

Projektarbeit für Algorithmik & Statistik für Data Science 2 Lab

Black Friday - eine statistische Analyse der Verkaufszahlen

Christoph Rabensteiner: 1810837995

Valentin Muhr: 1810837102

Magdalena Breu: 1810837995

Jochen Paul Hollich: 1810837475

16.06.2019

Einleitung

Beschreibung des Datensatzes

Der Datensatz Black Friday enthält Daten über den Warenkorb von Kunden bei einem Onlineshop. Im Datensatz befinden sich 537577 Beobachtungen für 12 Variablen, u.a. Geschlecht, Alter, Beruf, Beziehungsstatus und Kaufkraft. Die Variablen Age, Occupation, City_Category, Stay_In_Current_City_Years Product_Categories sind als Faktoren codiert. Die genauen Definitionen der einzelnen Faktoren (z.B. City_Category A,B & C) sind nicht genau definiert. Dafür schauen wir uns alle Variablen genauer an.

Hintergrundinformationen zum Datensatz

Der Datensatz wurde bereitgestellt durch einen Hackathon der Firma Analytics Vidhya (<https://datahack.analyticsvidhya.com/contest/black-friday/>) und heruntergeladen über Kaggle.com (<https://www.kaggle.com/mehdidag/black-friday>).

Aufbau dieser Arbeit

Unser Ziel ist es das Konsumverhalten der Kunden genauer zu analysieren. Dafür schauen wir uns als erstes die Daten genauer an und versuchen anhand der erstellten Plots, die Zusammenhänge zwischen den einzelnen Variablen zu erkennen.

Auf diesem Wissen aufbauend werden wir verschiedene ausgewählte Modelle berechnen und analysieren. Mit dem Modellen möchten wir die Ausgaben der Kunden anhand der Prädiktoren vorhersagen können.

```
library(dplyr)
library(ggplot2)
library(tidyverse)
library(caret)
library(neuralnet)
library(dplyr)
library(tidyr)
library(scales)
library(grid)
library(gridExtra)
library(plotly)
library(RColorBrewer)
library(fastDummies)
library(ggcormplot)
library(reshape2)
```

Laden der Daten

```
set.seed(10091212)
bf <- read.csv2("BlackFriday.csv", dec=". ", header=TRUE, sep=",")
```

Zusammenfassung der Daten:

```
summary(bf)
```

```
##      User_ID          Product_ID    Gender     Age
##  Min.   :1000001   P00265242: 1858   F:132197   0-17 : 14707
##  1st Qu.:1001495   P00110742: 1591   M:405380   18-25: 97634
##  Median :1003031   P00025442: 1586           26-35:214690
##  Mean   :1002992   P00112142: 1539           36-45:107499
##  3rd Qu.:1004417   P00057642: 1430           46-50: 44526
##  Max.   :1006040   P00184942: 1424           51-55: 37618
##                  (Other)  :528149           55+   : 20903
##      Occupation    City_Category Stay_In_Current_City_Years
##  Min.   : 0.000   A:144638      0 : 72725
##  1st Qu.: 2.000   B:226493      1 :189192
##  Median : 7.000   C:166446      2 : 99459
##  Mean   : 8.083           3 : 93312
##  3rd Qu.:14.000           4+: 82889
##  Max.   :20.000
##
##      Marital_Status Product_Category_1 Product_Category_2 Product_Category_3
##  Min.   :0.0000   Min.   : 1.000   Min.   : 2.00   Min.   : 3.0
##  1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.: 5.00   1st Qu.: 9.0
##  Median :0.0000   Median : 5.000   Median : 9.00   Median :14.0
##  Mean   :0.4088   Mean   : 5.296   Mean   : 9.84   Mean   :12.7
##  3rd Qu.:1.0000   3rd Qu.: 8.000   3rd Qu.:15.00   3rd Qu.:16.0
##  Max.   :1.0000   Max.   :18.000   Max.   :18.00   Max.   :18.0
##                  NA's   :166986   NA's   :NA's       NA's   :373299
##
##      Purchase
##  Min.   : 185
##  1st Qu.: 5866
##  Median : 8062
##  Mean   : 9334
##  3rd Qu.:12073
##  Max.   :23961
##
```

Struktur der Daten

```
str(bf)
```

```
## 'data.frame': 537577 obs. of 12 variables:
## $ User_ID                      : int 1000001 1000001 1000001 1000001 1000002 ...
## $ Product_ID                    : Factor w/ 3623 levels "P00000142","P00000242",...
## $ Gender                        : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 2 2 2 ...
## $ Age                           : Factor w/ 7 levels "0-17","18-25",...
## $ Occupation                    : int 10 10 10 10 16 15 7 7 7 20 ...
## $ City_Category                 : Factor w/ 3 levels "A","B","C": 1 1 1 1 3 1 2 2 2 1 ...
## $ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",...
## $ Marital_Status                : int 0 0 0 0 0 1 1 1 1 ...
## $ Product_Category_1            : int 3 1 12 12 8 1 1 1 1 8 ...
```

```

## $ Product_Category_2      : int NA 6 NA 14 NA 2 8 15 16 NA ...
## $ Product_Category_3      : int NA 14 NA NA NA NA 17 NA NA NA ...
## $ Purchase                : int 8370 15200 1422 1057 7969 15227 19215 15854 15686 7871 ...

```

Beschreibung der Daten:

In der Tabelle gibt es insgesamt 12 verschiedene Variablen, die folgend erklärt werden:

User_ID: eindeutige ID eines Kunden

Product_ID: eindeutige ID eines Produktes

Gender: Geschlecht des Kunden

Age: Altersgruppe des Kunden

Occupation: Beschäftigungsgruppe des Kunden

City_Category: Wohnort des Kunden

Stay_In_Current_City_Years: Anzahl der Jahre seitdem der Kunde in der Stadt wohnt

Marital_Status: Ehestatus

Product_Category_1: Produktkategorie des Einkaufs

Product_Category_2: Produkt könnte dieser Kategorie angehören

Product_Category_3: Produkt könnte dieser Kategorie angehören

Purchase: Summe des Einkaufs

Bereinigung der Daten

Im Folgenden überprüfen wir die Anzahl der NAs je Feature.

```

## [1] 0 0 0 0 0 0 0 0 0 0 0
## [11] 166986 373299 0

```

Bei der Überprüfung der NAs fällt uns auf, dass wir nur in der Product Category 2 & 3 NAs-Werte enthalten haben. Wir gehen davon aus, dass von dieser Kategorie nichts gekauft wurde und ersetzen die Werte mit 0.

Die NAs-Werte werden mit 0 ersetzt. Zusätzlich werden die Produktkategorien von Numeric in Integer umgewandelt:

```

bf[is.na(bf)] <- 0
bf$Product_Category_3 <- as.integer(bf$Product_Category_3)
bf$Product_Category_2 <- as.integer(bf$Product_Category_2)

```

Ausgabe der neuen Datenstruktur ohne die NULL-Werte:

```

## 'data.frame': 537577 obs. of 12 variables:
## $ User_ID                  : int 1000001 1000001 1000001 1000002 1000003 1000004 1000004 ...
## $ Product_ID                : Factor w/ 3623 levels "P00000142","P00000242",...: 671 2375 851 827 273 ...
## $ Gender                   : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 2 2 2 2 ...
## $ Age                      : Factor w/ 7 levels "0-17","18-25",...: 1 1 1 1 7 3 5 5 5 3 ...
## $ Occupation               : int 10 10 10 10 16 15 7 7 7 20 ...
## $ City_Category             : Factor w/ 3 levels "A","B","C": 1 1 1 1 3 1 2 2 2 1 ...
## $ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",...: 3 3 3 3 5 4 3 3 3 2 ...
## $ Marital_Status            : int 0 0 0 0 0 1 1 1 1 ...
## $ Product_Category_1        : int 3 1 12 12 8 1 1 1 8 ...
## $ Product_Category_2        : int 0 6 0 14 0 2 8 15 16 0 ...

```

```
## $ Product_Category_3      : int  0 14 0 0 0 0 17 0 0 0 ...
## $ Purchase                 : int  8370 15200 1422 1057 7969 15227 19215 15854 15686 7871 ...
```

Deskriptive Analyse - Die Erkundung unserer Daten.

Bevor wir mit dem Modellieren beginnen, wollen wir unsere Daten genauer betrachten und verschiedenste Analysen aufstellen.

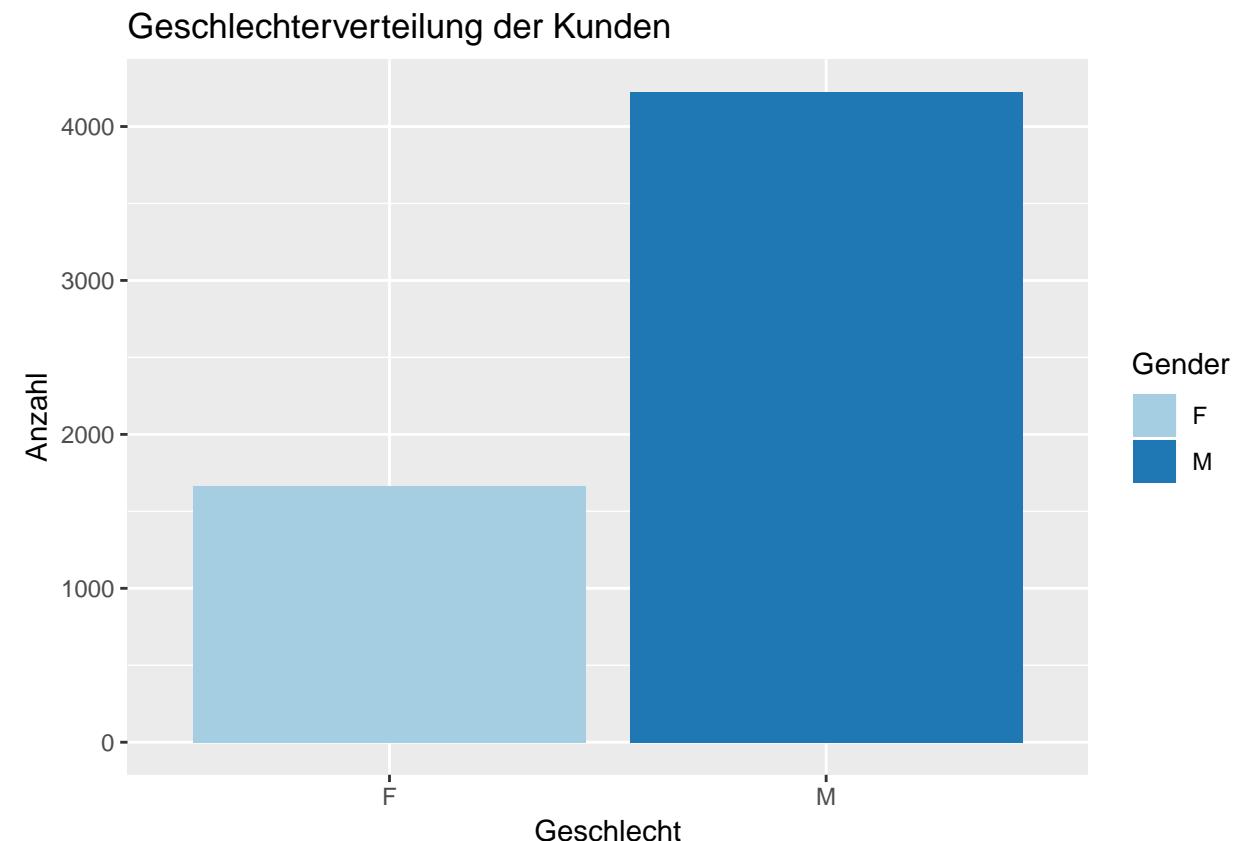
Häufigkeiten von User_ID, Product_ID und Occupation:

Anzahl eindeutiger Variablen	
User_ID	5891
Product_ID	3623
Occupation	21

Bei der Ausgabe der Datenstruktur sehen wir, dass die USER_ID mehrmals vorkommt, d.h. ein Kunde hat auch mehrfach Einkäufe getätigt. Insgesamt haben wir zwar 537.577 verschiedene Einkäufe. Diese wurden aber von 5.891 verschiedenen Kunden getätigt. Zudem wurden insgesamt 3.623 verschiedene Artikel gekauft. Die Kunden sind in insgesamt 21 verschiedenen Berufsgruppen zugeordnet. Wir analysieren deshalb später, welche Berufsgruppe die Konsumfreudigsten sind.

Geschlecht

Im Folgenden sehen wir uns die Verteilung der Geschlechter genauer an.



Summe der Geschlechter:

```
##      F      M
## 1666 4225
```

Wir haben insgesamt 1.666 weibliche und 4.225 männliche Kunden. Da wir weitaus mehr männliche Kunden haben, rückschließen wir, dass das Geschäft für männliche Käufer ausgelegt bzw. attraktiver gestaltet wurde und weibliche Konsumenten weniger anspricht. Gleichzeitig können wir aber auch sagen, dass wir einen unbalanzierten Datensatz haben, da die männlichen Konsumenten 3-mal öfters vorkommen als die weiblichen.

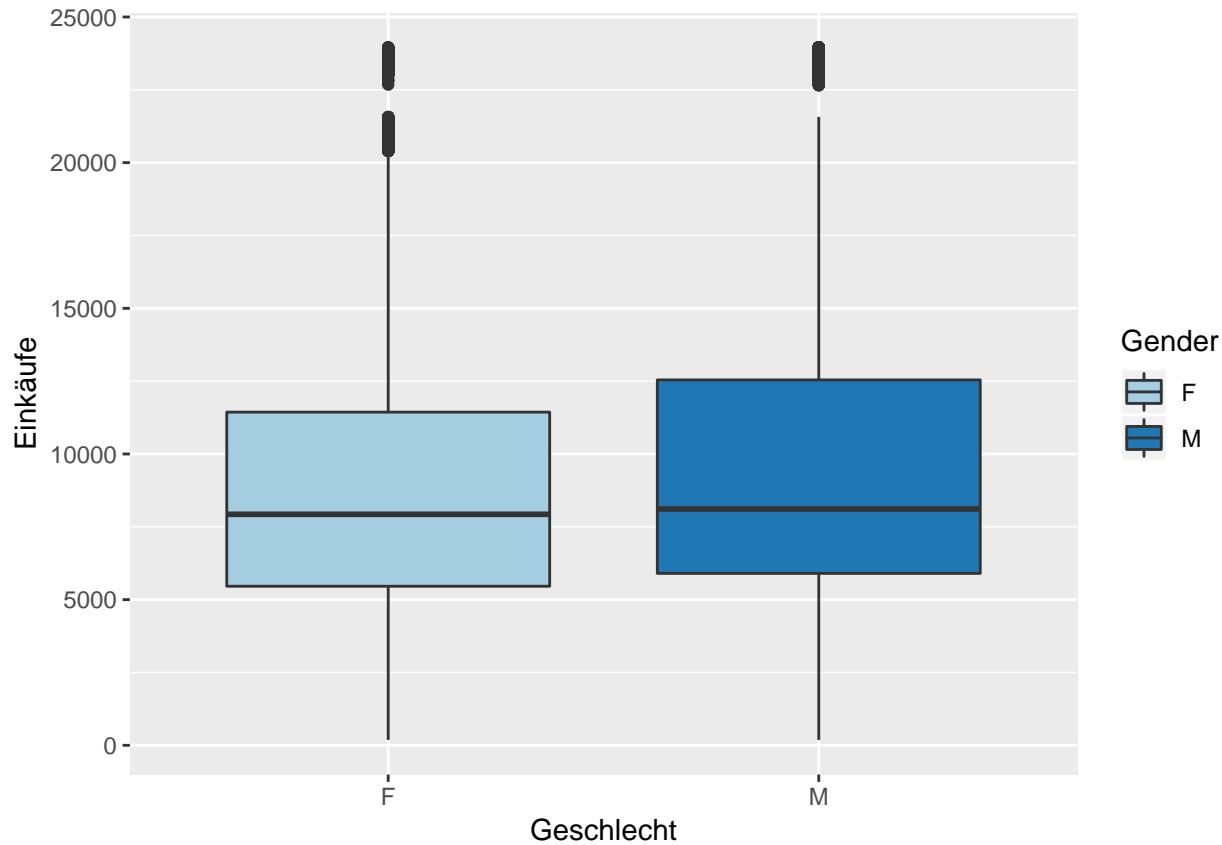
Wir interessieren uns auch dafür, wie die Verteilung der Warenkörbe zwischen den Geschlechten aussieht. Dafür gruppieren wir zuerst alle Einkäufe pro Kunden und geben die Summe der Einkäufe pro Geschlecht wieder:

```
Purchase_Male <- subset(bf, bf$Gender=='M')
Purchase_Female <- subset(bf, bf$Gender=='F')
count_male = length(unique(Purchase_Male$User_ID))
count_female = length(unique(Purchase_Female$User_ID))
average_male=sum(Purchase_Male$Purchase)/count_male
average_female=sum(Purchase_Female$Purchase)/count_female
gender_split = data.frame(c("Male", "Female"),
                           c(sum(Purchase_Male$Purchase), sum(Purchase_Female$Purchase)),
                           c(count_male, count_female),
                           c(average_male, average_female))

colnames(gender_split) <- c("Gender", "Total Purchase", "Count", "Average")
grid.arrange(tableGrob(gender_split))
```

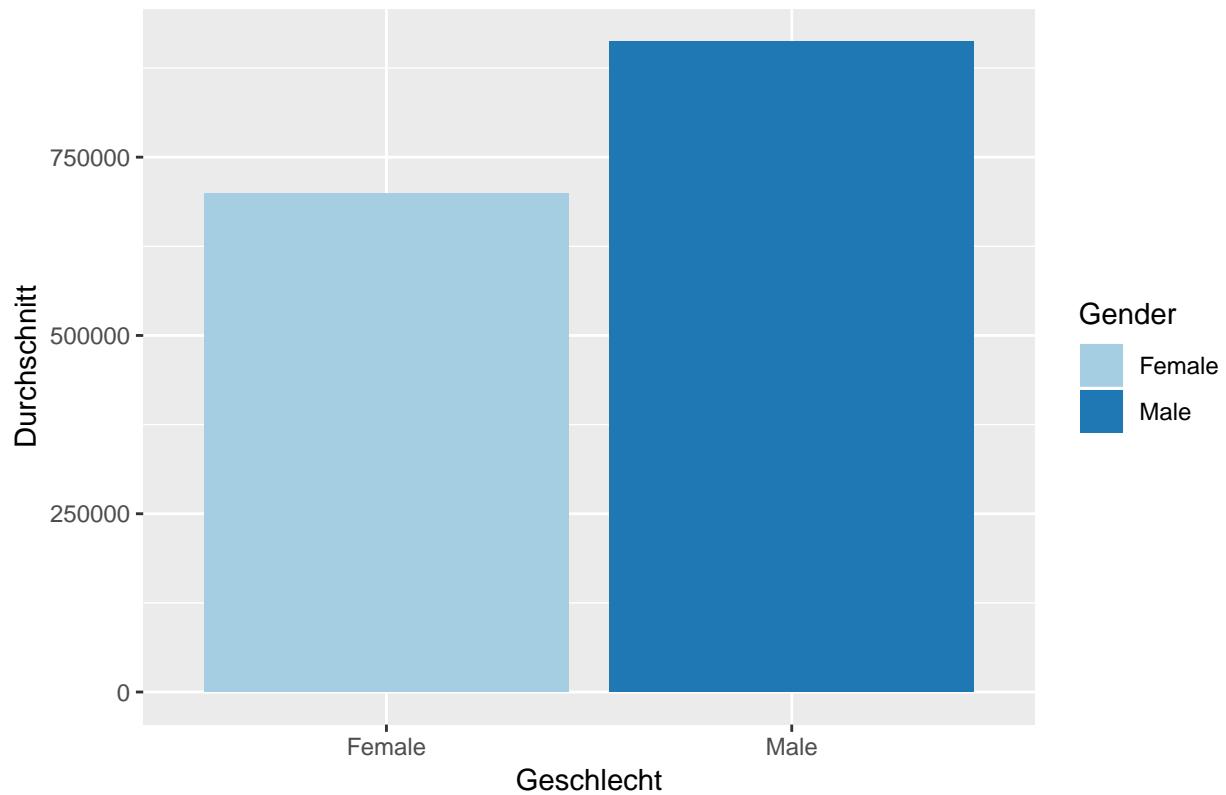
Gender	Total Purchase	Count	Average
1 Male	3853044357	4225	911963.2
2 Female	1164624021	1666	699054.0

Boxplot der Einkäufe nach Geschlecht:



In unserem Boxplot erkennen wir, dass es Ausreißer nach oben gibt. Allgemein gilt es sich zu überlegen, ob Ausreißer aus dem Datensatz entfernt werden sollen oder nicht. Wir entscheiden uns die Ausreißer nicht zu entfernen, da wir annehmen, dass an einem Black Friday Event Ausreißer sehr wohl vorkommen können. Die Verteilung der Einkäufe nach Geschlecht ist ungefähr gleich.

Durchschnittle Ausgaben je Geschlecht



Im Druchschnitt geben männliche Kunden mehr als weibliche Kunden aus. Es kaufen weniger Frauen im Shop ein als Männer und sie kaufen im Verhältnis pro Einkauf weniger als die männlichen Kunden. Im Druchschnitt geben Männer 911.963,2 aus, das sind im Schnitt 30% mehr als bei den Frauen. Die Währung ist uns nicht bekannt. Auf das Konto der Männer gehen insgesamt 4255 von insgesamt 5891 Einkäufen, das entspricht 72% aller Einkäufe.

Produkte

Auch die Produkte sehen wir uns genauer an. Als erstes geben wir die zehn häufigsten gekauften Produkte aus:

```
bestverkaufte_Produkte = bf %>%
  count(Product_ID, sort=T)
grid.arrange(tableGrob(head(bestverkaufte_Produkte, 10)))
```

	Product_ID	n
1	P00265242	1858
2	P00110742	1591
3	P00025442	1586
4	P00112142	1539
5	P00057642	1430
6	P00184942	1424
7	P00046742	1417
8	P00058042	1396
9	P00059442	1384
10	P00145042	1384

Das am meisten verkaufte Produkt ist P00265242 und zwar mit einer Anzahl von insgesamt 1.858 Stück.

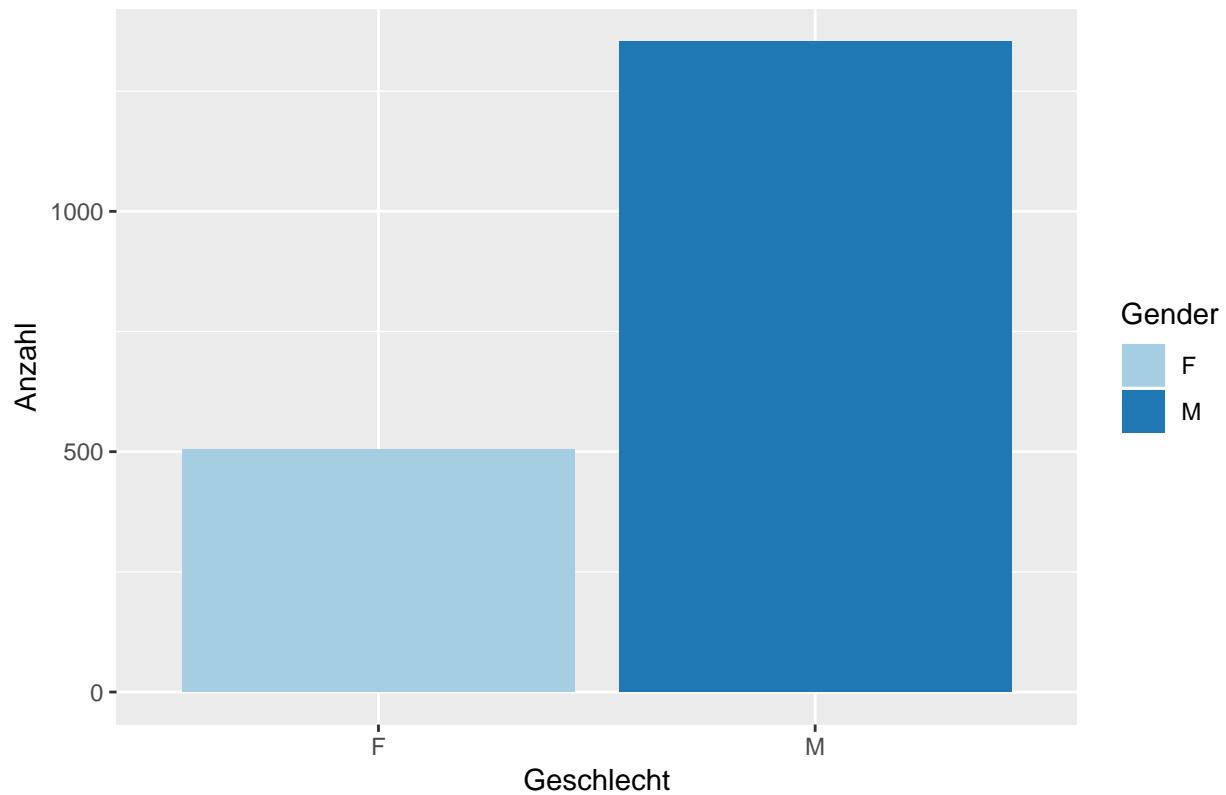
```
bestes_Prodkt = bf[bf$Product_ID=='P00265242',]
head(bestes_Prodkt)
```

```
##      User_ID Product_ID Gender   Age Occupation City_Category
## 400 1000066  P00265242     M 26-35       18          C
## 1192 1000196  P00265242     F 36-45        9          C
## 1373 1000222  P00265242     M 26-35        1          A
## 1846 1000301  P00265242     M 18-25        4          B
## 2210 1000345  P00265242     M 26-35       12          A
## 2405 1000383  P00265242     F 26-35        7          A
##      Stay_In_Current_City_Years Marital_Status Product_Category_1
## 400                      2                  0          5
## 1192                     4+                 0          5
## 1373                      1                  0          5
## 1846                     4+                 0          5
## 2210                      2                  1          5
## 2405                     4+                 1          5
##      Product_Category_2 Product_Category_3 Purchase
## 400                      8                  0     8652
## 1192                     8                  0     8767
## 1373                     8                  0     6944
## 1846                     8                  0     8628
## 2210                     8                  0     8593
## 2405                     8                  0     6998
```

In der Tabelle wird ersichtlich, dass das Produkt in die Product_Category_1 = 5 und in Product_Category_2=8 fällt. Die Bedeutung der Nummer in den Produktkategorie Spalten ist nicht bekannt. Da dieselbe Produktnummer in verschiedene Produktkategorien fällt, nehmen wir an, dass die Produkt_ID nicht für das Produkt einzigartig ist. Es fällt auf, dass es für dasselbe Produkt unterschiedliche Preise bezahlt wurden.

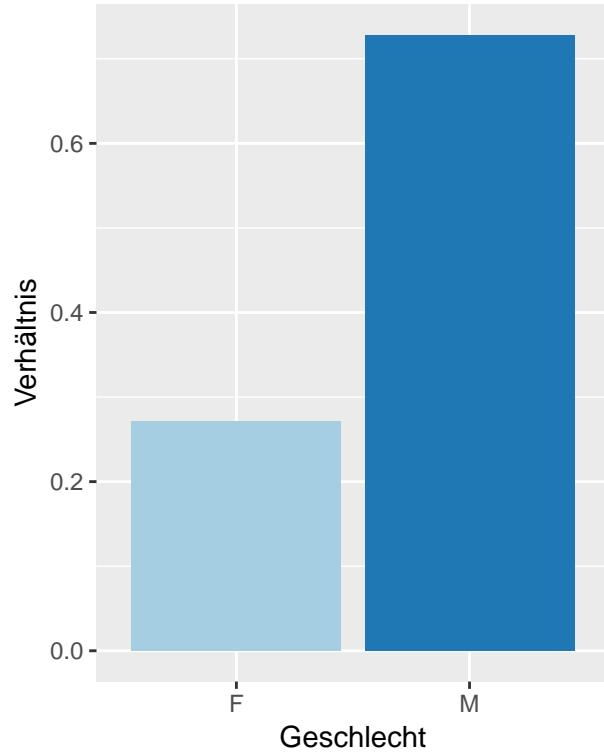
Im weiteren betrachten wir, ob es einen Zusammenhang zum bestverkauften Produkt und den Geschlecht gibt:

Verteilung des Geschlechts am meistverkauften Produkt

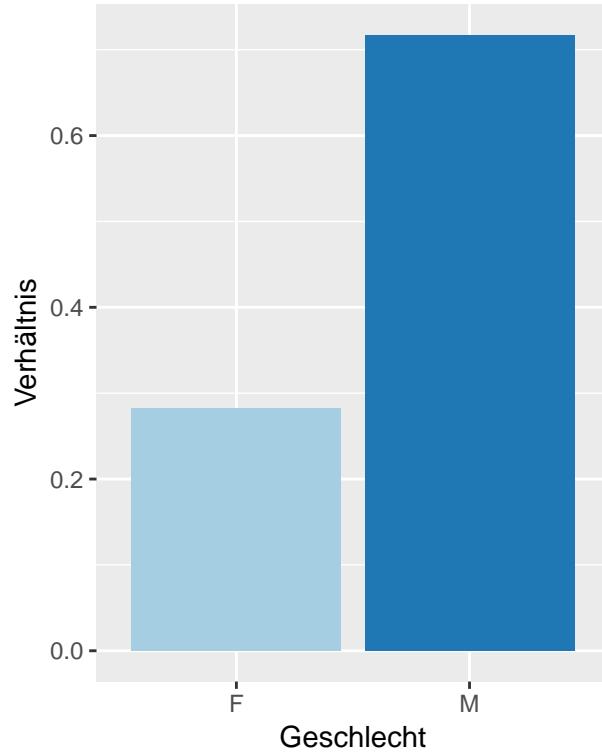


Es fällt auf, dass die Verteilung der Geschlechter am meistverkauften Produkt ähnlich aussieht, wie die Verteilung der Geschlechter auf alle Produkte. Deshalb stellen wir die Verteilungen nochmal gegenüber:

Geschlechterverteilung
– bestes Produkt



Geschlechterverteilung
– aller Produkte



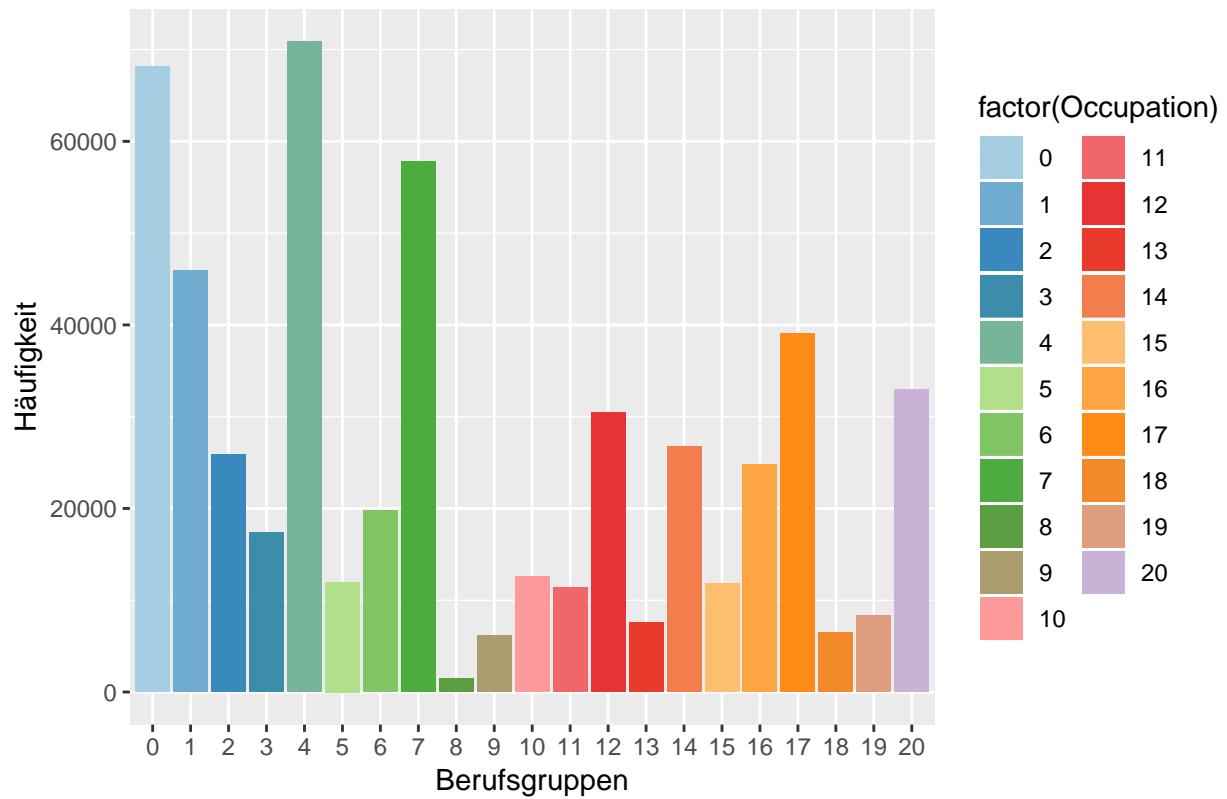
Wir sehen bei der Gegenüberstellung sofort, dass die Verteilung der Geschlechter am bestverkauften Produkt gegenüber alle Produkte sehr ähnlich sind. Daraus schließen wir, dass das bestverkaufte Produkt keinem Geschlecht mehr anspricht. Das Verhältnis der Geschlechter beim meistverkauften und über alle Produkte liegt bei 25% für weiblich und 75% für männlich.

Berufsgruppe

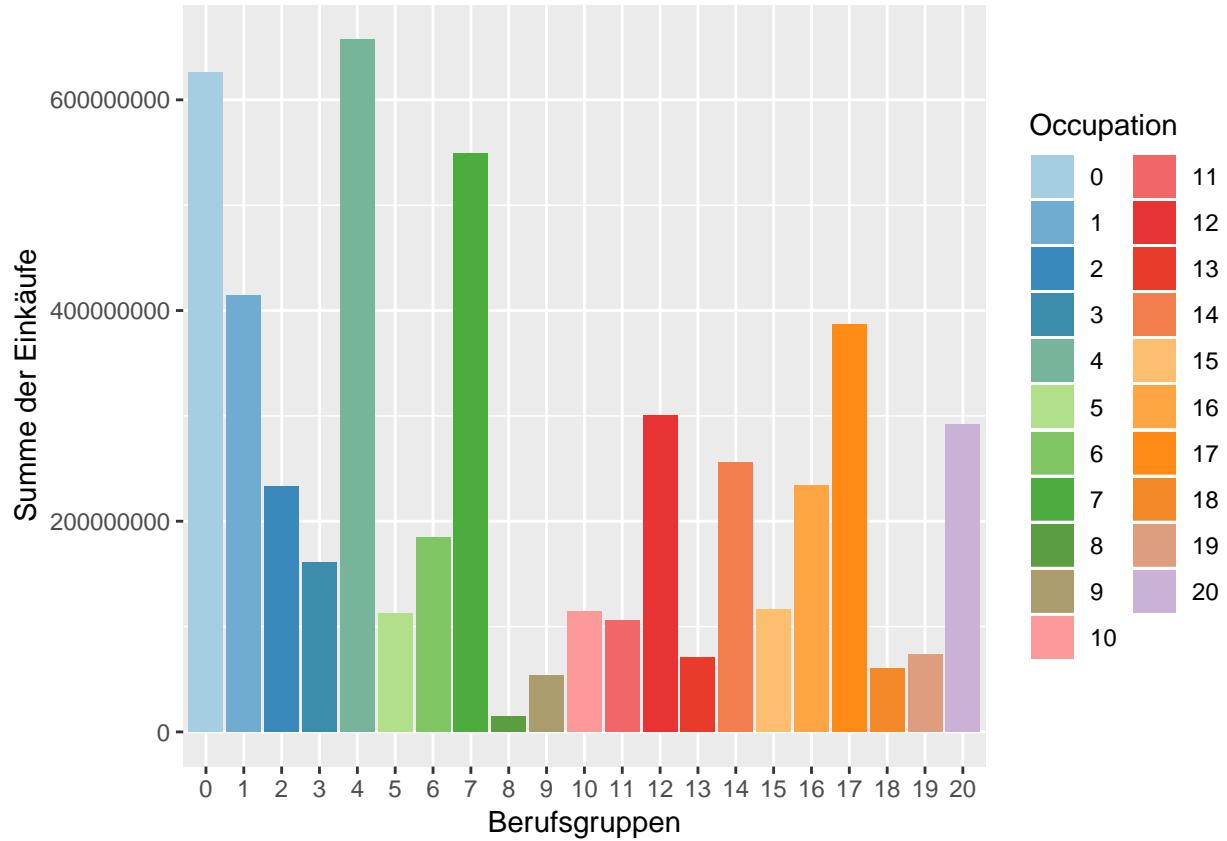
Im Histogramm sehen wir, dass die erste Berufsgruppe am Häufigsten vorkommt, gefolgt von der vierten und siebten Berufsgruppen:

```
## Warning: Ignoring unknown aesthetics: binwidth
```

Histogramm der Berufsgruppen



Wir interessieren uns für die Summe der Einkäufe der einzelnen Berufsgruppen:



Die Summe der Einkäufe ist bei der Berufsgruppe 0, 4 und 7 am höchsten.

Wir interessieren uns dafür, wie die einzelne Verteilung der Produkt Kategorien zwischen männlichen und weiblichen Kunden aussieht, bzw ob Kategorie 2 und 3 von Männern oder Frauen öfters gekauft wurde.

	Kategorie1	Kategorie2	Kategorie3
Male	498	650	383
Female	443	539	276

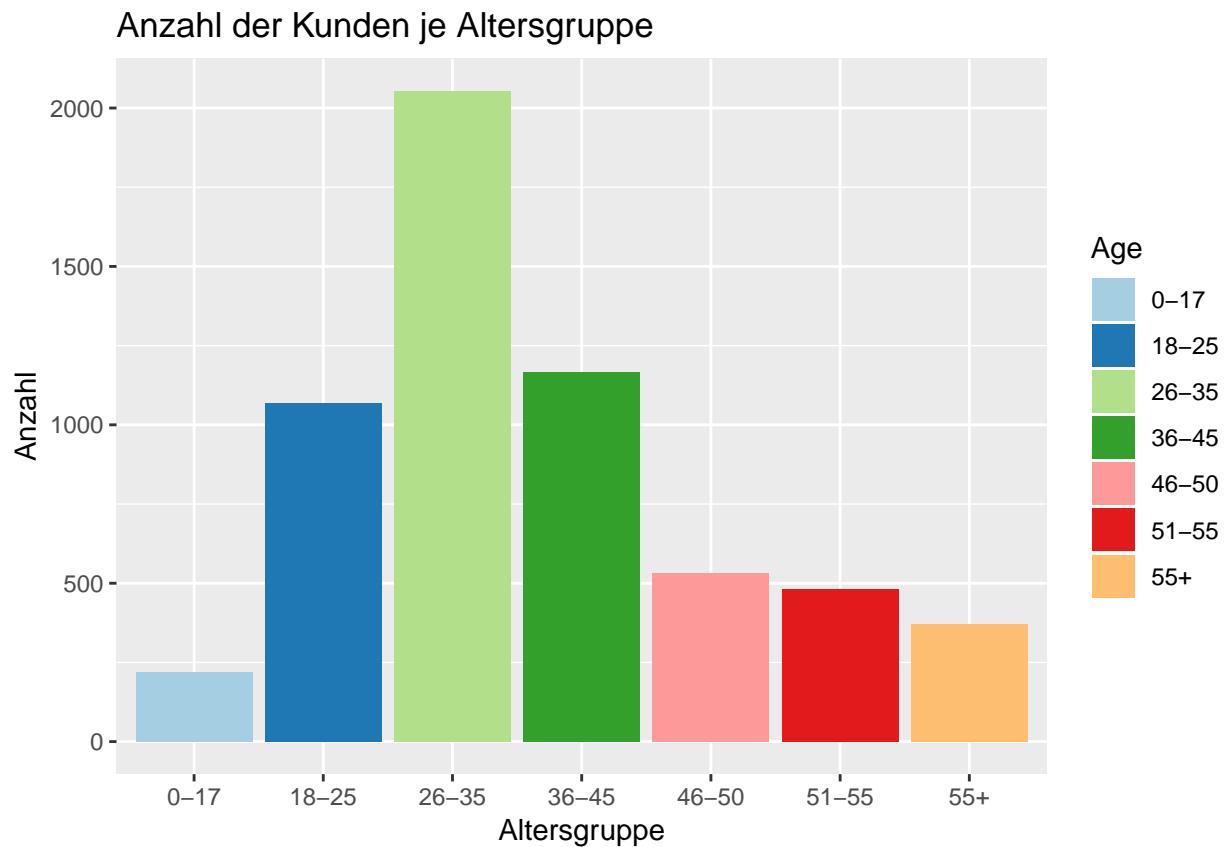
Im Durchschnitt kaufen Männer in allen 3 Kategorien mehr ein (jeweils 12%, 20%, 38% mehr). Wir interpretieren daraus, dass die Produkte generell mehr männliche Kunden anspricht. Speziell Kategorie 2 und 3 wurde gehäuft von Männern eingekauft und könnte ein Indiz dafür sein, dass die Zielgruppe für Kat 2 und 3 eher die Männer sind.

Alter

Als nächsten betrachten wir das Alter unserer Kunden in einer Tabelle:

	Age	n
1	0–17	218
2	18–25	1069
3	26–35	2053
4	36–45	1167
5	46–50	531
6	51–55	481
7	55+	372

Geplottet sieht die Verteilung folgendermaßen aus:

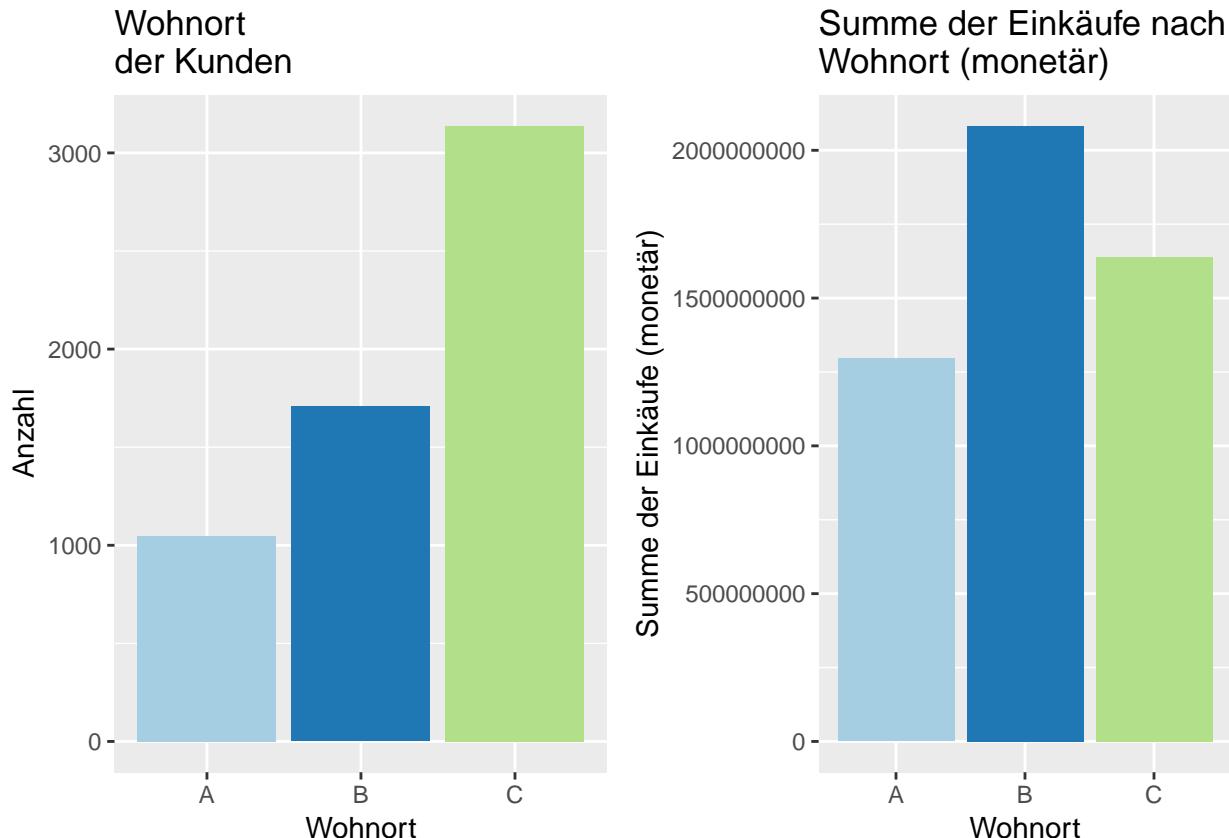


In der Grafik wird ersichtlich, dass die meisten Käufer zwischen 26–35 und 36–45 Jahre alt sind.

Wohnort

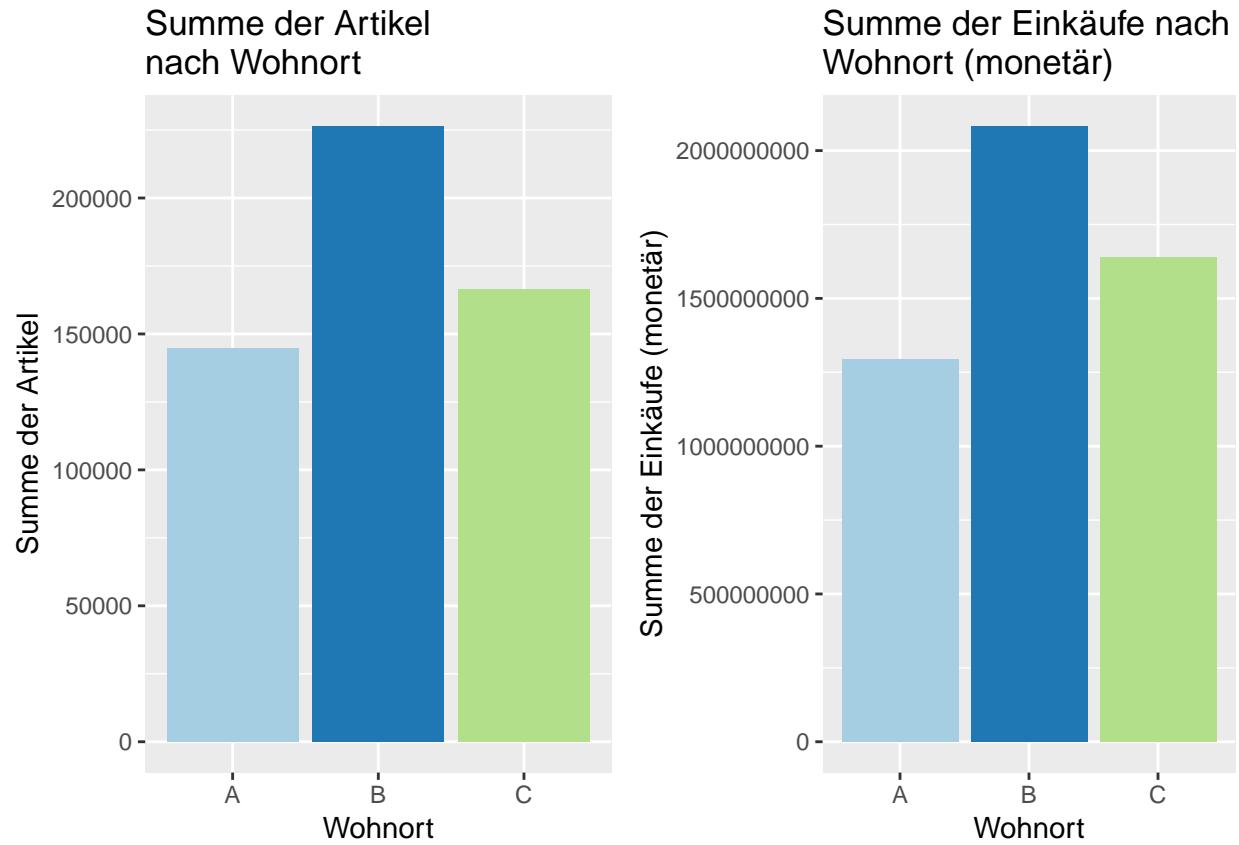
In diesem Teil beschäftigen wir uns mit der Analyse des Wohnortes unserer Kundne und gehen auf folgende Fragen ein:

- Wo leben unsere Kunden?
- Welche Kunden generieren den größten Umsatz?
- Welche Kunden generieren den größten Absatz?
- Wieviel trägt das Best-Verkaufste Produkt zu dem Gesamtumsatz in einem Wohnort bei?



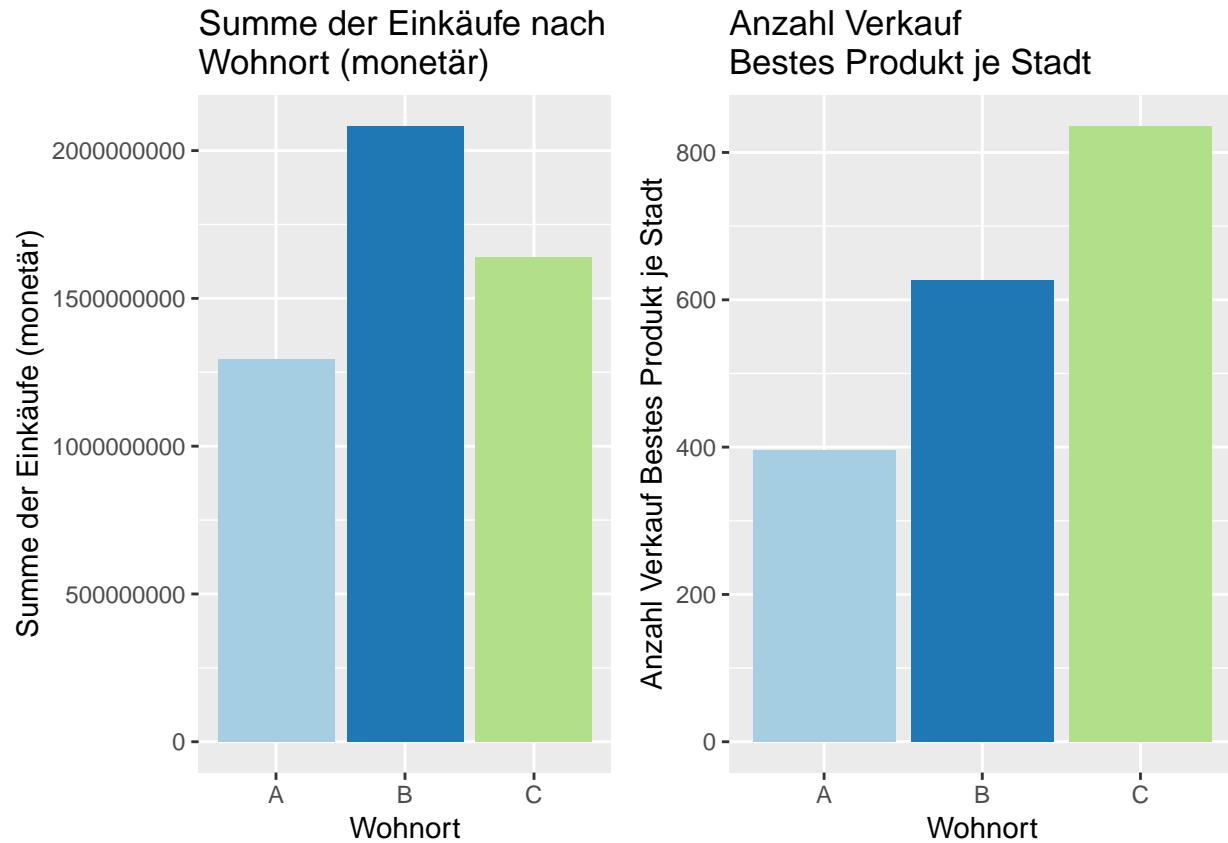
Obwohl die meisten Kunden aus **Wohnort C** stammen, haben die Kunden aus **Wohnort B** in Summe deutlich mehr für ihre Einkäufe ausgegeben, was anhand der oberen Grafiken ersichtlich ist.

Im Weiteren wollen wir uns näher mit eben diesem Phänomen beschäftigen, dazu betrachten wir die Anzahl der Artikel die in den Wohnorten erstanden hat.



Aus der oberen Gegenüberstellung ist ersichtlich dass der Wohnort B am meisten Waren an diesem Tag umgesetzt hat. Dies spiegelt sich auch in den Summen aller Einkäufe im Bezug zu den Wohnorten wieder. Damit kann nicht behauptet werden dass Die Kunden aus Wohnort B ausschließlich hochpreisige Produkte gekauft hätten, das beobachtete Peak wäre auch zu erklären wenn die Kunden eine größere Anzahl an Waren umgesetzt hätten.

Im weiteren Verlauf wollen wir nun das best-verkaufteste Produkt je Wohnort ermitteln. Dazu greifen wir auf die bereits existierende Information “bestes_Prod” zurück.

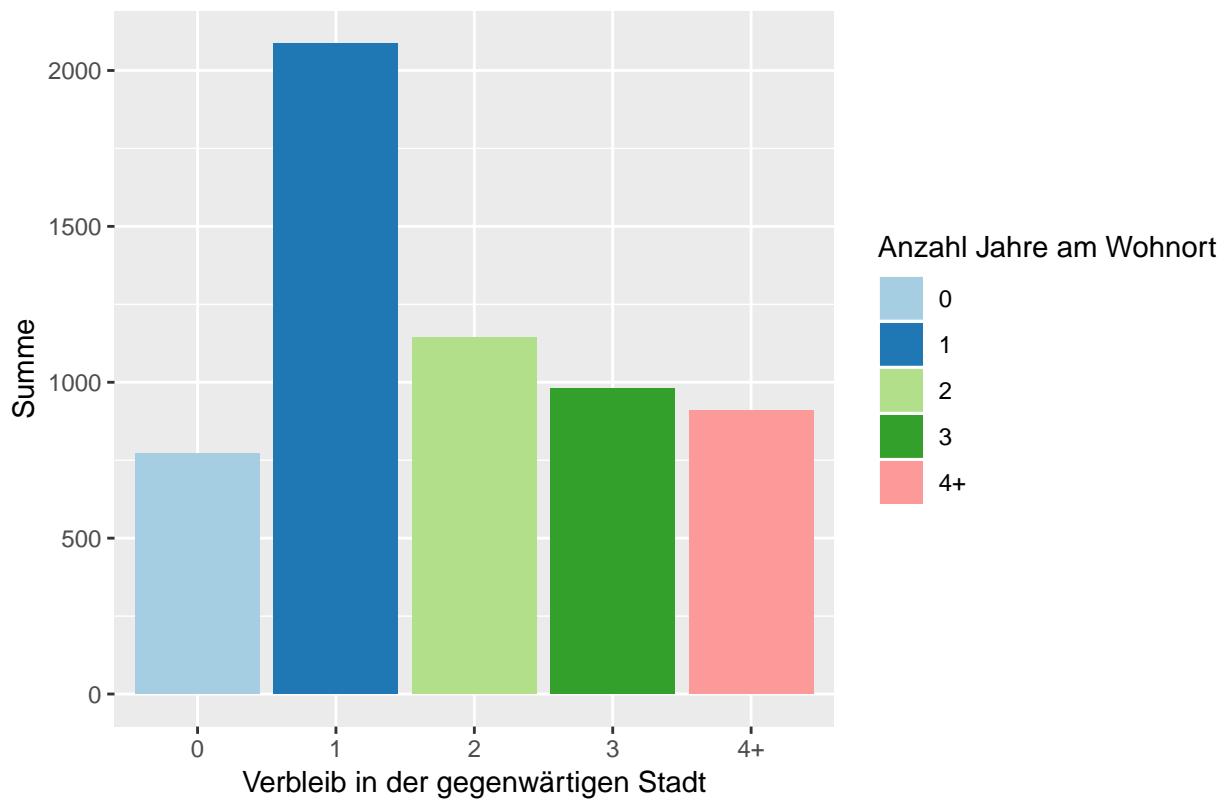


Mit der oberen Auswertung ist ersichtlich, dass zwar **Wohnort B** den meisten Usatz generiert. Dennoch wird das best-verkaufteste Produkt unerwarteterweise am Häufigsten in **Wohnort C** verkauft.

Kunden an einem spezifischem Wohnort

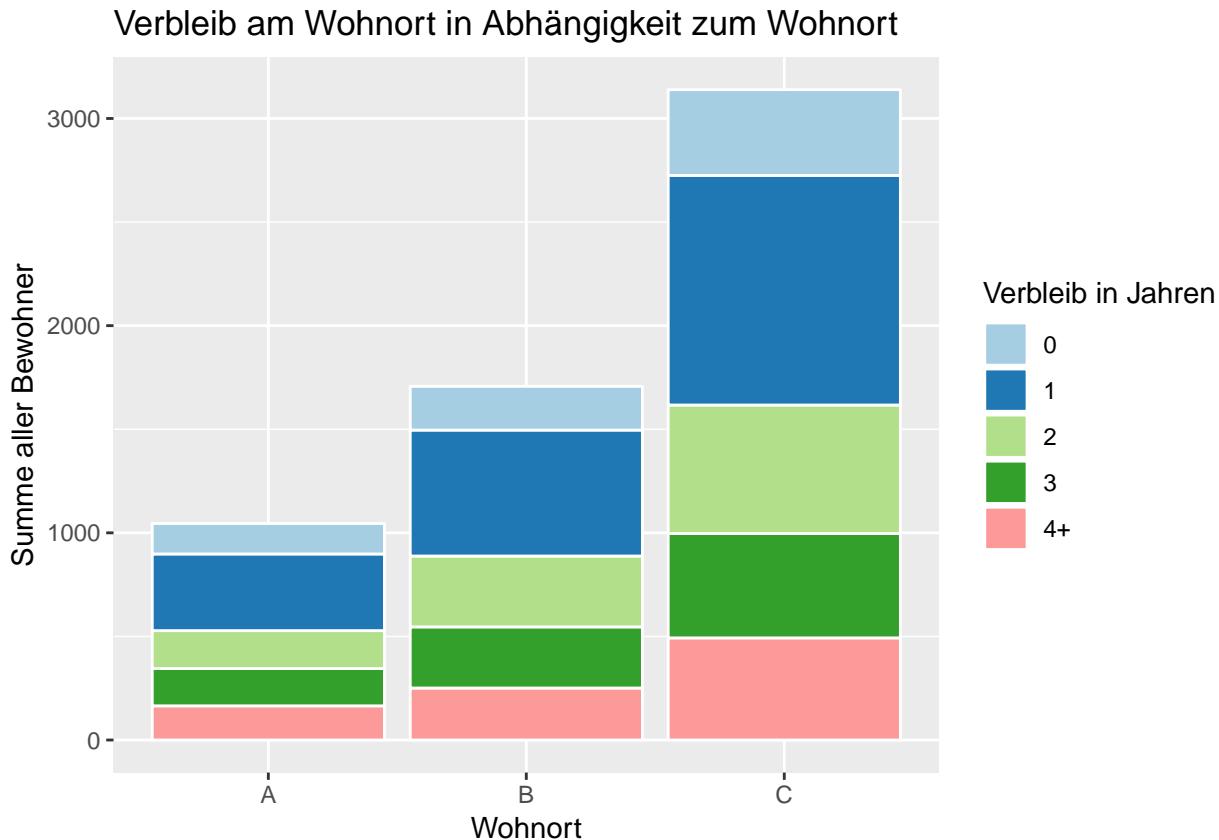
Im nächsten Schritt analysieren wir die Kundengruppe und der Aufenthalt am gegenwärtigen Wohnort genauer. Diese Information kann als Grundlage für die Analyse welche Produkte welchem Kunden in welcher Lebenssituation am besten vorgeschlagen werden verwendet werden. Dazu betrachten wir zunächst wie lange alle unsere Kunden an Ihrem gegenwärtigen Wohnort leben:

Bisheriger Verbleib am Wohnort



Anhand der oberen Grafik ist abzulesen, dass die meisten Kunden an ihrem jeweiligen Standort gegenwärtig 1 Jahr leben, jedoch bezieht sich diese Information auf alle drei Wohnorte kummoliert. Eine Erklärung wäre, dass alle drei Wohnorte vor einem Jahr eine einen starken „Zuzug“ erlebt hatten. Eine alternative Interpretation kann aber auch sein, dass in diesen Wohnorten die meisten Bürger nach einem Jahr erneut umziehen. Dies ist jedoch mit den gegebenen Daten nicht eindeutig bestimmbar.

Dennoch analysieren wir das obere Phänomen etwas granularer betrachten indem wir den Verbleib am bisherigen Wohnort nach den Städten aufteilen



Durch die Aufteilung ist zu erkennen, dass in allen drei Wohnorten die größte Bürgergruppe diejenige ist, welche gegenwärtig 1 Jahr ihren Wohnsitz dort gemeldet hat. Ob dies ein gegenwärtiger Trend ist lässt sich mit der gegebenen Datenlage nicht genauer beschreiben.

Feature Engineering

Wir schauen uns nochmal die Struktur der Daten an. Dabei fällt uns auf, dass manche Features noch nicht im richtigen Format sind.

```
## 'data.frame': 537577 obs. of 12 variables:
## $ User_ID          : int 1000001 1000001 1000001 1000001 1000002 ...
## $ Product_ID        : Factor w/ 3623 levels "P00000142","P00000242",...
## $ Gender            : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 2 2 2 ...
## $ Age               : Factor w/ 7 levels "0-17","18-25",...
## $ Occupation        : int 10 10 10 10 16 15 7 7 7 20 ...
## $ City_Category     : Factor w/ 3 levels "A","B","C": 1 1 1 1 3 1 2 2 2 1 ...
## $ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",...
## $ Marital_Status    : int 0 0 0 0 0 1 1 1 1 ...
## $ Product_Category_1: int 3 1 12 12 8 1 1 1 1 8 ...
## $ Product_Category_2: int 0 6 0 14 0 2 8 15 16 0 ...
## $ Product_Category_3: int 0 14 0 0 0 0 17 0 0 0 ...
## $ Purchase          : int 8370 15200 1422 1057 7969 15227 19215 15854 15686 7871 ...
```

Im folgenden restrukturieren wir unsere Daten. Die meisten Prädiktoren werden zu faktorielle Variablen umgewandelt. Die Produktkategorien werden als Integer belassen.

```

bf$User_ID <- as.factor(bf$User_ID)
bf$Occupation <- as.factor(bf$Occupation)
bf$Marital_Status <- as.factor(bf$Marital_Status)
str(bf)

## 'data.frame': 537577 obs. of 12 variables:
## $ User_ID : Factor w/ 5891 levels "1000001","1000002",...
## $ Product_ID : Factor w/ 3623 levels "P00000142","P00000242",...
## $ Gender : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 2 2 2 ...
## $ Age : Factor w/ 7 levels "0-17","18-25",...
## $ Occupation : Factor w/ 21 levels "0","1","2","3",...
## $ City_Category : Factor w/ 3 levels "A","B","C": 1 1 1 1 3 1 2 2 2 ...
## $ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",...
## $ Marital_Status : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 ...
## $ Product_Category_1 : int 3 1 12 12 8 1 1 1 8 ...
## $ Product_Category_2 : int 0 6 0 14 0 2 8 15 16 0 ...
## $ Product_Category_3 : int 0 14 0 0 0 0 17 0 0 0 ...
## $ Purchase : int 8370 15200 1422 1057 7969 15227 19215 15854 15686 7871 ...

```

Die nicht-nummerischen Variablen Gender und City-Category werden in Dummy-Variablen codiert. Die ursprünglichen Variablen Gender und City_Category werden entfernt.

```

bf_cor <- bf
bf_cor <- fastDummies::dummy_cols(bf_cor, select_columns = c("Gender", "City_Category"))
bf_cor$Gender = NULL
bf_cor$City_Category = NULL

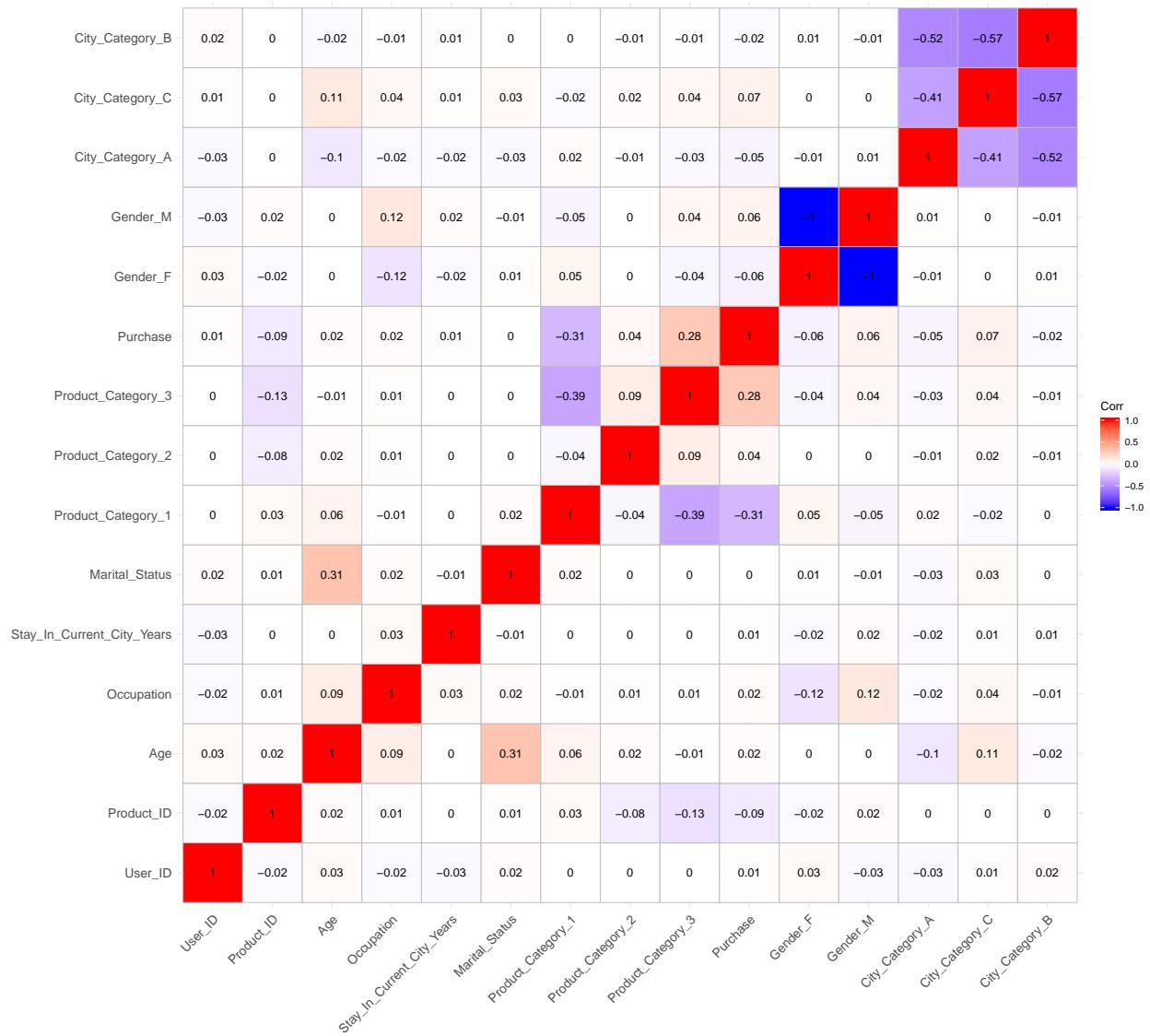
```

Visualisierung der Korrelationen anhand der Heatmap

```

bf_int <- bf_cor
bf_int[] <- lapply(bf_int, as.integer)
cormat <- round(cor(bf_int), 2)
ggcorrplot(cormat, lab=T)

```



In unserem Datensatz finden wir nur sehr schwache Korrelationen zwischen unseren Variablen. Die Variablen **Gender_M** und **Gender_F** korrelieren selbstverständlich negativ, da sie sich von Natur aus ausschließen. Eine schwache positive Korrelation ist zwischen **Purchase** und **Product_Category_3** zu erkennen, was uns sagt, dass mehr Geld für Produkt 3 ausgegeben wurde. Umgekehrt haben wir eine schwache negative Korrelation zwischen 'Purchase' und **Product_Category_1**, also weniger Geld das für Produkt 1 ausgegeben wurde. Es könnte entweder bedeuten, dass Produkte der Kategorie 3 einfacher teurer sind als Produkte der Kategorie 1 oder, dass Produkte der Kategorie 3 öfter gekauft wurden als die der Kategorie 1. Auch zu erkennen ist die schwache negative Korrelation zwischen Kategorie 1 und 3, was darauf schließen lässt, dass wenn Produkte aus Kategorie 1 gekauft wurden, wurden eher weniger Produkte der Kategorie 3 gekauft. Auch weisen die **City_Category A, B, C** untereinander negative Korrelationen auf, die für uns nicht weiter wichtig sind. Die schwache Korrelation zwischen Alter und Ehestatus (z.B. ältere Leute sind eher verheiratet) ist aufgrund sozialer Eigenschaften erklärt und für uns uninteressant.

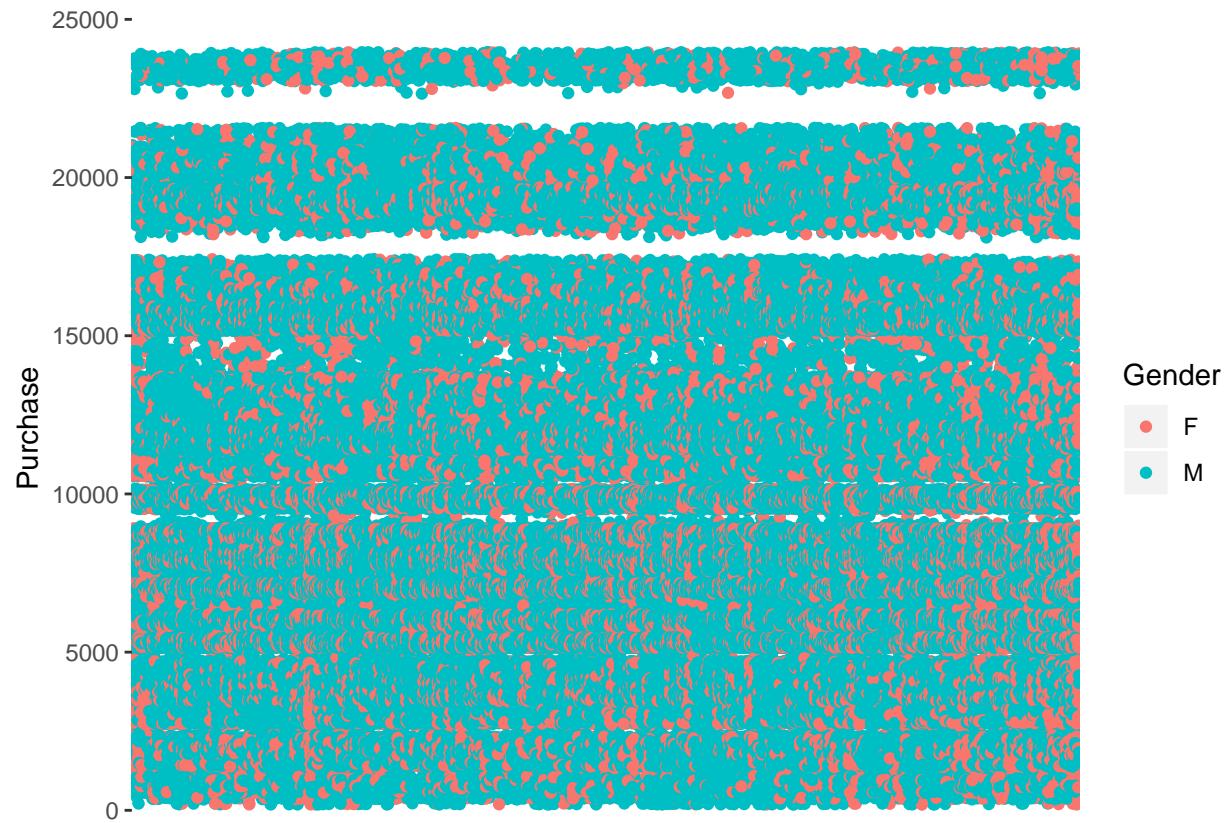
Skalierung

```
max_data = apply(bf_int, 2, max)
min_data = apply(bf_int, 2, min)

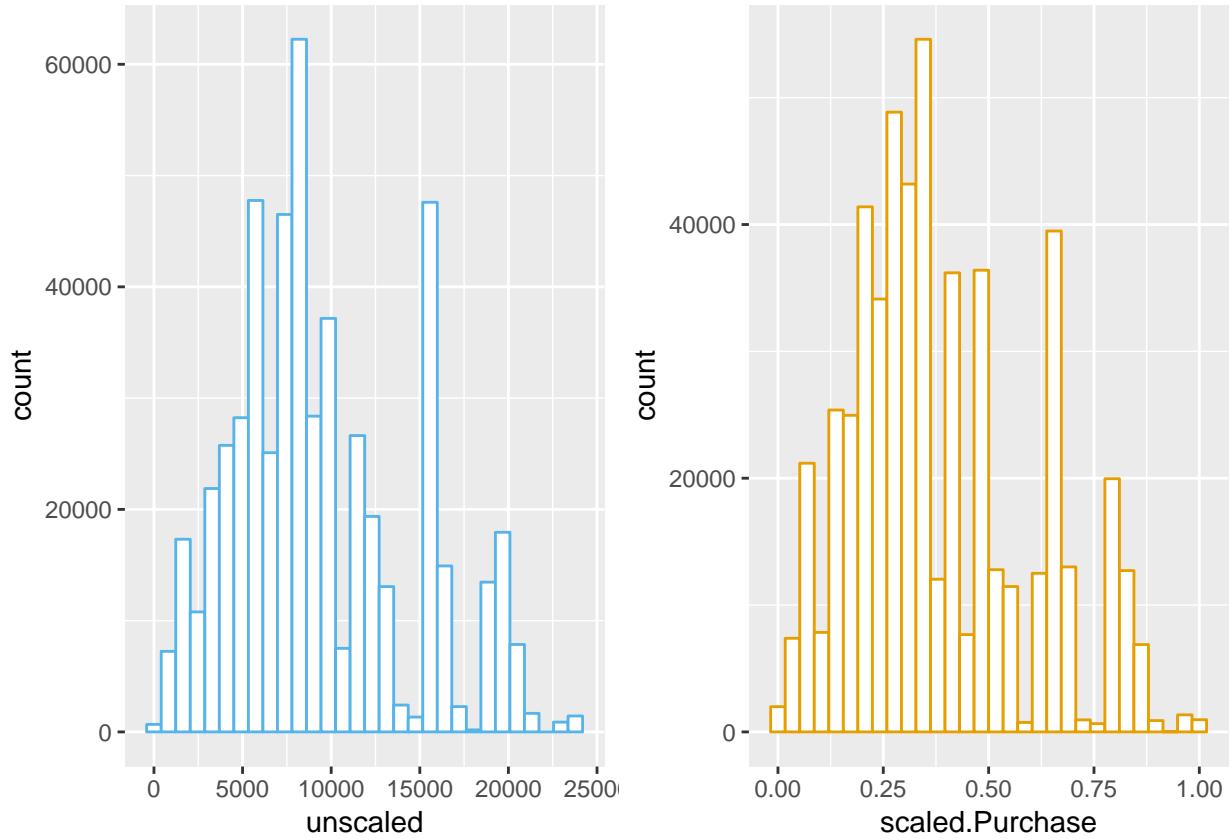
scaled = data.frame(scale(bf_int, center=min_data, scale=max_data-min_data))
summary(scaled)
```

	User_ID	Product_ID	Age	Occupation
##	Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000
##	1st Qu.:0.2462	1st Qu.:0.2559	1st Qu.:0.3333	1st Qu.:0.1000
##	Median :0.5008	Median :0.4544	Median :0.3333	Median :0.3500
##	Mean :0.4945	Mean :0.4672	Mean :0.4158	Mean :0.4041
##	3rd Qu.:0.7307	3rd Qu.:0.6996	3rd Qu.:0.5000	3rd Qu.:0.7000
##	Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000
##	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	
##	Min. :0.0000	Min. :0.0000	Min. :0.0000	
##	1st Qu.:0.2500	1st Qu.:0.0000	1st Qu.:0.0000	
##	Median :0.5000	Median :0.0000	Median :0.2353	
##	Mean :0.4649	Mean :0.4088	Mean :0.2527	
##	3rd Qu.:0.7500	3rd Qu.:1.0000	3rd Qu.:0.4118	
##	Max. :1.0000	Max. :1.0000	Max. :1.0000	
##	Product_Category_2	Product_Category_3	Purchase	Gender_F
##	Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000
##	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.2389	1st Qu.:0.0000
##	Median :0.2778	Median :0.0000	Median :0.3313	Median :0.0000
##	Mean :0.3769	Mean :0.2151	Mean :0.3848	Mean :0.2459
##	3rd Qu.:0.7778	3rd Qu.:0.4444	3rd Qu.:0.5000	3rd Qu.:0.0000
##	Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000
##	Gender_M	City_Category_A	City_Category_C	City_Category_B
##	Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000
##	1st Qu.:1.0000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000
##	Median :1.0000	Median :0.0000	Median :0.0000	Median :0.0000
##	Mean :0.7541	Mean :0.2691	Mean :0.3096	Mean :0.4213
##	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:1.0000
##	Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

Ein einfacher Scatterplot codiert nach männlichen und weiblichen Customern zeigt, dass das Volumen der einzelnen Einkäufe relativ uniform verteilt ist mit deutlichen Clustern im hoch-preisigen Segment.



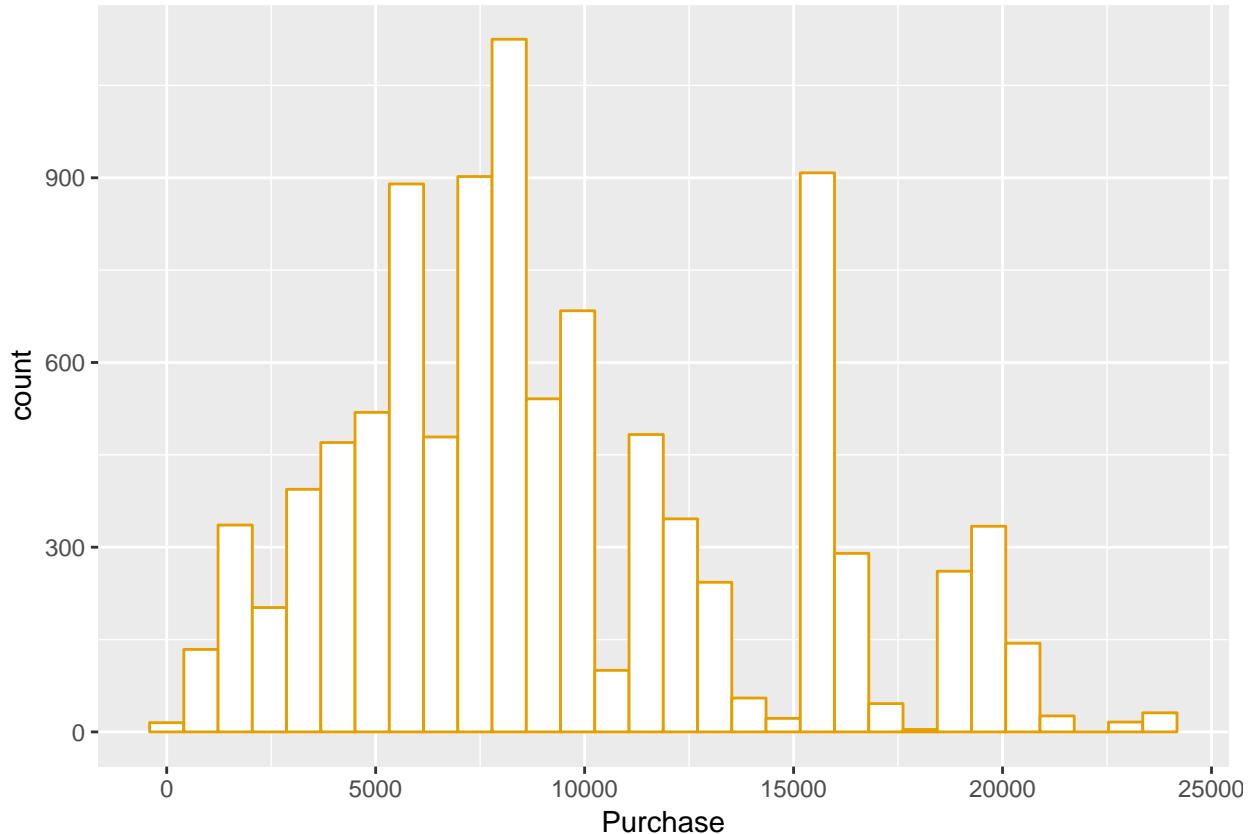
Eine der Annahmen fuer ein lineares Regressionsmodell ist die Normalverteilung der Residuen. Dafuer wird das Histogram der dependent Variable Purchase betrachtet:



Das Histogramm der Variable `Purchase` zeigt eine annähernde Normalverteilung mit einem leichten Skew nach links. Eine Skalierung des Datensatzes zeigt eine noch größere Annäherung an die Normalfunktion, deshalb wird für die folgenden Modelle im Preprocessing skaliert.

Der Datensatz enthält über 537.577 Instanzen. Da die Modellierung eines so grossen Datensatzes sehr viel Rechenkapazität beansprucht, wird mit einem Subsample von 10.000 Instanzen weitergearbeitet. Da die für uns interessante Variable `Purchase` annähernd normalverteilt ist, wird das Subsample zufällig gezogen. Da bei den unten beschriebenen Modellen Resampling durchgeführt, kann anhand des Subsamples wieder auf die ursprüngliche Verteilung des Datensatzes rückgeschlossen werden.

```
bf <- bf[sample(nrow(bf), 10000), ]  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Die Variable `Purchase` im Subsample ist immer noch normalverteilt.

Train Test Split

Im Folgenden unterteilen wir den Datensatz in Training und Test Daten. 70% werden als Trainingsdaten zugeteilt, damit ausreichend viele Beobachtungen im späteren Training verarbeitet werden können. 30% werden den Testdaten zugeteilt, um ein repräsentatives Testszenario zu bilden.

```
options(scipen=999)
index <- createDataPartition(
  y = bf$Purchase,
  ## Purchase ist dependent Variable
  p = .7,
  ## Verhältnis traing - test set
  list = FALSE
)
training <- bf[ index,]
testing <- bf[ -index,]
```

Modellierung des Sales Volumen

Für Werbetreibende ist besonders interessant, ihre Kunden gezielt mit Werbung zu bespielen, die auf die jeweilige demographische Gruppe angepasst ist. Werden Werbeanzeigen z.B. bei Google oder Instagram gekauft, kann genau definiert werden, welchem Kundensegment die Werbung angezeigt werden soll (z.B. nur Frauen zwischen 18 und 24). Damit schon vor Schalten der Werbung untersucht werden kann, welche Kunden besonders kauffreudig sind, soll der Preis, für den der jeweilige Kunde einkaufen wird, vorausgesagt werden.

Lineare Regression mit Resampling

Als Benchmark fuer alle weiteren Modelle, wird ein einfaches lineares Modell der Variable `Purchase` gefittet. Wie oben beschrieben, wird `Purchase` skaliert. Fuer das folgende Regression wird Resampling mit 5-facher Cross-validation angewendet. Durch das Resampling wird die Variabilität der zu erklärenden Variable simuliert.

```
##Modell mit Resampling

##resampling
ctrl <- trainControl(method = "repeatedcv", repeats = 5)

## Model Fitting
model_lin_regression <- train(
  Purchase ~ .,
  data = training,
  method = "lm",
  preProc = c('center', 'scale'),
  tuneLength = 15,
  na.action = na.pass,
  trControl = ctrl
)

#Anzeigen finales Modell
model_lin_regression$finalModel

#Prediction mit Testing Data
pred_lin_regression <- unname(predict(model_lin_regression, testing))
#RMSE
lin_model_rmse <- RMSE(testing$Purchase,pred_lin_regression)
lin_model_rmse
```

Der RMSE ist ziemmlich hoch. Dieser Wert soll nun in den folgenden Modellen verbessert werden.

Random Forest Regression (mit Resampling)

Für das folgende Modell wird ein Random Forest gefittet. Es wird derselbe Trainings- und Test-Datensatz verwendet.

```
#Tuning Grid
tune.grid <- seq(1,101, by=2)
tune <- expand.grid(mtry = 1:10)

#Random Forest Model
model_RF <- train(
  Purchase ~ .,
  data = training,
  method = "rf",
  preProc = c('center', 'scale'),
  trControl = ctrl,
  na.action = na.pass,
  ntree = 100,
  tuneGrid =tune)

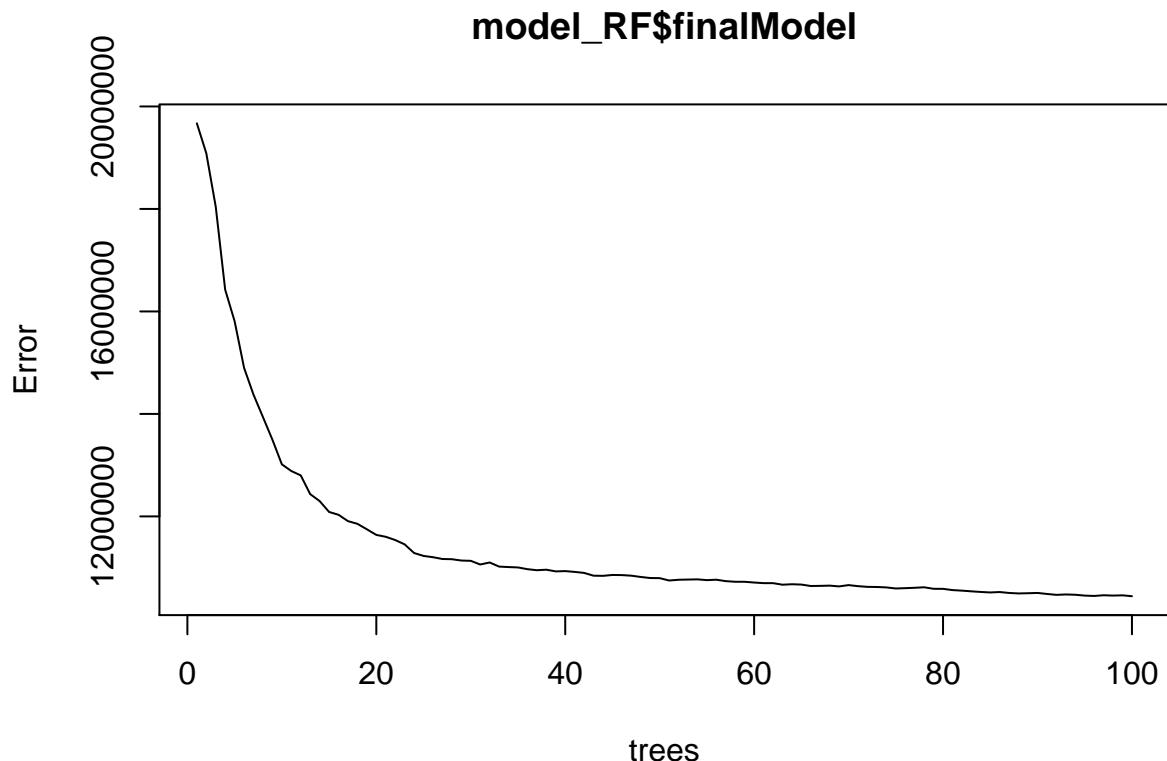
#prediction
pred_RF_test <- predict(model_RF,testing)
```

```
#RMSE  
RF_rmse_test <- RMSE(testing$Purchase,pred_RF_test)  
RF_rmse_test
```

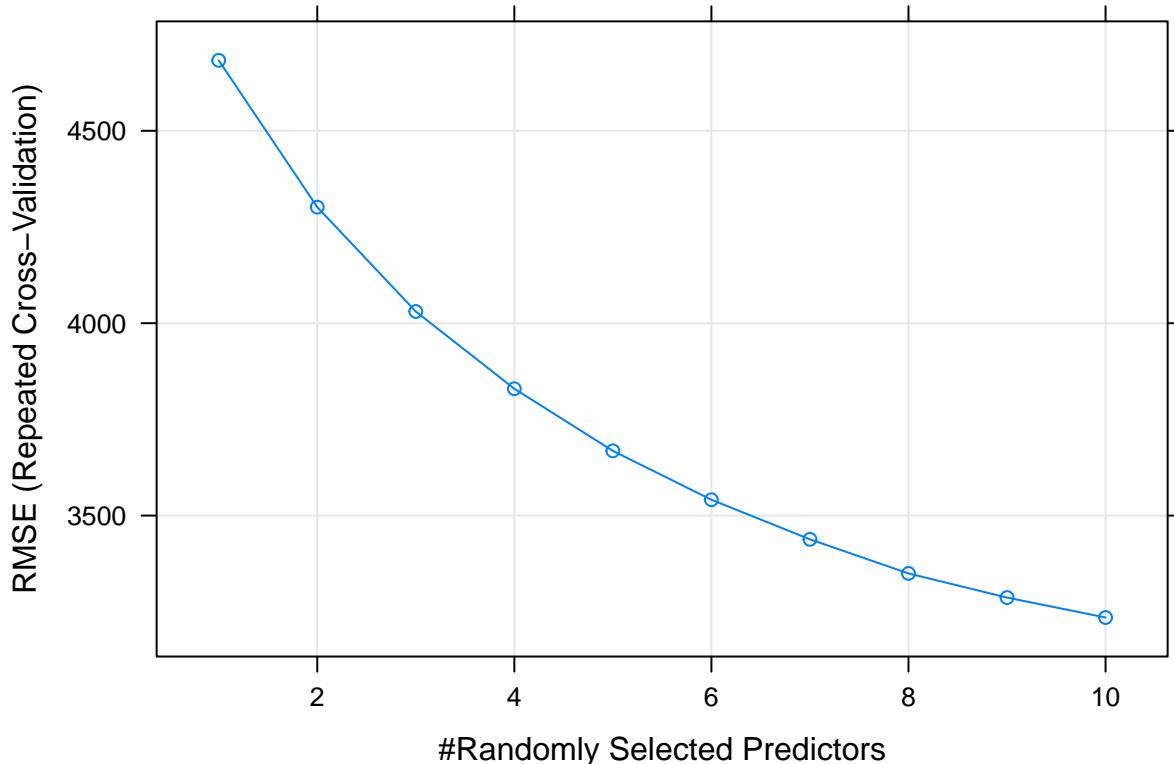
```
## [1] 3268.95
```

Durch dieses Modell konnte der RMSE nochmal um einiges Verbessert werden.

```
plot(model_RF$finalModel)
```



```
plot(model_RF)
```



Der obere Plot zeigt, wie sehr sich der Error im Laufe der getesteten Baeume verringert. Der untere Plot zeigt die Verringerung des RMSE durch das Tuning der Variable `mtry`, wobei `mtry` die Anzahl der Variablen ist, die zufaellig pro Split gesampelt werden. Je hoehher `mtry` desto besser der Regressionsbaum. Als optimaler Wert fuer `mtry` wurde hier 10 gewählt. Für eine Verbesserung des RMSE könnte ein noch höherer Wert für `mtry` gewählt werden, das würde aber unsere Rechenleistung sprengen.

Modelle fuer einzelne Customer Segmente

Durch die oben getesteten Modelle konnte die Prediction der Variable `Purchase` zwar verbessert werden, der RMSE ist aber immer noch recht hoch. Deswegen sollen fuer einzelne Customer Segmente jeweils eigene Modelle erstellt werden.

Männliche Customers haben in jeder Altersklasse eine durchschnittlich höhere Kaufkraft als Frauen (Frauen sind im Datensatz unterrepräsentiert; Verhältnis Männer/Frauen: 4:1,3). Besonders auffallend ist der Peak in der Altergruppe 26-35 bei der Anzahl der Purchases bei männlichen Customers, im Mean, ist dieser allerdings nicht mehr so signifikant, daher es ist mit einer hohen Varianz zu rechnen (siehe Section Descriptive Analyse: Geschlecht).

Im folgenden Modell wird jeweils ein Random-Forest Modell für weibliche & männliche Customer erstellt.

```
#Trainings/Test Split -> subset aus dem originalen Trainings/Test Split
fem_training <- subset(training,Gender == "F")
fem_testing <- subset(testing,Gender == "F")
male_training <- subset(training,Gender == "M")
male_testing <- subset(testing,Gender == "M")
```

```
#RF Model for Female
RF_female <- train(
```

```

Purchase ~ .,
data = fem_training,
method = "rf",
preProc = c('center', 'scale'),
trControl = ctrl,
na.action = na.pass,
ntree = 100,
tuneGrid =tune)

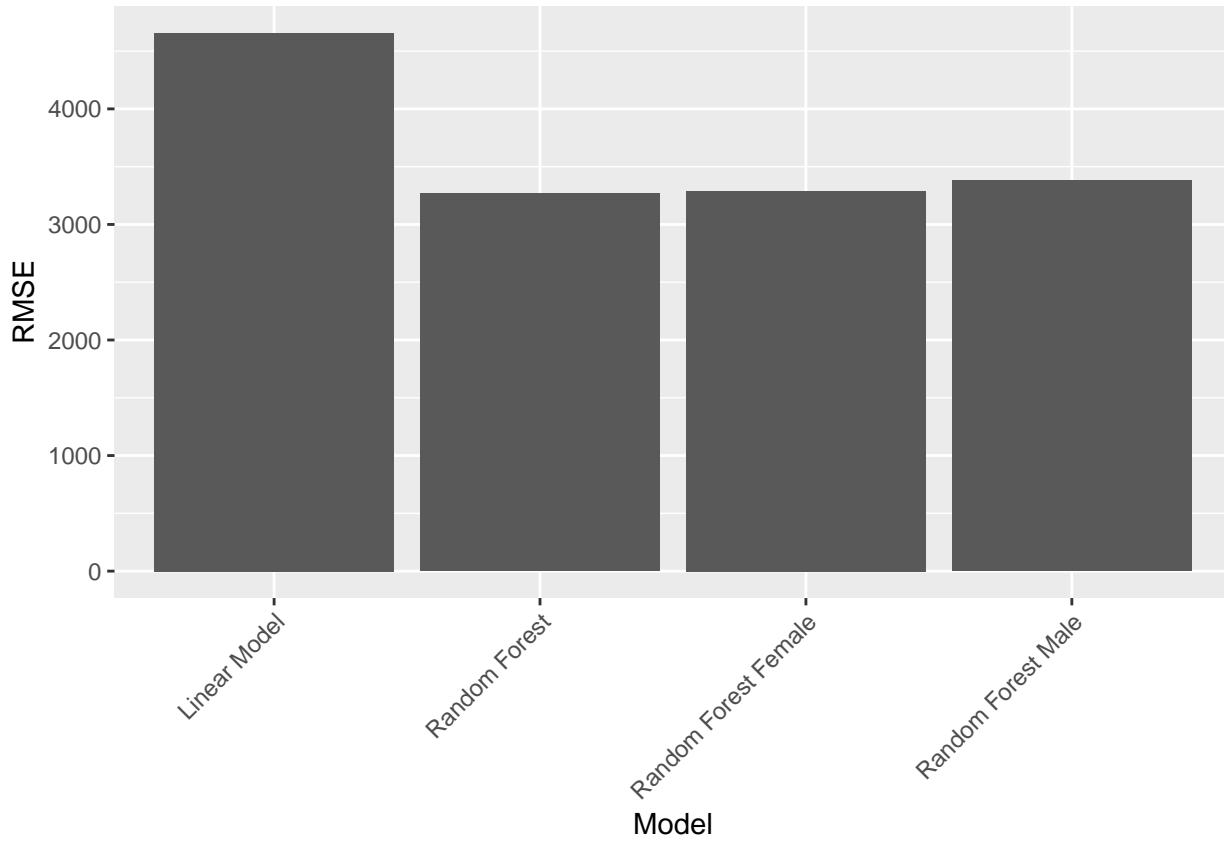
#RF Model for Male
RF_male <- train(
Purchase ~ .,
data = male_training,
method = "rf",
preProc = c('center', 'scale'),
trControl = ctrl,
na.action = na.pass,
ntree = 100,
tuneGrid =tune)

#Predictions
pred_RF_female <- predict(RF_female,fem_testing)
pred_RF_male <- predict(RF_male,male_testing)

#RMSE
RF_rmse_female <- RMSE(fem_testing$Purchase,pred_RF_female)
RF_rmse_male <- RMSE(male_testing$Purchase,pred_RF_male)

```

Evaluierung der Modelle



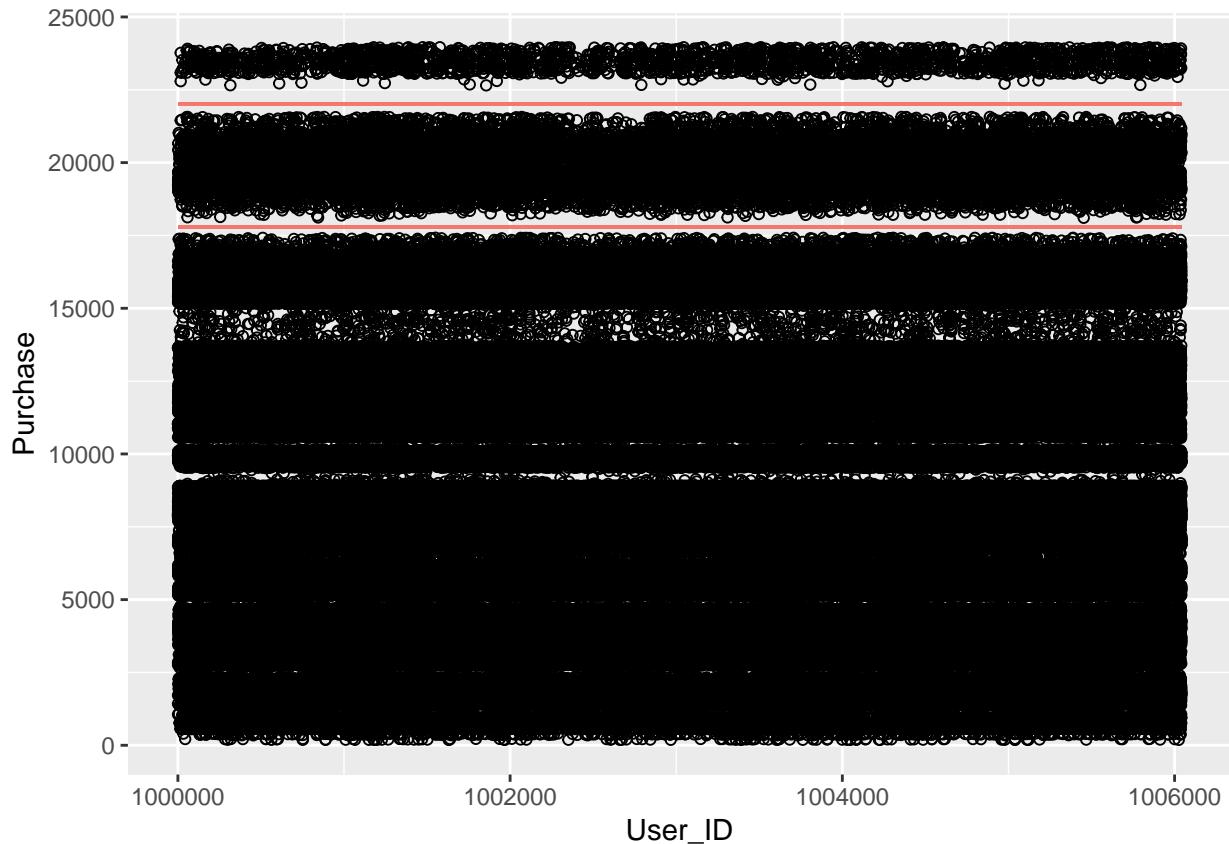
Der Vergleich aller 4 Modelle zeigt, dass für die allgemeinen Modelle das Random Forest Modell am besten performt. Segmentiert man den Datensatz in männliche und weibliche Customers und erstellt zwei separate Modelle, verbessert sich der RMSE jeweils noch mehr. In den obigen Barcharts ist zu sehen, dass bei männlichen Customers in der Alterklasse 26-35 zwar eine besonders große Anzahl von Purchases vorkommen, diese sich aber im Mean von den anderen Alterklassen nicht mehr so stark unterscheiden. Daher ist anzunehmen, das in dieser Altersklasse eine besonder hohe Varianz zu erwarten ist, die durch das lineare Modell nicht ausreichend repräsentiert wurde.

Eine Segmentierung der Customer macht dahingehend Sinn, da man im Internet & Social Media die Personengruppe, der gezielt Werbung vorgespielt wird teilweise sehr genau eingrenzen kann. Ein Prediction Modell fuer die einzelnen Kundengruppen führt daher zur zuverlaessigeren Vorhersage der Kaufkraft der entsprechenden Gruppe und beantwortet z.B. die Frage, ob es sich lohnt in Werbung für diese Gruppe zu investieren.

Multilabel Klassifizierung mit Neuralnet

Beim Betrachten des Scatterplots der Variable Purchase fällt auf, dass sich harte Grenzen im Bereich zwischen 17800 und 22000 und ab 22000 aufwärts befinden. Wir nehmen an, dass diese Usergruppen auch die realen

Cluster wiederspiegeln. Diese Cluster stellen die Kunden dar, die besonders viel und extrem viel einkaufen. Der Mobile Game Markt lebt z.B. von den sogenannten ‘Whales’, d.h. Kunden die besonders viel Geld für Ingame-Währungen ausgeben. Daher ist dieses Kundensegment besonders attraktiv. Der Großteil der Kunden befinden sich im untersten Segment.



Jetzt soll ein Klassifizierungsmodell für die Cluster gefügt werden. Dafür wird als erstes die Variable Purchase in 3 Faktoren low, medium & high gebinnt. Im Anschluss wird ein multilable Classification Model mithilfe eines Neural Networks modelliert.

```

training$Purchase <- cut(training$Purchase,breaks=c(0,17800,22000,max(training$Purchase)),
                           labels = c("low","medium","high"))
testing$Purchase <- cut(testing$Purchase,breaks=c(0,17800,22000,max(testing$Purchase)),
                        labels = c("low","medium","high"))

nnmodel <- neuralnet(Purchase.low+Purchase.medium+Purchase.high ~ .,
                      data=data_training,
                      hidden=c(4,3),
                      threshold = 1,
                      act.fct = "logistic",
                      stepmax=1000000)

confusionMatrix(actual_value,prediction)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   low  medium  high
##       low    2711      49      0
##       medium   100     100     100
##       high    100     100     100
##       total   2811     200     100
## 
## 
## 
## 
## 
## 
## 
## 
```

```

##      medium  202     25     0
##      high     12     0     0
##
## Overall Statistics
##
##          Accuracy : 0.9123
## 95% CI : (0.9016, 0.9222)
## No Information Rate : 0.9753
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.1277
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: low Class: medium Class: high
## Sensitivity          0.9268      0.337838       NA
## Specificity           0.3378      0.930940    0.995999
## Pos Pred Value        0.9822      0.110132       NA
## Neg Pred Value        0.1046      0.982323       NA
## Prevalence            0.9753      0.024675   0.000000
## Detection Rate        0.9040      0.008336   0.000000
## Detection Prevalence  0.9203      0.075692   0.004001
## Balanced Accuracy     0.6323      0.634389       NA

```

Die Confusion Matrix zeigt die Anzahl der richtig kategorisierten Instanzen. Durch die Überrepräsentation der Kategorie `low` wurde nur wenige Instanz aus der Kategorie `medium` und gar keine aus der Kategorie `high` richtig geschätzt, was allerdings trotzdem zu einer hohen Accuracy von >90% führt. Die Verteilung der Klassen im Verhältnis von 190:16:1 (`low:medium:high`) ist sehr unausgewogen, was zu einem hohen Bias des Modells führen kann. Auch sind die Kategorien nicht äquidistant, sonder aufgrund der visuell bestimmten Cluster gewählt worden. Da Aufgrund von Kapazitätsproblemen, nicht der gesamte Datensatz als Trainingssatz verwendet werden konnte, könnte die Accuracy noch weiter verbessert werden, da sich die Performance von Neuronalen Netzen mit zunehmenden Trainingsdaten steigert. Für eine weiterführende Analyse sollten die Klassen ausbalanciert werden, um den Bias zu vermeiden.

```

table(training$Purchase)
##
##      low medium  high
##     6424    542    35

```

Die Verteilung der Faktoren in `Purchase` zeigt, dass die Kategorie `high` mit nur 35 Instanzen unterrepräsentiert ist. Bei einem balancierten Datensatz, würde man aus jeder Klasse deshalb 35 Instanzen sammeln. Das ist aber für das Training eines NNs viel zu wenig. Deshalb wird wieder auf den gesamten Datensatz zurückgegriffen.

```

##
##      low medium  high
## 494150  41111   2316

```

Jetzt befinden sich 2316 Instanzen in der Kategorie `high`. Deshalb werden aus den beiden anderen Klassen jetzt 2316 Instanzen gesampled.

```

low = subset(data, data$Purchase=='low')
medium = subset(data, data$Purchase=='medium')
high = data.frame(subset(data, data$Purchase=='high'))

```

```

low <- data.frame(low[sample(nrow(low), 2316), ])
medium2 <- data.frame(medium[sample(nrow(medium), 2316), ])

nn_data <- bind_rows(low, medium2, high)
table(nn_data$Purchase)

##
##      low   medium    high
##      2316    2316    2316

```

Trainings/Test Split f?r NN

```

index <- createDataPartition(
  y = nn_data$Purchase,
  p = .7,
  list = FALSE)

nn_training <- nn_data[ index, ]
nn_testing <- nn_data[-index,]

nnmodel2 <- neuralnet(Purchase.low+Purchase.medium+Purchase.high ~ .,
                        data=nn_training,
                        hidden=c(10,10),
                        threshold = 1,
                        act.fct = "logistic",
                        stepmax=1000000)

confusionMatrix(actual_value,prediction)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction low   medium   high
##      low     489      174      31
##      medium    74      578      42
##      high      0       0     694
##
## Overall Statistics
##
##          Accuracy : 0.8458
##          95% CI : (0.8296, 0.8611)
##      No Information Rate : 0.3684
##      P-Value [Acc > NIR] : < 0.00000000000000022
##
##          Kappa : 0.7687
##
## McNemar's Test P-Value : < 0.00000000000000022
##
## Statistics by Class:
##
##          Class: low Class: medium Class: high
## Sensitivity           0.8686        0.7686        0.9048
## Specificity            0.8650        0.9128        1.0000

```

```

## Pos Pred Value      0.7046      0.8329      1.0000
## Neg Pred Value     0.9467      0.8746      0.9474
## Prevalence         0.2704      0.3612      0.3684
## Detection Rate    0.2349      0.2776      0.3333
## Detection Prevalence 0.3333      0.3333      0.3333
## Balanced Accuracy  0.8668      0.8407      0.9524

```

Die Accuracy mit den Ausbalancierte Klassen ist nun etwas niedriger. Besonders die Klassen `high` und `medium` werden gut klassifiziert, bei der Klasse `low` wird oft als `medium` klassifiziert. Zuletzt wird noch gestestet, wie gut das Modell am gesamten Datensatz performt:

```
confusionMatrix(actual_value,prediction)
```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    low medium   high
##   low       341724 128907  23519
##   medium     3576  34854   2681
##   high        0     0   2316
##
## Overall Statistics
##
##                 Accuracy : 0.7048
##                           95% CI : (0.7036, 0.706)
##   No Information Rate : 0.6423
##   P-Value [Acc > NIR] : < 0.0000000000000022
##
##                 Kappa : 0.2354
##
## McNemar's Test P-Value : < 0.0000000000000022
##
## Statistics by Class:
##
##                         Class: low Class: medium Class: high
## Sensitivity           0.9896      0.21283     0.081218
## Specificity          0.2073      0.98326     1.000000
## Pos Pred Value       0.6915      0.84780     1.000000
## Neg Pred Value       0.9177      0.74035     0.951052
## Prevalence           0.6423      0.30463     0.053045
## Detection Rate       0.6357      0.06484     0.004308
## Detection Prevalence 0.9192      0.07647     0.004308
## Balanced Accuracy    0.5985      0.59805     0.540609

```

Basierend auf dem gesamten Datensatz ist die Accuracy auf 70% gesunken. Unser Modell generalisiert also nicht so gut wie gehofft, bzw. wie es uns der vorherige Test gezeigt hat. Gründe dafür könnten die folgenden sein:

- (1) Die Trainingsdaten für das ausbalancierte Training waren zu wenig, um eine gute Generalisierung zu bewirken. Es spiegelt sich wieder, dass `low` sehr oft als `medium` klassifiziert wird. Es ist also davon auszugehen, dass die Vielseitigkeit der Daten (`low`, `medium`) über die geringe Anzahl der Trainingsdaten nicht repräsentiert werden konnte.
- (2) Das Training war nicht ausreichend genug und das Modell zu schwach, was zum underfitting führte.

Zusätzlich enthielt der gesamte Datensatz wiederum den Anteil aller Trainingsdaten, was unseren Test zur Generalisierung verfälschte. Wir haben, das aber vernachlässigt, da die Anzahl der Trainingsdaten im Vergleich zur ganzen Stichprobe nur 1.3% betrug. Würden wir diese explizit aus diesem Test entfernen,

dann würde sich die Accuracy um wenige %% weiter verringern.

Aufgrund der Verteilung der Daten konnte keines der obigen Regressionsmodelle eine genaueren Vorhersage des Sales Volumen liefern. Das anfängliche Modell einer lineare Regression konnte durch ein Random Forest Modell sowie durch separate Modelle für männliche und weibliche Customer jedoch verbessert werden. Betrachtet man allerdings das Vorhersage-Problem anstatt als eine Regression als ein Klassifizierungsproblem, kann hier eine Accuracy von über 85% erreicht werden. Die Cluster wurden hier visuell bestimmt. Ein Clustering mithilfe eines Algorithmus z.B. KNN würde zu einer schärferen Trennung der Cluster führen. Zur weiteren Segmentierung könnte man zu jedem der Cluster `low`, `medium` & `high` ein eigenes Regressionsmodell fitten.

Schlusswort

Nach dem wir unsere Daten deskriptiv analysiert und explorative aufbereitet haben, konnten wir tiefergreifendere Informationen aus unseren Kundendaten beziehen. Diese Projektarbeit gestaltete sich für uns als eine observativen Studie. Dabei haben wir `Purchase` als Response-Variable und alle restlichen Features (exkl. `User_ID`, `Product_ID`) als Prädiktoren ausgewählt. Unsere Random Forest Modell (RMSE 3268) performte signifikant besser als unser lineares Modell (RMSE 4655). Weitgehend ist der RMSE vom Datensatz abhängig, welcher in unserem Fall sehr unbalanciert ist und, dahergehend die Vorhersagegenauigkeit limitiert.

Weitere Herangehensweise

Unsere weitere Herangehensweise wäre drei individuelle Trainingsdatensätze aus den drei Clustern (`high`, `medium`, `low`) zu samplen. Dadurch könnten wir Random Forest Modelle trainieren die darauf spezialisiert sind, die Purchases der jeweiligen Cluster best-möglich vorherzusagen. Unsere Vermutung ist, dass sich der RMSE dadurch noch weiter senken ließe, indem wir Testdaten zuerst über das Neuronale Netzwerk in die drei Cluster (`high`, `medium`, `low`) klassifizieren und dann dem entsprechenden Random Forest Modell (trainiert auf Cluster `high`, `medium`, `low`) übergeben. Aus unserer Sicht hätte das jedoch den Scope dieses Projektes überschritte und wir beschlossen uns diese Idee gedanklich durchzuspielen aber nicht umzusetzen.