch in den ersten zwei Jahre ab der aktuellen Abrechnung die Erwerbsbevölkerungsszenarien des BFS für die Schätzung der Beitragsentwicklung verwenden würden. Die durchgeführten Berechnungen sind identisch wie in `mod_beitragssumme.R` (vgl. Kapitel 4.18.1) und werden hier nicht noch einmal erläutert.

Der spezifische Teil der Berechnungen in `mod_strukturfaktor` beginnt mit dem folgenden Code:

```
# Berechnung von delta (rate_ept)
rate_bl <- rate_bl %>%
    pivot_longer(cols = !(jahr | btr_nom), names_to = "series",
        values_to = "y") %>%
    mutate(y = y/100) %>%
    pivot_wider(names_from = series, values_from = y) %>%
    mutate(rate_ept = (1 + rate_btr_nom)/((1 + rate_li_nom) *
        (1 + rate_str_bl) * (1 + rate_staf)) - 1) %>%
    pivot_longer(cols = !(jahr | btr_nom), names_to = "series",
        values_to = "y") %>%
    mutate(y = y * 100) %>%
    pivot_wider(names_from = series, values_from = y)
```

Entscheidend ist hier die Zeile `mutate(rate_ept = (1+rate_btr_nom)/((1+rate_li_nom)* (1+rate_str_bl)*(1+rate_staf))-1)`. Die darin auf der rechten Seite der Gleichung enthaltenen Variablen repräsentieren die folgenden Werte:

-`rate_btr_nom`: Das nominale Wachstum in % der totalen Lohnbeiträge von einem Jahr zum nächsten.
-`rate_li_nom`: Das Wachstum des nominalen Lohnindexes in %.
-`rate_str_bl`: Der von uns angenommene Strukturfaktor ohne Anpassung des Strukturfaktors an die BESTA, also 0.003 hier.

Dadurch ergibt sich, dass `rate_ept` der Wachstumsrate der Lohnbeiträge abzüglich dem Teil dieses Wachstums, das durch das Wachstum des Lohnindexes und des Strukturfaktors erklärt ist, entspricht. Es handelt sich also effektiv um den Anteil des Wachstums der Lohnbeiträge, der durch das Bevölkerungswachstum gemäss dem BFS-Bevölkerungsszenario erklärt ist.

Mit dem folgenden Code wird das Wachstum der Beschäftigung gemäss der BESTA-Projektionen extrahiert:

```
# # Laufende Version der SECO Eckwerte
besta_eckw <- besta_seco

# rate_ept: baseline
besta_series <- full_join(besta_eckw, rate_bl %>%
    select(jahr, rate_ept) %>%
    filter(jahr >= start_eck), by = "jahr")
# rate_besta für alle Jahre definiert
besta_series <- besta_series %>%
    mutate(rate_besta = case_when(between(jahr, jahr_abr + 1,
```

```
        jahr_abr + 2) ~ rate_besta, jahr > jahr_abr + 2 ~ rate_ept))
```

Die Beschränkung auf `jahr_abr+1` und `jahr_abr+2` erklärt sich dadurch, dass die BESTA-Projektion nur für die ersten 2 Jahre ab der aktuellen Abrechnung existiert.

```
# # Laufende Version der SECO Eckwerte
besta_eckw <- besta_seco

# rate_ept: baseline
besta_series <- full_join(besta_eckw, rate_bl %>%
    select(jahr, rate_ept) %>%
    filter(jahr >= start_eck), by = "jahr")
# rate_besta für alle Jahre definiert
besta_series <- besta_series %>%
    mutate(rate_besta = case_when(between(jahr, jahr_abr + 1,
        jahr_abr + 2) ~ rate_besta, jahr > jahr_abr + 2 ~ rate_ept))
```

Hierauf folgt die eigentliche Berechnung der Anpassung des Strukturfaktors an die Entwicklung der Beschäftigung:

```
# Anpassung des Strukturfaktors an die Entwicklung der
# Beschäftigung
str_besta <- besta_series %>%
    mutate(rate_str_new = 100 * ((1 + 0.3/100) * (1 + rate_besta/100)/(1 +
        rate_ept/100) - 1))
```

`rate_str_new`, also der angepasste Strukturfaktor, entspricht dem Strukturfaktor von 0.3 multipliziert mit der Abweichung zwischen dem Wachstum der Erwerbsbevölkerung (respektive der Beschäftigung) gemäss BESTA, und dem Wachstum der Erwerbsbevölkerung gemäss BFS-Szenario. Somit stellt der Strukturfaktor sicher, dass bei späterer Verwendung der Formel $Wachstum\ Lohnbeiträge\ in\ \% \approx Wachstum\ SLI + Wachstum\ Erwerbsevölkerung\ gemäss\ Szenario\ BFS + 0.3\%$ der Teil $Wachstum\ Erwerbsevölkerung\ gemäss\ Szenario\ BFS + 0.3\%$ für die ersten zwei Jahre ab der Abrechnung $Wachstum\ Beschäftigung\ gemäss\ BESTA + 0.3\%$ entspricht.

Der Rest des Codes dient dann dazu, den angepassten Strukturfaktor zum gemäss `PARAM_GLOBAL` ausgewählten ECKWERTE-Szenario hinzuzufügen (d.h., da den Strukturfaktor für die Ersten 2 Jahre zu ersetzten), und die Eckwerte an das Finanzperspektivenmodell, respektive `wrap_vorb_berechn.R` zurückzugeben:

```
besta <- str_besta %>%
    filter(jahr >= start_eck & jahr <= end_eck) %>%
    select("rate_besta") %>%
    rename(besta = rate_besta)


rate_str_new <- str_besta %>%
    filter(jahr >= start_eck & jahr <= end_eck) %>%
    select(jahr, rate_str_new) %>%
    select(jahr, rate_str_new) %>%
    mutate(rate_str_new = round(rate_str_new, 2))
ECKWERTE_SCENARIO <- ECKWERTE_SCEN %>%
    select("id", "laufjahr", "version", "stand", "jahr", "lohn") %>%
    left_join(rate_str_new, by = "jahr") %>%
    rename(struktur = rate_str_new) %>%
    mutate(struktur = if_else(is.na(struktur), 0.3, struktur)) %>%
    add_column(ECKWERTE_SCEN %>%
```

```
        select("preis")) %>%
    add_column(besta %>%
        select("besta") %>%
        mutate(besta = round(besta, 1))) %>%
    add_column(ECKWERTE_SCEN %>%
        select("BIP_nominal", "BIP_real", "BIP_deflator"))


ECKWERTE <- ECKWERTE %>%
    filter(id != id_eckwerte) %>%
    add_row(ECKWERTE_SCENARIO)

# Output
# ------------------------------------------------------------------------

mod_return(ECKWERTE)
```

4.29 Modul mod_rentenentwicklung.R

Dokumentation zuletzt aktualisiert am 23.09.2024

`mod_strukturfaktor` wird im Skript `wrap_vorb_berechn.R` wie folgt aufgerufen:

```
tl_rentenentwicklung <- mod_rentenentwicklung(list = c(tl_inp,
    tl_eckwerte))
```

Die Funktion verlangt die folgenden Argumente:

```
mod_rentenentwicklung <- function(PARAM_GLOBAL,
                                  ECKWERTE_EXTENDED,
                                  MINIMALRENTE,
                                  list = NULL) {
```

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.25)
- `MINIMALRENTE`: Historische Zeitreihe mit der im jeweiligen Jahr gemäss Verordnung gültigen Minimalrente.

Als erstes wird das letzte Jahr mit einer gemäss Verordnung festgelegten Minimalrente extrahiert (typischerweise das aktuelle oder das nächste Jahr):

```
# Dernière année enregistrée dans le fichier MINIMALRENTE
last_year <- last(MINIMALRENTE$jahr)
```

Als nächstes wird anhand der projizierten Lohn- und Preisentwicklung, welche im Dataframe `ECKWERTE_EXTENDED` enthalten ist, die Entwicklung der für die Minimalrentenberechnung relevanten Indizes (insbesondere des Mischindexes) projiziert. Eine Erläuterung zu der Berechnungsformel findet sich in den Hintergrunddokumentationen zum Mischindex (auf Anfrage beim Bereich MATH des BSV erhältlich):

```
  RENTENINDEX_ZRO <- ECKWERTE_EXTENDED %>%
    mutate(
      # remplacer les NA pour l'année 1979 par 0 (pour faciliter les calculs
      # avec cumprod)
      lohn = if_else(is.na(lohn), 0, lohn),
      preis = if_else(is.na(preis), 0, preis),
      # calcul de l'évolution de l'indice des salaires et du prix à l'aide des
```

```
    # Eckwerte
    lientw = cumprod(1 + (lohn / 100)),
    pientw = cumprod(1 + (preis / 100)),
    # calcul de l'indice des salaires et du prix à partir de leur évolution
    # projetée
    lohnindex = round(1004 * lientw),
    preisindex = round(104.1 * pientw, digits = 1),
    # Calcul des composantes du salaire et du prix
    licomp = round(lag(lohnindex) / 10.04, digits = 4),
    picomp = round(lag(preisindex) / 1.041, digits = 4),
    # Calcul de l'indice mixte
    mischindex = round((PARAM_GLOBAL$gew_li_comp * licomp +
      PARAM_GLOBAL$gew_pi_comp * picomp) /
      (PARAM_GLOBAL$gew_li_comp + PARAM_GLOBAL$gew_pi_comp), digits = 4),
    # Calcul de la rente minimale non arrondie (rw = raw)
    minimalrente_rw = 5.5 * mischindex,
    # Calcul de la rente minimale arrondie à PARAM_GLOBAL$minrente_rd
    minimalrente_rd = PARAM_GLOBAL$minrente_rd * round(minimalrente_rw
    / PARAM_GLOBAL$minrente_rd)
  ) %>%
```

Als nächstes wird die historische Minimalrente angefügt, und berücksichtigt, dass die Anpassung der Minimalrente nur alle zwei Jahre erfolgt:

```
  # Ajout de la rente minimale selon l'ordonnance (Verordnung)
  left_join(MINIMALRENTE %>%
    mutate_all(list(~as.numeric(.))),
  by = "jahr"
  ) %>%
  # renommer la variable minimalrente avec le suffixe vo (Verordnung)
  rename(minimalrente_vo = minimalrente) %>%
  mutate(
    # Créer une variable dummy "rentenanpassung" qui prend la valeur 1 tous
    # les deux ans car l'adaptation des rentes se fait au besoin tous les deux
    # ans.Vorsicht: Eine flexible Handhabung des Rentenanpassungsrhythmus im
    # Zuge sich verändernder Eckwerte ist noch nicht implementiert!
    rentenanpassung = case_when(
      jahr <= last_year ~ if_else(dminimalrente > 0, 1, 0),
      TRUE ~ jahr %% 2
    ),
    # Projection de la rente minimale : si l'adaptation des rentes a eu lieu
    # l'année précédente (t-1 => rentenanpassung_t = 0), prendre la rente
    # minimale de l'année précédente, sinon prendre la rente minimale de
    # l'année en cours t (rentenanpassung_t = 1).
    # Vorsicht: Falls beide Jahre mit Rentenanpassung 0 ist minimalrente_pj 0.
    minimalrente_pj = pmax(
      rentenanpassung * minimalrente_rd,
      lag(rentenanpassung * minimalrente_rd)
    )
  )
```

die variable `rentenanpassung` wird so erstellt, dass sie `=1` ist für alle Jahre bis `last_year`, in welchem eine Rentenanpassung erfolgte, und danach alle 2 Jahre. Rentenanpassung wird dann für die Erstellung der Minimalrentenprojektion verwendet, wobei `minimalrente_pj` der weiter oben für das entsprechende

Jahr projizierten Minimalrente entspricht falls es sich um ein Rentenanpassungsjahr handelt, und sonst der Minimalrentenprojektion für das Vorjahr.

Der nächste Codeteil fügt die historische Minimalrenten-Zeitreihe mit der projizierten Zeitreihe zusammen:

```
# Jahr mit der letzten bekannten Minimalrente aus der
# Verordnung
cut_year <- RENTENINDEX_ZR0 %>%
    filter(jahr == PARAM_GLOBAL$jahr_abr)

if ("next_ra_known" %in% colnames(PARAM_GLOBAL)) {
    if (PARAM_GLOBAL$next_ra_known) {
        if (cut_year$rentenanpassung == 1) {
            cut_jahr = PARAM_GLOBAL$jahr_abr + 3
        } else {
            cut_jahr = PARAM_GLOBAL$jahr_abr + 2
        }
    } else {
        if (cut_year$rentenanpassung == 1) {
            cut_jahr = PARAM_GLOBAL$jahr_abr + 1
        } else {
            cut_jahr = PARAM_GLOBAL$jahr_abr + 2
        }
    }
} else {
    if (cut_year$rentenanpassung == 1) {
        cut_jahr = PARAM_GLOBAL$jahr_abr + 1
    } else {
        cut_jahr = PARAM_GLOBAL$jahr_abr + 2
    }
}

# Zusammensetzen Vektor mit den Minimalrenten
RENTENINDEX_ZR <- RENTENINDEX_ZR0 %>%
    mutate(minimalrente = case_when(jahr <= cut_jahr ~ minimalrente_vo,
        TRUE ~ minimalrente_pj)) %>%
    dplyr::select(-c(struktur, preis, lohn, dminimalrente, minimalrente_pj))
```

Das Dataframe `cut_year` wird so extrahiert dass es den Wert des Renenindexes im letzten Abrechnungsjahr enthält. Der Codeteil bis zum vorletzten `else`-Befehl sind nur relevant, wenn `next_ra_known" %in% colnames(PARAM_GLOBAL)==TRUE` ist, was Stand 2024 nicht der Fall ist. Daher wird hier näher darauf eingegangen. Im Teil nach dem vorletzten `else`-Befehl wird das `cut_jahr` festgelegt, welches bestimmt, bis wann die durch die Verordnung festgelegte Minimalrente verwendet werden soll, und ab wann die projizierte Minimalrente (ausser im Falle eines Backtestings wird durch diese Bedingung immer die durch die Verordnung vorgebene Zeitreihe ausgewählt, soweit diese Verfügbar ist):

```
# Zusammensetzen Vektor mit den Minimalrenten
RENTENINDEX_ZR <- RENTENINDEX_ZR0 %>%
    mutate(minimalrente = case_when(jahr <= cut_jahr ~ minimalrente_vo,
        TRUE ~ minimalrente_pj)) %>%
    dplyr::select(-c(struktur, preis, lohn, dminimalrente, minimalrente_pj))
```

Als nächstes wird das Dataframe `RENTENENTWICKLUNG` berechnet, welches die Mnimalrente sowie deren kummulierte Wachstumsrate ab dem letzten Jahr, für welches das Rentenregister verfügbar ist (`PARAM_GLOBAL$jahr_rr`), enthält:

```
  # Calcul de la Rentenentwicklung
  RENTENENTWICKLUNG <- RENTENINDEX_ZR %>%
    # On prend les rentes minimales de l'ordonnance pour les années antérieures
    # ou égales à l'année précédente et les rentes minimales projetées pour les
    # années actuelle et futures
    mutate(
      # Calculer le taux de croissance de la rente minimale
      dminimalrente = (minimalrente - lag(minimalrente)) / lag(minimalrente),
      # L'évolution des rentes suit celle de la rente minimale projetée pour les
      # années allant de jahr_lj à jahr_ende, sinon elle est de 1 (on prend la
      # rente minimale de l'ordonnance)
      rentenentwicklung = case_when(
        jahr > PARAM_GLOBAL$jahr_rr ~ cumprod(1 +
          if_else(jahr > PARAM_GLOBAL$jahr_rr,
            dminimalrente,
            0
          )),
        TRUE ~ 1
      )
    ) %>%
    filter(jahr >= PARAM_GLOBAL$jahr_beginn) %>%
    dplyr::select(jahr, lohnindex, preisindex, minimalrente, rentenentwicklung)
```

Somit ist die Berechnung des `RENTENENTWICKLUNG` sowie des `RENTENINDEX_ZR` Dataframes abgeschlossen, wodurch diese an das Modul `wrap_vorb_berechn.R` zurückgegeben werden können:

```
mod_return(RENTENENTWICKLUNG, RENTENINDEX_ZR)
```

4.30 Modul mod_zins_scen.R

Dokumentation zuletzt aktualisiert am 23.09.2024

`mod_zins_scen.R` Bestimmt - ausgehend von der gewählten Preisentwicklung - für alle Jahre des Projektionzeitraums die Zinssätze, mit denen einerseits der IV-Fonds und andererseits die IV-Schulden verzinst werden. Es wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
tl_zins <- mod_zins_scen(list = c(tl_inp, tl_eckwerte))
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.25)
- `ZINS_SCEN`: Szenarien mit realen Zinssätzen für den IV-Fonds und die IV-Schulden

Das Modul `mod_zins_scen.R` extrahiert das in `PARAM_GLOBAL$zins_scen` ausgewählte Zinsszenario, und kopiert die Zinssätze im letzten Jahr, für welches in `zins_scen` Zinssätze spezifiziert sind, für alle Jahre bis zum Ende des Projektionshorizonts:

```
# Choix du scénario
ZINS_SCEN_FILT <- ZINS_SCEN %>%
    filter(scen == PARAM_GLOBAL$zins_scen, jahr >= PARAM_GLOBAL$jahr_abr)

if (last(ZINS_SCEN_FILT$jahr) < PARAM_GLOBAL$jahr_ende) {
    ZINS_SCEN_PR <- crossing(jahr = (last(ZINS_SCEN_FILT$jahr) +
        1):PARAM_GLOBAL$jahr_ende, tail(ZINS_SCEN_FILT %>%
        dplyr::select(-jahr), 1))
```

```
    ZINS_SCEN_FILT <- ZINS_SCEN_FILT %>%
        bind_rows(ZINS_SCEN_PR, .id = "version")

}

ZINS_SCEN_EXTENDED <- ZINS_SCEN_FILT
```

Danach wird das Dataframe `ZINS` erstellt, in welchem die im jeweiligen Jahr gültigen Zinssätze für die IV-Schuld (Variable `iv_zins`), sowie die Zinssätze für den IV-Fonds (`ahv_zins`, diese Benennung rührt daher, dass auf die Fondsstände aller Sozialversicherungen der gleiche Zinssatz angewandt wird, wodurch gemäss der Finanzperspektiven-Konvention die Benennung nach der AHV erfolgt). In untenstehendem Code ist ersichtlich, dass ab dem Jahr 2024 der für `iv_zins` massgebliche reale Zinssatz `real_p_zins` entspricht, und der für `ahv_zins` massgebliche reale Zinssatz `real_a_zins`. Der nominale Zinssatz im jeweiligen Jahr entspricht dann jeweils der Summe des realen Zinssatzes und der Preisentwicklung im entsprechenden Jahr:

```
  ZINS <-
    # Prendre le scénario complet des Eckwerte (pour inclure des chocs possibles)
    ECKWERTE_EXTENDED %>%
    dplyr::select(jahr, preis) %>%
    filter(jahr >= PARAM_GLOBAL$jahr_abr) %>%
    left_join(ZINS_SCEN_EXTENDED %>%
      dplyr::select(-scen, -version), by = "jahr") %>%
    mutate_all(list(~as.numeric(.))) %>%
    mutate(
      # IV-Zins
      iv_zins = case_when(
        jahr %in% c(2016:2017) ~ kap_zins, # KAP-Program
        jahr %in% c(2018:2023) ~ com_zins, # Compenswiss
        TRUE ~ (preis + real_p_zins)
      ) / 100,
      # AHV-Zins
      ahv_zins = if_else(jahr == 2017,
        preis + real_a_zins - korr_zins,
        preis + real_a_zins
      ) / 100
    ) %>%
    dplyr::select(
      jahr,
      ahv_zins,
      iv_zins
    )
```

Somit ist die Berechnung des `ZINS` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_vorb_berechn.R` zurückgegeben werden kann:

```
#------ Output ------------------------------------------------------#

mod_return(ZINS)
```

4.31 Modul mod_sv_satz.R

Dokumentation zuletzt aktualisiert am 03.10.2024

`mod_sv_satz.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
tl_sv_satz <- mod_sv_satz(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    SV_BEITRAGSSATZ = tl_inp$SV_BEITRAGSSATZ)
```

Die Funktion verwendet neben `PARAM_GLOBAL` das folgende Dataframe:

- `SV_BEITRAGSSATZ`: Dataframe mit allen historischen Beitragssätzen für Lohnbeiträge, insbesondere auch den IV-Beitragssatz.

Das Modul `mod_sv_satz.R` extrahiert den letzten verfügbaren gültigen historischen Beitragssatz (Stand 2024 den in 2024 gültigen Beitragssatz, wobei bei ALV der Wert für 2024 in den Daten fehlt und daher der Wert von 2023 verwendet wird), und kopiert diesen für alle Jahre bis zum Ende des Projoektionshorizonts. Danach fügt er die Zeitreihe mit den projizierten Beitragssätzen mit den historischen Beitragssätzen zusammen:

```
#------ Bereinigen und Gesamtsatz berechnen ---------------------------#
  SV_SATZ       <- SV_BEITRAGSSATZ %>%
      fill(-jahr, .direction = "down") %>%  #Potentiell fehlende Zellen mit letztem
      ↪  Bekannten Wert ersetzten
    select(jahr, ahv_vs_ag, iv_vs_ag, eo_vs_ag, alv_vs_ag) %>%
      mutate(across(c(-jahr), list(~if_else(is.na(.), 0, .)), .names = '{.col}') ) %>%
      # Fortschreibung der Sätze bis jahr_ende
      # keine Änderung der Sätze bekannt, Werte aus letzten bekannten Jahr werden
      ↪  weitergeführt
      bind_rows(tibble(jahr = seq(max(SV_BEITRAGSSATZ$jahr, na.rm = TRUE) + 1, 2070)))
      ↪  %>%
      fill(everything(), .direction = "down") %>%
      # Gesamtsatz, relevant für AG-Anteil
      mutate(sv_vs_ag = ahv_vs_ag + iv_vs_ag + eo_vs_ag + alv_vs_ag)
```

Somit ist die Berechnung des `SV_SATZ` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden kann:

```
#------ Output ------------------------------------------------------------#
mod_return(SV_SATZ)
```

4.32 Modul mod_iv_filter_inp.R

Dokumentation zuletzt aktualisiert am 23.09.2024

`mod_iv_filter_inp.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
tl_iv_input <- mod_iv_filter_inp(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    UMFRAGEN = tl_inp$UMFRAGEN, ESTV_IV = tl_inp$ESTV_IV)
```

Die Funktion verwendet neben `PARAM_GLOBAL` das folgende Dataframe:

- `UMFRAGEN`: Dataframe mit allen Umfrageschätzungen.
- `ESTV_IV`: Dataframe mit allen MWST-Schätzungen der ESTV.

Das Modul `mod_iv_filter_inp.R` extrahiert die gemäss `PARAM_GLOBAL` ausgewählten Umfrageschätzungen für die IV, sowie die in `PARAM_GLOBAL` ausgewählte MWST-Schwätzung der ESTV:

```
#------ Umfragewerte der IV für das Scenario --------------------------#
UMFRAGE_IV <- UMFRAGEN %>%
    filter(umfrageid == PARAM_GLOBAL$id_umfragen, vz == "iv") %>%
    select(jahr, varname, schaetzung) %>%
    pivot_wider(names_from = varname, values_from = schaetzung,
        values_fill = 0)
```

```
if (min(UMFRAGE_IV$jahr) <= PARAM_GLOBAL$jahr_abr) {
    print("Falscher Parameter id_umfragen")
    UMFRAGE_IV <- UMFRAGE_IV %>%
        filter(jahr > PARAM_GLOBAL$jahr_abr)
}



#------ Scenario der Mehrwerteuerschätzung der IV ---------------------#
# Sicherheitsüberprüfung für PARAM_GLOBAL$id_estv_iv Prüfe,
# ob id_estv_iv in ESTV_IV vorhanden ist
if (!PARAM_GLOBAL$id_estv_iv %in% ESTV_IV$idivestv) {
    stop("Die id_estv_iv in PARAM_GLOBAL existiert nicht im File  /iv/go/ESTV_IV.csv in
    ↪  den input-Daten)")
}
ESTV_IV_SCEN <- ESTV_IV %>%
    filter(idivestv == PARAM_GLOBAL$id_estv_iv)
```

Somit ist die Berechnung des `UMFRAGE_IV` sowie des `ESTV_IV_SCEN` Dataframes abgeschlossen, wodurch diese an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden können:

```
#------ Output ------------------------------------------------------------#
mod_return(UMFRAGE_IV, ESTV_IV_SCEN)
```

4.33 Modul mod_iv_fortschreibung.R

Dokumentation zuletzt aktualisiert am 07.10.2024

`mod_iv_fortschreibung.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
  tl_iv_fortschreibung  <- mod_iv_fortschreibung(
     PARAM_GLOBAL       = tl_inp$PARAM_GLOBAL
   , ECKWERTE_EXTENDED  = tl_inp$ECKWERTE_EXTENDED
   , EINK_ENTWICKLUNG   = tl_inp$EINK_ENTWICKLUNG
   , AKTIVE_BEV         = tl_inp$AKTIVE_BEV
   , IK                 = tl_inp$IK
   , IV_ABRECHNUNG      = tl_inp$IV_ABRECHNUNG
   , DISKONTFAKTOR      = tl_inp$DISKONTFAKTOR
   , ESTV_IV_SCEN      = tl_iv_input$ESTV_IV_SCEN
   , RENTENINDEX_ZR      = tl_inp$RENTENINDEX_ZR

  )
```

Die Funktion verwendet neben `PARAM_GLOBAL` das folgende Dataframe:

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.25)
- `AKTIVE_BEV`: Die Wohnbevölkerung gemäss Statistik fortgeschrieben mit den Erwerbsbevölkerungsszenarien des BFS. Es werden ausschliesslich die Wachstumsraten über die Zeit innerhalb der Zellen für die Projektion verwendet werden. Das Dataframe `AKTIVE_BEV` wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `IK`: Die Daten gemäss der individuellen Konten der ZAS. Das Dataframe `IK` wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.

- `EINK_ENTWICKLUNG`: Zeitreihe mit der Entwicklung des Durchschnittslohnes $eink\_entwicklung = \frac{SLI_t^{nominal} * (1 + Sturkturfaktor)}{SLI_{Abrechnungsjahr}^{nominal}}$. Wichtig ist hier der Spezialfall für die ersten zwei Jahre ab dem aktuellen Abrechnungsjahr (also Stand 2024 2024 und 2025). Für diese Periode enthält `eink_entwicklung` zusätzlich noch das Beschäftigungswachstum gemäss der Projektionen der BESTA, oder genauer gesagt die Abweichung des Beschäftigungswachstum gemäss BESTA von dem Erwerbsbevölkerungswachstum gemäss Erwerbsbevölkerungsszenario des BFS. Die genaue Berechnung ist im Modul `mod_eink_entwicklung.R` (vgl. Kapitel 4.26) ersichtlich. Das Dataframe `EINK_ENTWICKLUNG` wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `IV_ABRECHNUNG`: Die Abrechnung wird verwendet, um die durchschnittliche Abweichung der Registerdaten von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren. Die IV-Abrechnung wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `DISKONTFAKTOR`: Da das Modell in realen grössen gerechnet wird, wird der Diskontfaktor benötigt, um nominale grössen vor der Modellberechnung in reale Grössen umzuwandeln, respektive um die realen Modellprognosen am Ende in nominale Grössen umzuwandeln. Der Diskontfaktor wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `ESTV_IV_SCEN`: Die MWST-Projektionen der ESTV, welche für die Projektion der Wachstumsrate des Bundesbeitrags für die ersten Jahre verwendet werden.
- `RENTENINDEX_ZR`: Lohn- und Mischindex für die Projektion der Wachstumsrate des Bundesbeitrages.

Das Modul `mod_iv_fortschreibung.R` berechnet zuerst die Variable `lohnidx`, welche das kummulierte Wachstum des Schweizerischen Lohnindexes (SLI) ab dem letzten Jahr, in welchem die Umfrageprojektionen verwendet werden sollen (`PARAM_GLOBAL$jahr_umfragen_fs - 1`) darstellt, und in allen vorangehenden Jahren 1 beträgt:

```
--- Berechnung der Lohnentwicklung nach dem Umfragejahr -----------#
  FORTSCHREIBUNG_IV  <- ECKWERTE_EXTENDED %>%
    mutate(across(c(-jahr) , list(~if_else(is.na(.), 0, .)), .names = '{.col}' )) %>%
    mutate(lohnidx = cumprod(1 + lohn/100))

  startwert     <- as.numeric(FORTSCHREIBUNG_IV[FORTSCHREIBUNG_IV$jahr ==
   (PARAM_GLOBAL$jahr_umfragen_fs - 1), c('lohnidx')] )

  FORTSCHREIBUNG_IV  <- FORTSCHREIBUNG_IV %>%
    mutate(lohnidx = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs, 1, lohnidx/startwert)
     ↪  )
```

Als nächstes berechnet `mod_iv_fortschreibung.R` die Variable `lohnsummeidx`, welche das kummulierte Wachstum der in der Schweiz ausbezahlten Lohnsumme ab dem letzten Jahr, in welchem die Umfrageprojektionen verwendet werden sollen (`PARAM_GLOBAL$jahr_umfragen_fs - 1`) darstellt, und in allen vorangehenden Jahren 1 beträgt. Ausserdem wird die Variable `lohnsumme` berechnet, welches die jährliche Wachstumsrate der Lohnsumme darstellt:

```
#------ Berechnung für IV: vz setzen ----------------------------------#
PARAM_GLOBAL$vz <- "iv"

tl_iv_beitrag <- mod_beitragssumme(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = AKTIVE_BEV, IK = IK, EINK_ENTWICKLUNG = EINK_ENTWICKLUNG,
    ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR,
    BEITRAGSSATZ = as_tibble(data.frame()))
```

Hierfür wird zuerst das Modul `mod_beitragssumme.R` ausgeführt (vgl. Kapitel 4.18.1), um die Lohnsumme zu berechnen. Danach wird die jährliche Wachstumsrate der Lohnsumme berechnet, und die Variable `lohnsummeidx` gebildet, welche das kummulierte Wachstum der in der Schweiz ausbezahlten Lohnsumme ab dem letzten Jahr, in welchem die Umfrageprojektionen verwendet werden sollen (`PARAM_GLOBAL$jahr_umfragen_fs - 1`) darstellt:

```
FORTSCHREIBUNG_IV <- FORTSCHREIBUNG_IV %>%
    left_join(tl_iv_beitrag$LOHNSUMME) %>%
    mutate(lohnsumme = ifelse(!is.na(ahv_lohnsumme/lag(ahv_lohnsumme)),
        (ahv_lohnsumme/lag(ahv_lohnsumme) - 1) * 100, 0)) %>%
    select(-ahv_lohnsumme) %>%
    mutate(lohnsummeidx = cumprod(1 + lohnsumme/100))


startwert <- as.numeric(FORTSCHREIBUNG_IV[FORTSCHREIBUNG_IV$jahr ==
    (PARAM_GLOBAL$jahr_umfragen_fs - 1), c("lohnsummeidx")])

FORTSCHREIBUNG_IV <- FORTSCHREIBUNG_IV %>%
    mutate(lohnsummeidx = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
        1, lohnsummeidx/startwert))
```

Zum Schluss wird die Wachstumsrate des Bundesbeitrages gemäss Artikel 78 des IVG berechnet:[36]

```
  # Wachstumsfaktor für Bundesbeitrag berechnen:
  FORTSCHREIBUNG_IV      <- FORTSCHREIBUNG_IV %>%
      left_join(RENTENINDEX_ZR %>%
                    # Lohn und Mischindex für den Diskontfaktor
                    select(jahr, lientw,  pientw, lohnindex) %>%
                    mutate(mischidx = lag(lientw) + lag(pientw)) %>%
                    # Wachstumrate gemäss ESTV
                    left_join(ESTV_IV_SCEN %>% select(jahr, mwst_zuwachs), by =
                    ↪  c('jahr'))
      ) %>%
      mutate(mwst_zuwachs = if_else(!is.na(mwst_zuwachs), mwst_zuwachs*100, lohnsumme),
             mwst_diskont = mischidx / lag(mischidx) * lag(lientw, 2) / lag(lientw),
             bundesbeitrag = ((1+mwst_zuwachs/100)*mwst_diskont-1)*100) %>%
      select(-lientw,  -pientw, -lohnindex, -mischidx, -mwst_zuwachs, -mwst_diskont)
```

Hierzu wird das Dataframe `RENTENINDEX_ZR` (vgl. Kapitel 4.29) verwendet, welches die gemäss IVG relevanten Indexwerte für die Lohn- und Preisentwicklung enthält. Zudem wird die MWST-Projektion der ESTV verwendet, welche im Dataframe `ESTV_IV_SCEN` enthalten ist. Die restlichen Berechnungen stellen sicher, dass die Wachstumsrate des Bundesbeitrages, welcher in der Variable `bundesbeitrag` enthalten ist, der in Artikel 78 des IVG spezifizierten Formel entspricht. Eine genauere Erläuterung dieser Formel findet sich in der nicht-technischen Dokumentation zum Finanzperspektivenmodell IV.

Somit ist die Berechnung des `FORTSCHREIBUNG_IV` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden kann:

```
#------ Output ------------------------------------------------------#
mod_return(SV_SATZ)
```

[36] Dieser Codeteil wurde auf Empfehlung des Expertenberichts „Finanzperspektiven der IV: Modellanalyse“ angepasst.