DAV

DEUTSCHE
AKTUARVEREINIGUNG e.V.

Schriftliche Prüfung im Vertiefungswissen

Actuarial Data Science Completion

gemäß Prüfungsordnung 5
der Deutschen Aktuarvereinigung e. V.

Zeitraum: 11.09.2023 – 11.10.2023

Hinweise:

- Die Gesamtpunktzahl beträgt 180 Punkte. Die Klausur ist bestanden, wenn mindestens 90 Punkte erreicht werden.

- Bitte prüfen Sie die Ihnen vorliegende Prüfungsklausur auf Vollständigkeit. Die Prüfung besteht aus zwei Blöcken A, B und beinhaltet insgesamt 9 Seiten. Zusätzliche Materialen (Datensätze etc.) sollten Sie mit der Prüfung erhalten haben. Auf diese wird in den Blöcken eingegangen.

- Für jeden Block ist ein PDF-Dokument einzureichen. Falls ein Notebook zu erstellen ist, dann ist dieses zusätzlich einzureichen. Auch die Notebooks sind einschließlich des Outputs einzureichen. Aufgabenteile mit erforderlichem, aber fehlendem Output werden mit 0 Punkten bewertet.

- Für jeden Block sind die Aufgaben in der vorgegebenen Reihenfolge zu bearbeiten. Die Lösung muss direkt nach der Aufgabenstellung in einer strukturierten und übersichtlichen Weise dargestellt werden. Nichtbeachtung dieser Vorgabe führt zu Punkteabzug.

- Mit der Einreichung eines Ergebnisnotebooks erklärt die zu prüfende Person, dass der vorliegende Ergebnisbericht inklusive Code- und Beschreibungsanteilen eigenständig erzeugt worden ist. Verstöße gegen diese Grundlagen

können vom Prüfungsausschuss sanktioniert werden und zum Ausschluss von der Prüfung führen.

Mitglieder der Prüfungskommission:

Steffen Lotter, Prof. Dr. Fabian Transchel, Dr. Johannes Schupp

Block A (Rechtsschutzversicherung) *[60 Punkte]*

Hinweise:

- Für diesen Block ist ein Notebook (R-Markdown oder Jupyter Notebook) mit R oder Python zu erstellen. Zudem ist ein Report im Notebook- oder PDF-Format zu erstellen.
- Benötigte Materialien: Vertragsdaten.csv, Schadendaten.csv

Sie sind für einen deutschlandweit aktiven Erstversicherer im Komposit-Aktuariat tätig. Hier kommt es zu folgender Konstellation: Nach Einführung eines neuen Rechtsschutz-Produkts fällt der Schadenabteilung auf, dass binnen kurzer Zeit eine erhebliche Anzahl Schäden gemeldet wird. Diese lassen sich auf ein Schlupfloch in den Versicherungsbedingungen zurückführen, die einen spezifisch üblichen Deckungsausschluss bzgl. streitiger Lebensversicherungsverzinsungsfälle nicht enthielten. Nach einiger Recherche wird klar, dass findige Kanzleien, die in Verzinsungsfällen mandatiert sind, allen Mandanten zum Abschluss des Produkts geraten habe. Es ist daher davon auszugehen, dass zu den bereits gemeldeten noch weitere Schäden hinzukommen werden.

Als die Gesellschaft auf den Umstand aufmerksam wird, sind etwa 2.400 Policen abgeschlossen worden. Der Produktvertrieb wird umgehend gestoppt und erst nach Korrektur der Vertragsbedingungen wieder angefahren. Es ist dennoch ein nicht unerheblicher Schaden entstanden.

Ihr Bereichsleiter muss dem Vorstand aufgrund der Situation einen Bericht erstatten. Sie bereiten dazu die gemeinsam mit der Schadenabteilung erarbeiteten Erkenntnisse quantitativ auf.

Die konkreten Vorgaben Ihres Chefs lauten:

Aufgabe 1. *[Explorative Datenanalyse] [10 Punkte]*

Fertigen Sie eine Übersicht über den Bestand an. Differenzieren Sie, wo sinnvoll, die wesentlichen Eigenschaften der einzelnen Verträge und Schadenfälle.

Aufgabe 2. *[Use Case: Eingrenzung des Effekts] [15 Punkte]*

Geben Sie eine Schätzung ab, wie viele Schadenfälle aufgrund der spezifischen Situation noch zusätzlich zu erwarten sind. Begründen Sie die dabei getroffenen Annahmen.

Aufgabe 3. [Use Case: Ökonomische Bewertung] *[10 Punkte]*

Schätzen Sie den ökonomischen Schaden quantitativ ab. Begründen Sie getroffenen Annahmen und Abschätzungen.

Aufgabe 4. *[Anomalieerkennung] [15 Punkte]*

Es soll für zukünftige Fälle eine Frühwarn-Heuristik auf Basis von Algorithmen der Anomalie-Erkennung konstruiert werden. Skizzieren Sie das Vorgehen, wie auf Basis der vorliegenden Daten die Kalibrierung eines solchen Systems aussehen könnte und weisen Sie die Tauglichkeit mittels eines geeigneten Algorithmus nach.

Aufgabe 5. *[Reporting von Data Analytics] [10 Punkte]*

Fassen Sie abschließend die Erkenntnisse in einem PDF/HTML-File vom Umfang maximal 2 Seiten (inkl. Grafiken) zusammen, wobei 1/2 Seite Management Summary und 1 1/2 Seiten Beschreibung der Erkenntnisse sein sollen.

(Hinweis: Dies muss nicht als separate Datei erfolgen, sofern Sie aus ihrem Notebook ein PDF oder HTML erzeugen.)

Beschreibung des Datensatzes

Vertragsdaten.csv

Vertragsnummer (int), Jahreseinheiten (float), Alter (int in [18,99], Familienstand (0=ledig, 1=verheiratet, 2=geschieden, 3=verwitwet), Tätigkeitstyp (1=Vollzeit, 2=Teilzeit, 3=Ausbildung, 4=erwerbslos, 5=Rente, 6=unbekannt), Abschlussart (1=Makler, 2=Direktvertrieb, 3=Aggregator, 4=andere), Bausteine (direkte Summe aus: 1=Standard, 2=Bonusleistung, 4=Kostenlose Beratung, 8=Freie Anwaltswahl), Vertragsdatum (timestamp), Vermittler (0=direkt, 1-9998=Makler-ID, 9999=unbekannt), Prämie (float)

Schadendaten.csv

Vertragsnummer (int), Schadenmeldungsnummer (int), Schadentyp (1=Beklagt, 2=Kläger, 3=Sonderfall), Reserve (float), Problemfall (0=kein Problem i.S.d. Aufgabe, 1=Problemfall i.S.d. Aufgabe), Schadendatum (timestamp), Endreserve (float)

Block B (Risikolebensversicherung) *[120 Punkte]*

- Für diesen Block ist ein Notebook (R-Markdown oder Jupyter Notebook) mit R oder Python zu erstellen.
- Benötigte Materialien: 2015-vbt-unismoke-alb-anb.xlsx, bav_Bestand.csv, DAV 2008 T mit R-NR.xlsx, NHANES_Extract.csv

Aufgabe 1. *[Business Case Risikolebensversicherung I] [35 Punkte]*

Sie sind im Aktuariat der Lebensversicherung „Alte Leben", einem etablierten Unternehmen am Markt, angestellt. In nahezu allen Produktlinien sind die Verkaufszahlen rückläufig. Sie haben als Unternehmen deshalb den Vorstand Frau Anders von der Konkurrenz abgeworben. Diese bittet Sie, ein neues Produkt für die Risikolebensversicherung zu entwerfen und meint zu ihnen: "Die Konkurrenz biete beispielsweise einen Akademikerbonus von 5 % an. Da müsste doch noch viel mehr gehen oder es müssten sich noch andere beitragsrelevante Merkmale finden lassen. Können wir nicht weitere lukrative Segmente identifizieren, die wir dann mit einem differenzierten Pricing ansprechen können?!".

Sie sind in der DAV gut vernetzt und sprechen mit ihrem Netzwerk. Herr Dr. Netzer berichtet von seinen Erfahrungen bei der Erstellung eines Ergebnisberichts für den Ausschuss Leben, vgl. https://aktuar.de/unsere-themen/fachgrundsaetze-oeffentlich/2022-09-21-DAV-Ergebnisbericht_Big_Data_Leben_NHANES_final.pdf. Dabei wurde ein US-Datensatz „NHANES" durch die Arbeitsgruppe aufbereitet, der seiner Ansicht nach zur Analyse dieser Fragestellungen prinzipiell geeignet wäre. Die Übertragbarkeit von Erkenntnissen aus anderen Ländern sei auch bei anderen Untersuchungen gängig und möglich, z.B. Pflege- oder Sterbetafeln. Der Datensatz für diese Prüfungsaufgabe ist aus diesem Datensatz abgeleitet.

Sie entschließen sich mögliche Risikofaktoren für ein differenziertes Pricing auf diesem Datensatz zu analysieren und diese Erkenntnisse dann in Deutschland für ihr Pricing auf Basis der DAV2008T 2. Ordnung anzuwenden.

(a) *Explorative Datenanalyse und Verständnis*

Lesen Sie die Daten ein. Es handelt sich um eine Kohorten-Studie, verwenden Sie daher zu jeder ID nur die neueste Beobachtung. *Kurzbeschreibung (weitere Details entnehmen sie z.B. dem Ergebnisbericht): Eine im Jahr „year_DOB" geborene Personen wurden im Jahr „year_DOS" erfasst und dabei wurden zahlreiche medizinische und sozioökonomische Merkmale erhoben. Im Jahr „year_DOE" ist die Person entweder verstorben („Death" =1) oder die Person*

lebt. Somit wurde die Person „Surv" Monate während der Studie beobachtet. Führen Sie eine explorative Datenanalyse durch. Visualisieren Sie u.a. den Einfluss verschiedenerer Variablen auf die Überlebenszeit. Berücksichtigen Sie dabei auch die Kovariable Beschäftigungsstatus, die als Proxy für den Akademiker-Status dienen kann. Diskutieren Sie die Übertragbarkeit auf den Akademiker-Status. Was würden Sie zusätzlich benötigen, um hier noch sicherer zu sein?

(b) *Survival Modell*

Entwickeln Sie ein CoxPH-Modell als Referenzmodell, um abschätzen zu können, ob der Datensatz ihren Erwartungen hinsichtlich des Einflusses auf die Überlebenszeit entspricht.

Das 1972 von David Cox vorgeschlagene Regressionsmodell ist eines der bekanntesten und verbreitetsten Verfahren zur Modellierung von Überlebenszeiten, welches es ermöglicht, den Einfluss von Kovariablen (x) auf die Überlebenszeit zu analysieren. Die Hazard-Rate oder Ausfallrate wird hier über ein semiparametrisches Modell modelliert:

$$h(t; x_i) = h_0(t) * \exp(x_i \beta).$$

Dabei seien t die Zeit, $h_0(t)$ die für alle Individuen identische Baseline Hazard-Rate und x_i die Kovariablen, z.B. Alter und Geschlecht für die i-te Beobachtung. Bei der Kalibrierung wird der Koeffizientenvektor β geschätzt, der den Einfluss der einzelnen Kovariablen beschreibt.

Verwenden Sie als erklärende Kovariablen die Kovariablen Einkommen, Beziehungsstatus, Rauchstatus, Geschlecht, Beschäftigungsverhältnis und verdichten Sie diese vorab geeignet.

Bilden Sie ein weiteres Modell, welches zusätzlich die Kovariable des Alters bei Beginn der Kohortenstudie berücksichtigt. Vergleichen und Interpretieren Sie die Ergebnisse.

Diskutieren Sie die Anwendbarkeit ihres CoxPH-Modells zur Ableitung eines differenzierten Pricings ihrer Risikolebensversicherung.

Aufgabe 2. *[Datenbearbeitung] [25 Punkte]*

Sie haben erkannt, dass das Alter sehr wichtig für die Prognose der Sterblichkeit ist und sie entschließen sich deshalb eine Tafel der Amerikanischen Aktuarsvereinigung (SOA) als Baseline-Sterblichkeit vorzugeben. Das Modell muss somit die altersbedingte Sterblichkeitsstruktur nicht selbst erlernen. Sie verwenden die Ultimate qx (Spalte ult.) der Unismoke VBT Tafel 2015, vgl.

https://www.soa.org/resources/experience-studies/2015/2015-valuation-basic-tables/

Gehen Sie davon aus, dass der vorliegende Datensatz einer repräsentativen Stichprobe der Gesamtbevölkerung der USA entspricht. Wie in Deutschland ist auch dort das Sterblichkeitsniveau von Versicherten geringer als das Sterblichkeitsniveau der Gesamtbevölkerung. Deshalb erhöhen Sie die qx der SOA mit einem pauschalen Multiplikator m1 und einem additiven Term m2 für die Alter >= 50, d.h. $qx_{Baseline} = m1 \cdot qx_{VBT} + m2 \cdot 1_{\{x \geq 50\}}$. Ermitteln Sie diese Faktoren auf Basis der vorliegenden Daten getrennt nach Männern und Frauen und einmal gemeinsam.

(a) *Zensierung von Daten und Bearbeitung*

Ermitteln Sie für jedes Alter ab 25 die empirischen Sterbewahrscheinlichkeiten $qx_{Empirisch}$ des Datensatzes. Berücksichtigen Sie Links- und Rechtszensierung angemessen.

Um ein Maß für die Unsicherheit zu erhalten, ermitteln Sie zusätzlich einen angemessenen Konfidenzbereich auf Basis der empirischen Sterbewahrscheinlichkeit.

(b) *VBT-Tafeln visualisieren und Faktoren ermitteln*

Lesen Sie die VBT-Tafeln ein und visualisieren Sie diese für Männer und Frauen gemeinsam mit der empirischen Sterbewahrscheinlichkeit und dem Konfidenzintervall aus Teilaufgabe 2a). Begründen Sie die prinzipielle Angemessenheit der VBT-Tafeln für die Anwendung.

Ermitteln Sie optimale Parameter m1, m2 (Optimierungs-Maß: gewichteter quadratischer Fehler zwischen $qx_{Empirisch}$ und $qx_{Baseline}$) für Männer und Frauen getrennt und gemeinsam. Welche Faktoren verwenden Sie anschließend für die Analyse? Verwenden Sie nur den für die spätere Anwendung relevanten Altersbereich bis einschließlich Alter 70.

Aufgabe 3. *[Insurance Analytics] [50 Punkte]*

(a) *Datensatz transformieren*

Überführen Sie die Aufgabenstellung in ein Klassifikationsproblem. Erzeugen Sie dazu aus jedem Datensatz (i.A.) mehrere Zeilen, indem Sie für jedes beobachtete Altersjahr der Person schauen, ob die betroffene Person gestorben ist (label=1) oder ob diese überlebt hat (label=0), siehe Tabelle 1. Berücksichtigen

Sie auch die Zensierung der Daten. Verwenden Sie nur den für das Pricing besonders relevanten Altersbereich bis einschließlich Alter 70.

ID	AGE	SMOKER	LABEL
23	60	1	0
23	61	1	0
23	62	1	1
46	58	0	0

Tabelle 1: Beispieldatensatz Klassifikation

(b) *GLM*

Entwickeln Sie ein GLM unter Verwendung der Kovariablen Einkommen, Beziehungsstatus, Raucherstatus, Geschlecht und Beschäftigungsverhältnis. Zudem berücksichtigen Sie Vorerkrankungen Diabetes und den BMI, die ggf. zu Zuschlägen im Pricing führen und deshalb hier berücksichtigt/kontrolliert werden sollen.

Wählen Sie eine geeignete Verteilungsannahme und setzen Sie die VBT-Tafel erhöht um den Faktor 10% und addiert mit 0,3% ab Alter 50 als Baseline-Sterblichkeit an. Imputieren Sie fehlende Werte des BMI geeignet und begründen Sie die getroffene Maßnahme.

Plausibilisieren Sie die Ergebnisse. Gibt es neben dem Akademikerbonus weitere Ansatzpunkte für Differenzierung? Welche würden Sie vorschlagen im Pricing zu nutzen? Welche Versicherungsnehmer sollten im Rahmen einer Risikoprüfung grundsätzlich ausgeschlossen werden? Stellen Sie die vorhergesagten Sterbewahrscheinlichkeiten des Modells für eine ID und ein Alter anhand eines Wasserfalldiagramms dar.

(c) *CatBoost*

Entwickeln Sie ein CatBoost-Modell zum Vergleich unter Verwendung der Kovariablen Einkommen, Beziehungsstatus, Rauchstatus, Geschlecht und Beschäftigungsverhältnis. Zudem berücksichtigen Sie Vorerkrankungen Diabetes und den BMI, die ggf. zu Zuschlägen im Pricing oder zu Ausschlüssen in der Risikoprüfung führen und deshalb hier berücksichtigt/kontrolliert werden sollen.

Zerlegen Sie den Datensatz in Training und Validierung. Definieren Sie ein CatBoost-Modell und kalibrieren Sie dieses. Wählen Sie eine geeignete Parametrisierung (Hinweis: ein ausführliches Parameter-Tuning ist nicht notwendig) und setzen Sie die VBT-Tafel erhöht um den Faktor 10% und addiert mit 0,3% ab Alter 50 als Baseline-Sterblichkeit an.

Plausibilisieren Sie die Ergebnisse. Erstellen Sie Modellvorhersagen und vergleichen Sie diese mit dem GLM und mit der empirischen Beobachtung sowie der in Aufgabe 2 ermittelten Baseline Sterblichkeit. Bilden Sie dazu mittlere Vorhersagen für jedes Alter und vergleichen Sie diese (jeweils für Training und Test) visuell.

(d) *Modellverständnis und Interpretation*

Machen Sie den Einfluss der Merkmale Einkommen und Beschäftigungsart anhand eines Accumulated Dependence Plots sichtbar. Was ist der Unterschied zu den häufiger verwendeten Partial Dependence Plots? Erstellen Sie einen Partial Dependence Plot für das Merkmal BMI. Vergleichen Sie die Ergebnisse mit denen des GLM. Welche Auffälligkeit sehen Sie beim BMI und wie ist diese zu begründen und ggf. zu beheben?

Zerlegen Sie die Vorhersage des CatBoost-Modells erneut für eine einzelne Vorhersage des Validierungsdatensatzes mittels des SHAP-Value. Interpretieren Sie das Ergebnis. Diskutieren Sie die Vergleichbarkeit der Ergebnisse mit denen des GLMs.

Entwickeln Sie einen SHAP-Force Plot für eine hinreichend große Stichprobe des Validierungsdatensatzes. Gruppieren Sie nach dem Geschlecht. Interpretieren Sie das Ergebnis. Was leiten Sie daraus ab?

Aufgabe 4. *[Business Case Risikolebensversicherung II] [10 Punkte]*

Verwenden und übertragen Sie das erstellte CatBoost/GLM-Modell für das Pricing in Deutschland. Ermitteln Sie den Preis für einen geschlossenen bAV-Bestand (bav_Bestand.csv) mit einjähriger Risikoleben inkl. pauschal 5 EUR Stückkosten und Sicherheitszuschlag mit einer Versicherungssumme von 100.000 EUR. D.h. der Preis einer einzelnen Police i beträgt: $P(i) = 5EUR + qx(i) * 100.000EUR$. Verwenden Sie die DAV2008T 2. Ordnung mit einem Mischungsverhältnis von Mann:Frau = 60:40. Welchen Preis stellen Sie dem Unternehmen in Rechnung? Welchen Akademikerbonus rechnen Sie dabei ein?

Aufgabenteil A - Rechtsschutzversicherung

In [9]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import random
import seaborn as sns
from datetime import datetime
from datetime import timedelta
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
```

Hinweise:

- Für diesen Block ist ein Notebook (R-Markdown oder Jupyter Notebook) mit R oder Python zu erstellen. Zudem ist ein Report im Notebook- oder PDF-Format zu erstellen.
- Benötigte Materialien: Vertragsdaten.csv, Schadendaten.csv

Sie sind für einen deutschlandweit aktiven Erstversicherer im Komposit-Aktuariat tätig. Hier kommt es zu folgender Konstellation: Nach Einführung eines neuen RS-Produkts fällt der Schadenabteilung auf, dass binnen kurzer Zeit eine erhebliche Anzahl Schäden gemeldet wird. Diese lassen sich auf ein Schlupfloch in den Versicherungsbedingungen zurückführen, die einen spezifisch üblichen Deckungsausschluss bzgl. streitiger Lebensversiche-rungsverzinsungsfälle nicht enthielten. Nach einiger Recherche wird klar, dass findige Kanzleien, die in Verzinsungsfällen mandatiert sind, allen Man-danten zum Abschluss des Produkts geraten habe; es ist daher davon aus-zugehen, dass zu den bereits gemeldeten noch weitere Schäden hinzukom-men werden.

Als die Gesellschaft auf den Umstand aufmerksam wird, sind etwa 2.400 Policen abgeschlossen worden. Der Produktvertrieb wird umgehend gestoppt und erst nach Korrektur der Vertragsbedingungen wieder angefahren. Es ist dennoch ein nicht unerheblicher Schaden entstanden. Ihr Bereichsleiter muss dem Vorstand aufgrund der Situation einen Bericht erstatten. Sie bereiten dazu die gemeinsam mit der Schadenabteilung erar-beiteten Erkenntnisse quantitativ auf. Die konkreten Vorgaben Ihres Chefs lauten:

Aufgabe 1. [Explorative Datenanalyse] [10 Punkte]

Fertigen Sie eine Übersicht über den Bestand an. Differenzieren Sie, wo sinnvoll, die wesentlichen Eigenschaften der einzelnen Verträge und Schadensfälle.

Aufgabe 2. [Use Case: Eingrenzung des Effekts] [15 Punkte]

Geben Sie eine Schätzung ab, wie viele Schadensfälle aufgrund der spezifischen Situation noch zusätzlich zu erwarten sind. Begründen Sie dabei getroffene Annahmen.

Aufgabe 3. [Use Case: Ökonomische Bewertung] [10 Punkte]

Schätzen Sie den ökonomischen Schaden quantitativ ab. Begründen Sie getroffene Annahmen und Abschätzungen.

Aufgabe 4. [Anomalieerkennung] [15 Punkte]

Es soll für zukünftige Fälle eine Frühwarn-Heuristik auf Basis von Algorith-men der Anomalie-Erkennung konstruiert werden. Skizzieren Sie das Vorge-hen, wie auf Basis der vorliegenden Daten die Kalibrierung eines solchen Systems aussehen könnte und weisen Sie die Tauglichkeit mittels eines geeigneten Algorithmus nach.

Aufgabe 5. [Reporting von Data Analytics] [10 Punkte]

Fassen Sie abschließend die Erkenntnisse in einem PDF/HTML-File vom Um-fang maximal 2 Seiten zusammen, wobei 1/2 Seite Management Summary und 1 1/2 Seiten Beschreibung der Erkenntnisse sein sollen.

(Hinweis: Dies muss nicht als separate Datei erfolgen, sofern Sie aus ihrem Notebook ein PDF oder HTML erzeugen.)

Beschreibung des Datensatzes

Vertragsdaten.csv

- Vertragsnummer (int),
- Jahreseinheiten (float),
- Alter (int in [18,99]),
- Familienstand (0=ledig, 1=verheiratet, 2=geschieden, 3=verwitwet),
- Tätigkeitstyp (1=Vollzeit, 2=Teilzeit, 3=Ausbildung, 4=erwerbslos, 5=Rente, 6=unbekannt),
- Abschlussart (1=Makler, 2=Direktvertrieb, 3=Aggregator, 4=andere),
- Bausteine (direkte Summe aus: 1=Standard, 2=Kostenlose Beratung, 4=Freie Anwaltswahl, 8=Bonusleistung),
- Vertragsdatum (timestamp),
- Vermittler (0=direkt, 1-9998=Makler-ID, 9999=unbekannt),
- Prämie (float)

Schadendaten.csv

- Vertragsnummer (int),
- Schadenmeldungsnummer (int),
- Schadentyp (1=Beklagt, 2=Kläger, 3=Sonderfall),
- Reserve (float),
- Problemfall (0=kein Problem i.S.d. Aufgabe, 1=Problemfall i.S.d. Aufgabe),
- Schadendatum (timestamp),
- Reserve (float),
- Endreserve (float)

Musterlösung

Aufgabe 1

Um uns der Frage qualitativ zu nähern, wird zunächst eine normale explorative Datenanalyse durchgeführt. Hiermit ist die Hoffnung verknüpft, *interessante* bzw. *relevante* Aspekte zu identifizieren.

EDA

In [10]:
```
df_contracts = pd.read_csv("vertragsdaten.csv")
df_claims = pd.read_csv("schadendaten.csv")
```

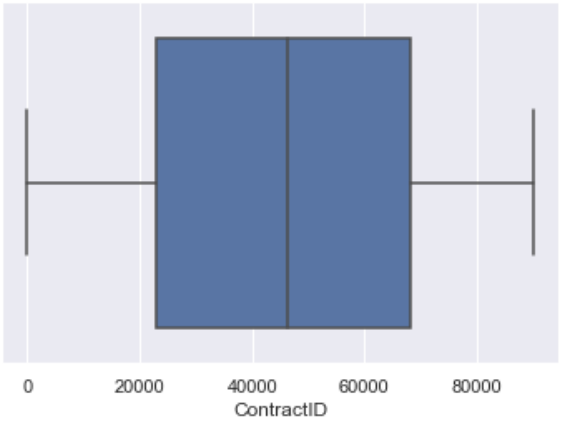
Wir beginnen die Analyse mit Boxplots und einem Pairplot für `df_contracts`.

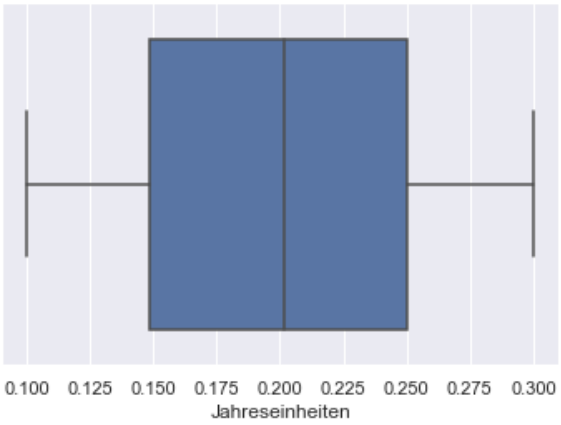
In [11]:
```
sns.color_palette("YlOrBr", as_cmap=True, n_colors=4)
sns.set()
sns.pairplot(df_contracts, hue="Abschlussart")
```

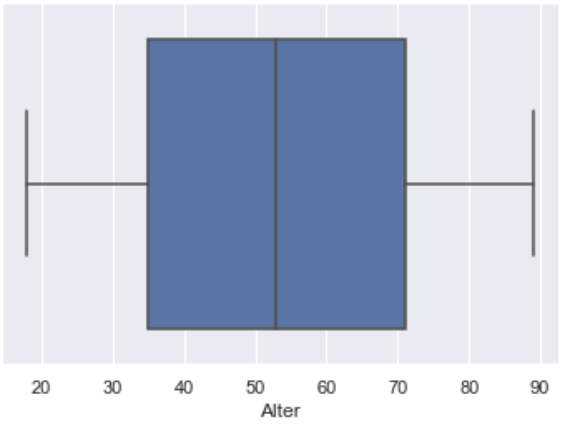
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x1b88e138160>
```

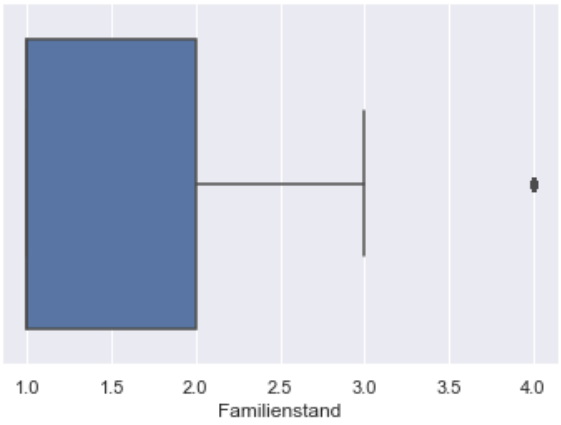
Bevor wir zu detaillierteren Übersichten kommen, scheint es nützlich, zusätzlich zum Pairplot doch auch noch Boxplots anzusehen:

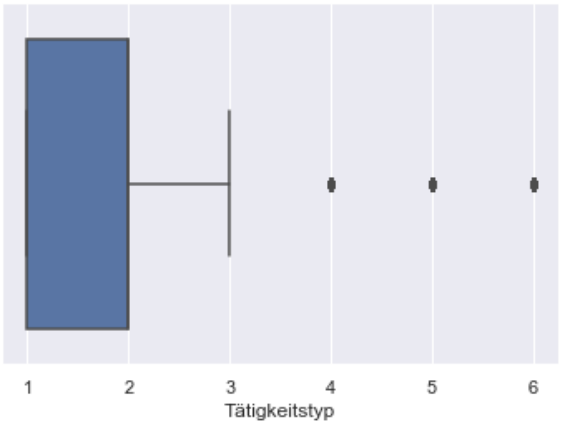
In [12]:
```
contracts_numeric = df_contracts.select_dtypes(include=['int', 'float'])
contracts_numeric_columns = list(contracts_numeric.drop('Unnamed: 0',axis=1).columns)
for column in (contracts_numeric_columns):
    sns.boxplot(data=df_contracts, x=column)
    plt.show()
```

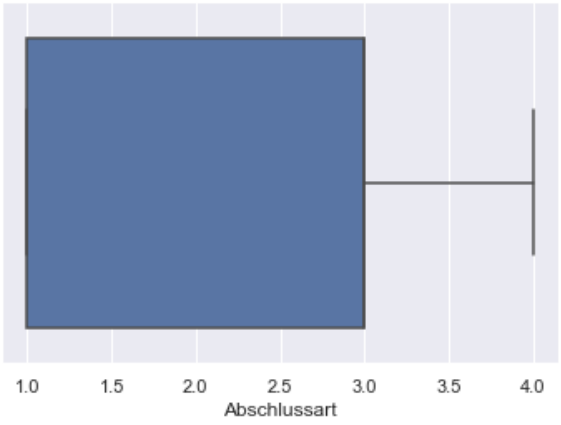


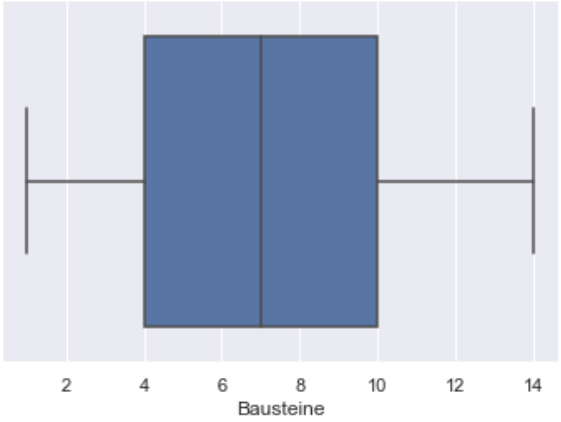

0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275 0.300
Jahreseinheiten

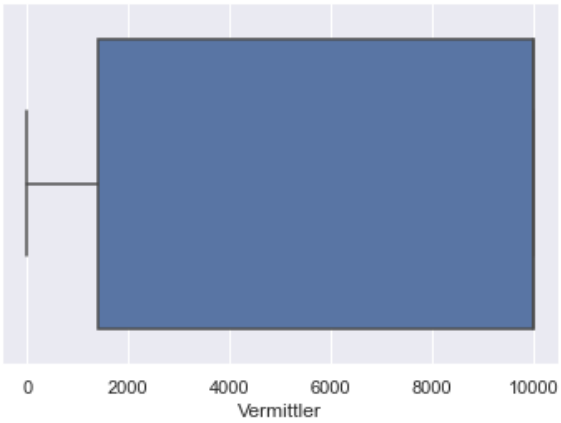

20 30 40 50 60 70 80 90
Alter

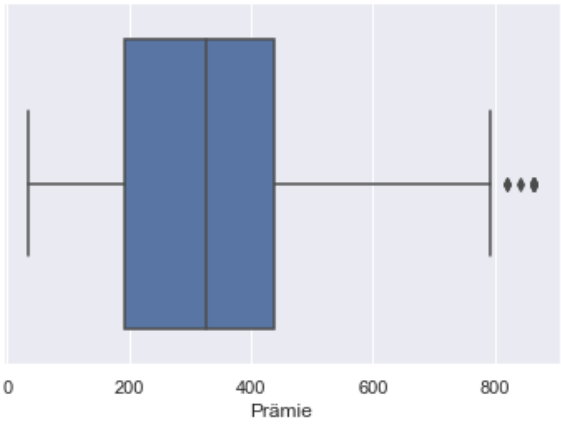

1.0 1.5 2.0 2.5 3.0 3.5 4.0
Familienstand


1 2 3 4 5 6
Tätigkeitstyp

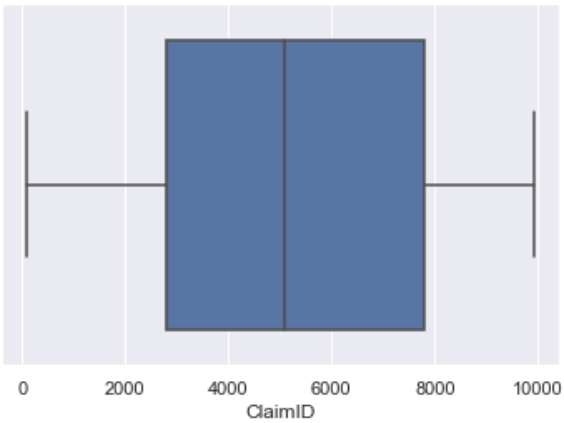

1.0 1.5 2.0 2.5 3.0 3.5 4.0
Abschlussart

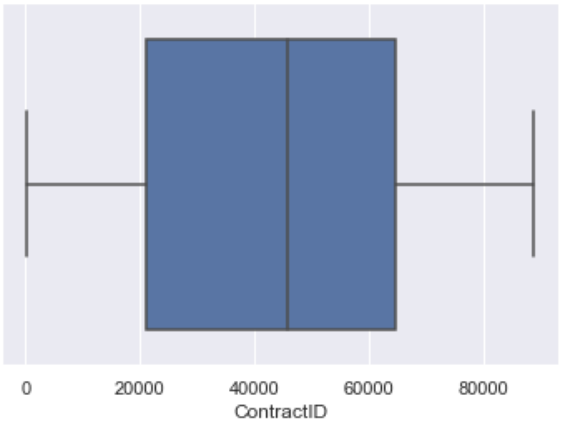


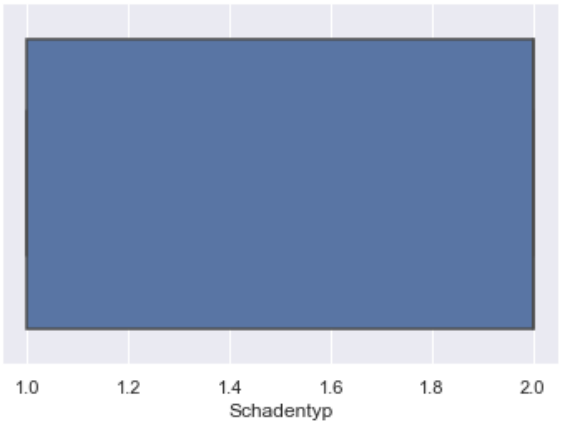


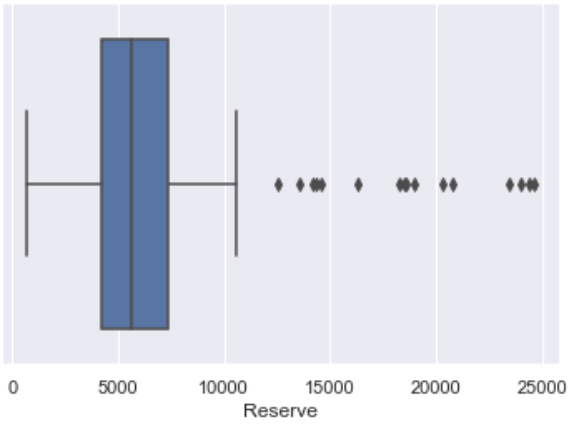


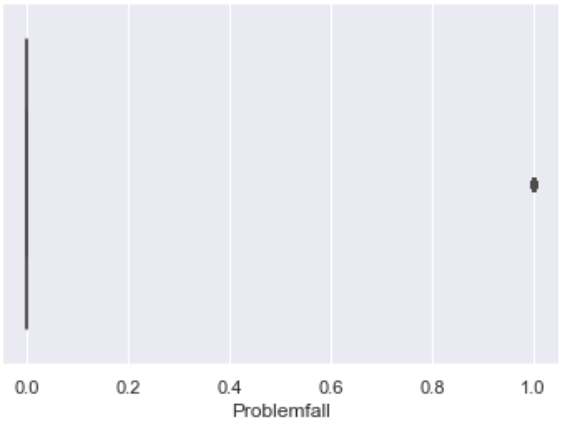
In [13]:
```
claims_numeric = df_claims.select_dtypes(include=['int', 'float'])
claims_numeric_columns = list(claims_numeric.drop('Unnamed: 0',axis=1).columns)
for column in (claims_numeric_columns):
    sns.boxplot(data=df_claims, x=column)
    plt.show()
```

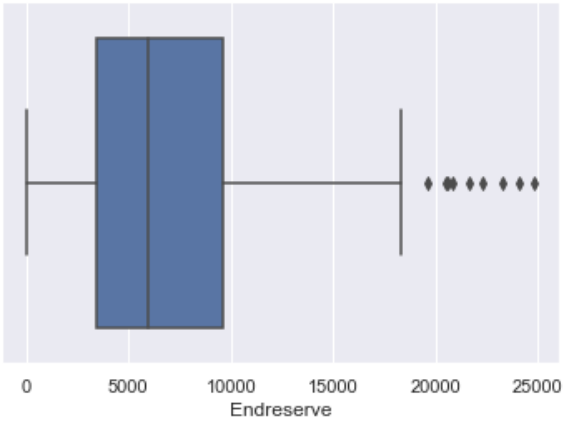



0
20000
40000
60000
80000
ContractID


1.0
1.2
1.4
1.6
1.8
2.0
Schadentyp

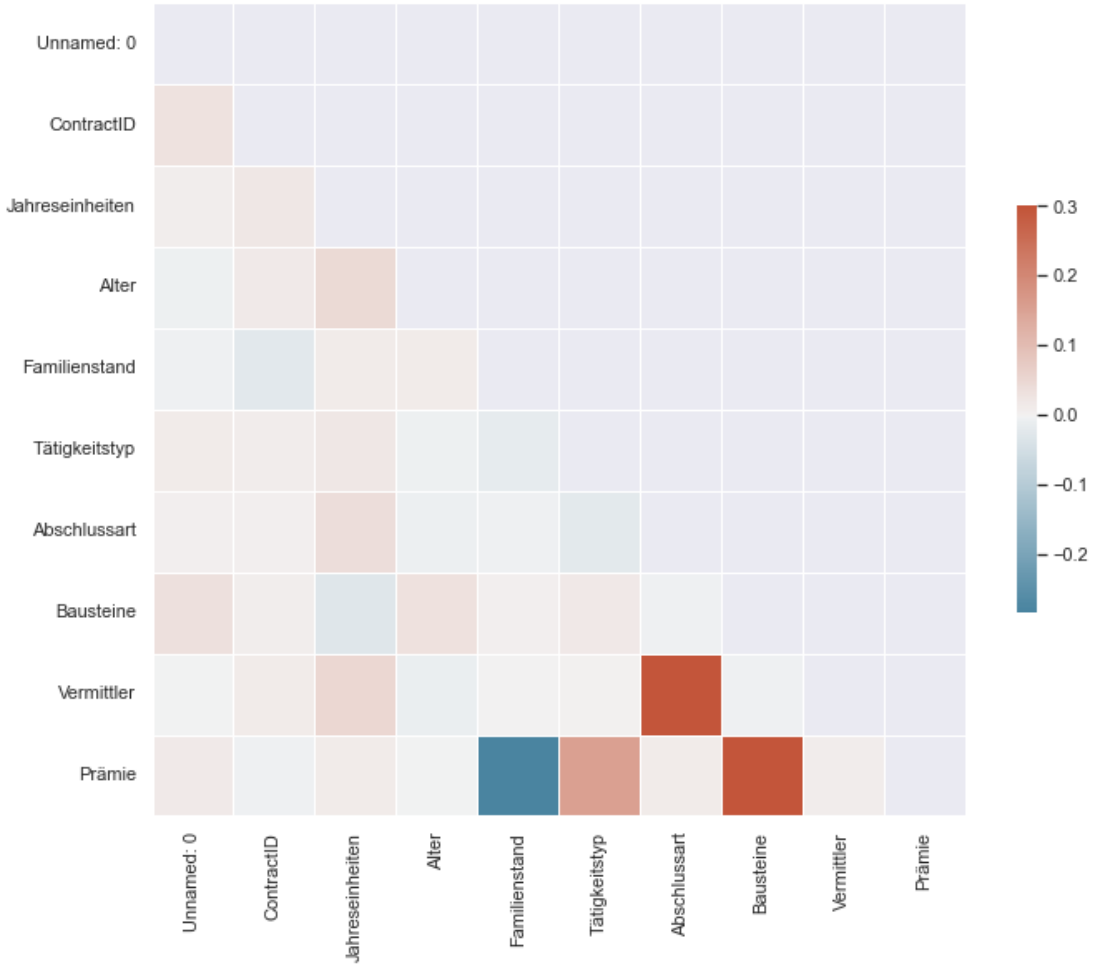

0
5000
10000
15000
20000
25000
Reserve


0.0
0.2
0.4
0.6
0.8
1.0
Problemfall


0
5000
10000
15000
20000
25000
Endreserve

In [14]:
```
corr = df_contracts.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

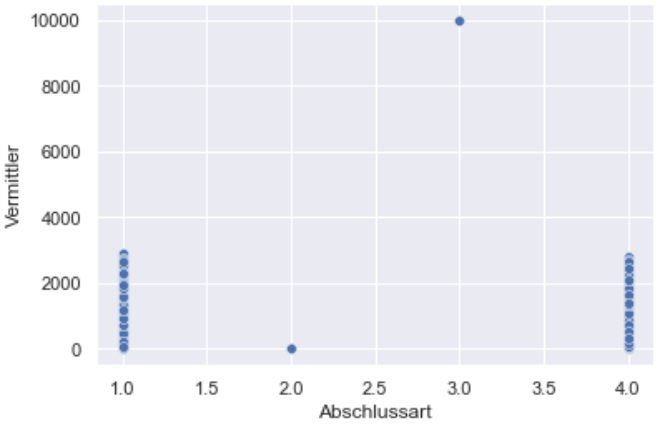
Out[14]:
```
<AxesSubplot:>
```



Das erste, was uns hier auffällt ist, dass die `Abschlussart` mit dem Merkmal `Vermittler` korreliert scheint, außerdem ist `Bausteine` mit der `Prämie` korreliert (was Sinn ergibt). Dies wollen wir anhand von spezifischen Plots weiter prüfen.

In [15]:
```
sns.scatterplot(x=df_contracts['Abschlussart'],y=df_contracts['Vermittler'])
```

Out[15]:
```
<AxesSubplot:xlabel='Abschlussart', ylabel='Vermittler'>
```



Möglicherweise nicht informativ, sondern ein Artefakt aus der Anordnung der IDs. Wir äußern die Vermutung, dass der sehr große Wert hier entweder eine Fehleingabe oder eine Miss-Kodierung darstellt.

In [16]:
```
df_contracts[df_contracts['Vermittler']>=9000]
```

Out[16]:

	Unnamed: 0	ContractID	Jahreseinheiten	Alter	Familienstand	Tätigkeitstyp	Abschlussart	Bausteine	Vertragsdatum	Vermittler	
2	2	3278	0.246399	28	1	2	3	5	2021-02-27	9999	25(
3	3	36048	0.219732	82	1	6	3	8	2021-03-02	9999	40
4	4	32098	0.131204	42	3	3	3	4	2021-04-28	9999	10
5	5	29256	0.131199	29	3	1	3	13	2021-02-17	9999	13
12	12	55302	0.266489	40	1	5	3	4	2021-02-28	9999	46
...	...	...	...	...	...	...	...	...	...	...	
2381	2381	2482	0.271512	34	2	2	3	9	2021-03-10	9999	46(
2384	2384	54890	0.163773	77	3	2	3	11	2021-03-22	9999	12
2385	2385	19848	0.216440	18	1	2	3	4	2021-02-08	9999	32
2389	2389	65774	0.237681	22	3	2	3	4	2021-02-02	9999	8(
2390	2390	80147	0.174911	31	1	2	3	10	2021-04-20	9999	33

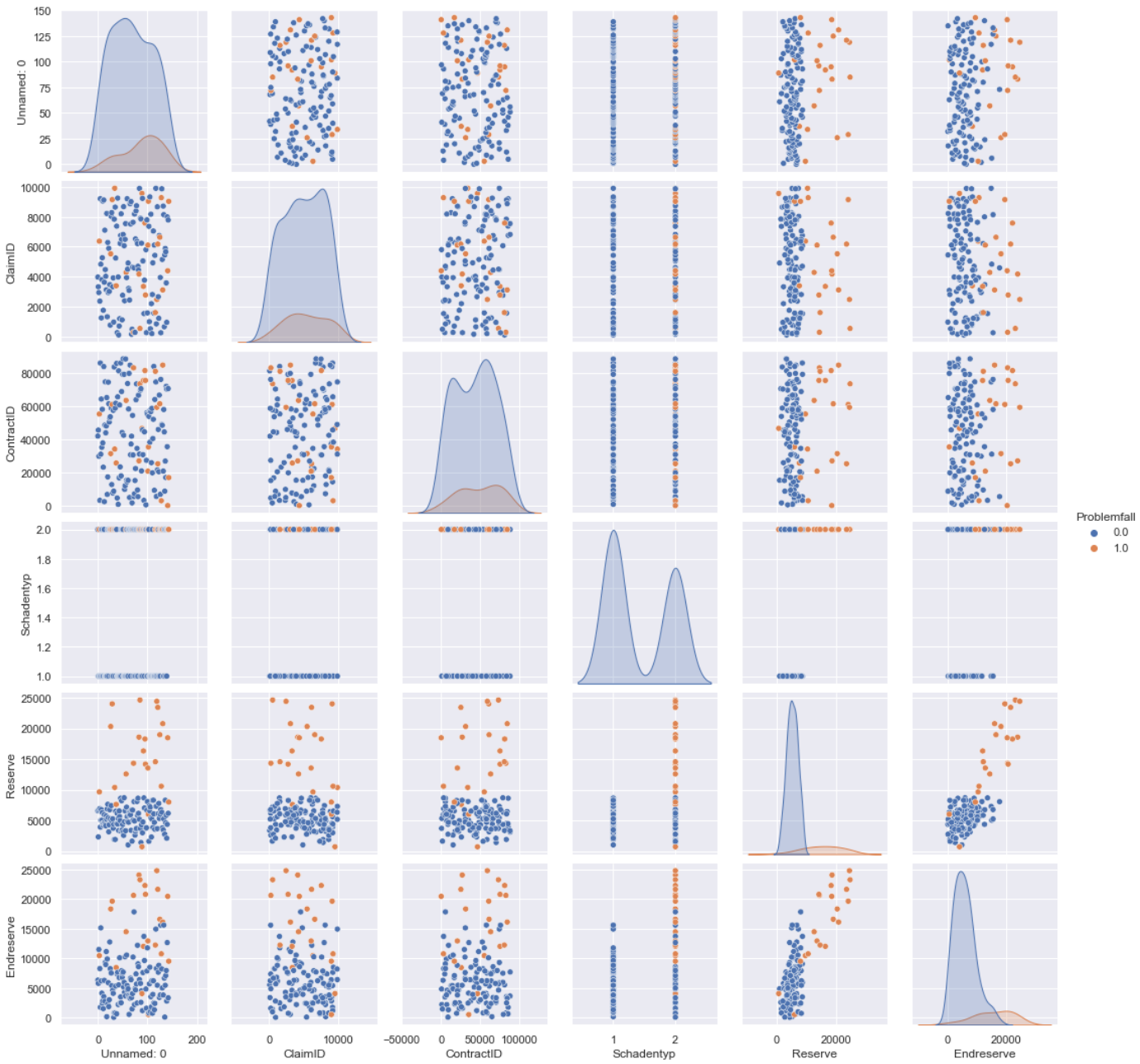
1250 rows × 11 columns

Wir erkennen, dass bei 1250 Verträgen der Vermittler die ID `9999` trägt - womöglich eine (schlechte!) Art und Weise "unbekannt" zu schlüsseln.

Als nächstes wollen wir uns den als "Problemfällen" bekannten Verträgen nähern. Dazu beginnen wir mit einem eingefärbten Pairplot.

In [17]:
```
sns.pairplot(df_claims, hue="Problemfall")
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x1b894b6f7c0>
```

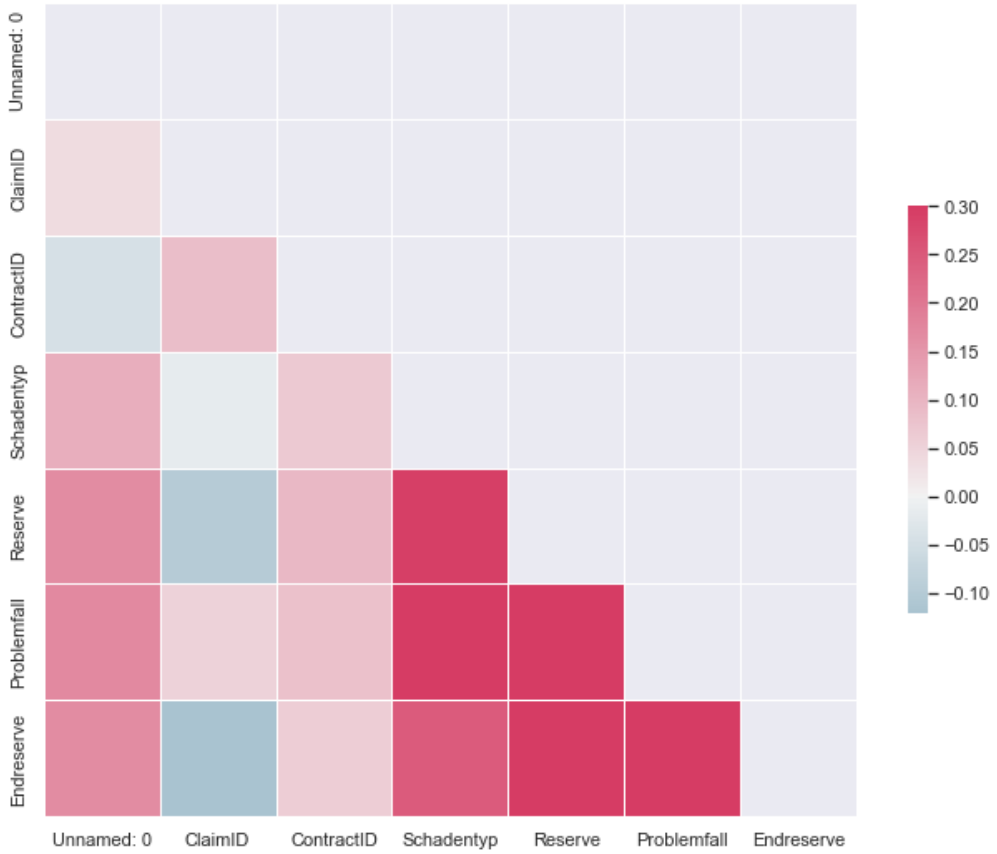


Es ist zu erkennen, dass problematische Fälle den Schadentyp `2` aufweisen und außerdem eine deutliche Differenzierung hinsichtlich der `Reserven` aufweisen.

Als nächstes wird überprüft, inwiefern die über gemeldete Schäden (also in `df_claims`) vorhandenen Informationen korreliert sind.

In [18]:
```
corr = df_claims.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 2, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[18]:
```
<AxesSubplot:>
```



Zusammenfassung:

Der Bestand sieht verhältnismäßig normal und frei von großen Ausreißern aus. Es lässt sich zudem vermuten, dass der Datensatz synthetisch ist, weil insbesondere die Verteilung der Abschlusszeitpunkte relativ wenig Volatilität enthält und bei einem neuen Produkt zu erwarten wäre, dass dieses zunächst gar keine (oder wenige) Abschlüsse enthält, die dann mit der Zeit steigen.

Als interessant gelten hier die Schadenhöhen und Reservierungen sowie der zeitliche Verlauf der als `problematisch` markierten Fälle. Ein Problem an der Datenbasis ist, dass *a priori* noch keine Trennbarkeit zwischen Fällen, die in der Zukunft *stochastisch* einen (normalen) Schaden erzeugen werden und solchen, die *sicher* einen Schaden erzeugen, feststellbar ist. Es ist also anzunehmen, dass ein gewisser Teil der Überlegungen sich auf '*IBNR*' *(incurred but [yet] not reported)*-Schäden beziehen wird.

Bezüglich des Merkmals `Probemfall` lässt sich sagen, dass dieses stets im `Schadentyp` `2` auftritt. Selbst dort ist es aber nur eine kleine Population.

Aufgabe 2

Um eine fundierte Schätzung der weiteren Schäden abgeben zu können, scheint es unerlässlich, beide Tabellen miteinander zu verknüpfen (joinen), um relevante Korrelationen innerhalb der Vertragsmerkmale zu identifizieren.

In [19]:
```
df_joined = df_contracts.join(df_claims, lsuffix='ContractID', rsuffix='ContractID')
df_joined
```

Out[19]:

	Unnamed: 0ContractID	ContractIDContractID	Jahreseinheiten	Alter	Familienstand	Tätigkeitstyp	Abschlussart	Bausteine	Vertragsdatum	Ve
0	0	83810	0.174908	30	2	2	2	4	2021-02-07	
1	1	14592	0.290143	48	1	1	1	6	2021-02-17	
2	2	3278	0.246399	28	1	2	3	5	2021-02-27	
3	3	36048	0.219732	82	1	6	3	8	2021-03-02	
4	4	32098	0.131204	42	3	3	3	4	2021-04-28	
...	...	...	...	...	...	...	...	...	...	
2386	2386	54127	0.174234	40	1	2	4	10	2021-04-25	
2387	2387	26724	0.220215	35	1	4	1	13	2021-02-10	
2388	2388	53788	0.241117	34	1	1	1	10	2021-04-18	
2389	2389	65774	0.237681	22	3	2	3	4	2021-02-02	
2390	2390	80147	0.174911	31	1	2	3	10	2021-04-20	

2391 rows × 19 columns

Wir betrachten von den verknüpften Daten ebenso einen Pairplot.

In [20]:
```
df_joined = df_joined.dropna()
df_joined['Problemfall'] = df_joined['Problemfall'].astype('int')
df_joined_claims = df_joined[(df_joined['Problemfall'] == 1.0) | (df_joined['Problemfall'] == 0.0)]
df_joinded_claims_for_pp = df_joined_claims.loc[:,['Alter','Jahreseinheiten','Familienstand',
                                                   'Tätigkeitstyp','Abschlussart','Bausteine',
                                                   'Vermittler','Prämie','Schadentyp',
                                                   'Reserve','Endreserve','Problemfall']]
sns.pairplot(df_joinded_claims_for_pp,hue='Problemfall')
```

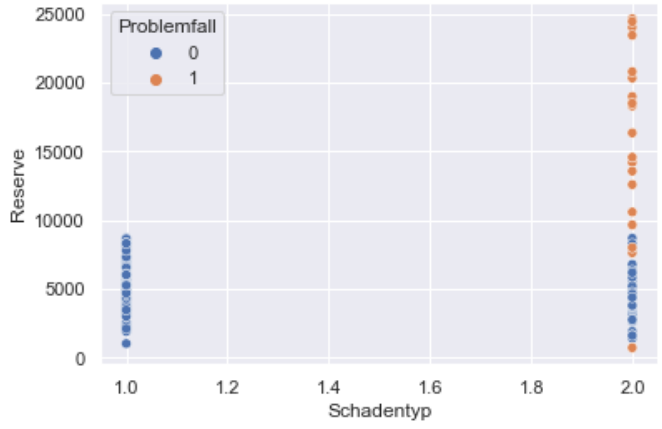
```
C:\Users\FABIAN~1\AppData\Local\Temp/ipykernel_29252/926607906.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_joined['Problemfall'] = df_joined['Problemfall'].astype('int')
```

Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x1b88870c3d0>
```

Als nächstes können wir abermals Korrelations- und Scattermatrizen betrachten. Hierbei ist eine Begrenzung auf Schadenfälle zweckmäßig.

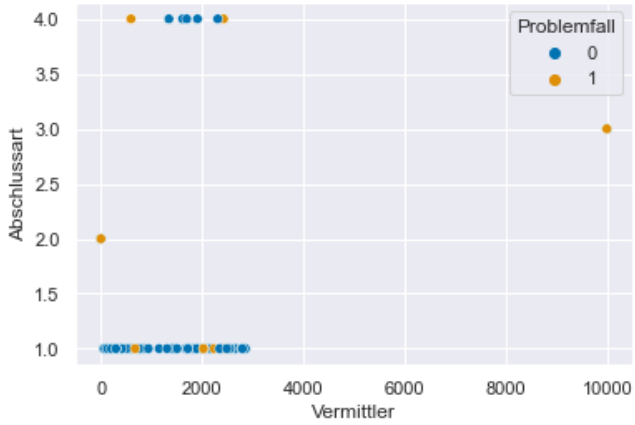
In [21]:
```
sns.scatterplot(x=df_joined_claims['Schadentyp'],y=df_joined_claims['Reserve'],hue=df_joined_claims['Problemfall'])
```

Out[21]:
```
<AxesSubplot:xlabel='Schadentyp', ylabel='Reserve'>
```



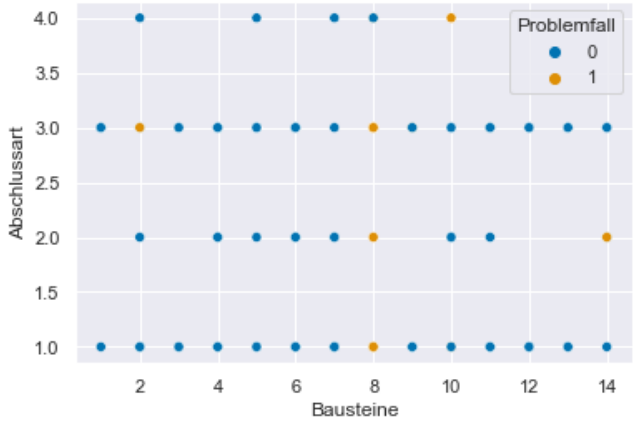
In [22]:
```
cmap = sns.diverging_palette(230, 2, as_cmap=True)
sns.scatterplot(x=df_joined_claims['Vermittler'],y=df_joined_claims['Abschlussart'],hue=df_joined_claims['Problemfal
```

Out[22]:
```
<AxesSubplot:xlabel='Vermittler', ylabel='Abschlussart'>
```



In [23]:
```
cmap = sns.diverging_palette(230, 2, as_cmap=True)
sns.scatterplot(x=df_joined_claims['Bausteine'],y=df_joined_claims['Abschlussart'],hue=df_joined_claims['Problemfall
```

Out[23]:
```
<AxesSubplot:xlabel='Bausteine', ylabel='Abschlussart'>
```



Hier zeigt sich ein interessantes Muster: Zwar gibt es nicht (wie man sich womöglich wünschen könnte) nur eine einzige Bausteinkombination mit Problemschäden, doch sind zwei Dinge auffällig:

- Erstens gibt es für jede Abschlussart Problemfälle mit dem Baustein `8`.
- Zweitens ist die Anzahl (bzw. Pakete) mit Problemfällen immer dadurch gekennzeichnet, dass Baustein `8` enthalten ist: `8 = 8+0`, `10 = 8+2`, `14 = 8+2+4`.

Dies lässt vermuten, dass die angestrebte Deckung (von den vertretenden Kanzleien instruiert) in Baustein `8` realisiert ist und Fälle mit Bausteinen `10` oder `14` bezüglich des Sachverhalts Fehl- bzw. Überabschlüsse darstellen:

In [25]:
```
filtered_data = df_joined_claims[df_joined_claims['Problemfall'] == 1.0]
value_counts = (filtered_data['Bausteine'].sort_values()).value_counts()
value_counts
```

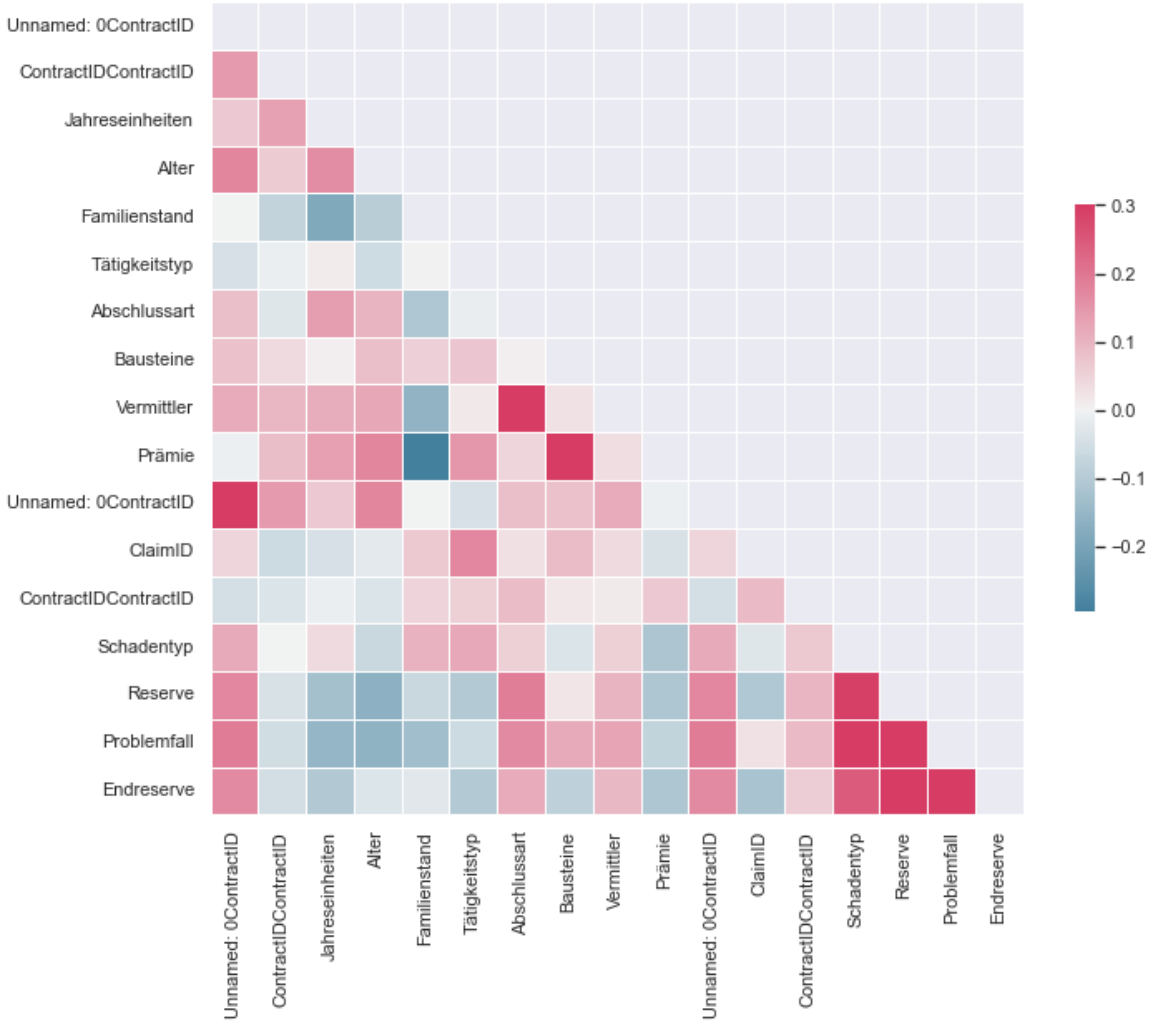
Out[25]:
```
8     16
10     2
2      1
9      1
11     1
14     1
Name: Bausteine, dtype: int64
```

Es könnte sein, dass der Einzelschaden mit Baustein `2` fälschlicherweise angenommen wurde und eigentlich nicht gedeckt wäre. Dies lässt sich ggf. mit der Neuheit des Produktes für die Schadenabteilung erklären.

Im Folgenden sollen weitere Hinweise für die Umfangschätzung gesammelt werden. Hierzu wird eine Korrelationsmatrix der gejointen Datan betrachtet.

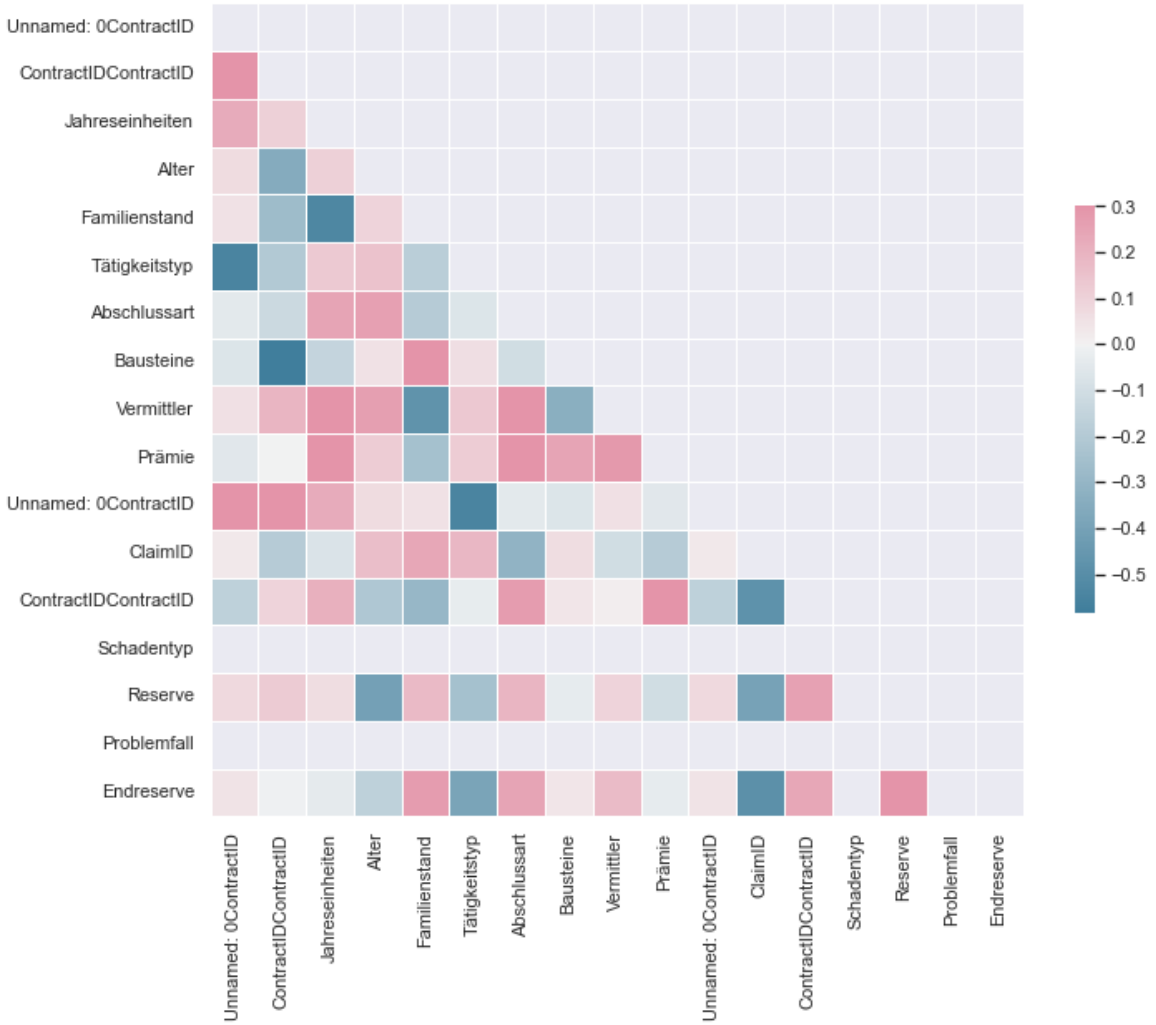
In [26]:
```
corr = df_joined_claims.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 2, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[26]: <AxesSubplot:>



In [27]:
```
corr = df_joined_claims[df_joined_claims['Problemfall'] == 1.0].corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 2, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[27]: <AxesSubplot:>



Sehen wir uns einmal die häufigsten Vermittler unserer Problemfälle an:

In [28]:
```
filtered_data_neg = df_joined_claims[df_joined_claims['Problemfall'] == 1.0]
filtered_data_pos = df_joined_claims[df_joined_claims['Problemfall'] == 0.0]
value_counts = (filtered_data_neg['Vermittler'].sort_values()).value_counts().sort_values()
value_counts
```

Out[28]:
```
595      1
676      1
2027     1
2240     1
2424     1
2454     1
2503     1
0        2
9999    13
Name: Vermittler, dtype: int64
```

Es ist zu erkennen, dass die Hälfte der Fälle auf die Vermittler-ID `9999` entfällt - möglicherweise ist damit unsere Annahme in Aufgabe 1, dass hier `unbekannt` codiert wird, falsch und es ist vielmehr ein Code für entweder `online-Abschluss` oder auch `Direktvertrieb`. Beides ergibt Sinn, da Kunden, die eine Police spezifisch zur Meldung des beschriebenen Schadenfalls abschließen, von der sie vertretenden Kanzlei genau instruiert würden. Obwohl dies plausibel erscheint, ist der Anteil nicht groß genug, als dass es sich lohnen würde, sich nur auf diese Fälle zu konzentrieren. Sobald aber eine weitere Schätzung möglich ist, lässt sich die Anzahl zu erwartender Schäden anhand des Verhältnisses weiter plausibilisieren.

In [29]:
```
filtered_data = df_joined_claims[df_joined_claims['Problemfall'] == 1.0]
value_counts = (filtered_data_neg['Bausteine'].sort_values()).value_counts().sort_values()
value_counts
```

Out[29]:
```
2      1
9      1
11     1
14     1
10     2
8     16
Name: Bausteine, dtype: int64
```

Als nächstes wollen wir überprüfen, wann die bisher bekannten Schäden gemeldet worden sind und welchen Abstand zwischen Abschluss und Schadenmeldung es gibt.

In [30]:
```
filtered_data_neg['Schadendatum']
```

Out[30]:
```
3      2021-02-18
26     2021-02-17
29     2021-02-07
37     2021-02-18
57     2021-02-08
72     2021-02-28
83     2021-02-04
85     2021-02-17
89     2021-02-27
92     2021-02-20
95     2021-02-13
96     2021-02-10
101    2021-02-24
102    2021-05-02
116    2021-04-14
119    2021-02-06
121    2021-04-17
125    2021-02-03
128    2021-02-13
131    2021-02-25
141    2021-02-02
143    2021-02-18
Name: Schadendatum, dtype: object
```

In [31]:
```
print("Mittlere Dauer bis Schadenmeldung Problemfälle:",
      np.mean(np.array(list(map(lambda x:datetime.strptime(x, '%Y-%m-%d'),
                                filtered_data_neg['Schadendatum'])))) -
              np.array(list(map(lambda x:datetime.strptime(x, '%Y-%m-%d'),
                                filtered_data_neg['Vertragsdatum'])))))
print("Mittlere Dauer bis Schadenmeldung Normalschäden:",
      np.mean(np.array(list(map(lambda x:datetime.strptime(x, '%Y-%m-%d'),
                                filtered_data_pos['Schadendatum'])))) -
              np.array(list(map(lambda x:datetime.strptime(x, '%Y-%m-%d'),
                                filtered_data_pos['Vertragsdatum'])))))
```

```
Mittlere Dauer bis Schadenmeldung Problemfälle: 6 days, 5:27:16.363636
Mittlere Dauer bis Schadenmeldung Normalschäden: 42 days, 0:47:36.198347
```

Wie erwartet zeigt sich, dass Problemschadenfälle recht schnell gemeldet werden. Die Liste der Fälle zeigt zudem, dass nur wenige Fälle neuer als Februar sind. Somit lässt sich die Hypothese formulieren, dass für März-Juni noch Fälle zu erwarten sind. Dies könnten bspw. unter der Annahme geschehen, dass das Verhältnis von Schadenmeldungen per Zeitintervall ungefähr gleichbleibend ist. (Das ist fraglos nicht ganz richtig, weil die Schadenmeldungen eigentlich als zensiert betrachtet werden müssten - der Unterschied dürfte aber nicht zu groß sein, sodass auf die Schätzung einer Hazard-Funktion hier verzichtet wird.)

Im Folgenden soll nun erst einmal geschaut werden, wie hoch der mittlere Schaden ist:

In [32]:
```
print("Mittlere Schadenhöhe für Problemfälle per Reserve:",filtered_data['Reserve'].mean())
mean_claim_size = filtered_data['Endreserve'].mean()
print("Mittlere Schadenhöhe für Problemfälle per Endreserve:",mean_claim_size)
```

```
Mittlere Schadenhöhe für Problemfälle per Reserve: 15455.310534142625
Mittlere Schadenhöhe für Problemfälle per Endreserve: 15639.303138586041
```

Es fällt auf, dass die mittlere Schadenhöhe nach Abwicklung mit `14606€` etwas niedriger ist als bei der Reservierung. Dies könnte darauf hindeuten, dass einige (wenige?) Fälle per Vergleich beendet wurden. Dies ist aus den Daten aber ohne Schadenakte nicht direkt ersichtlich. Wir können aber ohne weiteres die Annahme treffen, dass die mittlere Schadenhöhe auch für IBNR-Schäden ungefähr angemessen wäre.

Zwischenfazit

Unsere Arbeitshypothese besteht nun darin, dass problematische Verträge mindestens den Baustein `8` enthalten und für März - Juni noch keine Schäden im Datensatz gemeldet sind. (Diese können bzw. werden also in der Schadenbearbeitung "festhängen".)

Betrachten wir nun noch einmal die Verträge, für die noch keine Schäden gemeldet wurden, so können wir für das Verhältnis von Problem- zu Normalschäden im Februar extrapolieren:

In [39]:
```
anzahl_neg_claims_feb = len(filtered_data_neg[filtered_data_neg['Schadendatum'] < '2021-03-01'])
ratio_problems_in_claims = len(filtered_data_neg) / len(df_claims)
print("Anteil Problemfälle in den Schadenfällen:",ratio_problems_in_claims)
ratio_problems_in_feb_claims = anzahl_neg_claims_feb /
(len(filtered_data_pos[filtered_data_pos['Schadendatum'] < '2021-03-01']) + anzahl_neg_claims_feb)
print("Anteil Problemfälle im Februar:",ratio_problems_in_feb_claims)
contracts_in_feb = df_contracts[df_contracts['Vertragsdatum'] < '2021-03-01']
ratio_problems_in_baustein_8 = len(filtered_data_neg[filtered_data_neg['Bausteine'] == 8]) /
len(contracts_in_feb[contracts_in_feb['Bausteine']  == 8])
print("Anteil Problemfälle in Baustein 8:",ratio_problems_in_baustein_8)
```

```
ratio_problems_in_baustein_9 = len(filtered_data_neg[filtered_data_neg['Bausteine'] == 9]) /
len(contracts_in_feb[contracts_in_feb['Bausteine']  == 9])
print("Anteil Problemfälle in Baustein 9:",ratio_problems_in_baustein_9)
ratio_problems_in_baustein_10 = len(filtered_data_neg[filtered_data_neg['Bausteine'] == 10]) /
len(contracts_in_feb[contracts_in_feb['Bausteine']  == 10])
print("Anteil Problemfälle in Baustein 10:",ratio_problems_in_baustein_10)
ratio_problems_in_baustein_11 = len(filtered_data_neg[filtered_data_neg['Bausteine'] == 11]) /
len(contracts_in_feb[contracts_in_feb['Bausteine']  == 11])
print("Anteil Problemfälle in Baustein 11:",ratio_problems_in_baustein_11)
ratio_problems_in_baustein_14 = len(filtered_data_neg[filtered_data_neg['Bausteine'] == 14]) /
len(contracts_in_feb[contracts_in_feb['Bausteine']  == 14])
print("Anteil Problemfälle in Baustein 14:",ratio_problems_in_baustein_14)
```

```
Anteil Problemfälle in den Schadenfällen: 0.1527777777777778
Anteil Problemfälle im Februar: 0.76
Anteil Problemfälle in Baustein 8: 0.25396825396825395
Anteil Problemfälle in Baustein 9: 0.020833333333333332
Anteil Problemfälle in Baustein 10: 0.044444444444444446
Anteil Problemfälle in Baustein 11: 0.02
Anteil Problemfälle in Baustein 14: 0.0196078431372549
```

Man erkennt, dass der Großteil der Problemschäden in Baustein 8 gemeldet ist. Trotzdem soll für unsere Schätzung auch der kleinere Anteil weiterer Bausteine angewendet werden. Wir verzichten dabei auf die Anteile mit Baustein 2, da wir begründet davon ausgehen, dass hier ein unversicherter Schaden reguliert wurde. Als nächstes bestimmen wir also die Anzahl der zu erwartenden Schäden aus dem Anteil der jeweils vorhandenen Verträge von März bis Juni.

(Anmerkung: Die Schreibweise ohne Schleifen dient der Aufbereitung für ein eventuelles Audit: So ist sichergestellt, dass nachvollziehbar bleibt, was genau hier gerechnet wurde.)

In [41]:
```
contracts_after_feb = df_contracts[df_contracts['Vertragsdatum'] > '2021-02-28']
contracts_baustein_8_after_feb = len(contracts_after_feb[contracts_after_feb['Bausteine'] == 8])
contracts_baustein_9_after_feb = len(contracts_after_feb[contracts_after_feb['Bausteine'] == 9])
contracts_baustein_10_after_feb = len(contracts_after_feb[contracts_after_feb['Bausteine'] == 10])
contracts_baustein_11_after_feb = len(contracts_after_feb[contracts_after_feb['Bausteine'] == 11])
contracts_baustein_14_after_feb = len(contracts_after_feb[contracts_after_feb['Bausteine'] == 14])

claims_estimate = contracts_baustein_8_after_feb * ratio_problems_in_baustein_8
+ contracts_baustein_9_after_feb * ratio_problems_in_baustein_9
+ contracts_baustein_10_after_feb * ratio_problems_in_baustein_10
+ contracts_baustein_11_after_feb * ratio_problems_in_baustein_11
+ contracts_baustein_14_after_feb * ratio_problems_in_baustein_14
print("Es ist die folgende Zahl weiterer Schäden zu erwarten:",claims_estimate)
```

```
Es ist die folgende Zahl weiterer Schäden zu erwarten: 39.20712651727358
```

Zusammenfassung

Es ergibt sich also unter Betrachtung der relativen Häufigkeiten der Problemschäden (insbesondere durch Betrachtung der Tarif-Bausteine 8+) die Erwartung von etwa **40** weiteren Schäden. Hinzu treten die bereits bekannten **22** Schäden.

Zum Vergleich berechnen wir noch einmal, wie sich die Anzahl entwickelte, wenn wir einfach die Anzahl der Schäden im Februar als gleichverteile Basis für die Folgemonate ansetzten:

In [42]:
```
anzahl_neg_claims_feb * 4
```

Out[42]:
```
76
```

Es ist plausibel, dass die Anzahl der geschätzten Schäden niedriger liegt, da ein gewisser Sättigungseffekt dadurch zu erwarten ist, dass vermutlich die beratenden Kanzleien Ihren Mandanten mehr oder weniger umgehend nach Erkenntnis der Bedingungslücke Bescheid gegeben haben und somit die Anzahl der Verträge mit insb. Baustein 8 in den Folgemonaten rückläufig ist.

Aufgabe 3

Aus unserer Sicht ist die Bestimmung der zu erwartenden Anzahl zusätzlicher Schäden zentral für die Beantwortung der Frage. Wir treffen zudem die Zusatzannahme, dass der Schadendurchschnitt ungefähr konstant bleibt, also auch für zukünftige und IBNR-Schäden ~15.000€ beträgt. Dann können wir mit den Ergebnissen aus Aufgabe 2 die Schätzung wie folgt vornehmen:

In [43]:
```
incurred_costs_so_far = filtered_data_neg['Endreserve'].sum()
print("Es sind bereits " +  str(len(filtered_data_neg)) + " Verträge mit insgesamt Kosten von "
      + str(np.round(incurred_costs_so_far,2)) + " bekannt.")
```

```
Es sind bereits 22 Verträge mit insgesamt Kosten von 344064.67 bekannt.
```

In [44]:
```
cost_estimate = claims_estimate * mean_claim_size
print("Es ist die folgende Höhe der zusätzlichen Schadenkosten zu erwarten:",np.round(cost_estimate,2))
print("Gesamthöhe der Schäden:",(cost_estimate + incurred_costs_so_far))
```

```
Es ist die folgende Höhe der zusätzlichen Schadenkosten zu erwarten: 613172.14
Gesamthöhe der Schäden: 957236.8058454297
```

Zusamenfassung

Es ergibt sich also die Erwartung von etwa **40** weiteren Schäden mit Kosten von etwa **600.000€**. Hinzu treten die bereits bekannten **22** Schäden mit Kosten von **335.935€**.

Insgesamt ergibt sich also eine Schädigung von geschätzt **~934.620€**.

Aufgabe 4

Es soll eine Anomalieerkennung erprobt werden, die geeignet wäre, die eingetretenen Schäden zu erkennen und ggf. eine manuelle Überprüfung des Schadengeschehens zu erwirken. Im vorliegenden Fall bietet es sich an, dies als quasi-überwachten Fall einer Klassifikation (anormal: ja/nein) zu betrachten. Daher wird der Datensatz vom Februar in Trainings- und Testdaten zerlegt und mit dem Algorithmus Isolation Forest behandelt.

In [226…
```
y = df_joined_claims['Problemfall']
X = df_joined_claims.drop(["Unnamed: 0ContractID","ContractIDContractID",
                           "Jahreseinheiten","ContractIDContractID",
                           "ClaimID","Problemfall","Vertragsdatum","Schadendatum"],axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [217…
```
X_train
```

Out[217]:

	Alter	Familienstand	Tätigkeitstyp	Abschlussart	Bausteine	Vermittler	Prämie	Schadentyp	Reserve	Endreserve
112	86	1	2	4	8	2309	307.611441	1.0	8309.336632	6224.454049
43	87	1	2	3	13	9999	369.468026	1.0	6552.174746	6758.149459
119	22	1	1	3	8	9999	194.081211	2.0	24419.055156	24790.189702
128	70	3	2	2	14	0	135.402510	2.0	10580.050505	10766.586832
97	60	3	1	1	6	2555	129.403025	2.0	3714.556812	4829.778457
...	...	...	...	...	...	...	...	...	...	...
72	58	2	1	4	10	2424	553.131450	2.0	14331.620427	20619.807877
107	88	2	2	3	13	9999	527.811466	1.0	5391.281057	3530.891686
14	61	1	1	1	4	2626	299.677339	1.0	4668.842055	2758.285897
93	51	2	1	1	3	2782	368.754300	2.0	8675.884060	13702.476652
103	54	3	1	1	11	2796	165.246292	2.0	1607.367530	-803.193667

107 rows × 10 columns

In [218…
```
y_train
```

Out[218]:
```
112    0
43     0
119    1
128    1
97     0
      ..
72     1
107    0
14     0
93     0
103    0
Name: Problemfall, Length: 107, dtype: int32
```

In [296…
```
clf = IsolationForest(random_state=0).fit(X_train)
clf.predict(X_test)
```

Out[296]:
```
array([ 1,  1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1,  1, -1,
        1, -1, -1,  1, -1,  1,  1,  1, -1, -1, -1, -1, -1,  1,  1,  1,
       -1, -1])
```

In [227…
```
# Erzeuge Confusion Matrix für Pseudo-Classifier
y_true = y_test
y_pred = clf.predict(X_test)
# Mappe -1 -> 1, 1 -> 0
conditions = [y_pred == -1, y_pred == 1]
mappings = [1, 0]
y_pred = np.select(conditions, mappings, y_pred)
```

```
confusion_matrix(y_true, y_pred)
```

Out[227]:
```
array([[11, 18],
       [ 3,  4]], dtype=int64)
```

Das ist sicherlich nicht besonders gut, weil viele normale Fälle als Anomalie betrachtet werden. Der Aufwand für die Rückfallebene wäre mit dem Isolation Forest also unökonomisch hoch.

Wir können aber noch weitere Ansätze ausprobieren, zum Beispiel den Local Outlier Factor, der vom dbscan-Algorithmus abgeleitet ist:

In [232…
```
y_train[y_train == 1].count()
```

Out[232]:
```
15
```

In [294…
```
clof = LocalOutlierFactor(n_neighbors=3,novelty=True)
clof.fit(X)
np.sort(clof.negative_outlier_factor_)
```

Out[294]:
```
array([-2.74772347, -2.11915192, -1.91805336, -1.76856139, -1.75488719,
       -1.74021894, -1.73415364, -1.72197938, -1.66067001, -1.64514721,
       -1.64234025, -1.62895532, -1.601907  , -1.57076495, -1.55735023,
       -1.5432606 , -1.4174171 , -1.40154153, -1.35840927, -1.3543551 ,
       -1.34301688, -1.34088409, -1.32210866, -1.29836077, -1.26452385,
       -1.25834087, -1.25625711, -1.24643844, -1.24166774, -1.23216682,
       -1.21889193, -1.20054343, -1.1994413 , -1.19506684, -1.18980988,
       -1.17883099, -1.17817346, -1.1780983 , -1.16682743, -1.16295906,
       -1.1629101 , -1.16187847, -1.15659951, -1.14438185, -1.137044  ,
       -1.13624282, -1.13509457, -1.13084158, -1.12879342, -1.11455298,
       -1.11286356, -1.11243855, -1.11206416, -1.10846225, -1.10415669,
       -1.1031218 , -1.10281182, -1.10231391, -1.09790205, -1.09656307,
       -1.09023144, -1.08671633, -1.08495648, -1.07911059, -1.07718192,
       -1.07519839, -1.06894265, -1.06823043, -1.06680534, -1.06187197,
       -1.06013722, -1.05484676, -1.0542898 , -1.05169005, -1.05102161,
       -1.05045149, -1.04991766, -1.04991766, -1.0480359 , -1.04567304,
       -1.04494537, -1.03777449, -1.03712869, -1.03687299, -1.03583229,
       -1.0331113 , -1.03305158, -1.03149944, -1.03014281, -1.02960594,
       -1.02494167, -1.02429587, -1.0232132 , -1.01477133, -1.01450871,
       -1.01264283, -1.01184745, -1.01080687, -1.00599309, -1.00590425,
       -1.00502264, -1.00262234, -1.00081887, -1.00081887, -1.00036043,
       -1.00036043, -0.99999348, -0.99918716, -0.99550887, -0.99530784,
       -0.9930184 , -0.9923248 , -0.98776983, -0.98269515, -0.98131399,
       -0.97830296, -0.9782602 , -0.97730555, -0.9769758 , -0.97287641,
       -0.97229111, -0.9703754 , -0.97012632, -0.96926773, -0.96613454,
       -0.96613454, -0.95709121, -0.95687374, -0.95274888, -0.95269889,
       -0.94588552, -0.94507253, -0.93701978, -0.92614775, -0.92475101,
       -0.92370682, -0.92290044, -0.92160523, -0.92098512, -0.89998498,
       -0.89001936, -0.88061982, -0.86833411])
```

Da wir wissen, dass es im Trainingsset genau 15 Problemfälle gibt, können wir den Schwellwert gerade so wählen, dass der 15-kleinste Wert verwendet wird:

In [292…
```
clof_preds = clof.predict(X_test)
confusion_matrix(y_true, y_pred)
```

```
D:\Dokumente\Anaconda3\envs\hsh_ml\lib\site-packages\sklearn\base.py:445: UserWarning: X does not have valid featur
e names, but LocalOutlierFactor was fitted with feature names
  warnings.warn(
```

Out[292]:
```
array([[ 0, 29],
       [ 0,  7]], dtype=int64)
```

Leider führt uns die Verwendung von ML-Methoden hier auf die falsche Spur. Die Abhilfe bringt eine klassische Schwellwert-Überlegung in Bezug auf die Reserve, die auch gut zu plausibilisieren ist: Angenommen, es gibt mehrere Schäden mit hoher Reserve (z.B. > 10.000€), wann und in welchem Zeithorizont sollte hier Alarm geschlagen werden?

(*Anmerkung*: Selbstverständlich sind hier der Kreativität nicht unbedingt Grenzen gesetzt und dieser Teil stellt ausdrücklich nur einen Lösungs***vorschlag*** dar. Es sind vielerlei andere Vorgehensweisen möglich.)

Wir erstellen einen Algorithmus auf Basis der Abfolge von Schadenreservierungen: Dazu wird betrachtet, dass bei Problemschäden die Reservierung mindestens im Bereich 10.000€ liegt. Unser Frühwarnsystem springt also an, sobald es mehr als X solche Schäden pro Woche gibt:

In [362…
```
df_joined_claims['Vertragsdatum']= pd.to_datetime(df_joined_claims['Vertragsdatum'])
df_joined_claims['Schadendatum']= pd.to_datetime(df_joined_claims['Schadendatum'])
df_joined_claims['Endreserve'] = pd.to_numeric(df_joined_claims['Endreserve'])
df_sorted_joinded_claims = df_joined_claims.sort_values("Schadendatum").dropna(axis=1)
# betrachte nur Februar:
```

```
df_sorted_claims_feb = df_sorted_joinded_claims[df_sorted_joinded_claims['Schadendatum']<'2021-03-01']
erster_schaden = df_sorted_claims_feb.iloc[0]
letzter_schaden = df_sorted_claims_feb.iloc[-1]
anzahl_wochen = int((letzter_schaden['Schadendatum']-erster_schaden['Schadendatum']).days / 7)+1
print(anzahl_wochen)
thresh = 15000.0
ratio_of_claims_above_threshold = []

claims_in_time_frame_1_pre = df_sorted_claims_feb[df_sorted_claims_feb['Schadendatum'] < '2021-02-08']
claims_in_time_frame_1 = claims_in_time_frame_1_pre[df_sorted_claims_feb['Schadendatum'] > '2021-01-31']
ratio_of_claims_above_threshold_1 = ((claims_in_time_frame_1[claims_in_time_frame_1['Endreserve'] > thresh])['Endres
                                    > thresh).sum() / len(claims_in_time_frame_1)
ratio_of_claims_above_threshold.append(ratio_of_claims_above_threshold_1)

claims_in_time_frame_2_pre = df_sorted_claims_feb[df_sorted_claims_feb['Schadendatum'] < '2021-02-15']
claims_in_time_frame_2 = claims_in_time_frame_2_pre[df_sorted_claims_feb['Schadendatum'] > '2021-02-07']
ratio_of_claims_above_threshold_2 = (claims_in_time_frame_2[claims_in_time_frame_2['Endreserve'] > thresh]['Endreser
                                    > thresh).sum() / len(claims_in_time_frame_2)
ratio_of_claims_above_threshold.append(ratio_of_claims_above_threshold_2)

claims_in_time_frame_3_pre = df_sorted_claims_feb[df_sorted_claims_feb['Schadendatum'] < '2021-02-22']
claims_in_time_frame_3 = claims_in_time_frame_3_pre[df_sorted_claims_feb['Schadendatum'] > '2021-02-14']
ratio_of_claims_above_threshold_3 = (claims_in_time_frame_3[claims_in_time_frame_3['Endreserve'] > thresh]['Endreser
                                    > thresh).sum() / len(claims_in_time_frame_3)
ratio_of_claims_above_threshold.append(ratio_of_claims_above_threshold_3)

claims_in_time_frame_4_pre = df_sorted_claims_feb[df_sorted_claims_feb['Schadendatum'] < '2021-03-01']
claims_in_time_frame_4 = claims_in_time_frame_4_pre[df_sorted_claims_feb['Schadendatum'] > '2021-02-21']
ratio_of_claims_above_threshold_4 = (claims_in_time_frame_4[claims_in_time_frame_4['Endreserve'] > thresh]['Endreser
                                    > thresh).sum() / len(claims_in_time_frame_4)
ratio_of_claims_above_threshold.append(ratio_of_claims_above_threshold_4)

ratio_of_claims_above_threshold
```

```
4
C:\Users\FABIAN~1\AppData\Local\Temp/ipykernel_21876/2316237042.py:15: UserWarning: Boolean Series key will be rein
dexed to match DataFrame index.
  claims_in_time_frame_1 = claims_in_time_frame_1_pre[df_sorted_claims_feb['Schadendatum'] > '2021-01-31']
C:\Users\FABIAN~1\AppData\Local\Temp/ipykernel_21876/2316237042.py:20: UserWarning: Boolean Series key will be rein
dexed to match DataFrame index.
  claims_in_time_frame_2 = claims_in_time_frame_2_pre[df_sorted_claims_feb['Schadendatum'] > '2021-02-07']
C:\Users\FABIAN~1\AppData\Local\Temp/ipykernel_21876/2316237042.py:25: UserWarning: Boolean Series key will be rein
dexed to match DataFrame index.
  claims_in_time_frame_3 = claims_in_time_frame_3_pre[df_sorted_claims_feb['Schadendatum'] > '2021-02-14']
```

Out[362]:
```
[0.7142857142857143, 0.3333333333333333, 0.2857142857142857, 0.4]
```

Man kann sicherlich verschiedene Schwellwerte ausprobieren, aber klar ist, dass mit einer konservativen Wahl von 10.000-15.000€ unter Berücksichtigung der durchschnittlichen "normalen" Schadenhöhe eine Ratio von selbst 0.3 bereits verdächtig wäre und somit von Anfang an ein Fokus auf das Problem gelegt wäre.

Als weitere Herausforderung könnte sich sich die Geschwindigkeit des Reportings darstellen - die Latenz (Verzögerung) der Schadenmeldung ist hier ein zentrales Hindernis und es wird empohlen, die Prozesskette so auszugestalten, dass die Erstreserve möglichst früh nach Meldung im System hinterlegt wird.

Ebenso ist vorstellbar, dass man dies weiter ausnutzt, indem man die Dauer von Vertragsabschluss bis zur Schadenmeldung einbezieht, die mit ~6 (Problemschaden) zu ~42 Tage (Normalschaden) auch erheblich auffällig ist.

Wie viel Schaden kann man so verhindern? Angenommen, die Abwicklung der Februar-Schäden ist komplett erfolgt (d.h. es gibt für Februar keine IBNR-Schäden mehr), so ließe sich mit dem Frühwarn-System bereits nach einer Woche eine Auffälligkeit erkennen. Sicherlich braucht die Analyse in dem Falle noch mehr Zeit, sodass ein Produktstopp wahrscheinlich eher erst nach zwei Wochen erfolgen würde.

Nach zwei Wochen waren folgende Schäden eingetreten:

In [367…
```
claims_half_feb = (df_sorted_claims_feb[df_sorted_claims_feb['Schadendatum'] < '2021-02-15'])
prob_claims_half_feb = claims_half_feb[claims_half_feb['Problemfall'] == 1.0]
value_from_prob_claims_half_feb = prob_claims_half_feb['Endreserve'].sum()
print("Schadenwert von Schäden innerhalb der ersten zwei Februarwochen:",value_from_prob_claims_half_feb)
```

```
Schadenwert von Schäden innerhalb der ersten zwei Februarwochen: 173874.29274231195
```

Zusammenfassung

Wenngleich ML-basierte Anomalieerkennung hier nicht zum Erfolg führt, kann ein Schwellwert-basierter Ansatz vorgeschlagen werden, der den Anteil an "Großschäden" je Zeiteinheit (hier: pro Woche) betrachtet. Beispielsweise legen die Beobachtungen nahe, ab einem Verhältnis von 0.33 eine spezifische Untersuchung der eingegangenen Schäden zu betreiben.

Es wären unter diesen Annahmen nur etwa **175.000€** Schaden entstanden, allerdings ist zu erwähnen, dass dies stark davon abhängt, wie schnell die tatsächliche Schadenbearbeitung vonstatten geht - es steht zu befürchten, dass im beschriebenen Fall gerade ein verzögertes Reporting vorliegt, gerade *weil* es zu massenhaften Schadenmeldungen innerhalb kurzer Zeit kommt. Dies sieht man unter anderem daran, dass die Zeit von Vertragsbeginn bis zur Schadenmeldung sehr kurz ist.

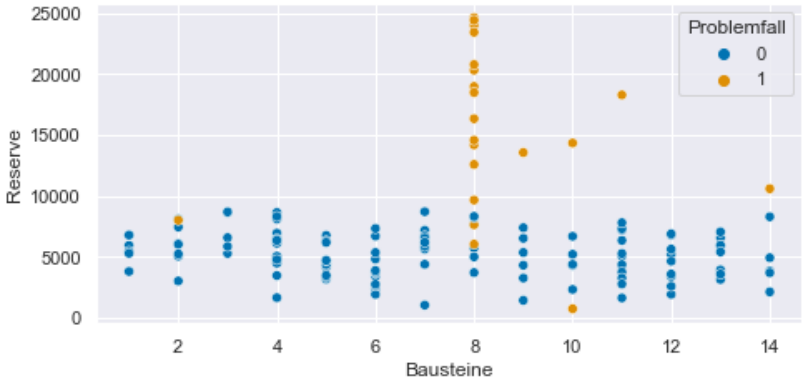
Aufgabe 5

Sammeln des benötigten Materials:

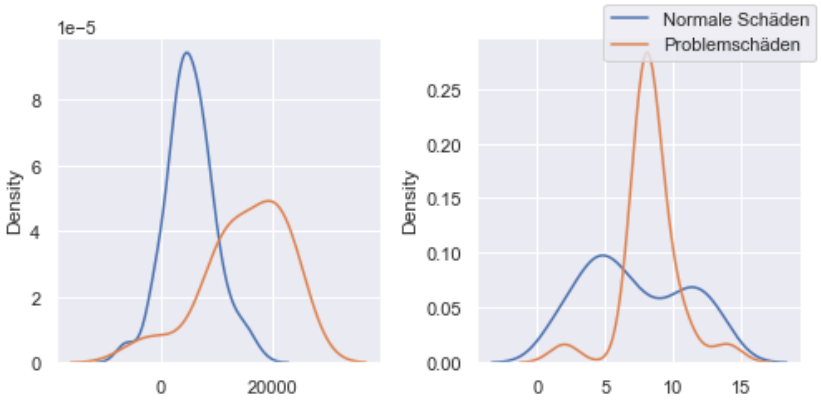
Vor der Erstellung der eigentlichen Reports werden hier noch einmal die benötigten Materialien aus der Analyse zusammengestellt.

In [148…
```
sns.scatterplot(x=df_joined_claims['Bausteine'],
                y=df_joined_claims['Reserve'],
                hue=df_joined_claims['Problemfall'],
                palette='colorblind')
```

Out[148]:
```
<AxesSubplot:xlabel='Bausteine', ylabel='Reserve'>
```



In [198…
```
plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True
fig, axs = plt.subplots(ncols=2)
problemfaelle= df_joined_claims[df_joined_claims['Problemfall'] == 1.0]
normalfaelle = df_joined_claims[df_joined_claims['Problemfall'] == 0.0]
sns.kdeplot(x=normalfaelle['Endreserve'].values, ax=axs[0], label="Normale Schäden")
sns.kdeplot(x=problemfaelle['Endreserve'].values, ax=axs[0], label = "Problemschäden")
sns.kdeplot(x=normalfaelle["Bausteine"].values, ax=axs[1], label="Normale Schäden")
sns.kdeplot(x=problemfaelle["Bausteine"].values, ax=axs[1], label = "Problemschäden")
fig.legend(['Normale Schäden','Problemschäden'])
plt.show()
```



In [203…
```
# Berechnung Sicherheitsaufschlag:
934620*1.15
```

Out[203]:
```
1074813.0
```

Management Summary

Situationsbeschreibung

Nach Einführung des neuen RS-Produkts fiel der Schadenabteilung auf, dass binnen kurzer Zeit eine erhebliche Anzahl Schäden gemeldet wurde. Diese ließen sich auf ein **Schlupfloch in den Versicherungsbedingungen zurückführen**, die einen bestimmten Sachverhaltauschluss bzgl. streitiger Lebensversicherungsverzinsungsfällen nicht enthielten. Nach einiger Recherche wurde klar, dass Kanzleien, die in Verzinsungsfällen von Lebensversicherungen mandatiert sind, ihren Mandanten zum Abschluss des Produkts

geraten haben; es war daher davon auszugehen, dass zu den bereits gemeldeten noch weitere Schäden hinzukommen werden. Zum Zeitpunkt der temporären Einstellung der Verkäufe waren **rund 1.500 Policen ausgestellt**.

Einordnung und Maßnahmen

- Nach unseren Erkenntnissen lässt sich bestätigen, dass zu den bisher *gemeldeten 22 Schäden* weitere IBNR-Meldungen hinzutreten werden. Auf Basis der verfügbaren Erkenntnisse entfallen diese auf Policen mit den Ausgestaltungsmerkmalen der Bausteine `8+`, von denen es ungefähr **40** geben dürfte. Zu den bisher aufgelaufenen Kosten von **335.935€** kommen bei diesen ~40 Schäden weitere **600.000€** auf Basis der durchschnittlichen Schadenhöhe von **15.269€** pro Police hinzu.
- Vorläufige Tests mit Methoden der Anomalieerkennung legen nahe, dass Schwellwert-basierte Ansätze geeignet sind, im automatischen Monitoring datengetriebene Plausibilisierungen anzustoßen, die regelbasiert eine manuelle Überprüfung des laufenden Schadengeschehens erlauben. Die Empfehlung dazu lautet, dass im (automatischen) wöchentlichen Monitoring dieses Produkts bei Überschreitung der Schadenhöhe von 10.000€ in mehr als einem Drittel der Fälle eine spezifische Untersuchung eingeleitet werden sollte. Eine grobe Abschätzung legt nahe, dass im vorliegenden Fall der Sachverhalt bereits mitte Februar hätte erkannt werden können, **was die eingetretenen Schäden auf ~175.000€ beschränkt hätte**.

Bericht über die Erkenntnisse

Nach Einführung des neuen RS-Produkts fiel der Schadenabteilung auf, dass binnen kurzer Zeit eine erhebliche Anzahl Schäden gemeldet wurde. Diese ließen sich auf ein Schlupfloch in den Versicherungsbedingungen zurückführen, die einen bestimmten Sachverhaltsauschluss bzgl. streitiger Lebensversicherungsverzinsungsfällen nicht enthielten. Nach einiger Recherche wurde klar, dass Kanzleien, die in Verzinsungsfällen von Lebensversicherungen mandatiert sind, ihren Mandanten zum Abschluss des Produkts geraten habe; es war daher davon auszugehen, dass zu den bereits gemeldeten noch weitere Schäden hinzukommen werden. Zum Zeitpunkt der temporären Einstellung der Verkäufe waren rund 1.500 Policen ausgestellt. Ausschnitt aus den Schadenakten siehe **Tabelle 1**.

In [164…
```
filtered_data_neg.drop(["Unnamed: 0ContractID",
                        "ContractIDContractID",
                        "Jahreseinheiten",
                        "ContractIDContractID"],
                       axis=1).head(10)
```

Out[164]:

	Alter	Familienstand	Tätigkeitstyp	Abschlussart	Bausteine	Vertragsdatum	Vermittler	Prämie	ClaimID	Schadentyp	Reserv
3	82	1	6	3	8	2021-03-02	9999	402.846406	6375.0	2.0	9648.72280
26	20	1	5	3	8	2021-02-25	9999	184.566864	5530.0	2.0	20317.17860
29	29	3	2	2	8	2021-02-03	0	80.471896	9156.0	2.0	24005.22598
37	40	1	2	3	8	2021-02-05	9999	333.625613	3394.0	2.0	7621.93626
57	22	1	1	1	8	2021-02-07	2240	322.361913	4285.0	2.0	12574.44990
72	58	2	1	4	10	2021-02-16	2424	553.131450	280.0	2.0	14331.62042
83	56	2	1	3	8	2021-02-02	9999	184.377150	4177.0	2.0	18582.15101
85	27	1	2	1	8	2021-02-12	676	194.081211	539.0	2.0	24624.19181
89	37	1	2	1	10	2021-02-23	2454	446.488100	9568.0	2.0	718.62293
92	28	1	1	4	8	2021-02-13	595	250.846326	3350.0	2.0	16330.88932

Tab. 1: Übersicht der ersten 10 exemplarischen Problemschäden.

Es fallen folgende Eigenschaften der fraglichen Schäden auf (siehe **Abbildung 1**):

- Diese bestehen zum Großteil aus Verträgen der Bausteinklasse `8` (oder höher).
- Die Reserve liegt mit ~15.000€ deutlich höher als der Durchschnittsschaden.

In [198…
```
plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True
fig, axs = plt.subplots(ncols=2)
problemfaelle= df_joined_claims[df_joined_claims['Problemfall'] == 1.0]
normalfaelle = df_joined_claims[df_joined_claims['Problemfall'] == 0.0]
sns.kdeplot(x=normalfaelle['Endreserve'].values, ax=axs[0], label="Normale Schäden")
sns.kdeplot(x=problemfaelle['Endreserve'].values, ax=axs[0], label = "Problemschäden")
sns.kdeplot(x=normalfaelle["Bausteine"].values, ax=axs[1], label="Normale Schäden")
sns.kdeplot(x=problemfaelle["Bausteine"].values, ax=axs[1], label = "Problemschäden")
fig.legend(['Normale Schäden','Problemschäden'])
plt.show()
```

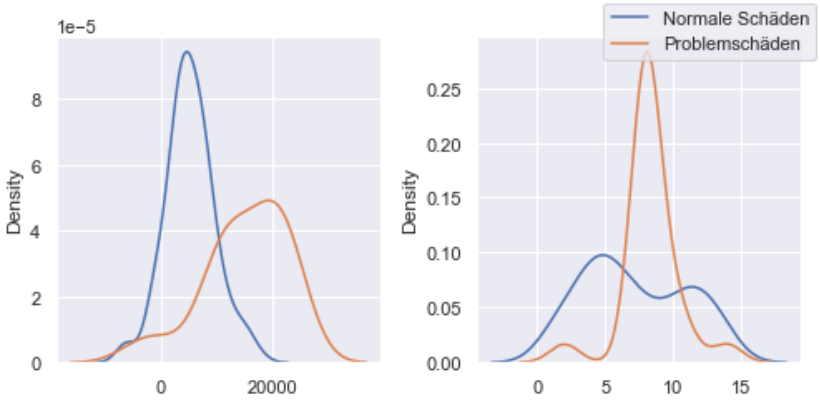


Abb. 1: a) Häufigkeitsverteilung der Reservehöhen. b) Häufigkeitsverteilung der Produktbausteine. Farbcodierung: Blau -> Normalschäden. Orange -> Problemschäden.

Aus der Betrachtung des zeitlichen Verlaufs (siehe Tab. 2) wird deutlich, dass bisher gemeldete Schäden überwiegend aus dem Februar stammen - es ist damit davon auszugehen, dass im Zeitraum März-Juni weitere Schäden hinzutreten werden.

Die folgenden Überlegungen liegen der Anzahl- und Schadenaufwandsschätzung zugrunde:

- Die weiteren Schadenfälle werden weiterhin überwiegend aus Policen mit den Bausteinen 8, 10, 11 und 14 stammen - die Häufigkeitsverteilungen sind hierbei als unverändert angenommen.
- Die weiteren Schadenfälle werden eine gleiche oder zumindest ähnliche Größenordnung Schadenhöhe aufweisen - also etwa ~15.000€.

Daraus ergibt sich folgende Abschätzung:

- Es ergibt sich also die Erwartung von etwa 40 weiteren Schäden mit Kosten von etwa 600.000€. Hinzu treten die bereits bekannten 22 Schäden mit Kosten von 335.935€.
- **Insgesamt ergibt sich eine Schädigung von geschätzt etwa ~934.620€**.

Unsicherheitsoffenlegung: Die beschriebenen Zahlen basieren auf den obigen Annahmen. Es ist möglich, dass a) einerseits ein Sättigungseffekt zu Gunsten der Gesellschaft eintritt und so weniger Schäden anfallen oder aber b) wesentliche Aspekte des Schadengeschehens nicht erfasst wurden. Es erfolgt deshalb die Empfehlung, einen Sicherheitsaufschlag von 15% einzukalkulieren und insgesamt **1.074.813€** für die Abwicklung der Problemschäden zu reservieren.

Zukünftige Früherkennung / Monitoring

Es wurde untersucht, welche Möglichkeiten zur Früherkennung bestehen. Hierfür lautet die vorläufige Empfehlung, dass im (automatischen) wöchentlichen Monitoring dieses Produkts bei **Überschreitung der Schadenhöhe von 10.000€ in mehr als einem Drittel der Fälle** eine spezifische Untersuchung eingeleitet werden sollte. Problematisch hierbei stellt sich die Geschwindigkeit des Reportings dar - die Latenz (Verzögerung) der Schadenmeldung ist hier ein zentrales Hindernis und es wird empohlen die Prozesskette so austzugestalten, dass die Erstreserve möglichst früh nach Meldung im System hinterlegt wird.

Die eingetretenen Schäden im vorliegenden Fall hätten so auf **~175.000€** beschränkt werden können.

Aufgabe B

PK Completion

2023-06-02

Aufgabenstellung

Sie sind im Aktuariat der Lebensversicherung “Alte Leben”, einem etablierten Unternehmen am Markt, angestellt. In nahezu allen Produktlinien sind die Verkaufszahlen rückläufig. Sie haben als Unternehmen deshalb den Vorstand Frau Anders von der Konkurrenz abgeworben. Diese bittet Sie, ein neues Produkt für die Risikolebensversicherung zu entwerfen und meint zu ihnen: “Die Konkurrenz biete beispielsweise einen Akademikerbonus von 5 % an. Da müsste doch noch viel mehr gehen oder es müssten sich noch andere beitragsrelevante Merkmale finden lassen. Können wir nicht weitere lukrative Segmente identifizieren, die wir dann mit einem differenzierten Pricing ansprechen können?!”.

Sie sind in der DAV gut vernetzt und sprechen mit ihrem Netzwerk. Herr Dr. Netzer berichtet von seinen Erfahrungen bei der Erstellung eines Ergebnisberichts für den Ausschuss Leben, vgl. https://aktuar.de/unsere-themen/fachgrundsaetze-oeffentlich/2022-09-21-DAV-Ergebnisbericht_Big_Data_Leben_NHANES_final.pdf (https://aktuar.de/unsere-themen/fachgrundsaetze-oeffentlich/2022-09-21-DAV-Ergebnisbericht_Big_Data_Leben_NHANES_final.pdf). Dabei wurde ein US-Datensatz “NHANES” durch die Arbeitsgruppe aufbereitet, der seiner Ansicht nach zur Analyse dieser Fragestellungen prinzipiell geeignet wäre. Die Übertragbarkeit von Erkenntnissen aus anderen Ländern sei auch bei anderen Untersuchungen gängig und möglich, z.B. Pflege- oder Sterbetafeln. Der Datensatz für diese Prüfungsaufgabe ist aus diesem Datensatz abgeleitet.

Sie entschließen sich mögliche Risikofaktoren für ein differenziertes Pricing auf diesem Datensatz zu analysieren und diese Erkenntnisse dann in Deutschland für ihr Pricing auf Basis der DAV2008T 2. Ordnung anzuwenden.

Aufgabe 1

Teil a): Datenaufbereitung und Verständnis

Lesen Sie die Daten ein. Es handelt sich um eine Kohorten-Studie, verwenden Sie daher zu jeder ID nur die neueste Beobachtung. Führen Sie eine explorative Datenanalyse durch. [Beschreibung Daten gek.]

Visualisieren Sie u.a. den Einfluss verschiedenerer Variablen auf die Überlebenszeit. Berücksichtigen Sie dabei auch die Kovariable Beschäftigungsstatus, die als Proxy für den Akademiker-Status dienen kann. Diskutieren Sie die Übertragbarkeit auf den Akademiker-Status. Was würden Sie zusätzlich benötigen, um hier noch sicherer zu sein?

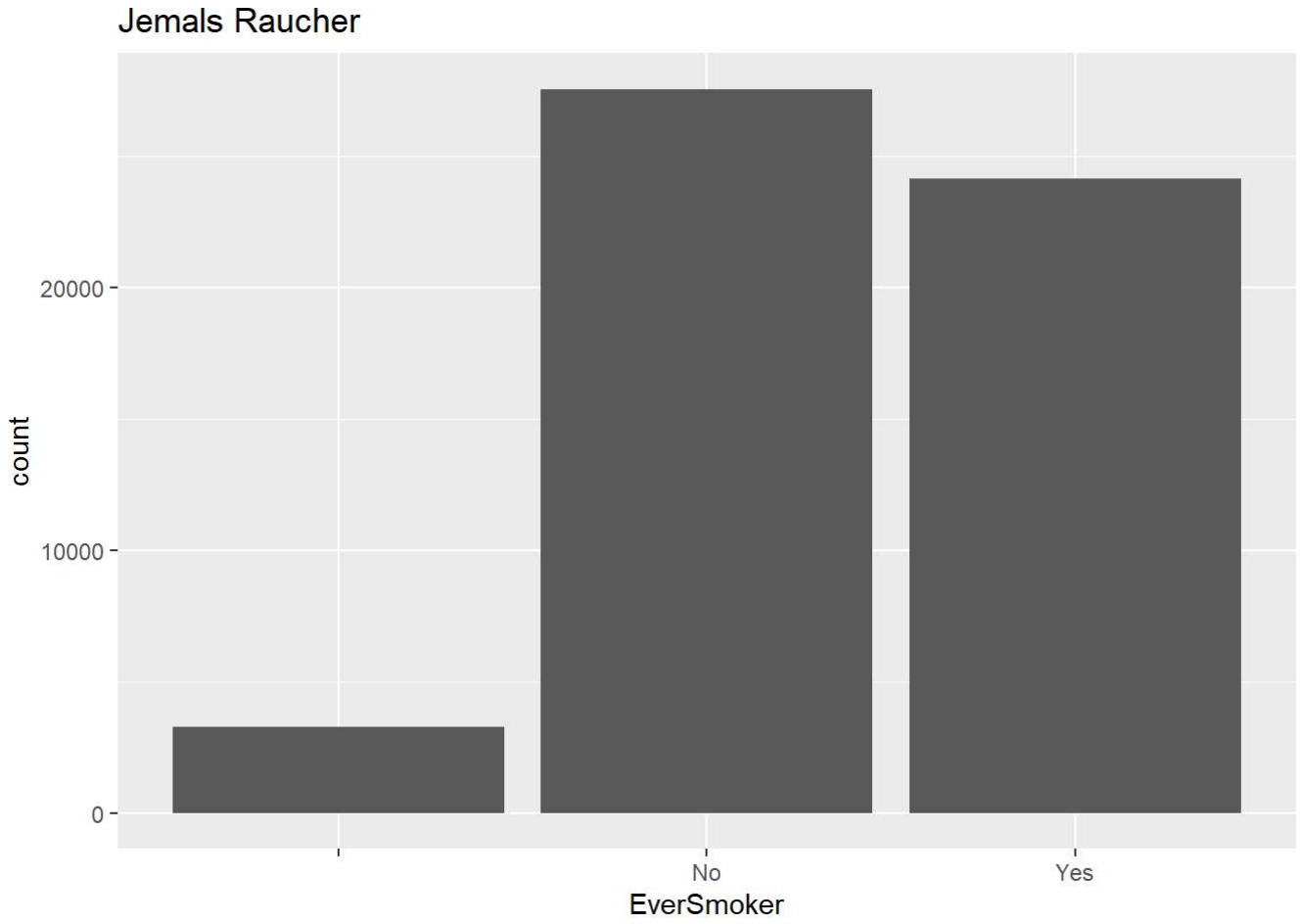
Lösungsvorschlag:

```
data = fread("NHANES_Extract.csv", sep = ";", dec = ",",stringsAsFactors = T,na.strings = "")


doubles = data %>% group_by(SEQN) %>% summarize(anzahl = n()) %>% ungroup() %>% filter(anzahl
> 1) %>% pull(SEQN)


#
`%notin%` = Negate(`%in%`)
data %<>% filter(SEQN %notin% doubles |
                   (SEQN %in% doubles &  Study == "NHANES Continuous"))
#check
print(str_c("Höchstanzahl ist: ",data %>%
        group_by(SEQN)%>%
        summarise(anzahl = n()) %>%
        ungroup() %>% pull(anzahl) %>% max()))
```

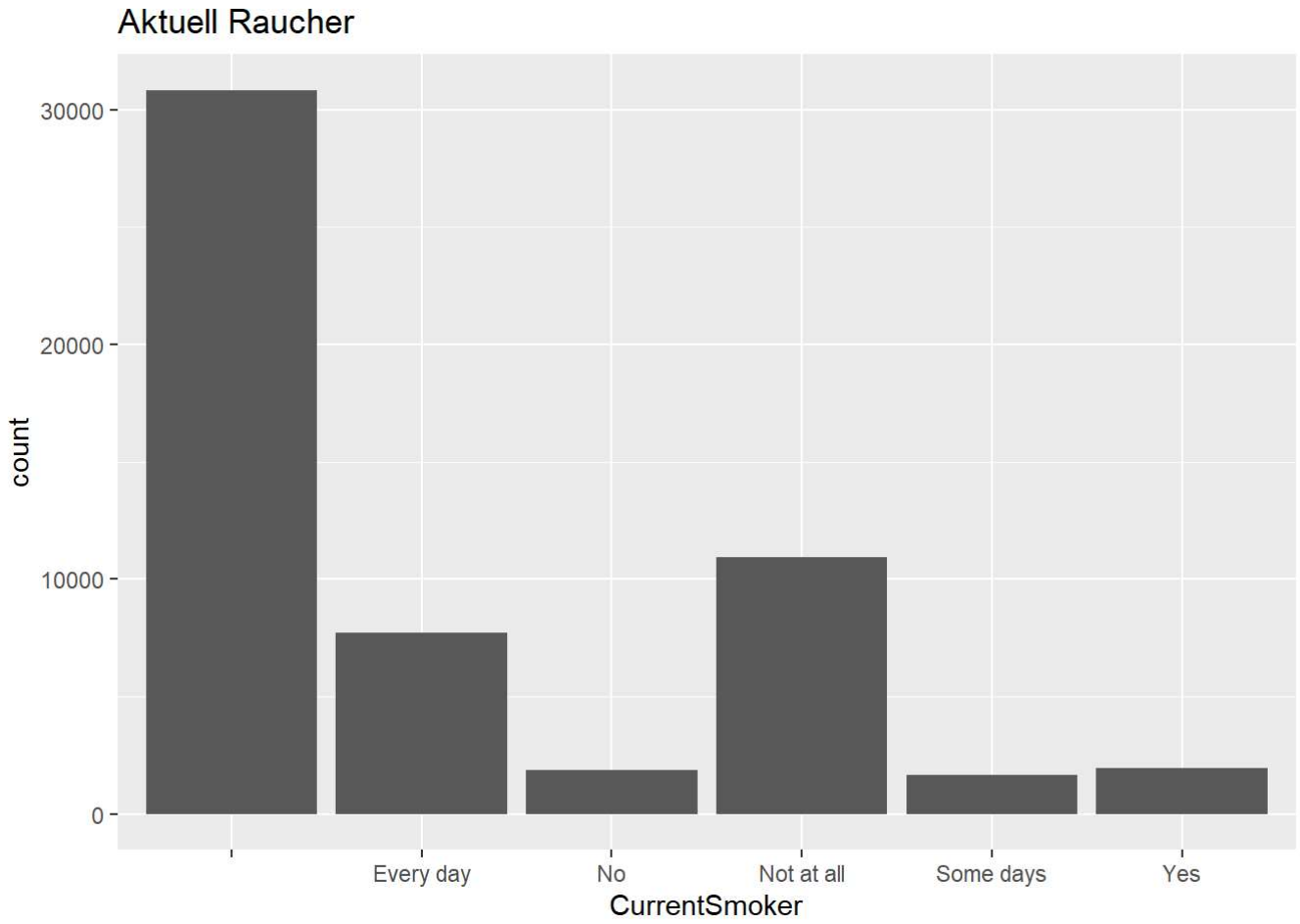
```
## [1] "Höchstanzahl ist: 1"
```

```
rm(doubles)
```

```
ggplot(data = data) +
  geom_bar(mapping = aes(x = EverSmoker)) + labs(title = "Jemals Raucher")
```

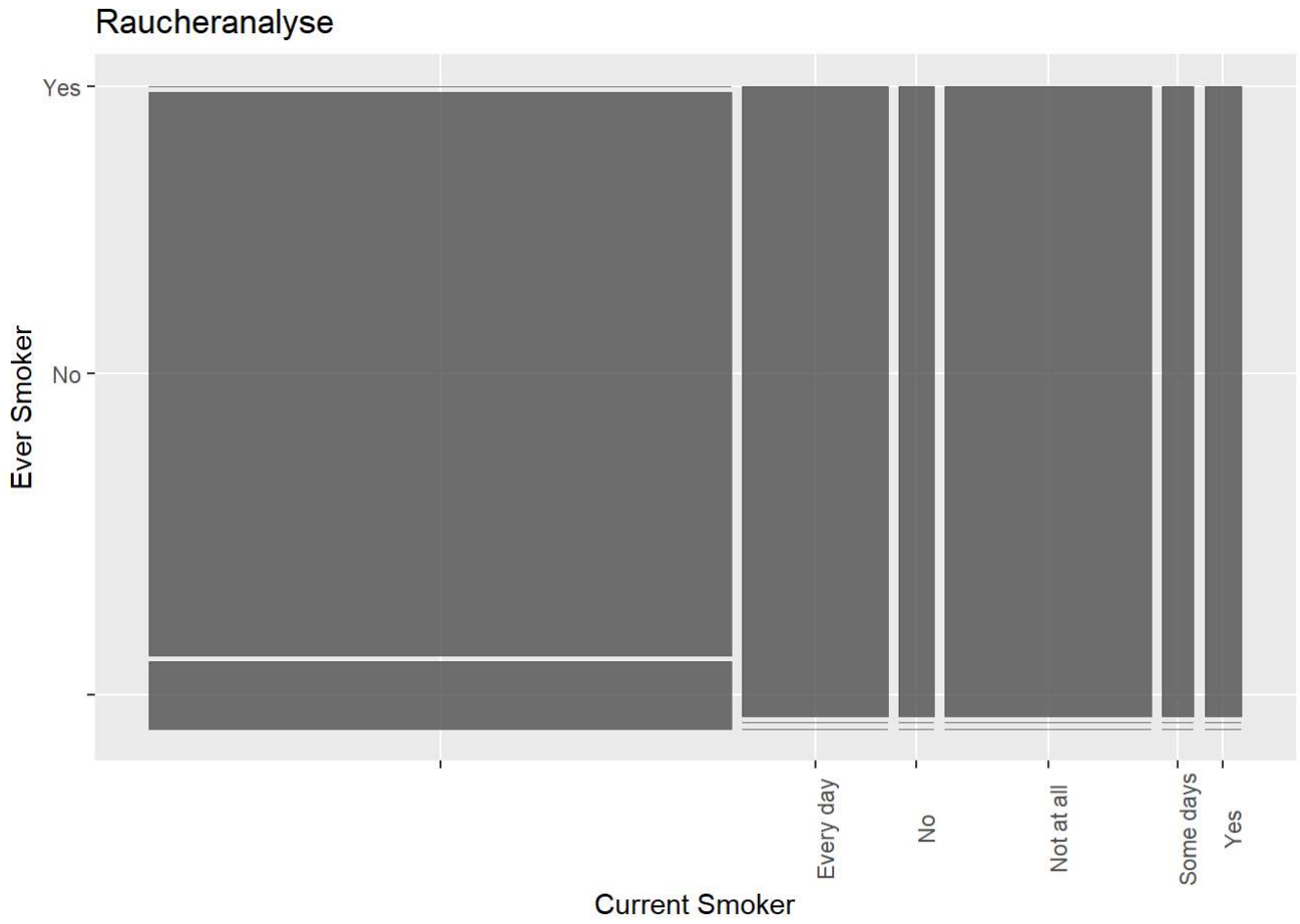


```
ggplot(data = data) +
  geom_bar(mapping = aes(x = CurrentSmoker)) + labs(title = "Aktuell Raucher")
```

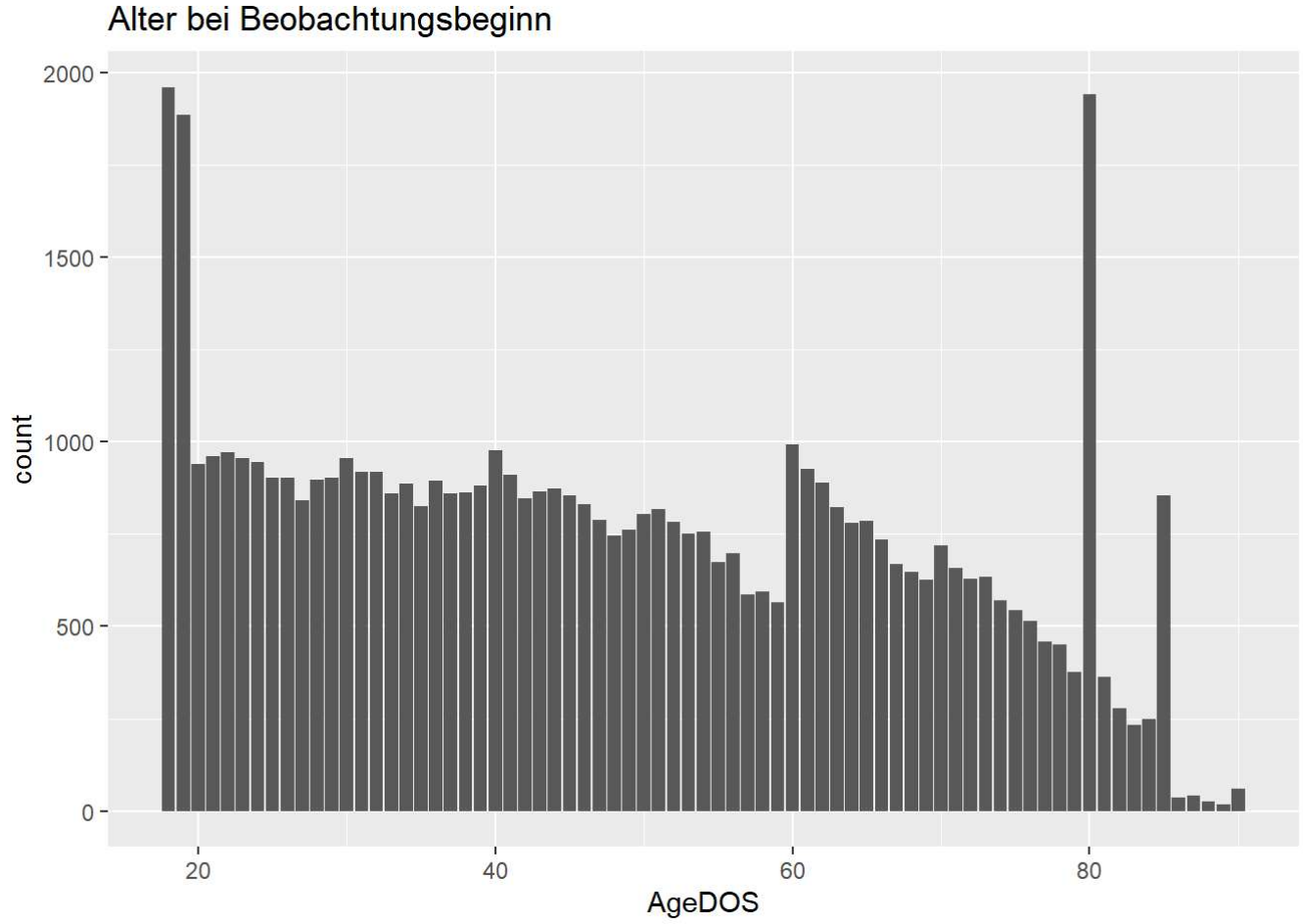



```
ggplot(data = data) +
  geom_mosaic(mapping = aes(x = product(EverSmoker,CurrentSmoker)) )+
  labs(x="Current Smoker", y="Ever Smoker", title = "Raucheranalyse")  +
  theme(axis.text.x = element_text(angle = 90))
```

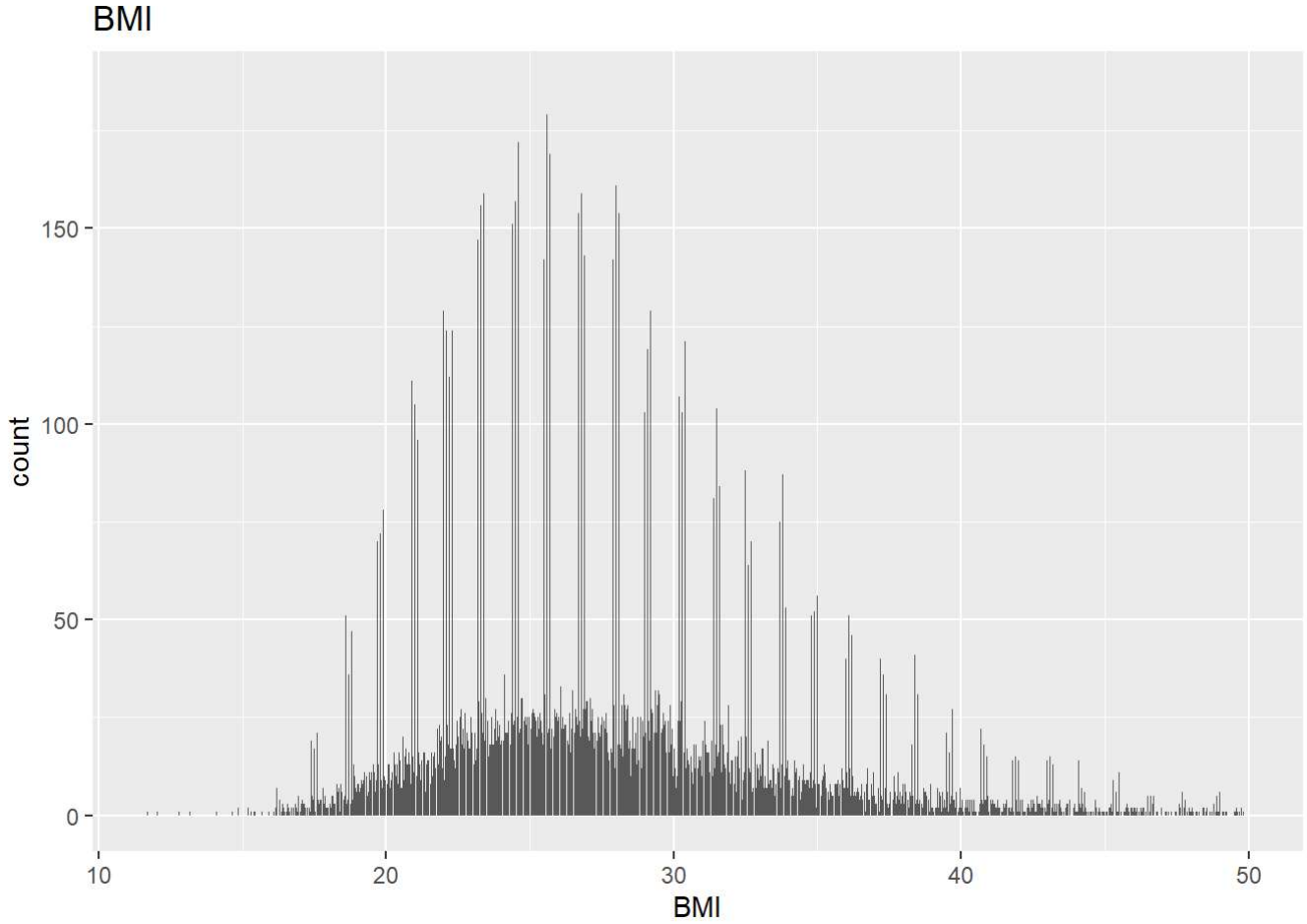
```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## ℹ Please use `unite()` instead.
## ℹ The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



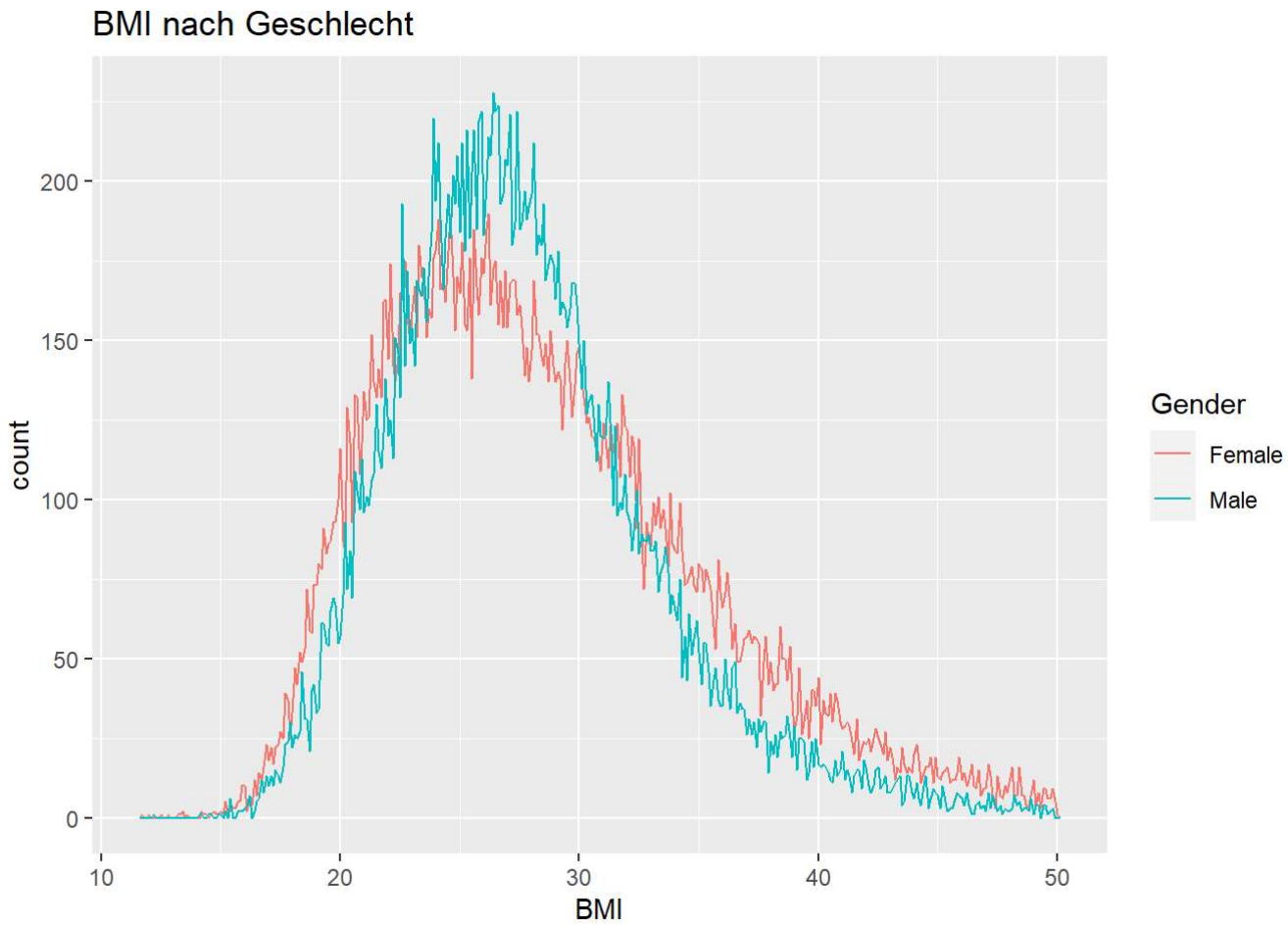

```
ggplot(data = data) +
  geom_bar(mapping = aes(x = AgeDOS)) + labs(title = "Alter bei Beobachtungsbeginn")
```



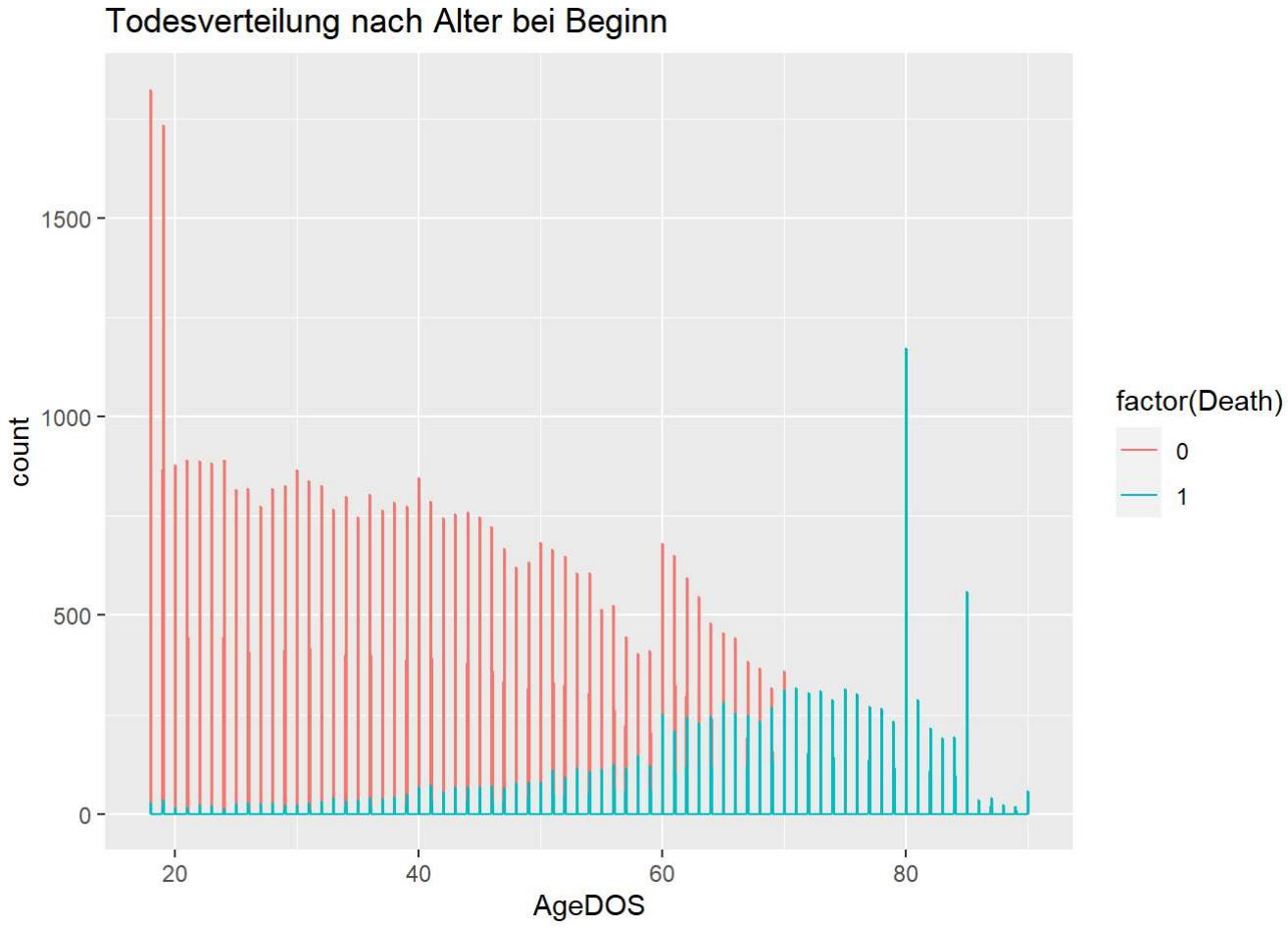
```
ggplot(data = data %>% filter(BMI < 50)) +
  geom_bar(mapping = aes(x = BMI)) + labs(title = "BMI")
```



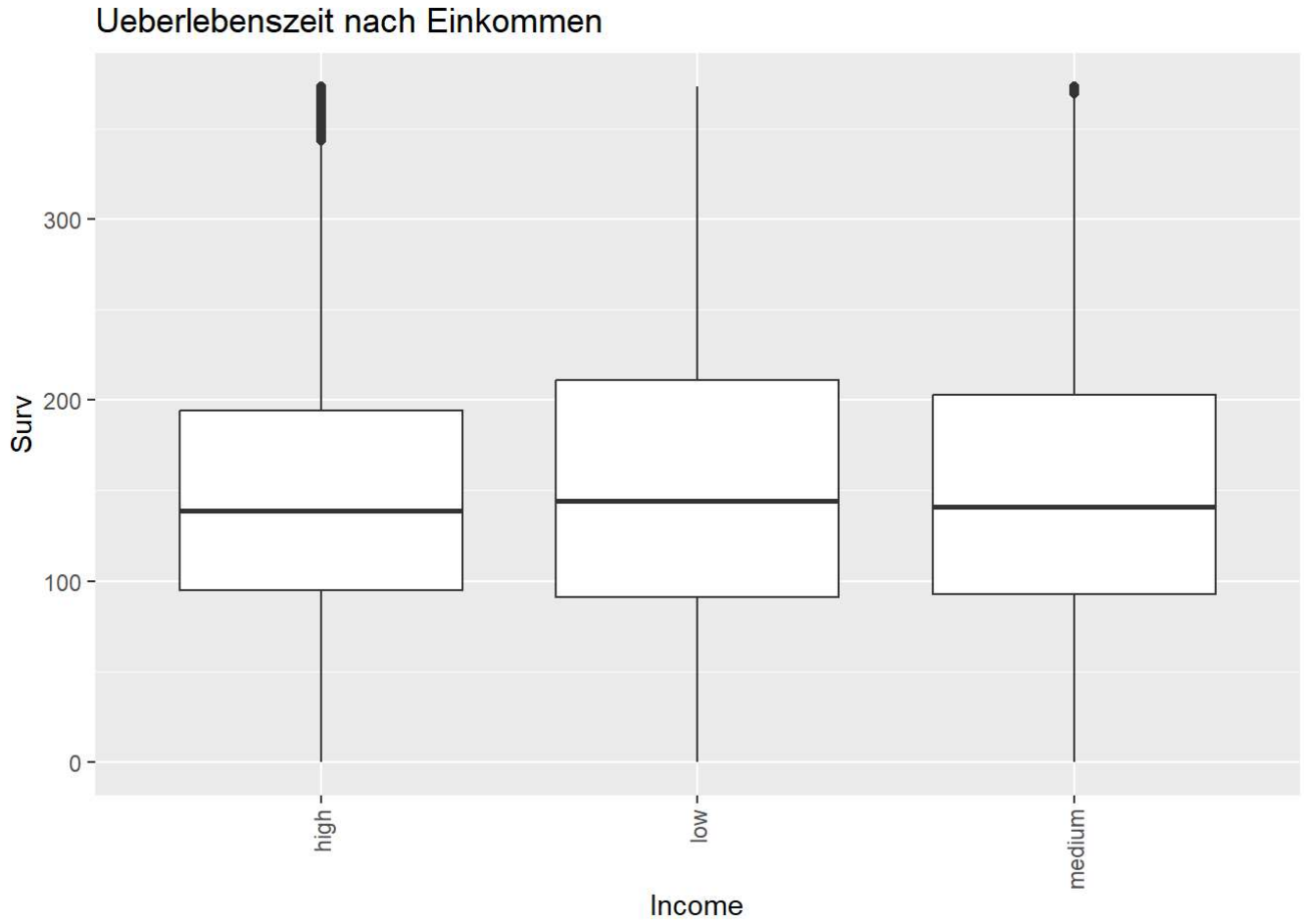

```
ggplot(data = data %>% filter(BMI < 50), mapping = aes(x = BMI, colour = Gender)) +
  geom_freqpoly(binwidth = 0.1) + labs(title = "BMI nach Geschlecht")
```




```
ggplot(data = data %>% filter(BMI < 50), mapping = aes(x = AgeDOS, colour = factor(Death))) +
  geom_freqpoly(binwidth = 0.1) + labs(title = "Todesverteilung nach Alter bei Beginn")
```



```
ggplot(data = data, mapping = aes(x =Income , y = Surv)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) + labs
(title = "Ueberlebenszeit nach Einkommen")
```



Antwort: Mit zunehmender Beobachtungszeit kann der Effekt Blue/White ggü. einem Akademikerbonus abgeglichen werden, z.B. mit Actual-to-Expected Analysen. Die Übertragbarkeit ist deshalb vorsichtig anzusetzen und regelmäßig zu kontrollieren.

Teil b): Survival Modell

Entwickeln Sie ein CoxPH-Modell als Referenzmodell, um abschätzen zu können, ob der Datensatz ihren Erwartungen hinsichtlich des Einflusses auf die Überlebenszeit entspricht. [Beschreibung Cox-Modell gek.]

Schritt 1: Aufbereitung und Kalibrierung

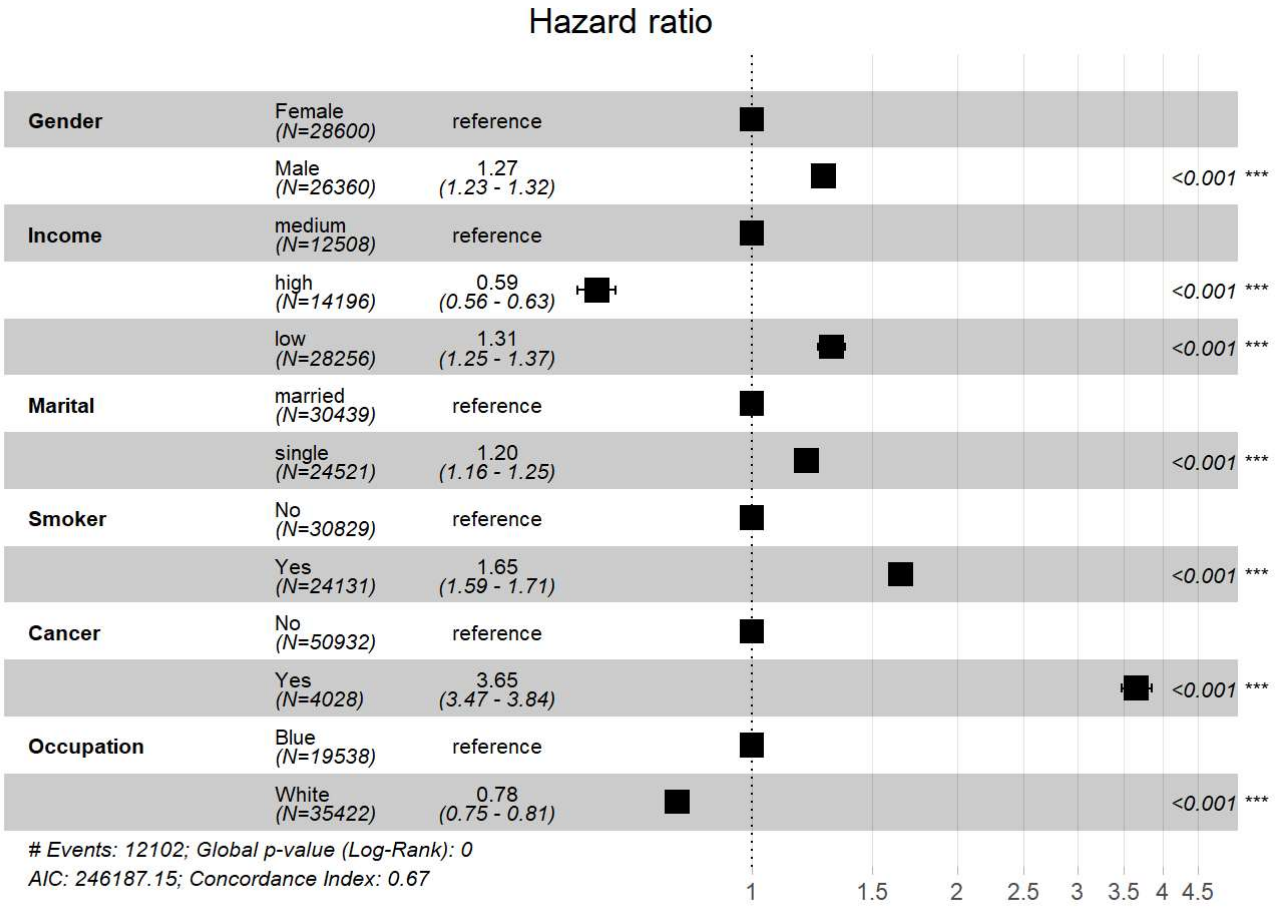
Verwenden Sie als erklärende Kovariablen die Kovariablen Einkommen, Beziehungsstatus, Rauchstatus, Geschlecht, Beschäftigungsverhältnis und verdichten Sie diese vorab geeignet.

Bilden Sie ein weiteres Modell, welches zusätzlich die Kovariable des Alters bei Beginn der Kohortenstudie berücksichtigt. Vergleichen und Interpretieren Sie die Ergebnisse.

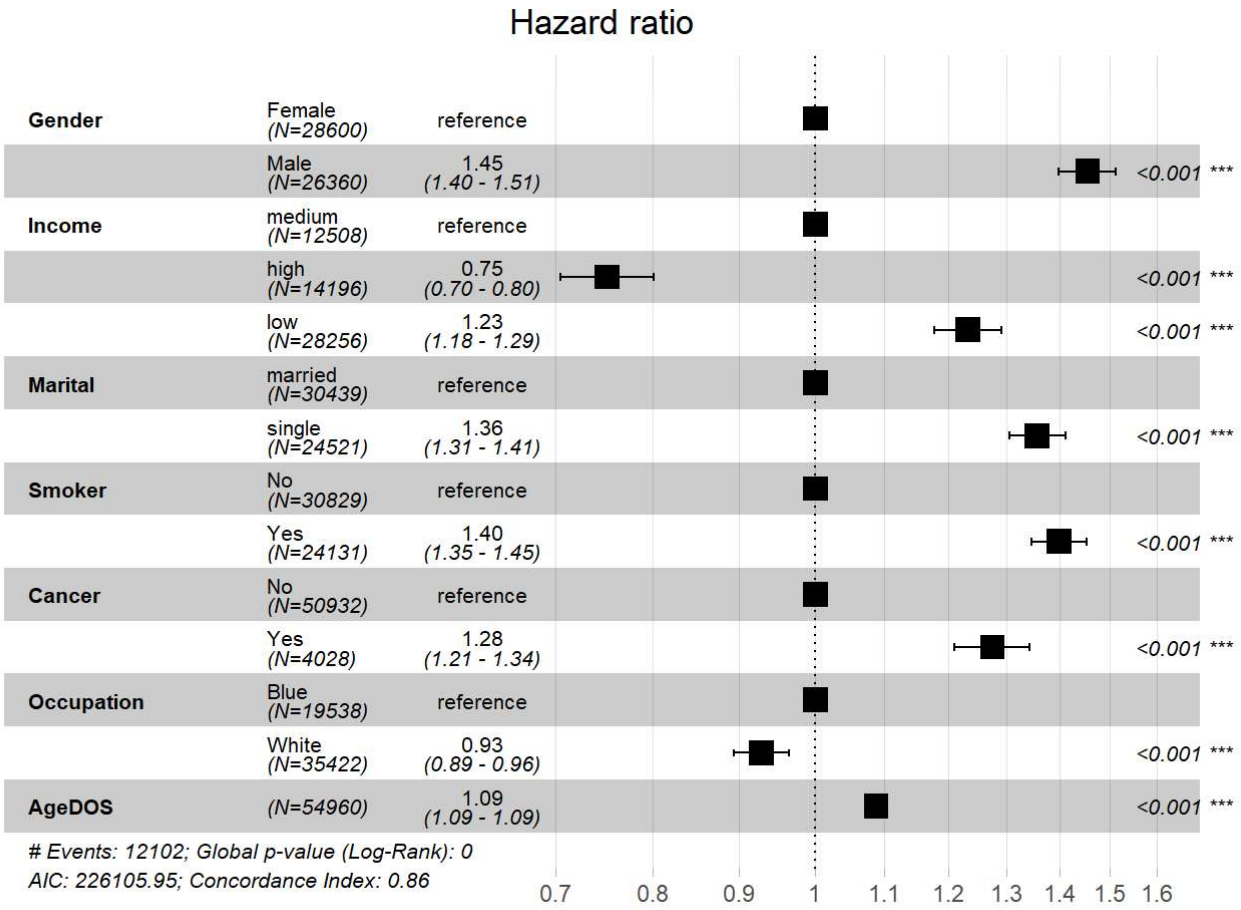
Lösungsvorschlag:

Schritt 2: Visualisierung

```
survival_formula = as.formula(paste(lhs, "~", rhs2))
fcox_standard2 = coxph(survival_formula , data = data)


pic_standard = ggforest(fcox_standard, data = data)
print(pic_standard)
```




```
pic_standard = ggforest(fcox_standard2, data = data)
print(pic_standard)
```



Antwort:

- Das Alter hat einen positiven Risikofaktor.
- Durch explizite Berücksichtigung der Kovariable Alter sinkt der Einfluss aller anderen Kovariablen erheblich. Dies ist genauso zu erwarten, da die Sterblichkeit mit zunehmendem Alter bekanntermaßen zunimmt.

Schritt 3: Interpretation

Antwort:

- Das CoxPH Modell schätzt keine Baseline Sterblichkeit, sodass keine qx Sterbewahrscheinlichkeiten abgeleitet werden können.

Aufgabe 2

Sie haben erkannt, dass das Alter sehr wichtig für die Prognose der Sterblichkeit ist und sie entschließen sich deshalb eine Tafel der Amerikanischen Aktuarsvereinigung (SOA) als Baseline-Sterblichkeit vorzugeben. Das Modell muss somit die altersbedingte Sterblichkeitsstruktur nicht selbst erlernen. Sie verwenden die Ultimate qx (Spalte ult.) der Unismoke VBT Tafel 2015, vgl. https://www.soa.org/resources/experience-studies/2015/2015-valuation-basic-tables/ (https://www.soa.org/resources/experience-studies/2015/2015-valuation-basic-tables/)

Gehen Sie davon aus, dass der vorliegende Datensatz einer repräsentativen Stichprobe der Gesamtbevölkerung der USA entspricht. Wie in Deutschland ist auch dort das Sterblichkeitsniveau von Versicherten geringer als das Sterblichkeitsniveau der Gesamtbevölkerung. Deshalb erhöhen Sie die qx der SOA mit einem pauschalen Multiplikator m1 und einem additiven Term m2 für die Alter >= 50, d.h. qx = m2*1_{x>=50} + m1*qx(SOA). Ermitteln Sie diese Faktoren auf Basis der vorliegenden Daten getrennt nach Männern und Frauen und einmal gemeinsam.

Teil a): Zensierung von Daten und Bearbeitung

Ermitteln Sie für jedes Alter ab 25 die empirischen Sterbewahrscheinlichkeiten des Datensatzes. Berücksichtigen Sie Links- und Rechtszensierung angemessen.

Um ein Maß für die Unsicherheit zu erhalten, ermitteln Sie zusätzlich einen angemessenen Konfidenzbereich auf Basis der empirischen Sterbewahrscheinlichkeit.

Lösungsvorschlag:

```
Calc_empiric_qx = function(Data){

  #Ausgabe initialisieren
  Empiric_qx = data.frame(Age = 25:120)
  Empiric_qx$Anzahl = 0
  Empiric_qx$Anzahl_Tote = 0

  #Alter durchgehen
  for(i in Empiric_qx$Age){
    index_age = which(Empiric_qx$Age==i)

    #Fall 1: Stirbt vorher oder noch nicht beobachtet wird ausgefiltert
    data_temp = Data %>% filter((yearDOE - yearDOB) >= i)

    #Fall 2: Alter wird beobachtet
    data_temp %<>% filter(AgeDOS < (i+1))

    #Damit wichtigste Zensierung abgedeckt

    #Berechnung der Quotienten
    #Fall 3: i jaehrige die ueberleben (ganze Jahre)
    data_f3 = data_temp %>% filter(AgeDOS <= i & (yearDOE - yearDOB) >= (i+1))
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] + nrow(data_f3)
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] + 0

    #Fall 4a i jaehrig sterbend (zensiert)
    data_f4a = data_temp %>% filter(AgeDOS <= i & (yearDOE - yearDOB) < (i+1) & Death == 1)
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] + nrow(data_f4a)
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] + nrow(data_f4a)

    #Damit sind die wichtigsten Quotienten abgedeckt

    #Fall 4b i jaehrig lebend zensiert
    data_f4b = data_temp %>% filter(AgeDOS <= i & (yearDOE - yearDOB) < (i+1) & Death == 0)
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] +nrow(data_f4b)/2
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] + 0

    #Fall 5a i jaehrig lebend (halbes Exposure)
    data_f5 = data_temp %>% filter(AgeDOS > i & (yearDOE - yearDOB) >= (i+1))
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] +nrow(data_f5)/2
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] + 0

    #Fall 6a unter jaehrig sterbend (halbes Exposure)
    data_f6a = data_temp %>% filter(AgeDOS > i & (yearDOE - yearDOB) < (i+1) & Death == 1)
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] +nrow(data_f6a)/2
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] +nrow(data_f6a)

    #Fall 6b unter jaehrig lebend zensiert
    data_f6b = data_temp %>% filter(AgeDOS > i & (yearDOE - yearDOB) < (i+1) & Death == 0)
    Empiric_qx$Anzahl[index_age] = Empiric_qx$Anzahl[index_age] +nrow(data_f6b)/2
    Empiric_qx$Anzahl_Tote[index_age] = Empiric_qx$Anzahl_Tote[index_age] + 0


  }
  #Quotienten ermitteln
```

```
  Empiric_qx %<>% mutate(qx = Anzahl_Tote/ Anzahl)

  return(Empiric_qx)
  }




##### Extrahiere alle x jaehrigen aus den Daten die Beobachtet werden und zaehle anzahl tote
/ lebende #####
Empiric_qx = Calc_empiric_qx(data)

#Werte für Konfidenzintervalle bestimmen (alternativen möglich)
Empiric_qx %<>% mutate(qx95 = qbinom(0.95,round(Anzahl),qx)/Anzahl)
Empiric_qx %<>% mutate(qx05 = qbinom(0.05,round(Anzahl),qx)/Anzahl)
```

Teil b): VBT-Tafeln visualisieren und Faktoren ermitteln

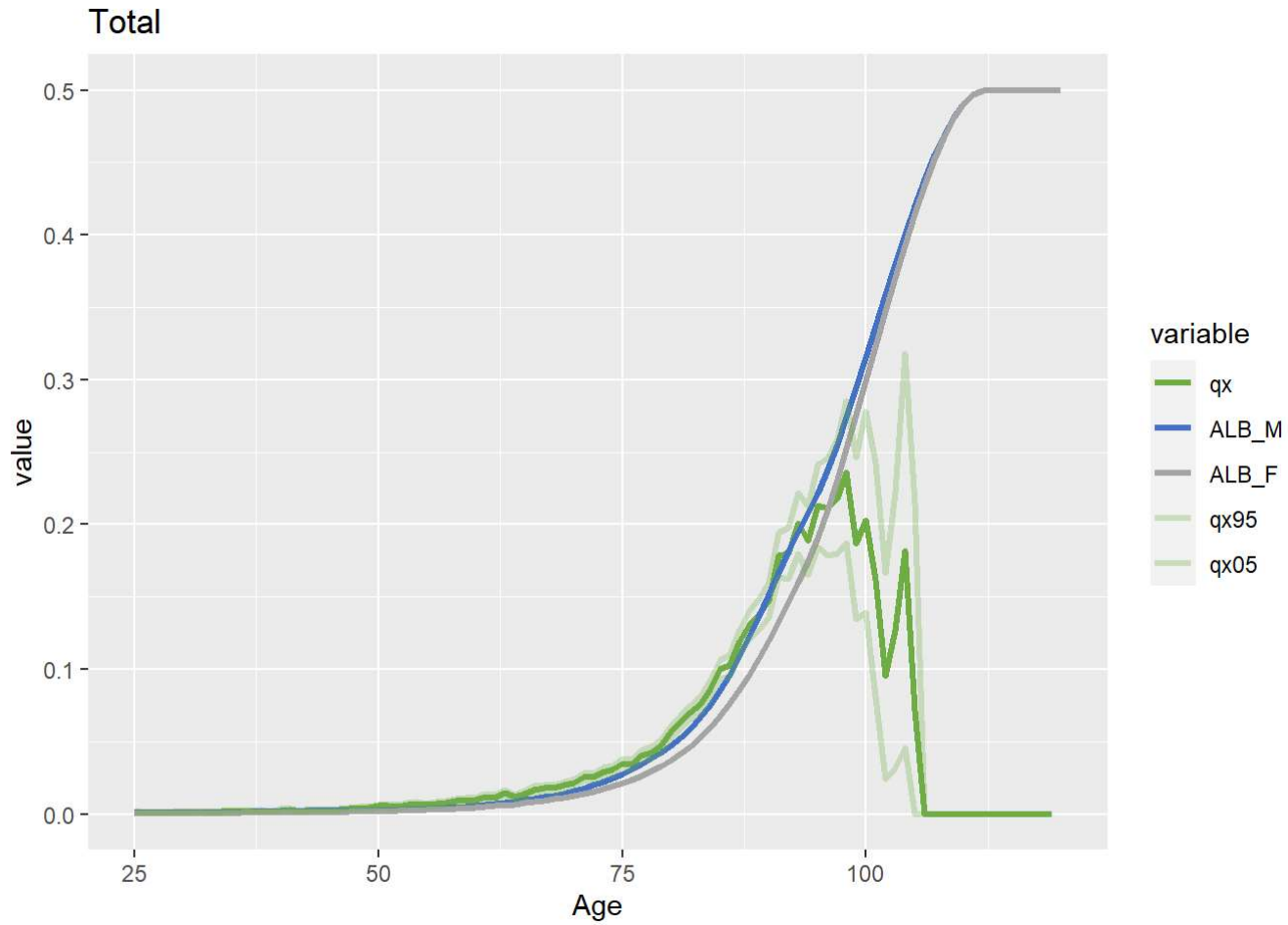
Lesen Sie die VBT-Tafeln ein und visualisieren Sie diese für Männer und Frauen gemeinsam mit der empirischen Sterbewahrscheinlichkeit und dem Konfidenzintervall aus Teilaufgabe 2a). Begründen Sie die prinzipielle Angemessenheit der VBT-Tafeln für die Anwendung.

Ermitteln Sie optimale Parameter m1, m2 (Optimierungs-Maß: gewichteter quadratischer Fehler) für Männer und Frauen getrennt und gemeinsam. Welche Faktoren verwenden Sie anschließend für die Analyse? Verwenden Sie nur den für die spätere Anwendung relevanten Altersbereich bis einschließlich Alter 70.

Lösungsvorschlag:

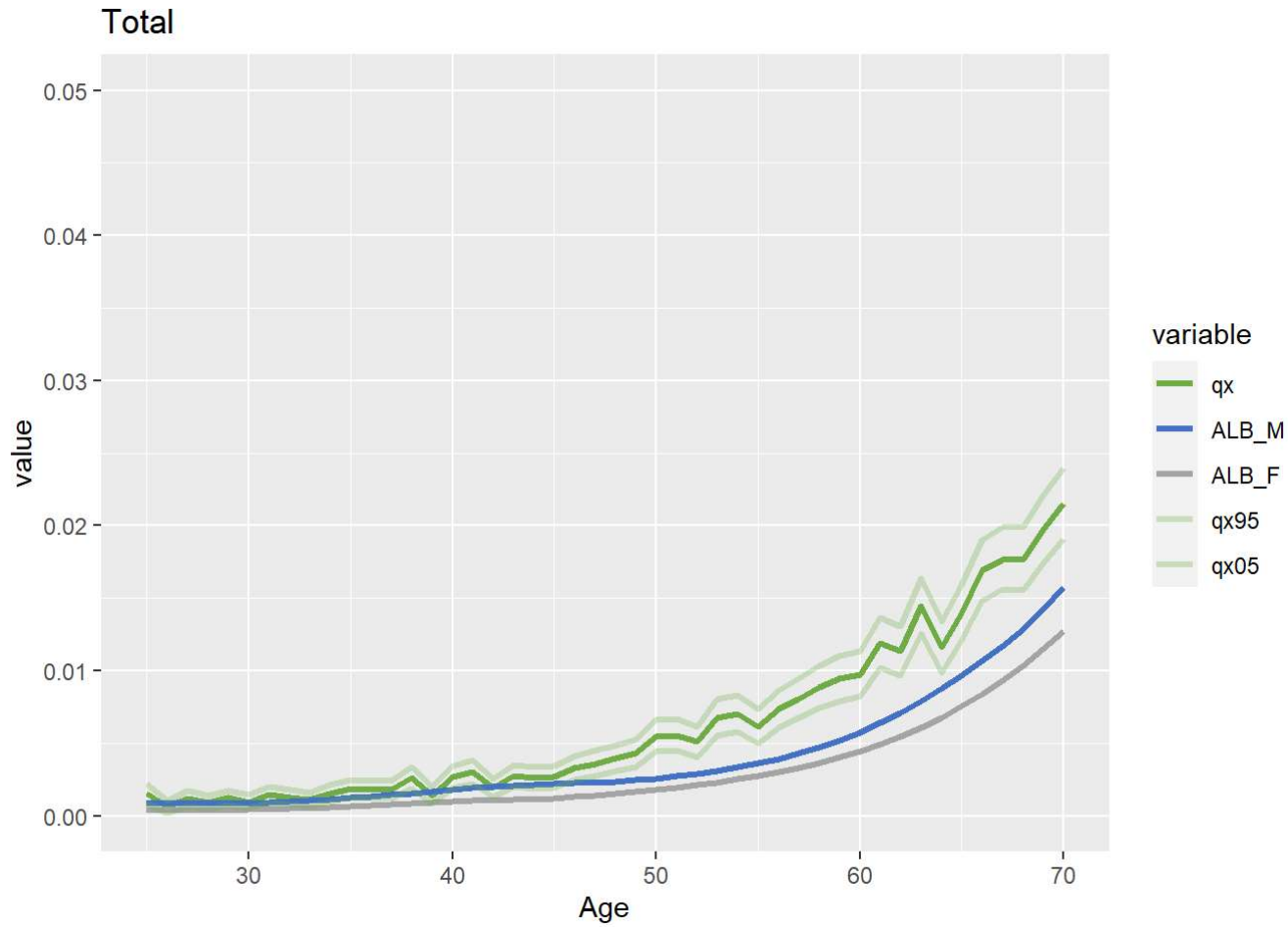
```
VBTM_ALB = read.xlsx('2015-vbt-unismoke-alb-anb.xlsx',sheet='Male Unismoke ALB',detectDates =
T,startRow = 3)
VBTF_ALB = read.xlsx('2015-vbt-unismoke-alb-anb.xlsx',sheet='Female Unismoke ALB',detectDates
= T,startRow = 3)


#Verwendung der Ultimate Tafel
Empiric_qx$ALB_M = VBTM_ALB %>% pull(Ult.)/1000
Empiric_qx$ALB_F = VBTF_ALB %>% pull(Ult.)/1000


#Visualisierung
plot_data = reshape2::melt(Empiric_qx[,c("Age","qx","ALB_M","ALB_F","qx95","qx05")],id.vars
="Age")
pq <- plot_data %>%
  ggplot( aes(x=Age, y = value, colour=variable,alpha = variable)) +
  geom_line(lwd = 1.2) +
  scale_colour_manual(values=c("#70AD47", "#4472C4","#A5A5A5","#70AD47","#70AD47")) +
  scale_alpha_manual(values = c(1,1,1, 0.3, 0.3))+
  labs(fill="")+ylim(0,0.5)+ggtitle("Total")
pq
```

```
## Warning: Removed 3 rows containing missing values (`geom_line()`).
```




```
plot_data = reshape2::melt(Empiric_qx[,c("Age","qx","ALB_M","ALB_F","qx95","qx05")],id.vars
="Age")
pq2 <- plot_data %>%
  ggplot( aes(x=Age, y = value, colour=variable,alpha = variable)) +
  geom_line(lwd = 1.2) +
  scale_colour_manual(values=c("#70AD47", "#4472C4","#A5A5A5","#70AD47","#70AD47")) +
  scale_alpha_manual(values = c(1,1,1, 0.3, 0.3))+
  labs(fill="")+ylim(0,0.05)+xlim(25,70)+ggtitle("Total")
pq2
```

```
## Warning: Removed 250 rows containing missing values (`geom_line()`).
```



Antwort:

Struktur und Niveau der Tafeln ist vergleichbar, insbesondere für den relevanten Altersbereich.

```
#Optimierungsfunktion:
f_individual = function(m,qxdata){
  Diff = sqrt(weighted.mean((qxdata$qx - m*qxdata$Table)^2,w = qxdata$Anzahl))
  return(Diff)
}
f_indmultadd = function(m,qxdata){
  Diff = sqrt(weighted.mean((qxdata$qx - (m[1]*(qxdata$Age>=50)+m[2]*qxdata$Table))^2,w = qxd
ata$Anzahl_Tote))
}



#Frauen
data_F = data %>% filter(Gender == "Female")
Empiric_qx_F = Calc_empiric_qx(data_F)
Empiric_qx_F$Table = VBTF_ALB %>% pull(Ult.)/1000
v1_f = nlm(f_individual,p = 1, qxdata = Empiric_qx_F[1:46,])
v2_f = optimx(par = c(0,1),fn = f_indmultadd,qxdata = Empiric_qx_F[1:46,])



#Männer
data_M = data %>% filter(Gender == "Male")
Empiric_qx_M = Calc_empiric_qx(data_M)
Empiric_qx_M$Table = VBTM_ALB %>% pull(Ult.)/1000
v1_m = nlm(f_individual,p = 1, qxdata = Empiric_qx_M[1:46,])
v2_m = optimx(par = c(0,1),fn = f_indmultadd,qxdata = Empiric_qx_M[1:46,])


#Gesamt
Empiric_qx = Calc_empiric_qx(data)
Empiric_qx$TableM = VBTM_ALB %>% pull(Ult.)/1000
Empiric_qx$TableF = VBTF_ALB %>% pull(Ult.)/1000
Empiric_qx$Table = (Empiric_qx$TableM * Empiric_qx_M$Anzahl + Empiric_qx$TableF * Empiric_qx_
F$Anzahl) /(Empiric_qx$Anzahl)
v1 = nlm(f_individual,p = 1, qxdata = Empiric_qx[1:46,])
v2 = optimx(par = c(0,1),fn = f_indmultadd,qxdata = Empiric_qx_F[1:46,])

print(v1)
```

```
## $minimum
## [1] 0.00114378
##
## $estimate
## [1] 1.714548
##
## $gradient
## [1] -5.542295e-08
##
## $code
## [1] 1
##
## $iterations
## [1] 4
```

```
print(v2)
```

```
##                     p1       p2       value fevals gevals niter convcode kkt1
## Nelder-Mead 0.003141175 1.117722 0.001092709     63     NA    NA        0 TRUE
## BFGS        0.003141576 1.117697 0.001092709     22     11    NA        0 TRUE
##             kkt2 xtime
## Nelder-Mead TRUE     0
## BFGS        TRUE     0
```

Antwort:

- Um ein Unisex-Pricing durchführen zu können, wird der gemeinsame Faktor, d.h. 12% und ein additiver Zuschlag von 0,3 % verwendet.

Aufgabe 3

Teil a): Datensatz transformieren

Überführen Sie die Aufgabenstellung in ein Klassifikationsproblem. Erzeugen Sie dazu aus jedem Datensatz (i.A.) mehrere Zeilen, indem Sie für jedes beobachtete Altersjahr der Person schauen, ob die betroffene Person gestorben ist (label=1) oder ob diese überlebt hat (label=0), siehe Tabelle 1. Berücksichtigen Sie auch die Zensierung der Daten. Verwenden Sie nur den für das Pricing besonders relevanten Altersbereich bis einschließlich Alter 70.

Table 1: Classification Data

ID	Age	Smoker	Label
23	60	1	0
23	61	1	0
23	62	1	1
46	58	0	0

Lösungsvorschlag:

```
#Analoges Vorgehen wie bei der Berücksichtigung von Zensierung für die Ermittlung der qx
aggregation_func = function(Data,age){
  obs_age = list()

  #F1: Stirbt vorher oder noch nicht beobachtet
  data_temp = Data %>% filter((yearDOE - yearDOB) >= age) #Alter bei Tod war >= i    oder ist
noch am Leben

  #F2: Beobachtung startet nach i + 1
  data_temp %<>% filter(AgeDOS < (age+1))  #Kunde wurde mit Eintrittsalter <i+1 recorded

  #F3: i jaehrige die ueberleben
  data_f3 = data_temp %>% filter(AgeDOS <= age & (yearDOE - yearDOB) >= (age+1))

  #F4a i jaehrig sterbend
  data_f4a = data_temp %>% filter(AgeDOS <= age & (yearDOE - yearDOB) < (age+1) & Death == 1)

  #F4b i jaehrig lebend ende zensiert
  data_f4b = data_temp %>% filter(AgeDOS <= age & (yearDOE - yearDOB) < (age+1) & Death == 0)

  #F5a i jaehrig lebend anfang zensiert
  data_f5 = data_temp %>% filter(AgeDOS > age & (yearDOE - yearDOB) >= (age+1))

  #F6a unter jaehrig sterbend
  data_f6a = data_temp %>% filter(AgeDOS > age & (yearDOE - yearDOB) < (age+1) & Death == 1)

  #F6b unter jaehrig lebend zensiert
  data_f6b = data_temp %>% filter(AgeDOS > age & (yearDOE - yearDOB) < (age+1) & Death == 0)

  obs_age$surv = c(data_f3$SEQN)
  obs_age$death = c(data_f4a$SEQN)
  return(obs_age)
}


index25 = aggregation_func(data,25)

df_classifs = data %>% filter(SEQN %in% index25$surv)
df_classifs %<>% mutate(age = 25,
                        Death_indicator = 0)
df_classifd = data %>% filter(SEQN %in% index25$death)
df_classifd %<>% mutate(age = 25,
                        Death_indicator = 1)

df_classif = rbind(df_classifs,df_classifd)
for(age_range in 26:70){

  index = aggregation_func(data,age_range)

  df_classifs = data %>% filter(SEQN %in% index$surv)
  df_classifs %<>% mutate(age = age_range,
                          Death_indicator = 0)
  df_classifd = data %>% filter(SEQN %in% index$death)
  df_classifd %<>% mutate(age = age_range,
                          Death_indicator = 1)
```

```
  df_classif = rbind(df_classif, df_classifs,df_classifd)
}


df_classif %<>% mutate(Death_indicator = factor(Death_indicator))
```

Teil b): GLM

Schritt 1: Kalibrierung GLM

Entwickeln Sie ein GLM unter Verwendung der Kovariablen Einkommen, Beziehungsstatus, Raucherstatus, Geschlecht und Beschäftigungsverhältnis. Zudem berücksichtigen Sie Vorerkrankungen Diabetes und den BMI, die ggf. zu Zuschlägen im Pricing führen und deshalb hier berücksichtigt/kontrolliert werden sollen.

Wählen Sie eine geeignete Verteilungsannahme und setzen Sie die VBT-Tafel erhöht um den Faktor 10% und addiert mit 0,3% ab Alter 50 als Baseline-Sterblichkeit an. Imputieren Sie fehlende Werte des BMI geeignet und begründen Sie die getroffene Maßnahme.

Lösungsvorschlag:

```
df_classif %<>% left_join(Empiric_qx %>% select("Age","Table"),by = c("age"="Age"))
df_classif %<>% mutate(qx = Table * 1.1 + 0.003*(age>=50))

# Berücksichtigung der link funktion im Offset
df_classif %<>% mutate(logitoff = log(qx/(1-qx)))
df_classif %<>% mutate(BMI = replace_na(BMI,mean(df_classif$BMI,na.rm = T)))


m6 = glm(Death_indicator ~ Income + Marital + Smoker + Gender +
           Occupation + Diabetes + BMI,# + Marital * Gender ,
         data = df_classif,
         family = binomial,
         offset = logitoff)
```

Schritt 2: Modellanalyse

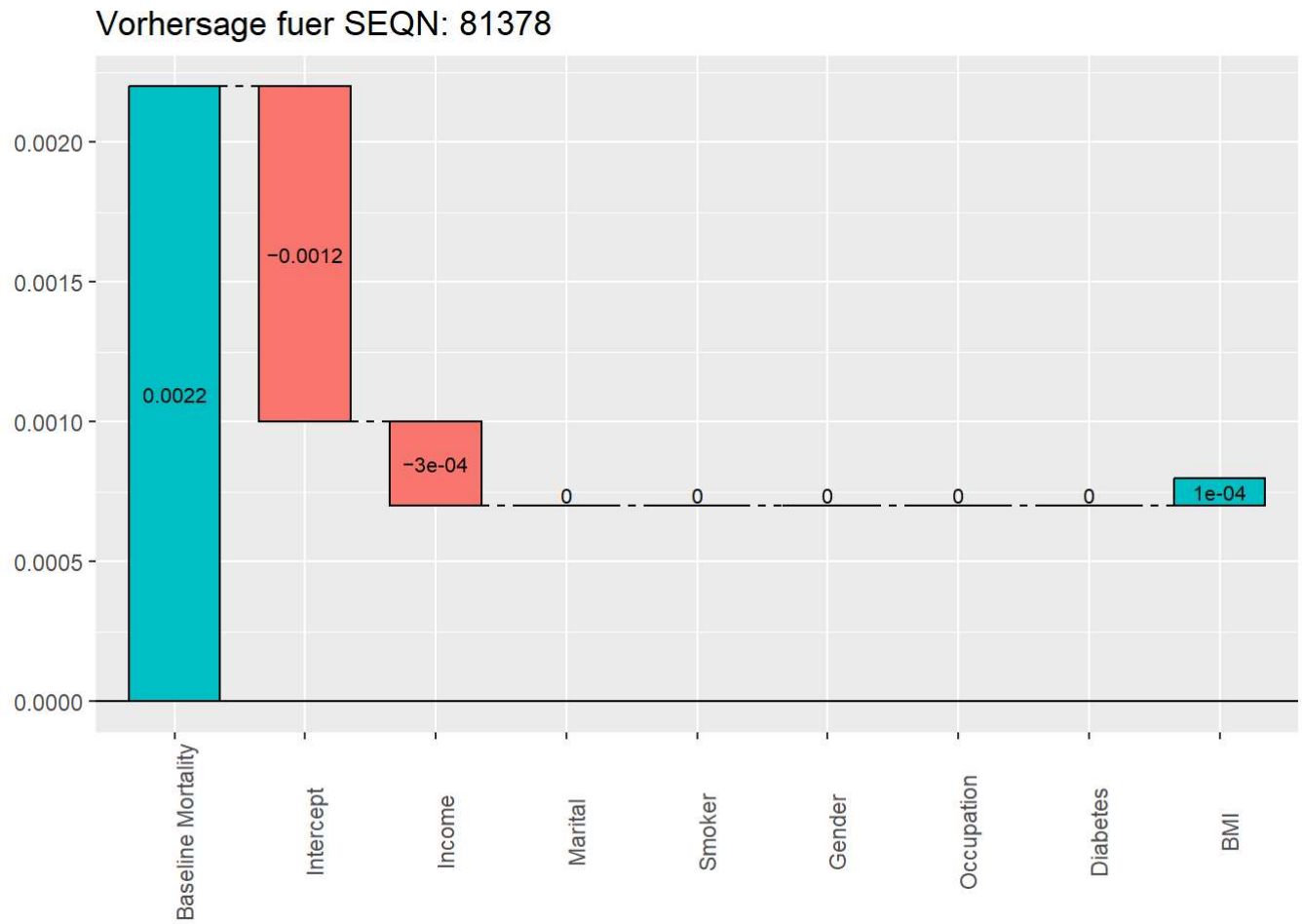
Plausibilisieren Sie die Ergebnisse. Gibt es neben dem Akademikerbonus weitere Ansatzpunkte für Differenzierung? Welche würden Sie vorschlagen im Pricing zu nutzen? Welche Versicherungsnehmer sollten im Rahmen einer Risikoprüfung grundsätzlich ausgeschlossen werden? Stellen Sie die vorhergesagten Sterbewahrscheinlichkeiten des Modells für eine ID und ein Alter anhand eines Wasserfalldiagramms dar.

Lösungsvorschlag:

```
summary(m6)
```

```
##
## Call:
## glm(formula = Death_indicator ~ Income + Marital + Smoker + Gender +
##     Occupation + Diabetes + BMI, family = binomial, data = df_classif,
##     offset = logitoff)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7251  -0.1198  -0.0678  -0.0441   3.9950
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.801897   0.103355  -7.759 8.58e-15 ***
## Incomehigh      -0.425064   0.062865  -6.762 1.37e-11 ***
## Incomelow        0.336556   0.047455   7.092 1.32e-12 ***
## Maritalsingle    0.405340   0.037806  10.721  < 2e-16 ***
## SmokerYes        0.569911   0.039522  14.420  < 2e-16 ***
## GenderMale       0.382484   0.038186  10.016  < 2e-16 ***
## OccupationWhite -0.096031   0.038585  -2.489   0.0128 *
## DiabetesYes      1.656394   0.055218  29.997  < 2e-16 ***
## BMI              0.006338   0.002766   2.292   0.0219 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36251  on 522634  degrees of freedom
## Residual deviance: 34524  on 522626  degrees of freedom
## AIC: 34542
##
## Number of Fisher Scoring iterations: 9
```

```
coef_m6 = coef(m6)
index = which(df_classif$SEQN == 81378 & df_classif$age==49)


v_index = c(
            df_classif$logitoff[index],
            coef_m6[1],
            ifelse(df_classif$Income[index]=="low",coef_m6[2],coef_m6[3]),
           ifelse(df_classif$Marital[index] =="alleinstehend",coef_m6[4],0),
           ifelse(df_classif$Smoker[index] == "Ja",coef_m6[5],0),
           ifelse(df_classif$Gender[index] == "Male",coef_m6[6],0),
           ifelse(df_classif$Occupation[index]=="White",coef_m6[7],0),
           ifelse(df_classif$Diabetes[index] == "Yes",coef_m6[8],0),
           df_classif$BMI[index] *  coef_m6[9])

v_index_mult = exp(cumsum(v_index))/(1+exp(sum(v_index)))
waterfall(values= c(v_index_mult[1],diff(v_index_mult)) %>% round(4),
          labels = c("Baseline Mortality","Intercept","Income","Marital","Smoker","Gender",
                                    "Occupation","Diabetes","BMI")) +
          labs(title = str_c("Vorhersage fuer SEQN: ",81378)) +
          theme(axis.text.x = element_text(angle = 90))
```



Antwort: Zahlreiche weitere untersuchte Merkmale zeigen sogar einen größeren Einfluss in der Analyse als deras Beschäftigungsverhältnis / Akademikerbonus und können im Pricing berücksichtigt werden. Die Faktoren für Diabetes sind sehr hoch. Dies kann ein Anhaltspunkt für einen Ausschluss sein oder einen sehr hohen Zuschlag rechtfertigen. #####

Teil c): CatBoost

Entwickeln Sie ein CatBoost-Modell zum Vergleich unter Verwendung der Kovariablen Einkommen, Beziehungsstatus, Rauchstatus, Geschlecht und Beschäftigungsverhältnis. Zudem berücksichtigen Sie Vorerkrankungen Diabetes und den BMI, die ggf. zu Zuschlägen im Pricing oder zu Ausschlüssen in der Risikoprüfung führen und deshalb hier berücksichtigt/kontrolliert werden sollen.

Zerlegen Sie den Datensatz in Training und Validierung. Definieren Sie ein CatBoost-Modell und kalibrieren Sie dieses. Wählen Sie eine geeignete Parametrisierung (Hinweis: ein ausführliches Parameter-Tuning ist nicht notwendig) und setzen Sie die VBT-Tafel erhöht um den Faktor 10% und addiert mit 0,3% ab Alter 50 als Baseline-Sterblichkeit an.

Plausibilisieren Sie die Ergebnisse. Erstellen Sie Modellvorhersagen und vergleichen Sie diese mit dem GLM und mit der empirischen Beobachtung sowie der in Aufgabe 2 ermittelten Baseline Sterblichkeit. Bilden Sie dazu mittlere Vorhersagen für jedes Alter und vergleichen Sie diese (jeweils für Training und Test) visuell.

```
#Anleitung für Catboost Download
#Download R Package:  https://github.com/catboost/catboost/releases
#manuell installierern:
#install.packages("D:/DAV/ADS Completion/NHANES/catboost Download/catboost-R-Windows-1.1.1/ca
tboost-R-Windows-1.1.1(1).tar", repos = NULL,type= "source")
```

Schritt 1: Training Test

Lösungsvorschlag:

```
data  = df_classif %>% select(Death_indicator,Income,Marital,
                              Smoker,Gender, Occupation ,Diabetes,BMI)

# Split out validation dataset
# create a list of 80% of the rows in the original dataset we can use for training
set.seed(7)
training_index <- createDataPartition(data$Death_indicator, p=0.80, list=FALSE)
# select 20% of the data for validation
validation <- data[-training_index,]
# use the remaining 80% of data to training and testing the models
train <- data[training_index,]

y_train <- ifelse(as.numeric(train$Death_indicator)==1,0,1)
X_train <- train %>% select(-Death_indicator)
y_valid <- ifelse(as.numeric(validation$Death_indicator)==1,0,1)
X_valid <- validation %>% select(-Death_indicator)
```

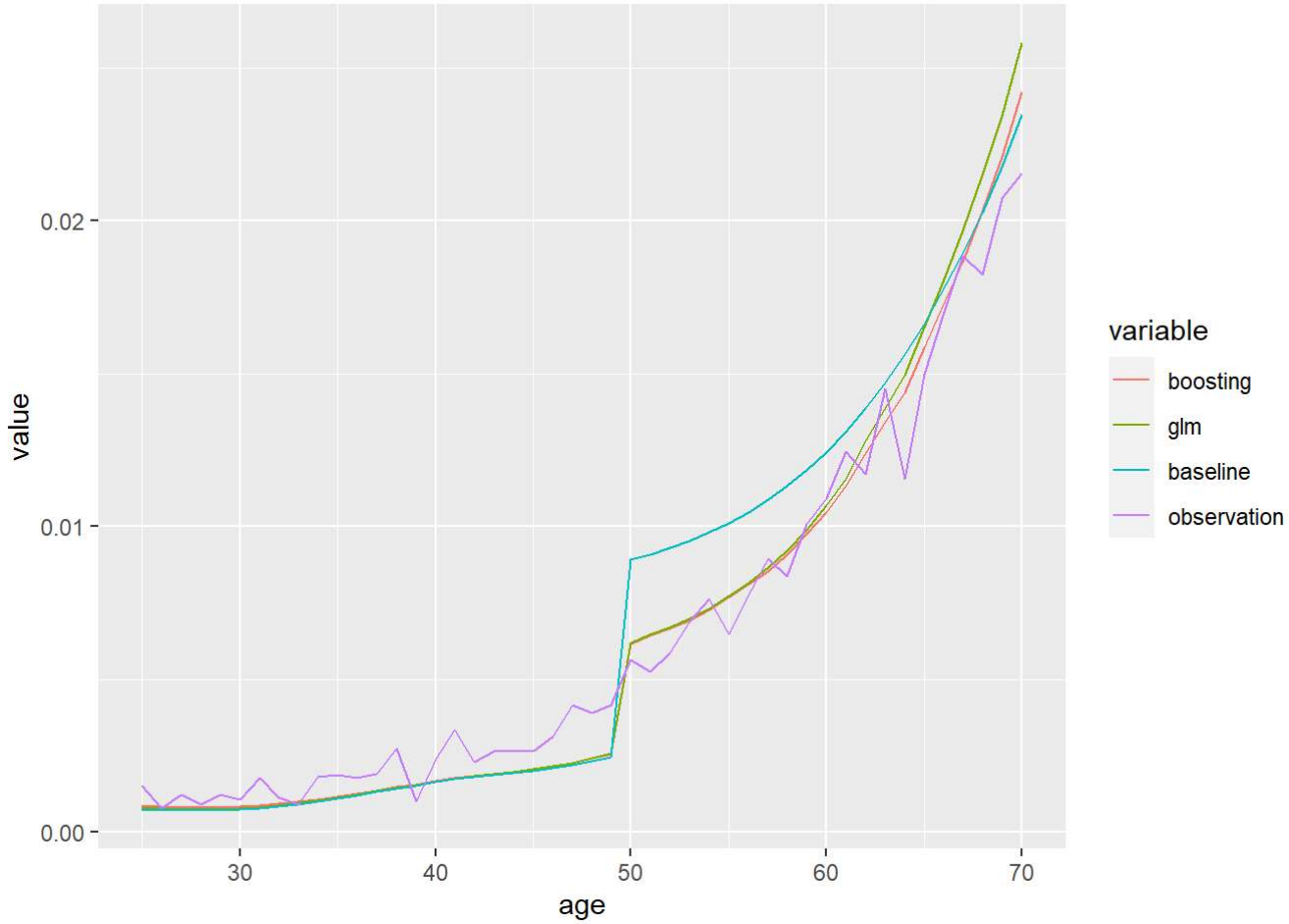
Schritt 2: Definition und Kalibrierung

```
train_pool <- catboost.load_pool(data = X_train, label = y_train, baseline = matrix(df_classi
f$logitoff[training_index]))
test_pool <- catboost.load_pool(data = X_valid, label = y_valid, baseline = matrix(df_classif
$logitoff[-training_index]))

# build model
params <- list(iterations=1000,
               learning_rate=0.01,
               depth=10,
               loss_function='Logloss',
               eval_metric='Logloss',
               random_seed = 55,
               metric_period = 50,
               use_best_model=TRUE)

model <- catboost.train(train_pool, test_pool, params)
```

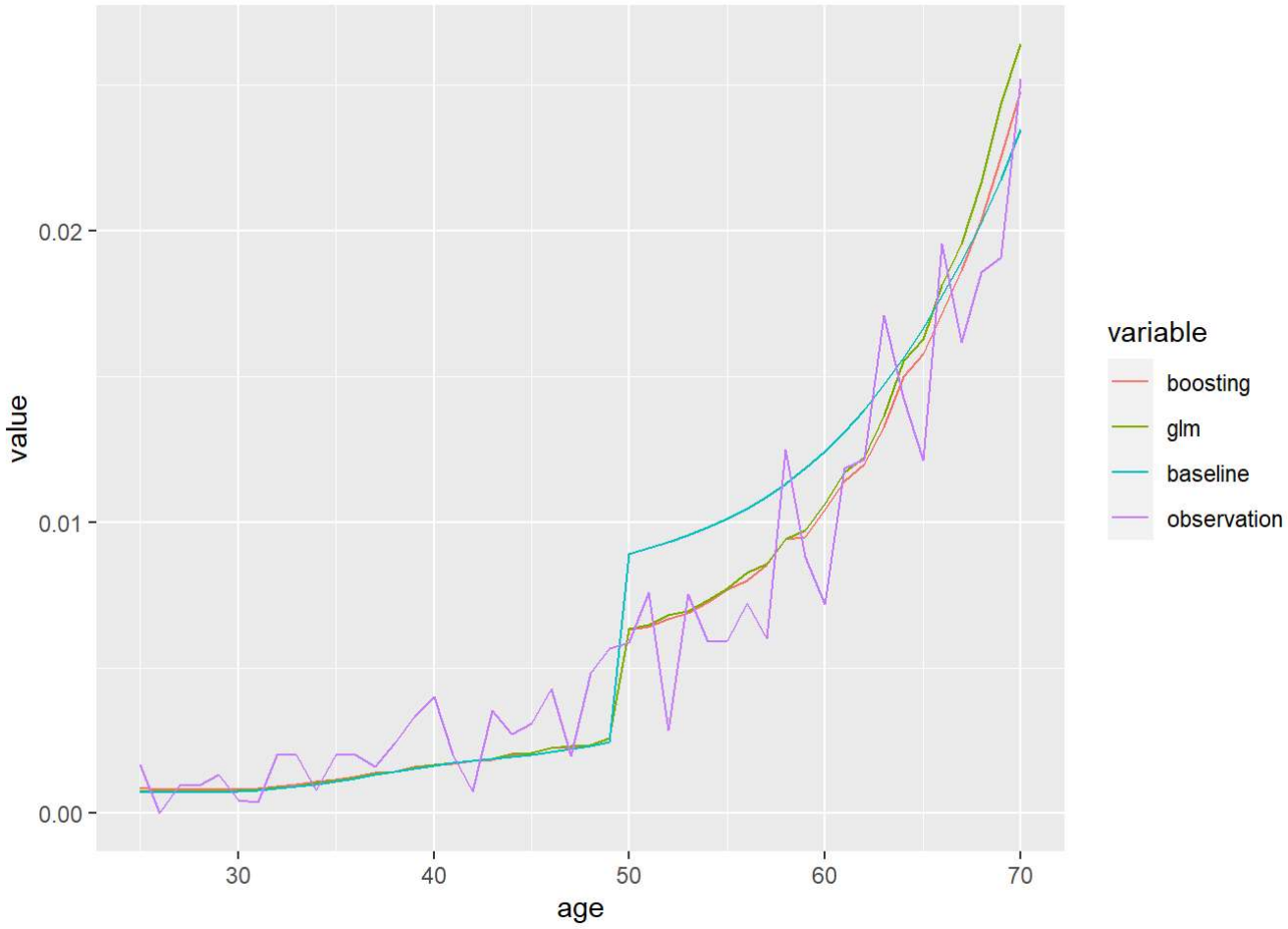
```
## 0:   learn: 0.0348063    test: 0.0347645 best: 0.0347645 (0) total: 392ms    remaining: 6m
31s
## 50:  learn: 0.0336355    test: 0.0337767 best: 0.0337767 (50)    total: 8.99s    remainin
g: 2m 47s
## 100: learn: 0.0330685    test: 0.0333408 best: 0.0333408 (100)   total: 17.1s    remainin
g: 2m 31s
## 150: learn: 0.0327609    test: 0.0331484 best: 0.0331484 (150)   total: 24.5s    remainin
g: 2m 17s
## 200: learn: 0.0325670    test: 0.0330602 best: 0.0330602 (200)   total: 32.9s    remainin
g: 2m 10s
## 250: learn: 0.0324452    test: 0.0330193 best: 0.0330193 (250)   total: 40.6s    remainin
g: 2m 1s
## 300: learn: 0.0323326    test: 0.0330011 best: 0.0330011 (300)   total: 50.3s    remainin
g: 1m 56s
## 350: learn: 0.0322261    test: 0.0330001 best: 0.0330001 (350)   total: 1m 1s    remainin
g: 1m 53s
## 400: learn: 0.0321303    test: 0.0330040 best: 0.0330001 (350)   total: 1m 13s   remainin
g: 1m 50s
## 450: learn: 0.0320616    test: 0.0330129 best: 0.0330001 (350)   total: 1m 24s   remainin
g: 1m 43s
## 500: learn: 0.0319987    test: 0.0330204 best: 0.0330001 (350)   total: 1m 35s   remainin
g: 1m 35s
## 550: learn: 0.0319502    test: 0.0330235 best: 0.0330001 (350)   total: 1m 45s   remainin
g: 1m 26s
## 600: learn: 0.0319182    test: 0.0330275 best: 0.0330001 (350)   total: 1m 54s   remainin
g: 1m 16s
## 650: learn: 0.0318843    test: 0.0330283 best: 0.0330001 (350)   total: 2m 3s    remainin
g: 1m 6s
## 700: learn: 0.0318449    test: 0.0330307 best: 0.0330001 (350)   total: 2m 13s   remainin
g: 56.9s
## 750: learn: 0.0318120    test: 0.0330359 best: 0.0330001 (350)   total: 2m 22s   remainin
g: 47.2s
## 800: learn: 0.0317819    test: 0.0330402 best: 0.0330001 (350)   total: 2m 31s   remainin
g: 37.7s
## 850: learn: 0.0317549    test: 0.0330459 best: 0.0330001 (350)   total: 2m 40s   remainin
g: 28.1s
## 900: learn: 0.0317356    test: 0.0330443 best: 0.0330001 (350)   total: 2m 48s   remainin
g: 18.5s
## 950: learn: 0.0317032    test: 0.0330516 best: 0.0330001 (350)   total: 2m 57s   remainin
g: 9.14s
## 999: learn: 0.0316797    test: 0.0330535 best: 0.0330001 (350)   total: 3m 6s    remainin
g: 0us
##
## bestTest = 0.03300010519
## bestIteration = 350
##
## Shrink model to first 351 iterations.
```

Schritt 3: Modellbewertung und Visualisierung

```
#Vorhersagen erstellen GLm und Catboost
y_pred=catboost.predict(model,train_pool)
y_predglm = predict(m6,newdata = df_classif[training_index,])


#train
train = df_classif[training_index,]
train$vorhersage_boost = exp(y_pred)/(1+exp(y_pred))
train$vorhersage_glm = exp(y_predglm)/(1+exp(y_predglm))
train$obs = ifelse(as.numeric(train$Death_indicator)==1,0,1)

temp = train %>% group_by(age) %>%
  summarise(boosting = mean(vorhersage_boost),
            glm = mean(vorhersage_glm,na.rm=T),
            baseline=mean(qx*1.1+0.003*(age>=50)),
            observation = mean(obs)) %>% ungroup()

melttemp = melt(temp, id = c("age"))
pic = ggplot(melttemp, aes(x=age,y = value,color = variable) )+ geom_line()
pic
```



```
#test
y_pred=catboost.predict(model,test_pool)
y_predglm = predict(m6,newdata = df_classif[-training_index,])


validation = df_classif[-training_index,]
validation$vorhersage_boost = exp(y_pred)/(1+exp(y_pred))
validation$vorhersage_glm = exp(y_predglm)/(1+exp(y_predglm))
validation$obs = ifelse(as.numeric(validation$Death_indicator)==1,0,1)

temp = validation %>% group_by(age) %>%
  summarise(boosting = mean(vorhersage_boost),
            glm = mean(vorhersage_glm,na.rm=T),
            baseline = mean(qx*1.1+0.003*(age>=50)),
            observation = mean(obs)) %>% ungroup()

melttemp = melt(temp, id = c("age"))
pic = ggplot(melttemp, aes(x=age,y = value,color = variable) )+ geom_line()
pic
```



Antwort: GLM und CatBoost erreichen sehr ähnliche Vorhersagen, die die generelle Struktur der Beobachtungen sehr gut abbilden.

Teil d): Modellanalyse

Machen Sie den Einfluss der Merkmale Einkommen und Beschäftigungsart anhand eines Accumulated Dependence Plots sichtbar. Was ist der Unterschied zu den häufiger verwendeten Partial Dependence Plots? Erstellen Sie einen Partial Dependence Plot für das Merkmal BMI. Vergleichen Sie die Ergebnisse mit denen

des GLM. Welche Auffälligkeit sehen Sie beim BMI und wie ist diese zu begründen und ggf. zu beheben?

Zerlegen Sie die Vorhersage des CatBoost-Modells erneut für eine einzelne Vorhersage des Validierungsdatensatzes mittels des SHAP-Value. Interpretieren Sie das Ergebnis. Diskutieren Sie die Vergleichbarkeit der Ergebnisse mit denen des GLMs.

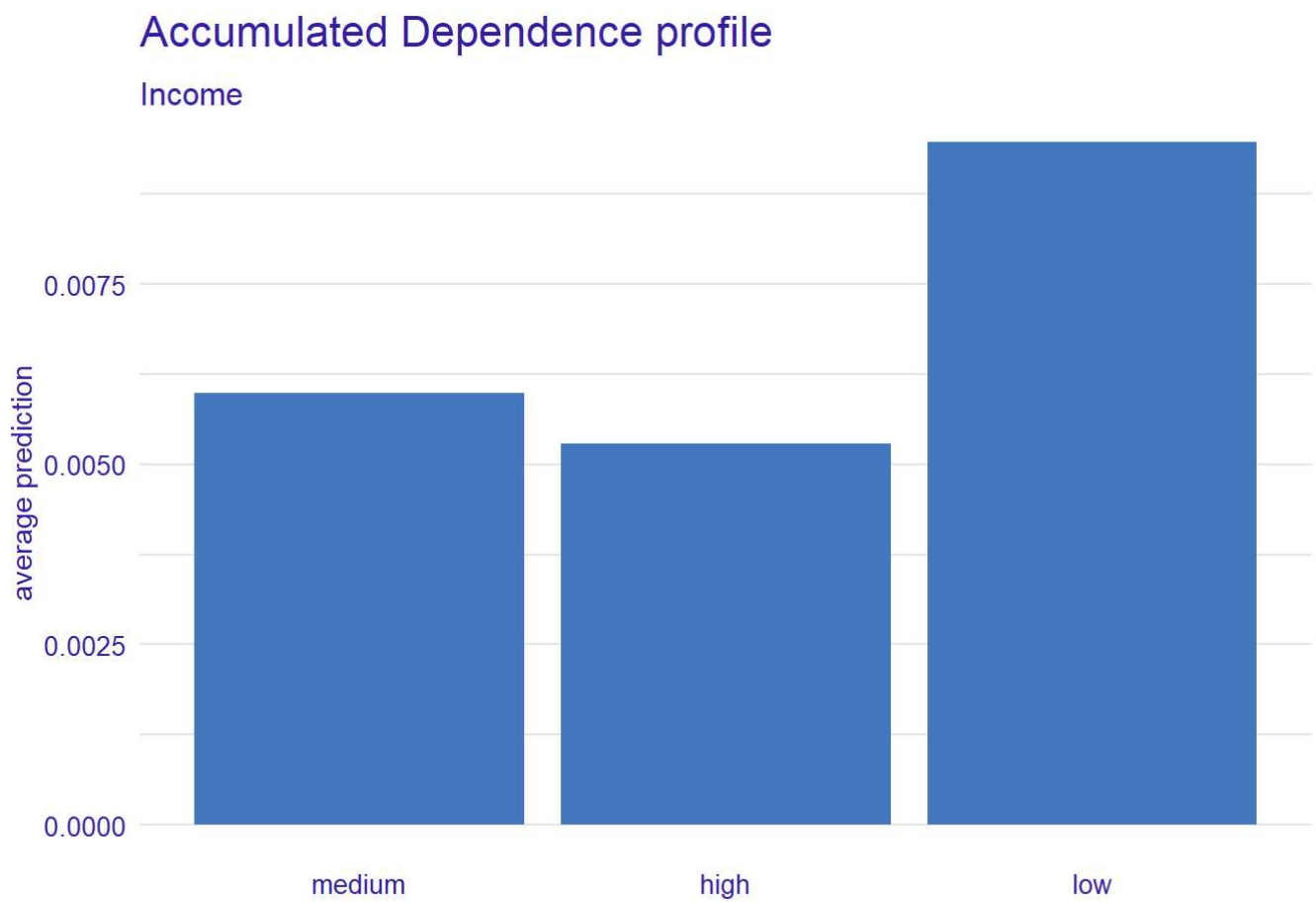
Entwickeln Sie einen SHAP-Force Plot für eine hinreichend große Stichprobe des Validierungsdatensatzes. Gruppieren Sie nach dem Geschlecht. Interpretieren Sie das Ergebnis. Was leiten Sie daraus ab?

Lösungsvorschlag:

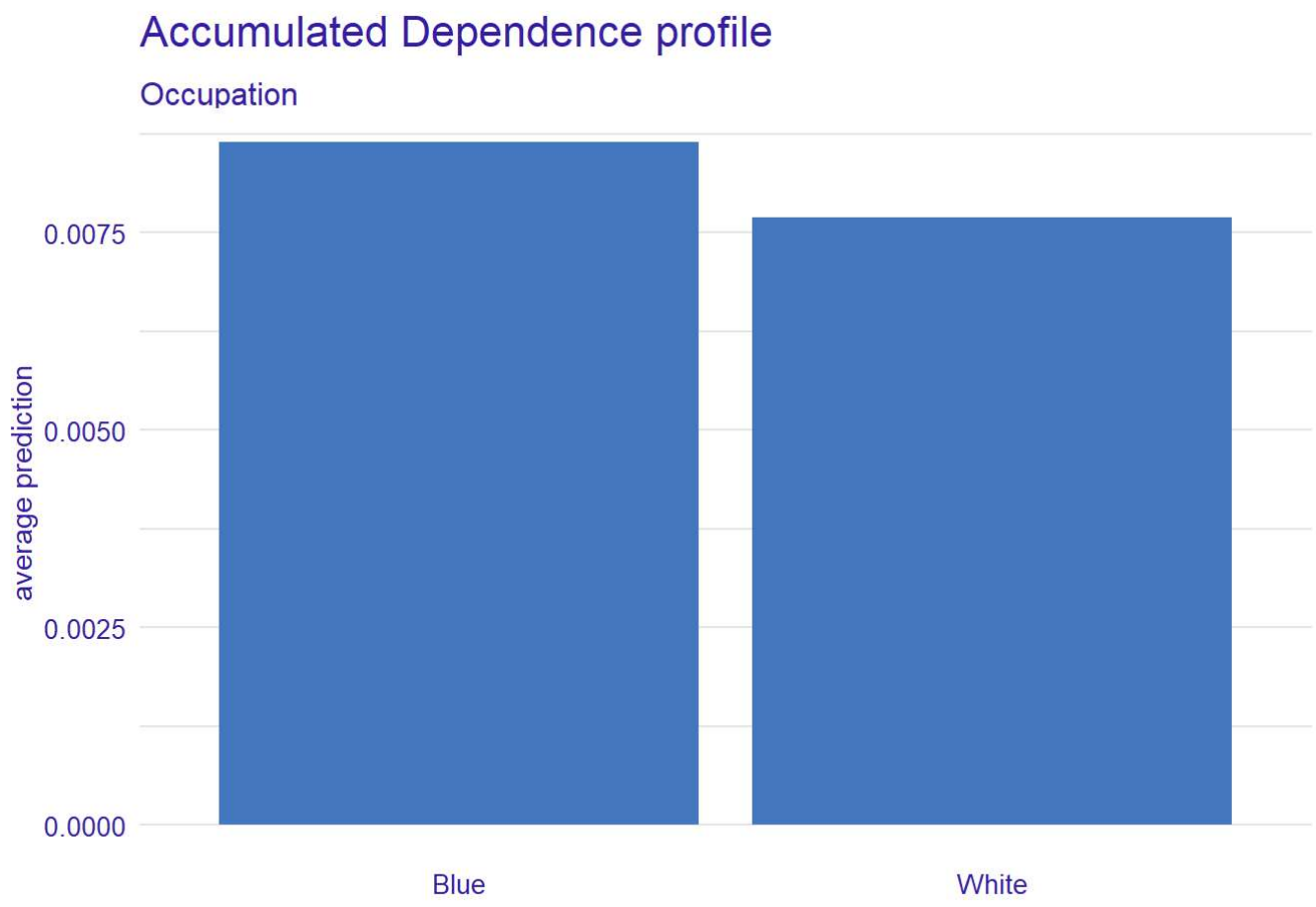
```
#Explainer Object anlegen
catboost_predict <- function(object, newdata) {
  train_data = newdata[,1:(ncol(newdata)-1)]
  offset = matrix(newdata$offset)
  newdata_pool <- catboost.load_pool(data = newdata,
                                     baseline = offset)
  return( exp(catboost.predict(object, newdata_pool)))
}

X_train = cbind(X_train, offset = df_classif$logitoff[training_index])
explainer_cat <- explain(model,
                         data = X_train,
                         y = y_train,
                         predict_function = catboost_predict,
                         label = "CatBoost",
                         colorize = FALSE)
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  CatBoost
##   -> data              :  418109  rows  8  cols
##   -> target variable   :  418109  values
##   -> predict function  :  catboost_predict
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package Model of class: catboost.Model package unrecognized , ve
r. Unknown , task regression (  default  )
##   -> predicted values  :  numerical, min =  0.0001136702 , mean =  0.006025176 , max =  0.
645552
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -0.4800155 , mean =  1.870219e-05 , max =  0.9
997433
##   A new explainer has been created!
```

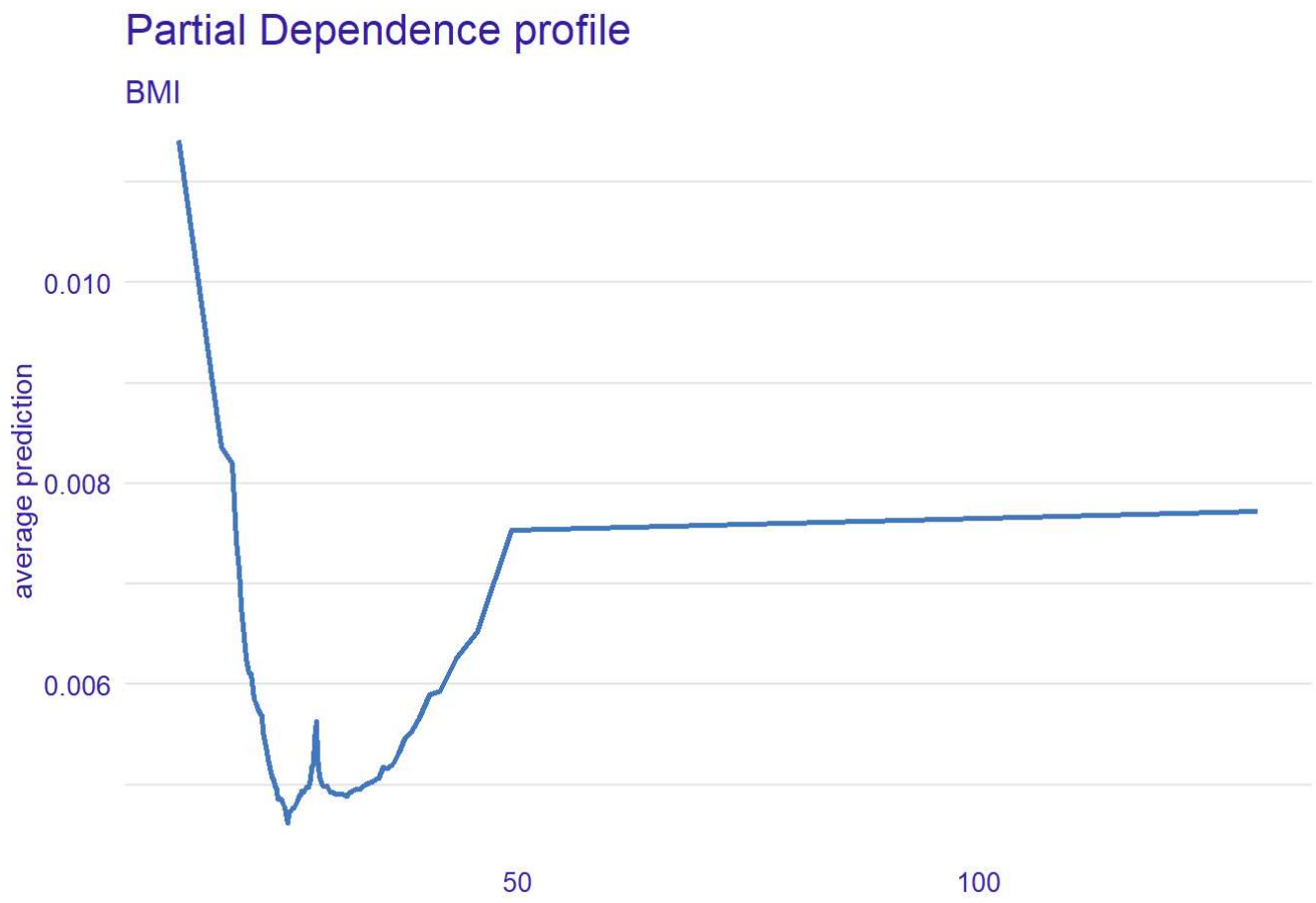
```
ale_cat_inc  <- model_profile(explainer_cat,variable = "Income",
                                            type = "accumulated")
plot(ale_cat_inc,subtitle = "")
```




```
ale_cat_occ  <- model_profile(explainer_cat,variable = "Occupation",
                          type = "accumulated")
plot(ale_cat_occ,subtitle = "")
```




```
pdp_cat  <- model_profile(explainer_cat,variable = "BMI",
                                          type = "partial")
plot(pdp_cat,subtitle = "")
```



Antwort:

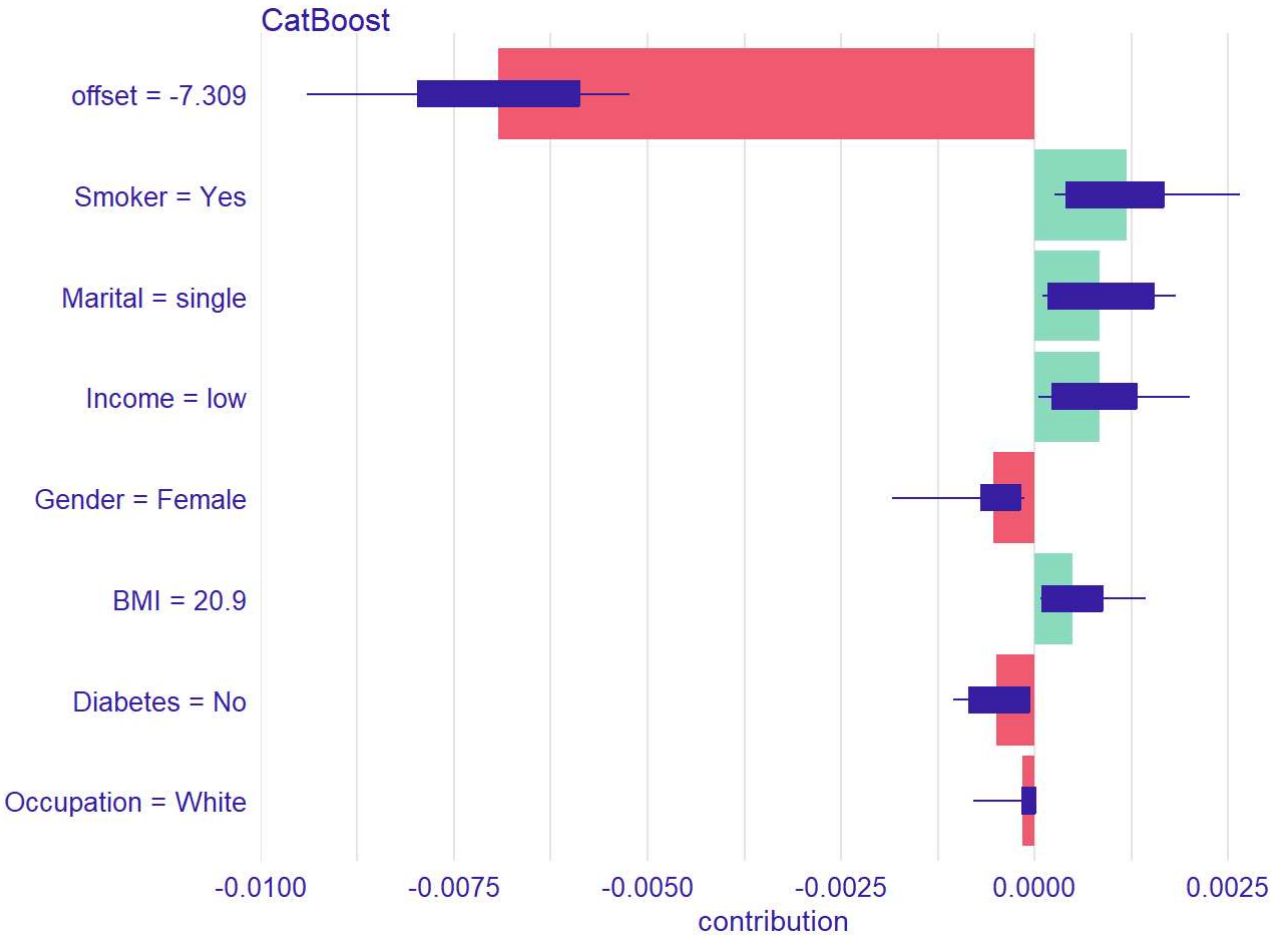
- Für numerische Merkmale kann sehr viel genauer modelliert werden. Für kategoriale Merkmale ist das Ergebnis vergleichbar.
- Die Imputierung des Merkmals BMI hatte einen ungewünschten Effekt. Die Werte für den imputierten Mittelwert (28.22) sind deutlich höher als die umliegenden Werte. Es scheint also eine Systematik vorzuliegen.
- Unterschied ALE vs. PDP: ALE klappt auch gut für Regionen mit spärlicher Besetzung.

Lösungsvorschlag:

```
X_valid = cbind(X_valid, offset = df_classif$logitoff[-training_index])
explainer_cat <- explain(model,
                         data = X_valid,
                         y = y_valid,
                         predict_function = catboost_predict,
                         label = "CatBoost",
                         colorize = FALSE)
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  CatBoost
##   -> data              :  104526  rows  8  cols
##   -> target variable   :  104526  values
##   -> predict function  :  catboost_predict
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package Model of class: catboost.Model package unrecognized , ve
r. Unknown , task regression (  default  )
##   -> predicted values  :  numerical, min =  0.0001166266 , mean =  0.006038146 , max =  0.
607395
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -0.607395 , mean =  -1.370371e-06 , max =  0.9
997527
##   A new explainer has been created!
```

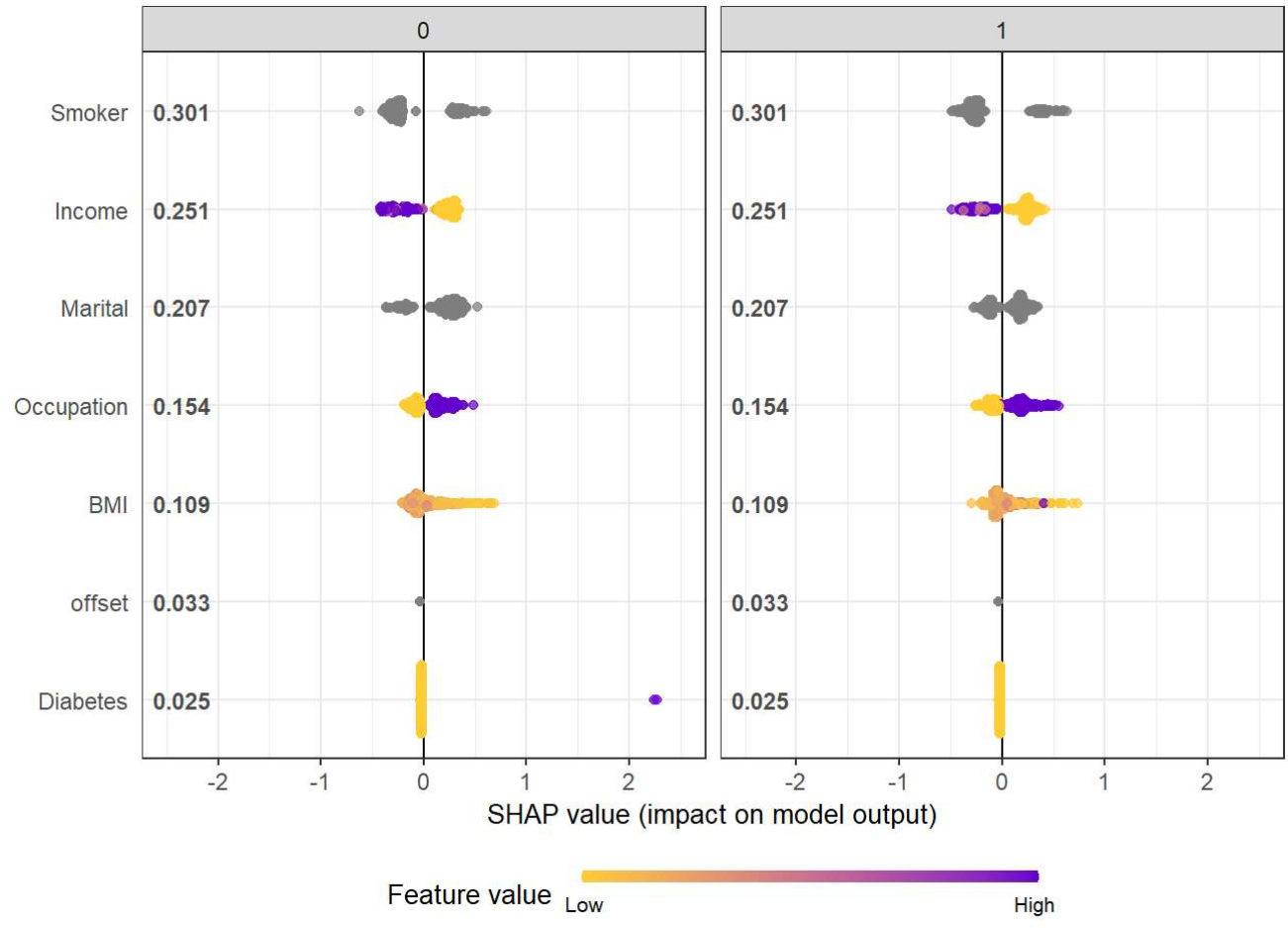
```
#individuelle vorhersagen
nobs = X_valid[1,]
sp_cat <- predict_parts(explainer_cat,
                        new_observation = nobs,
                        type = "shap")
plot(sp_cat)
```



Interpretation: SHAP Values haben einen anderen Fokus und können nicht direkt mit den Ergebnissen des GLM verglichen werden.

Lösungsvorschlag:

```
shapvals = catboost.get_feature_importance(model, pool = test_pool, type = "ShapValues")

#numerische Umkodierung nötig für Plot - wie genau ist aber irrelevant.
X_num = X_valid %>% select(BMI,offset)
X_num %<>% mutate(Income = ifelse(X_valid$Income=="low",0,ifelse(X_train$Income=="medium",1,
2)))
X_num %<>% mutate(Marital = ifelse(X_valid$Marital == "alleinstehend",0,1),
                  Smoker = ifelse(X_valid$Smoker == "Nein",0,1),
                  Gender = ifelse(X_valid$Gender == "Male",0,1),
                  Occupation = ifelse(X_valid$Occupation == "White",0,1),
                  Diabetes = ifelse(X_valid$Diabetes == "No",0,1))

shapvalue <- data.frame(shapvals)
names(shapvalue) = names(X_valid)


shap_long_game <- shap.prep(shap_contrib = shapvalue[1:1000,],
                            X_train = X_num[1:1000,])

shap_summplot <- shap.plot.summary(shap_long_game, scientific = F)

shap_long_game <- shap.prep(shap_contrib = shapvalue[1:1000,],
                            X_train = X_num[1:1000,],
                            var_cat = "Gender")

shap.plot.summary(shap_long_game, scientific = F) +
  ggplot2::facet_wrap(~ Gender)
```



Interpretation: Aus diesem Plot können Interaktionen identifiziert werden. Beispiele sind: höhere Streuung beim Heiratsstatus bei Männern oder geringere Streuung beim Angestelltenverhältnis können Hinweise für Interaktionen sein.

Aufgabe 4

Verwenden und übertragen Sie das erstellte CatBoost/GLM-Modell für das Pricing in Deutschland. Ermitteln Sie den Preis für einen geschlossenen bAV-Bestand (bav_Bestand.csv) mit einjähriger Risikoleben inkl. pauschal 5 EUR Stückkosten und Sicherheitszuschlag mit einer Versicherungssumme von 100.000 EUR. D.h. der Preis einer einzelnen Police i beträgt: P(i) = 5EUR +qx(i)*100.000EUR. Verwenden Sie die DAV2008T 2. Ordnung mit einem Mischungsverhältnis von Mann:Frau = 60:40. Welchen Preis stellen Sie dem Unternehmen in Rechnung? Welchen Akademikerbonus rechnen Sie dabei ein?

Lösungsvorschlag:

```
bav = fread("bav_Bestand.csv", sep = ";", dec = ",",stringsAsFactors = T,na.strings = c("","<
NA>"))

bav %<>% rename(Income=Einkommen,
                Marital=Heiratsstatus,
                Smoker=Raucherstatus,
                Gender=Geschlecht,
                Occupation= Berufsgruppe,
                age = Alter)

bav %<>% mutate(Marital = fct_na_value_to_level(Marital, "(Missing)"))

bav %<>% mutate(Income =  fct_recode(Income,high = "hoch",
                                     medium = "mittel",
                                     low = "gering"),
                Gender = fct_recode(Gender, Male = "Mann",Female ="Frau"),
                Occupation = fct_recode(Occupation, White = "Akademiker",Blue ="Nichtakademik
er"),
                Diabetes = fct_recode(Diabetes, Yes = "Ja",No ="Nein"),
                Smoker = fct_recode(Smoker, Yes = "Ja",No ="Nein"),
                Marital = fct_recode(Marital, single = "alleinstehend",
                                     married = "in Lebenspartnerschaft",
                                     married ="(Missing)"))

# baseline vorgeben:

dav2008TM = read_excel("D:/DAV/ADS Completion/DAV 2008 T mit R-NR.xlsx",sheet = "DAV 2008 T M
änner",skip = 2) %>% select("...1", "Sterblichkeit 2. Ordnung")
dav2008TF = read_excel("D:/DAV/ADS Completion/DAV 2008 T mit R-NR.xlsx",sheet = "DAV 2008 T F
rauen",skip = 2) %>% select("...1", "Sterblichkeit 2. Ordnung")

dav2008T = dav2008TM
dav2008T %<>% rename(Alter = "...1",qxM = "Sterblichkeit 2. Ordnung")
dav2008T %<>% left_join(dav2008TF,by=c("Alter" = "...1"))
dav2008T %<>% rename(qxF = "Sterblichkeit 2. Ordnung")
dav2008T %<>% mutate(mixed = 0.6*qxM + 0.4*qxF,
                     mixedmultiplied = 1.1* mixed + 0.003 *(Alter>=50))

bav %<>% left_join(dav2008T %>% select(Alter,mixed,mixedmultiplied), by = c("age"="Alter"))

#da es sich um einen Versicherungsbestand handelt, kann direkt mixed verwendet werden.
bav %<>% mutate(logitoff = log(mixed/(1-mixed)))

qx = predict(m6,bav,type = "response")

Price = sum(qx * 100000 + 5)
```

Der Preis zur Absicherung des Portfolios beträgt: 2.71 Mio EUR. Dabei wird -0.096 als Akademikerbonus verwendet.