Finanzperspektiven der IV

Technische Dokumentation

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

20.08.2025

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 einzulesenden Rohdaten in einer zentralen Datei ersichtlich sind, und andererseits, dass Änderungen an den einzulesenden Rohdaten einfach, d.h. ohne Anpassungen am Programmcode, vorgenommen werden können. Nachfolgend ein Auszug aus `PARAM_INPUTS.csv` (Stand November 2024) als Beispiel:

key	value
file_iv_abrechnung_fin	sv_iv_fin.xlsx
sheet_iv_abrechnung_def	daten
sheet_iv_abrechnung_prov	daten_prov

Wir sehen in der Tabelle die Angabe, in welcher Datei sich die IV-Abrechnung befindet, sowie die Angabe, 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 August 2025) als Beispiel:

key	value
jahr_ende	2070
MA_years	2

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 zwei Jahre für die Berechnung des gleitenden Durchschnitts bei der Invalidisierungsrate und der Mutationsrate im Rentenmodell verwendet werden sollen (der Modellteil, in welchem diese Parameter verwendet werden, ist in Kapitel 4.16 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 4.12 beschrieben). Diese werden in der Datei `PARAM_MASSNAHMEN_BASE.csv` spezifiziert. Diese Datei sieht Stand August 2025 wie folgt aus:

key	value
aktivierte_massnahmen_int	iv_tabellenlohn_ac_rentiers , iv_tabellenlohn_nv_rentiers, iv_fruh_int_autismus, iv_re
aktivierte_massnahmen_ext	
mass_f_1	iv_fruh_int_autismus, iv_tabellenlohn_ac_rentiers, iv_tabellenlohn_nv_rentiers
mass_f_2	iv_rechnungslegung_25
group1	
group2	

Wir sehen in der Tabelle in der ersten Zeile `aktivierte_massnahmen_int` die Politikmassnahmen, wessen Auswirkung im IV Finanzperspektivenmodell geschätzt werden soll. Die Zeile `aktivierte_massnahmen_ext` enthält allfällige Massnahmen, für welche ein vorberechneter Vektor mit jährlichen Auswirkungend der Massnahmen eingelesen werden soll. 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 18.03.2025

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

2.1 Daten zu Wohn- und Erwerbsbevölkerung, und Grenzgänger

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. Zusätzlich nutzen wir Daten zur Erwerbsbevölkerung nach Alter und Geschlecht (in Anzahl Personen und in Vollzeitäquivalenten). Diese Daten berechnen wir, anhand des Vorgehens des BFS, direkt aus den Rohdaten der Schweizerischen Arbeitskräfteerhebung (SAKE).[5] Ebenfalls nutzen wir die Daten zu den Grenzgängern nach Alter und Geschlecht aus der Grenzgängerstatistik.[6]

Unsere Projektionen basieren auf den durch das BFS in den Wohnbevölkerungsszenarien[7], den Szenarien der Erwerbsbevölkerung in Vollzeitäquivalenten[8], sowie den Grenzgängerszenarien projizierten Entwicklungen. Diese Szenarien werden alle fünf Jahre aktualisiert, wobei Stand 2024 die letzte Aktualisierung 2020 stattgefunden hat.

Die Bestandes- und Szenariodaten zur Wohn- und Erwerbsbevlölkerung basieren auf dem Inländer-Konzept. Das heisst, dass sie die Entwicklung der ständigen Wohnbevölkerung respektive die Erwerbsbevölkerung in Vollzeitäquivalenten innerhalb der ständigen Wohnbevölkerung abdecken. Um die Population der Erwerbsbevölkerung nach dem Inland-Konzept (also die Population der in der Schweiz Erwerbstätigen zuzüglich der inländischen Erwerbslosen) zu erhalten, welche für die Projektion der Entwicklung der Lohnbeiträge relevant

[3] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/statpop.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/arbeit-erwerb/erhebungen/sake.html
[6] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erhebungen/ggs.html
[7] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/szenarien.html
[8] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erwerbstaetigkeit-arbeitszeit/erwerbsbevoelkerung/kuenftige-entwicklung-erwerbsbevoelkerung.html

ist, kombinieren wir die Erwerbsbevölkerung nach dem Inländer-Konzept mit den Grenzgängern. Um die Versichertenpopulation im Inland zu erhalten, welche für die Projektion der Ausgaben für Renten und weitere Leistungen der IV relevant ist, kombinieren wir die Wohnbevölkerung nach dem Inländer-Konzept mit den Grenzgängern. Wir basieren uns im Finanzperspektivenmodell durchgehend auf das Referenzszenario für die Wohnbevölkerung, die Erwerbsbevölkerung in Vollzeitäquivalenten, und die Grenzgänger.

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[9]. Für die Preisentwicklung basieren wir uns auf den Landesindex der Konsumentenpreise (LIK) mit Basis 1977=100[10].[11]

Unsere Projektionen für die Lohn- und Preisentwicklung gemäss SLI respektive LIK basieren auf den durch die Eidgenössische Finanzverwaltung (EFV) projizierten Eckwerte für die Finanzplanung.[12] 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 zur Entwicklung der Beschäftigung und der Arbeitslosenquote

Zur Abschätzung der Entwicklung der Einnahmen aus Beiträgen von Versicherten und Arbeitnehmern (Lohnbeiträge) in der kurzen Frist (vgl. Kapitel 4.32.1) verwenden wir das projizierte Beschäftigungswachstum gemäss BESTA, sowie die projizierte Arbeitslosenquote (gemäss SECO-Definition und nicht gemäss ILO-Definition) aus den durch die Eidgenössische Finanzverwaltung (EFV) projizierten Eckwerte für die Finanzplanung. Zusätzlich nutzen wir die historische Arbeitslosenquote gemäss SECO-Definition.[13]

2.4 Daten aus den zentralen Registern der 1. Säule

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

2.4.1 Daten aus den individuellen Konten (AHV-IK)

Die Daten der Individuelle Konten (IK) enthalten individuelle Daten zur Erwerbstätigkeit jeder in der Schweiz beitragspflichtigen Person.[14] 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

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

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

[11] Die Verwendung der Basisjahre folgt den gesetztlichen Anforderungen für die Berechnung des Mischindexes für die Bestimmung der Minimalrenten. Grundsätzlich verwenden wir das Jahresmittel des LIK zur Bestimmung der Teuerung. Eine Ausnahme bildet die Berechnung der Minimalrentenentwicklung, bei welcher wir entsprechend der Praxis in der Vergangenheit bis ins Jahr 2017 den LIK Stand Dezmeber (respektive die jährliche Veränderung des LIK Stand Dezember) für die Bestimmung der für die Entwicklung der Minimalrente massgeblichen Teuerung nutzen, und ab 2017 das Jahresmittel des LIK, respektive die jährliche Verändungsrate des Jahresmittels des LIK.

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

[13] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erwerbslosigkeit-unterbeschaeftigung/registrierte-arbeitslose-seco.html

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

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.

2.4.2 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.

2.4.3 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.

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.

2.5 Daten aus der IV Abrechnung

Wir verwenden die Abrechnungsdaten der IV gemäss der von Compenswiss publizierten Jahresrechnung (Bilanz und der Erfolgsrechnung der IV).[15]

2.6 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

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

Erhöhung gibt, oder bei einer Änderung der Bemessensgrundlage (Einführung einer neuen Steuerausnahme oder Aufhebung einer Steuerausnahme).[16]

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.

2.7 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 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).[17]

3.1 Modul prepare_input.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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

[16] 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.

[17] 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 hier nicht beschrieben.

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

`path` 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:

```
# Parameter-File einlesen
PARAM_INPUTS <- read_param(path)
```

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

```
# Pfade zu Input-Daten definieren
PARAM_INPUTS$path_allgemein_go <- paste0(PARAM_INPUTS$raw_data,
    "/allgemein/go/")
PARAM_INPUTS$path_allgemein_massnahmen <- paste0(PARAM_INPUTS$raw_data,
    "/allgemein/massnahmen/")

PARAM_INPUTS$path_ahv_go <- paste0(PARAM_INPUTS$raw_data, "/ahv/go/")
PARAM_INPUTS$path_ahv_massnahmen <- paste0(PARAM_INPUTS$raw_data,
    "/ahv/massnahmen/")

PARAM_INPUTS$path_iv_go <- paste0(PARAM_INPUTS$raw_data, "/iv/go/")
PARAM_INPUTS$path_iv_massnahmen <- paste0(PARAM_INPUTS$raw_data,
    "/iv/massnahmen/")

PARAM_INPUTS$path_eo_go <- paste0(PARAM_INPUTS$raw_data, "/eo/go/")
PARAM_INPUTS$path_eo_massnahmen <- paste0(PARAM_INPUTS$raw_data,
    "/eo/massnahmen/")

PARAM_INPUTS$path_el_go <- paste0(PARAM_INPUTS$raw_data, "/el/go/")
PARAM_INPUTS$path_el_massnahmen <- paste0(PARAM_INPUTS$raw_data,
    "/el/massnahmen/")

PARAM_INPUTS$path_ul_go <- paste0(PARAM_INPUTS$raw_data, "/ul/go/")
PARAM_INPUTS$path_ul_massnahmen <- paste0(PARAM_INPUTS$raw_data,
```

```
    "/ul/massnahmen/")

PARAM_INPUTS$path_rententab <- paste0(PARAM_INPUTS$raw_data,
    "/rententab")

PARAM_INPUTS$path_beitragstab <- paste0(PARAM_INPUTS$raw_data,
    "/beitragstab")

PARAM_INPUTS$path_eotab <- paste0(PARAM_INPUTS$raw_data, "/eotab")
```

Es wird später für jede Sozialversicherung ein separater Ordner für die aufbereiteten Input-Daten erstellt. `path_iv_go` enthält beispielsweise den Pfad zum Ordner, in welchem Input-Daten für die Berechnungen der IV-Finanzperspektiven später abgespeichert werden sollen, und `path_iv_massnahmen` den Pfad zum Ordner, in welchem die Input-Daten für die Berechnung der Politikmassnahmen für IV AHV abgespeichert werden sollen. Es gibt Input-Daten, welche für mehrere Finanzperspektivenmodelle verwendet werden, beispielsweise die Bevölkerungszahlen und -szenarien, oder die Projektionen zur Lohn- und Preisentwicklung. Um Duplikate zu vermeiden werden diese Input-Daten ausschliesslich im Ordner `path_allgemein_go` abegelegt.

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_allgemein_go)
ensure_path(inp_path_allgemein_massnahmen)

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_el_go)
ensure_path(inp_path_el_massnahmen)

ensure_path(inp_path_ul_go)
ensure_path(inp_path_ul_massnahmen)

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 anhand der IV-spezifischen Rohdaten erläutert:

```
### IV: GELTENDE ORDNUNG ###

tl_iv_abrechnung <- mod_input_iv_abrechnung(PARAM_INPUTS = PARAM_INPUTS)
tl_ivrenten <- mod_input_ivrenten(PARAM_INPUTS)
UMFRAGEN <- mod_input_umfragen(PARAM_INPUTS)
ESTV_IV <- mod_input_estv_iv(PARAM_INPUTS)
tl_iv_med_massnahmen <- mod_input_iv_med_massnahmen(PARAM_INPUTS)

iv_go <- c(tl_iv_abrechnung, tl_ivrenten, tl_iv_med_massnahmen,
    list(UMFRAGEN = UMFRAGEN, ESTV_IV = ESTV_IV))

### IV: MASSNAHMEN ###

iv_massnahmen <- mod_input_massnahmen_iv(PARAM_INPUTS)
```

D.h. es werden nacheinander die verschiedenen für die Berechnung der IV-Finanzperspektiven relevanten Rohdaten eingelesen, wobei für jede Art von Rohdaten ein spezifisches Modul verwendet wird. Die IV-spezifischen Rohdaten für die Berechnung der IV-Finanzperspektiven nach geltender Ordnung werden danach in der Liste `iv_go` zusammengefasst. Ebenso werden die IV-spezifischen Rohdaten für die Berechnung der Massnahmen in der Liste `iv_massnahmen` zusammengefasst. Die einzelnen Module für das Einlesen der Rohdaten werden in den nachfolgenden Abschnitten dieses Kapitels erläutert.

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:

```
tidylist_write(iv_go, inp_path_iv_go)
tidylist_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_bev_bestand.R

Dokumentation zuletzt aktualisiert am 31.05.2025

Das Modul `mod_input_bev_bestand.R` liest die Daten für die Bevölkerungsbestände des BFS ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
BEV_BESTAND <- mod_input_bev_bestand(PARAM_INPUTS = PARAM_INPUTS)
```

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

```
    #---------------------------------LOAD RData FILES (ESPOP, ERWBEV)-

    # Funktion zum Einlesen der RData-Dateien
    loadRData <- function(fileName) {
        temp_env <- new.env()
        load(fileName, envir = temp_env)
        temp_env[[ls(temp_env)[1]]]
    }

    # Lade die benötigten RData-Dateien
    ESPOP <- loadRData(file.path(PARAM_INPUTS$path_allgemein_go, "BEVOELKERUNG",
↪   PARAM_INPUTS$name_espop)) |>
      pivot_wider(
        names_from = variable,
        values_from = value
      ) |>
      arrange(jahr, sex, nat, alt)

    BEV_INLAENDER_EPT <- loadRData(file.path(PARAM_INPUTS$path_allgemein_go,
↪   "BEVOELKERUNG", PARAM_INPUTS$erwbev_bestand)) %>%
      mutate( # Berechnung von Erwerbsquote und Erwerbsquote in VZÄ, da diese auch für
       ↪  die Szenariodaten verfügbar sind
        erwq=erwbev/erwbev_and_nerwbev*100,
        erwqept=ept/erwbev_and_nerwbev*100
        ) %>%
      select(-erwbev_and_nerwbev) # Variable mit Summe von Erwerbs- und
       ↪  Nichterwerbsbevölkerung gemäss SAKE entfernen (da nicht in Szenariodaten
       ↪  enthalten)
```

Die Funktion `loadRData` stellt sicher, dass die Daten im R-Format korrekt eingelesen werden. Darauffolgend werden nacheinander die Daten zu Beständen und Flüssen (bspw. Ein- und Auswanderung, Geburten, Todesfälle) der Wohnbevölkerung (Dataframe ESPOP) sowie zum Bestand der Inländer-Erwerbsbevölkerung (BEV_INLAENDER_EPT) eingelesen.[18]

Als nächstes werden die Daten zu den Grenzgänger-Beständen eingelesen, wobei die Komplexität des Codes einzig daher rührt, dass die Rohdaten aus dem von uns gelieferten .xlsx-Format eingelesen werden müssen:

```
#---------------------------------FRONTALIERS DATA--------------------------

# Funktion zum Einlesen der Sheets 'Männer' und 'Frauen'
process_sheet <- function(sheet_name, sex_label) {
    # Datei einlesen mit automatischer Vergabe eindeutiger
    # Spaltennamen
    data <- read_excel(file.path(PARAM_INPUTS$path_allgemein_go,
        "BEVOELKERUNG", PARAM_INPUTS$file_frontaliers), sheet = sheet_name,
        col_names = FALSE, .name_repair = "unique_quiet")

    # Spalte B entfernen (zweite Spalte)
    data <- data |>
        select(-2)
```

[18] Bis 2010 wurden die Bestände und Flüsse der Wohnbevölkerung in der Synthesestatistik von Stand und Struktur der Bevölkerung (ESOPOP) abgebildet, ab 2010 in der Statistik der Bevölkerung und der Haushalte (STATPOP). Das BFS stellt die Daten aus den beiden Statistiken kombiniert bereit.

```
    # Spaltennamen setzen basierend auf der zweiten Zeile
    colnames(data) <- c("jahr", as.character(as.numeric(data[2,
        -c(1, ncol(data))])), "99")

    # Entferne erste und dritte Zeile
    data <- data[-c(1, 2, 3), ]

    # In tibble umwandeln und auf lange Form bringen
    data |>
        as_tibble() |>
        pivot_longer(cols = -jahr, names_to = "alt", values_to = "frontaliers") |>
        mutate(jahr = as.numeric(jahr), frontaliers = as.numeric(frontaliers),
            alt = as.numeric(alt), sex = sex_label, nat = "au")
}

# Daten aus beiden Sheets einlesen
FRONTALIERS <- bind_rows(process_sheet("Männer", "m"), process_sheet("Frauen",
    "f"))
```

Als nächstes werden die Daten zu den Saisonnier-Beständen ab 1971 eingelesen, wobei die Komplexität des Codes wiederum einzig daher rührt, dass die Rohdaten aus dem von uns gelieferten .xlsx-Format eingelesen werden müssen:

```
    #---------------------------------SAISONNIERS DATA--------------------------

    # Lade und verarbeite die Saisonniers-Daten
    SAISONNIERS <- read_excel(
        file.path(PARAM_INPUTS$path_allgemein_go, "BEVOELKERUNG", PARAM_INPUTS$file_sm),
        sheet = "bestaende",
        skip = 10
    ) %>%
        as_tibble() %>%
        pivot_longer(
            cols = -jahr, # Alle Spalten außer "jahr"
            names_to = "alt", # Alters-Spalte erstellen
            values_to = "saisonniers" # Werte in "saisonniers" speichern
        ) %>%
        mutate(
            alt = as.integer(gsub("age", "", alt, fixed = TRUE)), # "age" aus
            ↪  Alterswerten entfernen
            jahr = as.integer(jahr), # Jahr numerisch formatieren
            nat = "au" # Nationalität hinzufügen
        ) %>%
        crossing(sex = c("m", "f")) %>% # Geschlechter-Kombination
        filter(jahr >= 1971, alt <= 99) # Filtere relevante Jahre und Alter
```

Als nächstes werden die Daten zu den Beständen an freiwillig Versicherten eingelesen, wobei die Komplexität des Codes wiederum einzig daher rührt, dass die Rohdaten aus dem von uns gelieferten .xlsx-Format eingelesen werden müssen:

```r
    #--------------------------------ASSURES FACULTATIFS DATA-----------------

    # Importiere die Assures-Facultatifs-Daten
    tfile <- file.path(PARAM_INPUTS$path_allgemein_go, "BEVOELKERUNG",
↪   PARAM_INPUTS$file_af)

    # Bestimme relevante Sheets (Frauen und Männer)
    sheets_names_sel <- readxl::excel_sheets(tfile) %>%
        grep("^(f_|h_)", ., value = TRUE)

    # Funktion zum Lesen und Umformen von Excel-Daten
    af_fct_data <- function(tfile, sheet) {
        suppressMessages( # Suppress "New names" message
            read_excel(tfile, sheet, skip = 1) %>%
                as_tibble() %>%
                select(-TOTAL) %>%  # Entferne TOTAL-Spalte
                rename(jahr = "...1") %>%  # Benenne erste Spalte um
                pivot_longer(-jahr, names_to = "alt", values_to = "assures_facultatifs")
                ↪  %>%
                mutate(
                    alt = as.integer(alt),  # Wandle alt in Integer um
                    jahr = as.integer(jahr)  # Wandle jahr in Integer um
                ) %>%
                filter(alt <= 99) %>%  # Filtere Alterswerte bis 99
                arrange(alt)  # Sortiere nach Alter
        )
    }

    # Verarbeite und kombiniere die Assures-Facultatifs-Daten
    ASSURES_FACULTATIFS <- tibble(sheet = sheets_names_sel) %>%
        mutate(
            sex = ifelse(grepl("^f_", sheet), "f", "m"),  # Geschlecht aus Sheetname
            ↪  ableiten
            nat = ifelse(grepl("_ch", sheet), "ch", "au")  # Nationalität aus Sheetname
            ↪  ableiten
        ) %>%
        mutate(data = purrr::map(sheet, ~ af_fct_data(tfile, .x))) %>%  # Daten einlesen
        unnest(data) %>%  # Daten entpacken
        select(jahr, sex, nat, alt, assures_facultatifs)  # Spalten auswählen
```

Der nachfolgende Codeblock stellt sicher, dass für alle Variable (Variablen sind beispielsweise Grenzgänger, Saisonniers, Wohnbevölkerung), welche für ein gegebenes Jahr vorhanden sind, Daten für jedes Alter zwischen 0 und 99, Geschlecht, und jede Nationalität (Schweizer oder Ausländer) vorhanden sind.[19] Wenn für ein Jahr, für welches Daten für die gegebene Variable vorhanden sind, in einer Alter-Geschlecht-Nationalität Zelle keine Werte geliefert wurden, wird dieser Wert auf 0 gesetzt. Dies stellt später sicher, dass später nicht-vorhandene Daten von 0-Werten unterscheiden werden können. Zusätzlich werden in nachfolgendem Codeblock auch alle Alter über 99 aggregiert und der Altersklasse 99 zugerechnet.

[19] Die Unterscheidung der Nationalitäten ist für die EL nicht relevant, wird aber hier trotzdem gemacht, da diese für die AHV relevant ist und die Datenaufbereitung für alle Sozialversicherungen gemeinsam gemacht wird. Für die EL werden dann später im Code Schweizer und Ausländer aggregiert, womit die Nationalitätsunterscheidung entfällt.

```r
#--------------------------------ENSURE ALT RANGE 0-99---------------------

# Funktion zur Sicherstellung vollständiger Daten für alt von 0 bis 99 für jahre wo
↪  Daten vorhanden sind
ensure_alt_range <- function(df, value_cols) {
    # Ensure value_cols is treated as a vector
    value_cols <- as.vector(value_cols)

    # Create a complete grid with all combinations of jahr, sex, nat, and alt
    complete_grid <- expand_grid(
        jahr = unique(df$jahr),  # All unique years in the dataset
        sex = c("m", "f"),       # Both sexes
        nat = c("ch", "au"),     # Both nationalities
        alt = 0:99               # Age group from 0 to 99
    )

    # Summarize and complete the data
    df %>%
        mutate(alt = ifelse(alt >= 100, 99, alt)) %>% # Summiere alt >= 100 in alt ==
        ↪  99
        group_by(jahr, sex, nat, alt) %>%
        summarize(across(all_of(value_cols), \(x) sum(x, na.rm = TRUE)), .groups =
        ↪  "drop") %>% # Summiere Werte
        right_join(complete_grid, by = c("jahr", "sex", "nat", "alt")) %>% # Merge
        ↪  with complete grid
        mutate(across(all_of(value_cols), ~ replace_na(., 0))) # Replace NA with 0 in
        ↪  value_cols
}


# Wende die Normalisierung auf alle relevanten DataFrames an
ESPOP <- ensure_alt_range(ESPOP,
↪  c("bevanfangj","bevendejahr","geburt","tod","einbuergerung","einwanderung","auswanderung","bereinig
BEV_INLAENDER_EPT <- ensure_alt_range(BEV_INLAENDER_EPT, c("erwbev","ept", "erwq",
↪  "erwqept"))
FRONTALIERS <- ensure_alt_range(FRONTALIERS, "frontaliers")
SAISONNIERS <- ensure_alt_range(SAISONNIERS, "saisonniers")
ASSURES_FACULTATIFS <- ensure_alt_range(ASSURES_FACULTATIFS, "assures_facultatifs")
```

Der Nachfolgende Codeblock dient dazu, Fehler im Register für die freiwillig Versicherten zu korrigieren. Konkret fehlen in den Jahren 2002 und 2004 Beobachtungen im Register. Daher werden die Werte für 2002 und 2004 aus den aufgerundeten Mittelwerten der Werte in der jeweiligen Geschlecht-Nationalität-Alter Zelle der angrenzenden Jahre (bspw. 2001 und 2003 für 2002) gebildet:

```r
#---------------KORREKTUR REGISTERFEHLER ASSURES_FACULTATIFS----------------
# Correction des erreurs d'observations des registres pour
# les années 2002 et 2004 (non reported observations)

ASSURES_FACULTATIFS <- ASSURES_FACULTATIFS %>%
    group_by(sex, nat, alt) %>%
    # Pour 2002 et 2004, remplacer les valeurs par la
    # moyenne des observations de l'année précédente et
    # l'année suivante.
```

```
mutate(assures_facultatifs = if_else(jahr %in% c(2002, 2004),
    ceiling((lag(assures_facultatifs) + lead(assures_facultatifs))/2),
    assures_facultatifs)) %>%
    ungroup()
```

Zum Schluss werden alle Daten in einem Dataframe `BEV_BESTAND` zusammengefasst. Hierbei wird für die Jahre bis 2009 ESPOP für die Wohnbevölekrung und die Auswanderung verwendet, und ab 2010 STATPOP:

```
    #---------------------------------COMBINE DATA-----------------------------

    # Kombiniere die verschiedenen Datenquellen
    BEV_BESTAND <- ESPOP %>%
      full_join(BEV_INLAENDER_EPT, by = c("jahr", "sex", "nat", "alt")) %>%
      full_join(FRONTALIERS, by = c("jahr", "sex", "nat", "alt")) %>%
      left_join(SAISONNIERS, by = c("jahr", "sex", "nat", "alt")) %>% # Left-join für
      ↪  Saisonniers, da diese in Rohdaten bis 2065 auf 0 sind
      full_join(ASSURES_FACULTATIFS, by = c("jahr", "sex", "nat", "alt")) %>%
      arrange(jahr, sex, nat, alt)

    return(BEV_BESTAND = BEV_BESTAND)
```

3.3 Modul mod_input_bev_scenario.R

Dokumentation zuletzt aktualisiert am 31.05.2025

Das Modul `mod_input_bev_scenario.R` liest, analog zum Modul `mod_input_bev_bestand.R` für die Bevölkerungsbestände (vgl. Kapitel 3.2), die Daten für die Bevölkerungsszenarien des BFS ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
BEV_SCENARIO <- mod_input_bev_scenario(PARAM_INPUTS = PARAM_INPUTS)
```

Als erstes werden in`mod_input_bev_scenario.R` die verschiedenen Rohdateien mit den Bevölkerungsszenarien eingelesen:

```
    #---------------------------------LOAD RData FILES (ESPOP, STATPOP, ERWBEV)-

    # Funktion zum Einlesen der RData-Dateien
    loadRData <- function(fileName) {
        temp_env <- new.env()
        load(fileName, envir = temp_env)
        temp_env[[ls(temp_env)[1]]]
    }

    # Load Scenarios population
    SCENARIO_POP <- loadRData(file.path(
      PARAM_INPUTS$path_allgemein_go,
      "BEVOELKERUNG",
      PARAM_INPUTS$name_scenario_pop
    )) %>%
      filter(str_detect(scenario, "^(A-00|B-00|C-00)")) %>%
      mutate(scenario = str_replace_all(scenario, "-", "_")) %>% # Replace - with _
```

```
    select(-geburtnachnat)

  # Load Scenarios ept
  SCENARIO_EPT <- loadRData(file.path(
    PARAM_INPUTS$path_allgemein_go,
    "BEVOELKERUNG",
    PARAM_INPUTS$name_scenario_ept
  )) %>%
    filter(str_detect(scenario, "^(A-00|B-00|C-00)")) %>%
    mutate(scenario = str_replace_all(scenario, "-", "_")) %>% # Replace - with _
    rename(ept = erwbevept) %>%
    select(alt, jahr, nat, sex, scenario, erwbev, ept)
```

Die Funktion `loadRData` stellt sicher, dass die Daten im R-Format korrekt eingelesen werden. Darauffolgend werden nacheinander die Daten den Bevölkerungsszenarien (SCENARIO_POP)[20], sowie die Erwerbsbevölkerungsszenarien (SCENARIO_EPT)[21] eingelesen.

Als nächstes werden die Daten zu den Grenzgänger-Szenarien eingelesen:

```
SCENARIO_FRONTALIERS <- loadRData(file.path(PARAM_INPUTS$path_allgemein_go,
    "BEVOELKERUNG", PARAM_INPUTS$name_scenario_frontaliers)) %>%
    filter(str_detect(scenario, "^(A_00|B_00|C_00)"))
```

Der nachfolgende Codeblock stellt sicher, dass für alle Variable (Variablen sind beispielsweise Grenzgänger, Erwerbsbevölkerung, Wohnbevölkerung), welche für ein gegebenes Jahr vorhanden sind, Daten für jedes Alter zwischen 0 und 120, Geschlecht, und jede Nationalität (Schweizer oder Ausländer) vorhanden sind.[22] Wenn für ein Jahr, für welches Daten für die gegebene Variable vorhanden sind, in einer Alter-Geschlecht-Nationalität Zelle keine Werte geliefert wurden, wird dieser Wert auf 0 gesetzt. Dies stellt später sicher, dass später nicht-vorhandene Daten von 0-Werten unterscheiden werden können. Zusätzlich werden in nachfolgendem Codeblock auch alle Alter über 120 aggregiert und der Altersklasse 120 zugerechnet.

```
    #--------------------------------ENSURE ALT RANGE 0-120---------------------

    # Funktion zur Sicherstellung vollständiger Daten für alt von 0 bis 120 für jahre wo
    ↪  Daten vorhanden sind
    ensure_alt_range <- function(df, value_cols) {

      # Create a complete grid with all combinations of existing jahr, sex, nat, alt, and
      ↪  scenario
      complete_grid <- df %>%
        select(jahr, scenario) %>%
        distinct() %>%
        expand_grid(
          sex = c("m", "f"),
          nat = c("ch", "au"),
          alt = 0:120
        )
```

[20] https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/erhebungen/szenarien.html

[21] https://www.bfs.admin.ch/bfs/de/home/statistiken/arbeit-erwerb/erwerbstaetigkeit-arbeitszeit/erwerbsbevoelkerung/kuenftige-entwicklung-erwerbsbevoelkerung.html

[22] Die Unterscheidung der Nationalitäten ist für die IV nicht relevant, wird aber hier trotzdem gemacht, da diese für die AHV relevant ist und die Datenaufbereitung für alle Sozialversicherungen gemeinsam gemacht wird. Für die IV werden dann später im Code Schweizer und Ausländer aggregiert, womit die Nationalitätsunterscheidung entfällt.

```
    # Summarize, complete, and drop rows with all value_cols as NA
    df %>%
      mutate(alt = ifelse(alt >= 121, 120, alt)) %>%  # Gruppiere alt >= 121 zu 120
      group_by(jahr, sex, nat, alt, scenario) %>%
      summarize(across(all_of(value_cols), \(x) sum(x, na.rm = TRUE)), .groups =
      ↪  "drop") %>%
      right_join(complete_grid, by = c("jahr", "scenario", "sex", "nat", "alt")) %>%  #
      ↪  Ergänze fehlende Kombinationen innerhalb jedes scenario
      mutate(across(all_of(value_cols), \(x) replace_na(x, 0)))  # Ersetze NA-Werte
      ↪  durch 0
  }


  # Wende die Normalisierung auf alle relevanten DataFrames an
  SCENARIO_POP <- ensure_alt_range(SCENARIO_POP,
↪ c("bevanfangj","bevendejahr","geburt","tod","einbuergerung","einwanderung","auswanderung","geburtmu
  SCENARIO_EPT <- ensure_alt_range(SCENARIO_EPT, c("erwbev","ept","erwq","erwqept"))
  SCENARIO_FRONTALIERS <- ensure_alt_range(SCENARIO_FRONTALIERS, "frontaliers")
```

Zum Schluss werden alle Daten in einem Dataframe `SCENARIO_POP` zusammengefasst:

```
#--------------------------------COMBINE DATA-------------------------------

# Kombiniere die verschiedenen Datenquellen
BEV_SCENARIO <- SCENARIO_POP %>%
    full_join(SCENARIO_EPT, by = c("jahr", "sex", "nat", "alt",
        "scenario")) %>%
    full_join(SCENARIO_FRONTALIERS, by = c("jahr", "sex", "nat",
        "alt", "scenario")) %>%
    arrange(jahr, sex, nat, alt, scenario)

return(BEV_SCENARIO = BEV_SCENARIO)
```

3.4 Modul mod_input_ikregister.R

Dokumentation zuletzt aktualisiert am 18.11.2024

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:

```
IK <- 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) {
    ik_file <- paste0(PARAM_INPUTS$path_allgemein_go, "IK/ik_",
        l_j, PARAM_INPUTS$file_ik)
    z <- read.csv(ik_file, 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 die CSV-Datei mit den IK-Daten dieses Jahres einliest. 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:

```
return(IK = IK)
```

3.5 Modul mod_input_sv_beitragssatz.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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:

```
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:

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

3.6 Modul mod_input_indices.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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:

```
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$path_allgemein_go,
    PARAM_INPUTS$file_indices), sheet = PARAM_INPUTS$sheet_indices,
    range = readxl::cell_cols(1:8))

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

return(INDICES = INDICES)
```

3.7 Modul mod_input_eckwerte.R

Dokumentation zuletzt aktualisiert am 18.11.2024

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:

```
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$path_allgemein_go,
    PARAM_INPUTS$file_eckwerte), sheet = PARAM_INPUTS$sheet_eckwerte,
    range = readxl::cell_cols(1:12))

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

return(ECKWERTE = ECKWERTE)
```

3.8 Modul mod_input_minimalrente.R

Dokumentation zuletzt aktualisiert am 14.01.2025

Das Modul `mod_input_minimalrente.R` liest die historische Zeitreihe zur Höhe der Minimalrente ein. Es wird wie folgt in `prepare_input.R` aufgerufen:

```
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 Minimalrenten
# ------------------------------------------------#

MINIMALRENTE <- read_excel(paste0(PARAM_INPUTS$path_allgemein_go,
    PARAM_INPUTS$file_minimalrente), sheet = PARAM_INPUTS$sheet_minimalrente,
    range = readxl::cell_limits(c(11, 1), c(NA, NA))) %>%
    mutate(minimalrente = r_min/12) %>%
    select(jahr, minimalrente)

# Return tidylist mit Minimalrenten
# ----------------------------------------#

return(MINIMALRENTE = MINIMALRENTE)
```

3.9 Modul mod_input_iv_abrechnung.R

Dokumentation zuletzt aktualisiert am 25.11.2024

Das Modul `mod_input_iv_abrechnung.R` liest die IV-Abrechnung ein.[23] Es wird wie folgt in `prepare_input.R` aufgerufen:

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

```
tl_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$path_iv_go,
    PARAM_INPUTS$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 %>%
    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$path_iv_go,
    PARAM_INPUTS$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$path_iv_go,
    PARAM_INPUTS$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
# ------------------------------------------------------------------

return(list(IV_ABRECHNUNG_DEF = IV_ABRECHNUNG_DEF, IV_ABRECHNUNG_PROV =
↪   IV_ABRECHNUNG_PROV))
```

3.10 Modul mod_input_ivrenten.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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 <- read_excel(path = paste0(PARAM_INPUTS$path_iv_go,
    PARAM_INPUTS$file_iv_renten), sheet = PARAM_INPUTS$sheet_iv_renten,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:10)
) %>%
    select(-pop, -cinf_grp_bez, -cinf_grp, -sex_bez, -wohnort_ch,
        -wohnort_ch_bez) %>%
    rename(alt = alter) %>%
    mutate_if(is.numeric, ~coalesce(., 0))
```

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 <- IV_RENTEN %>%
    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 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_ERWACHSENE <- 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_ERWACHSENE <- IV_HE_ERWACHSENE %>%
    # Alle Kombinationen von jahr, sex, alt erstellen
complete(jahr = unique(IV_HE_ERWACHSENE$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)
```

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

```
#------ Output --------------------------------------------------------#
return(list(IV_RENTEN = IV_RENTEN, IV_HE_ERWACHSENE = IV_HE_ERWACHSENE,
    IV_HE_KINDER = IV_HE_KINDER))
```

3.11 Modul mod_input_umfragen.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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:

```
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$path_iv_go,
    PARAM_INPUTS$file_umfragen), sheet = PARAM_INPUTS$sheet_umfragen,
    range = readxl::cell_limits(c(1, 1), c(NA, NA))  # cell_cols(1:8)
)
#------ Output ----------------------------------------------------------#
return(UMFRAGEN = UMFRAGEN)
```

3.12 Modul mod_input_estv_iv.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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 **??** importiert wird. `mod_input_estv.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
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$path_iv_go,
    PARAM_INPUTS$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 ----------------------------------------------------------#
return(ESTV_IV = ESTV_IV)
```

3.13 Modul mod_input_iv_med_massnahmen.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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 4.19.2 und 4.19.6) 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.

3.13.1 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$path_iv_go,
                                 PARAM_INPUTS$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$path_iv_go,
                PARAM_INPUTS$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.

3.13.2 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 4.19.2):

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

# Excel-Datei einlesen
suppressMessages({
    IV_MED_MASSN_SONDEREFFEKTE <- read_excel(path = paste0(PARAM_INPUTS$path_iv_go,
        PARAM_INPUTS$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)
```

3.13.3 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 4.19.6):

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

3.13.4 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 ------------------#
return(list(IV_MED_MASSN_REGISTER = IV_MED_MASSN_REGISTER, IV_MED_MASSN_SONDEREFFEKTE =
↪   IV_MED_MASSN_SONDEREFFEKTE,
    IV_MED_MASSN_PARAMCHANGES = IV_MED_MASSN_PARAMCHANGES))
```

3.14 Modul mod_input_massnahmen_iv.R

Dokumentation zuletzt aktualisiert am 25.11.2024

Das Modul `mod_input_massnahmen_iv.R` liest die Daten ein, die für die Schätzung der Auswirkungen von Politikmassnahmen benötigt werden (vgl. Kapitel 1.2.2 und Kapitel 4.12) `mod_input_massnahmen_iv.R` wird wie folgt in `prepare_input.R` aufgerufen:

```
iv_massnahmen <- mod_input_massnahmen_iv(PARAM_INPUTS)
```

Das Modul liest zuerst die Ausgaben- respektive Einnahmevektoren der Politikmassnahmen, für welche die Berechnungen ausserhalb des IV Finanzperspektivenmodells durchgeführt wurden, ein (`MASSN_EXT`), sowie die Beschreibung dieser Massnahmen (`MASSN_EXT_DESC`). Danach werden die Daten zur Abschätzung der Auswirkungen der Deplafonierung eingelesen (`PLAF_IV`), sowie ein Auszug des IV-Rentenregisters für das letzte verfügbare Jahr (`IV_RR_RAM`):

```
#------ Lesen Massnahmen WEIV ----------------------------------------#
MASSN_EXT <- read_excel(path = paste0(PARAM_INPUTS$path_iv_massnahmen,
    PARAM_INPUTS$file_massn_ext_iv), sheet = PARAM_INPUTS$sheet_massn_ext_iv,
    range = cell_limits(c(1, 1), c(NA, NA)), col_types = "numeric")

MASSN_EXT_DESC <- read_excel(path = paste0(PARAM_INPUTS$path_iv_massnahmen,
    PARAM_INPUTS$file_massn_ext_iv), sheet = PARAM_INPUTS$sheet_desc_massn_ext_iv,
    range = cell_limits(c(1, 1), c(NA, NA)), col_types = "text")

PLAF_IV <- read_excel(file.path(PARAM_INPUTS$path_iv_massnahmen,
    PARAM_INPUTS$file_deplaf_iv))

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)


#------ Output -------------------------------------------------------#
return(list(MASSN_EXT = MASSN_EXT, MASSN_EXT_DESC = MASSN_EXT_DESC,
    PLAF_IV = PLAF_IV, IV_RR_RAM = IV_RR_RAM))
```

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.[24]

4.1 Modul run_iv.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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

```
run_iv(path_param = path_param, path_inp = path_inp, path_out = path_out)
```

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

Das Modul `run_iv` enthält die folgenden Elemente:

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

```
run_iv <- function(path_param, path_inp, path_out) {

    # Finanzperspektiven für alle Pfade nacheinander berechnen (für den Fall, dass
    ↪  "path_param" Pfade zu mehereren Parameter-Container enthält)
    if (length(path_param) > 1) {
    return(multi_run(path_param, run_iv, path_inp, path_out))
    }
```

Dieser erste 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:

Danach werden mit dem Modul `mod_read_data` (vgl. Kapitel 4.2) der unter `path_param` abgelegte Parametercontainer sowie die unter `path_inp` abgelegten Input-Daten eingelesen. Mit `data_folders=c("allgemein", "iv")` wird angegeben, dass sowohl die Allgemeinen als auch die IV-spezifischen Input-Daten eingelesen werden sollen:

```
# Parameterwerte und Inputdaten einlesen
tl_inp <- mod_read_data(path_param, path_inp, data_folders = c("allgemein",
    "iv"))
```

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

```
tl_inp$PARAM_GLOBAL <- mod_iv_param_global(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE = tl_inp$ECKWERTE, ESTV_IV = tl_inp$ESTV_IV, ESTV = tl_inp$ESTV,
    UMFRAGEN = tl_inp$UMFRAGEN, IV_ABRECHNUNG_PROV = tl_inp$IV_ABRECHNUNG_PROV,
    IV_ABRECHNUNG_DEF = tl_inp$IV_ABRECHNUNG_DEF, IK = tl_inp$IK,
    IV_RR_RAM = tl_inp$IV_RR_RAM, BEV_POP = tl_inp$BEV_POP, PARAM_MASSNAHMEN_BASE =
    ↪  tl_inp$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 4.3).[25]

Als nächstes wird das Modul `wrap_iv` (vgl. Kapitel 4.4) aufgerufen, in welchem die einzelnen Ausgaben- und Einnahmenpositionen des Finanzperspektivenmodells berechnet werden:

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

Zum Abschluss werden die Output-Dateien aufbereitet und in den vorangehend spezifizierten Output-Ordner geschrieben:

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

4.2 Modul mod_read_data.R

Dokumentation zuletzt aktualisiert am 14.01.2025

Das Modul `mod_read_data.R` liest den zu verwendenden Parametercontainer, sowie die Input-Daten ein. Es wird in `run_iv.R` wie folgt aufgerufen:

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

```
tl_inp <- mod_read_data(path_param, path_inp, data_folders = c("allgemein",
    "iv"))
```

Als Erstes werden in `mod_read_data.R` die Namen der .csv-Dateien mit den Parameterwerten in die Liste `csv_files_param` eingelesen, sowie die Namen der .csv-Dateien mit den Input-Daten in die Liste `csv_files_folders` eingelesen:

```
# Finde alle .csv-Dateien, die mit 'PARAM' beginnen
csv_files_param <- list.files(path = path_param, pattern = "^PARAM.*\\.csv$",
    full.names = TRUE, recursive = TRUE)

# Finde alle .csv-Dateien in den angegebenen Unterordnern
csv_files_folders <- map(data_folders, ~list.files(path = file.path(path_inp,
    .x), pattern = "\\.csv$", full.names = TRUE, recursive = TRUE)) %>%
    unlist()
```

Als nächstes werden die Funktionen definiert, mit welchen die einzelnen Paramter-Dateien (`process_file_param`) respektive die einzelnen Input-Dateien (`process_file_raw`) eingelesen werden:

```
    # Funktion für die 'PARAM'-Dateien mit Transformation
    process_file_param <- function(file) {
        cat("Reading: ", basename(file), "\n", sep = "")
        suppressMessages(
            read_delim(file, delim  = ";", show_col_types = FALSE, trim_ws = TRUE) # Lese
            ↪  Datei und lasse Spaltentypen automatisch bestimmen
         %>%
            select(1:2) %>%                                                  # Behalte nur
            ↪  die ersten zwei Spalten
            pivot_wider(names_from = "key", values_from = "value") %>%      #
            ↪  Transponiere die Daten
           { if (any(sapply(., is.character))) type_convert(.) else . }
        ) # Passe Spaltentypen an
        }

    # Funktion für die Ordner-Dateien ohne Transformation
    process_file_raw <- function(file) {
        cat("Reading: ", basename(file), "\n", sep = "")
        suppressMessages(
            read_delim(file, delim  = ";", show_col_types = FALSE, trim_ws = TRUE) %>% #
            ↪  Lese Datei und lasse Spaltentypen automatisch bestimmen
              { if (any(sapply(., is.character))) type_convert(.) else . }
↪   # Passe Spaltentypen an
        )
    }
```

Als nächstes werden die Parameter-Dateien und die Input-Dateien unter Verwendung der oben beschriebenen Funktionen eingelesen und der Liste `tl_inp` hinzugefügt:

```
# Verarbeite alle Dateien und benenne sie nach Dateinamen
tl_inp <- list()  # Initialisiere die Liste
```

```
# Füge die verarbeiteten 'PARAM'-Dateien hinzu
tl_inp <- c(tl_inp, set_names(map(csv_files_param, process_file_param),
    basename(csv_files_param) %>%
        str_remove("\\.csv$")))

# Füge output-id zu PARAM_GLOBAL hinzu
tl_inp$PARAM_GLOBAL$identifier_number <- ffh_identifier_number("iv",
    path_param)

# Füge die unbearbeiteten Dateien aus den angegebenen
# Ordnern hinzu
tl_inp <- c(tl_inp, set_names(map(csv_files_folders, process_file_raw),
    basename(csv_files_folders) %>%
        str_remove("\\.csv$")))
```

Zum Schluss wird die Liste `tl_inp` an `run_iv` zurückgeben:

```
return(tl_inp)
```

4.3 Modul mod_iv_param_global.R

Dokumentation zuletzt aktualisiert am 28.07.2025

Das Modul `mod_iv_param_global.R` setzt default-Parameterwerte in `PARAM_GLOBAL`. Es wird in `run_iv.R` wie folgt aufgerufen:[26]

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

Das Modul setzt 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. Das Modul enthält den folgenden Code, welcher selbsterklärend sein sollte:

```
    ## DEFAULT PARAMETER SETZEN, UM AKTUELLSTE WERTE ZU VERWENDEN (bspw. damit, falls
    ↪  nicht anders spezifiziert, die neusten ESTV MWST-Szenarien verwendet werden)

    # ESTV MwSt.-Szenario:
    # Überprüfe, ob id_estv existiert
    if (!"id_estv" %in% names(PARAM_GLOBAL)) {
        # Finde den höchsten Wert von "laufjahr" und "version"
        estv_selected <- ESTV |>
            filter(laufjahr == max(laufjahr)) |>
            filter(version == max(version))

        # Extrahiere eindeutige idestv
        unique_idestv <- estv_selected |>
            distinct(idestv)

        # Prüfe Anzahl verschiedener idestv
```

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

```
    if (nrow(unique_idestv) > 1) {
        # Zeige Warnung
        warning(
            paste0(
                "Mehrere verschiedene idestv im Dataframe ESTV mit dem höchsten Wert
                ↪  von laufjahr und version gefunden. Version ",
                unique_idestv$idestv[1],
                " ausgewählt (nur relevant, falls MWST-Zusatzfinanzierung aktiviert
                ↪  ist)."
            )
        )
        # Setze id_estv auf den ersten Eintrag
        PARAM_GLOBAL$id_estv <- unique_idestv$idestv[1]
    } else {
        # Setze id_estv auf den eindeutigen Eintrag
        PARAM_GLOBAL$id_estv <- unique_idestv$idestv
    }

    # Entferne Hilfsobjekte
    rm(estv_selected, unique_idestv)
}

# ESTV IV MwSt.-Szenario:
# Überprüfe, ob id_estv_iv existiert
if (!"id_estv_iv" %in% names(PARAM_GLOBAL)) {

    # Finde den höchsten Wert von "laufjahr" und "version"
    estviv_selected <- ESTV_IV %>%
        filter(laufjahr == max(laufjahr)) %>%               # Finde den höchsten
        ↪  "laufjahr"
        filter(version == max(version))                     # Finde den höchsten
        ↪  "version"

    # Prüfe, ob mehrere verschiedene idestv existieren
    unique_idivestv <- estviv_selected %>% distinct(idivestv)

    if (nrow(unique_idivestv) != 1) {
        stop("Mehrere verschiedene idestv im Dataframe ESTV_IV mit dem höchsten Wert
        ↪  von laufjahr und version gefunden. Code wird angehalten.")
    } else {
        # Setze id_estv in PARAM_GLOBAL auf den ausgewählten Wert
        PARAM_GLOBAL$id_estv_iv <- unique_idivestv$idivestv
    }
    rm(estviv_selected, unique_idivestv)
}

# ESTV IV MwSt.-Szenario:
# Überprüfe, ob id_estv_iv existiert
if (!"id_eckwerte" %in% names(PARAM_GLOBAL)) {

    # Finde den höchsten Wert von "laufjahr" und "version"
    id_selected <- ECKWERTE %>%
        filter(laufjahr == max(laufjahr)) %>%               # Finde den höchsten
        ↪  "laufjahr"
```

```
        filter(version == max(version))                    # Finde den höchsten
        ↪  "version"

    # Prüfe, ob mehrere verschiedene idestv existieren
    unique_id <- id_selected %>% distinct(id)

    if (nrow(unique_id) != 1) {
        stop("Mehrere verschiedene id im Dataframe ECKWERTE mit dem höchsten Wert von
        ↪  laufjahr und version gefunden. Code wird angehalten.")
    } else {
        # Setze id_estv in PARAM_GLOBAL auf den ausgewählten Wert
        PARAM_GLOBAL$id_eckwerte <- unique_id$id
    }
    rm(id_selected, unique_id)
}

# Umfragen:
# Überprüfe, ob id_umfragen existiert
if (!"id_umfragen" %in% names(PARAM_GLOBAL)) {

    # Finde den höchsten Wert von "laufjahr" und "version"
    umfrageid_selected <- UMFRAGEN %>%
        filter(laufjahr == max(laufjahr)) %>%              # Finde den höchsten
        ↪  "laufjahr"
        filter(version == max(version))                    # Finde den höchsten
        ↪  "version"

    # Prüfe, ob mehrere verschiedene id_umfragen existieren
    unique_umfrageid <- umfrageid_selected %>% distinct(umfrageid)

    if (nrow(unique_umfrageid) != 1) {
        stop("Mehrere verschiedene id_umfragen im Dataframe UMFRAGEN mit dem höchsten
        ↪  Wert von laufjahr und version gefunden. Code wird angehalten.")
    } else {
        # Setze id_umfragen in PARAM_GLOBAL auf den ausgewählten Wert
        PARAM_GLOBAL$id_umfragen <- unique_umfrageid$umfrageid
    }
    rm(umfrageid_selected, unique_umfrageid)
}

# Folgender Code nimmt das Referenzszenario an, falls kein Szenario in PARAM_GLOBAL
↪  spezifiziert wurde. Nur für Übergangsphase.
if (!"bev_scenario" %in% names(PARAM_GLOBAL)) {
    PARAM_GLOBAL$bev_scenario <- "A_00_2025"
    message("PARAM_GLOBAL$bev_scenario fehlt, Referenzszenario A_00_2025
    ↪  angenommen.")
}
# letztes Abrechnungsjahr festlegen, falls nicht festgelegt
if(!"jahr_abr" %in% names(PARAM_GLOBAL)) {
    if(PARAM_GLOBAL$abr_prov==TRUE){
        PARAM_GLOBAL$jahr_abr <- max(IV_ABRECHNUNG_PROV$jahr)
    }
    else{
```

```
            PARAM_GLOBAL$jahr_abr <- max(IV_ABRECHNUNG_DEF$jahr)
        }
    }
    # Preisbasis auf Abrechnungsjahr legen, falls nicht manuell gesetzt
    if(!"jahr_preisbasis" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_preisbasis <- PARAM_GLOBAL$jahr_abr
    }
    # laufendes Jahr jahr_lj festlegen, falls nicht manuell gesetzt
    if(!"jahr_lj" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_lj <- PARAM_GLOBAL$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(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_umfragen_fs <- max((UMFRAGEN %>%
↪   filter(umfrageid==PARAM_GLOBAL$id_umfragen))$jahr)+1
    }
    # Jahr des Beginns des Resultatevektors auf 1997 setzten, falls nicht manuell gesetzt
    if(!"jahr_beginn" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_beginn <- 1997
    }
    # Jahr des IK setzten (falls nicht manuell gesetzt)
    if(!"jahr_ik" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_ik <- max(IK$jahr)
    }
    # Jahr des Rentenregisters setzten (falls nicht manuell gesetzt)
    if(!"jahr_rr" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$jahr_rr <- max(IV_RR_RAM$an_rr)
    }
    # FHH für das mittlere Szenario (Basisszenario) berechnen, falls nicht festgelegt
    if(!"szenario_fhh" %in% names(PARAM_GLOBAL) || PARAM_GLOBAL$szenario_fhh == "mittel"
    ↪  || PARAM_GLOBAL$szenario_fhh == "basis") {
      PARAM_GLOBAL$szenario_fhh <- "referenz"
    }

    if(!"MA_years" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$MA_years <- 3
        warning("Parameter 'MA_years' ist in PARAM_GLOBAL nicht spezifiziert, 3
        ↪  angenommen.")
    }
    # Schattierung für .xls-FHH aktivieren, falls nicht anders spezifiziert
    if(!"year_shade" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$year_shade <- TRUE
    }
    # Schattierung ab Projektionsjahr jahr_abr+11 in .xls-FHH, falls nicht anders
    ↪  spezifiziert
    if(!"year_after_abr_shade" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$year_after_abr_shade <- 10
    }
    # Schattierte Werte in .xls-FHH runden, falls nicht anders spezifiziert
    if(!"round_shade" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$round_shade <- TRUE
    }
```

```
    # Spalten, die im Output in Millionen ausgegeben werden definieren, falls nicht
    ↪  definieren
    if(!"spaltenmio" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$spaltenmio <- "aus_tot_ivoz, aus_tot_iv, btr_vs_ag, btr_bund,
↪  regr_ein_tot, ein_tot_iv, erg_umlag_iv, anl_erg, erg_betr, fonds, fl_mtl,
↪  rueckzahlung, kap, schzins" # Spalten, die in den aufbereiteten Output-files (e.g.,
↪  FHH_IV.xlsx)  in Millionen Franken ausgegeben werden
    }
    # Spalten, die im Output mit Wachstumsraten gezeigt werden definieren, falls nicht
    ↪  definieren
    if(!"cols_with_wr" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$cols_with_wr <- "aus_tot_ivoz, schzins, aus_tot_iv, btr_vs_ag,
↪  btr_bund, regr_ein_tot, ein_tot_iv" # Spalten, für welche in den aufbereiteten
↪  Output-files (e.g., FHH_IV.xlsx) Jahr-zu-Jahr Wachstumsraten angezeigt werden.
    }
    # FHH real berechnen, falls nicht anders spezifiziert
    if(!"diskontierung" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$diskontierung <- TRUE # Parameter, der Angibt, ob gewisse outputs
↪  deflationiert werden sollen. Standardmässig auf TRUE
    }
    # Detaillierter Ausgaben-Output erstellen, falls nicht anders spezifiziert
    if(!"ausgaben_output" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$ausgaben_output <- FALSE # TRUE, falls auch eine xls-Tabelle mit
↪  einer detaillierten Auflistung der Ausgabepositionen erstellt werden soll (im Ordner
↪  post_process)
    }
    # Eckwerte für 5 Jahre anzeigen, falls nicht anders spezifiziert
    if(!"long_eckwerte" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$long_eckwerte <- 5 # Parameter, der angibt, für wie viele Jahre die
↪  Lohn- und Preiswachstumsraten unten rechts in den FHH angezeigt werden sollen.
    }

    # Setzte Untertitel leer falls er nicht existiert
    if (!"subtitle_d" %in% names(PARAM_GLOBAL)) {
        PARAM_GLOBAL$subtitle_d <- ""
    }
    if (!"subtitle_f" %in% names(PARAM_GLOBAL)) {
      PARAM_GLOBAL$subtitle_f <- ""
    }
    if (!"subtitle_i" %in% names(PARAM_GLOBAL)) {
      PARAM_GLOBAL$subtitle_i <- ""
    }

    # Ausgewähltes Szenario zum Titel hinzufügen
    if (PARAM_GLOBAL$szenario_fhh == "referenz") {
      PARAM_GLOBAL$title_d <- paste0(PARAM_GLOBAL$title_d, " (Referenzszenario)")
      PARAM_GLOBAL$title_f <- paste0(PARAM_GLOBAL$title_f, " (Scénario de référence)")
      PARAM_GLOBAL$title_i <- paste0(PARAM_GLOBAL$title_i, " (Scenario di riferimento)")

    } else if (PARAM_GLOBAL$szenario_fhh == "hoch") {
      PARAM_GLOBAL$title_d <- paste0(PARAM_GLOBAL$title_d, " (Hohes Szenario)")
      PARAM_GLOBAL$title_f <- paste0(PARAM_GLOBAL$title_f, " (Scénario haut)")
      PARAM_GLOBAL$title_i <- paste0(PARAM_GLOBAL$title_i, " (Scenario alto)")
```

```
  } else if (PARAM_GLOBAL$szenario_fhh == "tief") {
    PARAM_GLOBAL$title_d <- paste0(PARAM_GLOBAL$title_d, " (Tiefes Szenario)")
    PARAM_GLOBAL$title_f <- paste0(PARAM_GLOBAL$title_f, " (Scénario bas)")
    PARAM_GLOBAL$title_i <- paste0(PARAM_GLOBAL$title_i, " (Scenario basso)")
  }

  #------ Output ----------------------------------------------------------#
  return(PARAM_GLOBAL = PARAM_GLOBAL)
```

4.4 Modul wrap_iv.R

Dokumentation zuletzt aktualisiert am 25.11.2024

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 4.5 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 4.6 beschrieben.

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 4.12 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 4.13 beschrieben.

Darauffolgend werden Anhand der IV-Bilanz aus `wrap_iv_ergebnisse` verschiedene Indikatoren 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 4.14 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, DELTAS_EINN_AUSG =
    ↪  tl_iv_ergebnisse$DELTAS_EINN_AUSG,
    IV_INDICES = tl_iv_varia$IV_INDICES, DISKONTFAKTOR = tl_inp$DISKONTFAKTOR)
```

Das Modul `mod_iv_postprocessing` ist in Kapitel 4.15 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))
```

4.5 Modul wrap_iv_vorb_berechn.R

Dokumentation zuletzt aktualisiert am 18.03.2025

Das Modul `wrap_iv_vorb_berechn.R` führt Vorberechnungen für Projektionsvariablen durch. Es wird in `wrap_iv.R` wie folgt aufgerufen:

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

In einem ersten Schritt werden die Daten der IV-Abrechnung und der AHV-Abrechnung eingelesen (vgl. Kapitel 4.36). Die AHV-Abrechnung wird ebenfalls eingelesen, da diese später zur Projektion der Lohnbeiträge verwendet wird:

```
# Abrechnung
tl_abrechnung <- mod_abrechnung(tl_inp, versicherung = c("ahv",
    "iv"))
```

Als nächstes werden die Bevölkerungsdaten eingelesen und aufbereitet (vgl. Kapitel 4.37):

```
# Bevoelkerung
BEVOELKERUNG <- mod_population(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEV_BESTAND = tl_inp$BEV_BESTAND, BEV_SCENARIO = tl_inp$BEV_SCENARIO)
```

Als nächstes werden die Eckwerte zur wirtschaftlichen Entwicklung aufbereitet:

```
# Berechnung Eckwerte

tl_eckwerte <- mod_eckwerte(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE = tl_inp$ECKWERTE, LOHNINDEX = tl_inp$LOHNINDEX,
    PREISINDEX = tl_inp$PREISINDEX)
```

Das Modul `mod_eckwerte` (vgl. Kapitel 4.38) fügt historische Daten zur Lohn- und Preisentwicklung mit den Projektionsdaten zusammen und erstellt so eine Zeitreihe für die Entwicklung dieser volkswirtschaftlichen Eckwerte.

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

```
# Berechnung Diskontfaktor

DISKONTFAKTOR <- mod_diskontfaktor(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED)
```

Das Modul `mod_diskontfaktor` (vgl. Kapitel 4.39) 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 die Entwicklung der Minimalrente berechnet:

```
# Berechnung Rentenentwicklung gemaess Rentenanpassungen

RENTENENTWICKLUNG <- mod_rentenentwicklung(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED, MINIMALRENTE =
    ↪  tl_inp$MINIMALRENTE,
    PREISINDEX = tl_inp$PREISINDEX)
```

Hierbei wird im Modul `mod_rentenentwicklung` (vgl. Kapitel 4.40) 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 4.40).

Als nächstes wird die erwartete Rendite der nicht flüssigen Fondsanlagen sowie der Zins auf die IV-Schuld bei der AHV eingelesen (vgl. Kapitel 4.41:

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

ZINS <- mod_zins(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL, ZINS_RAW = tl_inp$ZINS_RAW,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED)
```

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 ------------------------#
BEITRAGSSAETZE <- mod_beitragssaetze(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    SV_BEITRAGSSATZ = tl_inp$SV_BEITRAGSSATZ)
```

Das Modul `mod_beitragssaetze` ist in Kapitel 4.42 beschrieben.

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 Umfragebasierten 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 4.43 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 -------------------------------------------#
FORTSCHREIBUNG_IV <- mod_iv_fortschreibung(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED, ECKWERTE_SCENARIO =
    ↪  tl_eckwerte$ECKWERTE_SCENARIO,
    BEVOELKERUNG = BEVOELKERUNG, IK = tl_inp$IK, IV_ABRECHNUNG =
    ↪  tl_abrechnung$IV_ABRECHNUNG,
    AHV_ABRECHNUNG = tl_abrechnung$AHV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR,
    ESTV_IV_SCEN = tl_iv_input$ESTV_IV_SCEN, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEV_BESTAND = tl_inp$BEV_BESTAND, BEITRAGSSAETZE = BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = tl_inp$ARBEITSLOSENQUOTE)
```

Das Modul `mod_iv_fortschreibung` ist in Kapitel 4.44 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 ---------------------------------------------------------#
c(tl_abrechnung, tl_eckwerte, tl_iv_input, list(BEVOELKERUNG = BEVOELKERUNG,
    ZINS = ZINS, DISKONTFAKTOR = DISKONTFAKTOR, BEITRAGSSAETZE = BEITRAGSSAETZE,
    FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV, RENTENENTWICKLUNG = RENTENENTWICKLUNG))
```

4.6 Modul wrap_iv_hauptberechnung.R

Dokumentation zuletzt aktualisiert am 16.03.2025

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

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

Als Erstes wird das Modul zur Berechnung der Ausgabeprojektionen aufgerufen (vgl. Kapitel 4.7):

```
tl_iv_ausgaben <- mod_iv_ausgaben(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEVOELKERUNG = tl_inp$BEVOELKERUNG, IV_ABRECHNUNG = tl_inp$IV_ABRECHNUNG,
    FORTSCHREIBUNG_IV = tl_inp$FORTSCHREIBUNG_IV, RENTENENTWICKLUNG =
    ↪  tl_inp$RENTENENTWICKLUNG,
    BEITRAGSSAETZE = tl_inp$BEITRAGSSAETZE, 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$DISKONTFAKTOR, IV_RENTEN = tl_inp$IV_RENTEN,
    IV_HE_ERWACHSENE = tl_inp$IV_HE_ERWACHSENE, IV_HE_KINDER = tl_inp$IV_HE_KINDER)
```

Darauffolgend wird das Modul zur Berechnung der Einnahmeprojektionen aufgerufen (vgl. Kapitel 4.11):

```
tl_iv_einnahmen <- mod_iv_einnahmen(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEVOELKERUNG = tl_inp$BEVOELKERUNG, BEV_BESTAND = tl_inp$BEV_BESTAND,
    IK = tl_inp$IK, AHV_ABRECHNUNG = tl_inp$AHV_ABRECHNUNG, IV_ABRECHNUNG =
    ↪  tl_inp$IV_ABRECHNUNG,
    AUSGABEN_IV = tl_iv_ausgaben$AUSGABEN_IV, ECKWERTE_SCENARIO =
    ↪  tl_inp$ECKWERTE_SCENARIO,
    ECKWERTE_EXTENDED = tl_inp$ECKWERTE_EXTENDED, BEITRAGSSAETZE = tl_inp$BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = tl_inp$ARBEITSLOSENQUOTE)
```

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

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

4.7 Modul mod_iv_ausgaben.R

Dokumentation zuletzt aktualisiert am 30.01.2025

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$IV_ABRECHNUNG,
    FORTSCHREIBUNG_IV = tl_inp$FORTSCHREIBUNG_IV, RENTENENTWICKLUNG =
    ↪  tl_inp$RENTENENTWICKLUNG,
    BEITRAGSSAETZE = tl_inp$BEITRAGSSAETZE, 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 = tl_inp$IV_RENTEN,
    IV_HE_ERWACHSENE = tl_inp$IV_HE_ERWACHSENE, IV_HE_KINDER = tl_inp$IV_HE_KINDER)
```

Als erstes wird das Modul zur Berechnung der Geldleistungen der IV aufgerufen (vgl. Kapitel 4.8):

```
#------ Geldleistungen / Prestations en espèces ----------------------#
tl_iv_geldleistungen <- mod_iv_geldleistungen(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
    FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEITRAGSSAETZE = BEITRAGSSAETZE, UMFRAGE_IV = UMFRAGE_IV,
    IV_RENTEN = IV_RENTEN, IV_HE_ERWACHSENE = IV_HE_ERWACHSENE,
    IV_HE_KINDER = IV_HE_KINDER)
```

Als nächstes wird das Modul zur Berechnung der Sachleistungen der IV aufgerufen (vgl. Kapitel 4.9):

```
#------ 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 4.10):

```
#------ Übrige Ausgaben ----------------------------------------------#
UEBRIGE_AUSGABEN_IV <- 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, welche erst im Bilanz-Modul berechnet werden können) enthält:

```
#------ Ausgaben in einem Dataframe zusammenfügen ---------------------#
AUSGABEN_IV <- tl_iv_geldleistungen$IV_GELDLEISTUNG %>%
    inner_join(tl_iv_sachleistungen$SACHLEISTUNG_IV, by = c("jahr")) %>%
    inner_join(UEBRIGE_AUSGABEN_IV, by = c("jahr")) %>%
    mutate(across(.cols = everything(), ~if_else(is.na(.), 0,
        .), .names = "{.col}"))
```

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:

```
#------ Output -------------------------------------------------------#
return(list(AUSGABEN_IV = AUSGABEN_IV, IV_MED_MASSN_FORECAST_AVERAGES =
↪  tl_iv_sachleistungen$IV_MED_MASSN_FORECAST_AVERAGES,
```

```
    BESTAND_IV = tl_iv_geldleistungen$BESTAND_IV, RENTENBESTAND_IV =
    ↪  tl_iv_geldleistungen$RENTENBESTAND_IV,
    HE_BESTAND_KINDER = tl_iv_geldleistungen$HE_BESTAND_KINDER,
    HE_BESTAND_ERWACHSENE = tl_iv_geldleistungen$HE_BESTAND_ERWACHSENE))
```

4.8 Modul mod_iv_geldleistungen.R

Dokumentation zuletzt aktualisiert am 30.01.2025

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,
    FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEITRAGSSAETZE = BEITRAGSSAETZE, UMFRAGE_IV = UMFRAGE_IV,
    IV_RENTEN = IV_RENTEN, IV_HE_ERWACHSENE = IV_HE_ERWACHSENE,
    IV_HE_KINDER = IV_HE_KINDER)
```

Als erstes wird das Modul zur Berechnung der Rentenausgaben aufgerufen (vgl. Kapitel 4.16):

```
#------ Renten ordentliche und ao / Rentes (ordinaires et exo.) -------#
tl_iv_rentensumme <- mod_iv_rentensumme(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEVOELKERUNG = BEVOELKERUNG, IV_RENTEN = IV_RENTEN)
```

Als nächstes wird das Modul zur Berechnung der Hilflosenentschädigung aufgerufen (vgl. Kapitel 4.18):

```
#------ Hilflosenentschädigungen / Allocations pour impotents ---------#
HE_IV <- mod_iv_he(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEVOELKERUNG = BEVOELKERUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    IV_HE_ERWACHSENE = IV_HE_ERWACHSENE, 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 4.17):

```
#------ Taggelder / Indemnités journalières ---------------------------#
TAGGELDER_IV <- 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 --#
IV_BEITRAGSANTEIL <- mod_iv_beitragsanteil_iv(PARAM_GLOBAL = PARAM_GLOBAL,
    TAGGELDER_IV = TAGGELDER_IV, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEITRAGSSAETZE = BEITRAGSSAETZE)
```

Zum Schluss wird ein Dataframe gebildet, das die projizierten Ausgabenvektoren für alle Geldleistungen enthält. Dieser Ausgabenvektor wird dann zusammen mit den Dataframes `BESTAND_IV` und `RENTENBESTAND_IV` an das Finanzperspektivenmodell, respektive das Modul `mod_iv_ausgaben`, zurückgegeben:

```
#------ Alle Geldleistungen ----------------------------------------#
  IV_GELDLEISTUNG     <- tl_iv_rentensumme$RENTENSUMME_IV %>%
     inner_join(tl_iv_he$HE_IV, by = c('jahr'))%>%
     inner_join(TAGGELDER_IV, by = c('jahr')) %>%
     inner_join(IV_BEITRAGSANTEIL, by = c('jahr'))


#------ Output -----------------------------------------------------#
    return(
      list(
          IV_GELDLEISTUNG = IV_GELDLEISTUNG,
          BESTAND_IV = tl_iv_rentensumme$BESTAND_IV,
          RENTENBESTAND_IV = tl_iv_rentensumme$RENTENBESTAND_IV,
          HE_BESTAND_KINDER = tl_iv_he$HE_BESTAND_KINDER,
          HE_BESTAND_ERWACHSENE = tl_iv_he$HE_BESTAND_ERWACHSENE
      )
    )

} # mod_iv_geldleistungen
```

4.9 Modul mod_iv_sachleistung.R

Dokumentation zuletzt aktualisiert am 14.01.2025

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 (vgl. Kapitel 4.19):

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

tl_iv_mm <- mod_iv_mm(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)
```

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

```
#------ Frühinterventionsmassnahmen / Mesures d'intervention précoce --#
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 -------------------#
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 ---------------#
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 ----#
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.-#
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 ------------------------------#
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 ---------------------------------#
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 -----------------#
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 ----#
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 4.20 bis 4.28 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(IV_FI, by = c("jahr")) %>%
    inner_join(IV_IM, by = c("jahr")) %>%
    inner_join(IV_BER_BEGL, by = c("jahr")) %>%
    inner_join(IV_MBA, by = c("jahr")) %>%
    inner_join(IV_AUS_UEBR, by = c("jahr")) %>%
    inner_join(IV_HM, by = c("jahr")) %>%
    inner_join(IV_RK, by = c("jahr")) %>%
    inner_join(IV_ASSB, by = c("jahr")) %>%
    inner_join(IV_RUECK_IM, by = c("jahr"))
```

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

```
return(list(SACHLEISTUNG_IV = SACHLEISTUNG_IV, IV_MED_MASSN_FORECAST_AVERAGES =
↪  tl_iv_mm$IV_MED_MASSN_FORECAST_AVERAGES))
```

4.10 Modul mod_iv_uebrigeausgaben.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 ------#
IV_INSTITUTIONEN <- mod_iv_institutionen(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, DISKONTFAKTOR =
    ↪  DISKONTFAKTOR)

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

#------ Andere Aufwände (Total) / Autres charges (total) ---------------#
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 4.29, 4.30 und 4.31 erläutert.

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

```
#------ Total alle übrigen Ausgaben ------------------------------------#
UEBRIGE_AUSGABEN_IV <- IV_INSTITUTIONEN %>%
    inner_join(IV_DURCHFUEHRUNG, by = c("jahr")) %>%
    inner_join(IV_VERWALTUNGSAUFWAND, by = c("jahr")) %>%
```

```
    inner_join(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 ------------------------------------------------------#
return(UEBRIGE_AUSGABEN_IV = UEBRIGE_AUSGABEN_IV)
```

4.11 Modul mod_iv_einnahmen.R

Dokumentation zuletzt aktualisiert am 16.03.2025

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

```
tl_iv_einnahmen <- mod_iv_einnahmen(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEVOELKERUNG = tl_inp$BEVOELKERUNG, BEV_BESTAND = tl_inp$BEV_BESTAND,
    IK = tl_inp$IK, AHV_ABRECHNUNG = tl_inp$AHV_ABRECHNUNG, IV_ABRECHNUNG =
    ↪  tl_inp$IV_ABRECHNUNG,
    AUSGABEN_IV = tl_iv_ausgaben$AUSGABEN_IV, ECKWERTE_SCENARIO =
    ↪  tl_inp$ECKWERTE_SCENARIO,
    ECKWERTE_EXTENDED = tl_inp$ECKWERTE_EXTENDED, BEITRAGSSAETZE = tl_inp$BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = tl_inp$ARBEITSLOSENQUOTE)
```

Als erstes wird in `mod_iv_einnahmen` das Modul zur Berechnung der Einnahmen aus Beiträgen von Arbeitgebern und Versicherten aufgerufen (vgl. Kapitel 4.32):

```
#------ Beiträge / Contributions ------------------------------------#
tl_iv_beitrag <- mod_iv_beitrag(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, BEV_BESTAND = BEV_BESTAND, IK = IK,
    ECKWERTE_EXTENDED = ECKWERTE_EXTENDED, ECKWERTE_SCENARIO = ECKWERTE_SCENARIO,
    AHV_ABRECHNUNG = AHV_ABRECHNUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEITRAGSSAETZE = BEITRAGSSAETZE, ARBEITSLOSENQUOTE = ARBEITSLOSENQUOTE)
```

Als nächstes wird das Modul zur Berechnung der übrigen Einnahmen aufgerufen (vgl. Kapitel 4.33):

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

Zum Schluss werden alle Einnahmen in einem Dataframe zusammengefasst. Dazu wird zuerst ein Dataframe erstellt, welches für die Spalten `btr_bund`, `btr_mwst`, `btr_zinsiv` die Werte der Abrechnungsdaten sowie 0 für alle zukünftigen Jahre enthält. Für `btr_mwst` und `btr_zinsiv` werden für die Zukunft 0-Werte angefügt, da momentan weder eine Finanzierung der IV über die MWST oder einen Sonderzins nach geltender Ordnung vorgesehen ist. 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 4.35) berechnet, und hier provisorisch auf 0 gesetzt.[27]

```
#------ Total der Einnahmen ----------------------------------------#
  EINNAHMEN_IV   <- data.frame(jahr = c(PARAM_GLOBAL$jahr_beginn:PARAM_GLOBAL$jahr_ende))
↪  %>%
     left_join(IV_ABRECHNUNG %>% select(jahr, btr_bund, btr_mwst, btr_zinsiv), by =
     ↪  "jahr") %>%
     mutate(
         btr_bund = coalesce(btr_bund, 0),       # Bundesbeitrag wird erst in mod_bilanz
         ↪  berechnet, und daher hier noch 0
         btr_mwst = coalesce(btr_mwst, 0),       # Ersetze NA durch 0 in btr_mwst
         btr_zinsiv = coalesce(btr_zinsiv, 0)    # Ersetze NA durch 0 in btr_zinsiv
     ) %>%
    inner_join(tl_iv_beitrag$IV_BEITRAGSSUMME,      by = c('jahr')) %>%
    inner_join(UEBRIGEEINNAHMEN_IV, by = c('jahr')) %>%
    mutate(across(where(is.numeric), ~if_else(is.na(.), 0, .), .names = '{.col}') )
```

Zum Schluss werden die Berechnungen an das Finanzperspektivenmodell, respektive das Modul `wrap_iv_hauptberechnung`, zurückgegeben:

```
#------ Output -----------------------------------------------------------#
return(list(EINNAHMEN_IV = EINNAHMEN_IV, IV_LOHNSUMME = tl_iv_beitrag$IV_LOHNSUMME))
```

4.12 Modul wrap_iv_massnahmen.R

Dokumentation zuletzt aktualisiert am 18.11.2024

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)
```

Als erstes werden zwei leere Dataframes erstellt, in welchen später die Auswirkungen von Massnahmen gespeichert werden:

```
# Liste und Dataframe für die Resultate initialisieren
tl_opt <- list()
DELTAS_MASSNAHMEN <- data.frame()
```

Danach folgt die Prüfung, ob die Auswirkungen von Massnahmen in die Finanzperspektiven-Berechnungen mit einbezogen werden sollen:

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

Wenn dies nicht der Fall ist (also `tl_inp$PARAM_GLOBAL$flag_param_massn==FALSE` ist), wird das Modul abgeschlossen und es werden die leeren Dataframes zurückgegeben:

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

```
#------ Output ------------------------------------------------------#
c(tl_opt, list(DELTAS_MASSNAHMEN = DELTAS_MASSNAHMEN  # Füge DELTAS_MASSNAHMEN hinzu
))
```

Falls Massnahmen spezifiziert wurden (also `tl_inp$PARAM_GLOBAL$flag_param_massn==TRUE` ist), wird die Massnahmen-Inputliste `tl_inp_massn` erstellt, welche die Input-Dataframes sowie die Dataframes der Hauptberechnungen enthält:

```
tl_inp_massn <- c(tl_inp, tl_iv_hauptberechnung)
```

Als nächstes wird überprüft, ob im Massnahmen-Parameterfile (vgl. Kapitel 1.2.2) Massnahmen spezifiziert sind, die innerhalb des Finanzperspektivenmodells berechnet werden sollen, also sogenannte „interne“ Massnahmen.

```
      ## INTERNE MASSNAHMEN BERECHNEN (falls solche ausgewählt sind)
      if (!is.na(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)) {
```

Wenn dies der Fall ist wir eine Liste der Massnahmen ausgelesen, und in `massn_int` abgelegt:

```
massn_int <- separate_at_comma(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)
```

Danach wird die Funktion spezifiziert, mit Hilfe welcher die Berechnungen der Auswirkungen der internen Massnahmen durchgeführt werden:

```
run_massnahmen <- function(massnahme_name, data) {
    funktion_name <- paste0("mod_opt_", massnahme_name)

    if (!exists(funktion_name)) {
        stop("Folgendes Massnahmen-Modul nicht in dmeasures gefunden: ",
            funktion_name)
    }

    # Call the module function with the data argument
    massnahme_effekte <- do.call(get(funktion_name), list(data))

    # Rename the data frames within massnahme_effekte
    names(massnahme_effekte) <- paste0(names(massnahme_effekte),
        "..", massnahme_name)

    # Return the modified massnahme_effekte
    return(massnahme_effekte)
}
```

Die Funktion `run_massnahmen` nimmt in `massnahme_name` den Namen der Massnahme entgegen, für welche die Auswirkungen berechnet werden sollen, und in `data` die Input-Daten, welche für die Berechnungen genutzt werden sollen. Als erstes überprüft die Funktion, ob eine Funktion (respektive ein Modul) zum Berechnen der Auswirkungen existiert. Wenn dies nicht der Fall ist wird eine Fehlermeldung ausgegeben, und ansonsten werden die Auswirkungen der Massnahme in `massnahme_name` berechnet und in die Liste `massnahme_effekte` abgelegt. Danach wird an den Namen der Dataframes in der Liste `massnahme_effekte` der Name der betreffenden Massnahme angefügt. Dies dient dazu, später die Dataframes mit den Massnahmeneffekte den richtigen Massnahmen zuzuordnen. Danach wird die Liste `massnahme_effekte` zurückgegeben.

Die Funktion wird mit dem `lappy` Befehl so aufgerufen, wodurch für jedes Element der Liste `massn_int` (also für jede spezifizierte Massnahme, die innerhalb des IV-Finanzperspektivenmodells berechnet werden soll) `run_massnahmen` unter Verwendung der Daten in `tl_inp_massn` ausgeführt wird. Der umhüllende `do.call` dient lediglich dazu sicherzustellen, dass die resultierende Liste `tl_massnahme_effekte` direkt die Dataframes aus den Listen der verschiedenen Massnahmen-Berechnungen enthält, anstatt die Listen selbst:

```
# Apply the function and store the result in
# tl_massnahme_effekte
tl_massnahme_effekte <- do.call(c, lapply(massn_int, run_massnahmen,
    data = tl_inp_massn))
```

Am Ende dieses Kapitels wird anhand des Beispiels des Moduls `mod_opt_iv_tabellenlohn_nv_rentiers.R` erläutert, wie die Module für die Massnahmen-Berechnungen aufgebaut sind.

Als nächstes werden die Dataframes mit den Massnahmen-Effekte aus `tl_massnahme_effekte` ausgelesen und in die Liste `tl_DELTAS_MASSNAHMEN` gelegt:

```
# Auswahl der Dataframes aus tl_massnahme_effekte, die mit
# 'DELTA' beginnen
tl_DELTAS_MASSNAHMEN <- tl_massnahme_effekte[grepl("^DELTA",
    names(tl_massnahme_effekte))]
```

Als nächstes werden für jedes Dataframe in `tl_DELTAS_MASSNAHMEN` die Massnahmen-EFfekte ausgelesen und im langen Format in das Dataframe `DELTA_MASSNAHMEN` gelegt:

```
# Prüfen und Umwandeln der Dataframes sowie Zeilenweise
# Verbinden
DELTAS_MASSNAHMEN <- lapply(names(tl_DELTAS_MASSNAHMEN), function(df_name) {
    df <- tl_DELTAS_MASSNAHMEN[[df_name]]

    # Prüfen, ob alle Spalten in IV_ABRECHNUNG existieren
    if (!all(colnames(df) %in% colnames(tl_inp_massn$IV_ABRECHNUNG))) {
        stop(sprintf("\033[31mFehler: Dataframe %s enthält Spalten, die nicht in
        ↪  IV_ABRECHNUNG enthalten sind.\033[0m",
            df_name))
    }

    # Umwandeln in Long-Format und 'massnahme' hinzufügen
    df_long <- df %>%
        pivot_longer(cols = -jahr, names_to = "konto", values_to = "wert") %>%
        mutate(massnahme = sub(".*\\.\\.", "", df_name))  # massnahme aus dem Namen
        ↪  extrahieren

    return(df_long)
}) %>%
    bind_rows()  # Alle Dataframes zeilenweise verbinden
```

Hierbei wird als erstes überprüft, ob die Spaltennamen im entsprechenden Dataframe `df` auch in der IV-Abrechnung enthalten sind. Zweck dieser Überprüfung ist es, sicherzustellen, dass alle finanziellen Auswirkungen der Massnahmen auch tatsächlich einer Position im Finanzperspektivenmodell zugewiesen werden können. Als nächstes wird `df` in das lange Format gebracht, was heisst, dass die Spaltennahmen (welche die von der Massnahme betroffenen Positionen der IV-Abrechnung enthalten) in die Spalte `konto` kopiert werden und die Massnahmen-Effekte im entsprechenden Jahr in die Spalte `wert`.

Zum Abschluss der Berechnungen für die internen Massnahmen werden auch noch die Dataframes mit den Auswirkungen der Massnahmen, die nicht die IV-Abrechnung betreffen, also die `OPT`-dataframes, ausgelesen (vgl. Kapitel 4.12.1):

```
# Auswahl der Dataframes aus tl_massnahme_effekte, die mit
# 'OPT' beginnen
tl_opt <- tl_massnahme_effekte[grepl("^OPT", names(tl_massnahme_effekte))]
```

Als nächstes wird überprüft, ob Massnahmen spezifiziert sind, deren Auswirkungen ausserhalb des IV-Finanzperspektivenmodells berechnet wurden (sogenannte „externe“ Massnahmen):

```
      ## EXTERNE MASSNAHMEN HINZUFÜGEN (falls solche ausgewählt sind)
      if (!is.na(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_ext)) {
```

Ist dies der Fall, wird eine Liste mit den externen Massnahmen ausgelesen:

```
massn_ext <- separate_at_comma(tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_ext)
```

Als nächstes wird das Dataframe `MASSN_EXT_DESC`, welches die Informationen zu den externen Massnahmen enthält, für die weitere Bearbeitung in die lange Form gebracht:

```
# Reshape MASSN_EXT_DESC, um Variablennamen als Spalten zu
# behandeln
MASSN_EXT_DESC <- tl_inp_massn$MASSN_EXT_DESC %>%
    rename(colname = massnahme) %>%
    pivot_longer(-colname, names_to = "massnahme", values_to = "value") %>%
    pivot_wider(names_from = colname, values_from = value)
```

Danach wird überprüft, ob alle in `massn_ext` spezifizierten Massnahmen einen Vektor mit Auswirkungen der betreffenden Massnahme in `MASSN_EXT`, sowie einen Eintrag in `MASSN_EXT_DESC` haben:

```
          # Überprüfen, ob alle Werte aus massn_ext in MASSN_EXT und als massnahme in
          ↪  MASSN_EXT_DESC vorhanden sind
          columns_in_massn_ext <- massn_ext %in% colnames(tl_inp_massn$MASSN_EXT)
          keys_in_massn_ext_desc <- massn_ext %in% MASSN_EXT_DESC$massnahme

          # Sicherstellen, dass alle benötigten Spalten vorhanden sind
          if (all(columns_in_massn_ext) && all(keys_in_massn_ext_desc)) {

              # Relevante Spalten aus MASSN_EXT auswählen und in Long-Format umwandeln
              massn_ext_long <- tl_inp_massn$MASSN_EXT %>%
                  select(jahr, all_of(massn_ext)) %>%
                  pivot_longer(cols = all_of(massn_ext), names_to = "massnahme",
                  ↪  values_to = "wert")

              # Daten kombinieren
              combined_data <- massn_ext_long %>%
                  left_join(MASSN_EXT_DESC %>% filter(massnahme %in% massn_ext) %>%
                  ↪  select(massnahme, konto), by = "massnahme") %>% # Daten basierend
                  ↪  auf massnahme zusammenführen
                  select(jahr, konto, wert, massnahme) # Endgültige Spalten auswählen
```

```
            # Zeilen an DELTAS_MASSNAHMEN anhängen
            DELTAS_MASSNAHMEN <- bind_rows(DELTAS_MASSNAHMEN, combined_data) # Daten zu
↪  DELTAS_MASSNAHMEN hinzufügen
        } else {
            stop("Nicht alle Werte aus massn_ext sind in MASSN_EXT oder MASSN_EXT_DESC
            ↪  vorhanden.") # Fehler auslösen, wenn Spalten fehlen
        }
```

Wenn dies der Fall ist (also `all(columns_in_massn_ext) && all(keys_in_massn_ext_desc) TRUE` ist), werden die in `massn_ext` spezifizierten Massnahmen aus `MASSN_EXT` ausgelesen, in das lange Format gebracht, wobei die Spalte `massnahme` den Massnahmen-Name enthält und `wert` die Auswirkung der Massnahme, und in das Dataframe `massn_ext_long` abgelegt. Danach wird für jede Massnahme aus dem Dataframe `MASSN_EXT_DESC` das betroffenen Konto ausgelesen und das resultierende Dataframe `combined_data` wird an das Dataframe `DELTA_MASSNAHMEN` angefügt.

Zum Schluss wird noch sichergestellt, dass `DELTA_MASSNAHMEN` keine NA-Werte enthält, und die Liste `tl_opt` sowie das Dataframe `DELTAS_MASSNAHMEN` werden an das Finanzperspektivenmodell, repsektive das Modul `wrap_iv.R` zurückgegeben:

```
# Überprüfe auf NA-Werte in der Spalte 'wert'
if (any(is.na(DELTAS_MASSNAHMEN$wert))) {
    stop("Massnahme-Schätzung enthält NA-Wert. Massnahme-Module überprüfen.")
}

#------ Output --------------------------------------------------------#
c(tl_opt, list(DELTAS_MASSNAHMEN = DELTAS_MASSNAHMEN  # Füge DELTAS_MASSNAHMEN hinzu
))
```

4.12.1 Modul mod_opt_iv_tabellenlohn_nv_rentiers.R

`mod_opt_iv_tabellenlohn_nv_rentiers.R` dient dazu, die Auswirkungen des Pauschalabzugs bei der Bemessung des Invaliditätsgrades abzuschätzen.[28] Nachfolgend wird an diesem Beispiel erläutert, wie die Massnahmen-Module aufgebaut sind.

Die Massnahmen-Module werden mit einer Funktion aufgerufen, die als einziges Argument die Liste `tl_inp_massn` enthält:

```
mod_opt_iv_tabellenlohn_nv_rentiers <- function(tl_inp_massn) {
```

Danach werden die zu verwendenden Dataframes aus der Liste `tl_inp_massn` extrahiert, und es werden die Berechnungen durchgeführt. Dieser Teil folgt keiner zwingenden Struktur. Wie nachfolgend ersichtlich ist, wird in vorliegendem Fall sogar auf ein Untermodul zurückgegriffen, in welchem die Berechnungen durchgeführt werden. Dies ist nicht zwingend, und es ist möglich und einfachheitshalber empfohlen, die Berechnungen direkt im jeweiligen `mod_opt` Modul durchzuführen.

```
PARAM_GLOBAL <- tl_inp_massn$PARAM_GLOBAL
PARAM_IV_NV_TABELLENLOHN <- tl_inp_massn$PARAM_IV_NV_TABELLENLOHN
IV_ABRECHNUNG <- tl_inp_massn$IV_ABRECHNUNG
IV_RR_RAM <- tl_inp_massn$IV_RR_RAM
```

[28] https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-98253.html

```
RENTENENTWICKLUNG <- tl_inp_massn$RENTENENTWICKLUNG
BESTAND_IV <- tl_inp_massn$BESTAND_IV


tl_iv_tabellenlohn_nv_rentier <- mod_iv_tabellenlohn_nv_rentier(PARAM_GLOBAL =
↪  PARAM_GLOBAL,
    PARAM_MASSNAHMEN = PARAM_IV_NV_TABELLENLOHN, IV_ABRECHNUNG = IV_ABRECHNUNG,
    IV_RR_RAM = IV_RR_RAM, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BESTAND_IV = BESTAND_IV)
```

Zum Schluss werden die Resultate der Berechnungen zurückgegeben. Hierbei ist es wichtig, zwei Arten von Dataframes zu unterscheiden:

1. Das `DELTA`-Dataframe, welches die finanziellen Auswirkungen der Massnahmen pro Jahr enthält Ein `DELTA`-Dataframe enthält die Spalte `jahr` sowie eine oder mehrere Spalten mit Ausgaben- oder Einnahmepositionen aus der IV-Abrechnung. In vorliegendem Fall enthält `DELTA` die Spalten `jahr` und `rent_ord`.
2. Eines oder mehrere `OPT`-Dataframes, welches die nicht-finanziellen Auswirkungen der Massnahme pro Jahr enthält. Das `OPT`-Dataframe enthält die Spalte `jahr` sowie eine oder mehrere Spalten mit Auswirkungen. IN vorliegendem Fall enhält `OPT` die Spalten `jahr` und `faelle`, wobei letztere Spalte die Anzahl Personen umfasst, die dank dem Pauschalabzug neu eine IV-Rente erhalten.

```
return(list(DELTA = tl_iv_tabellenlohn_nv_rentier$DELTA, OPT_FAELLE_NV_RENTIER =
↪  tl_iv_tabellenlohn_nv_rentier$OPT_FAELLE_NV_RENTIER))
```

4.13 Modul wrap_iv_ergebnisse.R

Dokumentation zuletzt aktualisiert am 29.01.2025

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)
```

Zuerst wird das Modul aufgerufen, das die Einnahmen- und Ausgaben aus den Hauptberechnungen und den Massnahmen-Berechnungen aggregiert (vgl. Kapitel 4.34), und danach das Modul, das die Bilanz der IV für jedes zukünftige Jahr berechnet (vgl. Kapitel 4.35).

```
#------ 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, DELTAS_MASSNAHMEN =
    ↪  tl_iv_mass$DELTAS_MASSNAHMEN)

#------ Berechnungen Umlage und Bilanz --------------------------------#
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,
    FORTSCHREIBUNG_IV = tl_inp$FORTSCHREIBUNG_IV, DELTAS_MASSNAHMEN =
    ↪  tl_iv_mass$DELTAS_MASSNAHMEN,
    tl_inp = tl_inp)
```

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,
```

4.14 Modul wrap_iv_varia.R und mod_iv_indices.R

Dokumentation zuletzt aktualisiert am 19.11.2024

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)
```

`wrap_iv_varia.R` besteht aus den folgenden Codezeilen:

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

#------ Hinzufügen des Entschuldungsszenarios -------------------------#
IV_SCHULD_SCEN <- tl_iv_ergebnisse$BILANZ_IV %>%
    select(jahr, schzins, kap, rueckzahlung) %>%
    filter(jahr >= 2009) %>%
    rename(zins_iv = schzins, schuld_iv = kap, rueckz_iv = rueckzahlung) %>%
    mutate(scen = paste0("iv_schuld_", format(Sys.Date(), "%Y_%m_%d"))) %>%
    bind_rows(tl_inp$IV_SCHULD_SCEN)  # Zeilen zusammenfügen


#------ Output --------------------------------------------------------#
return(list(IV_INDICES = IV_INDICES, IV_SCHULD_SCEN = IV_SCHULD_SCEN))
```

Im Dataframe `IV_SCHULD_SCEN` wird der neue Entschuldungsvektor für die IV Schuld bei der AHV berechnet, und an das Dataframe mit den verganenen Entschuldungsvektoren (`tl_inp$IV_SCHULD_SCEN`) angefügt. Der Entschuldungsvektor dient als Input für den AHV Finanzhaushalt, und muss im Falle, dass ein neuer Finanzhaushalt IV nach geltender Ordnung erstellt wird, nach Ausführen des IV-Finanzhaushalt aus dessen Output-Ordner in den Ordner `.../02_data_container` kopiert werden.

`mod_iv_indices` enthält den folgenden Code:

```
#------ Berechnen der Indices -----------------------------------------#
IV_INDICES <- BILANZ_IV %>%
    arrange(jahr) %>%
```

```
    mutate(indx_bund = 100 * btr_bund/aus_tot_iv) %>%
    mutate(indx_fl_mtl_ausg = (fonds/aus_tot_iv - PARAM_GLOBAL$fl_mittel_iv) *
        100)

#------ Output -----------------------------------------------------------#
return(IV_INDICES = IV_INDICES)
```

Dieser Code dient dazu, den Anteil des Bundesbeitrags an den Gesamtausgaben (`indx_bund`) sowie den Anteil der Flüssigen Mittel und Anlagen an den Gesamtausgaben zu berechnen. Beide Grössen werden in den letzten zwei Spalten der publizierten Finanzperspektiven ausgewiesen.

4.15 Modul mod_iv_postprocessing.R

Dokumentation zuletzt aktualisiert am 18.11.2024

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:

```
IV_FHH_NOM <- IV_INDICES

#------ realer FH ---------------------------------------------------------#
IV_FHH <- IV_BILANZ %>%
    diskontierung(DISKONTFAKTOR) %>%
    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) {

    DELTAS_EINN_AUSG_NOM <- DELTAS_EINN_AUSG
    DELTAS_EINN_AUSG_REAL <- DELTAS_EINN_AUSG %>%
        diskontierung(DISKONTFAKTOR)

    tl_out <- list(DELTAS_EINN_AUSG_REAL = DELTAS_EINN_AUSG_REAL,
        IV_FHH = IV_FHH, DELTAS_EINN_AUSG_NOM = DELTAS_EINN_AUSG_NOM,
        IV_FHH_NOM = IV_FHH_NOM)
} else {
    tl_out <- list(IV_FHH = IV_FHH, IV_FHH_NOM = 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:

```
#------ Output --------------------------------------------------------#
return(tl_out)
```

4.16 Modul mod_iv_rentensumme.R

Dokumentation zuletzt aktualisiert am 22.07.2025

In diese Kapitel wird die technische Umsetzung des Modells für die Rentenausgaben diskutiert. Um die Erläuterungen hier zu verstehen sollte vorgängig der Abschnitt zu den Renten im Dokument „Modellbeschrieb_IV“ gelesen werden.

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

```
tl_iv_rentensumme <- mod_iv_rentensumme(PARAM_GLOBAL = PARAM_GLOBAL,
    IV_ABRECHNUNG = IV_ABRECHNUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEVOELKERUNG = BEVOELKERUNG, IV_RENTEN = IV_RENTEN)
```

Das Modul `mod_iv_rentensumme.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`: Dataframe mit Details zum Retenbestand (gemäss Rentenregister) sowie der Neurenten nach Jahr, Alter und Geschlecht.

In einem ersten Schritt wird in `mod_iv_rentensumme` der Parameter `jahr_projektion` gesetzt:[29]

```
jahr_projektion <- PARAM_GLOBAL$jahr_abr + 1
```

4.16.1 Berechnung von Rentenbestand, Neurenten, und Mutationen

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

```
BESTAND <- IV_RENTEN %>%
    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:

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

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

```
        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.

4.16.2 Berechnung der Neurentenquote und der Mutationsrate und Szenarien

Um die Neurentenquote und die Mutationsraten zu berechnen, werden als Erstes die für die Berechnung dieser Quoten relevanten Bevölkerungsgruppen (=Zähler und Nenner der Quoten) extrahiert:

```
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) +
            sum(frontaliers, na.rm = TRUE))) %>%
    mutate(bev_neurenten = bevendejahr - bestand_vj_n, bev_mutationen = bestand_vj,
        bev_mutationen_n = bestand_vj_n) %>%
    filter(jahr >= PARAM_GLOBAL$jahr_abr - (PARAM_GLOBAL$MA_years -
        1), jahr <= PARAM_GLOBAL$jahr_abr) %>%
    select(sex, alt, neurenten, neurenten_n, bestandesmutationen,
        bestandesmutationen_n, bev_neurenten, bev_mutationen,
        bev_mutationen_n)
```

Im Falle der Neurentenquote ist der Zähler die Anzahl Neurenten (in Minimalrenten) in einem Jahr in dieser Altersgruppe (Variable `neurenten`), und der Nenner die Bevölkerung am Ende des Vorjahres in der Altersgruppe (Variable `bevendejahr`) abzüglich deren, die bereits eine IV-Rente haben (Variable `bestand_vj_n`), was dann in der Variable `bev_neurenten` abgebildet wird. Die Neurentenquote im Jahr t berechnet sich als der Anteil der Versichertenpopulation, der Anfang des Jahres t noch keine Invalidenrente bezieht aber Ende des Jahres t eine Invalidenrente bezieht. Daher wird 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 hier relevante Bevölkerung, das heisst die Population der Rentenversicherten, ist die Wohnbevölkerung zuzüglich der Grenzgänger.[30]

Die Neurentenquote 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. Im Falle der Mutationsrate ist der Zähler die Mutationen und der Nenner der Rentenbestand am Ende des Vorjahres (beides in Minimalrenten). Sobald diese Variablen Berechnet sind werden die Jahre ausgewählt, über welche die Neurentenquote und die Mutationsrate für die Fortschreibung geschätzt werden sollen.

Als nächstes wird die Funktion spezifiziert, mit welcher der Mittelwert sowie die Vertrauensintervalle der Neurentenquote und der Mutationsrate berechnet werden. Die Vertrauensintervalle für die Neurentenquo-

[30] Grundsätzlich ist die IV eine Erwerbsausfallversicherung, was nahelegen würde, die Inland-Erwerbsbevölkerung als Versichertenpopulation zu wählen. Auf Grund von Mitversicherungen von Ehepartnern, sowie ausserordentlichen Rentenansprüchen, zählen viele Personen, die Teil der Wohnbevölkerung und nicht Teil der Erwerbsbevölkerung sind, auch zur Versichertenpopulation. Wir erachten daher die Wohnbevölkerung zuzüglich der Grenzgänger als die bessere Approximation der Versichertenpopulation als die Inländer-Erwerbsbevölkerung zuzüglich der Grenzägnger (=Inland-Erwerbsbevölkerung).

te und die Mutationsrate dienen später dazu, die Szenarien für die Entwicklung der Neurenten und der Mutationen zu berechnen:

```
    # Hilfsfunktion zur Berechnung von Mittelwert und 95%-KI
    berechne_mittelwert_und_ki <- function(data, wert, n, bev) {
      data <- data |>
        mutate(
          mean_wert = .data[[wert]] / abs(.data[[n]]),                    # Berechne
          ↪  Mittelwert unter nicht-Nullen
          p = abs(.data[[n]]) / .data[[bev]],                             # Berechne Anteil
          ↪  nicht-Nullen
          wert_all = p * mean_wert                                        # Mittlere
          ↪  Quote/Rate in einem Jahr
        )

      mean_val <- mean(data$wert_all)                                     # Berechne
↪   Mittelwert über die Jahre in von (PARAM_GLOBAL$MA_years - 1) bis
↪   PARAM_GLOBAL$jahr_abr (Wert für mittleres Szenario)

      se <- sqrt(
        sum((data$p * (data$mean_wert - mean_val)^2 +
               (1 - data$p) * (0 - mean_val)^2) * data[[bev]]) /
          sum(data[[bev]])^2                                                # Berechne
          ↪  Standardfehler
      )

      tibble(
        mean = mean_val,
        lower = mean_val - 1.645 * se, # 90%-CI
        upper = mean_val + 1.645 * se  # 90%-CI
      )
    }
```

Hierzu wird die Formel zur Berechnung Ddes Standardfehlers eines gewichteten Mittelwerts bei binären Zufallsereignissen verwendet. Für jede Zeile wird angenommen, dass der Wert entweder mean_wert (mit Wahrscheinlichkeit p) oder 0 (mit Wahrscheinlichkeit $1 - p$) ist. Die Varianzbeiträge werden mit der Bevölkerungsgröße bev gewichtet, aufsummiert und durch das Quadrat der Gesamtbevölkerung geteilt. Formal lautet der Standardfehler:

$$\mathrm{SE} = \sqrt{\frac{\sum \left[p_i (x_i - \bar{x})^2 + (1 - p_i)(0 - \bar{x})^2\right] \cdot w_i}{\left(\sum w_i\right)^2}}$$

Da der Mittelwert normalverteilt ist, können mit Hilfe des kritischen Wertes von 1,645 für das 90%-Konfidenzintervall die Werte für das tiefe und das hohe Konfidenzband ausgewählt werden. Dies geschieht mit dem letzten Teil des obigen Codes.

Der nachfolgende Code dient dazu, mit Hilfe der oben beschriebenen Funktion die Konfidenzbänder für die Neurentenquote und die Mutationsrate zu berechnen. Das Suffix `_guenstig` beschreibt hier das hohe Szenario (hoch bezogen auf das Umlageergebnis), und `_ungunstig` das tiefe Szenario. Die Werte werden separat nach Alter und Geschlecht berechnet. Es werden sowohl die Werte für die Neurenten und Mutationen in Minimalerente, als auch die Werte für die Neurenten und Mutationen in Anzahl Personen (Variablen mit dem Suffix `_n`) berechnet:

```
# Neurenten je (sex, alt)
ci_neurenten <- INV_MUT |>
    group_by(sex, alt) |>
    group_modify(~{
        ki <- berechne_mittelwert_und_ki(.x, "neurenten", "neurenten_n",
            "bev_neurenten")
        tibble(neurentenquote = ki$mean, neurentenquote_guenstig = ki$lower,
            neurentenquote_unguenstig = ki$upper)
    }) |>
    ungroup()

ci_neurenten_n <- INV_MUT |>
    group_by(sex, alt) |>
    group_modify(~{
        ki <- berechne_mittelwert_und_ki(.x, "neurenten_n", "neurenten_n",
            "bev_neurenten")
        tibble(neurentenquote_n = ki$mean, neurentenquote_n_guenstig = ki$lower,
            neurentenquote_n_unguenstig = ki$upper)
    }) |>
    ungroup()

# Mutationen je (sex, alt)
ci_mutationen <- INV_MUT |>
    group_by(sex, alt) |>
    group_modify(~{
        ki <- berechne_mittelwert_und_ki(.x, "bestandesmutationen",
            "bestandesmutationen_n", "bev_mutationen")
        tibble(mutationsrate = ki$mean, mutationsrate_guenstig = ki$lower,
            mutationsrate_unguenstig = ki$upper)
    }) |>
    ungroup()

ci_mutationen_n <- INV_MUT |>
    group_by(sex, alt) |>
    group_modify(~{
        ki <- berechne_mittelwert_und_ki(.x, "bestandesmutationen_n",
            "bestandesmutationen_n", "bev_mutationen_n")
        tibble(mutationsrate_n = ki$mean, mutationsrate_n_guenstig = ki$lower,
            mutationsrate_n_unguenstig = ki$upper)
    }) |>
    ungroup()
```

Mit Hiilfe des nachfolgenden Codeblocks werden die Werte kombiniert und im Dataframe `RATEN_INV_MUT` zusammengefasst. Ausserdem wird sichergestellt, dass die Quoten/Raten alle zwischen 0 und 1 liegen:

```
# Kombiniere zu RATEN_INV_MUT
RATEN_INV_MUT <- ci_neurenten |>
    left_join(ci_neurenten_n, by = c("sex", "alt")) |>
    left_join(ci_mutationen, by = c("sex", "alt")) |>
    left_join(ci_mutationen_n, by = c("sex", "alt")) |>
    mutate(across(-c(sex, alt), ~replace_na(pmin(pmax(., -1),
        1), 0)  # Ersetze NaN durch 0, schneide Werte auf [-1, 1]
)) |>
```

```
    group_by(sex) |>
    mutate(across(contains("mutationsrate"), ~if_else(alt ==
        max(alt), -1, .)  # Setze mutationsrate-Werte bei max(alt) auf -1
)) |>
    ungroup()
```

4.16.3 Rentenbestand projizieren

In einem nächsten Schritt wird das Dataframe `RENTENBESTAND` initialisiert, in welchem die projizierten Flüsse (Bestandesmutationen und Neurenten) und Rentenbestände für jedes Szenario abgelegt werden sollen. Dafür werden die in `BESTAND` abgelegten historischen Rentenbestände verwendet:

```
RENTENBESTAND <- BESTAND %>%
    select(-contains("_hr"), -contains("_kr"), -contains("_vj"),
        -contains("_pfrt")) %>%
    mutate(neurenten_guenstig = neurenten, neurenten_unguenstig = neurenten,
        neurenten_n_guenstig = neurenten_n, neurenten_n_unguenstig = neurenten_n,
        bestandesmutationen_guenstig = bestandesmutationen,
        ↪  bestandesmutationen_unguenstig = bestandesmutationen,
        bestandesmutationen_n_guenstig = bestandesmutationen_n,
        bestandesmutationen_n_unguenstig = bestandesmutationen_n,
        bestand_guenstig = bestand, bestand_unguenstig = bestand,
        bestand_n_guenstig = bestand_n, bestand_n_unguenstig = bestand_n,
        quelle = "Register")
```

Der `filter`-Befehl wählt für die Berechnung der Neurentenquote und der Mutationsrate die spezifizierte Anzahl Jahre vor dem letzten Abrechnungsjahr aus (Stand 2025 nutzen wir 2 Jahre, also die Jahre 2023 und 2024). Danach wird die Neurentenquote/Mutationsrate aus der durchschnittlichen Neurentenquote/Mutationsrate der ausgewählten Jahre gebildet.

In der folgenden Schleife werden rekursiv die Rentenbestände projiziert, wobei zuerst die Neurentenquoten und Mutationsraten unter Berücksichtigung der Rentenaltererhöhung bei Frauen im Rahmen der AHV21 berechnet werden:

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

      RENTENBESTAND_NEU <- RENTENBESTAND |>
        filter(
          jahr == lJahr - 1,
          quelle == ifelse(lJahr == jahr_projektion, "Register", "Projektion"),
          (sex == "m" & alt <= 64) |
            (sex == "f" & lJahr < 2025 & alt <= 63) |
            (sex == "f" & lJahr >= 2025 & alt <= 64)
        ) |>
        select(jahr, sex, alt, starts_with("bestand")) |>
        full_join(
          BEVOELKERUNG |>
            group_by(jahr, sex, alt) |>
            summarise(bevendejahr = sum(bevendejahr) + sum(frontaliers, na.rm = TRUE),
            ↪  .groups = "drop") |>
            filter(
```

```
            alt >= 17,
            (sex == "m" & alt <= 64) |
              (sex == "f" & lJahr < 2025 & alt <= 63) |
              (sex == "f" & lJahr >= 2025 & alt <= 64),
            jahr == lJahr - 1
          )
      ) |>
      arrange(sex, alt) |>
      mutate(
        jahr = jahr + 1,
        alt = alt + 1
      ) |>
      left_join(
        bind_rows(
          RATEN_INV_MUT,
          RATEN_INV_MUT |>
            filter(sex == "f" & alt == 64) |>
            mutate(
              alt = alt + 1,
              across(contains("neurentenquote"), ~0),
              across(contains("mutationsrate"), ~ -1)
            )
        ) |>
          left_join(
            RATEN_INV_MUT |>
              filter(sex == "f" & alt == 63) |>
              mutate(alt = alt + 1) |>
              rename_with(
                ~ paste0(., "_ahv21"),
                .cols = c(contains("neurentenquote"), contains("mutationsrate"))
              )
          ) |>
          mutate(
            neurentenquote = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                     (1 - min(1, (lJahr - 2024) / 4)) * neurentenquote
↪  +
                                       min(1, (lJahr - 2024) / 4) *
                                       ↪  neurentenquote_ahv21,
                                     neurentenquote),
            neurentenquote_n = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                       (1 - min(1, (lJahr - 2024) / 4)) *
↪  neurentenquote_n +
                                         min(1, (lJahr - 2024) / 4) *
                                         ↪  neurentenquote_n_ahv21,
                                       neurentenquote_n),
            neurentenquote_guenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                              (1 - min(1, (lJahr - 2024) / 4)) *
↪  neurentenquote_guenstig +
                                                min(1, (lJahr - 2024) / 4) *
                                                ↪  neurentenquote_guenstig_ahv21,
                                              neurentenquote_guenstig),
            neurentenquote_unguenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                                (1 - min(1, (lJahr - 2024) / 4)) *
↪  neurentenquote_unguenstig +
```

```
                                                  min(1, (lJahr - 2024) / 4) *
                                                   ↪  neurentenquote_unguenstig_ahv21,
                                                  neurentenquote_unguenstig),
            neurentenquote_n_guenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                                  (1 - min(1, (lJahr - 2024) / 4)) *
↪  neurentenquote_n_guenstig +
                                                  min(1, (lJahr - 2024) / 4) *
                                                   ↪  neurentenquote_n_guenstig_ahv21,
                                                  neurentenquote_n_guenstig),
            neurentenquote_n_unguenstig = if_else(alt == 64 & sex == "f" & lJahr >=
             ↪  2025,
                                                  (1 - min(1, (lJahr - 2024) / 4)) *
↪  neurentenquote_n_unguenstig +
                                                    min(1, (lJahr - 2024) / 4) *
                                                     ↪  neurentenquote_n_unguenstig_ahv21,
                                                  neurentenquote_n_unguenstig),

            mutationsrate = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                    (1 - min(1, (lJahr - 2024) / 4)) * mutationsrate +
                                     min(1, (lJahr - 2024) / 4) * mutationsrate_ahv21,
                                    mutationsrate),
            mutationsrate_n = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                      (1 - min(1, (lJahr - 2024) / 4)) *
↪  mutationsrate_n +
                                       min(1, (lJahr - 2024) / 4) *
                                        ↪  mutationsrate_n_ahv21,
                                      mutationsrate_n),
            mutationsrate_guenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                             (1 - min(1, (lJahr - 2024) / 4)) *
↪  mutationsrate_guenstig +
                                               min(1, (lJahr - 2024) / 4) *
                                                ↪  mutationsrate_guenstig_ahv21,
                                              mutationsrate_guenstig),
            mutationsrate_unguenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                               (1 - min(1, (lJahr - 2024) / 4)) *
↪  mutationsrate_unguenstig +
                                                 min(1, (lJahr - 2024) / 4) *
                                                  ↪  mutationsrate_unguenstig_ahv21,
                                               mutationsrate_unguenstig),
            mutationsrate_n_guenstig = if_else(alt == 64 & sex == "f" & lJahr >= 2025,
                                               (1 - min(1, (lJahr - 2024) / 4)) *
↪  mutationsrate_n_guenstig +
                                                 min(1, (lJahr - 2024) / 4) *
                                                  ↪  mutationsrate_n_guenstig_ahv21,
                                               mutationsrate_n_guenstig),
            mutationsrate_n_unguenstig = if_else(alt == 64 & sex == "f" & lJahr >=
             ↪  2025,
                                                  (1 - min(1, (lJahr - 2024) / 4)) *
↪  mutationsrate_n_unguenstig +
                                                    min(1, (lJahr - 2024) / 4) *
                                                     ↪  mutationsrate_n_unguenstig_ahv21,
                                                  mutationsrate_n_unguenstig)
          )
```

```
)
```

Konkret setzten wir die Mutationsrate für Frauen mit Alter Ende Jahr 65 auf -1 und die Neurentenquote für diese Frauen auf 0. Dies ist per definition korrekt, da alle Frauen mit Alter 65 Ende Jahr auch nach Einführung der AHV21 am Ende des Jahres in die AHV übergegangen sein werden. Bezüglich der Frauen mit Alter 64 Ende Jahr (für welche die vorangehenden Überlegungen vor Einführung der AHV21 zutreffen) nehmen wir an, dass sich Frauen mit Alter Ende Jahr 64 nach Einführung der AHV21 gleich verhalten werden wie Frauen mit Alter Ende Jahr 63. Das heisst, wir nehmen die Neurentenquote der 63-jährigen Frauen für die 64-jährigen Frauen an, wobei wir berücksichtigen, dass die AHV21 schrittweise von 2025 bis 2028 eingeführt wird.

Mit dem nachfolgenden Code wird der Bestand an Rentenbeziehenden projiziert:

```
RENTENBESTAND_NEU[is.na(RENTENBESTAND_NEU)] <- 0

RENTENBESTAND_NEU <- RENTENBESTAND_NEU |>
  mutate(
    # Mittelwert
    neurenten = (bevendejahr - bestand_n) * neurentenquote,
    neurenten_n = (bevendejahr - bestand_n) * neurentenquote_n,
    bestandesmutationen = bestand * mutationsrate,
    bestandesmutationen_n = bestand_n * mutationsrate_n,
    bestand = bestand * (1 + mutationsrate) + neurenten,
    bestand_n = bestand_n * (1 + mutationsrate_n) + neurenten_n,

    # Lower
    neurenten_guenstig = (bevendejahr - bestand_n_guenstig) *
    ↪  neurentenquote_guenstig,
    neurenten_n_guenstig = (bevendejahr - bestand_n_guenstig) *
    ↪  neurentenquote_n_guenstig,
    bestandesmutationen_guenstig = bestand_guenstig * mutationsrate_guenstig,
    bestandesmutationen_n_guenstig = bestand_n_guenstig * mutationsrate_n_guenstig,
    bestand_guenstig = bestand_guenstig * (1 + mutationsrate_guenstig) +
    ↪  neurenten_guenstig,
    bestand_n_guenstig = bestand_n_guenstig * (1 + mutationsrate_n_guenstig) +
    ↪  neurenten_n_guenstig,

    # Upper
    neurenten_unguenstig = (bevendejahr - bestand_n_unguenstig) *
    ↪  neurentenquote_unguenstig,
    neurenten_n_unguenstig = (bevendejahr - bestand_n_unguenstig) *
    ↪  neurentenquote_n_unguenstig,
    bestandesmutationen_unguenstig = bestand_unguenstig * mutationsrate_unguenstig,
    bestandesmutationen_n_unguenstig = bestand_n_unguenstig *
    ↪  mutationsrate_n_unguenstig,
    bestand_unguenstig = bestand_unguenstig * (1 + mutationsrate_unguenstig) +
    ↪  neurenten_unguenstig,
    bestand_n_unguenstig = bestand_n_unguenstig * (1 + mutationsrate_n_unguenstig)
    ↪  + neurenten_n_unguenstig
  ) |>
  select(-bevendejahr, -contains("neurentenquote"), -contains("mutationsrate")) |>
  mutate(quelle = "Projektion")
```

```
    RENTENBESTAND <- RENTENBESTAND |>
      bind_rows(RENTENBESTAND_NEU)
  }
```

Die Berechnung der Bestände an Renten und Neurenten in den Projektionsjahren ist analog zur in Abschnitt 4.16.1 erläuterten Berechnung der Bestände an Renten und Neurenten aus den Registerdaten.

Zum Schluss werden mit dem folgenden Code die relevanten Variablen umbenennt und ausgewählt:

```
RENTENBESTAND_IV <- RENTENBESTAND |>
    mutate(bestand = case_when(PARAM_GLOBAL$szenario_fhh == "referenz" ~
        bestand, PARAM_GLOBAL$szenario_fhh == "tief" ~ bestand_unguenstig,
        PARAM_GLOBAL$szenario_fhh == "hoch" ~ bestand_guenstig),
        neurenten = case_when(PARAM_GLOBAL$szenario_fhh == "referenz" ~
            neurenten, PARAM_GLOBAL$szenario_fhh == "tief" ~
            neurenten_unguenstig, PARAM_GLOBAL$szenario_fhh ==
            "hoch" ~ neurenten_guenstig), bestandesmutationen =
            ↪  case_when(PARAM_GLOBAL$szenario_fhh ==
            "referenz" ~ bestandesmutationen, PARAM_GLOBAL$szenario_fhh ==
            "tief" ~ bestandesmutationen_unguenstig, PARAM_GLOBAL$szenario_fhh ==
            "hoch" ~ bestandesmutationen_guenstig), bestand_personen =
            ↪  case_when(PARAM_GLOBAL$szenario_fhh ==
            "referenz" ~ bestand_n, PARAM_GLOBAL$szenario_fhh ==
            "tief" ~ bestand_n_unguenstig, PARAM_GLOBAL$szenario_fhh ==
            "hoch" ~ bestand_n_guenstig), neurenten_personen =
            ↪  case_when(PARAM_GLOBAL$szenario_fhh ==
            "referenz" ~ neurenten_n, PARAM_GLOBAL$szenario_fhh ==
            "tief" ~ neurenten_n_unguenstig, PARAM_GLOBAL$szenario_fhh ==
            "hoch" ~ neurenten_n_guenstig), bestandesmutationen_personen =
            ↪  case_when(PARAM_GLOBAL$szenario_fhh ==
            "referenz" ~ bestandesmutationen_n, PARAM_GLOBAL$szenario_fhh ==
            "tief" ~ bestandesmutationen_n_unguenstig, PARAM_GLOBAL$szenario_fhh ==
            "hoch" ~ bestandesmutationen_n_guenstig)) |>
    select(jahr, sex, alt, bestand, neurenten, bestandesmutationen,
        bestand_personen, neurenten_personen, bestandesmutationen_personen)
```

4.16.4 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 - PARAM_GLOBAL$MA_years, jahr <
        jahr_projektion) %>%
    left_join(RENTENBESTAND_IV %>%
        group_by(jahr) %>%
        summarize(bestand = sum(bestand), neurenten = sum(neurenten),
            bestandesmutationen = sum(bestandesmutationen)) %>%
```

```
        ungroup() %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
            minimalrente * 12, bestandesmutationen = bestandesmutationen *
            minimalrente * 12) %>%
        filter(jahr >= jahr_projektion - PARAM_GLOBAL$MA_years,
            jahr < jahr_projektion)) %>%
    mutate(frac_rent_ord = rent_ord/(bestand - 0.5 * neurenten +
        0.5 * (-bestandesmutationen)), frac_rent_aord = rent_aord/(bestand -
        0.5 * neurenten + 0.5 * (-bestandesmutationen)), frac_rent_ord_nachz =
        ↪  rent_ord_nachz/neurenten,
        frac_rent_aord_nachz = rent_aord_nachz/neurenten, frac_ausl_rueck =
        ↪  ausl_rueck/bestand,
        frac_ausl_fs_leist = ausl_fs_leist/bestand, frac_rueck_erstattungsf =
        ↪  rueck_erstattungsf/bestand,
        frac_abschr_rueck_erstattungsf = abschr_rueck_erstattungsf/bestand) %>%
    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 Jahre, über welche wir auch die Neurentenquote und Abgangsraten schätzen, 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 `PARAM_GLOBAL$MA_years` Jahre, und fügen den gesamten Bestand an Renten und Neurenten (d.h. aggregiert über alle Alter und Geschlechter) an, wobei wir den Rentenbestand und die Neurenten in Jahresrenten in CHF umrechnen (mit den Formeln `bestand=bestand*minimalrente*12` respektive `neurenten=neurenten*minimalrente*12`). Danach berechnen wir für die verschiedenen Rentenausgabenpositionen (`rent_ord`, `rent_aord`, ...), welchen Anteil an den Rentenausgaben gemäss den Registerdaten diese in den jeweiligen Jahren ausgemacht haben. Eine Ausnahme bilden die Rentennachzahlungen. Rentennachzahlungen werden zum allergrössten Teil bei Neurentnern geleistet. Aus diesem Grund bilden wir die Anteile für die Rentennachzahlungen (`frac_rent_ord_nachz` und `frac_rent_aord_nachz`) als Anteil der Ausgaben für Neurenten in einem Jahr.[31] Zum Schluss bilden wir den Mittelwert der Anteile über die letzten `PARAM_GLOBAL$MA_years` 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),
        bestandesmutationen = sum(bestandesmutationen)) %>%
    ungroup() %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
```

[31] 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).

```
        minimalrente * 12, bestandesmutationen = bestandesmutationen *
        minimalrente * 12) %>%
    mutate(rent_ord = (bestand - 0.5 * neurenten + 0.5 * (-bestandesmutationen)) *
        ANTEILE$frac_rent_ord, rent_aord = (bestand - 0.5 * neurenten +
        0.5 * (-bestandesmutationen)) * ANTEILE$frac_rent_aord,
        rent_ord_nachz = neurenten * ANTEILE$frac_rent_ord_nachz,
        rent_aord_nachz = neurenten * ANTEILE$frac_rent_aord_nachz,
        ausl_rueck = bestand * ANTEILE$frac_ausl_rueck, ausl_fs_leist = bestand *
            ANTEILE$frac_ausl_fs_leist, rueck_erstattungsf = bestand *
            ANTEILE$frac_rueck_erstattungsf, abschr_rueck_erstattungsf = bestand *
            ANTEILE$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))
↪   %>%
    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.

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:

```
return(list(RENTENSUMME_IV = RENTENSUMME_IV, RENTENBESTAND_IV = RENTENBESTAND_IV,
    BESTAND_IV = BESTAND_IV))
```

4.17 Modul mod_iv_taggelder.R

Dokumentation zuletzt aktualisiert am 20.02.2025

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`).

Im Modul werden die Taggelder-Ausgaben aus dem letzten Abrechnungsjahr mit der Wachstumsrate der Lohnsumme fortgeschrieben. Die Berechnung wird in folgendem Codeblock durchgeführt:

```
# Fortschreibung der Taggelder mit Lohnsumme
  TAGGELDER_IV  <- FORTSCHREIBUNG_IV %>%
      select(jahr, lohnsumme) %>%
      filter(jahr>=min(IV_ABRECHNUNG$jahr)) %>%
      left_join(IV_ABRECHNUNG %>% select(jahr, tg_geld)) %>%
      mutate(
          cum_lohnsummengrowth = cumprod(if_else(jahr>PARAM_GLOBAL$jahr_abr, 1 +
          ↪  lohnsumme / 100, 1)),

          ↪  tg_geld=ifelse(jahr>PARAM_GLOBAL$jahr_abr,tg_geld[jahr==PARAM_GLOBAL$jahr_abr]*cum_lohnsu

      ) %>%
      select(jahr, tg_geld)
```

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:

```
return(TAGGELDER_IV = TAGGELDER_IV)
```

4.18 Modul mod_iv_he.R

Dokumentation zuletzt aktualisiert am 18.03.2025

In diese Kapitel wird die technische Umsetzung des Modells für die Hilflosenentschädigungen und den Intensivpflegezuschlag diskutiert. Um die Erläuterungen hier zu verstehen sollte vorgängig der Abschnitt zu den Hilflosenentschädigungen, respektive den Renten, im Dokument „Modellbeschrieb_IV“ gelesen werden.

Das Modul zu den Ausgaben für Hilflosenentschädigungen und den Intensivpflegezuschlag `mod_iv_he.R` ist analog zum Modul `mod_iv_rentensumme.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.R` die gleiche Struktur hat wie der Code in `mod_iv_rentensumme.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(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEVOELKERUNG = BEVOELKERUNG, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    IV_HE_ERWACHSENE = IV_HE_ERWACHSENE, IV_HE_KINDER = IV_HE_KINDER)
```

Das Modul `mod_iv_he.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`: 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 wird in `mod_iv_he` der Parameter `jahr_projektion` gesetzt:[32]

```
jahr_projektion <- PARAM_GLOBAL$jahr_abr + 1
```

4.18.1 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:

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

```
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_ERWACHSENE %>%
    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)

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.

4.18.2 Berechnung der HE-Neurentenquote und der HE-Mutationsrate

Die HE-Neurentenquote und die HE-Mutationsraten werden wie folgt berechnet:

```
RATEN_INV_MUT_HE_KINDER <- BESTAND_HE_KINDER %>%
    left_join(bind_rows(BEVOELKERUNG %>%
        mutate(bevendejahr = (bevendejahr + frontaliers)/2) %>%
        filter(alt == 0), BEVOELKERUNG %>%
        mutate(jahr = jahr + 1, alt = alt + 1)) %>%
        group_by(jahr, sex, alt) %>%
        summarize(bevendejahr = sum(bevendejahr, na.rm = TRUE) +
            sum(frontaliers, na.rm = TRUE))) %>%
    mutate(neurentenquote = neurenten/(bevendejahr - bestand_vj_n),
        neurentenquote_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)) %>%
    filter(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(neurentenquote = mean(neurentenquote, na.rm = TRUE),
        mutationsrate = mean(mutationsrate, na.rm = TRUE), neurentenquote_n =
        ↪  mean(neurentenquote_n,
            na.rm = TRUE), mutationsrate_n = mean(mutationsrate_n,
            na.rm = TRUE), ) %>%
    ungroup()

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) +
            sum(frontaliers, na.rm = TRUE))) %>%
    mutate(neurentenquote = neurenten/(bevendejahr - bestand_vj_n),
        neurentenquote_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)) %>%
    filter(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(neurentenquote = mean(neurentenquote, na.rm = TRUE),
        mutationsrate = mean(mutationsrate, na.rm = TRUE), neurentenquote_n =
        ↪  mean(neurentenquote_n,
            na.rm = TRUE), mutationsrate_n = mean(mutationsrate_n,
            na.rm = TRUE), ) %>%
    ungroup()
```

Die HE-Neurentenquote 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 hier relevante Bevölkerung, das heisst die Population der Rentenversicherten, ist die Wohnbevölkerung zuzüglich der Grenzgänger.[33]

Die HE-Neurentenquote 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.

4.18.3 HE-Bestand projizieren

Die Projektion der zukünftigen HE-Bestände ist 1:1 analog zur Projektion im Modell für die Renten (vgl. Kapitel 4.16).

4.18.4 HE-Ausgaben projizieren

Um die Projektionen für die gesamten HE-Ausgaben zu erstellen, werden die Bestandes- und Flussgrössen für die Erwachsenen-HE und die Kinder-HE separat nach Jahr aggregiert (nachfolgend am Beispiel der Erwachsenen-HE):

```
HE_BESTAND_ERWACHSENE <- HE_BESTAND_ERWACHSENE %>%
    rename(bestand_personen = bestand_n, neurenten_personen = neurenten_n,
        bestandesmutationen_personen = bestandesmutationen_n) %>%
```

[33] Grundsätzlich ist die IV eine Erwerbsausfallversicherung, was nahelegen würde, die Inland-Erwerbsbevölkerung als Versichertenpopulation zu wählen. Auf Grund von Mitversicherungen von Ehepartnern, sowie ausserordentlichen Rentenansprüchen, zählen viele Personen, die Teil der Wohnbevölkerung und nicht Teil der Erwerbsbevölkerung sind, auch zur Versichertenpopulation. Wir erachten daher die Wohnbevölkerung zuzüglich der Grenzgänger als die bessere Approximation der Versichertenpopulation als die Inländer-Erwerbsbevölkerung zuzüglich der Grenzägnger (=Inland-Erwerbsbevölkerung).

```
    select(jahr, sex, alt, bestand, neurenten, bestandesmutationen,
        bestand_personen, neurenten_personen, bestandesmutationen_personen)

BESTAND_IV <- HE_BESTAND_ERWACHSENE %>%
    group_by(jahr) %>%
    summarize(bestand = sum(bestand), neurenten = sum(neurenten),
        bestandesmutationen = sum(bestandesmutationen))
```

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

```
ANTEILE <- IV_ABRECHNUNG %>%
    filter(jahr >= jahr_projektion - PARAM_GLOBAL$MA_years, jahr <
        jahr_projektion) %>%
    left_join(BESTAND_IV %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
            minimalrente * 12, bestandesmutationen = bestandesmutationen *
            minimalrente * 12) %>%
        filter(jahr >= jahr_projektion - PARAM_GLOBAL$MA_years,
            jahr < jahr_projektion)) %>%
    left_join(BESTAND_IV_KINDER %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
            minimalrente * 12, bestandesmutationen = bestandesmutationen *
            minimalrente * 12) %>%
        rename(bestand_kinder = bestand, neurenten_kinder = neurenten,
            bestandesmutationen_kinder = bestandesmutationen) %>%
        filter(jahr >= jahr_projektion - PARAM_GLOBAL$MA_years,
            jahr < jahr_projektion)) %>%
    mutate(frac_he_nachz = (he - (bestand + bestand_kinder -
        0.5 * (neurenten + neurenten_kinder) + 0.5 * (-bestandesmutationen +
        (-bestandesmutationen_kinder))))/(bestand + bestand_kinder -
        0.5 * (neurenten + neurenten_kinder) + 0.5 * (-bestandesmutationen +
        (-bestandesmutationen_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 unter Berücksichtigung der unterjährigen Zu- und Abgänge. 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 `PARAM_GLOBAL$MA_years` Jahre entspricht. Tatsächlich ist die Abweichung gering und schwankt nur wenig 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, bestand, neurenten, bestandesmutationen) %>%
    left_join(RENTENENTWICKLUNG %>%
        select(jahr, minimalrente)) %>%
    mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
        minimalrente * 12, bestandesmutationen = bestandesmutationen *
        minimalrente * 12) %>%
    left_join(BESTAND_IV_KINDER %>%
        select(jahr, bestand, neurenten, bestandesmutationen) %>%
        left_join(RENTENENTWICKLUNG %>%
            select(jahr, minimalrente)) %>%
        mutate(bestand = bestand * minimalrente * 12, neurenten = neurenten *
            minimalrente * 12, bestandesmutationen = bestandesmutationen *
            minimalrente * 12) %>%
        rename(bestand_kinder = bestand, neurenten_kinder = neurenten,
            bestandesmutationen_kinder = bestandesmutationen)) %>%
    mutate(nachzahlungen = (bestand + bestand_kinder - 0.5 *
        (neurenten + neurenten_kinder) + 0.5 * (-bestandesmutationen +
        (-bestandesmutationen_kinder))) * ANTEILE$frac_he_nachz) %>%
    mutate(renten_total = bestand + bestand_kinder + nachzahlungen)
```

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 `bestand + bestand_kinder + nachzahlungen` die Variable `renten_total`, welche die totalen Ausgaben für HE darstellt.

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), RENTEN_IV %>%
                               filter(jahr>PARAM_GLOBAL$jahr_abr) %>%
                               group_by(jahr) %>%
                               summarize(he=sum(renten_total, na.rm = TRUE)) %>%
                               ungroup()
                           )

    })
    #------ Output --------------------------------------------------------#
    return(
        list(
            HE_IV = HE_IV,
            HE_BESTAND_KINDER = HE_BESTAND_KINDER,
            HE_BESTAND_ERWACHSENE = HE_BESTAND_ERWACHSENE
        )
    )
```

4.19 Modul mod_iv_mm.R

Dokumentation zuletzt aktualisiert am 17.03.2025

In diese Kapitel wird die technische Umsetzung des Modells für die medizinischen Massnahmen diskutiert. Um die Erläuterungen hier zu verstehen sollte vorgängig der Abschnitt zu den medizinischen Massnahmen im Dokument „Modellbeschrieb_IV“ gelesen werden.

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

```
tl_iv_mm <- mod_iv_mm(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.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.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)
```

4.19.1 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.

4.19.2 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.[34]
- 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 Jahren ab dem Sondereffektjahr als Schätzer für die Invalidisierung in den Jahren ab dem Sondereffektjahr.[35]

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 repräsentativ sind für die Zeit ab 2022. Auf Grund von Rechnungsverzögerungen kann aber auch nicht davon

[34] Für das Jahr, in welchem der temporäre Sondereffekt vermerkt ist, wird die Invalidisierung im Sondereffektjahr verwendet.
[35] 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.

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 4.19.8).

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))
        ) %>%
        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)
    )
```

4.19.3 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 4.19.2): 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.

4.19.4 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.

4.19.5 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)
```

4.19.6 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.

4.19.7 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).

4.19.8 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.[36] 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 ------------------------------------------------------#
return(list(IV_MED_MASSN_FORECAST_AVERAGES = IV_MED_MASSN_FORECAST_AVERAGES,
    IV_MM = IV_MM))
```

[36] 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.

4.20 Modul mod_iv_fi.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -------------------------------------------------------#
return(IV_FI = IV_FI)
```

4.21 Modul mod_iv_ber_begl.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 --------------------------------------------------------#
return(IV_BER_BEGL = IV_BER_BEGL)
```

4.22 Modul mod_iv_im.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -------------------------------------------------------#
return(IV_IM = IV_IM)
```

4.23 Modul mod_iv_mba.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 --------------------------------------------------------#
return(IV_MBA = IV_MBA)
```

4.24 Modul mod_iv_aus_uebr.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 --------------------------------------------------------#
return(IV_AUS_UEBR = IV_AUS_UEBR)
```

4.25 Modul mod_iv_hm.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -----------------------------------------------------#
return(IV_HM = IV_HM)
```

4.26 Modul mod_iv_rk.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -----------------------------------------------------------#
return(IV_RK = IV_RK)
```

4.27 Modul mod_iv_assb.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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_ASSB_PROJ <- IV_GELDLEISTUNG %>%
    select(jahr, he) %>%
    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/startwert)) %>%
    select(-he)
```

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 --------------------------------------------------------#
return(IV_ASSB = IV_ASSB)
```

4.28 Modul mod_iv_rueck_im.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -------------------------------------------------------#
return(IV_RUECK_IM = IV_RUECK_IM)
```

4.29 Modul mod_iv_institutionen.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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 -------------------------------------------------------#
return(IV_BTR_ORG = 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)
```

Somit können die Durchführungskosten an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output -------------------------------------------------------#
return(IV_INSTITUTIONEN = IV_INSTITUTIONEN)
```

4.30 Modul mod_iv_durchfuehrungskosten.R

Dokumentation zuletzt aktualisiert am 12.11.2024

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)
```

Somit können die Durchführungskosten an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output --------------------------------------------------------#
return(IV_DURCHFUEHRUNG = IV_DURCHFUEHRUNG)
```

4.31 Modul mod_iv_verwaltungsaufwand.R

Dokumentation zuletzt aktualisiert am 07.05.2025

Das Prognosemodell zum Verwaltungsaufwand ist im Skript `mod_iv_verwaltungsaufwand.R` enthalten, welches wie folgt im Skript `mod_iv_uebrigeausgaben.R` aufgerufen wird:

```
IV_VERWALTUNGSAUFWAND <- mod_iv_verwaltungsaufwand(PARAM_GLOBAL = PARAM_GLOBAL,
    UMFRAGE_IV = UMFRAGE_IV, IV_ABRECHNUNG = IV_ABRECHNUNG, FORTSCHREIBUNG_IV =
    ↪  FORTSCHREIBUNG_IV,
    DISKONTFAKTOR = DISKONTFAKTOR)
```

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.
- `DISKONTFAKTOR`: Der Diskontfaktor benötigt, um die Fortschreibung um das Wachstum des Landesindexes der Konsumentenpreise zu skalieren.

Im Modul `mod_iv_verwaltungsaufwand.R` werden nacheinander die folgenden Untermodule aufgerufen:

```
#------ Posttaxen / Taxes postales ------------------------------------#
IV_POSTTAX <- mod_iv_posttax(PARAM_GLOBAL = PARAM_GLOBAL, IV_ABRECHNUNG = IV_ABRECHNUNG,
    DISKONTFAKTOR = DISKONTFAKTOR)

#------ 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 ausser dem Modul `mod_iv_posttax`, in welchem lediglich der letzte Wert der Abrechnung mit der Inflation fortgeschrieben wird, gleich aufgebaut sind, wird hier die Struktur dieser Module nur am Beispiel des Moduls `mod_iv_verwaltungskosten.R` erläutert:

Zu Beginn des Moduls `mod_iv_verwaltungskosten` wird der betreffende Vektor der IV-Abrechnung ausgewählt, um diesen am Ende des Moduls dann an den Projektionsvektor anzufügen:

```
#------ Werte der Abrechnung ------------------------------------------#
IV_VERWK_ABR <- IV_ABRECHNUNG %>%
    select(jahr, verw_kost)
```

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("verw_kost")])
if (is.na(last_umfr_val)) {
    warnings(paste("fehlender Wert für die Verwaltungskosten 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_VERWK_PROJ <- FORTSCHREIBUNG_IV %>%
    select(jahr, lohnsummeidx) %>%
    filter(jahr > PARAM_GLOBAL$jahr_abr) %>%
    left_join(UMFRAGE_IV %>%
        select(jahr, verw_kost), by = c("jahr")) %>%
    mutate(verw_kost = if_else(jahr < PARAM_GLOBAL$jahr_umfragen_fs,
```

```
        verw_kost, last_umfr_val * lohnsummeidx)) %>%
    select(-lohnsummeidx)
```

dh., es wird die Lohnsumme für die Fortschreibung der Ausgaben nach der letzten Umfrageprojektion verwendet. Dieser ist in der Variable `lohnsummeidx` 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 Verwaltungskosten ---------------------------------------#
IV_VERWALTUNGSKOSTEN <- rbind(IV_VERWK_ABR, IV_VERWK_PROJ)

#------ Output --------------------------------------------------------#
return(IV_VERWALTUNGSKOSTEN = IV_VERWALTUNGSKOSTEN)
```

Die anderen oben erwähnten Module sind identisch aufgebaut, wobei bei `mod_iv_imm_absch.R` anstatt des Lohnsummenindex der Lohnindex 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)
```

Somit kann der Verwaltungsaufwand an das Finanzperspektivenmodell (respektive das Modul `mod_iv_uebrigeausgaben.R`) zurückgegeben werden:

```
#------ Output -------------------------------------------------------#
return(IV_VERWALTUNGSAUFWAND = IV_VERWALTUNGSAUFWAND)
```

4.32 Modul mod_iv_beitrag.R und mod_beitragssumme.R

Dokumentation zuletzt aktualisiert am 27.07.2025

In diese Kapitel wird die technische Umsetzung des Modells für die Beiträge von Versicherten und Arbeitnehmern diskutiert. Um die Erläuterungen hier zu verstehen sollte vorgängig der Abschnitt zu den Beiträgen von Versicherten und Arbeitnehmern im Dokument „Modellbeschrieb_IV“ gelesen 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,
    BEVOELKERUNG = BEVOELKERUNG, BEV_BESTAND = BEV_BESTAND, IK = IK,
    ECKWERTE_EXTENDED = ECKWERTE_EXTENDED, ECKWERTE_SCENARIO = ECKWERTE_SCENARIO,
    AHV_ABRECHNUNG = AHV_ABRECHNUNG, IV_ABRECHNUNG = IV_ABRECHNUNG,
    BEITRAGSSAETZE = BEITRAGSSAETZE, ARBEITSLOSENQUOTE = ARBEITSLOSENQUOTE)
```

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

- `BEVOELKERUNG`: Aus diesem Dataframe werden die Zeitreihen für die Inländer-Erwerbsbevölkerung sowie die Grenzgänger verwendet, welche zusammen die Erwerbsbevökerung im Inland ergeben.
- `BEV_BESTAND`: Dieses Dataframe wird einzig dazu verwendet, zu bestimmen, wann die letzten verfügbaren Bestandesdaten für die Erwerbsbevölkerung vorliegen.
- `IK`: Die Daten gemäss der individuellen Konten der ZAS.
- `ECKWERTE_EXTENDED`: Daten zur Entwicklung der Preise und des SLI.
- `ECKWERTE_SCENARIO`: Aus diesem Dataframe nutzen wir die Projektionen zur Arbeitslosenquote und dem Wachstum der Erwerbstätigen gemäss BESTA für die Anpassungen in der kurzen Frist. Ausserdem wird das Dataframe dazu verwendet, zu bestimmen, welche Jahre durch das Eckwerte-Szenario der ESTV abgedeckt sind.
- `AHV_ABRECHNUNG`: Da das Modell für die Lohnbeiträge für alle Versicherungen (i.e., AHV, IV und EO) basierend auf den Beiträgen an die AHV geschätzt wird, und die Beiträge für die anderen Versicherungen dann durch Umrechnung des Beitragssatzes berechnet werden, wird hier die AHV Abrechnung benötigt.
- `IV_ABRECHNUNG`: Die IV Abrechnung wird verwendet, um die Abweichung der Projektionen von den Abrechnungsdaten zu berechnen, und für diese zu korrigieren.
- `BEITRAGSSAETZE`: Die Beitragssätze für die AHV und IV werden benötigt, um bei der Schätzung des Modells für den Strukturfaktor sowie der Projektionen für allfällige Beitragssatzänderungen zu korrigieren.
- `ARBEITSLOSENQUOTE`: Die historische Arbeitslosenquote wird für die Anpassungen in der kurzen Frist verwendet (siehe `ECKWERTE_SCENARIO` oben).

4.32.1 Modul mod_beitragssumme.R

Als erstes wird das Modul `mod_beitragssumme.R` aufgerufen:

```
tl_iv_beitrag <- mod_beitragssumme(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, BEV_BESTAND = BEV_BESTAND, IK = IK,
    ECKWERTE_EXTENDED = ECKWERTE_EXTENDED, ECKWERTE_SCENARIO = ECKWERTE_SCENARIO,
```

```
    AHV_ABRECHNUNG = AHV_ABRECHNUNG, BEITRAGSSAETZE = BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = ARBEITSLOSENQUOTE, versicherung = "iv",
    ABRECHNUNG = IV_ABRECHNUNG)
```

Die Parameter `versicherung = ïv"` und `ABRECHNUNG = IV_ABRECHNUNG` stellen sicher, dass die Beiträge für die Invalidenversicherung geschätzt werden (zur Schätzung der Beiträge für die AHV oder EO müsste die Versicherung entsprechend auf `ahv` oder `eo` gesetzt werden, respektive die `AHV_ABRECHNUNG` oder die `EO_ABRECHNUNG` dem `ABRECHNUNG` Dataframe zugewiesen werden.).

Das Modul `mod_beitragssumme.R` beginnt mit der folgenden Überprüfung:

```
## SAFETY-CHECKS

# Bestimme das maximale Jahr aus BEV_BESTAND, bei dem ept
# nicht NA ist
max_jahr_ept_bestand <- BEV_BESTAND |>
    filter(!is.na(ept)) |>
    summarize(max_jahr = max(jahr)) |>
    pull(max_jahr)

# Bestimme das minimale Jahr aus ECKWERTE_SCENARIO, bei dem
# besta nicht NA ist
min_jahr_besta_projektion <- ECKWERTE_SCENARIO |>
    filter(!is.na(besta)) |>
    summarize(min_jahr = min(jahr)) |>
    pull(min_jahr)

# Überprüfe die Bedingung und stoppe ggf. die Ausführung
# mit einer Fehlermeldung
if (min_jahr_besta_projektion > (max_jahr_ept_bestand + 1)) {
    stop(paste0("Erwerbsbevölkerung ist Stand ", max_jahr_ept_bestand,
        " und BESTA-Projektion startet in ", min_jahr_besta_projektion,
        ". Bitte Erwerbsbevölkerung auf Stand ", min_jahr_besta_projektion -
            1, " bringen, oder Jahr ", min_jahr_besta_projektion -
            1, " zu Eckwerten hinzufügen."))
}
```

Mit dieser Überprüfung wird sichergestellt, dass die beobachtete Erwerbsbevölkerung bis ins Jahr vor der ersten BESTA-Projektion aus den ESTV-Eckwerten reicht. Damit wird sichergestellt, dass wir durch die Verwendung der BESTA-Projektionen korrekt für kurzfristige, nicht von den BFS-Szenarien abgedeckte, Anpassungen auf dem Arbeitsmarkt korrigieren. Die Bedingung kann bei rechtzeitiger Datenaktualisierung immer erfüllt werden.

Als nächstes wird überprüft, ob die BESTA-Beschäftigungsprognose Wachstumsraten von absolut gesehen mehr als 2% enthält:

```
# Auf grosse Veränderung bei der Beschäftigung überprüfen
max_besta <- ECKWERTE_SCENARIO |>
    summarize(max_besta = max(abs(besta), na.rm = TRUE)) |>
    pull(max_besta)

if (max_besta > 2) {
```

```
    message("Warnung: BESTA-Prognose sieht Absolutwert der Beschäftigungsveränderung von
    ↪  mehr als 2% vor. Überprüfen ob BESTA-Korrektur bei Lohnbeiträgen angemessen ist
    ↪  (siehe Dokumentation Lohnbeiträge).")
}
```

Ist dies der Fall wird eine Warnung ausgegeben, da dies auf eine volkswirtschaftlich ausserordentliche Sitautation hindeutet, bei welcher insbesondere überprüft werden soll, ob der angenommene direkte Übergang in t+2 vom (kurzfristigen) Erwerbsbevölkerungswachstum gemäss BESTA-Projektion zum (mittel- bis langfristigen) Erwerbsbevölkerungswachstum gemäss BFS-Szenarien gerechtfertigt ist. Es geht um die Frage, ob der Arbeitsmarkt nach t+2 bereits wieder im Gleichgewicht ist, oder ob ein Rückführungspfad angenommen werden muss.

Es folgt das eigentliche Modell zur Schätzung der Beiträge. Hierbei wird als erstes die Zeitspanne definiert, über welche der Strukturfaktor geschätzt werden soll:

```
# PROJEKTION LOHNSUMME UND BEITRAGSSUMME

est_struktur_start = max(IK$jahr) - 19
est_struktur_end = max(IK$jahr)
```

Als nächstes wird die für die Schätzung des Modells massgebende Erwerbsbevölkerung im Inland berechnet:

```
  # INLAND-ERWERBSBEVÖLKERUNG BERECHNEN
   EPT_INLAND <- BEVOELKERUNG %>%
       filter(jahr>=1996, alt>=18) %>%
       mutate(ept_inland=
                  case_when(
           alt<=65 & sex=="m" ~ ept+0.9*frontaliers, # Annahme: Männliche Grenzgänger
↪  arbeiten im Durchschnitt 90%
           alt<=65 & sex=="f" ~ ept+0.7*frontaliers, # Annahme: Weibliche Grenzgänger
↪  arbeiten im Durchschnitt 70%
           alt>65 & sex=="m" ~ 0.5*(ept+0.9*frontaliers), # Annahme: Lohnbeitragsquote
↪  der über 65-jährigen ist nur 50% der Lohnbeitragsquote der unter-65 Jährigen (wegen
↪  Beitragsbefreiung)
           alt>65 & sex=="f" ~ 0.5*(ept+0.7*frontaliers)
           )
       )  %>%
       group_by(jahr, sex) %>%
       summarize(ept_inland=sum(ept_inland, na.rm = TRUE))
```

Hierbei handelt es sich um die Kombination der Inländer-Erwerbsbevölkerung in Vollzeitäquivalenten (Variable `ept` im `BEVOELKERUNG` Dataframe) und der Grenzgänger (Variable `frontaliers` im `BEVOELKERUNG` Dataframe), wobei wir für die Grenzgänger nur über Daten und Szenarien zur Anzahl Personen, jedoch und nicht zur Anzahl Vollzeitäquivalenten, verfügen. Wir nehmen daher für die Konstruktion der Inländer-Erwerbsbevölkerung an, dass die männlichen Grenzgänger einen durchschnittlichen Beschäftigungsgrad von 90% aufweisen, und die weiblichen Grenzgänger einen durchschnittlichen Beschäftigungsgrad von 70%.[37] Zusätzlich gilt die Annahme, dass sämtliche Inländer-Erwerbspersonen auch im Inland erwerbstätig sind, also dass es keine Grenzgänger von der Schweiz ins Ausland gibt (d.h. in der Schweiz leben und im Ausland

[37] Basierend auf Favre, Föllmi, Zweimüller (2021) "Einkommensentwicklung von Grenzgängerinnen und Grenzgängern im Aufenthaltsverlauf"

arbeiten) gibt.[38] Diese Annahme, respektive das Ausmass, zu welchem diese Annahme nicht erfüllt ist, erachten wir gegeben der mutmasslich sehr geringen Anzahl von Grenzgänger von der Schweiz ins Ausland als unbedeutend.

Im nächsten Schritt wird der Strukturfaktor geschätzt:

```
# STRUKTURFAKTOR BASIEREND AUF AHV-DATEN BERECHNEN
STRUKTURFAKTOR <- IK |>
    filter(
        vz=="ahv",
        jahr>=est_struktur_start
        ) |>
    group_by(jahr, sex) |>
    summarize(
        ahv_beitraege=sum(beitrag_bsv, na.rm = TRUE),
        .groups = "drop"
        ) |>
    left_join(EPT_INLAND %>%
                  group_by(jahr, sex) %>%
                  summarize(ept_inland=sum(ept_inland, na.rm = TRUE))
              ) %>%
    left_join(ECKWERTE_EXTENDED) %>%
    left_join(AHV_ABRECHNUNG |> select(jahr, btr_vs_ag)) |>
    group_by(jahr) |> # Gruppieren nach Jahr
    mutate( # Korrektur IK-Daten um die Abweichung von den Beiträgen gemäss
    ↪  Abrechnung
        sum_ahv_beitraege = sum(ahv_beitraege, na.rm = TRUE), # Summe der Beiträge
        ↪  berechnen
        fraction = sum_ahv_beitraege / btr_vs_ag,    # Bruch berechnen
        ahv_beitraege = ahv_beitraege / fraction   # Beiträge durch den Bruch teilen
    ) |>
    ungroup() |> # Gruppierung entfernen
    left_join(BEITRAGSSAETZE %>% select(jahr, ahv_vs_ag)) |>
    mutate(
        beitragsprozent=ahv_beitraege/(ahv_vs_ag*100)
    ) |>
    select(-sum_ahv_beitraege, -ahv_vs_ag, -btr_vs_ag, -fraction) |>
    group_by(sex) |> # Gruppieren nach sex
    mutate(
        delta_ept = (ept_inland / lag(ept_inland) - 1),
        delta_sli = ifelse(jahr==est_struktur_start,NA,lohn/100)
    ) |>
    mutate(

        ↪  strukturfaktor=beitragsprozent/(lag(beitragsprozent)*(1+delta_ept)*(1+delta_sli))-1
    ) |>
    group_by(sex) |>  # Gruppieren nach sex
    summarize(
        strukturfaktor = (prod(1 + strukturfaktor, na.rm = TRUE))^(1 /
        ↪  sum(!is.na(strukturfaktor))) - 1
    ) %>%
```

[38] Da das Modell lediglich auf Wachstumsraten der Erwerbsbevölkerung aufbaut ist die schwächere Annahme auserichend, dass der Anteil an Grenzgänger von der Schweiz ins Ausland an der durch uns berechneten Inland-Erwerbsbevölkerung über die Zeit konstant ist.

```
        mutate(
            strukturfaktor=strukturfaktor*100
        )
```

Dieser für das Verständnis des Beitragsmodells zentrale Codeblock wird nachfolgend detailliert Stück für Stück erläutert.

In einem ersten Schritt werden die AHV-spezifischen Daten aus den Daten zu den individuellen Konten (IK) gefiltert:

```
    # STRUKTURFAKTOR BASIEREND AUF AHV-DATEN BERECHNEN
    STRUKTURFAKTOR <- IK |>
        filter(
            vz=="ahv",
            jahr>=est_struktur_start
            ) |>
        group_by(jahr, sex) |>
        summarize(
            ahv_beitraege=sum(beitrag_bsv, na.rm = TRUE),
            .groups = "drop"
            ) |>
```

Dies ist notwendig, da in den IK für alle Versicherungen separate Daten zu den geleisteten Beiträgen vorhanden sind (es gibt also einen Separaten Eintrag für AHV, IV, und EO), wobei die Beträge jeweils nahezu mit dem Beitragssatz der jeweiligen Versicherung proportional zueinander sind.[39] Wir nutzen die AHV-Daten zur Schätzung der Strukturfaktors für alle Versicherungen, da wir damit sicherstellen, dass für alle Versicherungen der gleiche Strukturfaktor verwendet wird, und dadurch die projizierten Lohnbeiträge bei konstantem Beitragssatz mit der genau gleichen Rate über alle Versicherungen hinweg wachsen.

Als nächstes werden die Daten zur Inland-Erwerbsbevölkerung (`EPT_INLAND`), zur Lohn- und Preisentwicklung (`ECKWERTE_EXTENDED`), sowie die Beiträge der Versicherten und Arbeitnehmern aus den AHV Abrechnungsdaten (`btr_vs_ag` aus `AHV_ABRECHNUNG`) angefügt:

```
        left_join(EPT_INLAND %>%
                      group_by(jahr, sex) %>%
                      summarize(ept_inland=sum(ept_inland, na.rm = TRUE))
                  ) %>%
        left_join(ECKWERTE_EXTENDED) %>%
        left_join(AHV_ABRECHNUNG |> select(jahr, btr_vs_ag)) |>
        group_by(jahr) |> # Gruppieren nach Jahr
        mutate( # Korrektur IK-Daten um die Abweichung von den Beiträgen gemäss
        ↪  Abrechnung
            sum_ahv_beitraege = sum(ahv_beitraege, na.rm = TRUE), # Summe der Beiträge
            ↪  berechnen
            fraction = sum_ahv_beitraege / btr_vs_ag,     # Bruch berechnen
            ahv_beitraege = ahv_beitraege / fraction   # Beiträge durch den Bruch teilen
        ) |>
```

Im `mutate`-Befehl oben wird zuerst die Summe der gemäss IK-Daten entrichteten Beiträge von Versicherten und Arbeitnehmern (`ahv_beitraege`) gebildet. Danach wird die Abweichung der total in einem Jahr entrichteten Beiträge gemäss IK (`sum_ahv_beitraege`) von den Beiträgen gemäss AHV-Abrechnung (`btr_vs_ag`)

[39] Die Abweichungen von der Proportionalität zum Beitragssatz sind lediglich in den Nachkommastellen ersichtlich. Der Grund liegt mutmasslich darin, dass die Mindestbeiträge für Nichterwerbstätige und Selbständigerwerbende für die einzelnen Versicherungen auf Grund von Rundungen nicht genau proportional zum Beitragssatz sind.

berechnet, und in der Variable `fraction` gespeichert. Diese Abweichungen entstehen, da in den IK-Daten lediglich die definitv abgerechneten Beiträge vermerkt sind, während in der AHV-Abrechnung auch Akontozahlungen von nicht definitiv verfügten Beiträgen vermerkt sind. Dies ist insbesondere im Bezug auf Selbständigerwerbende relevant, für welche die definitiven Beitragszahlungen oft erst ein paar Jahre im nachhinein basierend auf dem steuerbaren Einkommen verfügt werden. Dies führt dazu, dass die Beitragszahlungen dieser Personen nicht in den hier genutzten IK-Daten enthalten sind (wir nutzen IK-Daten im Jahr t mit Stand t+2, das heisst, es wird immer der Datenbestand im März vom übernächsten Jahr für das betreffende Jahr abgebildet, und die Daten werden für die Konsistenz über die Zeit, nicht aktualisiert). Die Variable `fraction` schwankt über den Zeitraum 2002 bis 2022 zwischen 0.91 und 0.97. Das heisst, dass die Beiträge der Versicherten und Arbeitnehmern auf Grund des Einbezugs von Akontozahlungen gemäss AHV-Abrechnung (`btr_vs_ag`) um 3-9% höher sind als die Beiträge in den IK. Zum Schluss werden die Beiträge aus den IK (`ahv_beitraege`), welche nach Geschlecht aggregiert sind, mit der Variable `fraction` dividiert. Damit wird sichergestellt, dass die Summe von `ahv_beitraege` über die beiden Geschlechter in einem gegebenen Jahr genau `btr_vs_ag` im entsprechenden Jahr entspricht.

Als nächstes wird der in jedem vergangenen Jahr jeweils gültige AHV-Beitragssatz (`ahv_vs_ag` im Dataframe `BEITRAGSSAETZE`) angefügt:

```
        ungroup() |> # Gruppierung entfernen
        left_join(BEITRAGSSAETZE %>% select(jahr, ahv_vs_ag)) |>
        mutate(
            beitragsprozent=ahv_beitraege/(ahv_vs_ag*100)
        ) |>
        select(-sum_ahv_beitraege, -ahv_vs_ag, -btr_vs_ag, -fraction) |>
```

Der Beitragssatz wird verwendet, um die Beiträge je Beitragsprozent zu berechnen. Da wir die Entwicklung der Beiträge unabhängig von Veränderungen in den Beitragssätzen schätzen wollen, ist diese Normalisierung notwendig. Die Variable `beitragsprozent` enthält daher dann die AHV-Beiträge pro Beitragsprozent.

Als nächstes wird die jährliche Wachstumsrate der Inländer-Erwerbsbevölkerung nach Geschlecht (`delta_ept`), sowie die Wachstumsrate des nominalen Schweizerischen Lohnindexes (`delta_sli`) berechnet:

```
        group_by(sex) |> # Gruppieren nach sex
        mutate(
            delta_ept = (ept_inland / lag(ept_inland) - 1),
            delta_sli = ifelse(jahr==est_struktur_start,NA,lohn/100)
        ) |>
        mutate(

             ↪  strukturfaktor=beitragsprozent/(lag(beitragsprozent)*(1+delta_ept)*(1+delta_sli))-1
        ) |>
```

Mit Hilfe dieser zwei Wachstumsraten wird dann im letzten `mutate`-Befehl der Strukturfaktor pro Jahr und Geschlecht berechnet. Der Strukturfaktor entspricht dem Teil des Wachstums der Beiträge (bei konstantem Beitragssatz), welcher nicht durch das Wachstum der Inland-Erwerbsbevölkerung (`delta_ept`) oder dem Wachstum des SLI (`delta_sli`) erklärt ist. Entsprechend ergibt sich der Strukturfaktor eines Geschlechts im Jahr `t` aus den Beiträgen (`beitragsprozent`) im Jahr `t` dividiert mit dem Produkt der Beiträge im Vorjahr (`lag(beitragsprozent)`), dem Wachstum der Inland-Erwerbsbevölkerung im aktuellen Jahr (`(1+delta_ept)`), sowie dem Wachstum des SLI im aktuellen Jahr (`(1+delta_sli)`).

Zum Schluss wird der geometrische Mittelwert des Strukturfaktors nach Alter und Geschlecht gebildet, und der Strukturfaktor wird mit 100 multipliziert, um einen Ausdruck in Prozent zu erhalten:

```
    group_by(sex) |>  # Gruppieren nach sex
    summarize(
        strukturfaktor = (prod(1 + strukturfaktor, na.rm = TRUE))^(1 /
        ↪  sum(!is.na(strukturfaktor))) - 1
    ) %>%
    mutate(
        strukturfaktor=strukturfaktor*100
    )
```

Der nächste Abschnitt dient dazu, die Projizierten Beiträge der Versicherten und Arbeitnehmern zu berechnen:

```
# PROJEKTION LOHNBEITRAEGE
BEITRAGSSUMME_SEX  <- IK |>
    filter(
        vz==versicherung,
        jahr>=min(ABRECHNUNG$jahr)
    ) |>
    group_by(jahr, sex) |>
    summarize(
        beitraege=sum(beitrag_bsv, na.rm = TRUE),
        .groups = "drop"
    ) |>
    full_join(EPT_INLAND %>%
                  filter(
                      jahr>=min(IK$jahr),
                      jahr>=min(ABRECHNUNG$jahr)
                      )
    ) %>%
    left_join(ECKWERTE_EXTENDED) %>%
    left_join(ABRECHNUNG |> select(jahr, btr_vs_ag)) |>
    group_by(jahr) |> # Gruppieren nach Jahr
    mutate( # Korrektur IK-Daten um die Abweichung von den Beiträgen gemäss
    ↪  Abrechnung
        sum_beitraege = sum(beitraege, na.rm = TRUE), # Summe der Beiträge berechnen
        fraction = sum_beitraege / btr_vs_ag,    # Bruch berechnen
        beitraege = beitraege / fraction,
        ept_inland_total=sum(ept_inland)
    ) |>
    ungroup() |>
    left_join(ARBEITSLOSENQUOTE) %>%
    left_join(ECKWERTE_SCENARIO %>%  select(jahr, besta, arbeitslos)) %>%
    group_by(sex) %>%
    mutate(
        arbeitslosenquote=ifelse(is.na(arbeitslosenquote), arbeitslos,
        ↪  round(arbeitslosenquote,1)),
        delta_arbeitslosenquote=arbeitslosenquote-lag(arbeitslosenquote),
        besta_alq_factor = ifelse(jahr>max_jahr_ept_bestand &
        ↪  jahr<=PARAM_GLOBAL$jahr_abr+3 & !is.na(besta) & !is.na(arbeitslos),
        ↪  lag(ept_inland_total) / ept_inland_total * (1 + besta/100 +
        ↪  delta_arbeitslosenquote/100) - 1, 0),
        delta_ept = (ept_inland / lag(ept_inland) - 1),
        delta_sli = lohn/100
```

```
        ) |>
        left_join(STRUKTURFAKTOR) |>
        mutate(
            beitraege_pred =
                ifelse(jahr>=max(IK$jahr)+1,
                       beitraege[jahr == max(IK$jahr)] *
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_ept, 0))) * #
                           ↪  Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_sli, 0))) *
                           ↪  # Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,besta_alq_factor,
                           ↪  0))) *
                           (1+strukturfaktor/100)^(jahr-max(IK$jahr)),
                       NA)
            ) %>%
        ungroup() |>
        group_by(jahr) |>
        mutate(sum_beitraege_pred = sum(beitraege_pred, na.rm = TRUE)) %>%
        group_by(sex) |>
        mutate(
            beitraege_pred = ifelse(jahr>=PARAM_GLOBAL$jahr_abr,
                                    beitraege_pred * btr_vs_ag[jahr ==
↪   PARAM_GLOBAL$jahr_abr] / sum_beitraege_pred[jahr == PARAM_GLOBAL$jahr_abr],
                                    beitraege_pred * btr_vs_ag / sum_beitraege_pred
            )
        ) |>
        select(-btr_vs_ag) |>
        mutate(
            btr_vs_ag=ifelse(jahr>=max(IK$jahr)+1,beitraege_pred,beitraege)
        ) %>%
        select(jahr, sex, btr_vs_ag)
```

Er wird nachfolgend wiederum Schritt für Schritt erläutert.

Als erstes werden die IK-Daten der IV (da hier `versicherung==iv` ist) ab dem Jahr, für welches die Beiträge von Versicherten und Arbeitnehmern auch in der IV-Abrechnung (da `ABRECHNUNG==IV_ABRECHNUNG` ist) verfügbar sind, eingelesen:

```
    # PROJEKTION LOHNBEITRAEGE
    BEITRAGSSUMME_SEX  <- IK |>
        filter(
            vz==versicherung,
            jahr>=min(ABRECHNUNG$jahr)
        ) |>
        group_by(jahr, sex) |>
        summarize(
            beitraege=sum(beitrag_bsv, na.rm = TRUE),
            .groups = "drop"
        ) |>
```

Mit dem `summarize`-Befehl werden dann die Beiträge gemäss IK nach Jahr und Geschlecht berechnet.

Als nächstes werden die Daten zur Inland-Erwerbsbevölkerung (`EPT_INLAND`) angefügt:

```
        full_join(EPT_INLAND %>%
                      filter(
                          jahr>=min(IK$jahr),
                          jahr>=min(ABRECHNUNG$jahr)
                          )
        ) %>%
        left_join(ECKWERTE_EXTENDED) %>%
        left_join(ABRECHNUNG |> select(jahr, btr_vs_ag)) |>
```

Hierbei wird ein `full_join`-Befehl verwendet. Dadurch wird das Dataframe um alle Jahre, für welche (Szenario)-Daten zur Inland-Erwerbsbevölkerung vorliegen, erweitert (Stand 2025 somit bis 2070). Dadurch werden auch gleich die Datenzeilen für die Projektionen geschaffen. Zusätzlich werden dann auch noch die Daten zur Lohn- und Preisentwicklung (`ECKWERTE_EXTENDED`), sowie die Beiträge von Versicherten und Arbeitnehmern gemäss IV-Abrechnung (`btr_vs_ag` im Dataframe `ABRECHNUNG`) angefügt.

Als nächstes werden, analog zum Vorgehen bei der Abschätzung des Strukturfaktors oben, die Beiträge gemäss IK mit den Beiträgen gemäss IV-Abrechnung hochskaliert:

```
        group_by(jahr) |> # Gruppieren nach Jahr
        mutate( # Korrektur IK-Daten um die Abweichung von den Beiträgen gemäss
        ↪  Abrechnung
            sum_beitraege = sum(beitraege, na.rm = TRUE), # Summe der Beiträge berechnen
            fraction = sum_beitraege / btr_vs_ag,    # Bruch berechnen
            beitraege = beitraege / fraction,
            ept_inland_total=sum(ept_inland)
        ) |>
        ungroup() |>
```

Im nächsten Schritt werden die Daten für die Korrekturen in der kurzen Frist angefügt. Der Hintergrund dieser Anpassung ist der Folgende: Die Erwerbsbevölkerungsszenarien des BFS werden nur alle 5 Jahre aktualisiert. Dies bedeutet, dass diese Szenarien kurzfristige konjunkturelle Entwicklungen auf dem Arbeitsmarkt nicht abbilden. Diese konjunkturellen Entwicklungen haben aber einen grossen Effekt auf die Entwicklung der Einnahmen aus Beiträgen von Versicherten und Arbeitnehmern. Aus diesem Grund nutzen wir für die kurze Frist die Arbeitsmarkt-Prognosen des SECO, welche Teil der volkswirtschaftlichen Eckwerte der ESTV sind. Insbesondere nutzen wir die Prognose über die Entwicklung der Erwerbstätigen gemäss BESTA, sowie die Prognose über die Entwicklung der Arbeitslosenquote. Wir nehmen dann an, dass die Inland-Erwerbsbvölkerung in den kommenden 2-3 Jahren ab dem Abrechnungsjahr sich wie folgt entwickelt (der Zeitraum kann entweder 2 oder 3 Jahre sein, da das Datum der Verfügbarkeit der neuen Abrechnung sich vom Datum der Verfügbarkeit des neuen BESTA- und Arbeitslosenquote-Progonsehorizont unterscheidet.): Das Wachstum der Inland-Erwerbsbevölkerung entspricht dem Wachstum der Beschäftigung gemäss BESTA zuzüglich dem Wachstum der Arbeitslosenquote (in Prozentpunkten). Der Einbezug des Wachstums der Arbeitslosenquote ist dadurch motiviert, dass Arbeitslose, die Taggeld beziehen, auch beitragspflichtig sind, und daher für die Entwicklung der Beiträge relevant sind.[40]

Eine durch dieses Vorgehen implizierte potentiell kritische Annahme ist, dass sich der Arbeitsmarkt ab dem Jahr t+3 wieder im Gleichgewicht befindet, und sich die Erwerbsbevölkerung daher wieder mit der langfristigen Wachstumsrate gemäss BFS-Szenario entwickelt. Diese Annahme sollte bei kleinen konjunkturellen Schwankungen unkritisch sein, kann jedoch zum Problem werden, wenn grössere wirtschaftliche Kristen die

[40] Zum Beispiel rechnen wir im Fall, dass die Beschäftigung nach BESTA um 2% zurückgeht, während die Arbeitslosenquote um 2%-Punkte steigt, mit konstant bleibenden Einnahmen aus Lohnbeiträgen, da der Rückgang der Beiträge von Erwerbstätigen mit dem Anstieg der Beiträge von den Taggelder der Arbeitslosen kompensiert wird (dass die Ersatzquote der Arbeitslosen lediglich 70-80% beträgt wird hier der Einfachheit des Modells halber bewusst ignoriert).

Arbeitsmarktentwicklung beeinflussen. Aus diesem Grund wurde eine Warnung eingebaut, welche ausgegeben wird, falls die BESTA-Projektion Wachstumsrate enthält, die im Absolutwert grösser sind als 2% (siehe Erläuterungen ganz am Anfang der Erläuterungen zu diesem Modul).

Nachfolgend der Code, der die Überlegungen der letzten zwei Abschnitte implementiert:

```
        left_join(ARBEITSLOSENQUOTE) %>%
        left_join(ECKWERTE_SCENARIO %>%  select(jahr, besta, arbeitslos)) %>%
        group_by(sex) %>%
        mutate(
            arbeitslosenquote=ifelse(is.na(arbeitslosenquote), arbeitslos,
            ↪  round(arbeitslosenquote,1)),
            delta_arbeitslosenquote=arbeitslosenquote-lag(arbeitslosenquote),
            besta_alq_factor = ifelse(jahr>max_jahr_ept_bestand &
            ↪  jahr<=PARAM_GLOBAL$jahr_abr+3 & !is.na(besta) & !is.na(arbeitslos),
            ↪  lag(ept_inland_total) / ept_inland_total * (1 + besta/100 +
            ↪  delta_arbeitslosenquote/100) - 1, 0),
            delta_ept = (ept_inland / lag(ept_inland) - 1),
            delta_sli = lohn/100
        ) |>
```

Als erstes werden also die historischen Daten zur Arbeitslosenquote (`ARBEITSLOSENQUOTE`), sowie die projizierte Wachstumsrate der Erwerbsbevölkerung gemäss BESTA sowie die projizierte Arbeitslosenquote gemäss den Eckwerten der ESTV (`besta` und `arbeitslos` im Dataframe `ECKWERTE_SCENARIO`) angefügt. Danach wird die Veränderung der Arbeitslosenquote über die Jahre in Prozentpunkte gebildet (`delta_arbeitslosenquote`), wobei die historische Arbeitslosenquote (in der Variable `arbeitslosenquote`) mit der projizerten Arbeitslosenquote (in der Variable `arbeitslos`) verbunden wird.

Danach wird für die relevanten Jahre der Faktor berechnet, um welchen das Wachstum der Inland-Erwerbsbevölkerung gemäss BESTA und Arbeitslosenquote-Entwicklung vom Wachstum der Inland-Erwerbsbevölkerung gemäss BFS-Szenarien abweicht. Die relevanten Jahre werden im `if`-Teil des `ifelse`-Befehls ausgewählt. Sie umfassen die Jahre, welche nach dem letzten Jahr mit beobachteter Inland-Erwerbsbevölkerung (`jahr>max_jahr_ept_bestand`) aber nicht über dem dritten Jahr nach der aktuellen Abrechnung (`jahr<=PARAM_GLOBAL$jahr_abr+3`) liegen, und für welche sowohl eine BESTA-Projektion (`!is.na(besta)`) als auch ein Projektion für die Arbeitslosenquote (`!is.na(arbeitslos)`) existiert. Für diese Jahre wird der Faktor als die Abweichung der Wachstumsrate der Inland-Erwerbsbevölkerung gemäss BESTA und Arbeistlosenquote (`* (1 + besta/100 + delta_arbeitslosenquote/100)`) von der Wachstumsrate der Inland-Erwerbsbevölkerung gemäss BFS-Szenario (`lag(ept_inland_total) / ept_inland_total`) bestimmt. Für alle Jahre die die `if`-Bedingung nicht erfüllen ist der Faktor `besta_alq_factor` gleich 0. Am Ende des Codeblocks werden noch die Wachstumsraten der Erwerbsbevölkerung gemäss BFS-Szenarien (`delta_ept`) sowie die Wachstumsrate des nominalen SLI (`delta_sli`) berechnet.

Im nächsten Schritt wird der vorangehend geschätzte geschlechterspezifische Strukturfaktor angefügt, und die eigentliche Projektion der Beiträge wird vorgenommen:

```
        left_join(STRUKTURFAKTOR) |>
        mutate(
            beitraege_pred =
                ifelse(jahr>=max(IK$jahr)+1,
                       beitraege[jahr == max(IK$jahr)] *
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_ept, 0))) * #
                           ↪  Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_sli, 0))) *
                           ↪  # Ersetze NA-Werte durch 0
```

```
                        cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,besta_alq_factor,
                        ↪  0))) *
                        (1+strukturfaktor/100)^(jahr-max(IK$jahr)),
                    NA)
            ) %>%
```

Beachte, dass das Dataframe immer noch nach Geschlecht gruppiert ist, wodurch `beitraege_pred` nach Geschlecht berechnet wird. Die `if`-Bedingung in der Gleichung für `beitraege_pred` stellt sicher, dass die Projektion nur für Jahre berechnet wird, für welche keine tatsächlichen Beitragszahlungen in den IKs beobachtet sind. Für die Jahre nach dem letzten IK-Jahr wird dann die jeweilige Beitragsprojektion gebildet, indem die Beiträge im letzten IK-Jahr mit dem kumulierten Wachstum der Inland-Erwerbsbevölkerung seit dem letzten IK-Jahr (`cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_ept, 0)))`), dem kummulierten Wachstum des SLI seit dem letzten IK-Jahr (`cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_sli, 0)))`), dem kumulierten Wachstum auf Grund des Anpassungsfaktors für die kurze Frist (`cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,besta_alq_factor, 0)))`), und dem kumulierten Wachstum auf Grund des Strukturfaktors (`(1+strukturfaktor/100)^(jahr-max(IK$jahr))`) multipliziert wird.

Als nächstes wird eine letzte Korrektur vorgenommen, welche wie folgt begründet ist: Die aktuellsten IK-Daten stammen jeweils aus dem Jahr `t-2`, während die aktuellste IV-Abrechnung für das Jahr `t-1` verfügbar ist. Die Projektionen werden ab dem letzten verfügbaren IK-Jahr geschätzt (siehe oben). Dies bedeutet, dass für das Jahr `t-1` sowohl eine Projektion für die Beiträge als auch ein Wert aus der Abrechnung verfügbar ist. Es liegt daher nahe, die Projektion für das Jahr `t-1` an den Wert aus der Abrechnung in `t-1` anzugleichen. Da die Projektion für die Beiträge lediglich auf (kumulierten) Wachstumsraten der Komponenten, Inland-Erwerbsbevölkerungswachstum, SLI-Wachstum, und Strukturbedingtes Wachstum beruht, liegt ausserdem nahe, auch alle Projektion nach dem Jahr `t-1` 1:1 mit der Abweichung zwischen der Projektion und dem Wert der Abrechnung in `t-1` zu korrigieren. Diese Korrektur wird im nachfolgenden Codeblock vorgenommen:

```
        ungroup() |>
        group_by(jahr) |>
        mutate(sum_beitraege_pred = sum(beitraege_pred, na.rm = TRUE)) %>%
        group_by(sex) |>
        mutate(
            beitraege_pred = ifelse(jahr>=PARAM_GLOBAL$jahr_abr,
                                    beitraege_pred * btr_vs_ag[jahr ==
↪  PARAM_GLOBAL$jahr_abr] / sum_beitraege_pred[jahr == PARAM_GLOBAL$jahr_abr],
                                    beitraege_pred * btr_vs_ag / sum_beitraege_pred
            )
        ) |>
```

Konkret wird zuerst die Summe der projizierten Beiträge über die Geschlechter gebildet (`sum_beitrage_pred`). Danach wird für alle Jahre ab dem Jahr der aktuellsten Abrechnung (`jahr>=PARAM_GLOBAL$jahr_abr`) die geschlechterspezifische Projektion für die Beiträge (`beitraege_pred`) mit der Abweichung zwischen der Summe der projizierten Beiträge im Jahr der aktuellsten Abrechnung (`sum_beitraege_pred[jahr == PARAM_GLOBAL$jahr_abr]`) und der Beiträge gemäss IV-Abrechnung im Jahr der aktuellsten IV-Abrechnung (`btr_vs_ag[jahr == PARAM_GLOBAL$jahr_abr]`) skaliert. Für alle Jahre vor dem Jahr der aktuellsten Abrechnung werden die geschlechterspezifischen Beiträge mit der Abweichung der Summe der projizierten Beiträge von den Beiträgen der Abrechnung in diesem Jahr skaliert.[41]

Zum Schluss wird die Variable `btr_vs_ag` gebildet, welche (Analog zur Benennung in der IV-Abrechnng), die geschlechterspezifischen Beiträge der Versicherten und Arbeitgebern für die vergangenen und die zukünftigen Jahre enthält:

[41] Dieser Fall ist nur relevant für die Zeit zwischen Februar und April, wo die provisorische IV-Abrechnung für das Jahr 't-1' vorliegt, aber die IK-Daten lediglich bis ins Jahr 't-3' verfügbar sind (die IK-Daten für das Jahr 't-2' erscheinen jeweils erst Anfang April).

```
select(-btr_vs_ag) |>
    mutate(btr_vs_ag = ifelse(jahr >= max(IK$jahr) + 1, beitraege_pred,
        beitraege)) %>%
    select(jahr, sex, btr_vs_ag)

BEITRAGSSUMME <- BEITRAGSSUMME_SEX %>%
    group_by(jahr) %>%
    summarize(btr_vs_ag = sum(btr_vs_ag))
```

Hierzu wird zuerst die Variable `btr_vs_ag` entfernt, welche die aggregierten Beiträge aus der IV-Abrechnung enthält. Danach wird die Variable `btr_vs_ag` gebildet, wobei für Jahre nach dem letzten IK-Jahr die projizierten Beiträge aus `beitraege_pred` verwendet werden, und für Jahre bis zum letzten IK-Jahr die anhand der IK-Daten und der Daten aus der IV-Abrechnung berechneten Beiträge aus der Variable `beitraege`. Damit ist die Projektion der Beiträge abgeschlossen. Der letzte Codeblock dient lediglich dazu, für das Dataframe `BEITRAGSSUMME` die Beiträge über die Geschlechter zu aggregieren.

Im nächsten Codeblock wird die Projektion für die AHV-Lohnsumme gebildet:

```
    # PROJEKTION AHV-LOHNSUMME
    LOHNSUMME_SEX <- IK |>
        filter(
            vz=="ahv"
        ) |>
        group_by(jahr, sex) |>
        summarize(
            ahv_beitraege=sum(beitrag_bsv, na.rm = TRUE),
            ahv_lohnsumme=sum(einkommen_ik, na.rm = TRUE),
            .groups = "drop"
        ) |>
        full_join(EPT_INLAND %>%
                      filter(jahr>=min(IK$jahr))
        ) %>%
        left_join(ECKWERTE_EXTENDED) %>%
        left_join(AHV_ABRECHNUNG |> select(jahr, btr_vs_ag)) |>
        group_by(jahr) |> # Gruppieren nach Jahr
        mutate( # Korrektur IK-Daten um die Abweichung von den Beiträgen gemäss
        ↪  Abrechnung
            sum_ahv_beitraege = sum(ahv_beitraege, na.rm = TRUE), # Summe der Beiträge
            ↪  berechnen
            fraction = sum_ahv_beitraege / btr_vs_ag,    # Bruch berechnen
            ahv_beitraege = ahv_beitraege / fraction,
            ahv_lohnsumme = ahv_lohnsumme / fraction   # Beiträge durch den Bruch teilen
        ) |>
        ungroup() |>
        left_join(ARBEITSLOSENQUOTE) %>%
        left_join(ECKWERTE_SCENARIO %>%  select(jahr, besta, arbeitslos)) %>%
        group_by(sex) %>%
        mutate(
            arbeitslosenquote=ifelse(is.na(arbeitslosenquote), arbeitslos,
            ↪  round(arbeitslosenquote,1)),
            delta_arbeitslosenquote=arbeitslosenquote-lag(arbeitslosenquote),
            besta_alq_factor = ifelse(jahr<=max_jahr_ept_bestand | is.na(besta) |
            ↪  is.na(arbeitslos) | jahr>PARAM_GLOBAL$jahr_abr+3, 0, lag(ept_inland) /
            ↪  ept_inland * (1 + besta/100 + delta_arbeitslosenquote/100) - 1),
```

```
            delta_ept = (ept_inland / lag(ept_inland) - 1),
            delta_sli = lohn/100
        ) |>
        left_join(STRUKTURFAKTOR) |>
        mutate(
            ahv_beitraege_pred =
                ifelse(jahr>=max(IK$jahr)+1,
                       ahv_beitraege[jahr == max(IK$jahr)] *
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_ept, 0))) * #
                           ↪  Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_sli, 0))) *
                           ↪  # Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,besta_alq_factor,
                           ↪  0))) *
                           (1+strukturfaktor/100)^(jahr-max(IK$jahr)),
                       NA),
            ahv_lohnsumme_pred =
                ifelse(jahr>=max(IK$jahr)+1,
                       ahv_lohnsumme[jahr == max(IK$jahr)] *
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_ept, 0))) * #
                           ↪  Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,delta_sli, 0))) *
                           ↪  # Ersetze NA-Werte durch 0
                           cumprod((1 + ifelse(jahr>=max(IK$jahr)+1,besta_alq_factor,
                           ↪  0))) *
                           (1+strukturfaktor/100)^(jahr-max(IK$jahr)),
                       NA)
        ) %>%
        ungroup() |>
        group_by(jahr) |>
        mutate(sum_ahv_beitraege_pred = sum(ahv_beitraege_pred, na.rm = TRUE)) %>%
        group_by(sex) |>
        mutate(
            ahv_lohnsumme_pred = ifelse(jahr>=PARAM_GLOBAL$jahr_abr,
                                        ahv_lohnsumme_pred * btr_vs_ag[jahr ==
↪   PARAM_GLOBAL$jahr_abr] / sum_ahv_beitraege_pred[jahr == PARAM_GLOBAL$jahr_abr],
                                        ahv_lohnsumme_pred * btr_vs_ag /
↪   sum_ahv_beitraege_pred
                )
        ) |>
        mutate(
            ahv_lohnsumme=ifelse(jahr>=max(IK$jahr)+1,ahv_lohnsumme_pred,ahv_lohnsumme)
        ) %>%
        select(jahr, sex, ahv_lohnsumme)


    LOHNSUMME <- LOHNSUMME_SEX %>%
        group_by(jahr) %>%
        summarize(ahv_lohnsumme=sum(ahv_lohnsumme))
```

Das Vorgehen hier ist analog zum Vorgehen bei der Projektion für die Beiträge von Versicherten und Arbeitnehmern, mit dem Unterschied, dass die Projektion der AHV-Lohnsumme ausschliesslich auf den IK Daten zur AHV sowie den AHV-Abrechnungsdaten basiert. Das heisst, AHV-Lohnsumme wird so berech-

net, dass sie in der Vergangenheit der Entwicklung der Einnahmen aus Beiträgen von Versicherten und Arbeitgebern aus der AHV-Abrechnung folgt. Für die Zukunft entsprechen die Wachstumsraten der AHV-Lohnsumme nach Geschlecht exakt den Wachstumsraten der Beiträge nach Geschlecht in der IV. Für die über die Geschlechter aggregierte Wachstumsrate der AHV-Lohnsumme können nichtsdestotrotz ganz leichte Abweichungen zur Wachstumsrate der Beiträge in der IV auftreten, da sich das Gewicht der Geschlechter auf Grund der Nutztung anderer IK-Daten (IK-AHV vs. IK-IV) marginal unterscheidet. Solche Unterschiede werden aber erst einige Stellen nach dem Komma ersichtlich sein, und sind gewollt und kein Fehler.

Zum Schluss werden die berechneten Dataframes an das Modul `mod_iv_beitrag.R` (oder das Modul `mod_iv_fortschreibung.R`) zurückgegeben:

```
return(list(BEITRAGSSUMME = BEITRAGSSUMME, LOHNSUMME = LOHNSUMME,
    BEITRAGSSUMME_SEX = BEITRAGSSUMME_SEX, LOHNSUMME_SEX = LOHNSUMME_SEX,
    STRUKTURFAKTOR = STRUKTURFAKTOR))
```

4.32.2 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}"))

#------ 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}"))) %>%
    select(-btr_vs_ag)
```

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 der Strukturfaktor an das Finanzperspektivenmodell (respektive das Modul `mod_iv_einnahmen.R`) zurückgegeben werden. Dies geschieht mit der folgenden abschliessenden Codezeile des Moduls:

```
return(list(IV_BEITRAGSSUMME = IV_BEITRAGSSUMME, STRUKTURFAKTOR =
↪  tl_iv_beitrag$STRUKTURFAKTOR))
```

4.33 Module mod_iv_uebrigeeinnahmen.R und mod_iv_regress.R

Dokumentation zuletzt aktualisiert am 14.01.2025

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) %>%
    mutate(leist = rentensumme_iv + he) %>%
    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 ----------------------------------------------------------#
return(IV_REGRESS = 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.

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 ----------------------------------------------------------#
return(UEBRIGEEINNAHMEN_IV = UEBRIGEEINNAHMEN_IV)
```

4.34 Modul mod_iv_einausgaben.R

Dokumentation zuletzt aktualisiert am 18.11.2024

`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, DELTAS_MASSNAHMEN =
    ↪  tl_iv_mass$DELTAS_MASSNAHMEN)
```

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 4.7) 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 4.11) und OHNE die Einnahmen auf Grund von Politikmassnahmen.
- `DELTAS_MASSNAHMEN`: Die Einnahme- und Ausgabeauswirkungen der verschiedenen Politikmassnahmen, die berechnet wurden (vgl. Kapitel 4.12). `DELTA_MASSNAHMEN` enthält Daten im langen Format in den Spalten `jahr`, `konto` (das betreffende Konto der IV-Abrechnung), `wert` (die finanzielle Auswirkung der Politikmassnahme im entsprechenden Jahr).

Als Erstes werden die Auswirkungen der Politikmassnahmen, welche im Modul `wrap_iv_massnahmen` (vgl. Kapitel 4.12) projiziert wurden, zu den in `wrap_iv_hauptberechnungen` projizierten Einnahme- und Ausgabevektoren addierte. Hierzu wird als Erstes überprüft, ob alle in `DELTAS_MASSNAHMEN` definierten Konti (also die IV-Abrechnungspositionen, welche von der jeweiligen Politikmassnahme betroffen sind) entweder im Dataframe `EINNAHMEN_IV` oder `AUSGABEN_IV` enthalten sind:

```
# Überprüfen, ob alle Werte von 'konto' in
# DELTAS_MASSNAHMEN in den Spaltennamen von EINNAHMEN_IV
# oder AUSGABEN_IV enthalten sind
konti_in_einnahmen <- DELTAS_MASSNAHMEN$konto %in% colnames(EINNAHMEN_IV)
konti_in_ausgaben <- DELTAS_MASSNAHMEN$konto %in% colnames(AUSGABEN_IV)

# Überprüfen, ob alle Konti gültig sind
if (!all(konti_in_einnahmen | konti_in_ausgaben)) {
    # Stoppe die Ausführung mit einer Fehlermeldung, wenn
    # ungültige Konti gefunden werden
    stop("Massnahmen betreffen konti, die nicht in der IV-Abrechnung enthalten sind.
    ↪  Bitte diese Konti in mod_iv_einausgaben zu EINNAHMEN_IV oder AUSGABEN_IV
    ↪  hinzufügen.")
}
```

Als nächstes wird jede Zeile in `DELTA_MASSNAHMEN` zu der entsprechenden Einnahme- oder Ausgabepositon addiert:

```
# Werte aus DELTAS_MASSNAHMEN zu den entsprechenden Spalten
# in EINNAHMEN_IV oder AUSGABEN_IV hinzufügen
aggregated_DELTAS_MASSNAHMEN <- DELTAS_MASSNAHMEN %>%
    group_by(konto, jahr) %>%
    summarise(wert_sum = sum(wert), .groups = "drop") %>%
    mutate(target = ifelse(konto %in% colnames(EINNAHMEN_IV),
        "EINNAHMEN_IV", "AUSGABEN_IV"))

# Iteriere über die Zeilen von aggregated_DELTAS_MASSNAHMEN
for (row in seq_len(nrow(aggregated_DELTAS_MASSNAHMEN))) {
    # Extrahiere Werte aus der aktuellen Zeile
    target <- aggregated_DELTAS_MASSNAHMEN[row, "target"][[1]]  # Ziel Dataframe
    konto <- aggregated_DELTAS_MASSNAHMEN[row, "konto"][[1]]  # Konto (Spaltenname)
    jahr <- aggregated_DELTAS_MASSNAHMEN[row, "jahr"][[1]]  # Jahr (Zeile)
    wert_sum <- aggregated_DELTAS_MASSNAHMEN[row, "wert_sum"][[1]]  # Wertsumme
```

```
    if (target == "EINNAHMEN_IV") {
        # Addiere Wert zur entsprechenden Spalte in
        # EINNAHMEN_IV
        EINNAHMEN_IV[EINNAHMEN_IV$jahr == jahr, konto] <- EINNAHMEN_IV[EINNAHMEN_IV$jahr
↪   ==
            jahr, konto] + wert_sum
    } else {
        # Addiere Wert zur entsprechenden Spalte in
        # AUSGABEN_IV
        AUSGABEN_IV[AUSGABEN_IV$jahr == jahr, konto] <- AUSGABEN_IV[AUSGABEN_IV$jahr ==
            jahr, konto] + wert_sum
    }
}
```

Zum Abschluss der Berechnungen der Einnahme- und Ausgabevektoren inklusive der Effekte der Massnahmen (d.h. die Vektoren `EINNAHMEN` respektive `AUSGABEN`) werden (zwischen-)summen für die Ausgaben und Einnahmen gebildet:

```
# Summen pro Kategorie für Ausgaben und Einnahmen berechnen, total berechnen
    EINNAHMEN <- EINNAHMEN_IV %>%
        mutate(
            btr_oeffh  = btr_bund + btr_mwst + btr_zinsiv,
            btr_vs_ag = 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,
            ein_tot_iv = btr_vs_ag + btr_bund + btr_mwst + btr_zinsiv + regr_ein_tot +
            ↪  ein_uebr_tot
        ) %>%
        relocate(
            btr_oeffh, .after = ein_uebr_tot # Verschiebe "btr_oeffh" nach "ein_uebr_tot"
        ) %>%
        relocate(
            btr_vs_ag, .after = btr_zinsiv # Verschiebe "btr_vs_ag" nach "btr_zinsiv"
        ) %>%
        relocate(
            ein_tot_iv, .after = btr_oeffh # Verschiebe "ein_tot_iv" nach "btr_oeffh"
        )

    AUSGABEN <- AUSGABEN_IV %>%
        mutate(
            rentensumme_iv = rent_ord + rent_ord_nachz + rent_aord + rent_aord_nachz +
                ausl_rueck + ausl_fs_leist + rueck_erstattungsf +
↪   abschr_rueck_erstattungsf, # Berechnung für "rentensumme_iv"
            geld_leist_iv = rentensumme_iv + he + tg_geld + btr_ant_iv, # Berechnung für
            ↪  "geld_leist_iv"
            ind_mass_tot_iv = mm + fi + im + ber_begl + mba +
                aus_uebr + hm + reise_kost + btr_ass + rueck_erstattungsf_ind, #
↪   Berechnung für "ind_mass_tot_iv"
            aus_tot_iv = geld_leist_iv + ind_mass_tot_iv +
                btr_inst_org_iv + df_kost_iv + verw_aufw_iv + a_aufw # Berechnung für
↪   "aus_tot_iv"
        ) %>%
```

```
        relocate(
            rentensumme_iv, .before = he # Verschiebe "rentensumme_iv" vor "he"
        ) %>%
        relocate(
            geld_leist_iv, .before = mm # Verschiebe "geld_leist_iv" vor "mm"
        ) %>%
        relocate(
            ind_mass_tot_iv, .before = btr_org # Verschiebe "ind_mass_tot_iv" vor
↪  "btr_org"
        ) %>%
        relocate(
            aus_tot_iv, .after = last_col() # Verschiebe "aus_tot_iv" ans Ende
        )
```

Als nächstes werden im Modul die Summen der Ausgaben und Einnahmen pro Massnahme gebildet:

```
## AUSGABEN- UND EINNAHMEEFFEKTE DER MASSNAHMEN PRO
## MASSNAHME AGGREGIEREN Erstelle leeres Dataframe
DELTAS_EINN_AUSG <- data.frame(jahr = unique(DELTAS_MASSNAHMEN$jahr))  # Initialisiere
↪  mit Jahr-Spalte

# Bearbeitung für Einnahmen
for (massnahme in unique(DELTAS_MASSNAHMEN$massnahme)) {
    # Filtere Konti, die Spalten in EINNAHMEN_IV sind
    valid_konti <- DELTAS_MASSNAHMEN %>%
        filter(massnahme == !!massnahme & konto %in% colnames(EINNAHMEN_IV))

    # Prüfe, ob es gültige Konti gibt
    if (nrow(valid_konti) > 0) {
        # Summiere über gültige Konti
        summarized <- valid_konti %>%
            group_by(jahr) %>%
            summarise(sum_einnahmen = sum(wert, na.rm = TRUE),
                .groups = "drop")

        # Füge die Spalte zu DELTAS_EINN_AUSG hinzu
        DELTAS_EINN_AUSG <- DELTAS_EINN_AUSG %>%
            left_join(summarized, by = "jahr") %>%
            rename_with(~paste0("einnahmen..", massnahme), "sum_einnahmen")
    }
}
```

Hierzu wird zuerst ein leeres Dataframe `DELTAS_EINN_AUSG` gebildet, in welchem danach die Resultate abgelegt werden. Danach wird für jede Massnahme in `DELTAS_MASSNAHME` überprüft, ob diese eines oder mehrere Einnahmen-Konti betrifft, und wenn dies der Fall ist wird die jährliche Summe der Einnahmeeffekte dieser Massnahme gebildet, und als Spalte mit dem Prefix `einnahmen..` gefolgt von dem Massnahmen-Name an das Dataframe `DELTAS_EINN_AUSG` gejoined.

Bezüglich der Ausgabeeffekte der Massnahme wird analog vorgegangen:

```
# Bearbeitung für Ausgaben
for (massnahme in unique(DELTAS_MASSNAHMEN$massnahme)) {
    # Filtere Konti, die Spalten in AUSGABEN_IV sind
```

```
    valid_konti <- DELTAS_MASSNAHMEN %>%
        filter(massnahme == !!massnahme & konto %in% colnames(AUSGABEN_IV))

    # Prüfe, ob es gültige Konti gibt
    if (nrow(valid_konti) > 0) {
        # Summiere über gültige Konti
        summarized <- valid_konti %>%
            group_by(jahr) %>%
            summarise(sum_ausgaben = sum(wert, na.rm = TRUE),
                .groups = "drop")

        # Füge die Spalte zu DELTAS_EINN_AUSG hinzu
        DELTAS_EINN_AUSG <- DELTAS_EINN_AUSG %>%
            left_join(summarized, by = "jahr") %>%
            rename_with(~paste0("ausgaben..", massnahme), "sum_ausgaben") %>%
            arrange(jahr)
    }
}

# Ersetze alle NA-Werte in DELTAS_EINN_AUSG durch 0
DELTAS_EINN_AUSG[is.na(DELTAS_EINN_AUSG)] <- 0
```

Somit sind die Berechnungen dieses Moduls abgeschlossen, und die Resultate werden an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_ergebnisse.R`) zurückgegeben:

```
#------ Output --------------------------------------------------------#
return(list(AUSGABEN = AUSGABEN, EINNAHMEN = EINNAHMEN, DELTAS_EINN_AUSG =
↪  DELTAS_EINN_AUSG))
```

4.35 Modul mod_bilanz_iv_rekursiv.R

Dokumentation zuletzt aktualisiert am 18.02.2025

`mod_bilanz_iv_rekursiv.R` wird im Skript `wrap_iv_ergebnisse.R` wie folgt aufgerufen:

```
BILANZ_IV <- mod_bilanz_iv_rekursiv(PARAM_GLOBAL, IV_ABRECHNUNG,
    AUSGABEN_IV, EINNAHMEN_IV, ZINS, FORTSCHREIBUNG_IV, DISKONTFAKTOR,
    DELTAS_MASSNAHMEN, tl_inp)
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `IV_ABRECHNUNG`: Die vergangenen Einnahmen und Ausgaben gemäss IV-Abrechnung.
- `tl_ein_aus$AUSGABEN`: Sämtliche Ausgabeprojektionen, mit Ausnahme des Zinsaufwandes (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.7), inklusive der Ausgaben auf Grund von Politikmassnahmen.
- `tl_ein_aus$EINNAHMEN`: Sämtliche Einnahmeprojektionen, mit Ausnahme des Zinsertrages (dieser kann erst nach Berechnung der Fondsstandprojektion ermittelt werden, vgl. Kapitel 4.11), inklusive der Einnahmen auf Grund von Politikmassnahmen.
- `ZINS`: Die für die Schulden und das Anlagekapital angenommenen Zinssätze.
- `FORTSCHREIBUNG_IV`: Dataframe mit den Fortschreibungsvariablen, insbesondere der berechnete Bundesbeitrag (Variable `bundesbeitrag`).

- `DELTAS_MASSNAHMEN`: Die Einnahme- und Ausgabeauswirkungen der verschiedenen Politikmassnahmen, die berechnet wurden (vgl. Kapitel 4.12). Das Dataframe wird nur hinzugefügt, um die Auswirkung der Rechnungsabgrenzung nach der neuen Rechnungslegung (Rechnungslegung 2025 respektive IPSAS-Rechungslegung) einzubeziehen und ist daher nur relevant, falls die neue Rechnungslegung ausgewählt ist (d.h., die Massnahme für die neue Rechnungslegung ausgewählt ist).
- `tl_inp`: Falls die IPSAS-Rechnungslegung ausgewählt ist (d.h., die Massnahme für die neue Rechnungslegung ausgewählt ist), werden Parameterwerte aus `tl_inp` benötigt.

In einem nächsten Schritt wird geprüft, ob die IPSAS-Rechnungslegung ausgewählt ist, und wenn ja, werden die Werte der Rechnungsabgrenzung in das Dataframe `RECHNUNGSABGRENZUNG` gelegt:

```
### IV-Bilanz und Erfolgsrechnung berechnen Prüfe, ob
### Rechnungslegung 2025 ausgewählt ist
if (PARAM_GLOBAL$flag_param_massn == TRUE && grepl("iv_rechnungslegung_25",
    tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int)) {

    # Identifiziere Spalten, die in AUSGABEN_IV und
    # EINNAHMEN_IV enthalten sind
    columns_in_ausgaben <- intersect(names(tl_inp$PARAM_IV_RECHNUNGSLEGUNG_25),
        names(AUSGABEN_IV))
    columns_in_einnahmen <- intersect(names(tl_inp$PARAM_IV_RECHNUNGSLEGUNG_25),
        names(EINNAHMEN_IV))

    # Summiere die Werte nach jahr
    RECHNUNGSABGRENZUNG <- DELTAS_MASSNAHMEN |>
        filter(massnahme == "iv_rechnungslegung_25") |>
        group_by(jahr) |>
        summarise(annual_rechnungsabgrenzung_passiv = sum(wert[konto %in%
            columns_in_ausgaben], na.rm = TRUE), annual_rechnungsabgrenzung_aktiv =
↪   sum(wert[konto %in%
            columns_in_einnahmen], na.rm = TRUE), .groups = "drop") |>
        complete(jahr = seq(min(AUSGABEN_IV$jahr), max(AUSGABEN_IV$jahr)),
            fill = list(annual_rechnungsabgrenzung_passiv = 0,
                annual_rechnungsabgrenzung_aktiv = 0)) |>
        mutate(rechnungsabgrenzung_passiv_netto = ifelse(jahr >=
            2023, cumsum(annual_rechnungsabgrenzung_passiv) -
            cumsum(annual_rechnungsabgrenzung_aktiv) +
            ↪  rowSums(select(tl_inp$PARAM_IV_RECHNUNGSLEGUNG_25,
            all_of(columns_in_ausgaben)) * 1e+06, na.rm = TRUE) -
            rowSums(select(tl_inp$PARAM_IV_RECHNUNGSLEGUNG_25,
                all_of(columns_in_einnahmen)) * 1e+06, na.rm = TRUE),
            0))

} else {
    # Erstelle RECHNUNGSABGRENZUNG mit
    # rechnungsabgrenzung_passiv == 0
    RECHNUNGSABGRENZUNG <- tibble(jahr = seq(min(AUSGABEN_IV$jahr),
        max(AUSGABEN_IV$jahr)), rechnungsabgrenzung_passiv_netto = 0,
        annual_rechnungsabgrenzung_aktiv = 0, annual_rechnungsabgrenzung_passiv = 0)
}
```

In einem nächsten Schritt werden die Einnahmen- und Ausgabenprojektionen mit der IV-Abrechnung zusammengefügt:

```
# Bilanz aus Einnahme- und Ausgabevektor und Abrechnung
# erstellen
BILANZ_IV <- AUSGABEN_IV %>%
    filter(jahr >= PARAM_GLOBAL$jahr_beginn) %>%
    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") %>%
    left_join(ZINS, by = "jahr") %>%
    left_join(RECHNUNGSABGRENZUNG) %>%
    mutate(erg_umlag_iv = ein_tot_iv - aus_tot_iv, rueckzahlung = ifelse(jahr >=
        2011, kap - lag(kap), 0), fonds = fonds - rechnungsabgrenzung_passiv_netto) %>%
    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. Zudem wird der Wert der Rechnungsabgrenzung angefügt und vom Fonds abgezogen (es handelt sich um Rückstellungen auf der Passivseite der Bilanz, die das Fondsvermögen verringern).

Danach wird der Anteil der nicht-flüssigen Anlagen in der Vergangenheit bestimmt:

```
    # Anteil des Fonds, der nicht flüssige Mittel oder Anlagen ist, berechnen (für
    ↪  Schätzung von Zinsertrag unten)
    BILANZ_IV <- BILANZ_IV |>
        mutate(ant_ausgaben_nichtanlagen =
        ↪  (lag(fonds)+lag(rechnungsabgrenzung_passiv_netto) - lag(fl_mtl))/aus_tot_iv)
        ↪  |> # Annahme: Vor Jahresende wird dem Fonds der Betrag entnommen, welcher für
        ↪  die Deckung der laufenden Ausgaben des kommenen Jahres benötigt wird.
        mutate(
            ant_ausgaben_nichtanlagen = if_else(
                jahr > PARAM_GLOBAL$jahr_abr,
                mean(
                    ant_ausgaben_nichtanlagen[
                        between(jahr, PARAM_GLOBAL$jahr_abr - 9, PARAM_GLOBAL$jahr_abr) #
                        ↪  Anteil für zukünftige Jahre als Durchschnittlicher Anteil der
                        ↪  letzten 10 Jahre
                    ],
                    na.rm = TRUE
                ),
                ant_ausgaben_nichtanlagen
            )
        )
```

Um die Konsistenz mit vergangenen Abrechnungen zu bewahren (in welchen keine Rechnungsabgrenzung gemacht wurde), wird die passive Reechnungsabgrenzung für die Berechnung des Anteils zum Fondsvermögen hinzugefügt, bevor die flüssigen Mittel und Anlagen vom Fondsvermögen abgezogen werden. Dies

hat auch eine logische Begründgung: Rechnungsabgrenzung ist rein buchhalterisch, und hat keinen direkten Einfluss auf die Geldflüsse, welche zur Bestimmung des Anteils des Fonds, welcher nicht angelegt werden kann (respektive in flüssigen Mitteln gehalten werden kann). Daher wird die Rechnungsabgrenzung bei der Abschätzung des Anteils der Ausgaben, die nicht angelegt sind, abgezogen. Mit letzten Mutatebefehl des Codeblocks wird der mittlere Anteil der Ausgaben, die nicht-Anlagen sind, der verganenen 10 Jahre für die Jahre der Zukunft eingefügt (was weiter unten zur Berechnung der flüssigen Mittel und Anlagen genutzt wird).

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:

```
## Rekursive Berechnung des Fondsstands, des Zinsertrags, sowie der (allfälligen)
↪  Schuldenrückzahlungen an die AHV
for(l_j in (PARAM_GLOBAL$jahr_abr+1):PARAM_GLOBAL$jahr_ende) {

    # Zinsaufwand für Schuld bei der AHV berechnen
    BILANZ_IV <- BILANZ_IV |>
        mutate(
            # Berechne Zinsaufwand für die Schuld bei der AHV
            schzins = if_else(
                jahr == l_j,
                -lag(kap) * nominal_zins_ivschuld,
                schzins
            )
        )
```

Als Erstes wird 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.

Als nächstes wird die Höhe des Bundesbeitrages berechnet:

```
    ## Bundesbeitrag berechnen
    # Berechnung des Bundesbeitrags mit den angegebenen Bedingungen
    berechneter_bund <- FORTSCHREIBUNG_IV |>
                            filter(jahr == l_j) |>
                            pull(berechnet_btr_bund)

    # Untergrenze 37.7% der Ausgaben (inkl. Zinsausgaben)
    untergrenze_bund <- 0.377 * (BILANZ_IV |>
                                    filter(jahr == l_j) |>
                                    pull(aus_tot_iv) +
                                    BILANZ_IV |>
                                    filter(jahr == l_j) |>
                                    pull(schzins))

    # Obergrenze 50% der Ausgaben (inkl. Zinsausgaben)
    obergrenze_bund <- 0.5 * (BILANZ_IV |>
                                filter(jahr == l_j) |>
                                pull(aus_tot_iv) +
                                BILANZ_IV |>
                                filter(jahr == l_j) |>
                                pull(schzins))
```

```
        # Überprüfe, ob berechneter_bund innerhalb der Grenzen liegt
        BILANZ_IV <- BILANZ_IV |>
            mutate(
                btr_bund = if_else(
                    jahr == l_j,
                    case_when(
                        berechneter_bund < untergrenze_bund ~ untergrenze_bund, # Setze
↪  Bundesbeitrag auf Untergrenze
                        berechneter_bund > obergrenze_bund ~ obergrenze_bund,   # Setze
↪  Bundesbeitrag auf Obergrenze
                        TRUE ~ berechneter_bund                                  #
                        ↪  Behalte berechneten Bundesbeitrag
                    ),
                    btr_bund
                )
            )
```

Hierzu wird zuerst die Variable `berechnet_btr_bund` aus `FORTSCHREIBUNG_IV` ausgelesen (vgl. Kapitel 4.44). Danach wird überprüft, ob `berechnet_btr_bund` innerhalb der gesetztlichen Grenze von 37.7% und 50% der Ausgaben (inklusive Schuldzinsszahlungen) liegt, und falls dies der Fall ist wird `berechnet_btr_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 `berechnet_btr_bund` unter- respektive überschritten wird.

In einem nächsten Schritt werden die totalen Einnahmen und das Umlageergebnis (nach IPSAS-Rechnungslegung, wobei weiter unten korrigiert wird, falls nicht die IPSAS-Rechnungslegung ausgewählt ist) berechnet:

```
        ## Gesamteinnahmen und Umlageergebnis unter Einbezug des Bundesbeitrages neu
        ↪  berechnen
        BILANZ_IV <- BILANZ_IV |>
            mutate(
                # Gesamteinnahmen anpassen
                ein_tot_iv = if_else(jahr == l_j, ein_tot_iv + btr_bund, ein_tot_iv),
                # Umlageergebnis anpassen
                erg_umlag_iv = if_else(jahr == l_j, ein_tot_iv - aus_tot_iv,
                ↪  erg_umlag_iv)
            )
```

Als nächstes wird das Anlagergebnis berechnet, also die in einem Projektionsjahr erwartete Anlagerendite, sowie das resultierende Betriebsergebnis:

```
        ## Zinsertrag und Betriebsergebnis berechnen
        BILANZ_IV <- BILANZ_IV |>
            mutate(
                # Zinsertrag berechnen (basierend auf Umlageergebnis ohne
                ↪  Rechnungsabgrenzung)
                anl_erg = if_else(
                    jahr == l_j,
                    (lag(fl_mtl) + (erg_umlag_iv + annual_rechnungsabgrenzung_passiv -
↪  annual_rechnungsabgrenzung_aktiv) / 2) * nominal_zins_fonds,
                    anl_erg
```

```
            ),
            # Betriebsergebnis berechnen
            erg_betr = if_else(
                jahr == l_j,
                erg_umlag_iv + anl_erg - schzins,
                erg_betr
            )
        )
```

Wir nehmen an, dass die flüssigen Mittel und Anlagen sowie die hälfte des Umlageergebnis (ohne Berücksichtigung der marginalen Rechnungsabgrenzung im jeweiligen Jahr) verzinst werden. Damit wird dann das Betriebsergebnis ermittelt.

In einem letzten Schritt wird nun der Stand des IV-Fonds, der Schulden der IV bei der AHV, sowie der Stand der flüssigen Mittel und Anlagen, ermittelt:

```
    ## Fondsstand und Schuldenrückzahlung an AHV
    # Schuldenrückzahlung und Schuldenstand
    BILANZ_IV <- BILANZ_IV |>
        arrange(jahr) |> # Sortiere nach Jahr, falls nicht sortiert
        mutate(
            rueckzahlung = if_else(
                jahr == l_j,
                case_when(
                    # Fall 1: Keine Rückzahlung
                    lag(fonds) + erg_betr + rechnungsabgrenzung_passiv_netto -
                    ↪  ant_ausgaben_nichtanlagen * (ifelse(jahr !=
                    ↪  PARAM_GLOBAL$jahr_ende, lead(aus_tot_iv) -
                    ↪  lead(annual_rechnungsabgrenzung_passiv),aus_tot_iv -
                    ↪  annual_rechnungsabgrenzung_passiv)) - # = flüssige Mittel und
                    ↪  Anlagen ohne Rückzahlung
                        0.5 * (aus_tot_iv + schzins) < 0 | lag(kap) == 0 ~ 0,
                    # Fall 2: Schuldenrückzahlung
                    TRUE ~ pmin(
                        -lag(kap),
                        lag(fonds) + erg_betr + rechnungsabgrenzung_passiv_netto -
                        ↪  ant_ausgaben_nichtanlagen * (ifelse(jahr !=
                        ↪  PARAM_GLOBAL$jahr_ende, lead(aus_tot_iv) -
                        ↪  lead(annual_rechnungsabgrenzung_passiv),aus_tot_iv -
                        ↪  annual_rechnungsabgrenzung_passiv)) - # = flüssige Mittel
                        ↪  und Anlagen ohne Rückzahlung
                            0.5 * (aus_tot_iv + schzins)
                    )
                ),
                rueckzahlung
            ),
            # Schuldenstand (kap) nach Rückzahlung aktualisieren
            kap = if_else(
                jahr == l_j,
                lag(kap) + rueckzahlung,
                kap
            )
        )
```

```
        # Fondsstand und Stand flüssige Mittel
        BILANZ_IV <- BILANZ_IV |>
            mutate(
                # Fondsstand nach Rückzahlung berechnen
                fonds = if_else(
                    jahr == l_j,
                    lag(fonds) + erg_betr - rueckzahlung,
                    fonds
                ),
                # Flüssige Mittel berechnen
                fl_mtl = case_when(
                    jahr == l_j & jahr != PARAM_GLOBAL$jahr_ende ~ fonds +
↪   rechnungsabgrenzung_passiv_netto - ant_ausgaben_nichtanlagen * (lead(aus_tot_iv) -
↪   lead(annual_rechnungsabgrenzung_passiv)),
                    jahr == l_j & jahr == PARAM_GLOBAL$jahr_ende ~ fonds +
↪   rechnungsabgrenzung_passiv_netto - ant_ausgaben_nichtanlagen * (aus_tot_iv -
↪   annual_rechnungsabgrenzung_passiv), # diesjährige Ausgaben verwenden im letzten
↪   Projektionsjahr
                    TRUE ~ fl_mtl
                )
            )
```

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 nichtflüssigen Anlagen der IV 50% der Jahresausgaben überschreiten. 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.

Für die Berechnung der flüssigen Mittel und Anlagen wird angenommen, dass die Compenswiss einen Teil der erwarteten Ausgaben vom nächsten Jahr (abzüglich der passiven Rechnungsabgrenzungen im nächsten Jahr, die aktiven Rechnungsabgrenzungen im nächsten Jahr werden hier nicht einbezogen, da sich diese auf die Einnahme-, aber nicht auf die Ausgabeseite, auswirken) für die Deckung der laufenden Ausgaben zurücklegt und nicht investiert. Dies ist das vorgehen von Compenswiss, gemäss dem Brief der Compenswiss an den Bundesrat vom Dezember 2024.

Zum Schluss wird noch die Ausgabespalte für den Aufwand ohne Schuldzins erstellt, welche im Falle der neuen Rechnungslegung gleich ist wie die totalen Ausgaben (aber nicht im Falle der alten Rechnungslegung). Danach werden, für den Fall dass die alte Rechnungslegung ausgewählt ist, die Schuldzinszahlungen zu den Totalausgaben hinzugezählt, respektive vom Umlageergebnis subtrahiert:

```
# aus_tot_ivoz erstellen (Für Ausgabe in Excel-Tabelle
# später, unabhängig von ausgewählter Rechnungslegung ohne
# Zinsaufwand)
BILANZ_IV <- BILANZ_IV %>%
    mutate(aus_tot_ivoz = aus_tot_iv)

# Zins ins Umlageergebnis aufnehmen, falls alte
# Rechnungslegung ausgewählt ist
if (!(PARAM_GLOBAL$flag_param_massn == TRUE && grepl("iv_rechnungslegung_25",
    tl_inp$PARAM_MASSNAHMEN_BASE$aktivierte_massnahmen_int))) {
    BILANZ_IV <- BILANZ_IV %>%
        mutate(aus_tot_iv = aus_tot_iv + schzins, erg_umlag_iv = erg_umlag_iv -
```

```
            schzins)
}
```

Zum Schluss wird die resultierende IV-Bilanz an das Finanzperspektivenmodell (respektive das Modul `wrap_iv_ergebnisse.R`) zurückgegeben:

```
#------ Output --------------------------------------------------------#
return(BILANZ_IV = BILANZ_IV)
```

4.36 Modul mod_abrechnung.R

Dokumentation zuletzt aktualisiert am 18.02.2025

`mod_abrechnung.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
tl_abrechnung <- mod_abrechnung(tl_inp, versicherung = c("ahv",
    "iv"))
```

Die Funktion verlangt die folgenden Argumente:

```
mod_abrechnung <- function(tl_inp,
                           versicherung
) {
```

- `tl_inp`: Sämtliche Input-Daten, aus welchen dann die Abrechnungsdaten der in `versicherung` ausgewählten Versicherungen ausgewählt werden.

Als nächstes wird eine Leere Liste `tl_abrechnung` erstellt, und die Abrechnungsdaten der ausgwählten Versicherungen werden in diese Liste geschrieben, wobei beachtet wird, ob die definitive oder die provisorische Abrechnung verwendet werden soll:

```
tl_abrechnung <- list()

for (vers in versicherung) {
    if (tl_inp$PARAM_GLOBAL$abr_prov) {
        # Füge DEF und PROV nach Bedingungen hinzu
        abrechnung_def <- tl_inp[[paste0(toupper(vers), "_ABRECHNUNG_DEF")]] %>%
            filter(jahr < tl_inp$PARAM_GLOBAL$jahr_abr)
        abrechnung_prov <- tl_inp[[paste0(toupper(vers), "_ABRECHNUNG_PROV")]] %>%
            filter(jahr == tl_inp$PARAM_GLOBAL$jahr_abr)

        # Kombiniere DEF und PROV
        tl_abrechnung[[paste0(toupper(vers), "_ABRECHNUNG")]] <- rbind(abrechnung_def,
            abrechnung_prov)
    } else {
        # Nutze nur DEF, wenn abr_prov nicht aktiviert ist
        tl_abrechnung[[paste0(toupper(vers), "_ABRECHNUNG")]] <-
↪   tl_inp[[paste0(toupper(vers),
            "_ABRECHNUNG_DEF")]]
    }

}
```

Zum Schluss wird die Liste `tl_abrechnung` an `wrap_iv_vorb_berechn.R` zurückgegeben:

```
return(tl_abrechnung)
```

4.37 Modul mod_population.R

Dokumentation zuletzt aktualisiert am 14.08.2025

`mod_population.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
BEVOELKERUNG <- mod_population(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    BEV_BESTAND = tl_inp$BEV_BESTAND, BEV_SCENARIO = tl_inp$BEV_SCENARIO)
```

Die Funktion verlangt, neben `PARAM_GLOBAL` die folgenden Argumente:

- `BEV_BESTAND`: Beobachtete Bevölkerung
- `BEV_SCENARIO`: Bevölkerungsszenarien

Zu beginn des Skripts werden die Alter über 99 Jahren in den Bevölkerungsszenarien zum Alter 99 hinzugezählt. Dies, da die Bevölkerungsszenarien nachfolgend mit den Bevölkerungsbeständen verbunden werden, und in der von uns verwendeten Bevölkerungsstatistik STATPOP Alter über 99 im Alter 99 zusammengefasst werden:

```
  # BEV_SCENARIO deckt Alter 0-120 ab, BEV_BESTAND nur 0-99. BEV_SCENARIO auf Alter 0-99
  ↪  einschränken.
  BEV_SCENARIO <- BEV_SCENARIO %>%
    mutate(alt = ifelse(alt >= 100, 99, alt)) %>%  # Gruppiere alt >= 100 zu 99
    group_by(jahr, sex, nat, alt, scenario) %>%

    ↪  summarize(across(all_of(c("bevanfangj","bevendejahr","geburt","tod","einbuergerung","einwanderu
    ↪  \(x) sum(x, na.rm = TRUE)), .groups = "drop")
```

Als nächstes werden die Bevölkerungsbestände gemäss dem Parameter `jahr_bev` gesetzt, sofern dieser Parameter in `PARAM_GLOBAL` spezifiziert wurde (ansonsten werden einfach die aktuellsten Bestände verwendet):

```
    if("jahr_bev" %in% names(PARAM_GLOBAL)) {
        BEV_BESTAND <- BEV_BESTAND %>%
            mutate(
                erwbev=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,erwbev),
                ept=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,ept),
                erwq=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,erwq),
                erwqept=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,erwqept),
                frontaliers=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,frontaliers),

                ↪  assures_facultatifs=ifelse(jahr>PARAM_GLOBAL$jahr_bev+1,NA,assures_facultatifs),

                bevanfangj=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,bevanfangj),
                bevendejahr=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,bevendejahr),
                geburt=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,geburt),
                tod=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,tod),
                einbuergerung=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,einbuergerung),
```

```
                einwanderung=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,einwanderung),
                auswanderung=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,auswanderung),
                bereinigung=ifelse(jahr>PARAM_GLOBAL$jahr_bev,NA,bereinigung),
            )
    }
```

Konkret wird für die Erwerbsbevölkerung, die Grenzgänger, und die freiwillig Versicherten der Bestand bis zum Jahr `jahr_bev` verwendet, und für die Wohnbevölkerungswerte der Bestand bis zu `jahr_bev`. Die unterschiedlichen Bestände sind wie folgt begründet: Die Bestände der Erwerbsbevölkerung, die Grenzgänger, und die freiwillig Versicherten sind jeweils im März des Folgejahres verfügbar. Die Wohnbevölkerungsbestände jeweils erst im August des Folgejahres. Im Zeitraum zwischen März und August, in welchen auch die Publikation der Finanzperspektiven fällt, sind also die Bestände der Erwerbsbevölkerung, die Grenzgänger, und die freiwillig Versicherten für ein Jahr länger verfügbar als die Wohnbevölkerungsbestände. Daher wird davon ausgegangen, dass sich `jahr_bev` auf die Wohnbevölkerung bezieht, und dass die weiteren Bestände ein Jahr länger verfügbar sind.

Als nächstes werden aus den Dataframes `BEV_BESTAND` und `BEV_SCENARIO` sämtliche Variablen ausgewählt, welche für die Spätere bearbeitung mit den Szenariodaten verknüpft werden sollen (also alle Variablen ausser `c("bevanfangj", "geburt", tod", ëinbuergerung", ëinwanderung", "bereinigung", ßaisonniers", ässures_facultatifs")`):

```
    # Faktoren zur Justierung der Szenarien-Bestände auf die letzten beobachteten
    ↪  Bestände berechnen
    fractions_bevoelkerung_scen <- BEV_BESTAND |>
        select(-c("bevanfangj", "geburt", "tod", "einbuergerung", "einwanderung",
        ↪  "bereinigung", "saisonniers", "assures_facultatifs")) |>
        mutate(quelle="bestand") |>
        bind_rows(BEV_SCENARIO |>
                      filter(scenario == PARAM_GLOBAL$bev_scenario) |>
                      select(-scenario) |>
                      select(-c("bevanfangj", "geburt", "tod", "einbuergerung",
                      ↪  "einwanderung", "geburtmutter")) |>
                      mutate(quelle="scenario")
        ) |>
        pivot_longer(cols = bevendejahr:frontaliers, names_to = "variable", values_to =
        ↪  "value") |>  # Transformiere die Daten in ein langes Format
        filter(!is.na(value)) |>
        group_by(variable) |>
        filter(jahr == max(jahr[quelle == "bestand"])) |>
        ungroup() |>
        arrange(sex, nat, alt, variable, quelle) |>
        pivot_wider(names_from = quelle, values_from = value) |>  # Trenne die Szenarien
        ↪  in Spalten
        mutate(
            fraction = ifelse(
                scenario == 0 | is.na(scenario),
                1,
                bestand / scenario
            )
        ) |>  # Berechne den Bruch
        select(sex, nat, alt, variable, fraction)  # Wähle die relevanten Spalten aus
```

Für diese Variablen wird nun der Faktor berechnet, um welchen der Wert des Szenarios in der jeweiligen Alter-Geschlecht-Nationalität Zelle vom in einem gegebenen Jahr vom zuletzt beobachteten Wert abweicht.

Die Berechnung des Faktors entfällt, wenn der erste Wert des Szenarios direkt an den zuletzt beobachteten Wert anknüpft (bspw. im Jahr 2025 starten die Wohnbevölkerungsszeanrien im Jahr 2024, und die letzten beobachteten Werte reichen bis 2023, die Justierung entfällt also). Dieser Faktor wird dann später genutzt, um einen Strukturbruch im Übergang von den beobachteten Werten auf das Szenario, auf Grund von zwischen dem Szenarioerstellungszeitpunkt und dem letzten beobachteten Jahr vom Szenario abweichenden Entwicklungen, zu verhindern.[42]

Als nächstes wird der Skalierungsfaktor für die freiwillig Versicherten berechnet (die Population an freiwillig Versicherten ist lediglich für den AHV Finanzhaushalt relevant. Dieser Codeteil kann also bei Betrachtung des EL Finanzhaushalts ignoriert werden):

```
fraction_assures_facultatifs <- BEV_BESTAND |>
    select(jahr, sex, nat, alt, bevendejahr, assures_facultatifs) |>
    mutate(fraction_assures_facultatifs = ifelse(bevendejahr !=
        0, assures_facultatifs/bevendejahr, 1)) |>
    filter(!is.na(assures_facultatifs), !is.na(bevendejahr)) |>
    filter(jahr %in% c(min(jahr), max(jahr))) %>%
    group_by(sex, nat, alt) |>
    summarize(fraction_assures_facultatifs_maxjahr =
    ↪  max(fraction_assures_facultatifs[jahr ==
        max(jahr)], na.rm = TRUE), fraction_assures_facultatifs_minjahr =
        ↪  max(fraction_assures_facultatifs[jahr ==
        min(jahr)], na.rm = TRUE), .groups = "drop")
```

Da für die freiwillig Versicherten keine Szenarien existieren, wird der Bestand an freiwillig Versicherten anhand des Anteils deren an der Wohnbevölkerung im letzten beobachteten Jahr (`max(jahr)`), multipliziert mit der Wohnbevölkerungsentwicklung, fortgeschrieben.In obigem Codeblock wird nur der Faktor berechnet, mit welchem (im Codeblock unten) die Wohnbevölkerung dann multipliziert werden soll. Genau gleich wird die Anzahl an freiwillig Versicherten für die Jahre, in welchen keine registerdaten existieren (also die Jahre vor `min(jahr)`), anhand des Anteils der freiwillig Versicherten in `min(jahr)` multipliziert mit der Wohnbevölkerung des entsprechenden Jahres geschätzt.

Analog wird auch ein Skalierungsfaktor für die Grenzgänger berechnet, wobei hier Szenarien existieren, und daher nur für die Jahre vor `min(jahr)` ein Faktor berechnet werden muss (dieser Teil ist wiederum nur für den AHV Finanzhaushalt relevant, da der EL Finanzhaushalt keine Grenzgägner-Daten vor 2010 verwendet):

```
fraction_frontaliers <- BEV_BESTAND |>
    select(jahr, sex, nat, alt, bevendejahr, frontaliers) |>
    mutate(fraction_frontaliers = ifelse(bevendejahr != 0, frontaliers/bevendejahr,
        1)) |>
    filter(!is.na(frontaliers)) |>
    filter(jahr %in% min(jahr)) |>
    select(sex, nat, alt, fraction_frontaliers)
```

Zum Schluss wird das Dataframe `BEVOELKERUNG` erstellt, welches die Daten zu Bevölkerungsbeständen und -Szenarien in einer durchgehenden Zeitreihe kombiniert:

```
    BEVOELKERUNG <- BEV_BESTAND |>
        select(-c("bevanfangj", "geburt", "tod", "einbuergerung", "einwanderung",
        ↪  "bereinigung", "saisonniers", "assures_facultatifs")) |>
        mutate(quelle = "bestand") |>
```

[42] Mathematisch ist das Vorgehen äquivalent dazu, die Wachstumsraten der Szenarien nach Alter-Geschlecht-Nationalität Zelle auf den letzten beobachteten Wert anzuwenden.

```
        bind_rows(
            BEV_SCENARIO |>
                filter(scenario == PARAM_GLOBAL$bev_scenario) |>
                select(-scenario) |>
                select(-c("bevanfangj", "geburt", "tod", "einbuergerung", "einwanderung",
                ↪  "geburtmutter")) |>
                mutate(quelle = "scenario")
        ) |>
        pivot_longer(cols = bevendejahr:frontaliers, names_to = "variable", values_to =
        ↪  "value") |>  # Transformiere die Daten in ein langes Format
        filter(!is.na(value)) |>
        group_by(variable) |>
        filter(quelle == "bestand" | jahr > max(jahr[quelle == "bestand"])) |>
        ungroup() |>
        left_join(fractions_bevoelkerung_scen, by = c("sex", "nat", "alt", "variable"))
        ↪  |>  # Verknüpfe mit fractions
        mutate(value = ifelse(quelle == "scenario", value * fraction, value)) |>  #
        ↪  Multipliziere den Wert mit der entsprechenden fraction
        select(-fraction, -quelle) |>  # Entferne die fraction-Spalte
        filter(!is.na(value)) |>
        pivot_wider(names_from = "variable", values_from = "value") |>  # Transformiere
        ↪  zurück ins weite Format
        left_join(BEV_BESTAND |> select(jahr, sex, nat, alt, saisonniers,
        ↪  assures_facultatifs)) |>
        left_join(fraction_assures_facultatifs) |>
        left_join(fraction_frontaliers) |>
        mutate(
            saisonniers = ifelse(jahr >= 2000 & is.na(saisonniers), 0, saisonniers),
            assures_facultatifs =  case_when(
                    jahr < 2000 & is.na(assures_facultatifs) ~ bevendejahr *
↪  fraction_assures_facultatifs_minjahr,
                    jahr > 2000 & is.na(assures_facultatifs) ~ bevendejahr *
↪  fraction_assures_facultatifs_maxjahr,
                    TRUE ~ assures_facultatifs
                ),
            frontaliers = ifelse(jahr<2000 & is.na(frontaliers), bevendejahr *
            ↪  fraction_frontaliers, frontaliers),
            auswanderung = auswanderung
        ) |>
        select(-fraction_assures_facultatifs_maxjahr,
        ↪  -fraction_assures_facultatifs_minjahr, -fraction_frontaliers) |>
        arrange(sex, nat, alt, jahr)
```

Hierfür werden in einem ersten Teil (Teil bis `pivot_wider(names_from = "variable", values_from = "value") |>`) die Werte des Bevölkerungsszenarios an die Bevölkerungs-Bestandesdaten, für welche Szenariodaten existieren, angehängt und jeweils mit den in `fractions_bevoelkerung_scen` ermittelten Skalierungsfaktoren multipliziert. Danach werden die Bevölkerungs-Bestandesdaten für die freiwillig Versicherten und die Grenzgänger hinzugefügt, und es werden die fehlenden Werte für die frewillig Versicherten und die Grenzgänger durch Multiplikation der in `fraction_assures_facultatifs` und `fraction_frontaliers` Anteile mit der Wohnbevölkerung im jeweiligen Jahr berechnet. Die letzte if-Bedingung für die Auswanderung kann bei Betrachtung des EL Finanzhaushalts wiederum ignoriert werden, da diese nur für den AHV Finanzhaushalt relevant ist.

Somit ist die Berechnung des `BEVOELKERUNG` Dataframes abgeschlossen, und ebendieses kann an das Modul

wrap_iv_vorb_berechn.R zurückgegeben werden:

```
return(BEVOELKERUNG = BEVOELKERUNG)
```

4.38 Modul mod_eckwerte.R

Dokumentation zuletzt aktualisiert am 14.08.2025

mod_eckwerte.R wird im Skript wrap_iv_vorb_berechn.R wie folgt aufgerufen:

```
tl_eckwerte <- mod_eckwerte(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE = tl_inp$ECKWERTE, LOHNINDEX = tl_inp$LOHNINDEX,
    PREISINDEX = tl_inp$PREISINDEX)
```

Die Funktion verlangt die folgenden Argumente:

```
mod_eckwerte <- function(PARAM_GLOBAL,
                         ECKWERTE,
                         LOHNINDEX,
                         PREISINDEX
                         ) {
```

- ECKWERTE: Datafrme mit allen Projektionen zu den Eckwerten zur wirtschaftlichen Entwicklung (insbesondere der Lohn- und Preisentwicklung) gemäss ESTV.
- LOHNINDEX: Historischer Lohnindex.
- PREISINDEX: Historischer Preisindex.

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))
}
```

In einem nächsten Schritt erfolgt die Berechnung des jährlichen Lohn- und Preiswachstums:

```
# Bereitstellen Eckwerte bis vor dem ersten Jahr in
# Eckwerte Scenario --------

ECKWERTE_HIST <- LOHNINDEX %>%
    full_join(PREISINDEX) %>%
    filter(jahr >= 1979, jahr < min(ECKWERTE_SCENARIO$jahr)) %>%
    mutate(lohn = 100 * (li - lag(li))/lag(li), preis = 100 *
        (lik_basis_1977 - lag(lik_basis_1977))/lag(lik_basis_1977)) %>%
    dplyr::select(jahr, lohn, preis)
```

Als nächstes wird die vorangehend ausgewählte Eckwerte-Projektion ausgewählt:

```
# Bereitstellen Eckwerte des ausgewählten Szenarios
# --------------------------

ECKWERTE_GO <- ECKWERTE_SCENARIO %>%
    dplyr::select(jahr, lohn, 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
```

```
    )
  }
```

Somit ist die Berechnung des `ECKWERTE_SCENARIO` sowie des `ECKWERTE_EXTENDED`Dataframes abgeschlossen, wodurch diese an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden können:

```
return(list(ECKWERTE_SCENARIO = ECKWERTE_SCENARIO, ECKWERTE_EXTENDED =
↪  ECKWERTE_EXTENDED))
```

4.39 Modul mod_diskontfaktor.R

Dokumentation zuletzt aktualisiert am 25.11.2024

`mod_diskontfaktor.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
DISKONTFAKTOR <- mod_diskontfaktor(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED)
```

Die Funktion verlangt die folgenden Argumente:

```
mod_diskontfaktor <- function(PARAM_GLOBAL,
                              ECKWERTE_EXTENDED
                              ) {
```

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.38)

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)
```

```r
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:

```r
# 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_iv_vorb_berechn.R` zurückgegeben werden kann:

```r
return(DISKONTFAKTOR = DISKONTFAKTOR)
```

4.40 Modul mod_rentenentwicklung.R

Dokumentation zuletzt aktualisiert am 14.08.2025

`mod_rentenentwicklung` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```r
RENTENENTWICKLUNG <- mod_rentenentwicklung(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED, MINIMALRENTE =
    ↪  tl_inp$MINIMALRENTE,
    PREISINDEX = tl_inp$PREISINDEX)
```

Die Funktion verlangt die folgenden Argumente:

```r
mod_rentenentwicklung <- function(PARAM_GLOBAL,
                                  ECKWERTE_EXTENDED,
                                  MINIMALRENTE,
                                  PREISINDEX
                                  ) {
```

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.38)
- `MINIMALRENTE`: Historische Zeitreihe mit der im jeweiligen Jahr gemäss Verordnung gültigen Minimalrente.
- `PREISINDEX`: Historischer Preisindex.

Als Erstes wird die relevante Wachstumsrate des Preisindexes für die Minimalrentenentwicklung gemäss Gesetz/Praxis in der Vergangenheit berechnet (vor 2017 Preisindex Ende Jahr, danach Jahresmittel Preisindex, ohne Rebasierung):

```
PREISWACHSTUM <- PREISINDEX %>%
    mutate(preisindex = ifelse(jahr < 2017, lik_dez_basis_1977,
        lik_basis_1977), preis_mischindex = 100 * (preisindex -
        lag(preisindex))/lag(preisindex)) %>%
    select(jahr, preis_mischindex)
```

Danach 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):

```
  RENTENENTWICKLUNG <- ECKWERTE_EXTENDED |>
      left_join(PREISWACHSTUM) %>%
      mutate(preis=ifelse(jahr<=2017,preis_mischindex,preis)) %>%
      select(-preis_mischindex) %>%
      mutate(
          lohn = if_else(is.na(lohn), 0, lohn), # NA-Werte ersetzen
          preis = if_else(is.na(preis), 0, preis), # NA-Werte ersetzen
          lientw = cumprod(1 + lohn / 100), # Indexentwicklung berechnen
          pientw = cumprod(1 + preis / 100), # Indexentwicklung berechnen
          lohnindex = round(1004 * lientw), # Lohnindex berechnen
          preisindex = round(104.1 * pientw, 1), # Preisindex berechnen
          licomp = round(lag(lohnindex) / 10.04, 4), # Lohnkomponente berechnen
          picomp = round(lag(preisindex) / 1.041, 4), # Preiskomponente berechnen
          mischindex = round((licomp + picomp) / 2, 4), # Mischindex berechnen
          minimalrente_rd = 5 * round(5.5 * mischindex / 5) # Gerundete Minimalrente
          ↪  berechnen
      ) |>
```

Als nächstes wird die historische Minimalrente angefügt, und berücksichtigt, dass die Anpassung der Minimalrente nur alle zwei Jahre erfolgt:

```
      left_join(MINIMALRENTE, by = "jahr") |>
      mutate(
          rentenanpassung = case_when(
              jahr <= max(MINIMALRENTE$jahr, na.rm = TRUE) ~ if_else(minimalrente -
↪  lag(minimalrente) > 0, 1, 0),
              jahr == max(MINIMALRENTE$jahr, na.rm = TRUE)+1 ~ if_else(lag(minimalrente)
↪  - lag(minimalrente, 2) > 0, 0, 1),
              TRUE ~ jahr %% 2
          ), # Anpassungsrhythmus bestimmen
          minimalrente_pj = pmax(
              rentenanpassung * minimalrente_rd,
              lag(rentenanpassung * minimalrente_rd, default = 0)
          ), # Projektierte Minimalrente berechnen
          minimalrente = case_when(
              jahr <= max(MINIMALRENTE$jahr, na.rm = TRUE) ~ minimalrente,
              jahr == max(MINIMALRENTE$jahr, na.rm = TRUE)+1 ~
↪  ifelse(rentenanpassung==1,minimalrente_pj,lag(minimalrente)),
              TRUE ~ minimalrente_pj
          ) # Finalisierte Minimalrente setzen
      ) %>%
```

```
    mutate(
        dminimalrente = (minimalrente - lag(minimalrente)) / lag(minimalrente, default
        ↪  = 0), # Wachstumsrate berechnen
        rentenentwicklung = cumprod(1 + if_else(jahr > PARAM_GLOBAL$jahr_rr,
        ↪  dminimalrente, 0)) # Rentenentwicklung berechnen
    ) %>%
    select(-c(preis, lohn, dminimalrente, minimalrente_pj, minimalrente_rd, licomp,
    ↪  picomp))
```

die variable `rentenanpassung` wird so erstellt, dass sie `=1` ist für alle Jahre bis `last_year`, in welchem eine Rentenanpassung erfolgte, und danach in allen ungeraden Jahren (was Stand 2025 dem Anpassungsrythmus entspricht). 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.

Somit ist die Berechnung des `RENTENENTWICKLUNG` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden kann:

```
return(RENTENENTWICKLUNG = RENTENENTWICKLUNG)
```

4.41 Modul mod_zins.R

Dokumentation zuletzt aktualisiert am 18.03.2025

`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:

```
ZINS <- mod_zins(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL, ZINS_RAW = tl_inp$ZINS_RAW,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED)
```

Die Funktion verwendet neben `PARAM_GLOBAL` die folgenden Dataframes:

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.38)
- `ZINS_RAW`: Zinssätze für den IV-Fonds und die IV-Schulden

Das Modul `mod_zins.R` extrahiert den realzins für die nichtflüssigen Fondsanlagen und generiert aus diesem durch Addition der Inflation den Nominalzins. Zudem schreibt es die Zinssätze aus dem letzten Jahr in `ZINS_RAW` bis zum Ende des Prognosehorizonts fort:

```
ZINS <- ZINS_RAW |>
    filter(jahr >= PARAM_GLOBAL$jahr_abr) |>
    left_join(ECKWERTE_EXTENDED |>
        select(jahr, preis)) |>
    mutate(nominal_zins_fonds = (real_zins_fonds + preis)/100,
        nominal_zins_ivschuld = nominal_zins_ivschuld/100) |>
    select(jahr, nominal_zins_fonds, nominal_zins_ivschuld)

ZINS <- ZINS %>%
    bind_rows(tibble(jahr = seq(max(ZINS$jahr) + 1, PARAM_GLOBAL$jahr_ende),
```

```
        nominal_zins_fonds = ZINS %>%
            filter(jahr == max(jahr)) %>%
            pull(nominal_zins_fonds), nominal_zins_ivschuld = ZINS %>%
            filter(jahr == max(jahr)) %>%
            pull(nominal_zins_ivschuld))) %>%
    arrange(jahr)
```

Somit ist die Berechnung des `ZINS` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden kann:

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

return(ZINS = ZINS)
```

4.42 Modul mod_beitragssaetze.R

Dokumentation zuletzt aktualisiert am 18.03.2025

`mod_beitragssaetze.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
BEITRAGSSAETZE <- mod_beitragssaetze(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 dem IV-Beitragssatz.

Das Modul `mod_beitragssaetze.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 Projektionshorizonts. Danach fügt er die Zeitreihe mit den projizierten Beitragssätzen mit den historischen Beitragssätzen zusammen:

```
#------ Bereinigen und Gesamtsatz berechnen ---------------------------#
    BEITRAGSSAETZE       <- 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 `BEITRAGSSAETZE` Dataframes abgeschlossen, wodurch dieses an das Modul `wrap_iv_vorb_berechn.R` zurückgegeben werden kann:

```
return(BEITRAGSSAETZE = BEITRAGSSAETZE)
```

4.43 Modul mod_iv_filter_inp.R

Dokumentation zuletzt aktualisiert am 12.11.2024

`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 ------------------------------------------------------#
return(list(UMFRAGE_IV = UMFRAGE_IV, ESTV_IV_SCEN = ESTV_IV_SCEN))
```

4.44 Modul mod_iv_fortschreibung.R

Dokumentation zuletzt aktualisiert am 18.03.2025

`mod_iv_fortschreibung.R` wird im Skript `wrap_iv_vorb_berechn.R` wie folgt aufgerufen:

```
FORTSCHREIBUNG_IV <- mod_iv_fortschreibung(PARAM_GLOBAL = tl_inp$PARAM_GLOBAL,
    ECKWERTE_EXTENDED = tl_eckwerte$ECKWERTE_EXTENDED, ECKWERTE_SCENARIO =
    ↪  tl_eckwerte$ECKWERTE_SCENARIO,
    BEVOELKERUNG = BEVOELKERUNG, IK = tl_inp$IK, IV_ABRECHNUNG =
    ↪  tl_abrechnung$IV_ABRECHNUNG,
    AHV_ABRECHNUNG = tl_abrechnung$AHV_ABRECHNUNG, DISKONTFAKTOR = DISKONTFAKTOR,
    ESTV_IV_SCEN = tl_iv_input$ESTV_IV_SCEN, RENTENENTWICKLUNG = RENTENENTWICKLUNG,
    BEV_BESTAND = tl_inp$BEV_BESTAND, BEITRAGSSAETZE = BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = tl_inp$ARBEITSLOSENQUOTE)
```

Die Funktion verwendet neben `PARAM_GLOBAL` das folgende Dataframe:

- `ECKWERTE_EXTENDED`: Zeitreihe mit der historischen und der projizierten Lohn- und Preisentwicklung (vgl. Kapitel 4.38)
- `ECKWERTE_SCENARIO`: Liste mit allen Eckwerten der ESTV. Dies wird hier zusätzlich zum Dataframe `ECKWERTE_EXTENDEN` benötigt, da in `ECKWERTE_EXTENDEN` lediglich die Lohn- und Preisentwicklung aus dem relevanten Eckwerte Szenario abgebildet ist, wir hier aber zusätzlich noch die Entwicklung der Arbeitslosenquote und der BESTA-Beschäftigung benötigen (für die Schätzung der AHV Lohnsumme).
- `BEVOELKERUNG` und `BEV_SCENARIO`: 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. Das Dataframe `IK` wird ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `IV_ABRECHNUNG`: Die IV-Abrechnung wird zur Bestimmung des vergangenen Bundesbeitrages verwendet.
- `AHV_ABRECHNUNG`: Die AHV-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.
- `RENTENENTWICKLUNG`: Lohn- und Mischindex für die Projektion der Wachstumsrate des Bundesbeitrages.
- `BEV_BESTAND`: Die Bestandesdaten zur Bevölkerung werden ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `BEITRAGSSAETZE`: Die Beitragssätze der AHV werden ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.
- `ARBEITSLOSENQUOTE`: Die historischen Arbeitslosenquoten werden ausschliesslich im Untermodul `mod_beitragssumme` für die Projektion der Lohnsumme verwendet.

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:

```
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:

```
# Lohnsummen-Wachstum berechnen
tl_iv_beitrag <- mod_beitragssumme(PARAM_GLOBAL = PARAM_GLOBAL,
    BEVOELKERUNG = BEVOELKERUNG, BEV_BESTAND = BEV_BESTAND, IK = IK,
    ECKWERTE_EXTENDED = ECKWERTE_EXTENDED, ECKWERTE_SCENARIO = ECKWERTE_SCENARIO,
    AHV_ABRECHNUNG = AHV_ABRECHNUNG, BEITRAGSSAETZE = BEITRAGSSAETZE,
    ARBEITSLOSENQUOTE = ARBEITSLOSENQUOTE, versicherung = "iv",
    ABRECHNUNG = IV_ABRECHNUNG)
```

Hierfür wird zuerst das Modul `mod_beitragssumme.R` ausgeführt (vgl. Kapitel 4.32.1), 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:[43]

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

```
# Wachstumsfaktor für Bundesbeitrag berechnen:
FORTSCHREIBUNG_IV      <- FORTSCHREIBUNG_IV %>%
    left_join(RENTENENTWICKLUNG %>%
                  # 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'))
    ) %>%
    left_join(IV_ABRECHNUNG %>% select(jahr, btr_bund)) %>%
    mutate(mwst_zuwachs = if_else(!is.na(mwst_zuwachs), mwst_zuwachs*100, lohnsumme),
           mwst_diskont = mischidx / lag(mischidx) * lag(lientw, 2) / lag(lientw),
           cumgrowth_bundesbeitrag = cumprod(
               1 + if_else(jahr > PARAM_GLOBAL$jahr_abr,
                           ((1 + mwst_zuwachs / 100) * mwst_diskont - 1),
                           0)
           ),
           berechnet_btr_bund = ifelse(jahr<PARAM_GLOBAL$jahr_abr, btr_bund,
           ↪  btr_bund[which(jahr == PARAM_GLOBAL$jahr_abr)] *
           ↪  cumgrowth_bundesbeitrag)
    ) %>%
    select(-lientw,  -pientw, -lohnindex, -mischidx, -mwst_zuwachs, -mwst_diskont,
    ↪  -btr_bund, -cumgrowth_bundesbeitrag)
```

Hierzu wird das Dataframe `RENTENENTWICKLUNG` (vgl. Kapitel 4.40) 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 der berechnete Bundesbeitrages, welcher in der Variable `berechnet_btr_bund` 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 ------------------------------------------------------------#
return(FORTSCHREIBUNG_IV = FORTSCHREIBUNG_IV)
```