Prüfungsleistung Data Science & Machine Learning: Salary by job title and country

Mathis, Julia und Jonas

2023-11-13

Table of contents

Die Analyse von Gehaltsdaten spielt eine entscheidende Rolle in der Forschung zur Entwicklung des Arbeitsmarktes. In dieser Arbeit wird der Datensatz “Salary by Job Title and Country” von der Website Kaggle als Grundlage für die Untersuchung von Gehältern unter verschiedenen Gesichtspunkten, wie beispielsweise Berufsfeld oder Geschlecht, genutzt. Der Datensatz wird vor der Analyse aufbereitet und anschließend eingehend analysiert.

1 1. Vorbereitung

In diesem Teil der Arbeit werden die erforderlichen Vorbereitungen getroffen, um die Durchführung des Projekts zu ermöglichen. Hierzu liegt ein Datensatz mit dem Titel “Salary by Job Title and Country” unter folgendem Link zur Verfügung:

https://www.kaggle.com/datasets/amirmahdiabbootalebi/salary-by-job-title-and-country

Dieser Datensatz stellt eine umfassende Sammlung von Gehaltsinformationen aus unterschiedlichen Branchen und Regionen bereit. Er enthält Angaben zu Berufsbezeichnungen, Gehältern, Berufserfahrungen, geografischen Standorten und weiteren Aspekten, die in

Umfragen gesammelt wurden. Die Analyse dieser Daten ermöglicht es, Einblicke in Arbeitsmarkttrends zu gewinnen und beispielsweise die Höhe der Gehälter in Abhängigkeit von unterschiedlichen Kriterien zu untersuchen.

Zusätzliche Quellen die für die Arbeit verwendet wurden befinden sich unter folgenden Links:

https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/

https://ggplot2.tidyverse.org/reference/

ggplot2 - Elegante R Plots (statistikprofis.com)

https://statologie.de/daten-standardisieren-r/

https://statologie.de/vorhergesagte-werte-plotten-r/

https://cran.r-project.org/web/packages/yardstick/yardstick.pdf

1.1 1.1 Importieren der benötigten Packages

Zuerst werden alle benötigten Packages für die Datenanalyse heruntergeladen.

```
library(tidyverse)
library(tidymodels)
library(corrplot)
library(explore)
library(ggplot2)
library(corrplot)
library(dplyr)
library(viridis)
library(rpart.plot)
library(yardstick)
```

Häufig kommt:
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding: https://cran.rstudio.com/bin/windows/Rtools/ Warning in install.packages : Paket ‘dplyr’ wird gerade benutzt und deshalb nicht installiert.

Installiere RTools nach Link: https://cran.rstudio.com/bin/windows/Rtools/rtools43/rtools.html

1.2 1.2 Einlesen der zu Analysierenden Daten

Daraufhin wird der Datensatz “Salary by Job Title and Country” eingelesen.

```
salary <- read_csv("Salary.csv")
```

```
Rows: 6684 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): Gender, Job Title, Country, Race
dbl (5): Age, Education Level, Years of Experience, Salary, Senior

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2 2. Erster Überblick der Daten

Um einen ersten Überblick über die Daten zu erhalten, werden die ersten 10 Zeilen der Tabelle ausgelesen:

```
head(salary, 10)
```

```
# A tibble: 10 x 9
     Age Gender `Education Level` `Job Title`       `Years of Experience` Salary
   <dbl> <chr>              <dbl> <chr>                             <dbl>  <dbl>
 1    32 Male                   1 Software Engineer                     5  90000
 2    28 Female                 2 Data Analyst                          3  65000
 3    45 Male                   3 Manager                              15 150000
 4    36 Female                 1 Sales Associate                       7  60000
 5    52 Male                   2 Director                             20 200000
 6    29 Male                   1 Marketing Analyst                     2  55000
 7    42 Female                 2 Product Manager                      12 120000
 8    31 Male                   1 Sales Manager                         4  80000
 9    26 Female                 1 Marketing Coordi~                     1  45000
10    38 Male                   3 Scientist                            10 110000
# i 3 more variables: Country <chr>, Race <chr>, Senior <dbl>
```

Mithilfe der “describe_tbl”- Funktion können die generellen Informationen über den Datensatz ermittelt werden.

```
describe_tbl(salary)
```

```
6 684 (6.7k) observations with 9 variables
0 observations containing missings (NA)
0 variables containing missings (NA)
0 variables with no variance
```

Hier lässt sich erkennen, dass der Datensatz 6684 Instanzen enthält, wovon keine Instanz einen Wert ohne Angabe (NA's) besitzt.

Nun wird ein kurzer Blick auf die Art der Merkmale geholfen. Gibt es kategorische oder nummerische Merkmale innerhalb des Datensatzes?

```
describe(salary)
```

```
# A tibble: 9 x 8
  variable            type     na na_pct unique   min      mean    max
  <chr>               <chr> <int>  <dbl>  <int> <dbl>     <dbl>  <dbl>
1 Age                 dbl       0      0     41    21     33.6      62
2 Gender              chr       0      0      2    NA     NA        NA
3 Education Level     dbl       0      0      4     0      1.62      3
4 Job Title           chr       0      0    129    NA     NA        NA
5 Years of Experience dbl       0      0     37     0      8.08     34
6 Salary              dbl       0      0    437   350 115307.   250000
7 Country             chr       0      0      5    NA     NA        NA
8 Race                chr       0      0     10    NA     NA        NA
9 Senior              dbl       0      0      2     0      0.14      1
```

Table 1: Wie bereits in der Tabelle zu erkennen gibt es innerhalb des Datensatzes nur zwei verschiedene Datentypen. Die Felder *Age, Education Level, Years of Experience, Salary, Senior* sind numerische Merkmale. Bei den Feldern *Gender, Job Title, Country, Race* handelt es sich um kategorische Merkmale.

Spalte	Typ	Bedeutung
Age	Numerisch	Alter
Gender	Kategorisch	Geschlecht
Education Level	Numerisch	Bildungsgrad
Job Title	Kategorisch	Jobtitel
Years of Experience	Numerisch	Arbeitserfahrung in Jahren
Salary	Numerisch	Gehalt
Country	Kategorisch	Land
Race	Kategorisch	Ethnizität
Senior	Numerisch	Senior position ja(1)/nein(0)

2.1 2.1 Bedeutung von Spalten und Datentypen

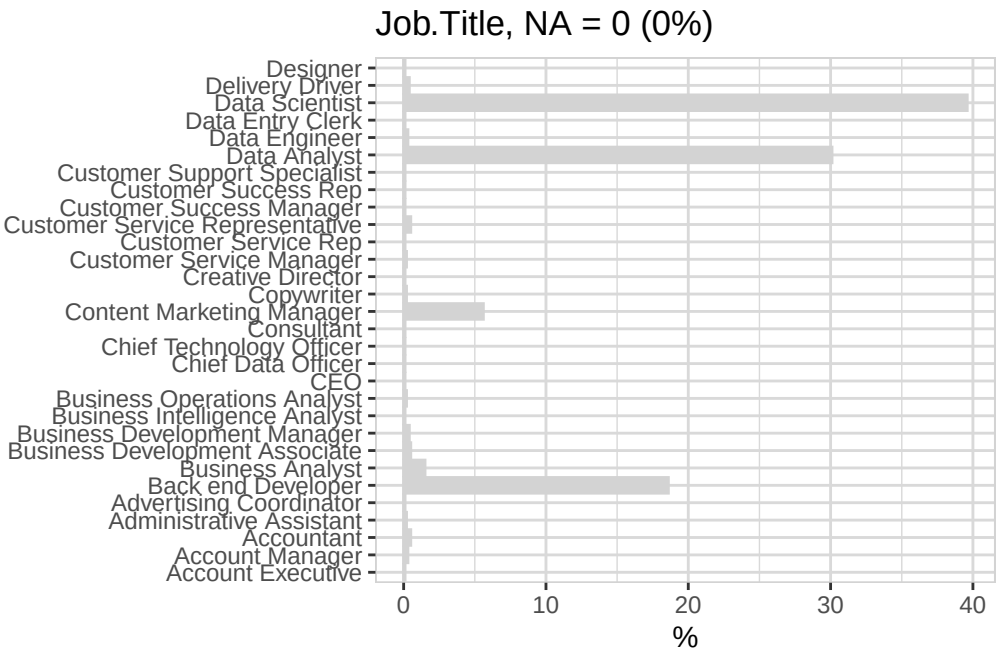
Im folgenden Abschnitt werden verschiedene Funktionen dafür verwendet, um die Datentypen und Bedeutung der Spalten zu verstehen.

```
salary <- salary |>
    rename(
      Job.Title = `Job Title`,
      Years.Of.Experience = `Years of Experience`,
      Education.Level = `Education Level`
    )
```

Hier werden die Spaltennamen derjenigen Spalten angepasst, die Leerzeichen enthalten. Konkret handelt es sich um die Spalten "Job Title", "Years of Experience" und "Education level". Das Leerzeichen wird durch einen Punkt ersetzt. Diese Anpassung erleichtert den zukünftigen Zugriff auf die Spaltennamen im Verlauf des Projektes und spart somit Zeit.

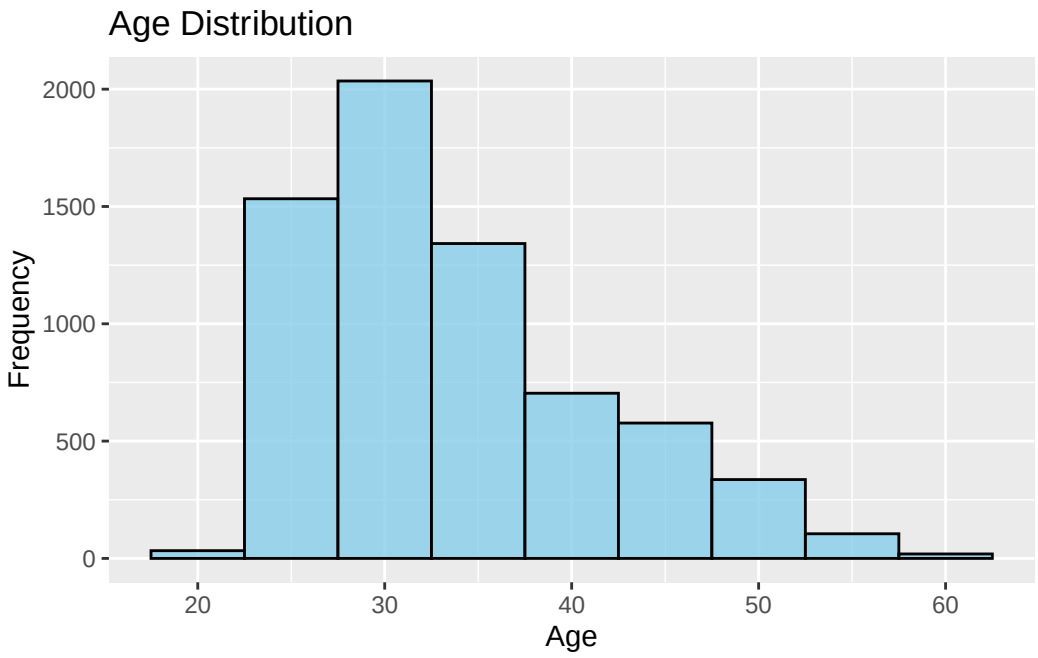
Im Anschluss verschaffen wir uns einen Überblick über die prozentuale Verteilung der Jobtitel. Die Grafik verdeutlicht, dass der Beruf des "Data Scientist" am häufigsten vertreten ist. Zudem zeigt sich innerhalb des Datensatzes eine signifikante Anzahl von "Data Analysten" sowie "Backend Developern".

```
explore (salary, Job.Title)
```



Altersverteilung:

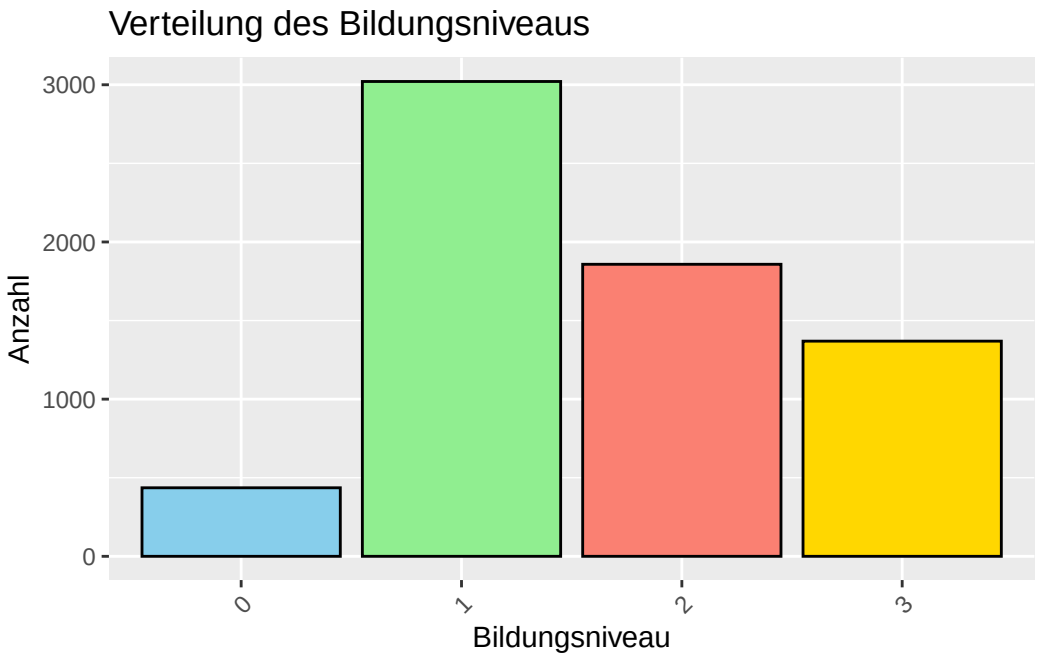
```
ggplot(salary, aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black", alpha = 0.8) +
  labs(title = "Age Distribution",
       x = "Age",
       y = "Frequency")
```



Verteilung des Bildungsniveaus:

Zunächst wird eine Farbpalette mit verschiedenen Farben definiert. Anschließend wird ein Balkendiagramm erstellt, wobei unterschiedliche Farben für jede Balkenstange verwendet werden, basierend auf dem Bildungsniveau.

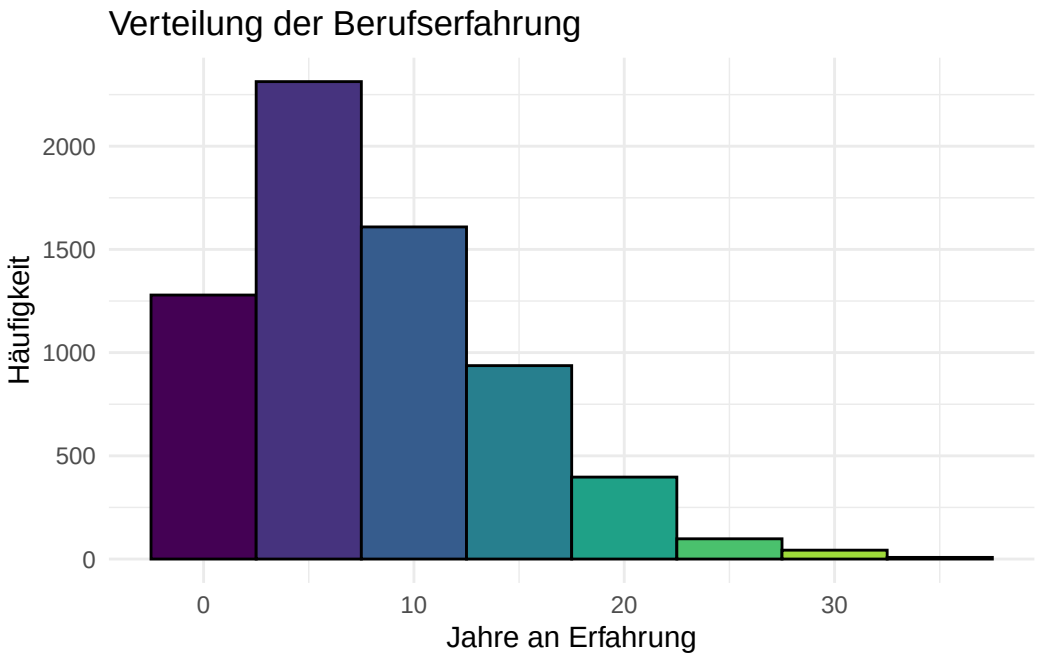
```
my_colors <- c("skyblue", "lightgreen", "salmon", "gold")
ggplot(salary, aes(x = factor(`Education.Level`))) +
  geom_bar(fill = my_colors, color = "black") +
  labs(title = "Verteilung des Bildungsniveaus",
       x = "Bildungsniveau",
       y = "Anzahl") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Verteilung der Arbeitserfahrung in Jahren:

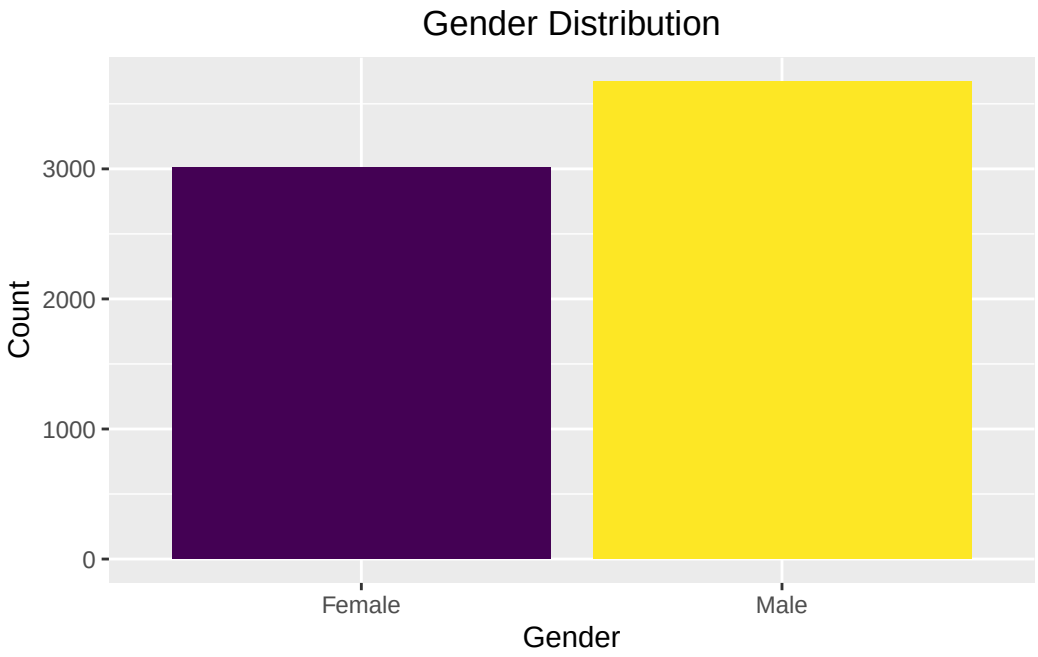
Ab diesem Punkt wurde teilweise das Paket “Viridis” für die farbliche Darstellung verwendet, um eine Alternative zur manuellen Deklaration der Farben aufzuzeigen.

```
ggplot(salary, aes(x = `Years.Of.Experience`)) +
  geom_histogram(binwidth = 5, fill = viridis(8), color = "black") +
  labs(title = "Verteilung der Berufserfahrung",
       x = "Jahre an Erfahrung",
       y = "Häufigkeit") +
  theme_minimal()
```



Verteilung der Geschlechter:

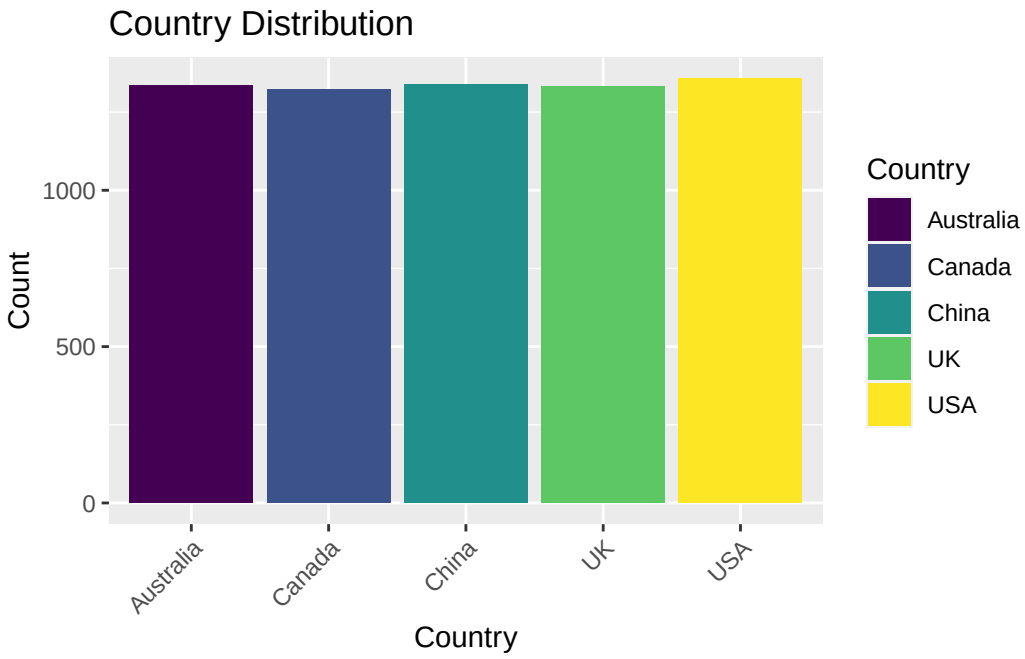
```
ggplot(salary, aes(x=Gender)) +
  geom_bar(fill=viridis(2)) +
  ggtitle("Gender Distribution") +
  xlab("Gender") +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5))
```



Verteilung der Länder:

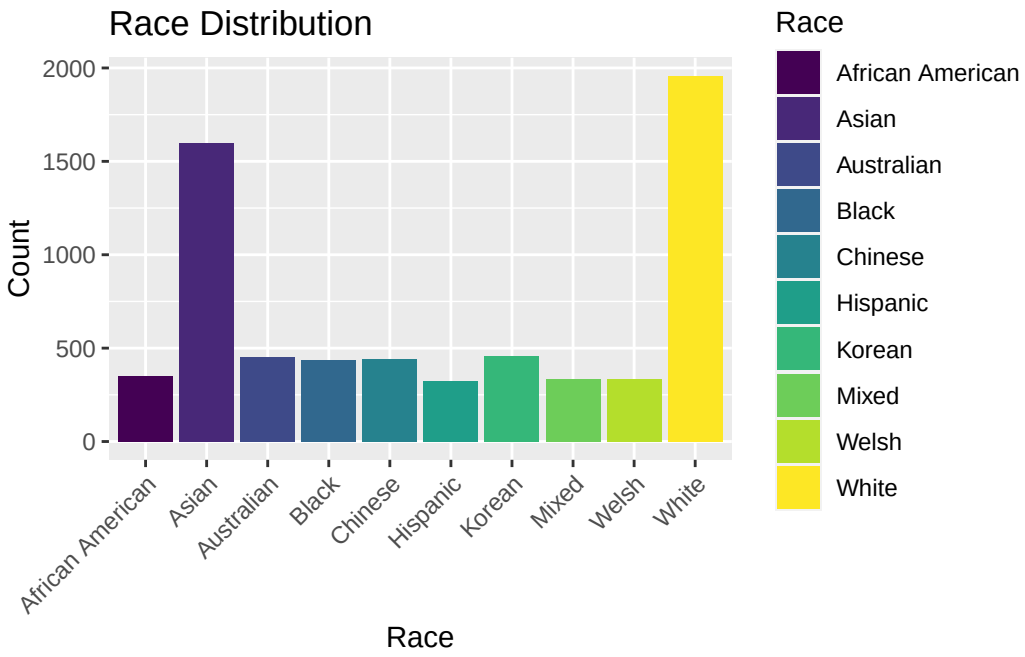
(Alternative Nutzung des Farbschemas):

```
ggplot(salary, aes(x = Country, fill = Country)) +
  geom_bar() +
  scale_fill_viridis(discrete = TRUE) +
  ggtitle("Country Distribution") +
  xlab("Country") +
  ylab("Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



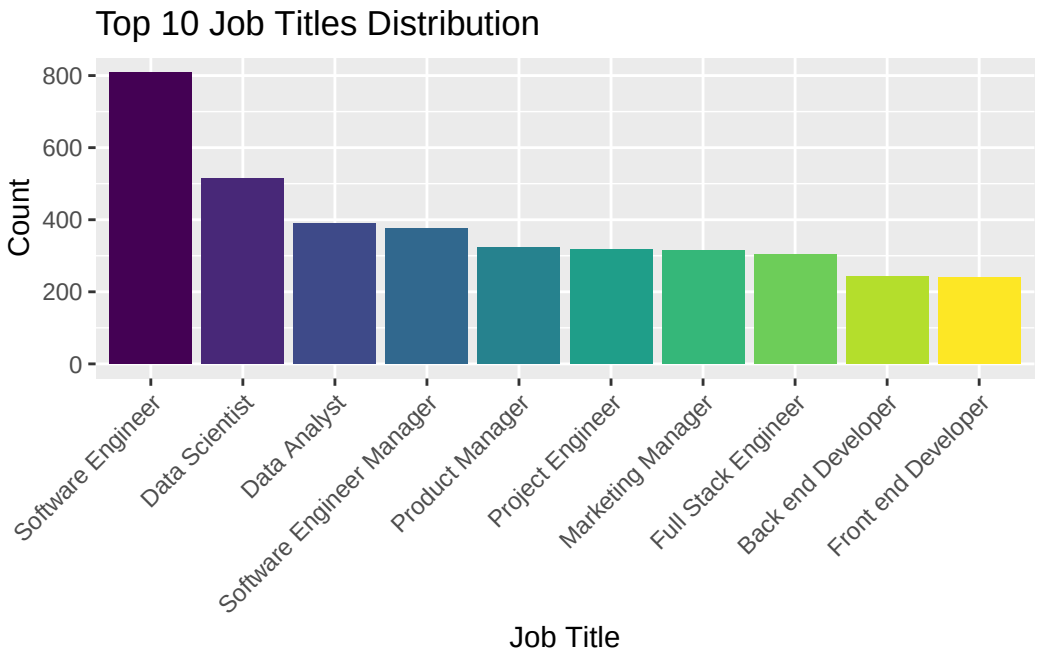
Verteilung der Ethnizitäten:

```
ggplot(salary, aes(x = Race, fill = Race)) +
  geom_bar() +
  scale_fill_viridis(discrete = TRUE) +
  ggtitle("Race Distribution") +
  xlab("Race") +
  ylab("Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Verteilung der 10 häufigsten Job Titel:

```r
# Die Top 10 Jobtitel auswählen
top_job_titles <- names(sort(table(salary$Job.Title), decreasing = TRUE)[1:10])

# Zufällige Farben für jeden Jobtitel generieren
job_colors <- rainbow(length(top_job_titles))

# Daten filtern und ggplot erstellen
ggplot(salary[salary$Job.Title %in% top_job_titles, ], aes(x = factor(Job.Title, levels =
  geom_bar(fill=viridis(10)) +
  scale_fill_manual(values = job_colors) +
  labs(title = "Top 10 Job Titles Distribution",
       x = "Job Title",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



2.2 2.2 Grundlegende Statistische Merkmale des Datensatzes

Zunächst wird mithilfe der Funktion "summary()" ein allgemeiner Überblick über die wichtigsten charakteristischen Merkmale der einzelnen Spalten gegeben.

```
summary(salary)
```

```
      Age            Gender           Education.Level  Job.Title
 Min.   :21.00   Length:6684        Min.   :0.000   Length:6684
 1st Qu.:28.00   Class :character   1st Qu.:1.000   Class :character
 Median :32.00   Mode  :character   Median :1.000   Mode  :character
 Mean   :33.61                      Mean   :1.622
 3rd Qu.:38.00                      3rd Qu.:2.000
 Max.   :62.00                      Max.   :3.000
 Years.Of.Experience     Salary         Country              Race
 Min.   : 0.000      Min.   :   350   Length:6684        Length:6684
 1st Qu.: 3.000      1st Qu.: 70000   Class :character   Class :character
 Median : 7.000      Median :115000   Mode  :character   Mode  :character
 Mean   : 8.078      Mean   :115307
 3rd Qu.:12.000      3rd Qu.:160000
 Max.   :34.000      Max.   :250000
```

```
     Senior
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.1435
 3rd Qu.:0.0000
 Max.   :1.0000
```

In einer ersten Analyse zeigt sich, dass der durchschnittliche Alterswert im Datensatz bei 32 Jahren liegt. Die Altersspanne erstreckt sich dabei von 21 bis 62 Jahren. Bezüglich des "Education-Level" variieren die Werte zwischen 1, 2 und 3, wobei der Durchschnitt bei 1 liegt. In Bezug auf die Berufserfahrung ("Years of Experience") reichen die Werte von 0 bis zu 34 Jahren, wobei der Median hier bei 7 liegt. Bei Auswertung der Gehaltsdaten ("Salary") ergibt sich, dass das Mindestgehalt bei 350 Dollar, das Höchstgehalt bei 250.000 Dollar und der Median bei 115.000 Dollar liegt.

3 3. Umstrukturierung des Datensatzes für eine verbesserte Visualisierung

Im folgenden wird der Datensatz temporär umstrukturiert, um den Datensatz besser analysieren und visualieren zu können.

Ein neuer Wert Namens "Value" wird erschaffen.

```
Salary_long <- select(salary, -Job.Title, -Gender, -Race, -Country, -Senior)
Salary_long <- pivot_longer(Salary_long, colnames(Salary_long))
Salary <- as.data.frame(Salary_long)
head(Salary_long)
```

```
# A tibble: 6 x 2
  name                value
  <chr>               <dbl>
1 Age                    32
2 Education.Level         1
3 Years.Of.Experience     5
4 Salary              90000
5 Age                    28
6 Education.Level         2
```

In diesem Codeabschnitt werden sämtliche Spalten, die keine numerischen Merkmale repräsentieren, entfernt. Der verbleibende Datensatz wird anschließend von einem breiten Format in ein längeres umgewandelt. Diese Transformation ermöglicht eine effizientere Analyse und Visualisierung der numerischen Daten, indem sie sie in einer strukturierteren und leichter interpretierbaren Form darstellt.

Hier kann man folgende Dinge erkennen:

- Age, Years of Experience und Education Level sind Linksschief und haben ggf. Bedarf einer Transformation für ML-Modelle
- Age und Years of Experience haben Extrempunkte im oberen Wertebereich, während Salary einer gleichmäßigen Verteilung folgt

Aufgrund der sorgfältigen Strukturierung der Daten, scheinen sie auf den ersten Blick gut für eine umfassende explorative Analyse geeignet zu sein.

3.1 3.1 Kategorisierung von Gehalt

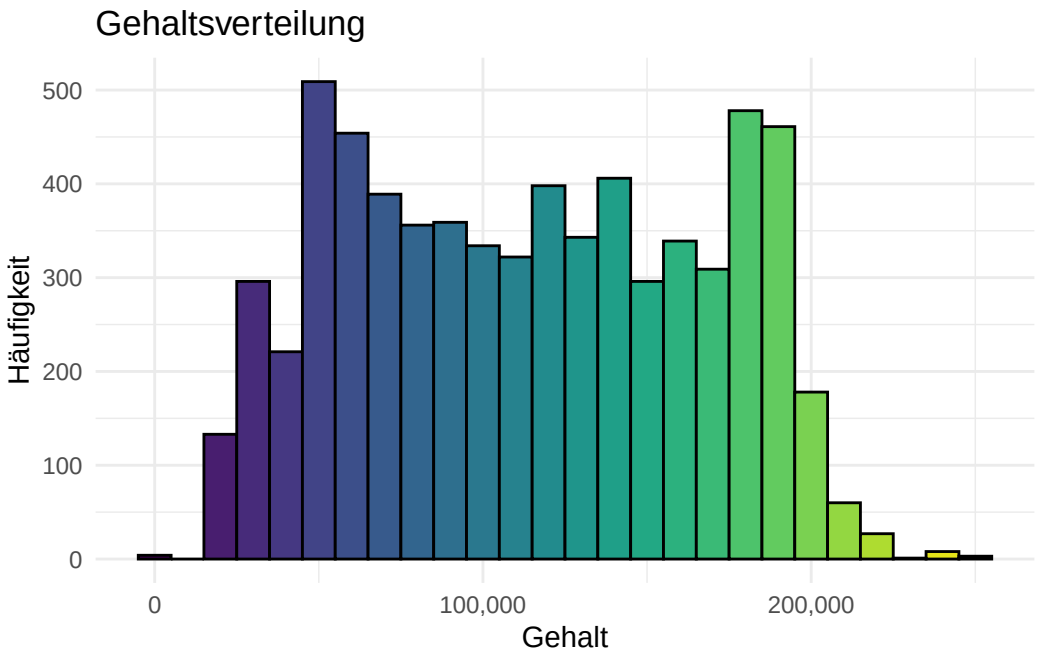
Zunächst werden die Daten aus dem Ausgangsdatensatz in einen finalen Datensatz "salary_final" geschrieben.

```
salary_final <- salary
```

Durch den Befehl "hist()" wird ein Histogramm erstellt. Es ermöglicht eine visuelle Darstellung der Häufigkeitsverteilung dieser Gehaltsdaten, indem es zeigt, wie oft bestimmte Gehaltsbereiche vorkommen.

Verteilung des Gehalts:

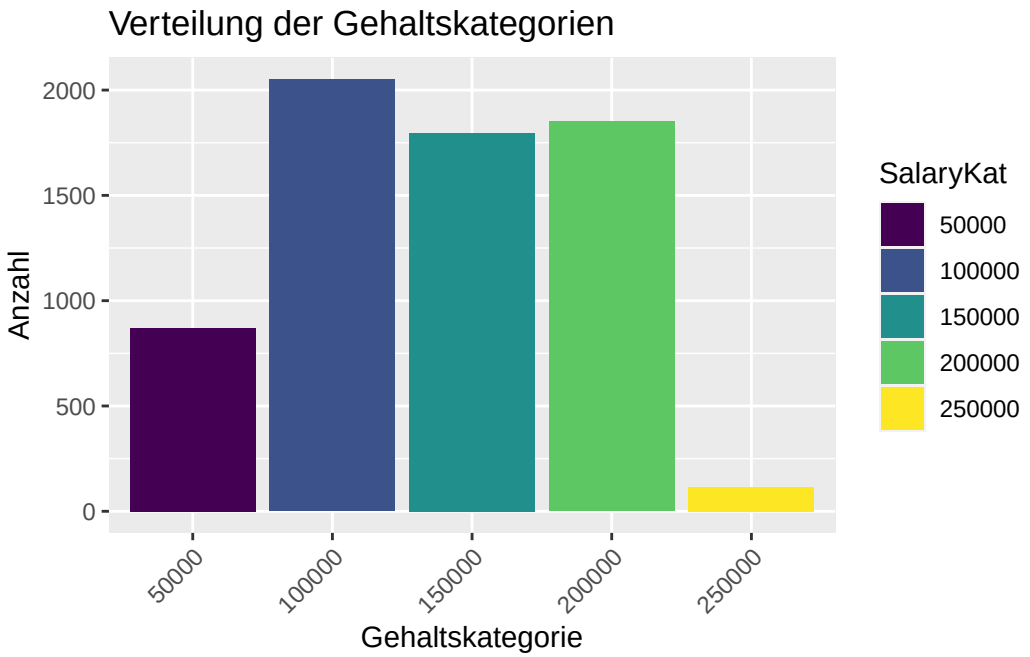
```
ggplot(salary, aes(x = Salary)) +
  geom_histogram(binwidth = 10000, fill = viridis(26), color = "black") +
  labs(title = "Gehaltsverteilung",
       x = "Gehalt",
       y = "Häufigkeit") +
  scale_y_continuous(labels = scales::comma) +
  scale_x_continuous(labels = scales::comma) +
  theme_minimal()
```



Im weiteren Verlauf wird eine neue Spalte namens “SalaryKat” erstellt, die kategorische Werte basierend auf den Gehältern enthält. Darauf aufbauend wird ein Balkendiagramm für die 5 Gehaltskategorien erstellt. Diese visuelle Darstellung ermöglicht eine effektive Einsicht in die Verteilung der Gehälter und erleichtert die Interpretation der Daten in übersichtlicher Form.

```
salary_final$SalaryKat <- cut(salary_final$Salary,
                  breaks = c(-Inf, 50000, 100000, 150000, 200000, 250000, Inf),
```

```
ggplot(salary_final, aes(x = SalaryKat, fill = SalaryKat)) +
  geom_bar() +
  scale_fill_viridis(discrete = TRUE) +
  ggtitle("Verteilung der Gehaltskategorien") +
  xlab("Gehaltskategorie") +
  ylab("Anzahl") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Die Analyse der erstellten Gehaltskategorie zeigt, dass die Kategorie mit einem Gehalt von 100.000 am häufigsten vertreten ist.

3.2 3.2 Korrelationen

Im weiteren Verlauf erfolgt die Berechnung der Korrelationen zwischen den verschiedenen Spalten.

Diese Vorgehensweise ermöglicht es, die Stärke und Richtung des Zusammenhangs zwischen zwei Variablen zu quantifizieren. Die Anwendung von Korrelationsanalysen spielt eine entscheidende Rolle bei der Modellvalidierung, da sie dabei hilft, potenzielle Probleme wie beispielsweise Multikollinearität zu identifizieren.

Die folgenden Korrelationen werden nun berechnet:

```
# Korrelation zwischen Salary und Years.Of.Experience berechnen
correlation_salary_experience <- cor(salary_final$Salary, salary_final$Years.Of.Experience

# Ausgabe des Ergebnisses
cat("Die Korrelation zwischen Salary und Years.Of.Experience ist:", correlation_salary_exp
```

```
Die Korrelation zwischen Salary und Years.Of.Experience ist: 0.8109416
```

```r
# Korrelation zwischen Salary und Age berechnen
correlation_salary_age <- cor(salary_final$Salary, salary_final$Age, use = "complete.obs")

# Ausgabe des Ergebnisses
cat("Die Korrelation zwischen Salary und Age ist:", correlation_salary_age, "\n")
```

```
Die Korrelation zwischen Salary und Age ist: 0.7283429
```

```r
# Korrelation zwischen Years.Of.Experience und Age berechnen
correlation_experience_age <- cor(salary_final$Years.Of.Experience, salary_final$Age, use

# Ausgabe des Ergebnisses
cat("Die Korrelation zwischen Years.Of.Experience und Age ist:", correlation_experience_ag
```

```
Die Korrelation zwischen Years.Of.Experience und Age ist: 0.9376094
```

```r
# Korrelation zwischen Seniority und Years.Of.Experience berechnen
correlation_seniority_experience <- cor(salary_final$Senior, salary_final$Years.Of.Experie

# Ausgabe des Ergebnisses
cat("Die Korrelation zwischen Seniority und Years.Of.Experience ist:", correlation_seniori
```

```
Die Korrelation zwischen Seniority und Years.Of.Experience ist: 0.3178772
```

Die Ergebnisse der Korrelationen:

- Von salary und years.of.experience ist es 0.81.
- Von Salary und Age ist es 0.73
- von Age und Years of Experience st es 0.93.

Nun stellt sich folgende Frage:

Wie entsteht bei den Werten so ein starker Unterschied, im Vergleich zu Salary, obwohl eine so hohe Korrelation zueinander besteht?

Lösungsansätze:

Verteilung der Daten: Es ist möglich, dass die Verteilung der Daten in den Variablen “Age” und “Years.Of.Experience” anders ist als in der Variable “Salary”. Wenn die Daten in “Age”

und “Years.Of.Experience” breiter gestreut sind, kann dies zu einer geringeren Korrelation führen, selbst wenn eine starke lineare Beziehung besteht.

Nicht-lineare Beziehung: Die Korrelation misst nur lineare Beziehungen. Wenn die Beziehung zwischen “Age” und “Years.Of.Experience” nicht linear ist, könnte dies zu einem niedrigeren Korrelationswert führen.

Ausreißer: Das Vorhandensein von Ausreißern kann die Korrelation beeinflussen. Wenn es Ausreißer in einer der Variablen gibt, kann dies den Korrelationswert beeinträchtigen.

Stichprobengröße: Bei kleineren Stichproben können Korrelationswerte instabiler sein.

4 4. Tests für die Thesen

Im weiteren Verlauf werden anhand der vorliegenden Daten verschiedene Tests durchgeführt, um Aussagen für die aufgestellten Thesen herauszufiltern. Dieser Prozess wird sowohl durch die Visualisierung der Beziehungen zwischen den verschiedenen Spalten als auch durch die Berechnung von Korrelationen unterstützt. Die Kombination dieser Methoden ermöglicht eine umfassende Analyse, die dazu beiträgt, Muster, Trends und potenzielle Zusammenhänge zwischen den betrachteten Variablen zu erkennen.

4.1 4.1 Korrelationen

Das Ergebnis dieses Codechunks ist eine Darstellung der Korrelationsmatrix:

```
correlations <- cor(salary_final[, c("Age", "Education.Level", "Years.Of.Experience", "Sal

print(correlations)
```

```
                          Age Education.Level Years.Of.Experience    Salary
Age                 1.0000000       0.5963804           0.9376094 0.7283429
Education.Level     0.5963804       1.0000000           0.6131650 0.6454436
Years.Of.Experience 0.9376094       0.6131650           1.0000000 0.8109416
Salary              0.7283429       0.6454436           0.8109416 1.0000000
```

Erkennbar ist eine starke Korrelation zwischen dem Alter und den “Years of Experience”. Des Weiteren besteht ebenfalls eine ausgeprägte Korrelation zwischen den Years of Experience und dem endgültigen Gehalt. Hingegen liegt die Korrelation zwischen dem Alter und dem Bildungsniveau mit einem Wert von ungefähr 0,6 nicht ganz so stark vor.

```r
filtered_data_numeric <- select(salary, Salary, Age, Years.Of.Experience, Education.Level)
glimpse(filtered_data_numeric)
```

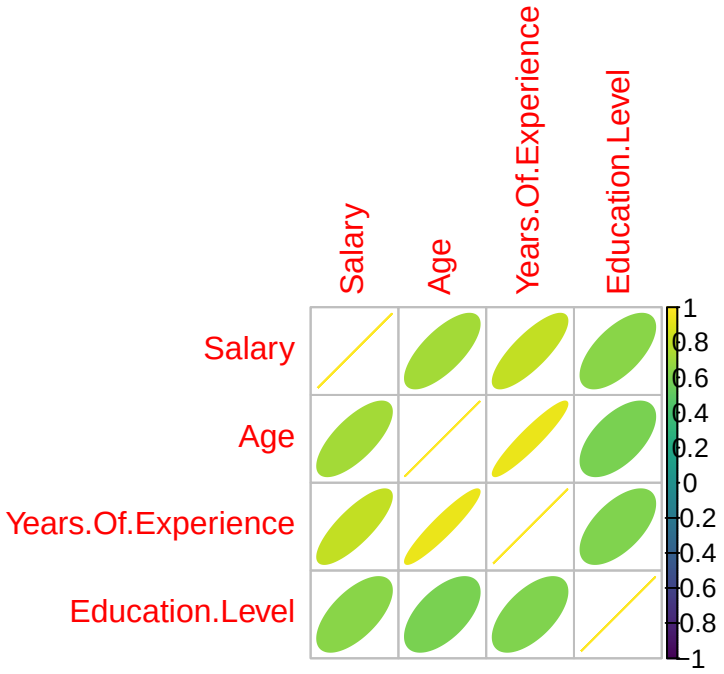
```
Rows: 6,684
Columns: 4
$ Salary               <dbl> 90000, 65000, 150000, 60000, 200000, 55000, 120000~
$ Age                  <dbl> 32, 28, 45, 36, 52, 29, 42, 31, 26, 38, 29, 48, 35~
$ Years.Of.Experience <dbl> 5, 3, 15, 7, 20, 2, 12, 4, 1, 10, 3, 18, 6, 14, 2,~
$ Education.Level      <dbl> 1, 2, 3, 1, 2, 1, 2, 1, 1, 3, 2, 1, 1, 2, 1, 1, 2,~
```

```r
cor(filtered_data_numeric)
```

```
                     Salary       Age Years.Of.Experience Education.Level
Salary            1.0000000 0.7283429           0.8109416       0.6454436
Age               0.7283429 1.0000000           0.9376094       0.5963804
Years.Of.Experience 0.8109416 0.9376094           1.0000000       0.6131650
Education.Level   0.6454436 0.5963804           0.6131650       1.0000000
```

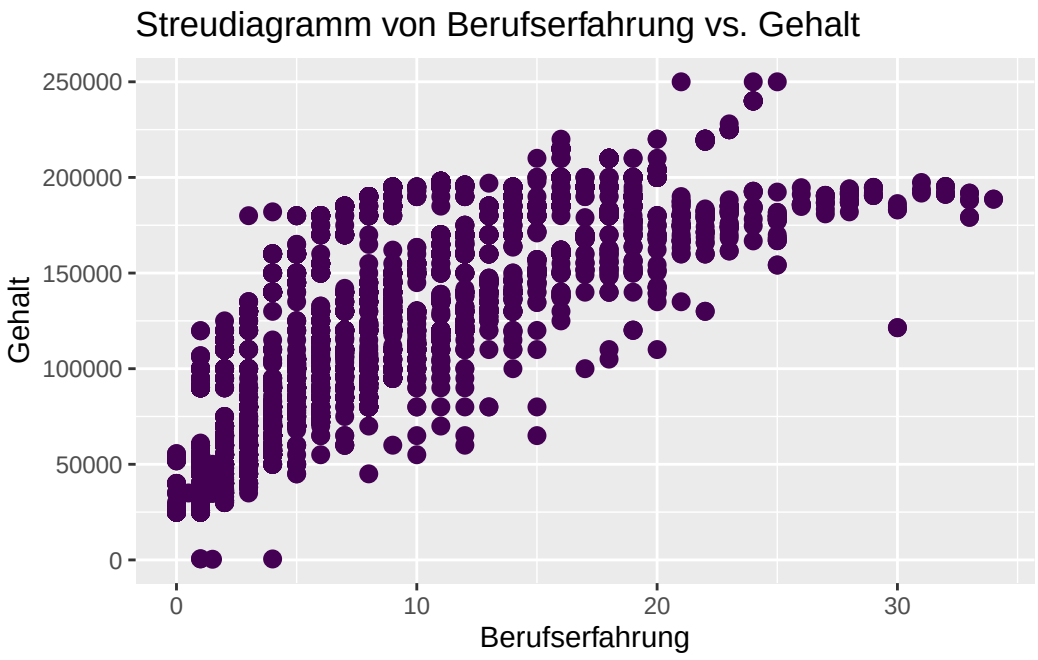
Nun wird der Korrelationsplot erstellt.

```r
corrplot(cor(filtered_data_numeric), method = "ellipse", col = viridis(200))
```



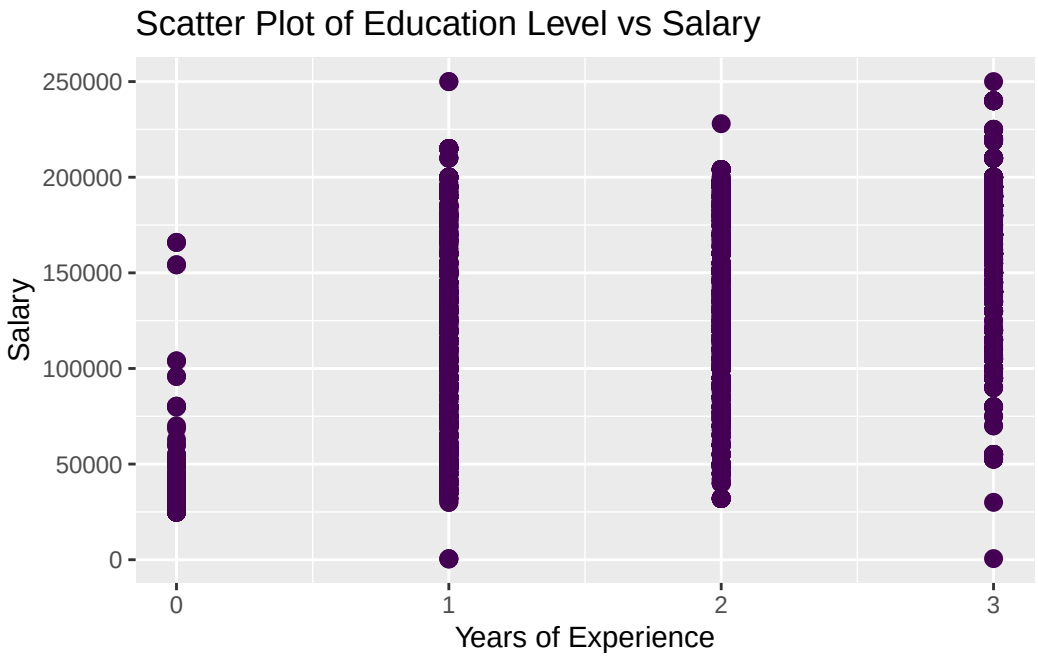
4.2 4.2 Streudiagramme

```
ggplot(salary_final, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point(color = viridis(2)[1], size = 3, shape = 16) +
  labs(title = "Streudiagramm von Berufserfahrung vs. Gehalt",
       x = "Berufserfahrung",
       y = "Gehalt")
```



Das Streudiagramm visualisiert deutlich einen eindeutigen aufsteigenden Trend, der mit einer zunehmenden Anzahl von "Years of Experience" einhergeht. Es wird ebenfalls sichtbar, dass die Höchstgehälter von 250.000€ im Bereich von 20 bis 30 Jahren Berufserfahrung konzentriert sind.

```
ggplot(salary_final, aes(x = Education.Level, y = Salary)) +
  geom_point(color = viridis(2)[1], size = 3, shape = 16) +
  labs(title = "Scatter Plot of Education Level vs Salary",
       x = "Years of Experience",
       y = "Salary")
```



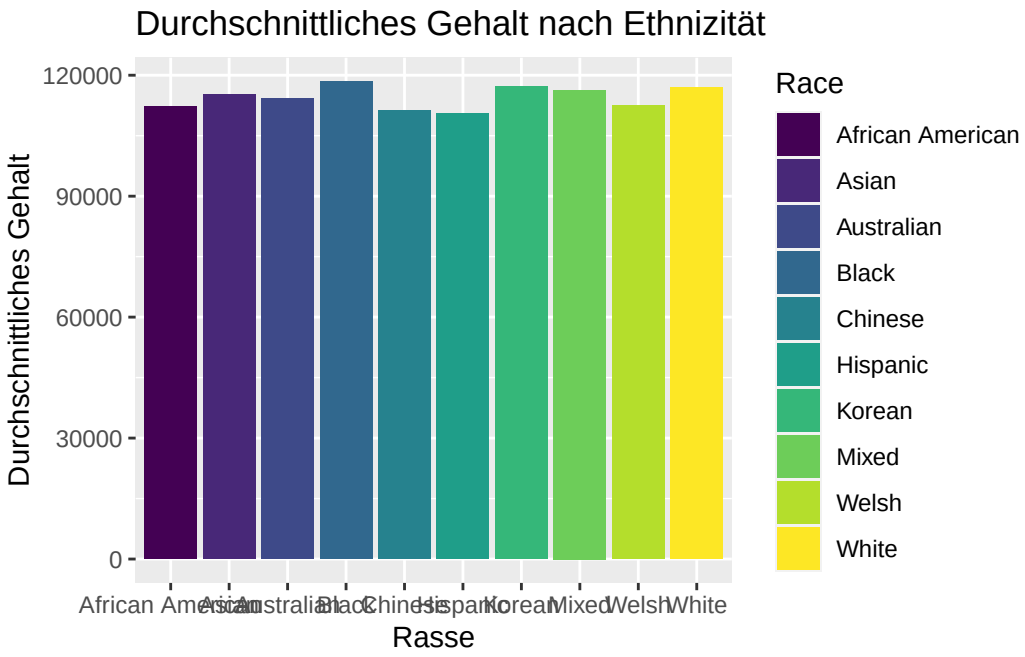
Die vorliegende Datenanalyse verdeutlicht einen klaren Trend zu höheren Gehaltsklassen, der mit einem Anstieg des Bildungsniveaus einhergeht. Diese Tendenz wird durch eine erhöhte Dichte in den oberen Gehaltsgruppen für Personen mit dem dritten Bildungsgrad im Vergleich zum zweiten und ersten Bildungsgrad deutlich.

4.3 4.3 Balkendiagramme mit 2 Variablen

In diesem Abschnitt werden Balkendiagramme verwendet, um den Datensatz auf Beziehungen zu analysieren. Die visuelle Darstellung durch Balkendiagramme ermöglicht uns eine anschauliche Interpretation der Daten.

4.3.1 4.3.1 Average Salary by Race

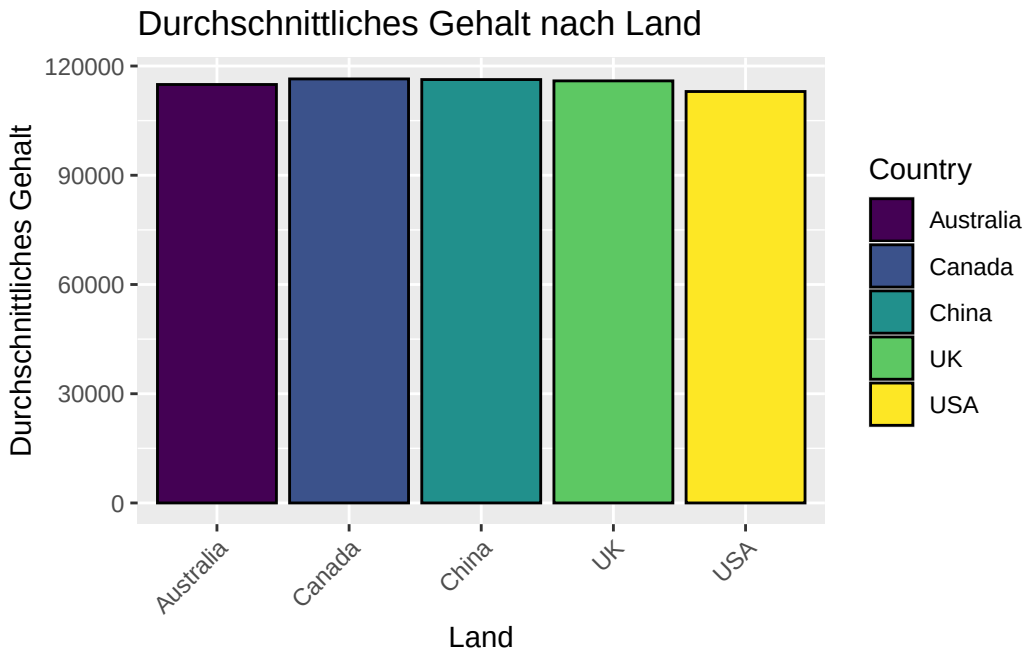
```
ggplot(salary_final, aes(x = Race, y = Salary, fill = Race)) +
  stat_summary(fun = "mean", geom = "bar") +
  scale_fill_viridis(discrete = TRUE) +
  ggtitle("Durchschnittliches Gehalt nach Ethnizität") +
  xlab("Rasse") +
  ylab("Durchschnittliches Gehalt")
```



Die Grafik macht deutlich, dass die Gruppen “Black, Korean, Mixed und White” im Durchschnitt am das höchste Einkommen erzielen.

4.3.2 4.3.2 Average Salary by Country

```
ggplot(salary_final, aes(x = Country, y = Salary, fill = Country)) +
  stat_summary(fun = "mean", geom = "bar", position = "dodge", color = "black") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Durchschnittliches Gehalt nach Land",
       x = "Land",
       y = "Durchschnittliches Gehalt") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



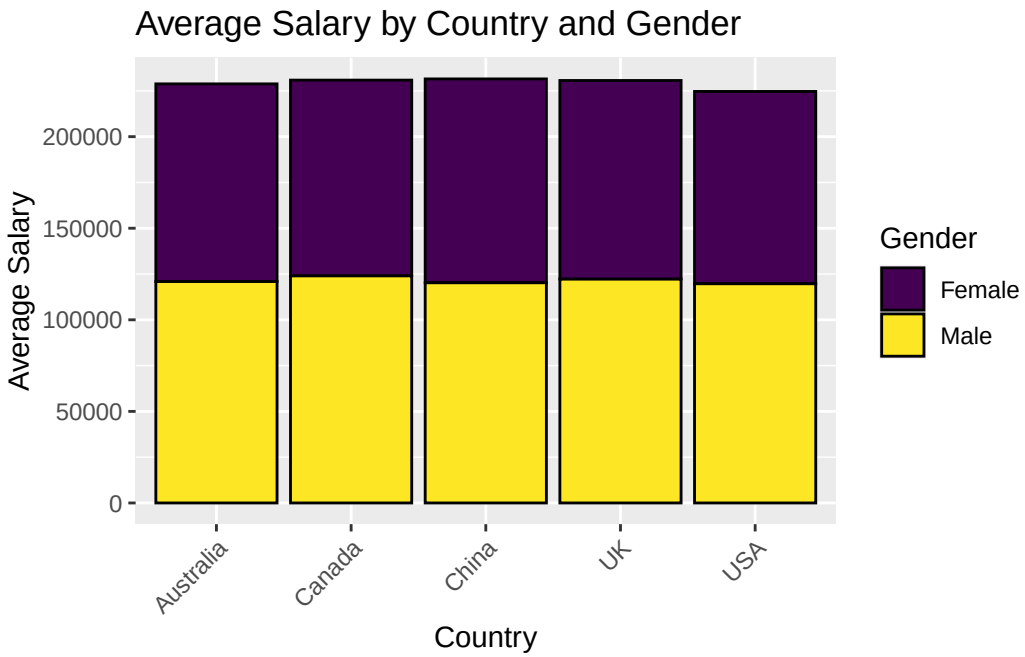
Es ist ersichtlich, dass in den Ländern "Canada und China" das durchschnittliche Gehalt am größten ist. Jedochg ist zu erwähnen, dass alle Länder nah bei einander liegen.

Es ist ersichtlich, dass in den Ländern "Canada und China" das durchschnittliche Gehalt am höchsten ist. Es ist jedoch zu betonen, dass die durchschnittlichen Gehälter in allen Ländern nahe beieinander liegen.

4.3.3 4.3.3 Average Salary by Country and Gender

Die Erstellung eines gestapelten Balkendiagramms ermöglicht uns einen detaillierten Vergleich der durchschnittlichen Gehälter zwischen verschiedenen Ländern und Geschlechtern. Die Balken sind entsprechend nach Geschlecht gruppiert und gestapelt, was eine übersichtliche Darstellung der Unterschiede in den Gehaltsstrukturen zwischen den betrachteten Gruppen ermöglicht.

```
ggplot(salary_final, aes(x = Country, y = Salary, fill = Gender)) +
  geom_bar(stat = "summary", fun = "mean", position = "stack", color = "black") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Average Salary by Country and Gender",
       x = "Country",
       y = "Average Salary") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



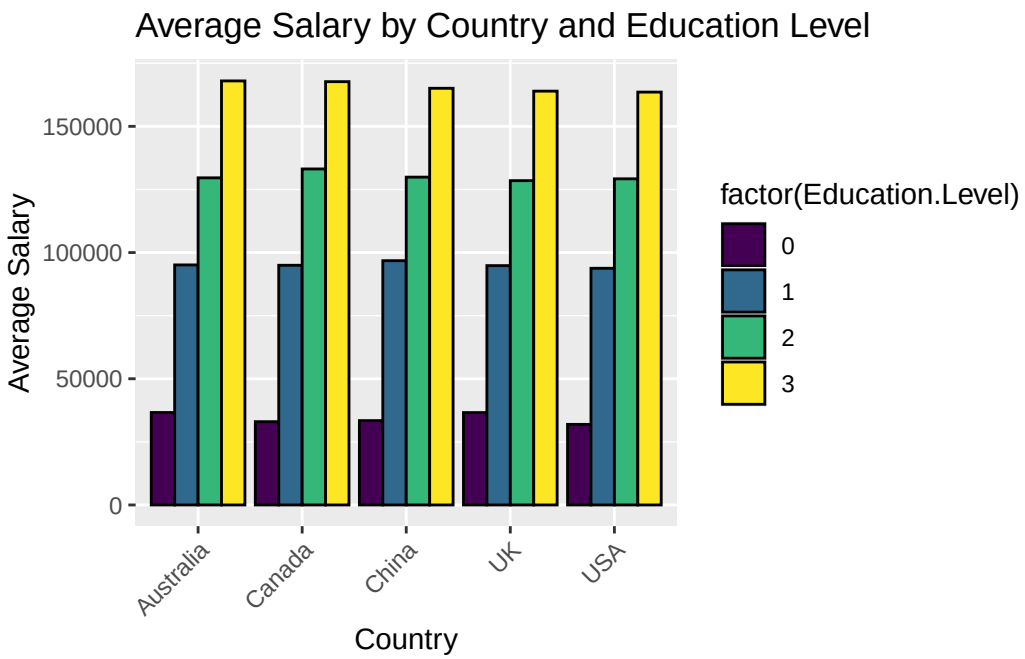
Hier ist zu erkennen, dass alle Länder eine ähnliche Verteilung zwischen den Geschlechtern "Male" und "Female" aufweisen.

4.3.4 4.3.4 Average Salary by Country and Education Level

Zur Visualisierung der durchnittlichen Bezahlung für Länder und Bildungsniveau wird ein gruppiertes Balkendiagramm verwendet. Die Balken sind hierbei nach Bildungsniveau gruppiert.

Zusätzlich wurden die Beschriftungen auf der X-Achse um 45 Grad gedreht, um eine verbesserte Veranschaulichung zu gewährleisten.

```
ggplot(salary_final, aes(x = Country, y = Salary, fill = factor(Education.Level))) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge", color = "black") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Average Salary by Country and Education Level",
       x = "Country",
       y = "Average Salary") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

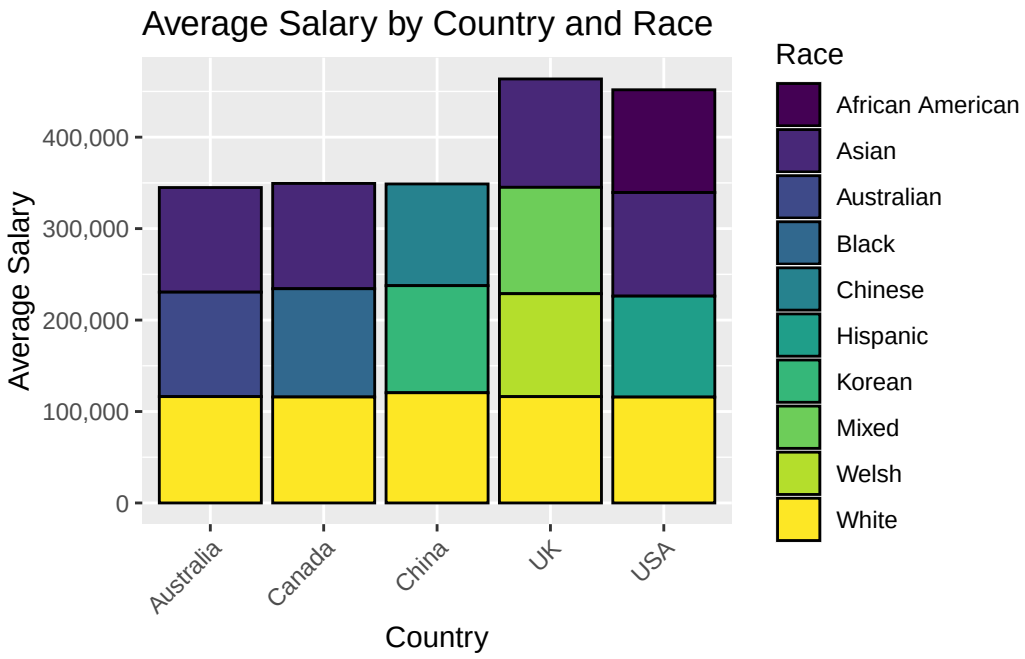


Es ist deutlich zu erkennen, dass die Gehälter in jedem Land deutlich ansteigen, je höher das Bildungsniveau ist.

4.3.5 4.3.5 Average Salary by Country and Race

Auch hier wird zum Vergleich der durchschnittlichen Gehälter zwischen den Ländern und ethnischen Gruppen, ein gestapeltes Balkendiagramm erstellt. Hierbei sind die Balken nach ethnischer Gruppe gruppiert und gestapelt.

```
ggplot(salary_final, aes(x = Country, y = Salary, fill = Race)) +
  geom_bar(stat = "summary", fun = "mean", position = "stack", color = "black") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Average Salary by Country and Race",
       x = "Country",
       y = "Average Salary") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)
```

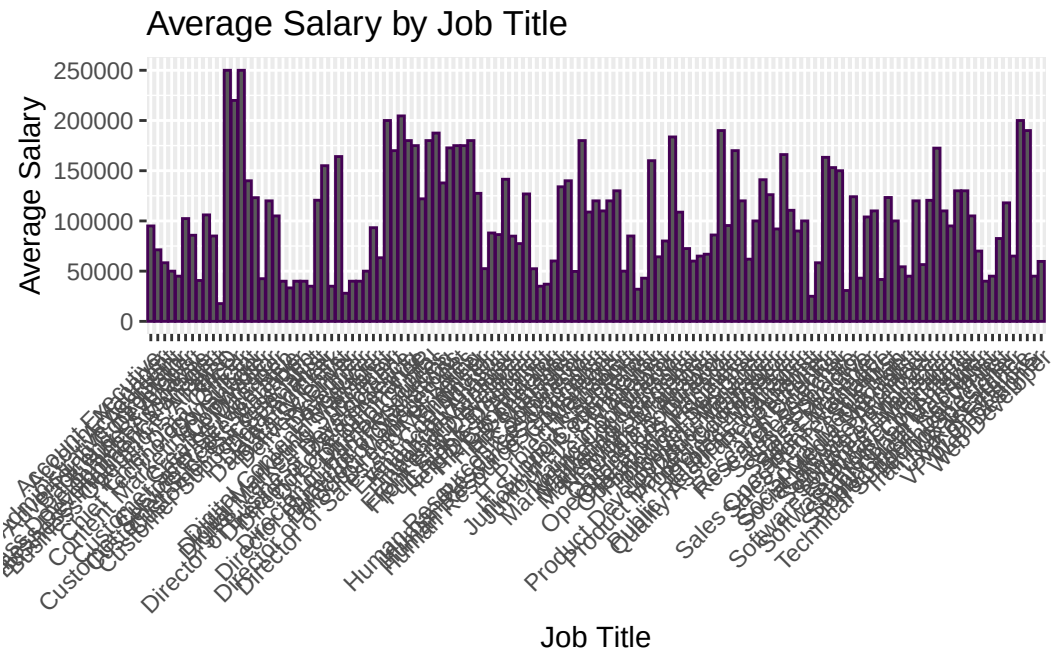


Die Grafik verdeutlicht, dass nicht zwangsläufig in jedem Land alle ethnischen Gruppen vertreten sind. Es ist ersichtlich, dass nur in 2 Ländern mehr als 3 verschiedene ethnische Gruppen in diesem Datensatz aufgeführt sind.

4.3.6 4.3.6 Average Salary by Job Title

```
ggplot(salary_final, aes(x = Job.Title, y = Salary)) +
  geom_bar(stat = "summary", fun = "mean", color = viridis(2)[1], color = viridis(2)[1]) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Average Salary by Job Title",
       x = "Job Title",
       y = "Average Salary") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
Warning: Duplicated aesthetics after name standardisation: colour
```



Hier wird festgestellt, dass in dem vorliegenden Datensatz eine hohe Anzahl von verschiedenen Jobtiteln existiert:

```
job_title_count <- table(salary_final$Job.Title)
print(job_title_count)
```

```
                Account Executive                 Account Manager
                                1                               4
                       Accountant          Administrative Assistant
                                6                               2
          Advertising Coordinator                Back end Developer
                                1                             242
                  Business Analyst     Business Development Associate
                               20                               7
      Business Development Manager      Business Intelligence Analyst
                                5                               1
       Business Operations Analyst                             CEO
                                2                               1
              Chief Data Officer           Chief Technology Officer
                                1                               1
                      Consultant          Content Marketing Manager
                                1                              73
```

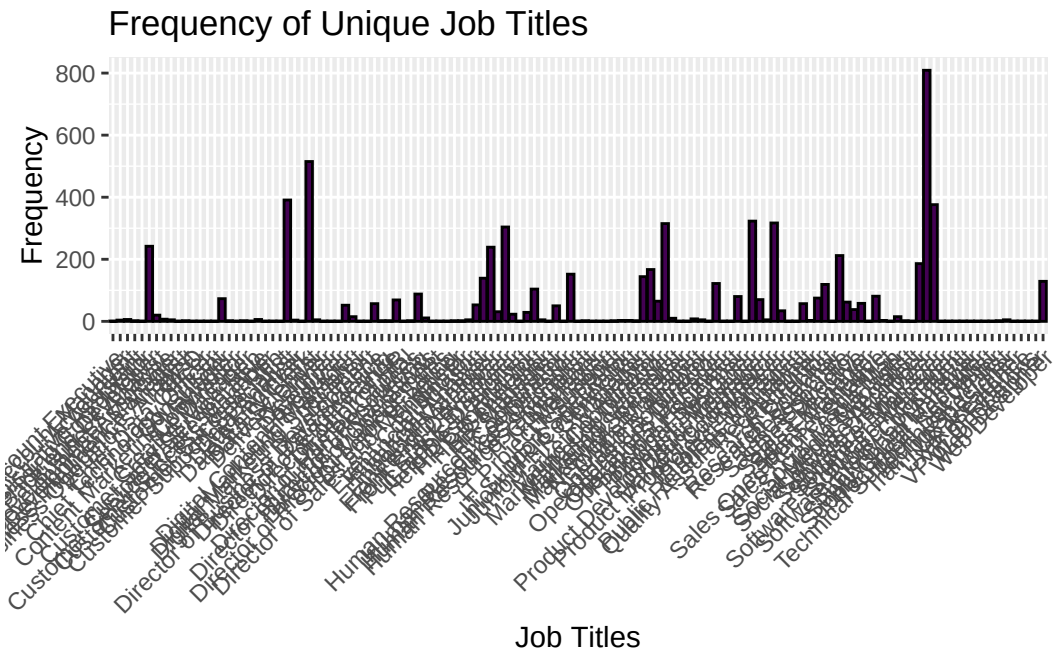
Copywriter	Creative Director
2	1
Customer Service Manager	Customer Service Rep
2	1
Customer Service Representative	Customer Success Manager
6	1
Customer Success Rep	Customer Support Specialist
1	1
Data Analyst	Data Engineer
391	4
Data Entry Clerk	Data Scientist
1	515
Delivery Driver	Designer
5	1
Developer	Digital Content Producer
1	1
Digital Marketing Manager	Digital Marketing Specialist
52	15
Director	Director of Business Development
1	1
Director of Data Science	Director of Engineering
57	2
Director of Finance	Director of HR
2	69
Director of Human Capital	Director of Human Resources
1	2
Director of Marketing	Director of Operations
88	11
Director of Product Management	Director of Sales
1	1
Director of Sales and Marketing	Engineer
1	2
Event Coordinator	Financial Advisor
2	5
Financial Analyst	Financial Manager
53	139
Front end Developer	Front End Developer
239	31
Full Stack Engineer	Graphic Designer
304	23
Help Desk Analyst	HR Coordinator
1	29
HR Generalist	HR Manager

104	5
HR Specialist	Human Resources Coordinator
1	50
Human Resources Director	Human Resources Manager
1	152
Human Resources Specialist	IT Consultant
1	2
IT Manager	IT Project Manager
1	1
IT Support	IT Support Specialist
1	2
Juniour HR Coordinator	Juniour HR Generalist
3	3
Manager	Marketing Analyst
2	144
Marketing Coordinator	Marketing Director
167	65
Marketing Manager	Marketing Specialist
315	10
Network Engineer	Office Manager
1	1
Operations Analyst	Operations Coordinator
8	5
Operations Director	Operations Manager
1	122
Principal Engineer	Principal Scientist
1	1
Product Designer	Product Development Manager
80	1
Product Manager	Product Marketing Manager
323	70
Project Coordinator	Project Engineer
5	317
Project Manager	Public Relations Manager
34	1
Quality Assurance Analyst	Receptionist
1	57
Recruiter	Research Director
3	75
Research Scientist	Researcher
119	1
Sales Associate	Sales Director
212	62

```
                Sales Executive                  Sales Manager
                             38                             58
      Sales Operations Manager            Sales Representative
                              1                             81
                      Scientist                Social Media Man
                              3                              1
           Social Media Manager           Social Media Specialist
                             15                              2
            Software Architect                Software Developer
                              1                            186
             Software Engineer          Software Engineer Manager
                            809                            376
              Software Manager           Software Project Manager
                              1                              1
            Strategy Consultant               Supply Chain Analyst
                              1                              1
           Supply Chain Manager                Technical Recruiter
                              1                              1
   Technical Support Specialist                   Technical Writer
                              1                              1
            Training Specialist                      UX Designer
                              2                              5
                  UX Researcher                  VP of Finance
                              1                              1
              VP of Operations                    Web Designer
                              1                              1
                 Web Developer
                            129
```

Hier nochmal umfangreicher grafisch ausgearbeitet:

```r
job_title_count <- table(salary_final$Job.Title)
job_title_df <- data.frame(Job_Title = names(job_title_count), Frequency = as.numeric(job_

ggplot(job_title_df, aes(x = Job_Title, y = Frequency)) +
  geom_bar(stat = "identity", fill = viridis(2)[1], color = "black") +
  labs(title = "Frequency of Unique Job Titles",
       x = "Job Titles",
       y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Jeder Job wurde aufgeführt. Es ist anzumerken, dass es aufgrund der enormen Vielfalt an unterschiedlichen Jobs nicht möglich ist, sämtliche in der Darstellung zu berücksichtigen.

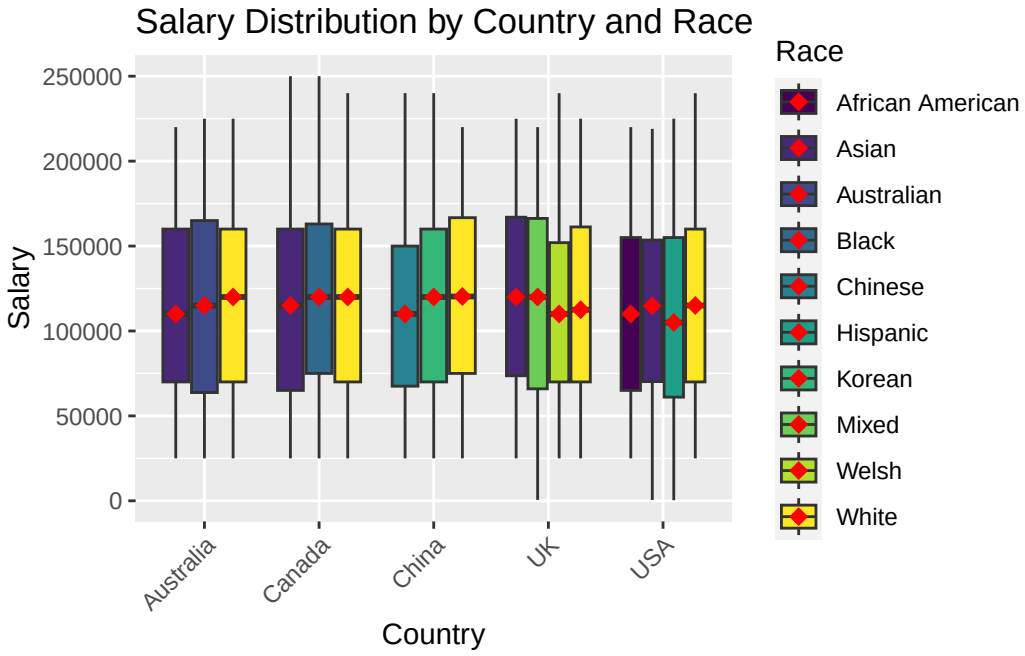
```
library(dplyr)

job_title_count <- salary %>%
  count(`Job.Title`, sort = TRUE)
job_title_count
```

```
# A tibble: 129 x 2
   Job.Title                      n
   <chr>                      <int>
 1 Software Engineer            809
 2 Data Scientist               515
 3 Data Analyst                 391
 4 Software Engineer Manager    376
 5 Product Manager              323
 6 Project Engineer             317
 7 Marketing Manager            315
 8 Full Stack Engineer          304
 9 Back end Developer           242
10 Front end Developer          239
```

```
# i 119 more rows
```

4.4 4.4 Boxplots

```
ggplot(salary_final, aes(x = Country, y = Salary, fill = Race)) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE) +
  stat_summary(fun = "median", geom = "point", shape = 18, size = 3, color = "red", positi
  labs(title = "Salary Distribution by Country and Race",
       x = "Country",
       y = "Salary") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



5 5. Daten Aufbereiten

(Tests der Thesen gibt uns die Antwort darauf wie die Daten aufbereitet werden müssen)

5.1 5.1 Jobs

Da in dem Datensatz teilweise Jobs nur einmalig vertreten sind, kann ein erhebliches Stichproben-Bias verursacht werden. Da das mittlere Einkommen ein wichtiges Merkmal in unserer explorativen Datenanalyse darstellt und mindestens 30 Einträge für eine aussagekräftige Stichprobe nötig sind, haben wir uns dazu entschlossen alle Einträge mit N<30 bei der Anzahl der Jobtitel (N) abzuschneiden.

```
filtered_data <- salary_final %>%
  group_by(Job.Title) %>%
  summarise(job_count = n()) %>%
  filter(job_count > 30) %>%
  inner_join(salary_final, by = "Job.Title")

print(filtered_data)
```

```
# A tibble: 6,398 x 11
   Job.Title   job_count   Age Gender Education.Level Years.Of.Experience Salary
   <chr>           <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
 1 Back end D~       242    33 Female               2                    5 110000
 2 Back end D~       242    32 Male                 1                    4  95000
 3 Back end D~       242    26 Female               2                    3  90000
 4 Back end D~       242    26 Female               2                    2  70000
 5 Back end D~       242    24 Female               1                    1  60000
 6 Back end D~       242    26 Female               2                    3  90000
 7 Back end D~       242    24 Female               2                    1  60000
 8 Back end D~       242    34 Male                 2                    6 125000
 9 Back end D~       242    29 Female               1                    3  85000
10 Back end D~       242    23 Male                 1                    1  55000
# i 6,388 more rows
# i 4 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>
```

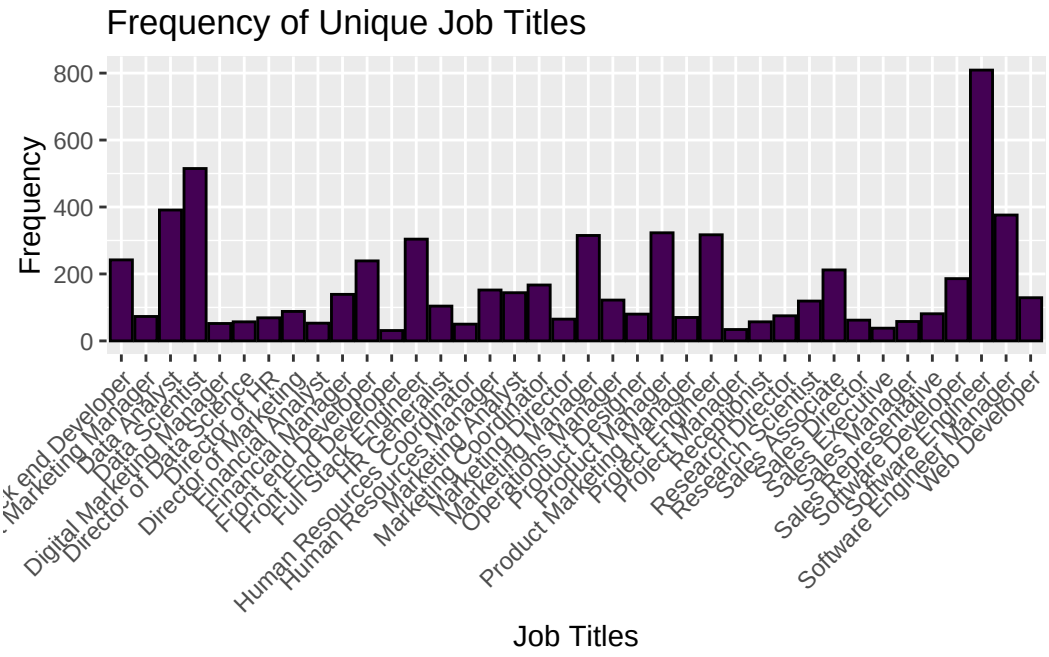
```
job_title_count <- table(filtered_data$Job.Title)
print(job_title_count)
```

```
        Back end Developer   Content Marketing Manager
                       242                          73
             Data Analyst             Data Scientist
                       391                         515
  Digital Marketing Manager    Director of Data Science
```

```
                        52                         57
            Director of HR       Director of Marketing
                        69                         88
          Financial Analyst          Financial Manager
                        53                        139
       Front end Developer        Front End Developer
                       239                         31
       Full Stack Engineer             HR Generalist
                       304                        104
Human Resources Coordinator    Human Resources Manager
                        50                        152
          Marketing Analyst      Marketing Coordinator
                       144                        167
         Marketing Director          Marketing Manager
                        65                        315
        Operations Manager           Product Designer
                       122                         80
           Product Manager  Product Marketing Manager
                       323                         70
          Project Engineer            Project Manager
                       317                         34
              Receptionist          Research Director
                        57                         75
       Research Scientist            Sales Associate
                       119                        212
            Sales Director            Sales Executive
                        62                         38
             Sales Manager       Sales Representative
                        58                         81
       Software Developer          Software Engineer
                       186                        809
 Software Engineer Manager             Web Developer
                       376                        129
```

```r
job_title_count <- table(filtered_data$Job.Title)
job_title_df <- data.frame(Job_Title = names(job_title_count), Frequency = as.numeric(job_

ggplot(job_title_df, aes(x = Job_Title, y = Frequency)) +
  geom_bar(stat = "identity", fill = viridis(2)[1], color = "black") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Frequency of Unique Job Titles",
       x = "Job Titles",
```

```
       y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Auf mindestens 30 Jobhäufigkeiten angepasst.

```
  job_title_count_filtered <- table(filtered_data$Job.Title)
  cat(paste(names(job_title_count_filtered), ":", job_title_count_filtered, "\n"))
```

```
Back end Developer : 242
 Content Marketing Manager : 73
 Data Analyst : 391
 Data Scientist : 515
 Digital Marketing Manager : 52
 Director of Data Science : 57
 Director of HR : 69
 Director of Marketing : 88
 Financial Analyst : 53
 Financial Manager : 139
 Front end Developer : 239
 Front End Developer : 31
 Full Stack Engineer : 304
```

```
HR Generalist : 104
Human Resources Coordinator : 50
Human Resources Manager : 152
Marketing Analyst : 144
Marketing Coordinator : 167
Marketing Director : 65
Marketing Manager : 315
Operations Manager : 122
Product Designer : 80
Product Manager : 323
Product Marketing Manager : 70
Project Engineer : 317
Project Manager : 34
Receptionist : 57
Research Director : 75
Research Scientist : 119
Sales Associate : 212
Sales Director : 62
Sales Executive : 38
Sales Manager : 58
Sales Representative : 81
Software Developer : 186
Software Engineer : 809
Software Engineer Manager : 376
Web Developer : 129
```

5.2 5.2 Job Typen

5.3 5.2.1 Anzahl der technischen / administrativen Jobs

Im Folgenden werden Jobs auf Basis Ihrer Jobtitel in technische und adminisztrative Kategorien unterteilt. Daraufhin werden die Datensätze “technische_jobs” und “admin_Jobs” erstellt.

```r
# Filtern nach technischen Jobs
technische_jobs <- filtered_data[grep("data|engineer|developer|analyst|scientist", tolower

# Filtern nach wirtschaftlichen/administrativen Jobs
admin_jobs <- filtered_data[grep("associate|director|manager|sales|coordinator|generalist|

# Beispiel für die Ausgabe der ersten paar Zeilen der gefilterten Daten
```

```r
head(technische_jobs)
```

```
# A tibble: 6 x 11
  Job.Title    job_count   Age Gender Education.Level Years.Of.Experience Salary
  <chr>            <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
1 Back end De~       242    33 Female               2                    5 110000
2 Back end De~       242    32 Male                 1                    4  95000
3 Back end De~       242    26 Female               2                    3  90000
4 Back end De~       242    26 Female               2                    2  70000
5 Back end De~       242    24 Female               1                    1  60000
6 Back end De~       242    26 Female               2                    3  90000
# i 4 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>
```

```r
head(admin_jobs)
```

```
# A tibble: 6 x 11
  Job.Title    job_count   Age Gender Education.Level Years.Of.Experience Salary
  <chr>            <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
1 Content Mar~        73    30 Female               1                    3  55000
2 Content Mar~        73    27 Female               2                    4  80000
3 Content Mar~        73    27 Female               2                    4  80000
4 Content Mar~        73    27 Female               2                    4  80000
5 Content Mar~        73    27 Female               2                    4  80000
6 Content Mar~        73    27 Female               2                    4  80000
# i 4 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>
```

```r
# Anzahl der technischen Jobs
anzahl_technische_jobs <- nrow(technische_jobs)
cat("Anzahl der technischen Jobs:", anzahl_technische_jobs, "\n")
```

```
Anzahl der technischen Jobs: 3912
```

```r
# Anzahl der administrativen Jobs
anzahl_admin_jobs <- nrow(admin_jobs)
cat("Anzahl der administrativen Jobs:", anzahl_admin_jobs, "\n")
```

```
Anzahl der administrativen Jobs: 2919
```

Zu erkennen ist hier, dass es in diesem Datensatz signifikant mehr technische Jobs als administrative Jobs gibt.

```
# Anzahl der Zeilen (Werte) in filtered_data
anzahl_werte_filtered_data <- nrow(filtered_data)

# Anzeigen der Anzahl der Werte
cat("Anzahl der Werte in filtered_data:", anzahl_werte_filtered_data, "\n")
```

```
Anzahl der Werte in filtered_data: 6398
```

Insgesamt gibt es in dem gefilterten Datensatz 6398 Einträge, was bedeutet, dass es sich bei ungefähr 60% um technische Jobs handelt und 40% administrative Jobs sind.

Nun wird nochmal der gefilterte Datensatz mit der Summe von "anzahl_technische_jobs" und "anzahl_administraive Jobs" verglichen.

```
anzahl_jobs <- anzahl_technische_jobs + anzahl_admin_jobs
cat(anzahl_jobs)
```

```
6831
```

Zu erkennen ist hier, dass es zwei unterschiedliche Werte für die beiden Datensätze gibt.

Wir vermuten, dass einige Jobs doppelt gezählt werden. Dies würde erklären, dass deutlich mehr Einträge in "anzahl_jobs" sind. Unser Lösungsvorschlag wäre hier, dass wir den Jobs IDs geben.

5.3.0.1 5.2.1.1 Lösungsansatz 1

Der erste Lösungsansatz sieht wie folgt aus:

Die Jobs werden durch Filter anhand bestimmter Namen (wie "data", "engineer" usw.) ausgewählt und alle Duplikate entfernt. Das Ergebnis ist nun ein neuer Datensatz namens "filtered_data_neu", der alle Zeilen außer die mit technischen und administrativen Jobs enthält.

```
filtered_data$ID <- 1:nrow(filtered_data)

technische_jobs2 <- unique(filtered_data[grep("data|engineer|developer|analyst|scientist",

admin_jobs2 <- unique(filtered_data[grep("associate|director|manager|sales|coordinator|gen
```

```r
ids_technische_jobs <- filtered_data$ID[filtered_data$Job.Title %in% technische_jobs2$Job.
ids_admin_jobs <- filtered_data$ID[filtered_data$Job.Title %in% admin_jobs2$Job.Title]

# Entferne die entsprechenden Zeilen aus filtered_data
filtered_data_neu <- filtered_data[!(filtered_data$ID %in% c(ids_technische_jobs, ids_admi

# Beispiel für die Ausgabe der ersten paar Zeilen der gefilterten Daten
head(filtered_data_neu)
```

```
# A tibble: 0 x 12
# i 12 variables: Job.Title <chr>, job_count <int>, Age <dbl>, Gender <chr>,
#   Education.Level <dbl>, Years.Of.Experience <dbl>, Salary <dbl>,
#   Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>, ID <int>
```

```r
anzahl_technische_jobs2 <- nrow(technische_jobs2)
cat("Anzahl der technischen Jobs2:", anzahl_technische_jobs2, "\n")
```

```
Anzahl der technischen Jobs2: 3912
```

```r
anzahl_admin_jobs2 <- nrow(admin_jobs2)
cat("Anzahl der administrativen Jobs2:", anzahl_admin_jobs2, "\n")
```

```
Anzahl der administrativen Jobs2: 2919
```

Es scheint als würde dieser Lösungsansatz nicht funktionieren, da der neue Datensatz keine Einträge enthält.

5.3.0.2 5.2.1.2 Lösungsansatz 2

Der zweite Lösungsansatz für das Problem der Klassifizierung der verschiedenen Jobtypen besteht darin, dass nicht in zwei Tabellen unterteilt wird, sondern dass jeder Zeile ein Wert des entsprechenden Jobtyps zugeordnet wird.

```r
filtered_data$job_type <- ifelse(
    grepl("data|engineer|developer|analyst|scientist", tolower(filtered_data$Job.Title)),
    0, # 0 für technische Jobs
    ifelse(
```

```
        grepl("associate|director|manager|sales|coordinator|generalist", tolower(filtered_
        1, # 1 für administrative Jobs
        NA  # NA für alle anderen
    )
)

total_rows <- nrow(filtered_data)
count_job_types <- table(filtered_data$job_type, useNA = "ifany")

print(paste("Gesamtanzahl der Zeilen im Datensatz:", total_rows))
```

```
[1] "Gesamtanzahl der Zeilen im Datensatz: 6398"
```

```
print("Anzahl der Zeilen für jede job_type-Ausprägung:")
```

```
[1] "Anzahl der Zeilen für jede job_type-Ausprägung:"
```

```
print(count_job_types)
```

```
   0    1 <NA>
3912 2349  137
```

Es wird eine neue Spalte namens “job_type” erstellt und mit Werten gefüllt. Dabei wird der Wert 0 für Zeilen mit technischen Jobs, 1 für administrative Jobs und NA für alle anderen Jobtypen vergeben.

Nun werden die eben gefundenen NAs in einen neuen Datensatz geschrieben.

```
na_job_type_rows <- subset(filtered_data, is.na(job_type))

na_job_type_rows
```

```
# A tibble: 137 x 13
   Job.Title   job_count   Age Gender Education.Level Years.Of.Experience Salary
   <chr>           <int> <dbl> <chr>            <dbl>               <dbl>  <dbl>
 1 Product De~        80    33 Male                 2                   6  90000
 2 Product De~        80    43 Female               3                  18 140000
 3 Product De~        80    45 Female               3                  15 150000
```

```
 4 Product De~        80    45 Female               3                  15 150000
 5 Product De~        80    44 Female               3                  15 150000
 6 Product De~        80    44 Female               3                  15 150000
 7 Product De~        80    27 Male                 1                   3  60000
 8 Product De~        80    27 Male                 1                   3  60000
 9 Product De~        80    27 Male                 1                   3  60000
10 Product De~        80    27 Male                 1                   3  60000
# i 127 more rows
# i 6 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>,
#   ID <int>, job_type <dbl>
```

Das Ergebnis dieser Abfrage ist eine Tabelle, welche nur aus Product Designer % Receptionist besteht. Diese werden nun den adminsitrativen Jobs hinzugefügt.

```r
filtered_data$job_type[filtered_data$Job.Title %in% c("Product Designer", "Receptionist")]

subset(filtered_data, Job.Title %in% c("Product Designer", "Receptionist"))
```

```
# A tibble: 137 x 13
   Job.Title   job_count   Age Gender Education.Level Years.Of.Experience Salary
   <chr>           <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
 1 Product De~        80    33 Male                 2                    6  90000
 2 Product De~        80    43 Female               3                   18 140000
 3 Product De~        80    45 Female               3                   15 150000
 4 Product De~        80    45 Female               3                   15 150000
 5 Product De~        80    44 Female               3                   15 150000
 6 Product De~        80    44 Female               3                   15 150000
 7 Product De~        80    27 Male                 1                    3  60000
 8 Product De~        80    27 Male                 1                    3  60000
 9 Product De~        80    27 Male                 1                    3  60000
10 Product De~        80    27 Male                 1                    3  60000
# i 127 more rows
# i 6 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>,
#   ID <int>, job_type <dbl>
```

```r
total_rows <- nrow(filtered_data)
count_job_types <- table(filtered_data$job_type, useNA = "ifany")

print(paste("Gesamtanzahl der Zeilen im Datensatz:", total_rows))
```

```
[1] "Gesamtanzahl der Zeilen im Datensatz: 6398"
```

```r
print("Anzahl der Zeilen für jede job_type-Ausprägung:")
```

```
[1] "Anzahl der Zeilen für jede job_type-Ausprägung:"
```

```r
print(count_job_types)
```

```
   0    1
3912 2486
```

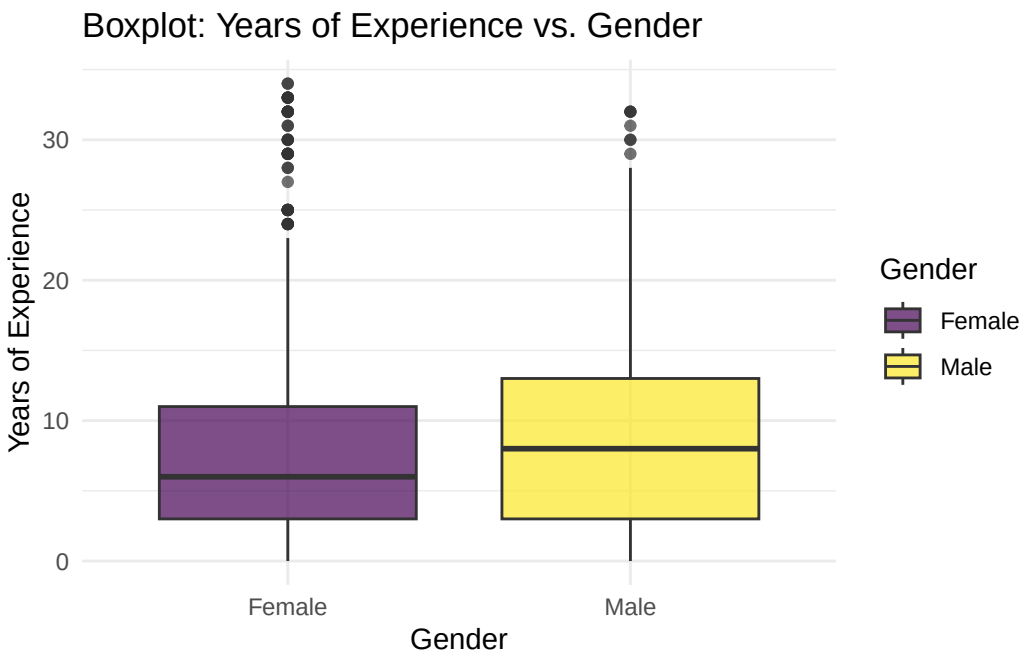
Nun ist das Klassifizierungsproblem gelöst. Aufgrund dessen können jetzt auch Diagramme über Job_types ausgewertet werden.

5.4 5.3 Zugewanderte (Expats) & Einheimische

Im Folgenden werden einige Boxplots erstellt.

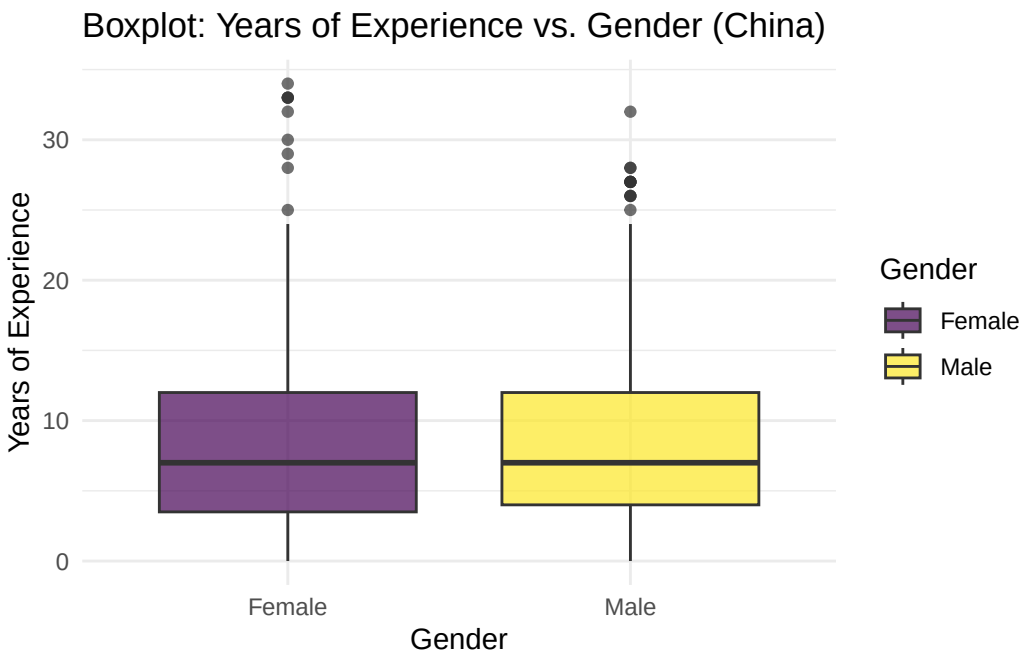
5.4.1 5.3.1 Years of Experience vs. Gender

```r
ggplot(filtered_data, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```



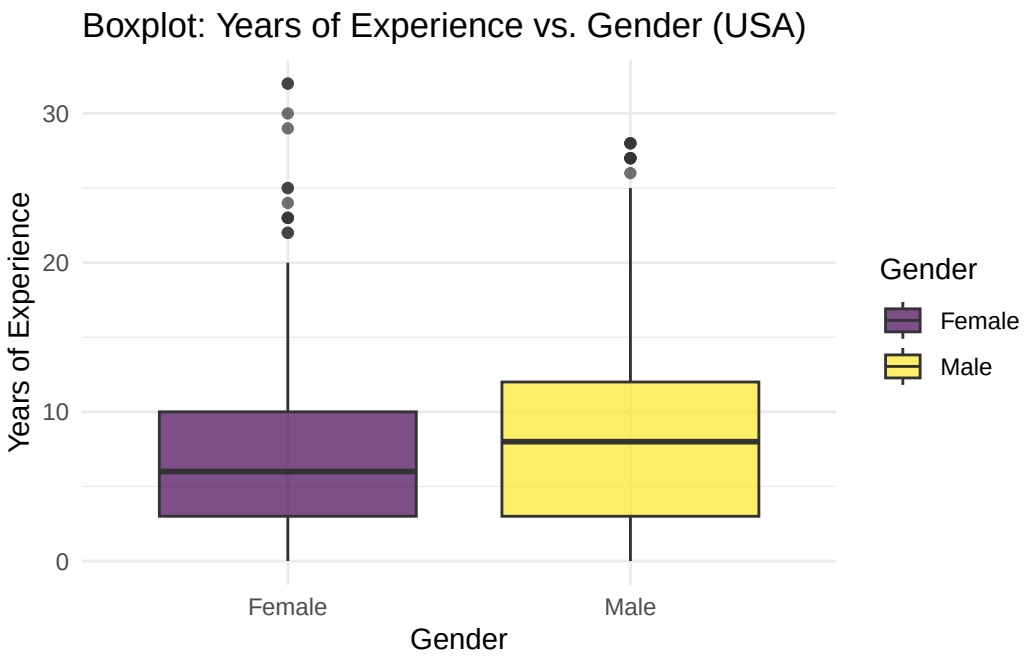
5.4.2 5.3.2 Years of Experience vs. Gender (China)

```r
library(ggplot2)

data_china <- subset(filtered_data, Country == "China")

ggplot(data_china, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender (China)",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```



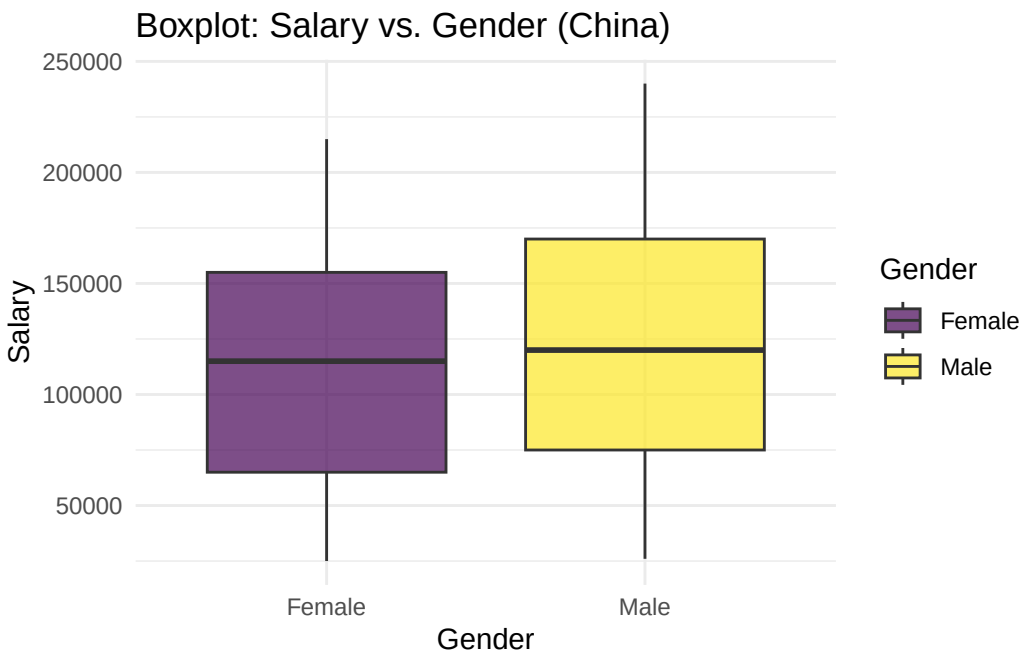
5.4.3 5.3.3 Years of Experience vs. Gender(USA)

```r
library(ggplot2)

data_usa <- subset(filtered_data, Country == "USA")

ggplot(data_usa, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender (USA)",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```



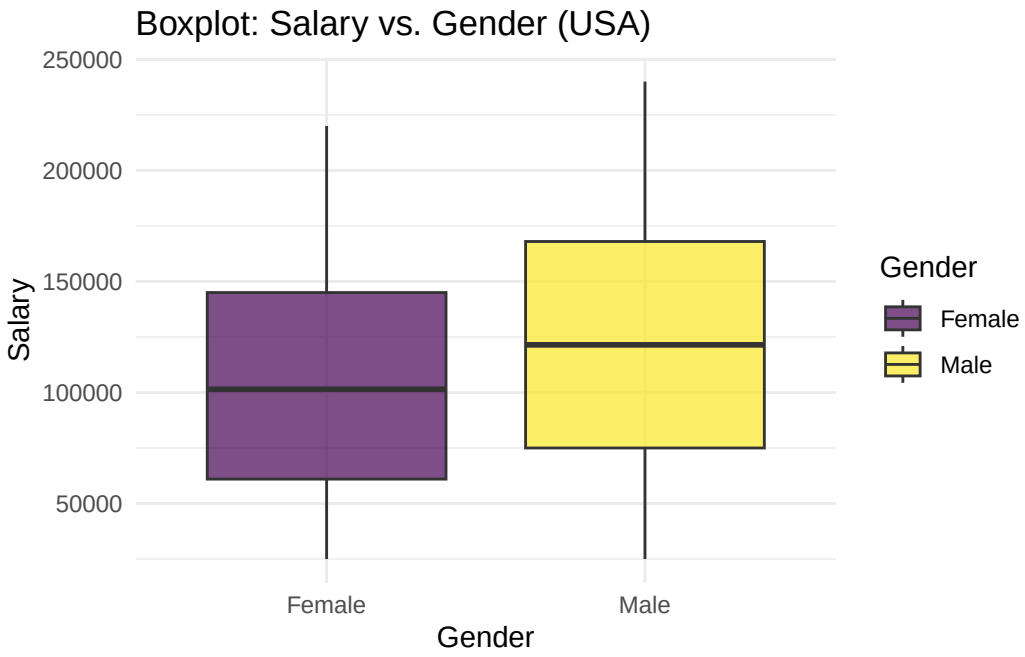
5.4.4 5.3.4 Salary vs. Gender (China)

```
library(ggplot2)

data_china <- subset(filtered_data, Country == "China")

ggplot(data_china, aes(x = factor(Gender), y = Salary, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Salary vs. Gender (China)",
       x = "Gender",
       y = "Salary",
       fill = "Gender") +
  theme_minimal()
```



5.4.5 5.3.5 Salary vs. Gender (USA)

```
library(ggplot2)

data_usa <- subset(filtered_data, Country == "USA")

ggplot(data_usa, aes(x = factor(Gender), y = Salary, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Salary vs. Gender (USA)",
       x = "Gender",
       y = "Salary",
       fill = "Gender") +
  theme_minimal()
```



Nach intensiver Betrachtung der Grafiken bezüglich der Salary, kommt folgende Frage auf:

Ist die Gender-Pay-Gap in China doch größer, da der größte Faktor für die Salary die Years of Experience sind?

Hierzu werden die Korrelationen errechnet.

5.4.6 5.3.6 Korrelationen zwischen den Spalten

```
correlation_salary_experience <- cor(filtered_data$Salary, filtered_data$Years.Of.Experien

cat("Die Korrelation zwischen Salary und Years.Of.Experience ist:", correlation_salary_exp
```

```
Die Korrelation zwischen Salary und Years.Of.Experience ist: 0.8103542
```

```
correlation_salary_education <- cor(filtered_data$Salary, filtered_data$Education.Level, u

cat("Die Korrelation zwischen Salary und Education.Level ist:", correlation_salary_educati
```

```
Die Korrelation zwischen Salary und Education.Level ist: 0.6374551
```

```r
correlation_salary_age <- cor(filtered_data$Salary, filtered_data$Age, use = "complete.obs

cat("Die Korrelation zwischen Salary und Age ist:", correlation_salary_age, "\n")
```

```
Die Korrelation zwischen Salary und Age ist: 0.7291603
```

```r
correlation_experience_age <- cor(filtered_data$Years.Of.Experience, filtered_data$Age, us

cat("Die Korrelation zwischen Years.Of.Experience und Age ist:", correlation_experience_ag
```

```
Die Korrelation zwischen Years.Of.Experience und Age ist: 0.9363709
```

```r
correlation_seniority_experience <- cor(filtered_data$Senior, filtered_data$Years.Of.Exper

cat("Die Korrelation zwischen Seniority und Years.Of.Experience ist:", correlation_seniori
```

```
Die Korrelation zwischen Seniority und Years.Of.Experience ist: 0.3192657
```

Das Ergebnis der Korrelationen:

- Von Salary und Years.of.experience ist es 0.81
- Von Salary und Age ist es 0.73.
- Von Age und Years.Of.Experience ist es 0.93.

Nun stellt sich folgende Frage:

Wie kommt es zu so einem großen Unterschied zwischen den Werten im Vergleich zu Salary, obwohl sie doch eine starke Korrelation zueinander haben?

Mögliche Antworten auf diese Frage wären:

Verteilung der Daten: Es ist möglich, dass die Verteilung der Daten in den Variablen “Age” und “Years.Of.Experience” anders ist als in der Variable “Salary”. Wenn die Daten in “Age” und “Years.Of.Experience” breiter gestreut sind, kann dies zu einer geringeren Korrelation führen, selbst wenn eine starke lineare Beziehung besteht.

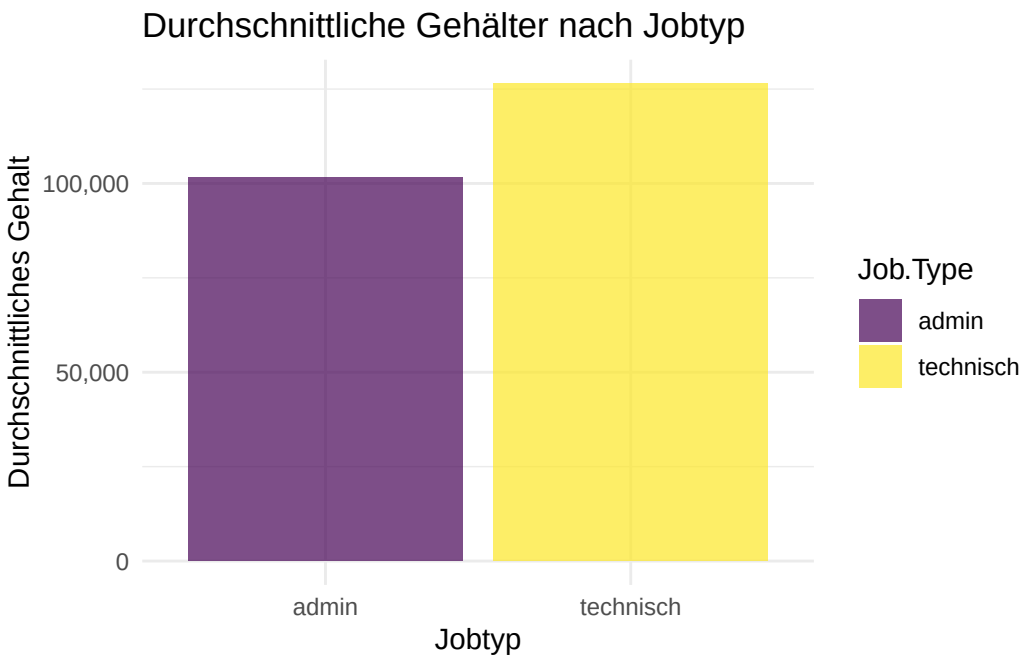
Nicht-lineare Beziehung: Die Korrelation misst nur lineare Beziehungen. Wenn die Beziehung zwischen “Age” und “Years.Of.Experience” nicht linear ist, könnte dies zu einem niedrigeren Korrelationswert führen.

Ausreißer: Das Vorhandensein von Ausreißern kann die Korrelation beeinflussen. Wenn es Ausreißer in einer der Variablen gibt, kann dies den Korrelationswert beeinträchtigen.

Stichprobengröße: Bei kleineren Stichproben können Korrelationswerte instabiler sein.

Nun werden die Daten auf technische und administrative Jobs gefiltert, anschließend werden die Gehälter der beiden Gruppen ausgegeben und in einem Diagramm dargestellt und verglichen.

```
technische_jobs <- subset(filtered_data, job_type == 0)
admin_jobs <- subset(filtered_data, job_type == 1)

average_salaries_technical <- mean(technische_jobs$Salary, na.rm = TRUE)

average_salaries_admin <- mean(admin_jobs$Salary, na.rm = TRUE)

all_average_salaries <- data.frame(Job.Type = c("technisch", "admin"),
                                   Average.Salary = c(average_salaries_technical, average_

ggplot(all_average_salaries, aes(x = Job.Type, y = Average.Salary, fill = Job.Type)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Durchschnittliche Gehälter nach Jobtyp",
       x = "Jobtyp",
       y = "Durchschnittliches Gehalt") +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```



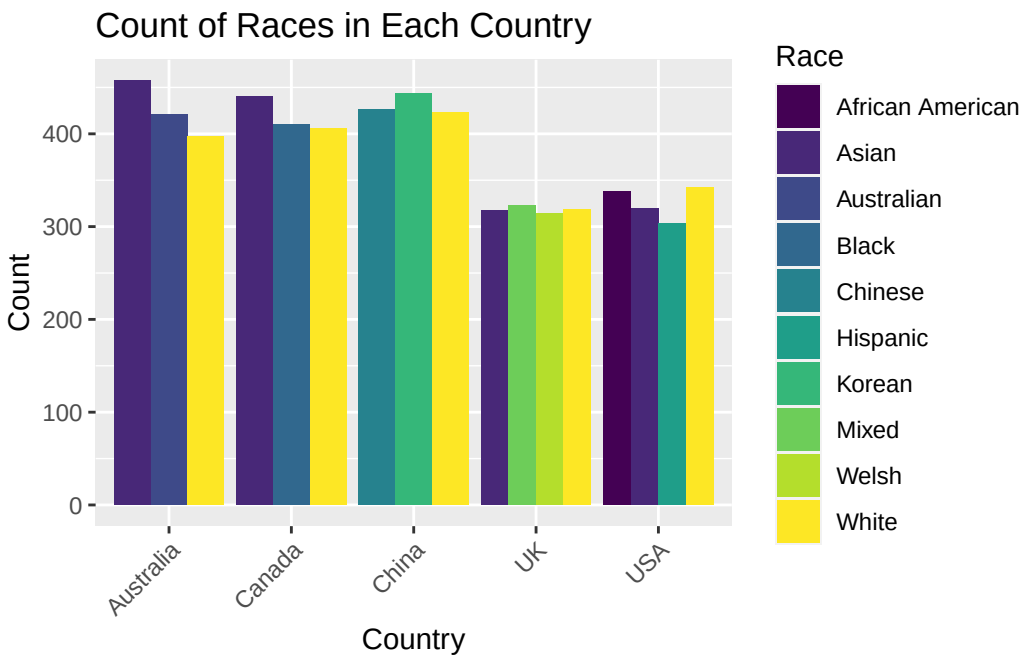
Obwohl in den administrativen Jobs auch Direktoren und Manager vertreten sind, nehmen wir an, dass ein Manager einen Job Titel, wie “Projekt Manager” und nicht “Manager für Projekte” hat und dieser Titel in unserem Datensatz “Director” ist (Thema Führungsposition).

Unsere Untersuchungen haben ergeben, dass wir die Daten für unsere Explorative Datenanalyse aber auch für die folgenden Regressionen neu aufbereiten müssen.

Dazu werden folgende Fragen untersucht:

Unterscheidet sich ein Native und Expat im jeweiligen Land? Welche Annahmen sind dafür nötig? Hier die Annahme das “White” generell nicht ausgewandert ist, da wir hier Länder mit ähnlicher Kultur und Salary haben.

```
ggplot(filtered_data, aes(x = Country, fill = Race)) +
  geom_bar(position = "dodge") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Count of Races in Each Country",
       x = "Country",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Dafür wird eine neue Spalte eingefügt, die mit numerischen Werten arbeitet. 0 steht für Einheimische und 1 für einen Expat. Den Wert null erhalten alle zeilen bei denen wir folgende Übereinstimmung feststellen: African American (USA) White (Canada, USA, UK, Australia) Chinese (China) Australian(Australia) Welsh (UK) jede andere Race ist dementsprechend Expat und erhält eine 1 in der Spalte Expat.

```
filtered_data$Expat <- 0

expat_conditions <- list(
  filtered_data$Race == "African American" & filtered_data$Country == "USA",
  filtered_data$Race %in% c("White", "Chinese", "Australian", "Welsh") &
    filtered_data$Country %in% c("Canada", "USA", "UK", "Australia"),
  TRUE
)

filtered_data$Expat <- ifelse(expat_conditions[[1]] | expat_conditions[[2]], 0,
                              ifelse(expat_conditions[[3]], 1, NA))

head(filtered_data)
```

```
# A tibble: 6 x 14
  Job.Title    job_count   Age Gender Education.Level Years.Of.Experience Salary
```

```
  <chr>            <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
1 Back end De~       242    33 Female               2                    5 110000
2 Back end De~       242    32 Male                 1                    4  95000
3 Back end De~       242    26 Female               2                    3  90000
4 Back end De~       242    26 Female               2                    2  70000
5 Back end De~       242    24 Female               1                    1  60000
6 Back end De~       242    26 Female               2                    3  90000
# i 7 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>,
#   ID <int>, job_type <dbl>, Expat <dbl>
```

Ausgabe hier sind nun die ersten Zeilen der aktualisierten Version von "filtered_data", indem die Spalte "Expat" basierend auf den oben genannten Kriterien befüllt wurde.

6 6. Thesen

Der folgende Absatz wird in unterschiedliche Teilabschnitte geteilt, um eine gute Leserlichkeit zu erreichen.

Basierend auf den durchgeführten Tests und der Vorarbeit ergeben sich folgende Thesen, die im Anschluss untersucht werden sollen:

6.1 6.1 Genderpaygap

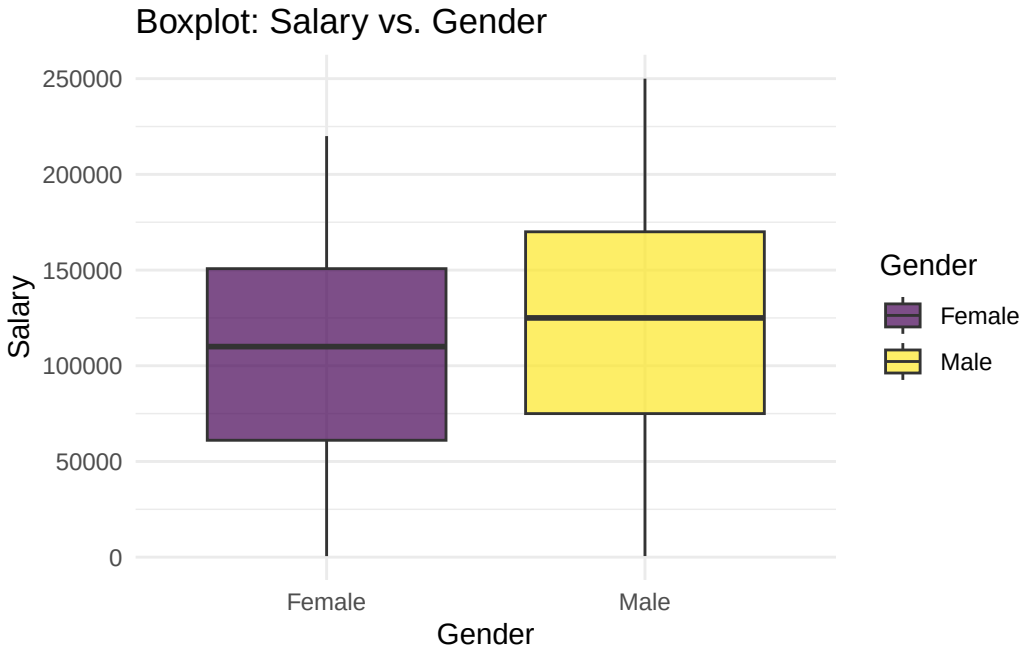
1. Männer verdienen mehr als Frauen.
2. Die Differenz der Salary zwischen den Geschlechtern ist in China höher als in den westlichen Ländern.
3. Männer haben im Durchschnitt mehr Years of Experience als Frauen -> Lässt sich die Genderpaygap auf die Yrs of Exp übertragen? Und gilt dies auch für China?

6.1.1 6.1.1 Männer verdienen mehr als Frauen

Zunächst stellt sich die Frage, ob Männer mehr verdienen als Frauen. Hierzu wird ein Boxplot verwendet.

```
ggplot(filtered_data, aes(x = factor(Gender), y = Salary, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Salary vs. Gender",
       x = "Gender",
       y = "Salary",
```

```
      fill = "Gender") +
  theme_minimal()
```



Anhand des Boxplots ist zu erkennen, dass der Median der Männer deutlich höher, als der Median der Frauen ist. Aufgrund dieser Tatsache, lässt sich sagen, dass diese Aussage korrekt ist.

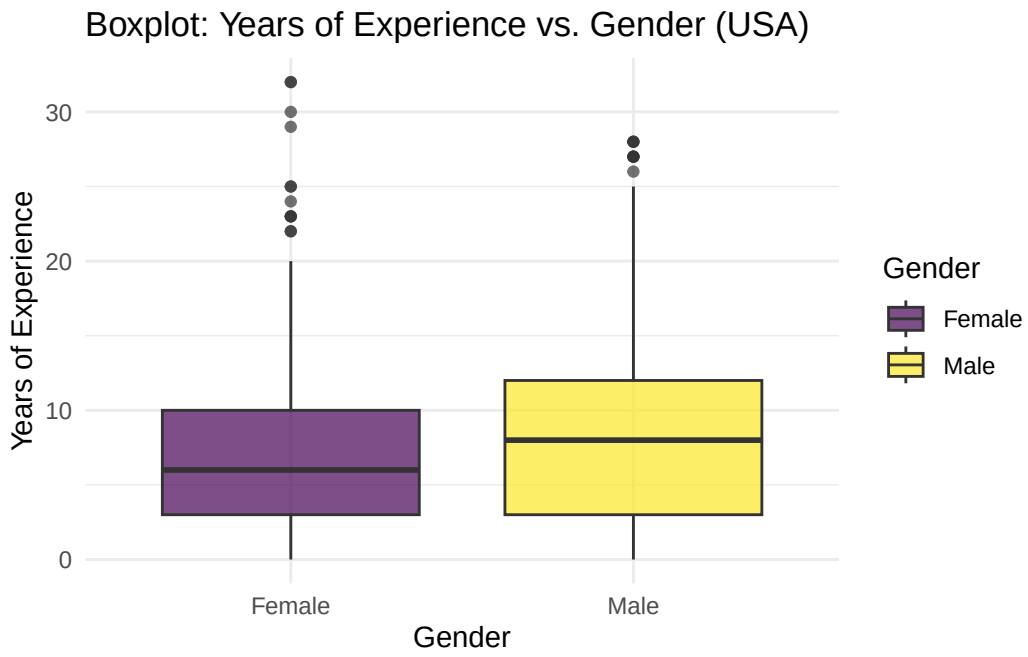
6.1.2 6.1.2.: Die Differenz der Salary zwischen den Geschlechtern ist in China höher als in den westlichen Ländern. Hier alle westlichen Länder hinzufügen

Um diese These zu beantworten werden zunächst die Erfahrungsjahre zwischen China und den USA verglichen. Anschließend werden die Gehälter verglichen. Nebenbei haben die Boxplots immer eine Differenzierung zwischen Männern und Frauen um den Unterschied darzustellen.

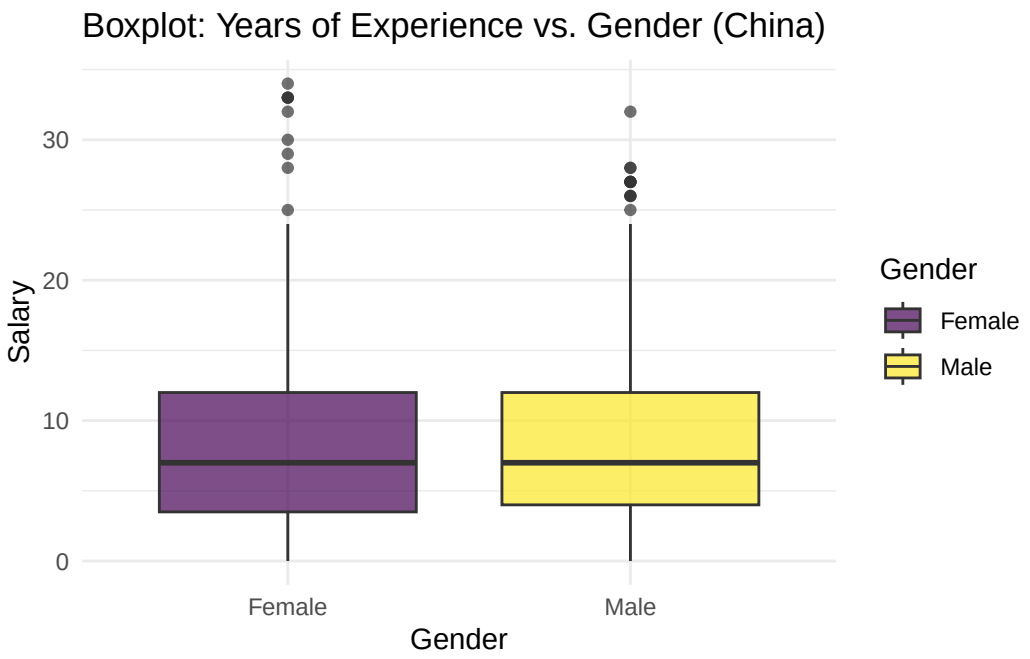
Zu Beantwortung dieser These werden Boxplots verwendet.

```
data_usa <- subset(filtered_data, Country == "USA")

ggplot(data_usa, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
```

```
  labs(title = "Boxplot: Years of Experience vs. Gender (USA)",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```

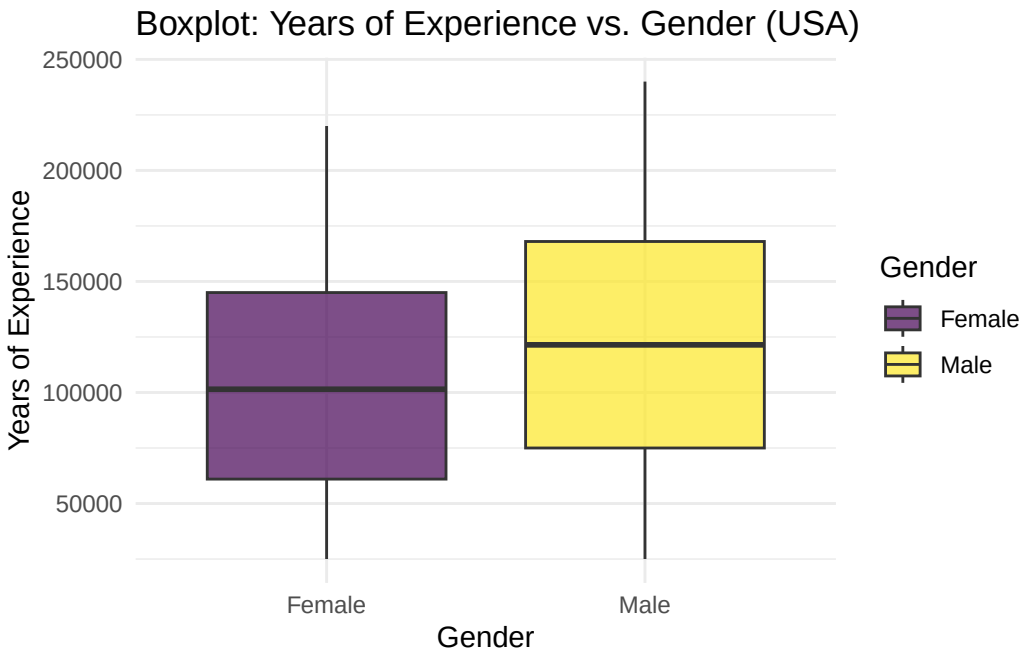



```
data_china <- subset(filtered_data, Country == "China")

ggplot(data_china, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender (China)",
       x = "Gender",
       y = "Salary",
       fill = "Gender") +
  theme_minimal()
```

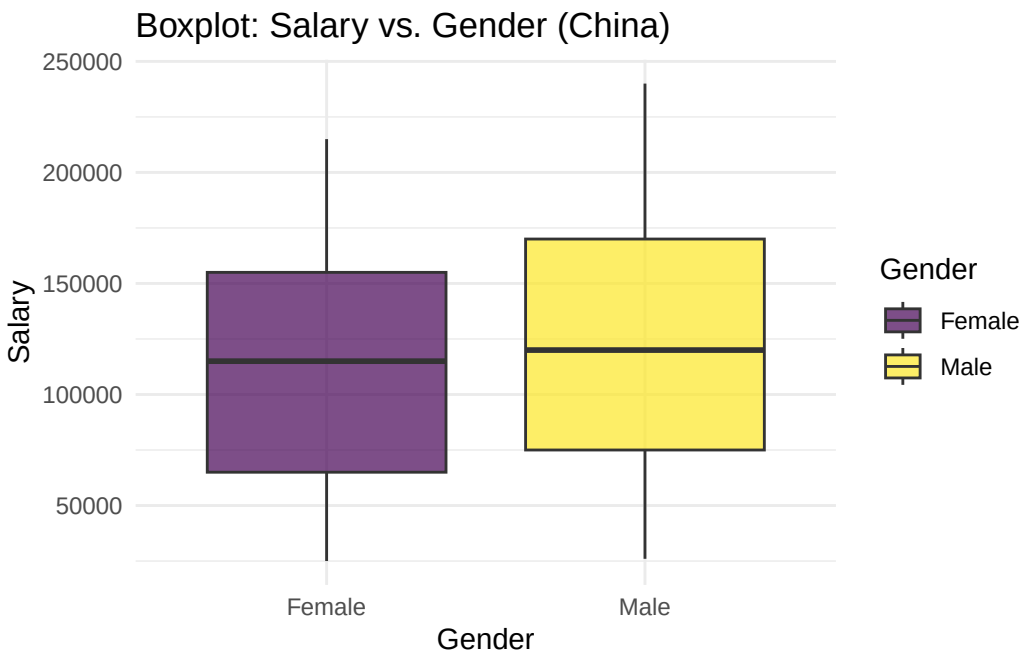


Anhand der ersten beiden Boxplots ist zu erkennen, dass Männer in der USA deutlich mehr Berufserfahrung haben als Frauen. Wohin gegen der Unterschied in China kaum zu erkennen ist.

```
data_usa <- subset(filtered_data, Country == "USA")

ggplot(data_usa, aes(x = factor(Gender), y = Salary, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender (USA)",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```




```r
data_china <- subset(filtered_data, Country == "China")

ggplot(data_china, aes(x = factor(Gender), y = Salary, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Salary vs. Gender (China)",
       x = "Gender",
       y = "Salary",
       fill = "Gender") +
  theme_minimal()
```



Durch die letzen beiden Boxplots ist zu erkennen, dass der Unterschied im durchschnittlichen Gehalt in China und der USA mit dem bloßen Auge nicht zu erkennen ist.
Unter Anbetracht der oberen beiden Boxplots ist jedoch, wie bereits erwähnt, ein deutlicher Unterschied zwischen den Geschlechtern im Bezug auf die Berufsaerfahrung zu erkennen.

```
correlation_salary_experience <- cor(filtered_data$Salary, filtered_data$Years.Of.Experien

cat("Die Korrelation zwischen Salary und Years.Of.Experience ist:", correlation_salary_exp
```

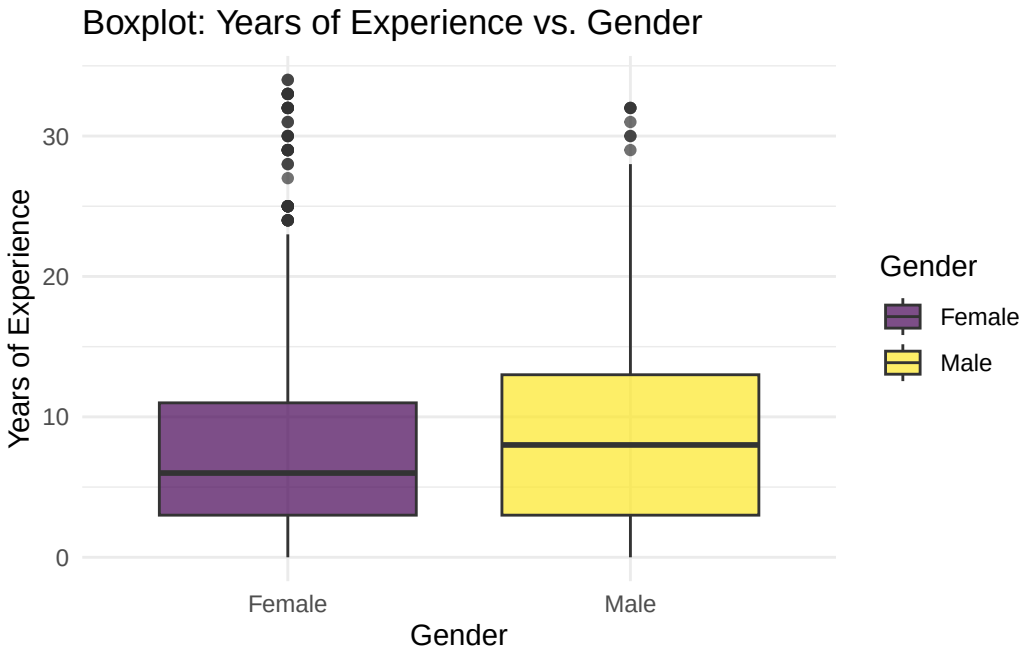
```
Die Korrelation zwischen Salary und Years.Of.Experience ist: 0.8103542
```

Aus den obigen Tests ist außerdem hervorgegangen, dass die Berufserfahrung eine hohe Korrelation zu dem Gehalt hat.

Deswegen lässt sich trotzdem sagen, dass diese These korrekt ist.

6.1.3 6.1.3.: Männer haben im Durchschnitt mehr Berufserfahrung als Frauen

```
ggplot(filtered_data, aes(x = factor(Gender), y = Years.Of.Experience, fill = Gender)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplot: Years of Experience vs. Gender",
       x = "Gender",
       y = "Years of Experience",
       fill = "Gender") +
  theme_minimal()
```



Die These ist korrekt, wie anhand des obigen Boxplots zu erkennen ist. Der Median liegt bei den Männern höher, als bei den Frauen.

6.2 6.2 Zugewanderte Menschen verdienen mehr als einheimische Menschen

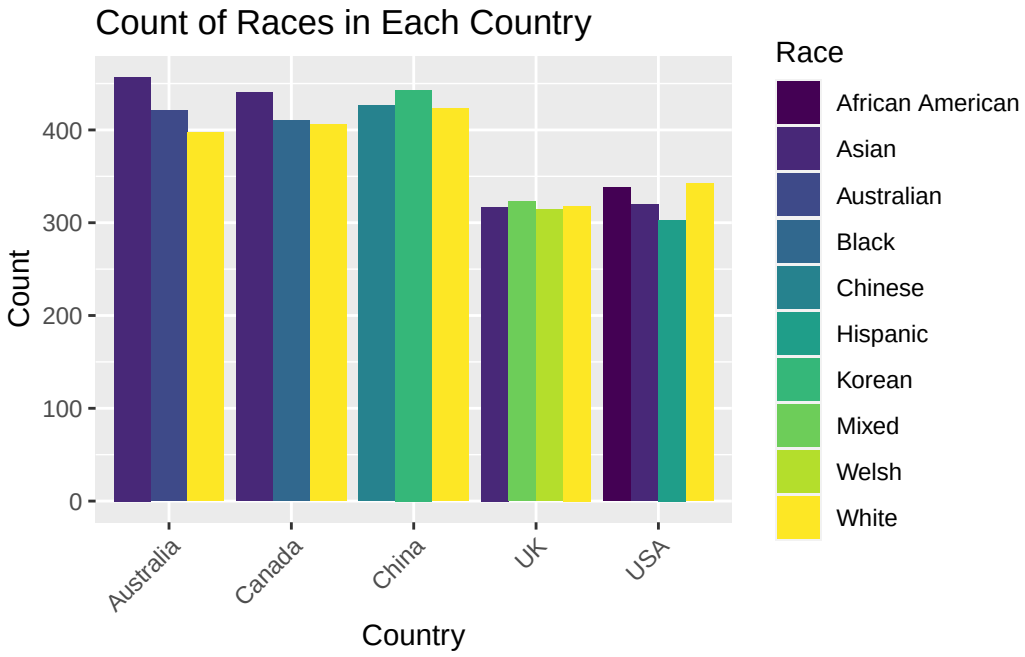
Unsere Untersuchungen haben ergeben, dass wir die Daten für unsere Explorative Datenanalyse, sowie auch die Regression neu aufbereiten müssen.

Dazu untersuchen wir:

Wie unterscheide ich einen Native und Expat im jeweiligen Land. Welche Annahmen sind dafür nötig? Hier die Annahme, dass "White" generell nicht ausgewandert ist, da der Datensatz Länder mit ähnlicher Kultur und Salary beinhaltet.

6.2.1 6.2.1.: Alle Ethnizitäten je Land

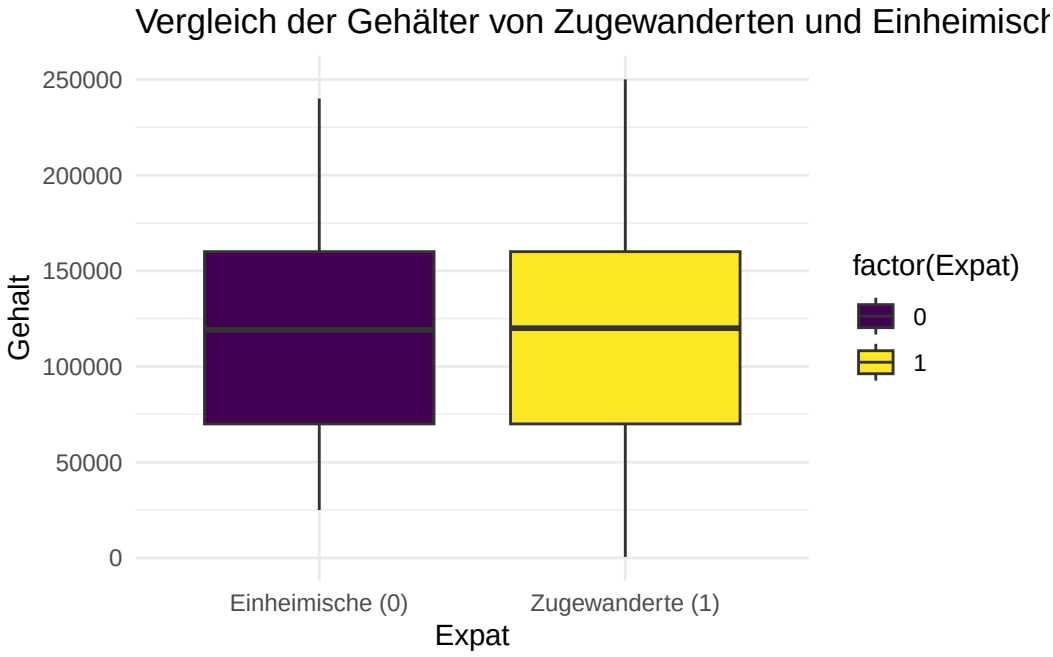
```
ggplot(filtered_data, aes(x = Country, fill = Race)) +
  geom_bar(position = "dodge") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Count of Races in Each Country",
       x = "Country",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Hier ist ist zu erkennen, welche Ethnizitäten in den unterschiedlichen Ländern existieren. So ist zu erkennen, dass es nie mehr als vier Ethnizitäten pro Land gibt.

6.2.2 6.2.2.: Gesamtbetrachtung

```
ggplot(filtered_data, aes(x = as.factor(Expat), y = Salary, fill = factor(Expat))) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Vergleich der Gehälter von Zugewanderten und Einheimischen",
       x = "Expat",
       y = "Gehalt") +
  scale_x_discrete(labels = c("Einheimische (0)", "Zugewanderte (1)")) +
  theme_minimal()
```



Aus diesem Diagramm geht hervor, dass vorerst kein Unterschied zu erkennen ist.

```
mean_salary_expat <- mean(filtered_data$Salary[filtered_data$Expat == 1], na.rm = TRUE)
mean_salary_expat
```

```
[1] 116928.1
```

```
mean_salary_native <- mean(filtered_data$Salary[filtered_data$Expat == 0], na.rm = TRUE)
mean_salary_native
```

```
[1] 116572.5
```

Auch nach Darstellung der Mittelwerte ist kein wirklicher Unterschied zu erkennen.

Deswegen wird die These verworfen, da es offensichtlich keine Unterschiede gibt.

6.3 6.3 Gibt es einen Unterschied im Gehalt zwischen den verschiedenen Bildungsniveaus?

Vorab muss erwähnt werden, dass ohne ein Mindestmaß an Bildung keine weitere Gehaltsentwicklung möglich ist.

Das sind die Bedeutungen der verschiedenen Bildungsniveaus:

0 = High School Abschluss

1 = Bachelor

2 = Master

3 = Doctor

Um diese These zu beantworten sind wir auf verschiedene Lösungsansätze gekommen, die uns helfen könnten:

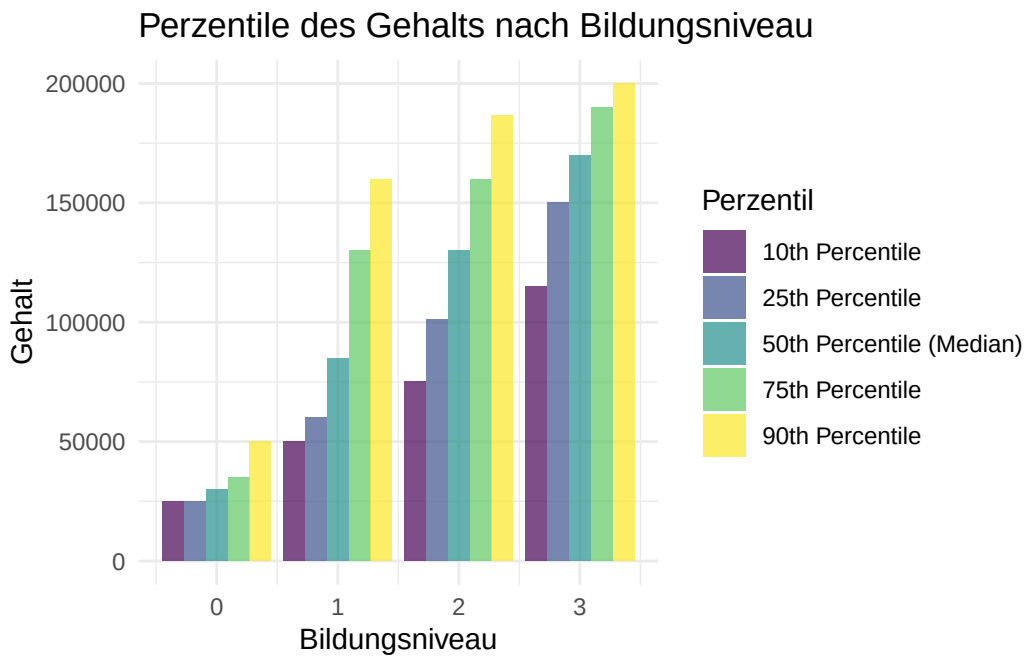
1. **Deskriptive Statistiken:** Es könnten Quantile oder Perzentile des Gehalts für jeden Bildungsniveau berechnet berechnet werden. Dies könnte einen Überblick über die Verteilung der Gehälter bieten und zeigt potenzielle Grenzwerte.
2. **Boxplots pro Bildungsniveau:** Es könnten Boxplots für jedes Bildungsniveau erstellt werden, um die Verteilung der Gehälter visuell zu vergleichen. Dies könnte unterstützend wirken, um Ausreißer und Unterschiede im Bildungsniveau zu identifizieren.
3. **Visualisierungen:** Es besteht die Möglichkeit verschiedene Visualiersierungen, wie Scatterplots oder Liniendiagramme zu erstellen, um Trends oder Muster zwischen Gehalt und Bildungsniveau zu erkennen.

6.3.1 6.3.1.: Desktiptive Statistiken:

Hierzu werden erst die Daten berechnet und diese anschließend für die Grafik “ge-reshaped”.

```
library(ggplot2)
library(dplyr)

# Daten berechnen
```

```r
salary_percentiles <- filtered_data %>%
  group_by(Education.Level) %>%
  summarise(`10th Percentile` = quantile(Salary, probs = 0.1, na.rm = TRUE),
            `25th Percentile` = quantile(Salary, probs = 0.25, na.rm = TRUE),
            `50th Percentile (Median)` = quantile(Salary, probs = 0.5, na.rm = TRUE),
            `75th Percentile` = quantile(Salary, probs = 0.75, na.rm = TRUE),
            `90th Percentile` = quantile(Salary, probs = 0.9, na.rm = TRUE))

salary_percentiles_long <- salary_percentiles %>%
  tidyr::pivot_longer(cols = -Education.Level, names_to = "Percentile", values_to = "Salar

ggplot(salary_percentiles_long, aes(x = Education.Level, y = Salary, fill = Percentile)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Perzentile des Gehalts nach Bildungsniveau",
       x = "Bildungsniveau",
       y = "Gehalt",
       fill = "Perzentil") +
  theme_minimal()
```



In dem gruppierten Balkendiagramm kann erkannt werden, dass die unterschiedlichen Perzentile stets mit dem Bildungsniveau zusammen ansteigen.

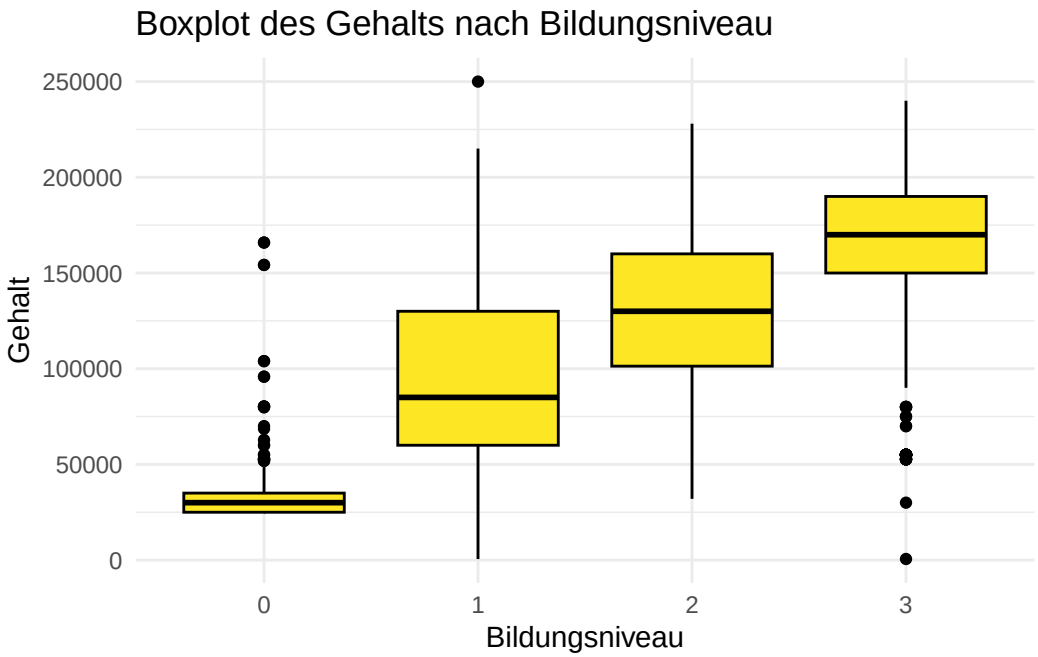
```r
average_salary_education <- aggregate(Salary ~ Education.Level, data = filtered_data, FUN

average_salary_education
```

```
  Education.Level    Salary
1               0  34511.62
2               1  97042.04
3               2 129983.77
4               3 165796.75
```

Die oben gestellte Aussage wird erneut bestätigt, durch die Ausgabe der "Average Salary". Hier ist zu erkennen, dass das durchschnittliche Gehalt höher ist, je höher das Bildungsniveau ist.

6.3.2 6.3.2.: Boxplots pro Bildungsniveau:

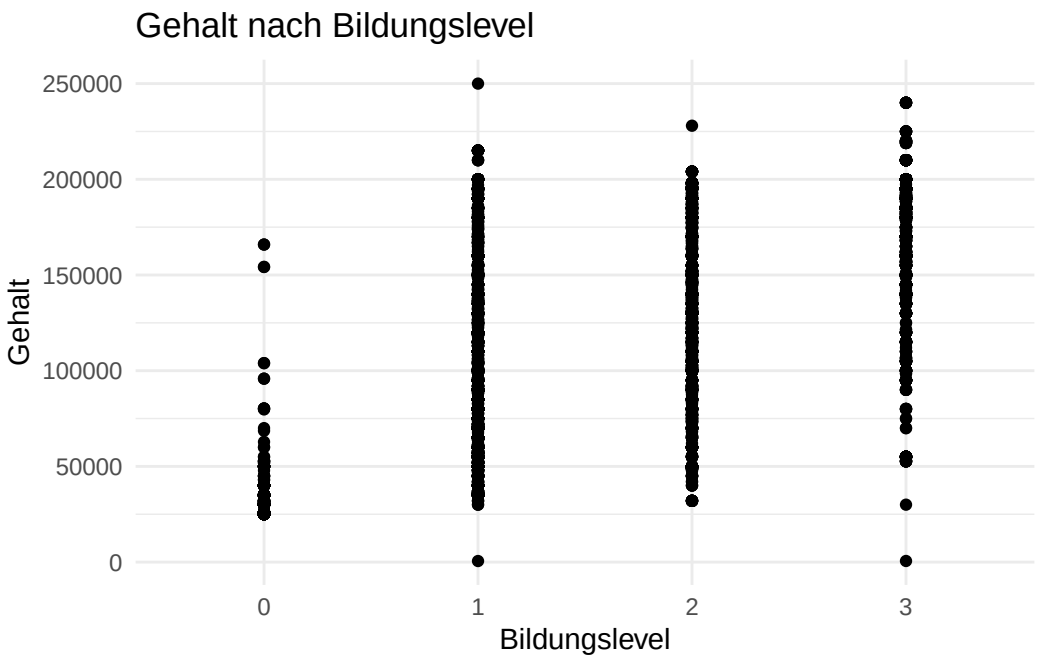
```r
filtered_data <- filtered_data %>%
  mutate(Education.Level = factor(Education.Level, levels = unique(sort(Education.Level))))

ggplot(filtered_data, aes(x = reorder(factor(Education.Level), Salary, FUN = median), y =
  geom_boxplot(color = "black", fill = viridis(2)[2]) +
  labs(title = "Boxplot des Gehalts nach Bildungsniveau",
       x = "Bildungsniveau",
       y = "Gehalt") +
  theme_minimal()
```



Auch in dem Balkendiagramm ist, genau wie oben, zu erkennen, dass der Median stets höher ist, je höher das Bildungsniveau geht.

6.3.3 6.3.3.: Visualisierungen:

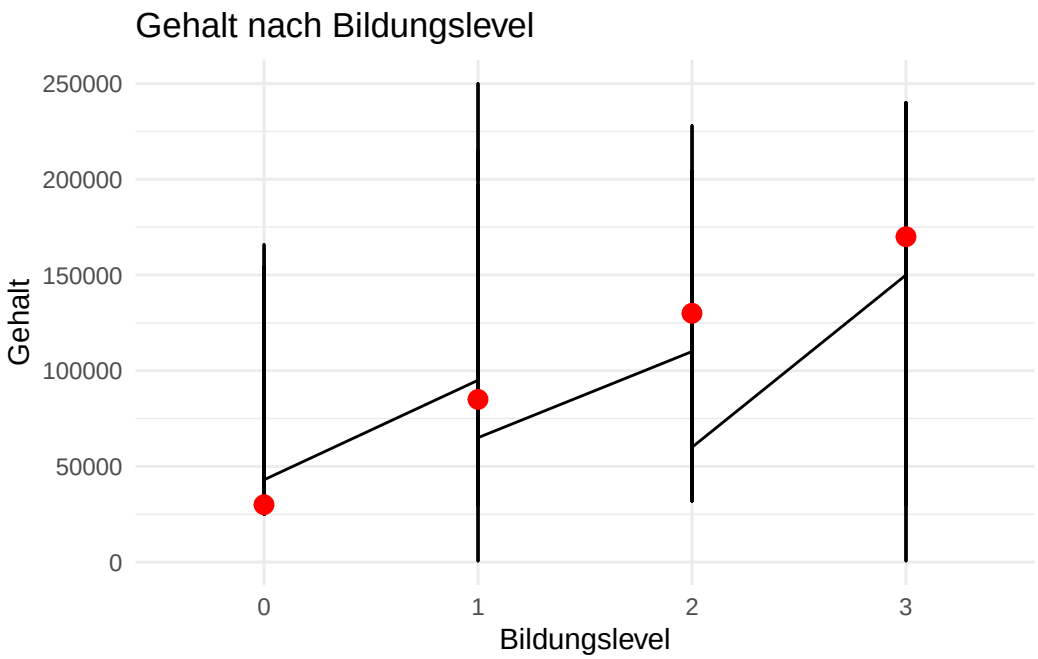
```
filtered_data <- filtered_data %>%
  mutate(Education.Level = factor(Education.Level, levels = unique(sort(Education.Level))))

ggplot(filtered_data, aes(x = reorder(factor(Education.Level), Salary, FUN = median), y =
  geom_point() +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Gehalt nach Bildungslevel",
       x = "Bildungslevel",
       y = "Gehalt") +
  theme_minimal()
```



Der folgende Code erstellt ein Liniendiagramm, dass das Gehalt in Abhängigkeit vom Bildungsniveau darstellt. Zunächst wird das Bildungsniveau neu angeordnet und danach das Liniendiagramm erstellt.

```
filtered_data$Education.Level <- factor(filtered_data$Education.Level, levels = c("0", "1"

ggplot(filtered_data, aes(x = Education.Level, y = Salary, group = 1)) +
  geom_line() +
  stat_summary(fun.y = median, geom = "point", size = 3, color = "red") +
  labs(title = "Gehalt nach Bildungslevel",
       x = "Bildungslevel",
       y = "Gehalt") +
  theme_minimal()
```

```
Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
i Please use the `fun` argument instead.
```



Die roten Punkte zeigen den Medianwert des Gehalts je nach Bildungsniveau. Die schrägen Linien zeigen den Trend steigender Gehälter bei höheren Bildungsstufen.

Ohne einen Hochschulabschluss gibt es eine Gehaltsgrenze. Die Top 90% ohne Hochschulabschluss fangen bei den unteren 10% mit Hochschulabschluss an aus der Sicht des Gehalts.

Die These kann als Korrekt angesehen werden, da alle Lösungsansätze im Allgemeinen, das gleiche Ergebnis liefern.

6.4 6.4 Die technischen Jobs haben ein höheres Gehalt als die administrativen Jobs

6.4.1 6.4.1.: Daten Aufbereiten

Unter dem Punkt “Daten aufbereiten 2” wurden bereits alle Jobs in administrativ und technisch unterteilt.

Zudem werden noch im folgenden alle Werte aus dem Datensatz gefiltert, die den Jobtitel “Director” enthalten, da dieser Job nicht eindeutig einer Gruppe zugeordnet werden kann (z.B “Engineering Director”).

```r
filtered_data2 <- filtered_data

director_jobs <- filtered_data2 %>%
  filter(grepl("Director", Job.Title))

filtered_data2 <- filtered_data2 %>%
  anti_join(director_jobs)
```

```
Joining with `by = join_by(Job.Title, job_count, Age, Gender, Education.Level,
Years.Of.Experience, Salary, Country, Race, Senior, SalaryKat, ID, job_type,
Expat)`
```

```r
nrow(director_jobs)
```

```
[1] 416
```

```r
nrow(filtered_data)
```

```
[1] 6398
```

```r
nrow(filtered_data2)
```

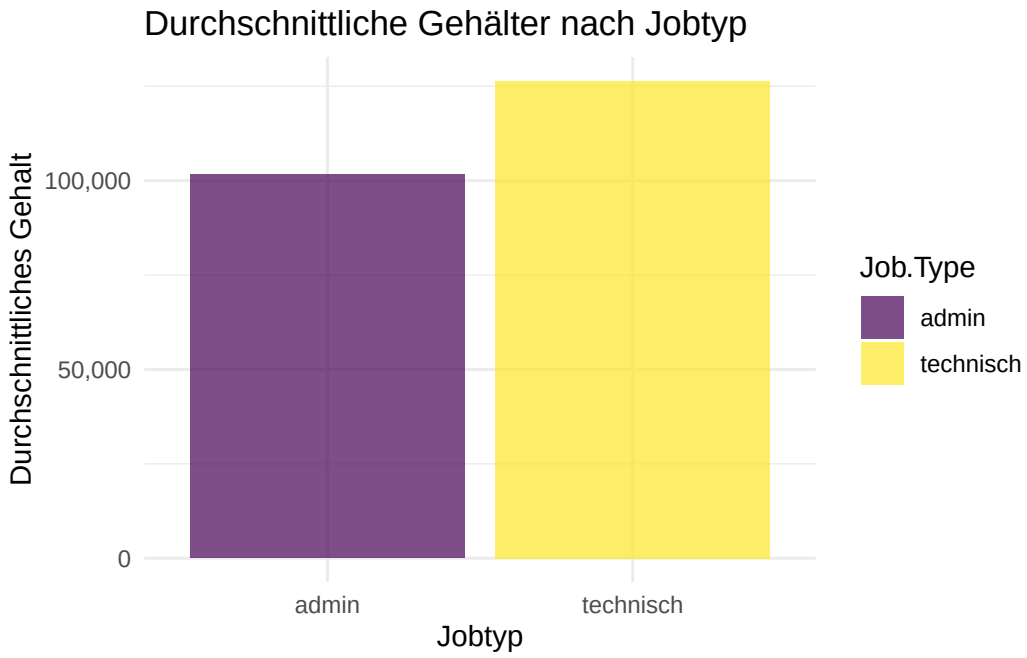
```
[1] 5982
```

6.4.2 6.4.2.: Insgesamt

In diesem Abschnitt wird eine Übersicht der durchschnittlichen Gehälter nach der Sortierung der Jobs nach technisch oder administrativ, erstellt. Dafür werden zunächst die Daten gefiltert. Danach werden die durchschnittlichen Gehälter der beiden Jobtypen berechnet. Anschließend werden dann die durchschnittlichen Gehälter in einen Datenrahmen zusammengeführt. Zum Schluss wird dann das Balkendiagramm erstellt.

```r
technische_jobs <- subset(filtered_data, job_type == 0)
admin_jobs <- subset(filtered_data, job_type == 1)

average_salaries_technical <- mean(technische_jobs$Salary, na.rm = TRUE)
```

```r
average_salaries_admin <- mean(admin_jobs$Salary, na.rm = TRUE)

all_average_salaries <- data.frame(Job.Type = c("technisch", "admin"),
                                   Average.Salary = c(average_salaries_technical, average_

ggplot(all_average_salaries, aes(x = Job.Type, y = Average.Salary, fill = Job.Type)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.7) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Durchschnittliche Gehälter nach Jobtyp",
       x = "Jobtyp",
       y = "Durchschnittliches Gehalt") +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```



In dem Balkendiagramm ist deutlich zu erkennen, dass das durchscnittliche Gehalt bei den technischen Jobs höher ist.

Anschließend werden die Werte der durchschnittlichen Gehälter auch nochmal separat ohne Grafik ausgegeben:

```r
technische_jobs <- subset(filtered_data, job_type == 0)
admin_jobs <- subset(filtered_data, job_type == 1)

average_salaries_technical <- mean(technische_jobs$Salary, na.rm = TRUE)

average_salaries_admin <- mean(admin_jobs$Salary, na.rm = TRUE)

cat("Durchschnittliches Gehalt für technische Jobs:", average_salaries_technical, "\n")
```

```
Durchschnittliches Gehalt für technische Jobs: 126441.3
```

```r
cat("Durchschnittliches Gehalt für administrative Jobs:", average_salaries_admin, "\n")
```

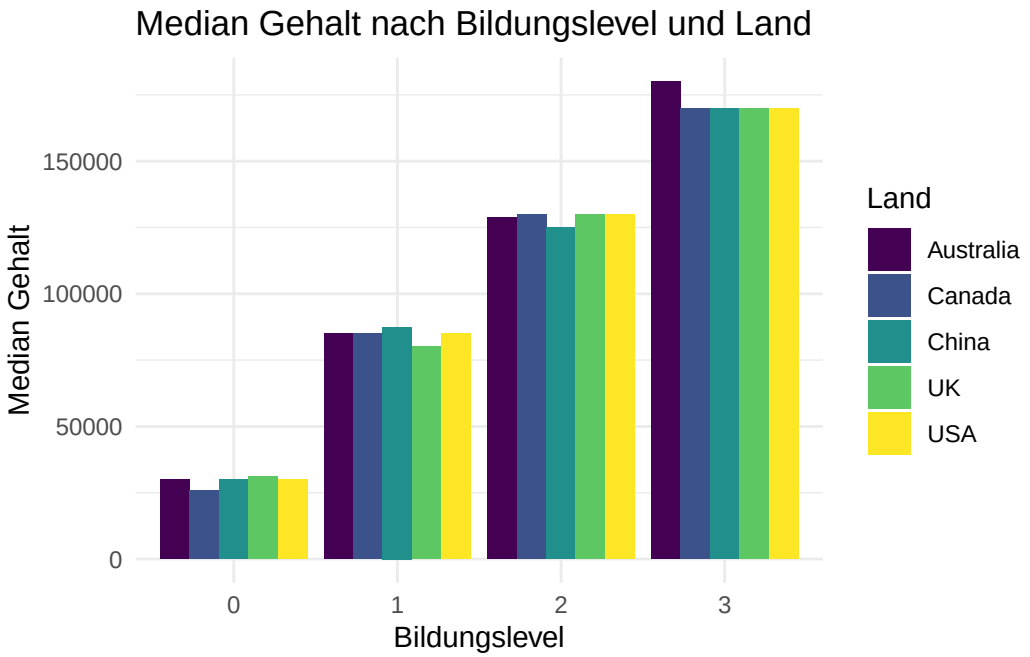
```
Durchschnittliches Gehalt für administrative Jobs: 101595.3
```

Auch hier ist zu erkennen, dass das durchschnittliche Gehalt bei technischen Jobs um rund 20.000 GE höher ist.

6.4.3 6.4.3.: Je Land

Nun wird eine Übersicht der durchschnittlichen Gehälter nach der Sortierung der Jobs, je Land erstellt.

```r
summary_data <- aggregate(Salary ~ Education.Level + Country, data = filtered_data, FUN =

ggplot(summary_data, aes(x = Education.Level, y = Salary, fill = Country)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Median Gehalt nach Bildungslevel und Land",
       x = "Bildungslevel",
       y = "Median Gehalt",
       fill = "Land") +
  theme_minimal()
```



Die These ergibt sich als korrekt und das obwohl in den Admin Jobs auch in seltenen Fällen Manager vertreten sind.

Es lässt sich zudem beobachten, dass Doktoren in Australien ein höheres Gehalt verdienen als in anderen Ländern.

6.5 6.5 Data Scientist verdienen aufgrund der hohen Nachfrage der Berufsgruppe im Schnitt mehr als andere Jobgruppen bei gleichbleibender Erfahrung und Abschlussniveau.

6.5.1 6.5.1.: Begründung

Der Beruf des Data Scientist ist laut dem Harvard Business Review der “Sexiest Job of the 21st Century”.

Siehe Link:

*https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century

Nach Anbetracht dieser Aussage, stellt sich die Frage, ob gerade das Gehalt von Data Scientists, zu dieser Aussage führt.

6.5.2 6.5.2.: Datenaufbereitung

Um die These beantworten zu können, müssen als erstes ein paar Anpassungen an dem Datensatz vorgenommen werden.

Zunächst wird eine Einengung nach Jobs, nach den Stichworten Data, Software, Developer und Egineer durchgeführt. Anschließend werden alle Manager und Direktoren rausgefiltert.

```
filtered_data3 <- filtered_data %>%
  filter(grepl("Data|Software|Developer|Engineer", Job.Title))

job_title_count <- table(filtered_data3$Job.Title)
job_title_df <- data.frame(Job_Title = names(job_title_count), Frequency = as.numeric(job_

job_title_df
```

```
                  Job_Title Frequency
1         Back end Developer       242
2               Data Analyst       391
3             Data Scientist       515
4   Director of Data Science        57
5        Front end Developer       239
6        Front End Developer        31
7        Full Stack Engineer       304
8           Project Engineer       317
9         Software Developer       186
10         Software Engineer       809
11 Software Engineer Manager       376
12             Web Developer       129
```

```
filtered_data3 <- filtered_data3 %>%
  filter(!grepl("Director|Manager", Job.Title))

job_title_count <- table(filtered_data3$Job.Title)
job_title_df <- data.frame(Job_Title = names(job_title_count), Frequency = as.numeric(job_

job_title_df
```

```
            Job_Title Frequency
1   Back end Developer       242
2         Data Analyst       391
```

```
3       Data Scientist       515
4  Front end Developer       239
5  Front End Developer        31
6  Full Stack Engineer       304
7     Project Engineer       317
8   Software Developer       186
9    Software Engineer       809
10       Web Developer       129
```

Anschließend wird, um die Leserlichkeit zu verbessern, eine Aufteilung der Jobs durchgeführt. Hierzu werden die Jobs in eine neue Spalte "data_job" aufgeteilt.

```r
filtered_data3 <- filtered_data3 %>%
  mutate(data_job = ifelse(grepl("Data", Job.Title), 1, 0))
```

```r
count_0 <- sum(filtered_data3$data_job == 0, na.rm = TRUE)
count_1 <- sum(filtered_data3$data_job == 1, na.rm = TRUE)

cat("Anzahl der Zeilen mit dem Wert 0 bei data_job (Data Scientists & Engineers):", count_
```

```
Anzahl der Zeilen mit dem Wert 0 bei data_job (Data Scientists & Engineers): 2257
```

```r
cat("Anzahl der Zeilen mit dem Wert 1 bei data_job (Software Engineers & Co):", count_1, "
```

```
Anzahl der Zeilen mit dem Wert 1 bei data_job (Software Engineers & Co): 906
```

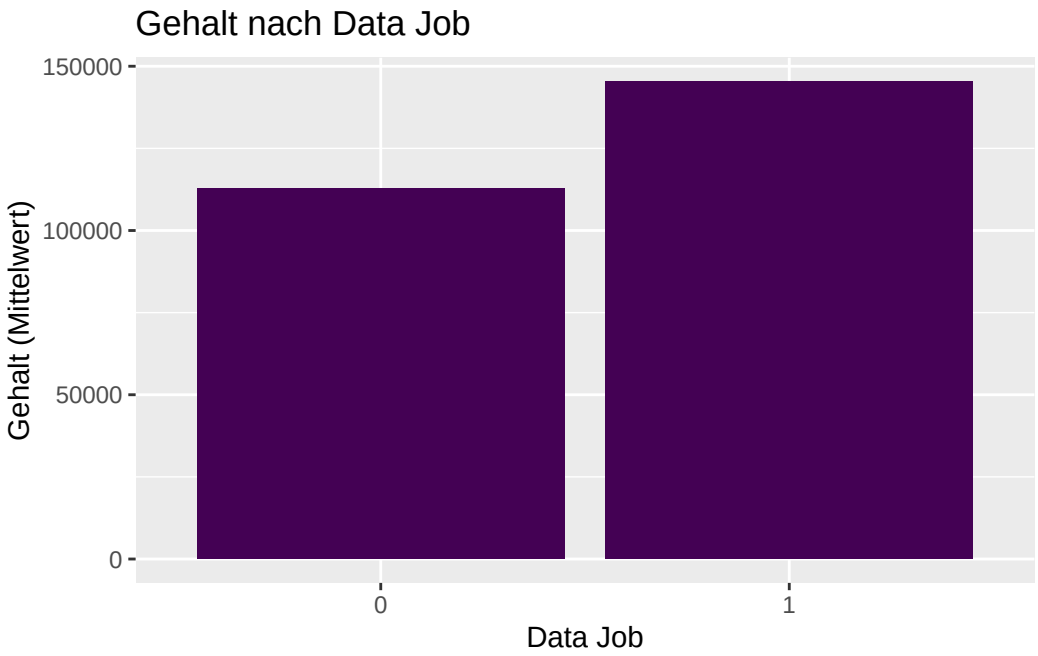
6.5.3 6.5.3.: Balkendiagramm

Um die Übersicht zu verbessern, wird die Berufserfahrung in Quantile eingeteilt. Die Bildungsniveaus hingegen wurden bereits im Vorfeld in vier Werte eingeteilt.

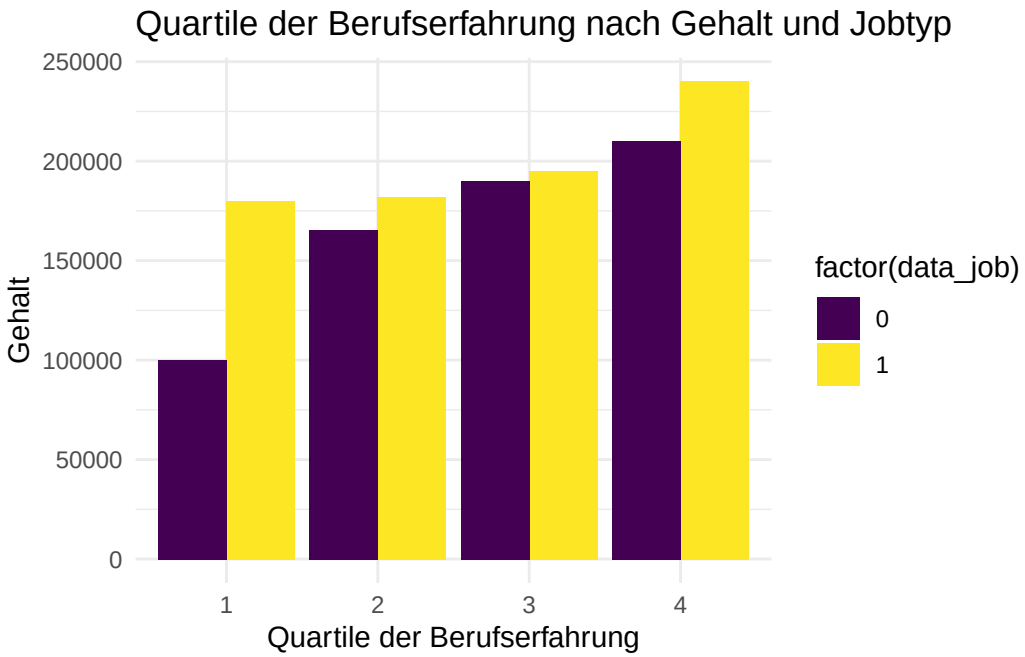
Als erstes werden die Quantile der Berufserfahrung berechnet und anschließend ein Balkendiagramm für den Datensatz "data_job" erstellt, um das Gehalt zu vergleichen.

```r
filtered_data4 <- filtered_data3 %>%
  mutate(Experience_Quartile = ntile(Years.Of.Experience, 4))

ggplot(filtered_data4, aes(x = factor(data_job), y = Salary)) +
  stat_summary(fun = "mean", geom = "bar", position = "dodge", fill = viridis(2)[1]) +
```

```
  labs(title = "Gehalt nach Data Job",
       x = "Data Job",
       y = "Gehalt (Mittelwert)")
```




```
ggplot(filtered_data4, aes(y = Salary, x = factor(Experience_Quartile))) +
  geom_bar(stat = "identity", position = "dodge", aes(fill = factor(data_job))) +
  labs(title = "Quartile der Berufserfahrung nach Gehalt und Jobtyp",
       x = "Quartile der Berufserfahrung",
       y = "Gehalt") +
  theme_minimal() +
  scale_fill_viridis(discrete = TRUE)
```



Zum Verständnis hier noch einmal die Codes des Bidlungsniveaus

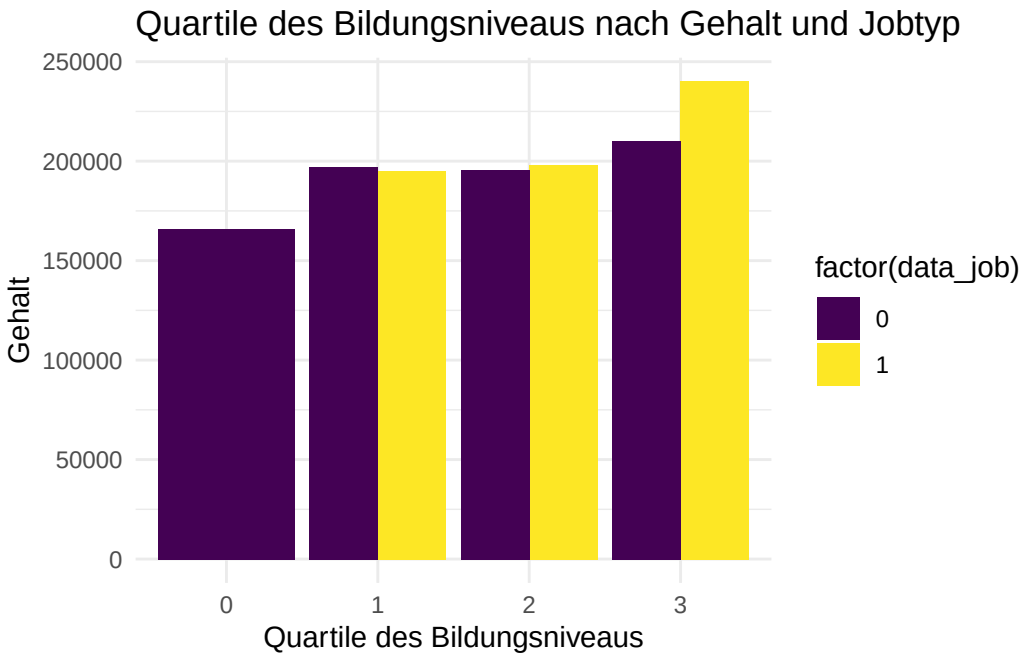
0 = High School Abschluss

1 = Bachelor

2 = Master

3 = Doctor

```
ggplot(filtered_data4, aes(y = Salary, x = factor(Education.Level))) +
  geom_bar(stat = "identity", position = "dodge", aes(fill = factor(data_job))) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Quartile des Bildungsniveaus nach Gehalt und Jobtyp",
       x = "Quartile des Bildungsniveaus",
       y = "Gehalt") +
  theme_minimal()
```



Hier ist zu erkennen, dass Data Scientists mehr als Arbeitnehmer aus der Software Enginnering % Co Gruppe. Es muss jedoch beachtet werden, dass die “data- Gruppe” mindestens einen Bachelor besitzt und erst nach einem Master mehr verdient, als ihr Counterpart. Im Bezug auf die Berufserfahrung lässt sich feststellen, dass es in jedem Quantil einen höheres Gehaltsniveau bei der “data-Gruppe” gibt.

Trotzdem lässt sich sagen, dass die These korrekt ist.

6.6 6.6 Die Gehälter sind in Ländern mit einem höheren BIP pro Kopf höher

Zum Verständnis sind hier einmal die BIP’s pro Kopf aus externen Quellen aufgelistet, da diese nicht im Datensatz vertreten sind:

Australien 64.813,85 US-Dollar Quelle: Australien - BIP pro Kopf bis 2028 | Statista

Canada 53.246,98 US-Dollar Quelle: Kanada - BIP pro Kopf bis 2028 | Statista

China 12.541,40 US-Dollar Quelle: China - BIP pro Kopf bis 2028 | Statista

UK 48.912,78 US-Dollar Quelle: Großbritannien - BIP pro Kopf bis 2028 | Statista

USA 76.343 US-Dollar Quelle: USA - BIP pro Kopf bis 2028 | Statista

6.6.1 6.6.1.: Aufbereitung

Vorab muss ein wenig Vorarbeit geleistet werden.

Zunächst werden die BIP-Werte den entsprechend Ländern zugewiesen.

```
filtered_data$BIP_Per_Person <- NA

filtered_data$BIP_Per_Person[filtered_data$Country == "Australia"] <- 64813.85
filtered_data$BIP_Per_Person[filtered_data$Country == "Canada"] <- 53246.98
filtered_data$BIP_Per_Person[filtered_data$Country == "China"] <- 12541.40
filtered_data$BIP_Per_Person[filtered_data$Country == "UK"] <- 48912.78
filtered_data$BIP_Per_Person[filtered_data$Country == "USA"] <- 76343.00
```

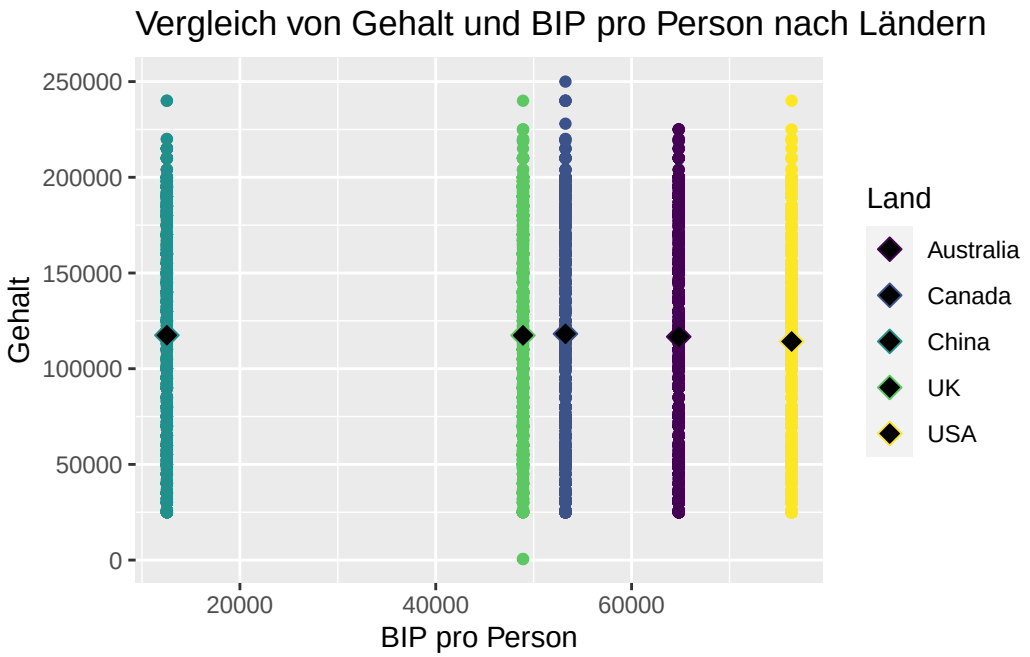
```
head(filtered_data3)
```

```
# A tibble: 6 x 15
  Job.Title    job_count   Age Gender Education.Level Years.Of.Experience Salary
  <chr>            <int> <dbl> <chr>  <fct>                         <dbl>  <dbl>
1 Back end De~       242    33 Female 2                                 5 110000
2 Back end De~       242    32 Male   1                                 4  95000
3 Back end De~       242    26 Female 2                                 3  90000
4 Back end De~       242    26 Female 2                                 2  70000
5 Back end De~       242    24 Female 1                                 1  60000
6 Back end De~       242    26 Female 2                                 3  90000
# i 8 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>,
#   ID <int>, job_type <dbl>, Expat <dbl>, data_job <dbl>
```

6.6.2 6.6.2.: Visualisierung und Berechnung

Mithilfe von Streudiagrammen und Berechnungen wird nun versucht die These zu widerlegen oder als richtig markieren zu können.

```
ggplot(filtered_data, aes(x = BIP_Per_Person, y = Salary, color = Country)) +
  geom_point() +
  stat_summary(fun = mean, geom = "point", shape = 23, size = 3, fill = "black") +
  labs(title = "Vergleich von Gehalt und BIP pro Person nach Ländern",
       x = "BIP pro Person",
       y = "Gehalt",
       color = "Land") +
  scale_color_viridis(discrete = TRUE)
```



Mithilfe dieses Streudiagramm ist viel zu erkennen. Deswegen werden weitere Berechnungen angestellt.

Zunächst werden die Mittelwerte nach Land berechnet und anschließend die Mittelwerte nach Land in Bezug auf das BIP.

```
mean_salaries_by_country <- filtered_data3 %>%
  group_by(Country) %>%
  summarise(mean_salary = mean(Salary, na.rm = TRUE))

mean_salaries_by_country
```

```
# A tibble: 5 x 2
  Country   mean_salary
  <chr>           <dbl>
1 Australia     121610.
2 Canada        125466.
3 China         120395.
4 UK            123564.
5 USA           119265.
```

```r
mean_salaries_by_country <- filtered_data %>%
  group_by(Country) %>%
  summarise(mean_salary = mean(Salary, na.rm = TRUE),
            BIP_Per_Person = first(BIP_Per_Person))
mean_salaries_by_country
```

```
# A tibble: 5 x 3
  Country   mean_salary BIP_Per_Person
  <chr>           <dbl>          <dbl>
1 Australia     116704.         64814.
2 Canada        118200.         53247.
3 China         117480.         12541.
4 UK            117412.         48913.
5 USA           114209.         76343
```

Außerdem wird noch die Korrelation zwischen der Salary und dem BIP Berechnet.

```r
cor(filtered_data$Salary, filtered_data$BIP_Per_Person, use = "complete.obs")
```

```
[1] -0.01632912
```

Es geht hervor, dass es keine große Korrelation ziwschen dem Gehalt und dem BIP gibt.

Nun wird das Ganze noch ohne China durchgeführt:

Zunächst wird ein Datensatz ohne China erstellt und anschließend wird die Korrelation erneut berechnet. Zudem wird die Anzahl der Datensätze mit dem land “China” in dem neuen Datensatz festgehalten.

```r
filtered_data_no_china <- filtered_data %>% filter(Country != "China")
```

```r
cor(filtered_data_no_china$Salary, filtered_data_no_china$BIP_Per_Person, use = "complete.
```

```
[1] -0.02614532
```

```r
count_china <- filtered_data_no_china %>% filter(Country == "China") %>% nrow()
count_china
```

```
[1] 0
```

Anhand der Berechnung und des Streudiagramms, lässt sich die These als nicht korrekt beantworten. Es gibt eine negative Korrelation zwischen dem Gehalt, welches ein Arbeitnehmer erhält und dem BIP des jeweiligen Landes. Selbst wenn China aus der Berechnung rausgenommen wird, welches aufgrund der hohen Einwohnerzahl und diversen Wirtschaft (Sonderverwaltungszonen und Kommunismus) einen sehr niedrigen BIP hat.

Nun stellt sich die Frage, ob eventuell in dem Datensatz bewusst Jobs mit hohem Gehalt gewählt wurden. Oder wurden Werte von spezifischen Firmen, die international tätig sind und gut bezahlen, genommen?

Siehe folgende Links:

*https://de.wikipedia.org/wiki/Politisches_System_der_Volksrepublik_China

*https://de.wikipedia.org/wiki/Sonderverwaltungszone

7 7. Regressionen

In dem folgenden Absatz werden die Regressionen durchgeführt.

7.1 7.1.: Einfache Lineare Regression Gehalt und Arbeitserfahrung

Zu Beginn wird eine einfache lineare Regression mit dem Gehalt und der Arbeitserfahrung durchgeführt

Die Korrelation von Arbeitserfahrung und Gehalt liegt bei 0.81, weshalb wir uns entscheiden haben diese als Regression zu verwenden. Des weiteren nutzen wir zusätzlich das Alter und das Bildungsniveau.

Die Korrelation zwischen der Arbeitserfahrung und Gehalt liegt bei 0.81. Deshalb wurde entschieden diese als Regression zu verwenden. Des Weiteren wird auch das Alter, so wie das Bildungsniveau verwendet.

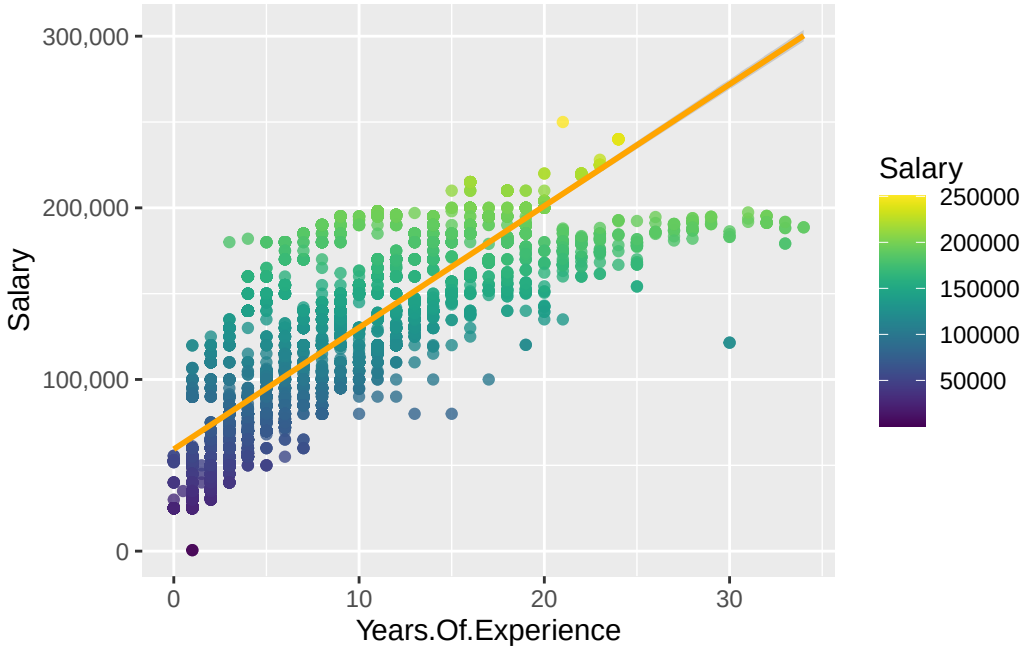
7.1.1 7.1.1.: Korrelationsmatrix

Zunächst wird ein Streudiagramm über die Beziehung zwischen dem Gehalt und der Berufserfahrung erstellt. Außerdem wird eine lineare Regressionsgerade eingefügt, um den Trend besser analysieren zu können.

```
filtered_data %>%
  ggplot() +
  aes(y = Salary, x = Years.Of.Experience) +
```

```
    geom_point(aes(color = Salary), alpha = 0.8) +
    geom_smooth(method = lm, color = "orange") +
    scale_color_viridis(option = "D") +
    scale_y_continuous(labels = scales::comma)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Anhand dieser Grafik kann gesagt werden, dass der Trend deutlich nach oben geht, je mehr Berufserfahrung eine Person hat.

7.1.2 7.1.2.: Datenaufbereitung

Um mit der Regressionsanalyse fortzufahren, müssen noch Änderungen am Datensatz durchgeführt werden.

Zunächst müssen alle Werte, die nicht für die Regression relevant sind, entfernt werden. Deswegen werden nur “Salary” und “Years of Experience” behalten.

```
filtered_data5 <- filtered_data %>%
  select(Salary, Years.Of.Experience)
```

7.1.2.1 7.1.2.1 Z-Skalierung

Zuerst wird eine Z-Skalierung von filtered_data_5 durchgeführt.

Die Z-Skalierung ist eine Methode zur Standardisierung von numerischen Variablen. Bei der Z-Skalierung werden alle numerischen Werte mit Ausnahme des Vorhersagewerts (Salary) skaliert, um die Auswirkungen von Ausreißern zu minimieren.

```
filtered_data5_z <- filtered_data5
filtered_data5_z$Years.Of.Experience <- scale(filtered_data5$Years.Of.Experience)
```

Ergebnis der Z-Skalierung:

```
summary(filtered_data5_z)
```

```
     Salary       Years.Of.Experience.V1
 Min.   :   550   Min.   :-1.350859
 1st Qu.: 70000   1st Qu.:-0.850649
 Median :120000   Median :-0.183702
 Mean   :116787   Mean   : 0.000000
 3rd Qu.:160000   3rd Qu.: 0.649981
 Max.   :250000   Max.   : 4.318187
```

Überprüfung der Standardabweichung für Arbeitserfahrung

```
sd(filtered_data5_z$Years.Of.Experience)
```

```
[1] 1
```

Aufteilung in Test- und Trainingsdaten:

```
set.seed(007)

filtered_data5_z <- initial_split(filtered_data5_z, prop = 0.8, strata = Years.Of.Experien

fd5_train <- training(filtered_data5_z)
fd5_test <- testing(filtered_data5_z)
```

Dieser Code teilt den Datensatz filtered_data5_z in Trainings- und Testdaten auf, um eine lineare Regression durchzuführen. Die Funktion set.seed(007) initialisiert den Zufallszahlengenerator mit einer festen Zahl, um sicherzustellen, dass die Ergebnisse bei jedem Durchlauf

reproduzierbar sind. Die Funktion initial_split() aus dem Paket rsample teilt den Datensatz in Trainings- und Testdaten auf. Der Parameter prop = 0,8 gibt an, dass 80% der Daten für das Training verwendet werden sollen, während die restlichen 20% für das Testen verwendet werden. Der Parameter strata = Years.Of.Experience sorgt dafür, dass die Daten nach dem Gehalts-Wert stratifiziert werden, um sicherzustellen, dass die Trainings- und Testdaten eine ähnliche Verteilung von Gehalts-Werten aufweisen. Die Funktion training() extrahiert die Trainingsdaten aus dem aufgeteilten Datensatz, während testing() die Testdaten extrahiert.

7.1.3 7.1.3.: Modell Initialisieren

```
lm_model <- linear_reg() |> set_engine("lm")
```

Lineare Regression von Salary (basierend auf der Berufserfahrung):

```
lm_fit <- lm_model |> fit(Salary ~ Years.Of.Experience, data = fd5_train)
```

Zusammenfassung vom Ergebnis:

```
summary <- lm_fit |> extract_fit_engine() |> summary()
summary
```

```
Call:
stats::lm(formula = Salary ~ Years.Of.Experience, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-149686  -22470   -6278   21338   94530

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         116445.0      429.6  271.05   <2e-16 ***
Years.Of.Experience  42366.7      428.0   98.98   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30730 on 5115 degrees of freedom
Multiple R-squared:  0.657, Adjusted R-squared:  0.6569
F-statistic:  9797 on 1 and 5115 DF,  p-value: < 2.2e-16
```

Vorhersagen auf Trainings- und Testdatensatz:

Nun werden Vorhersagen für die Tranings-, sowie Testdaten erstellt. Anschließend werden die tatsächlichen "Salary"-Werte mit den Vorhersagen kombiniert. Dies geschieht um zwei seperate Datenrahmen zu erstellen.

```
pred_train <- predict(lm_fit, new_data = fd5_train) |> rename("pred_train" = ".pred")
pred_test <- predict(lm_fit, new_data = fd5_test) |>  rename("pred_test" = ".pred")

compare_train <- fd5_train |>
  select(Salary) |>
  bind_cols(pred_train)
head(compare_train)
```

```
# A tibble: 6 x 2
  Salary pred_train
   <dbl>      <dbl>
1  60000     66278.
2  90000     80406.
3  85000     80406.
4  55000     66278.
5  75000     73342.
6  60000     66278.
```

```
compare_test <- fd5_test |>
  select(Salary) |>
  bind_cols(pred_test)
head(compare_test)
```

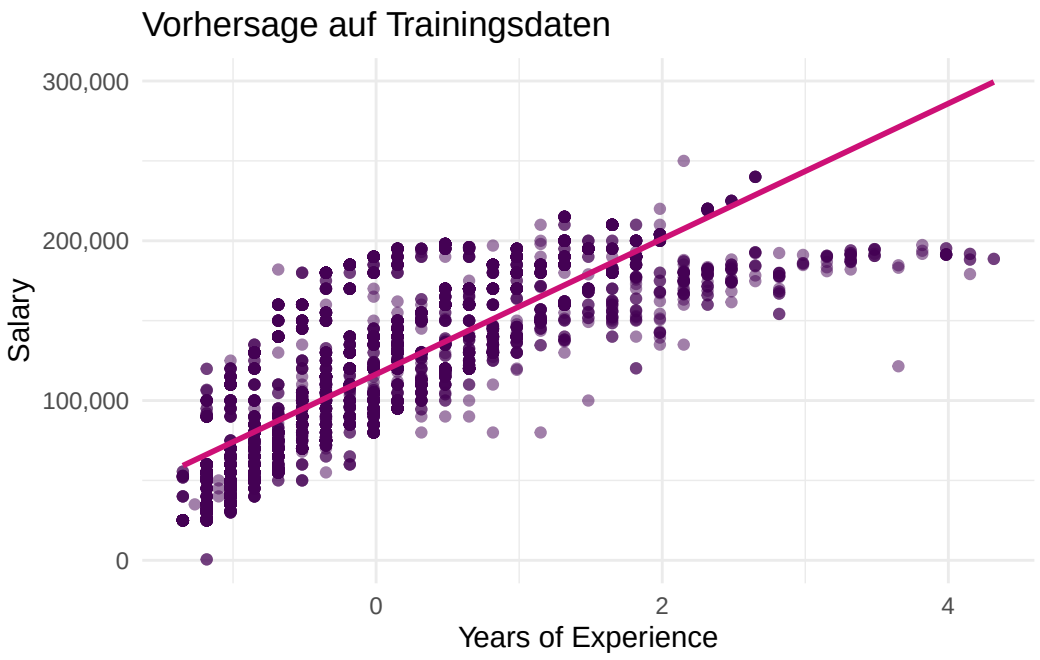
```
# A tibble: 6 x 2
  Salary pred_test
   <dbl>     <dbl>
1  90000    80406.
2  70000    73342.
3  60000    66278.
4 125000   101598.
5 130000   101598.
6  95000    87470.
```

7.1.4 7.1.4.: Grafische Darstellung

Im folgenden Codechunk, werden zwei Grafiken, zum Einen für die Testdaten und zum Anderen für die Trainingsdaten erstellt.

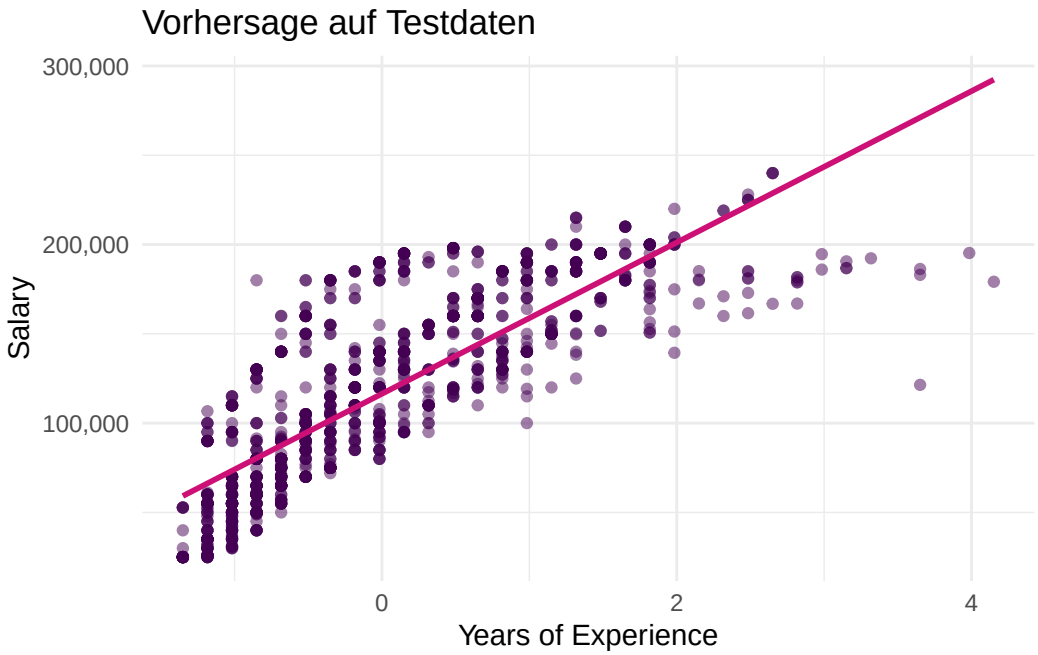
```
train_data <- cbind(fd5_train, pred_train)
test_data <- cbind(fd5_test, pred_test)

ggplot(train_data, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point(color = viridis(0.50), alpha = 0.5) +
  geom_line(aes(y = pred_train), color = "deeppink3", size = 1) +
  labs(title = "Vorhersage auf Trainingsdaten",
       x = "Years of Experience",
       y = "Salary") +
  scale_color_identity() +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```




```
ggplot(test_data, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point(color = viridis(0.50), alpha = 0.5) +
  geom_line(aes(y = pred_test), color = "deeppink3", size = 1) +
  labs(title = "Vorhersage auf Testdaten",
```

```
      x = "Years of Experience",
      y = "Salary") +
scale_color_identity() +
scale_y_continuous(labels = scales::comma) +
theme_minimal()
```



Anhand der zwei Grafiken ist zu erkennen, dass die unterschiedlichen Werte bei der Vorhersage der Trainingsdaten deutlich dichter zusammen liegen, als bei den Testdaten. Trotzdem sind die beiden Grafiken gleich, was den Trend angeht. Dies ist Mithilfe der Regressionsgeraden zu erkennen.

7.1.5 7.1.5.: Fehler

Mithilfe der "rmse"-Funktion, wird die Quadratwurzel aus dem Durchschnitt der quadrierten Differenzen zwischen den vorhergesagten und den tatsächlichen Werten.

Trainingsfehler:

```
rmse(compare_train, Salary, pred_train)
```

```
# A tibble: 1 x 3
```

```
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      30725.
```

Die durchschnittliche Abweichung beträgt rund 31.000.

Testfehler:

```
rmse(compare_test, Salary, pred_test)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      30764.
```

Auch hier beträgt die durchschnittliche Abweichung rund 31.000.

Zur Einordnung der Fehler wird die Verteilung von Gehalt angesehen:

```
describe(filtered_data5, Salary)
```

```
variable = Salary
type     = double
na       = 0 of 6 398 (0%)
unique   = 432
min|max  = 550 | 250 000
q05|q95  = 35 000 | 195 000
q25|q75  = 70 000 | 160 000
median   = 120 000
mean     = 116 787.2
```

Verteilung von Arbeitserfahrung:

```
describe(filtered_data5, Years.Of.Experience)
```

```
variable = Years.Of.Experience
type     = double
na       = 0 of 6 398 (0%)
unique   = 37
min|max  = 0 | 34
```

```
q05|q95  = 1 | 19
q25|q75  = 3 | 12
median   = 7
mean     = 8.101751
```

7.1.6 7.1.6.: Residuen

Residuen (auch Fehler oder Residuen genannt) sind die Unterschiede zwischen den beobachteten Werten und den vorhergesagten Werten in einem Regressionsmodell. Sie stellen die Abweichungen zwischen den tatsächlichen Daten und den durch das Modell vorhergesagten Werten dar. Idealerweise sollten die Residuen normalverteilt sein, um sicherzustellen, dass das Regressionsmodell angemessen ist.

Es ist üblich, die Residuen sowohl für Trainingsdaten als auch für Testdaten zu überprüfen, um die Leistung des Modells auf beiden Datensätzen zu evaluieren. Hier sind einige Gründe, warum es wichtig ist, die Residuen auf beiden Datensätzen zu betrachten:

1. **Trainingsdaten:**
    - Die Residuen der Trainingsdaten geben Ihnen einen Einblick in die Leistung des Modells auf den Daten, auf denen es trainiert wurde. Wenn die Residuen auf den Trainingsdaten ungewöhnliche Muster aufweisen, kann dies auf Modellprobleme oder Overfitting hinweisen.
2. **Testdaten:**
    - Die Residuen der Testdaten ermöglichen es Ihnen, die Generalisierungsfähigkeit des Modells auf neuen, nicht trainierten Daten zu überprüfen. Ein Modell kann auf den Trainingsdaten gut funktionieren, aber die Residuen auf den Testdaten können Ihnen sagen, wie gut es sich auf unbekannte Daten verallgemeinert.

Nun werden die Risiduen mit der “augment()”-Funktion auf die Trainings-, sowie Testdaten abgerufen. Anschließend werden Sie ausgegeben.

```r
residuals_train <- augment(lm_fit, new_data = fd5_train) %>% select(.resid)

residuals_test <- augment(lm_fit, new_data = fd5_test) %>% select(.resid)

print(residuals_train)
```

```
# A tibble: 5,117 x 1
    .resid
     <dbl>
```

```
 1  -6278.
 2   9594.
 3   4594.
 4 -11278.
 5   1658.
 6  -6278.
 7 -11278.
 8   9594.
 9  -3342.
10  -6278.
# i 5,107 more rows
```

```
print(residuals_test)
```

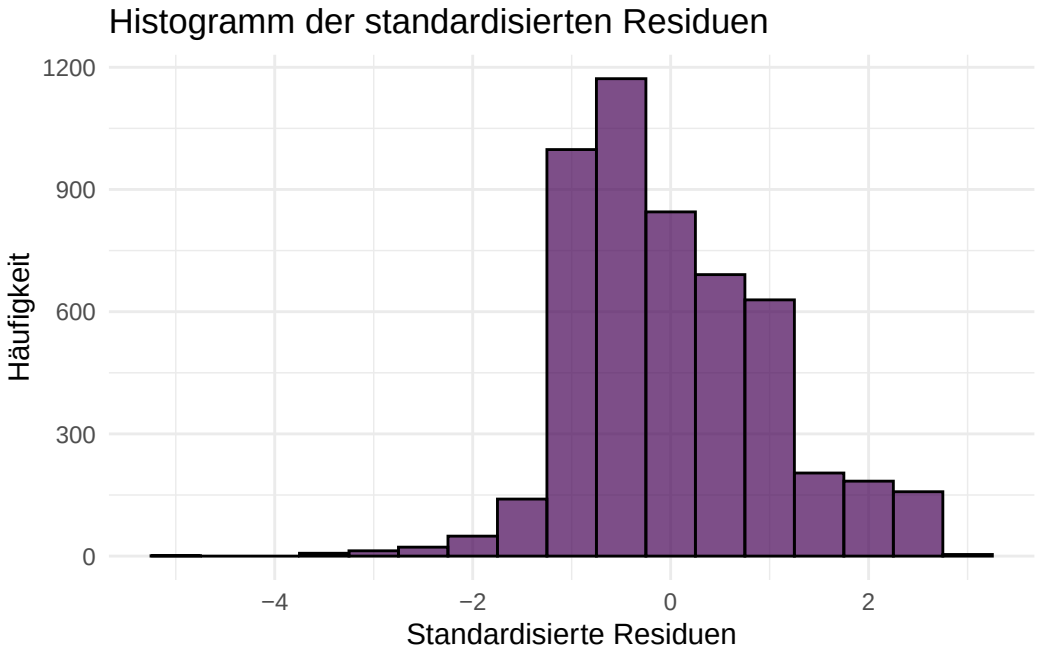
```
# A tibble: 1,281 x 1
   .resid
    <dbl>
 1  9594.
 2 -3342.
 3 -6278.
 4 23402.
 5 28402.
 6  7530.
 7 -3342.
 8 28402.
 9  9594.
10 -3342.
# i 1,271 more rows
```

Anschließend werden nun die Risiduen in einem Histogramm ausgegeben, nachdem Sie z-skaliert wurden.

Histogramm der Risiduen der Trainingsdaten:

```
residuals_train$standardized_resid <- scale(residuals_train$.resid)

ggplot(data = residuals_train, aes(x = standardized_resid)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
       x = "Standardisierte Residuen",
       y = "Häufigkeit") +
```

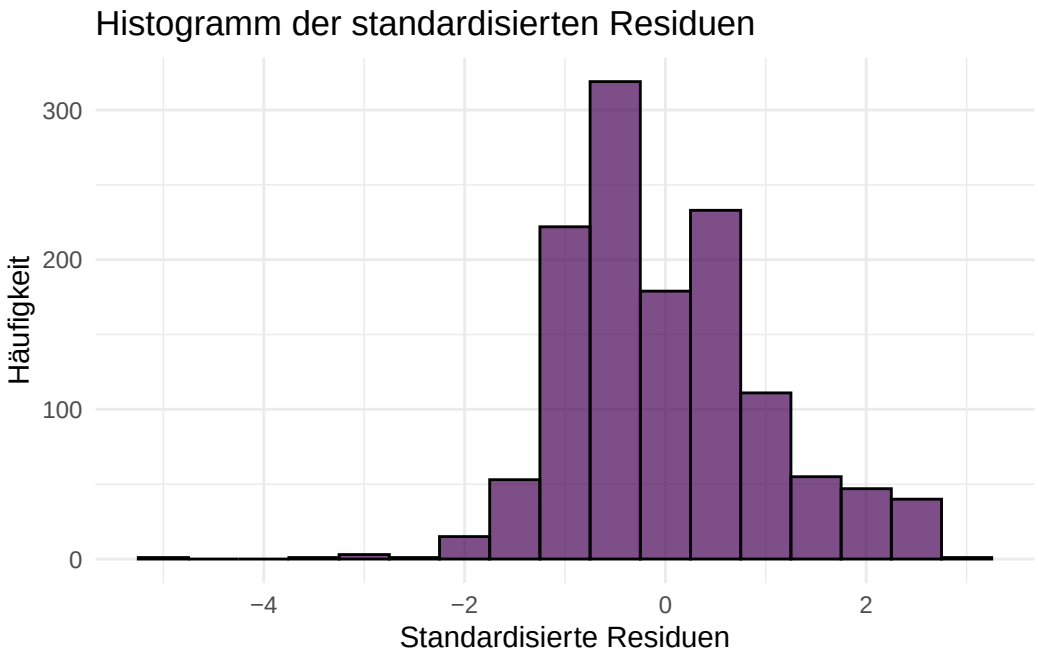
```
  scale_fill_viridis() +
  theme_minimal()
```



Zu erkennen ist, dass die Werte zwischen 2 und -2 am häufigsten vertreten sind.

Histogramm der Risiduen der Testdaten:

```
residuals_test$standardized_resid <- scale(residuals_test$.resid)

ggplot(data = residuals_test, aes(x = standardized_resid)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
       x = "Standardisierte Residuen",
       y = "Häufigkeit") +
  scale_fill_viridis() +
  theme_minimal()
```



Auch hier sind die Risiduen zwischen 2 und -2 am häufigsten.

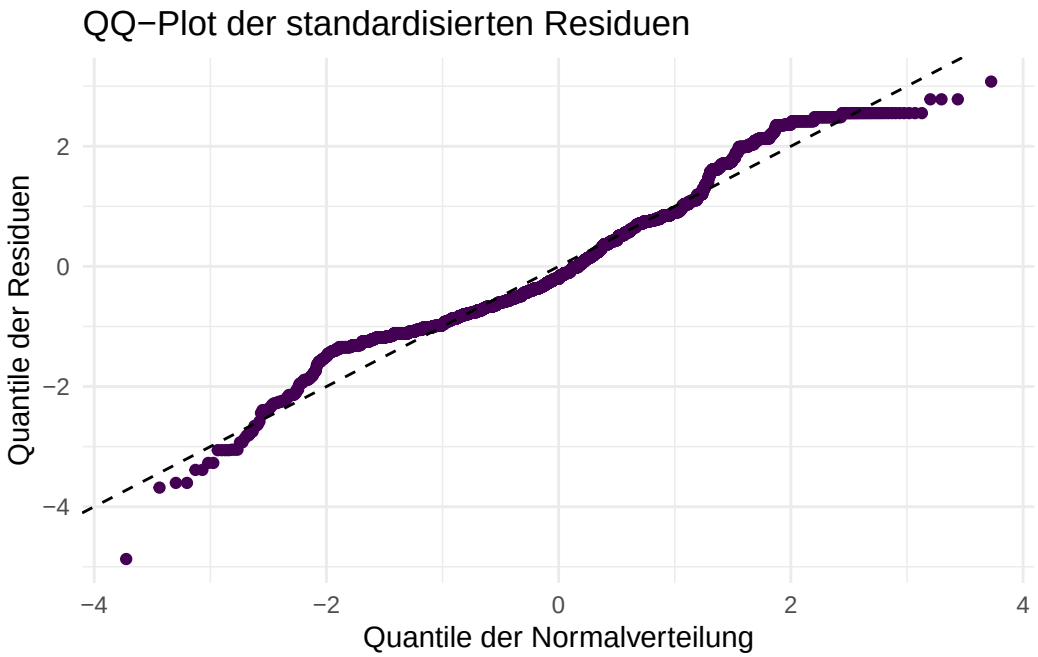
Aus den beiden Histogrammen geht also hervor, dass die Abweichungen zwischen den vorhergesagten Werten und tatsächlichen Werten nicht wirklich groß sind.

7.1.7 7.1.7.: Q-Q-Plot

Das "Quantil-Quantil-Diagramm" überprüft Risiduen auf Ihre Normalverteilung. Hierbei werden die Quantile der standartisierten Risiduen gegen die QUantile der Normalverteilung gestellt. Im Normalfall sollten die Punkte entlang der Diagonale (x=y) streuen.

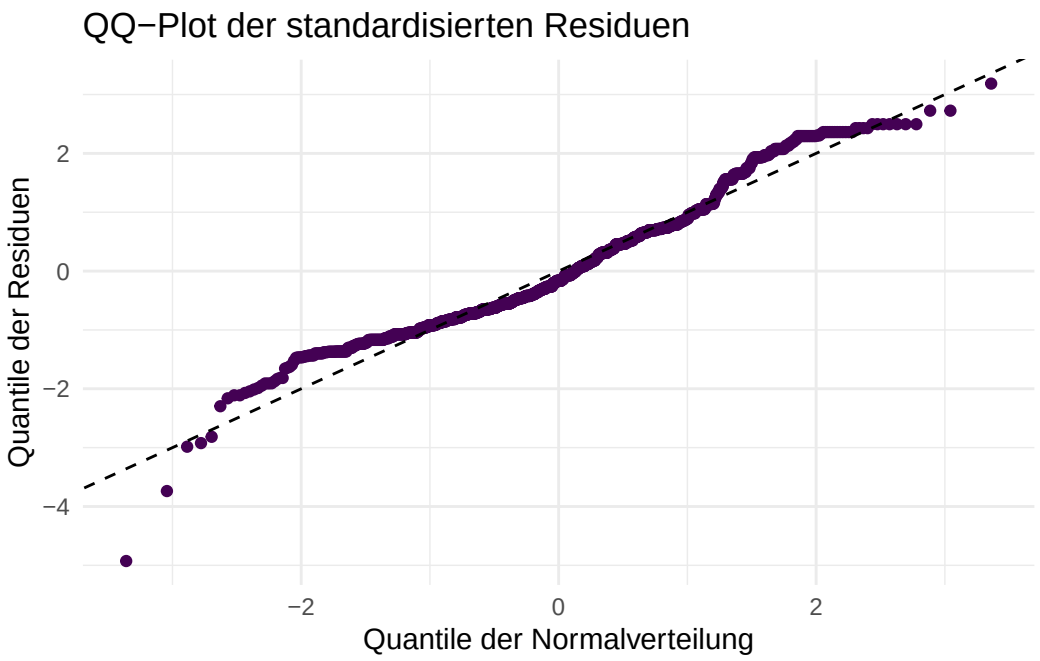
QQ-Plot für der Risiduen für die Trainingsdaten:

```
ggplot(data = residuals_train, aes(sample = standardized_resid)) +
  stat_qq(distribution = qnorm, dparams = list(mean = 0, sd = 1), color = viridis(1)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black") +
  labs(title = "QQ-Plot der standardisierten Residuen",
       x = "Quantile der Normalverteilung",
       y = "Quantile der Residuen") +
  scale_color_viridis() +
  theme_minimal()
```



QQ-Plot für der Risiduen für die Testdaten:

```
residuals_test$standardized_resid <- scale(residuals_test$.resid)
ggplot(data = residuals_test, aes(sample = standardized_resid)) +
  stat_qq(distribution = qnorm, dparams = list(mean = 0, sd = 1), color = viridis(1)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black") +
  labs(title = "QQ-Plot der standardisierten Residuen",
       x = "Quantile der Normalverteilung",
       y = "Quantile der Residuen") +
  scale_color_viridis() +
  theme_minimal()
```



Die beiden Q-Q-Plots bilden die "Ausreißer" vom Gehalt nach oben oder unten, entlang der 45-Grad-Linie ab, welche bereits im ersten Diagramm von 7.1 zu sehen sind. Es ist zu erkennen, dass die Risiduen nicht perfekt normalverteilt sind. Es besteht die Möglichkeit, dass esDatenpunkte gibt, welche die Risiduen beeinflussen. Aufgrund der explorativen Datenanalyse ist bereits bekannt, das Jobs mit hoher Berufserfahrung oft Führungsverantwortung beinhalten, welche nochmals zusätzlich monetär honoriert wird. Rechts oben im Graph ist eine flache Linie zu erkennen. Diese deutet auf einen Gehaltscap hin. Die "Ausreißer" am unteren Ende lassen sichh durch z.B Einstiegsjobs, sowie Niedriglohnjobs ohne Hochschulabschluss erläutern. Desweiteren ist noch anzumerken, dass Ausbildungsberufe nicht beachtet werden.

7.2 7.2.: Mehrfache Lineare Regression

In dem folgenden Absatz wird eine lineare mehrfache Regression durchgeführt.

7.2.1 7.2.1.: Vorbereitung

Zunächst muss eine gewisse Vorbereitung getroffen werden.

In diesem Fall wird einmal der Adjusted R-Squad-Wert berechnet. Dieser gibt an wie gut eine Variable, in diesem Fall "Years of Experience" die Variationen in der abhängigen Variable " Salary" in Ihrem Modell erklärt. Dies geschieht unter der Berücksichtigung der Anzahl von unabhängigen Variablen.

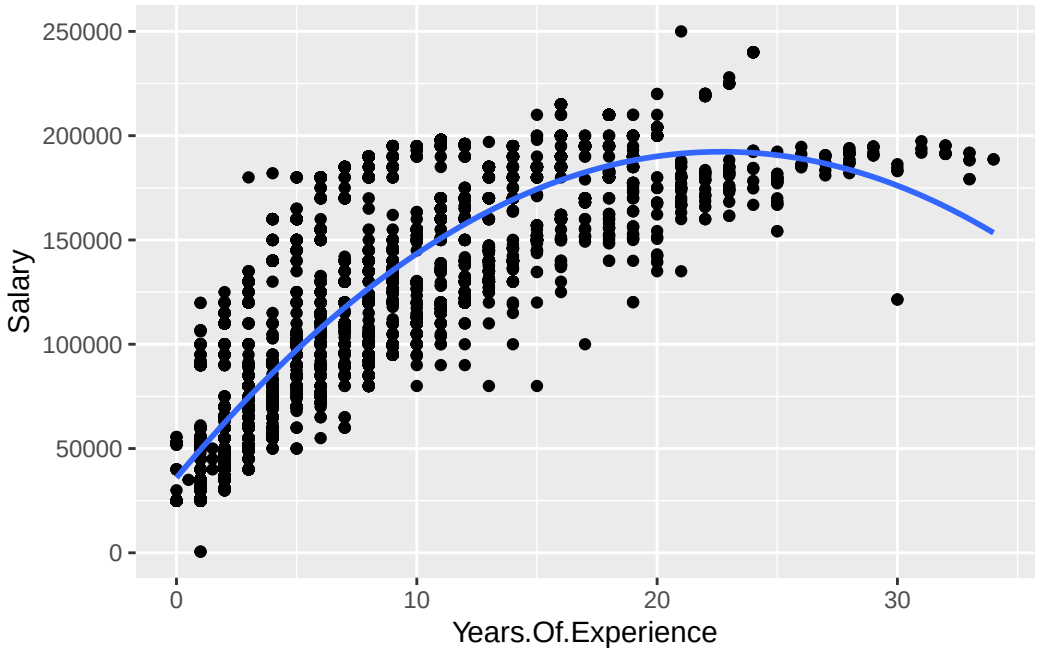
```r
linear_model <- lm(Salary ~ Years.Of.Experience, data = filtered_data3)

adjusted_r_squared <- summary(linear_model)$adj.r.squared

cat("Adjusted R-squared:", adjusted_r_squared, "\n")
```

```
Adjusted R-squared: 0.5713572
```

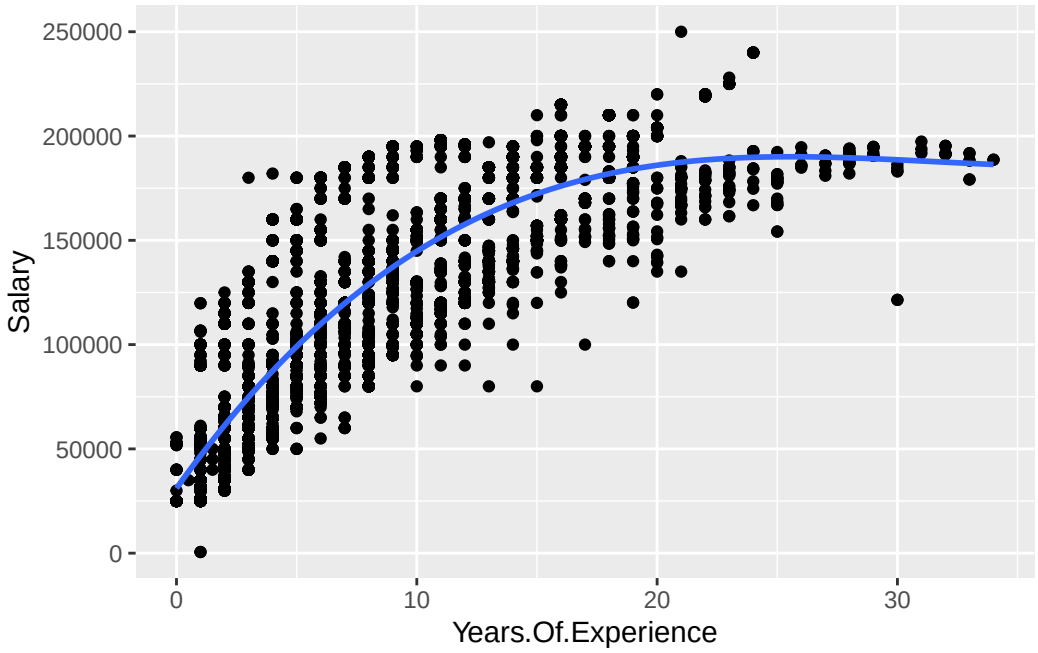
Der Adjusted R-squared-Wert liegt zwischen 0 und 1. In diesem Fall bedeutet 0.5713572, dass etwa 57,14% der Variationen in der abhängigen Variable "Salary" durch die unabhängige Variable "Years.Of.Experience" im Modell erklärt werden können. Ein höherer Wert wäre Wünschenswert.

Nun werden Streudiagramme mit einer glättenden Funktion erstellt. Hierbei werden 3 Diagramme erstellt, wobei mit unterschiedlichen Potenzen gerechnet wird.

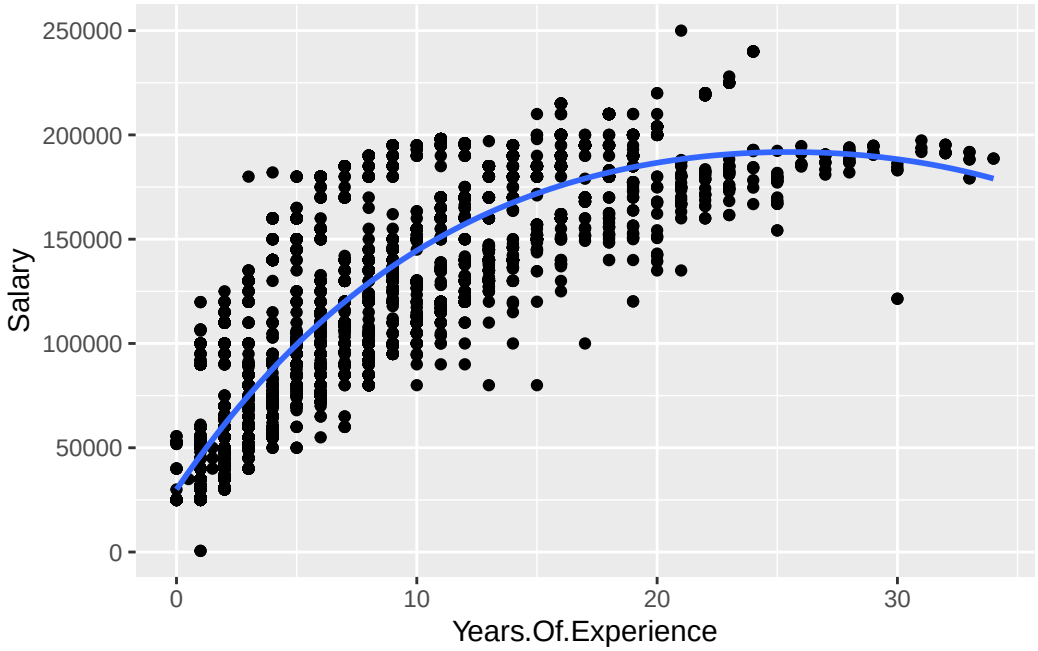
```r
library(ggplot2)
ggplot(data = filtered_data, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE)
```



```r
library(ggplot2)
ggplot(data = filtered_data, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE)
```




```r
library(ggplot2)
ggplot(data = filtered_data, aes(x = Years.Of.Experience, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 4), se = FALSE)
```



Aufgrund der Kurve wurde sich entschieden eine weitere Variable, sowie die Potenzen 2, 3 und 4 von ” Years of Experience” hinzuzufügen und anschließend je 3 unterschiedliche Regressionen durchzuführen. Anschließend wird dann geschaut, wie akkurat sich das Modell auf die Testdaten mit der jeweiligen Potenz verhält.

Da Age zwar einen hohen Korrelationswert hat aber auch mit den Years.Of.Experience einhergeht nehmen wir diesen Wert nicht, sondern stattdessen das Education.Level. Dies wird in den nächsten Absätzen behandelt.

7.2.2 7.2.2.: Korrelationen

Um die Korrelation von dem “Education Level” zu berechnen müssen die Werte erst von kategorisch zu nummerisch transformiert werden.

```
filtered_data$Education.Level <- as.numeric(as.character(filtered_data$Education.Level))

str(filtered_data$Education.Level)
```

```
num [1:6398] 2 1 2 2 1 2 2 2 1 1 ...
```

```r
correlations <- cor(filtered_data[c("Salary", "Age", "Years.Of.Experience", "Education.Lev

print(correlations)
```

```
                     Salary       Age Years.Of.Experience Education.Level
Salary             1.0000000 0.7291603           0.8103542       0.6374551
Age                0.7291603 1.0000000           0.9363709       0.5965278
Years.Of.Experience 0.8103542 0.9363709           1.0000000       0.6105127
Education.Level    0.6374551 0.5965278           0.6105127       1.0000000
```

Nun ist zu erkennen, dass zwischen dem Education Level und Salary eine Korrelation von 0.63 vorliegt. Zwischen dem Alter und den Bildungsniveau liegt dann eine Korrelation von 0.59. und dem " Years of Experience" eine Korrelation von 0.61.

Es ist also zu erkennen, dass Die Korrelationen zwischen dem Bildungsniveau und den anderen stets relativ hoch ist.

7.2.3 7.2.3.: Datenaufbereitung

Für die Regressionen sind nur die "Salary, Years of Experience, Education Level" relevant, also werden nur sie in den nen Datensatz geschrieben:

```r
filtered_data6 <- filtered_data %>%
  select(Salary, Years.Of.Experience, Education.Level)
```

Anschließend werden die "Years of Experience" potenziert und ausgegeben:

```r
filtered_data6 <- filtered_data6 %>%
  mutate(Years.Of.Experience_Squared = Years.Of.Experience^2) %>%
  mutate(Years.Of.Experience_Cubed = Years.Of.Experience^3) %>%
  mutate(Years.Of.Experience_Quartic = Years.Of.Experience^4)
```

```r
head(filtered_data6)
```

```
# A tibble: 6 x 6
  Salary Years.Of.Experience Education.Level Years.Of.Experience_Squared
   <dbl>               <dbl>           <dbl>                        <dbl>
1 110000                   5               2                           25
2  95000                   4               1                           16
3  90000                   3               2                            9
```

```
4  70000                  2              2                          4
5  60000                  1              1                          1
6  90000                  3              2                          9
# i 2 more variables: Years.Of.Experience_Cubed <dbl>,
#   Years.Of.Experience_Quartic <dbl>
```

Nun wird wieder eine Z-Skalierung von "filtered_data6" durchgeführt.

Hierzu wird zunächst eine Kopie des Datensatzes erstellt. Anschließend werden dann die "Years of Experience" und das "Education Level" z-skaliert.

```
# Erstellen Sie eine Kopie von filtered_data6
filtered_data6_z <- filtered_data6

# Z-Skalierung für Years.Of.Experience und Education.Level in filtered_data6_z durchführen
filtered_data6_z <- filtered_data6_z %>%
  mutate(
    Years.Of.Experience = scale(Years.Of.Experience),
    Education.Level = scale(Education.Level)
  )
```

Ergebnis der Z-Skalierung:

```
head(filtered_data6_z)
```

```
# A tibble: 6 x 6
  Salary Years.Of.Experience[,1] Education.Level[,1] Years.Of.Experience_Squared
   <dbl>                   <dbl>               <dbl>                        <dbl>
1 110000                 -0.517               0.411                           25
2  95000                 -0.684              -0.724                           16
3  90000                 -0.851               0.411                            9
4  70000                 -1.02                0.411                            4
5  60000                 -1.18               -0.724                            1
6  90000                 -0.851               0.411                            9
# i 2 more variables: Years.Of.Experience_Cubed <dbl>,
#   Years.Of.Experience_Quartic <dbl>
```

Nun wird die Standardabweichung der Berufserfahrung überprüft.

```
sd(filtered_data6_z$Years.Of.Experience)
```

```
[1] 1
```

```r
sd(filtered_data6_z$Education.Level)
```

```
[1] 1
```

Bei beiden liegt eine Abweichung von 1 vor.

Um eine lineare Regression durchzuführen wird der Datensatz filtered_data_z in Trainings- und Testdaten eingeteilt. Mithilfe der Funktion "set.seed(007)" initialisiert den Zufallsgenerator mit einer festen Zahl, um sicherzustellen, dass die Ergebnisse bei jedem Durchlauf reproduzierbar sind. "initial_split" teilt dann den Datensatz in Tranings- und Testdaten. Der Parameter strata = Years.Of.Experience sorgt dafür, dass die Daten nach dem Gehalts-Wert stratifiziert werden, um sicherzustellen, dass die Trainings- und Testdaten eine ähnliche Verteilung von Gehalts-Werten aufweisen. Da die Potenzen dieser Variable alle in der selben Reihe stehen muss hier nichts verändert werden. Die Funktion training() extrahiert die Trainingsdaten aus dem aufgeteilten Datensatz, während testing() die Testdaten extrahiert.

```r
set.seed(007)
filtered_data6_z <- initial_split(filtered_data6_z, prop = 0.8, strata = Years.Of.Experien

fd6_train <- training(filtered_data6_z)
fd6_test <- testing(filtered_data6_z)
```

7.2.4 7.2.3.: Modell Initialisieren

Nun muss das Modell initialisert werden.

```r
lm_model2 <- linear_reg() |> set_engine("lm")
```

Lineare Regression mit 2er Potenz:

```r
lm_fit2 <- lm_model2 |> fit(Salary ~ Years.Of.Experience + Education.Level + Years.Of.Expe
```

Lineare Regression mit 3er Potenz:

```r
lm_fit3 <- lm_model2 |> fit(Salary ~ Years.Of.Experience + Education.Level + Years.Of.Expe
```

Lineare Regression mit 3er Potenz:

```r
lm_fit4 <- lm_model2 |> fit(Salary ~ Years.Of.Experience + Education.Level + Years.Of.Expe
```

Zusammenfassung des Ergebnis (2er Potenz):

```
summary <- lm_fit2 |> extract_fit_engine() |> summary()
summary
```

```
Call:
stats::lm(formula = Salary ~ Years.Of.Experience + Education.Level +
    Years.Of.Experience_Squared, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-83843 -18771  -3740  10782  90478

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 143592.201    882.844  162.65   <2e-16 ***
Years.Of.Experience          73825.758   1234.509   59.80   <2e-16 ***
Education.Level               6527.658    493.036   13.24   <2e-16 ***
Years.Of.Experience_Squared   -266.380      7.883  -33.79   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26430 on 5113 degrees of freedom
Multiple R-squared:  0.7464,    Adjusted R-squared:  0.7463
F-statistic:  5017 on 3 and 5113 DF,  p-value: < 2.2e-16
```

Zusammenfassung des Ergebnis (3er Potenz):

```
# Zusammenfassung der Regression
summary <- lm_fit3 |> extract_fit_engine() |> summary()
summary
```

```
Call:
stats::lm(formula = Salary ~ Years.Of.Experience + Education.Level +
    Years.Of.Experience_Cubed, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-84267 -19790  -3860  11280  91008

Coefficients:
```

```
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  1.261e+05  4.865e+02  259.13   <2e-16 ***
Years.Of.Experience          5.507e+04  7.975e+02   69.05   <2e-16 ***
Education.Level              7.198e+03  4.977e+02   14.46   <2e-16 ***
Years.Of.Experience_Cubed -5.905e+00  1.911e-01  -30.90   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26830 on 5113 degrees of freedom
Multiple R-squared:  0.7386,    Adjusted R-squared:  0.7385
F-statistic:  4817 on 3 and 5113 DF,  p-value: < 2.2e-16
```

Zusammenfassung des Ergebnis (4er Potenz):

```r
summary <- lm_fit4 |> extract_fit_engine() |> summary()
summary
```

```
Call:
stats::lm(formula = Salary ~ Years.Of.Experience + Education.Level +
    Years.Of.Experience_Quartic, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-82732 -19598  -3836  11803  90306

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                   1.213e+05  4.176e+02  290.40   <2e-16 ***
Years.Of.Experience           4.801e+04  6.640e+02   72.31   <2e-16 ***
Education.Level               8.034e+03  5.007e+02   16.05   <2e-16 ***
Years.Of.Experience_Quartic -1.569e-01  5.637e-03  -27.83   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 27240 on 5113 degrees of freedom
Multiple R-squared:  0.7306,    Adjusted R-squared:  0.7305
F-statistic:  4623 on 3 and 5113 DF,  p-value: < 2.2e-16
```

Vorhersagen auf Trainings- und Testdatensatz (2er Potenz):

```
pred_train2 <- predict(lm_fit2, new_data = fd6_train) |> rename("pred_train2" = ".pred")
pred_test2 <- predict(lm_fit2, new_data = fd6_test) |>  rename("pred_test2" = ".pred")

compare_train2 <- fd5_train |>
  select(Salary) |>
  bind_cols(pred_train2)
head(compare_train2)
```

```
# A tibble: 6 x 2
  Salary pred_train2
   <dbl>       <dbl>
1  60000      51180.
2  90000      81077.
3  85000      73668.
4  55000      51180.
5  75000      70099.
6  60000      51180.
```

```
compare_test2 <- fd5_test |>
  select(Salary) |>
  bind_cols(pred_test2)
head(compare_test2)
```

```
# A tibble: 6 x 2
  Salary pred_test2
   <dbl>      <dbl>
1  90000     81077.
2  70000     70099.
3  60000     58589.
4 125000    110813.
5 130000    110813.
6  95000     84113.
```

Vorhersagen auf Trainings- und Testdatensatz (3er Potenz):

```
pred_train3 <- predict(lm_fit3, new_data = fd6_train) |> rename("pred_train3" = ".pred")
pred_test3 <- predict(lm_fit3, new_data = fd6_test) |>  rename("pred_test3" = ".pred")

compare_train3 <- fd6_train |>
  select(Salary) |>
```

```
  bind_cols(pred_train3)
head(compare_train3)
```

```
# A tibble: 6 x 2
  Salary pred_train3
   <dbl>       <dbl>
1  60000      55650.
2  90000      82029.
3  85000      73860.
4  55000      55650.
5  75000      72960.
6  60000      55650.
```

```
compare_test3 <- fd6_test |>
  select(Salary) |>
  bind_cols(pred_test3)
head(compare_test3)
```

```
# A tibble: 6 x 2
  Salary pred_test3
   <dbl>      <dbl>
1  90000     82029.
2  70000     72960.
3  60000     63819.
4 125000    108458.
5 130000    108458.
6  95000     82823.
```

Vorhersagen auf Trainings- und Testdatensatz (4er Potenz):

```
pred_train4 <- predict(lm_fit4, new_data = fd6_train) |> rename("pred_train4" = ".pred")
pred_test4 <- predict(lm_fit4, new_data = fd6_test) |>  rename("pred_test4" = ".pred")

compare_train4 <- fd6_train |>
  select(Salary) |>
  bind_cols(pred_train4)
head(compare_train4)
```

```
# A tibble: 6 x 2
```

```
  Salary pred_train4
   <dbl>       <dbl>
1  60000      58600.
2  90000      83717.
3  85000      74598.
4  55000      58600.
5  75000      75721.
6  60000      58600.
```

```
compare_test4 <- fd6_test |>
  select(Salary) |>
  bind_cols(pred_test4)
head(compare_test4)
```

```
# A tibble: 6 x 2
  Salary pred_test4
   <dbl>      <dbl>
1  90000     83717.
2  70000     75721.
3  60000     67719.
4 125000    107542.
5 130000    107542.
6  95000     82576.
```

7.2.5 7.2.4.: Grafische Darstellung

In dem folgenden Absatz werden die unterschiedlichen Potenzen mithilfe eines Vorhersage Plots grafisch dargestellt.

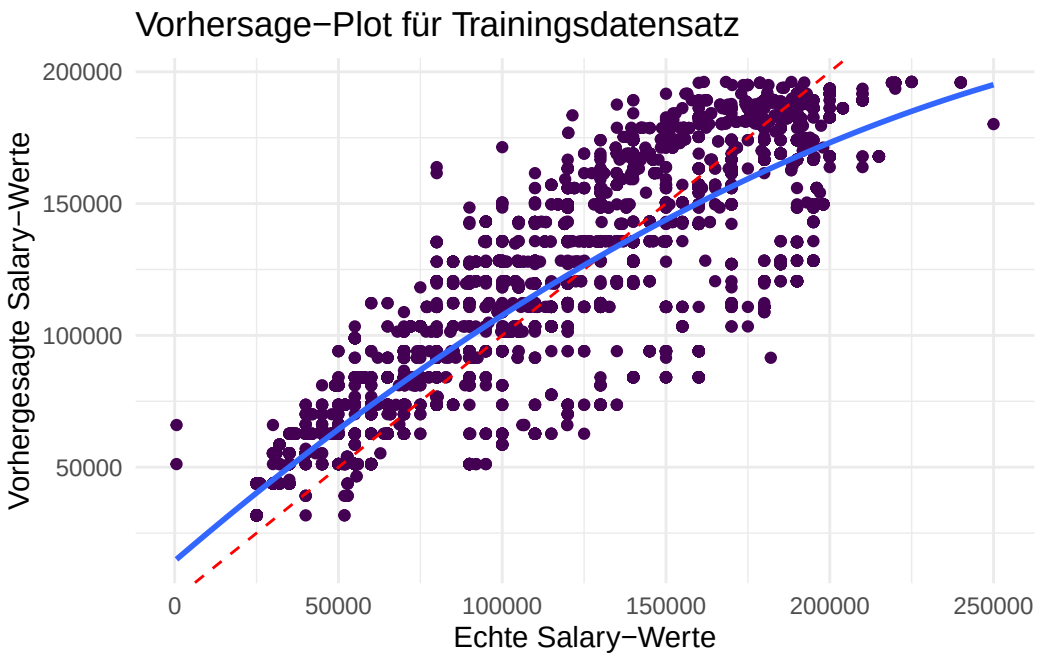
Hierzu wurde eine zusätzliche Quelle verwendet: (https://statologie.de/vorhergesagte-werte-plotten-r/)

Die folgenden Codechunks wird immer der gleiche Aufbau verwendet. Bloß stets mit den unterschiedlichen Potenzen. Zunächst wird der Graph für den Trainingdatensatz und anschließend für den Testdtaensatz erstellt.

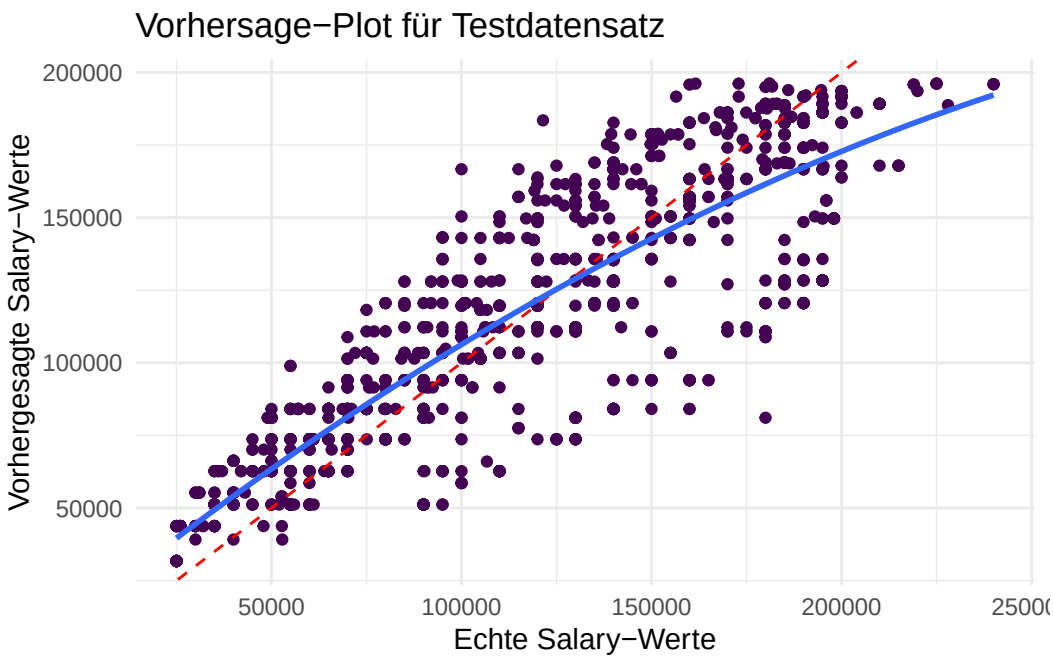
2er Potenz:

```
ggplot(compare_train2, aes(x = Salary, y = pred_train2)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE) +
```

```
  labs(title = "Vorhersage-Plot für Trainingsdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```

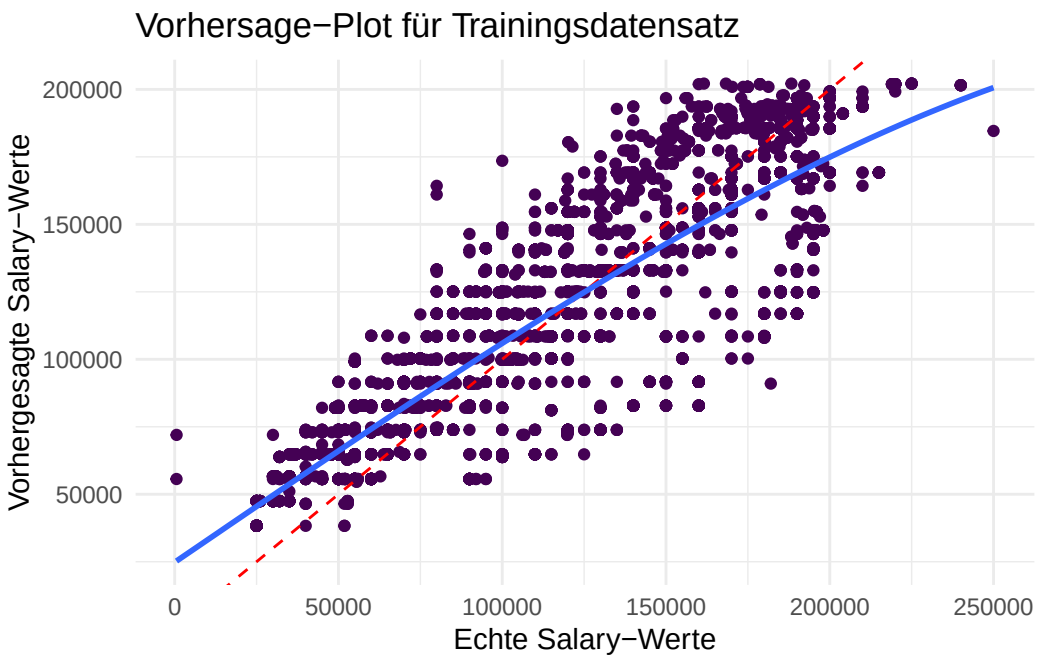



```
ggplot(compare_test2, aes(x = Salary, y = pred_test2)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE) +
  labs(title = "Vorhersage-Plot für Testdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```

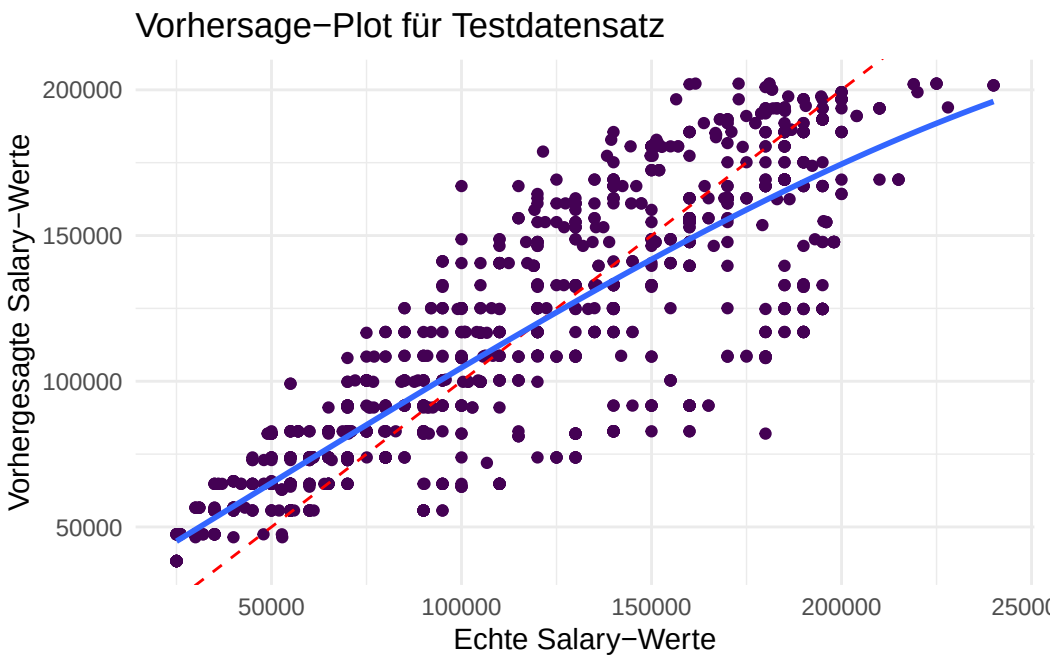


3er Potenz:

```
ggplot(compare_train3, aes(x = Salary, y = pred_train3)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE) +
  labs(title = "Vorhersage-Plot für Trainingsdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```

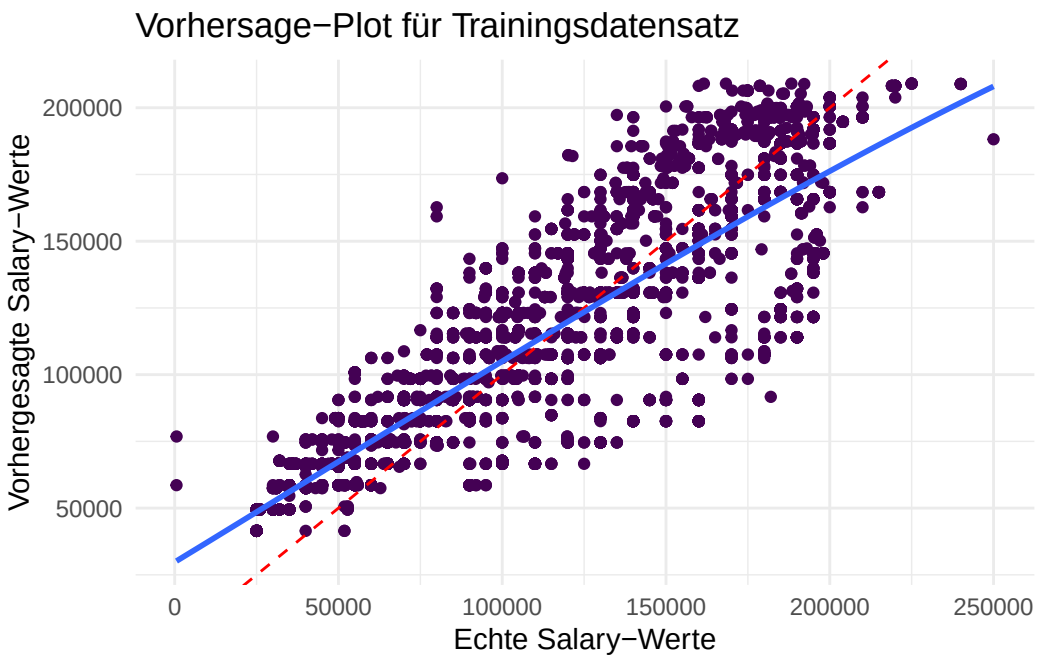



```
ggplot(compare_test3, aes(x = Salary, y = pred_test3)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE) +
  labs(title = "Vorhersage-Plot für Testdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```

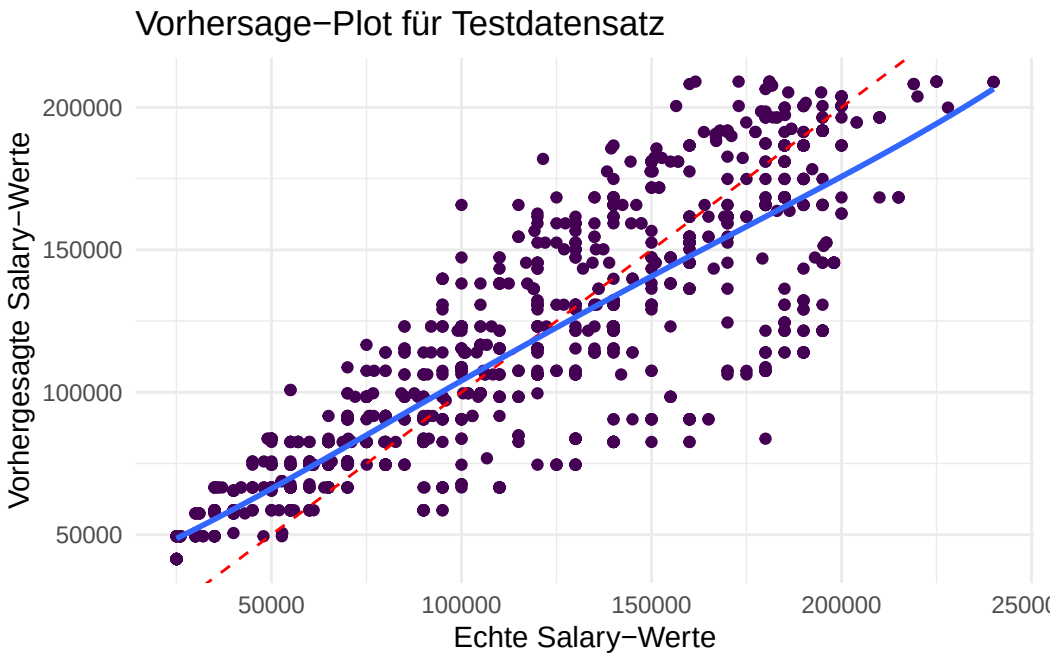


4er Potenz:

```
ggplot(compare_train4, aes(x = Salary, y = pred_train4)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE) +
  labs(title = "Vorhersage-Plot für Trainingsdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```




```
ggplot(compare_test4, aes(x = Salary, y = pred_test4)) +
  geom_point(color = viridis(0.50)) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 4), se = FALSE) +
  labs(title = "Vorhersage-Plot für Testdatensatz",
       x = "Echte Salary-Werte",
       y = "Vorhergesagte Salary-Werte") +
  theme_minimal()
```



Es ist bereits eine Tendenz zu erkennen. Die 4er Potenz der Berufserfahrung auf einer minimal genaueren Prognose bei den Testdaten, im Vergleich zu der 2er und 3er Potenz, resultiert.

7.2.6 7.2.5.: Fehler

Auch hier werden die Potenzen einzeln und hintereinander mit dem “rmse” untersucht.

2er Potenz

Trainingsfehler:

```
rmse(compare_train2, Salary, pred_train2)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      26417.
```

Testfehler:

```
rmse(compare_test2, Salary, pred_test2)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      26629.
```

Es geht hervor, dass das Modell auf den Trainingsdaten besser perfomt als auf den Testdaten.

Diese Tatsache weist auf Overfitting hin.

3er Potenz:

Trainingsfehler:

```
rmse(compare_train3, Salary, pred_train3)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      26820.
```

Testfehler:

```
rmse(compare_test3, Salary, pred_test3)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      27035.
```

Auch hier performt das Modell besser auf den Trainingsdaten. Dies weist ebenfalls auf Overfitting hin.

4er Potenz:

Trainingsfehler:

```
rmse(compare_test4, Salary, pred_test4)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      27457.
```

Testfehler:

```
rmse(compare_test4, Salary, pred_test4)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      27457.
```

Auch wenn alle Modelle nur geringe Unterschiede aufweisen, performt dieses auf den Trainingsdaten gleich wie auf den Testdaten. Demnach ist es das präziseste Modell und wird als einziges weiter verwendet und weiter verwendet.

Dies ließ sich bereits unter 7.2.1 erkennen, da das 3. Polynom das Geom Smooth grafisch die beste Linie abgebildet hat.

Zur Einordnung der Fehler die Verteilung von Gehalt ansehen:

Verteilung von Gehalt:

```
describe(filtered_data5, Salary)
```

```
variable = Salary
type     = double
na       = 0 of 6 398 (0%)
unique   = 432
min|max  = 550 | 250 000
q05|q95  = 35 000 | 195 000
q25|q75  = 70 000 | 160 000
median   = 120 000
mean     = 116 787.2
```

7.2.7 7.2.6.: Residuen

Der Aufbau, sowie die Definiton zu den Risiduen wurden bereits unter 7.1.6 aufgeführt.

Deswegen werden die Funktionen ebenfalls nicht erneut erläutert.

Es wird sich jediglich nur um die Ausgabe gekümmert, und anschließend beschrieben.

```
residuals_train2 <- augment(lm_fit4, new_data = fd6_train) %>% select(.resid)
```

```r
residuals_test2 <- augment(lm_fit4, new_data = fd6_test) %>% select(.resid)

print(residuals_train2)
```

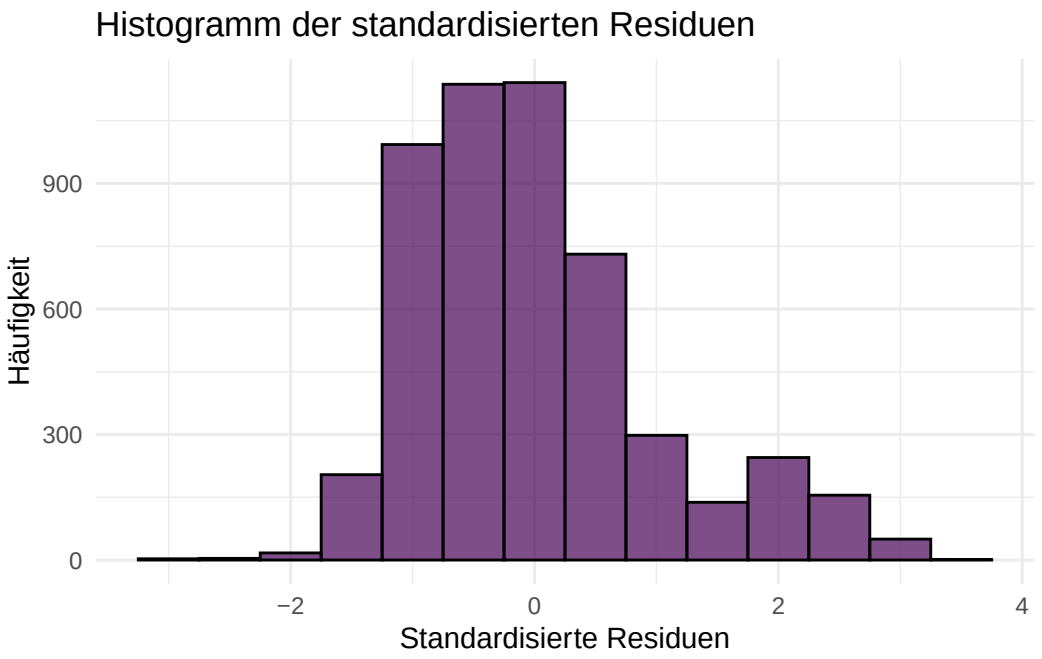
```
# A tibble: 5,117 x 1
   .resid
    <dbl>
 1  1400.
 2  6283.
 3 10402.
 4 -3600.
 5  -721.
 6  1400.
 7 -3600.
 8  6283.
 9 -5721.
10  1400.
# i 5,107 more rows
```

```r
print(residuals_test2)
```

```
# A tibble: 1,281 x 1
   .resid
    <dbl>
 1  6283.
 2 -5721.
 3 -7719.
 4 17458.
 5 22458.
 6 12424.
 7 -5721.
 8 22458.
 9  6283.
10 -5721.
# i 1,271 more rows
```

Histogramm der standardisierten Residuen der Trainingsdaten:

```r
residuals_train2$standardized_resid2 <- scale(residuals_train2$.resid)

ggplot(data = residuals_train2, aes(x = standardized_resid2)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
       x = "Standardisierte Residuen",
       y = "Häufigkeit") +
  scale_fill_viridis() +
  theme_minimal()
```

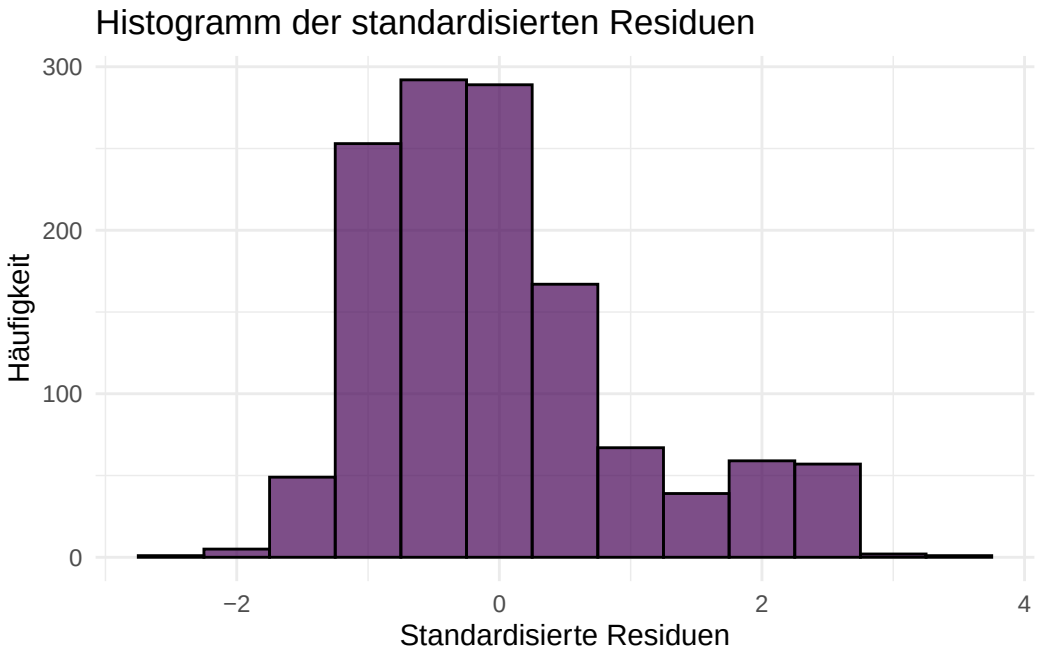


Zu erkennen ist, dass die Werte zwischen 2 und -2 am häufigsten vertreten sind.

Histogramm der standardisierten Residuen der Testdaten:

```r
# Standardisierung/Z-Skalierung der Residuen der Residuen
residuals_test2$standardized_resid2 <- scale(residuals_test2$.resid)

# Histogramm der standardisierten Residuen mit Viridis-Farbschema
ggplot(data = residuals_test2, aes(x = standardized_resid2)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
```

```
      x = "Standardisierte Residuen",
      y = "Häufigkeit") +
scale_fill_viridis() +  # Fügt das Viridis-Farbschema hinzu
theme_minimal()
```



Auch hier sind die Risiduen zwischen 2 und -2 am häufigsten.

Aus den beiden Histogrammen geht also hervor, dass die Abweichungen zwischen den vorhergesagten Werten und tatsächlichen Werten nicht wirklich groß sind.

7.2.8 7.2.7.: Q-Q-Plot

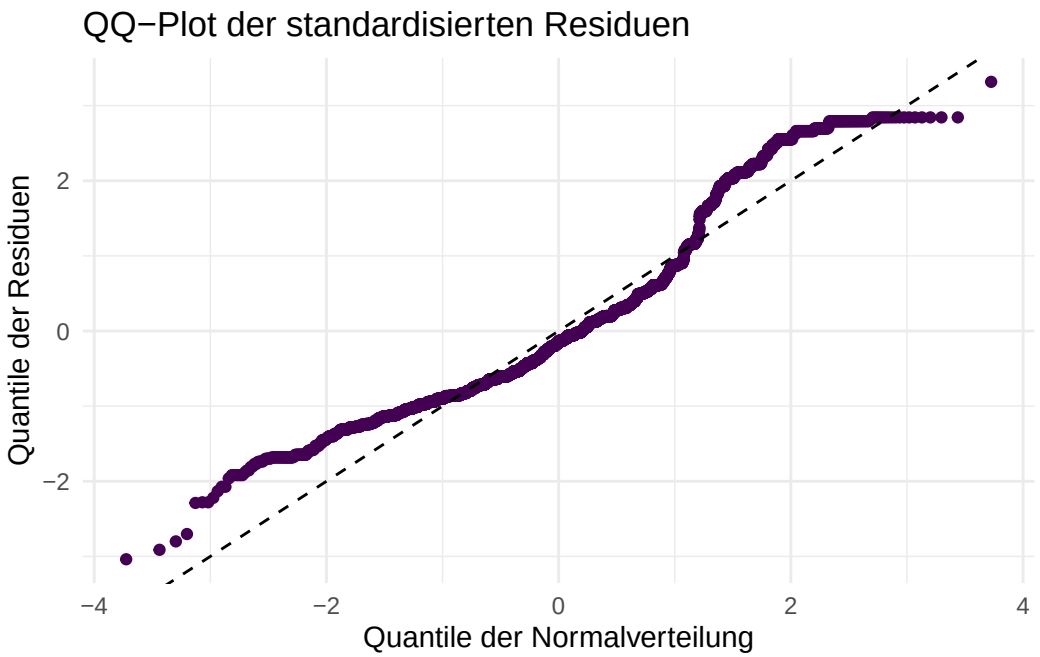
Wie auch bei den Risiduen bleibt hier der Aufbau gleich wie in 7.1.7.

Lediglich die Aussagen sind erneut anders.

Trainingsdaten:

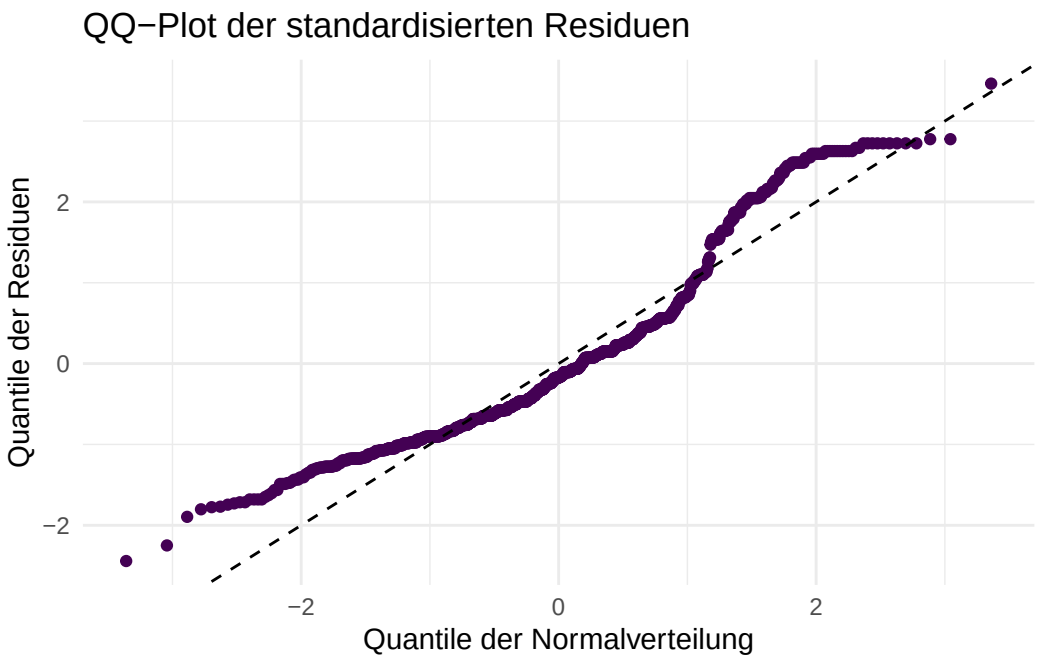
```
ggplot(data = residuals_train2, aes(sample = standardized_resid2)) +
  stat_qq(distribution = qnorm, dparams = list(mean = 0, sd = 1), color = viridis(1)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black") +
  labs(title = "QQ-Plot der standardisierten Residuen",
```

```
      x = "Quantile der Normalverteilung",
      y = "Quantile der Residuen") +
  scale_color_viridis() +
  theme_minimal()
```



Testdaten

```
ggplot(data = residuals_test2, aes(sample = standardized_resid2)) +
  stat_qq(distribution = qnorm, dparams = list(mean = 0, sd = 1), color = viridis(1)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black") +
  labs(title = "QQ-Plot der standardisierten Residuen",
       x = "Quantile der Normalverteilung",
       y = "Quantile der Residuen") +
  scale_color_viridis() +
  theme_minimal()
```



Der Plot bildet die Ausreißer vom Gehalt nach oben und unten ab, welche wir im ersten Diagramm von 7.2.1 sehen, die Residuen sind nicht perfekt normalverteilt. Eventuell gibt es hier Datenpunkte, welche die Residuen beeinflussen. Aufgrund der explorativen Datenanalyse wissen wir, dass Jobs mit hoher Arbeitserfahrung oft Führungsverantwortung beinhalten, welche nochmal zusätzlich monetär honoriert wird. Die Flache Linie durch die Gerade oben durch deutet auf ein Gehaltscap hin. Die Ausreißer am unteren Ende lassen sich durch Einstiegsjobs, so wie Niedriglohnjobs ohne Hochschulabchlüsse erklären. (Ausbildungsberufe werden hier nicht berücksichtigt).

Der Testfehler in der einfachen Linearen Regression betrug 30763, während der in der mehrfachen Linearen Regression mit der 4er Potenz von Arbeitserfahrung 27456 ergibt, welches eine verbesserung der Vorhersagen von 10% gegenüber der einfach Linearen regression aus 7.1 ergibt. Des Weiteren gibt es hier auch kein Overfitting des Modells mehr, da die Test und Trainingsfehler exakt die selben sind (27456) .

7.3 7.3.: Entscheidungsbaum

Mithilfe dieses Entscheidungsproblems soll eine Vorhersage, des Erreichens einer Managementsposition anhand von Faktoren mit der größten Korrelation, getroffen werden.

Hierzu werden die Software Engineering Jobs und deren Führungsposition bertrachtet. Die Führungsposition heißt “Software Engineering Manager”.

7.3.1 7.3.1.: Vorbereitung

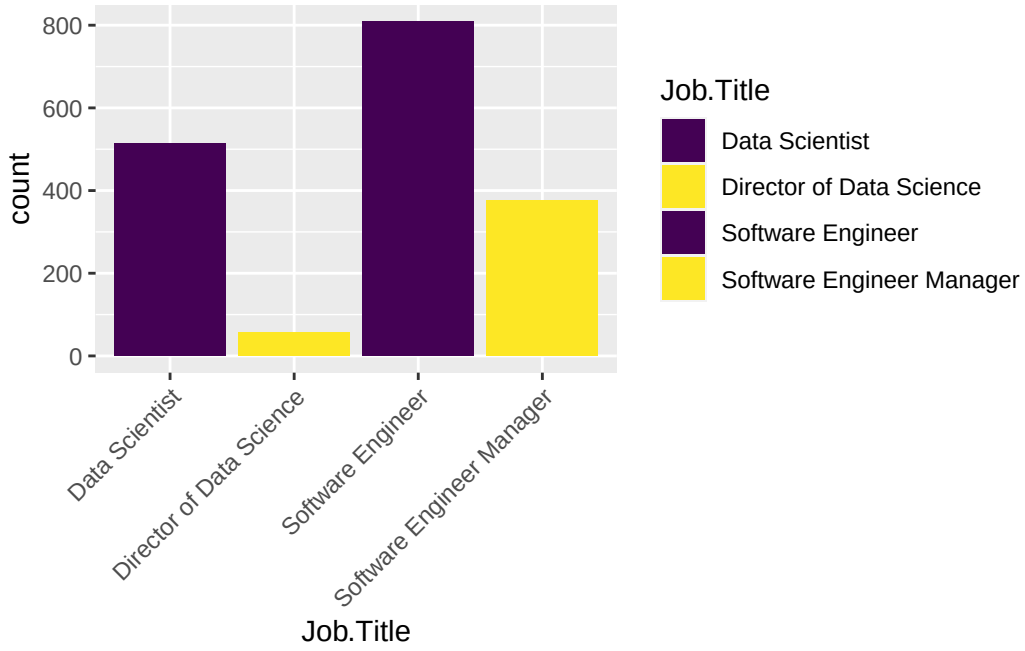
Zunächst werden beide Jobs nach dem Kriterium einer Häufigkeitsverteilung von N>30 ausgewählt.

Die Gründe sind in 5.1 aufgezählt.

Hierzu wird zunächst einen Datenrahmen erstellt, der die Job-Titel und ihre Häufigkeit enthält.

Anschließend wird dann ein Balkendiagramm erstellt.

```
library(ggplot2)
library(viridis)

jobs_count <- filtered_data %>%
  filter(Job.Title %in% c("Software Engineer", "Software Engineer Manager", "Data Scientis
  group_by(Job.Title) %>%
  summarize(count = n())

ggplot(jobs_count, aes(x = Job.Title, y = count, fill = Job.Title)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c(viridis(2), viridis(2))) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Hier ist zu erkennen, dass es deutlich weniger Director und Manager gibt.

Nun wird noch einmal, das Durchschnittsgehalt der verschiedenen Jobs berechnet.

```
filtered_data %>%
  filter(Job.Title %in% c("Software Engineer", "Software Engineer Manager", "Data Scientis
  group_by(Job.Title) %>%
  summarise(Avg_Salary = mean(Salary, na.rm = TRUE))
```

```
# A tibble: 4 x 2
  Job.Title                 Avg_Salary
  <chr>                          <dbl>
1 Data Scientist                164099.
2 Director of Data Science      204561.
3 Software Engineer             120541.
4 Software Engineer Manager     172502.
```

7.3.2 7.3.2.: Datenaufbereitung

Vorab ist es wichtig zu erwähnen, dass keine Z-Skalierung der Daten durchgeführt wird. Dies liegt daran, da die Diagramme sonst zu unübersichtlich werden.

Zunächst werden alle benötigten Jobtitel aus dem Datenrahmen rausgefiltert.

```
filtered_data7 <- filtered_data %>%
  filter(Job.Title %in% c("Software Engineer", "Software Engineer Manager", "Data Scientis
```

Anschließend werden den Managern und Director der Wert 1 zugewiesen.

```
filtered_data7 <- filtered_data7 %>%
  mutate(Manager = ifelse(Job.Title %in% c("Director of Data Science", "Software Engineer
```

Aufteilung in Test- & Trainingsdaten:

```
set.seed(007)
filtered_data7 <- initial_split(filtered_data7, prop = 0.8, strata = Years.Of.Experience)

fd7_train <- training(filtered_data7)
fd7_test <- testing(filtered_data7)
```

7.3.3 7.3.3.: Modell Initialisieren

Initialisierung:

```
tree_mod <- decision_tree(mode = "regression")
```

Modell trainieren:

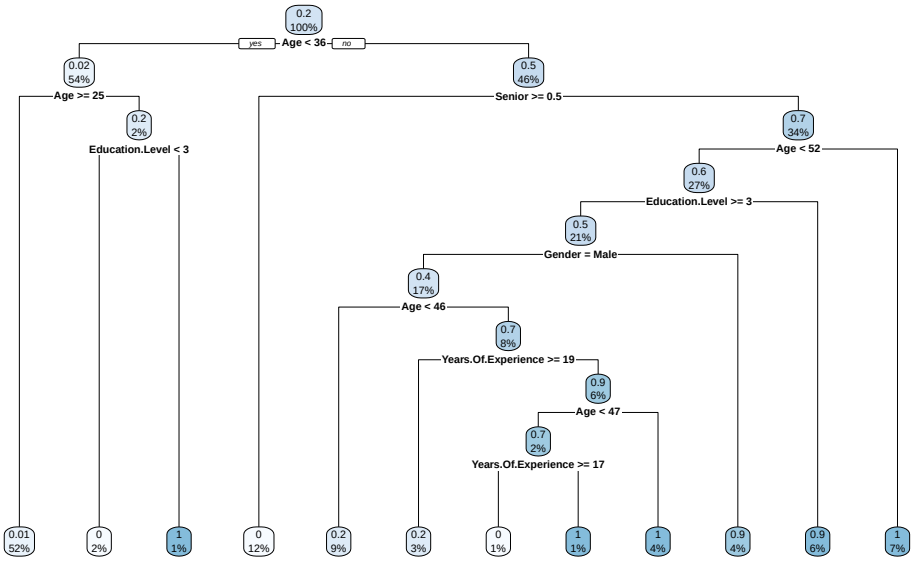
```
tree_fit <- tree_mod %>%
  fit(Manager ~ Age + Gender + Education.Level + Years.Of.Experience + Senior + Expat + Co
```

7.3.4 7.3.4.: Grafische Darstellung

Vorab eine grafische Darstellung mit den absoluten Zahlen:

```
tree_fit |>
  extract_fit_engine() |>
  rpart.plot(digits = 1)
```

```
Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and
To silence this warning:
    Call rpart.plot with roundint=FALSE,
    or rebuild the rpart model with model=TRUE.
```



Mit Beschriftungen (Als Klassifizierung umgesetzt/ In Klassen umgewandelt).

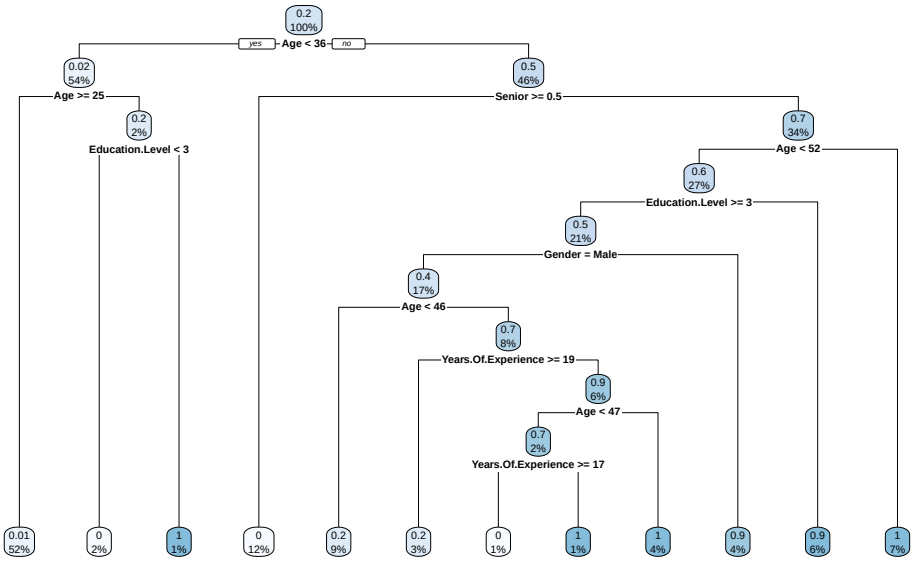
Nun wird das Ganze als Klassifikation umgesetzt. Hierbei wird in Klassen umgewandelt.

Hierzu wird zunächst eine Kopie des Trainingsatzes erstellt. Anschließend wird die Konvertierung von “Education Level” und “Manager” umgekehrt. Folgend wird dann das Baumodell erstellt, gefittet und zum Schluss visualisiert.

```
fd7_train2 <- fd7_train

fd7_train2$Education.Level <- factor(fd7_train2$Education.Level, levels = c(0, 1, 2, 3), l

fd7_train2$Manager <- factor(fd7_train2$Manager, levels = c(0, 1), labels = c("Employee",

fd7_train2$Senior <- factor(fd7_train2$Senior, levels = c(0, 1), labels = c("Junior", "Sen

tree_mod2 <- decision_tree(mode = "classification")

tree_fit2 <- tree_mod2 %>%
  fit(as.factor(Manager) ~ Age + Gender + Education.Level + Years.Of.Experience + Senior +

tree_fit |>
  extract_fit_engine() |>
```

```
rpart.plot(digits = 1)
```

```
Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and
To silence this warning:
    Call rpart.plot with roundint=FALSE,
    or rebuild the rpart model with model=TRUE.
```



Nach Ansehen des Entscheidungsbaumes, scheint es als hätten “Expat”, sowie “Country” eine schwache Vorhersagekraft und mangelnde Bedeutung auf den Wert Manager. Deswegen tauchen sie nicht im Entscheidungsbaum auf.

Es ist durchaus möglich, einen Entscheidungsbaum-Algorithmus zur Regression auf eine Variable anzuwenden, die lediglich Werte von 0 und 1 annimmt. In diesem Szenario würde der Baum versuchen, anhand der gegebenen Eingabevariablen (Features) eine Regression durchzuführen, um die kontinuierliche numerische Variable vorherzusagen.

Es gibt jedoch einige wichtige Aspekte zu beachten:

1. **Ausgabe als Kategorie interpretieren:** Wenn ihre Zielvariable tatsächlich binär ist und sie eine Klassifizierung (0 oder 1) vornehmen wollen, dann sollten Sie einen Klassifikationsalgorithmus verwenden, nicht eine Regressionsmethode. Ein Entscheidungsbaum für die Klassifikation würde besser geeignet sein, um Vorhersagen in Form von Kategorien (hier: 0 oder 1) zu treffen.

2. **Interpretation der Vorhersagen:** Wenn sie den Entscheidungsbaum für die Regression auf eine binäre Variable anwenden, werden die Vorhersagen des Modells als kontinuierliche Werte zwischen 0 und 1 liegen. Diese Vorhersagen können dann als Wahrscheinlichkeiten interpretiert werden, z.B. als die Wahrscheinlichkeit, dass eine bestimmte Beobachtung den Wert 1 annimmt. Um Klassen vorherzusagen, müssten Sie normalerweise einen Schwellenwert festlegen (zum Beispiel 0,5), um diese Wahrscheinlichkeiten in Klassen umzuwandeln.

Vorhersage auf Test- und Trainingsdaten:

```
head(fd7_train)
```

```
# A tibble: 6 x 16
  Job.Title    job_count   Age Gender Education.Level Years.Of.Experience Salary
  <chr>            <int> <dbl> <chr>            <dbl>                <dbl>  <dbl>
1 Data Scient~       515    29 Male                 2                    3  75000
2 Data Scient~       515    26 Female               2                    1.5  45000
3 Data Scient~       515    30 Male                 3                    5  180000
4 Data Scient~       515    30 Female               3                    5  180000
5 Data Scient~       515    27 Male                 3                    2  115000
6 Data Scient~       515    30 Female               3                    5  180000
# i 9 more variables: Country <chr>, Race <chr>, Senior <dbl>, SalaryKat <fct>,
#   ID <int>, job_type <dbl>, Expat <dbl>, BIP_Per_Person <dbl>, Manager <dbl>
```

```
pred_train7 <- predict(tree_fit, new_data = fd7_train) |> rename("pred_train" = ".pred")
pred_test7 <- predict(tree_fit, new_data = fd7_test) |>  rename("pred_test" = ".pred")

compare_train7 <- fd7_train |>
  select(Manager) |>
  bind_cols(pred_train7)
head(compare_train7)
```

```
# A tibble: 6 x 2
  Manager pred_train
    <dbl>      <dbl>
1       0     0.0110
2       0     0.0110
3       0     0.0110
4       0     0.0110
5       0     0.0110
6       0     0.0110
```

```r
compare_test7 <- fd7_test |>
  select(Manager) |>
  bind_cols(pred_test7)
head(compare_test7)
```

```
# A tibble: 6 x 2
  Manager pred_test
    <dbl>     <dbl>
1       0    0
2       0    0.0110
3       0    0.0110
4       0    0.0110
5       0    0.0110
6       0    0.0110
```

7.3.5 7.3.5.: Fehler

Trainings- und Testfehler:

```r
library(yardstick)

rmse_train <- compare_train7 %>%
  mutate(Manager = as.numeric(Manager)) %>%
  yardstick::rmse(truth = Manager, estimate = pred_train)

rmse_test <- compare_test7 %>%
  mutate(Manager = as.numeric(Manager)) %>%
  yardstick::rmse(truth = Manager, estimate = pred_test)

print(rmse_train)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       0.179
```

```r
print(rmse_test)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       0.198
```

Der Trainingsfehler ist geringer als der Testfehler. Dies weist darauf hin, dass das Modell leicht overfittet ist.

7.3.6 7.3.6.: Residuen

Trotz der binären Beschaffenheit des abhängigen Merkmals ('Manager') ist es möglich, ein Entscheidungsbaummodell für die Prognose von Wahrscheinlichkeiten zu nutzen. Das Modell wird Vorhersagen über die Wahrscheinlichkeiten des Eintretens oder Nicht-Eintretens des Ereignisses (hier: Manager oder nicht Manager) treffen.

Die Residuen in einem solchen Kontext repräsentieren die Differenz zwischen den tatsächlich beobachteten Werten (0 und 1) und den vorhergesagten Wahrscheinlichkeiten. Sie bieten somit eine Möglichkeit zur Überprüfung, wie gut das Modell die Beobachtungen anpasst. Auch wenn sie nicht unmittelbar die Vorhersage von 0 oder 1 anzeigen, spiegeln sie dennoch wider, wie gut das Modell die Wahrscheinlichkeiten einschätzt, dass ein bestimmtes Ereignis eintritt oder nicht.

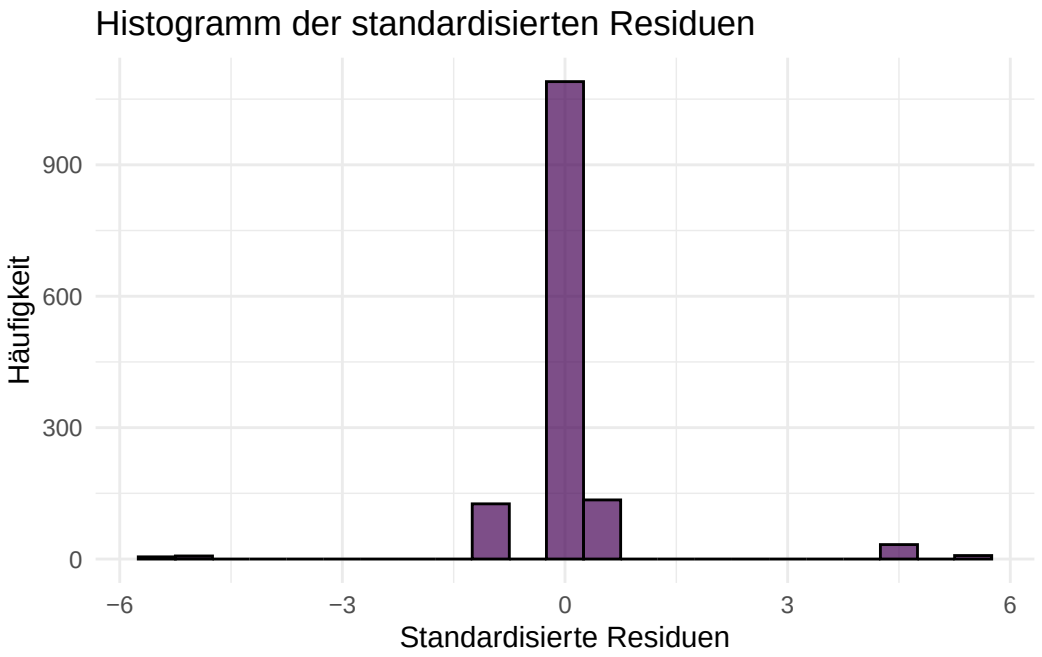
```r
residuals_train7 <- augment(tree_fit, new_data = fd7_train) %>% select(.resid)

residuals_test7 <- augment(tree_fit, new_data = fd7_test) %>% select(.resid)

print(residuals_train7)
```

```
# A tibble: 1,404 x 1
    .resid
     <dbl>
 1 -0.0110
 2 -0.0110
 3 -0.0110
 4 -0.0110
 5 -0.0110
 6 -0.0110
 7 -0.0110
 8 -0.0110
 9 -0.0110
10 -0.0110
# i 1,394 more rows
```

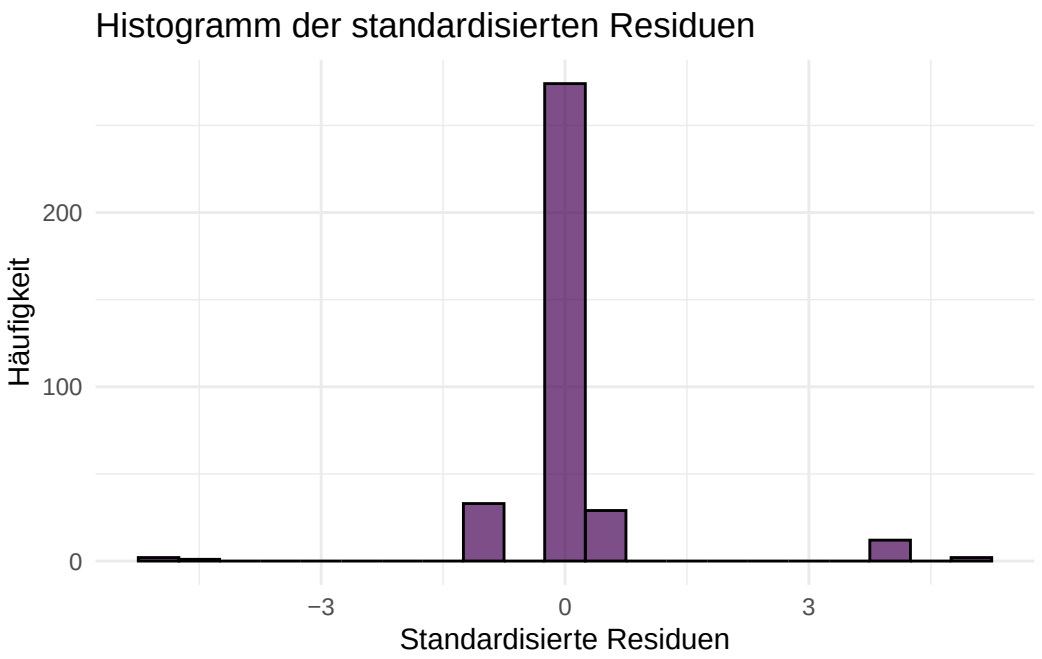
```r
print(residuals_test7)
```

```
# A tibble: 353 x 1
    .resid
     <dbl>
 1  0
 2 -0.0110
 3 -0.0110
 4 -0.0110
 5 -0.0110
 6 -0.0110
 7 -0.0110
 8 -0.0110
 9 -0.0110
10 -0.0110
# i 343 more rows
```

```r
residuals_train7$standardized_resid7 <- scale(residuals_train7$.resid)

ggplot(data = residuals_train7, aes(x = standardized_resid7)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
       x = "Standardisierte Residuen",
       y = "Häufigkeit") +
  scale_fill_viridis() +
  theme_minimal()
```




```
residuals_test7$standardized_resid7 <- scale(residuals_test7$.resid)

ggplot(data = residuals_test7, aes(x = standardized_resid7)) +
  geom_histogram(binwidth = 0.5, fill = viridis(1), color = "black", alpha = 0.7) +
  labs(title = "Histogramm der standardisierten Residuen",
       x = "Standardisierte Residuen",
       y = "Häufigkeit") +
  scale_fill_viridis() +
  theme_minimal()
```



7.3.7 7.3.7.: Q-Q-Plot / Accuracy test

Für ein Modell, das binäre Vorhersagen trifft (wie in diesem Fall mit einem Baummodell, das eine binäre Zielvariable vorhersagt), könnte ein QQ-Plot in Bezug auf die Residuen weniger sinnvoll sein. Der QQ-Plot wird üblicherweise verwendet, um die Normalität der Residuen in einem Modell zu überprüfen. Da bei binären Vorhersagen die Residuen nicht unbedingt einer Normalverteilung folgen müssen, kann ein QQ-Plot möglicherweise weniger aufschlussreich sein.

Insbesondere bei Klassifikationsmodellen, die binäre Kategorien vorhersagen, sind die Residuen möglicherweise nicht so interpretierbar wie bei Modellen mit kontinuierlichen Variablen.

Die Residuen bei binären Modellen können diskrete Werte oder eine Art von Klassifizierungsfehler (beispielsweise für binäre Klassifikationsmodelle) widerspiegeln. Ein QQ-Plot ist normalerweise dann sinnvoll, wenn die Residuen einer Normalverteilung folgen sollten, was jedoch bei binären Vorhersagen nicht zwangsläufig der Fall ist.

In solchen Fällen ist es oft wichtiger, andere Metriken wie Genauigkeit, Precision, Recall, ROC-Kurven oder AUC zu verwenden, um die Leistung des Modells zu bewerten und zu verstehen, wie gut es die binären Vorhersagen macht.

Siehe folgende links:

24 Evaluation Metrics for Binary Classification (And When to Use Them) (neptune.ai)

https://topepo.github.io/caret/measuring-performance.html

Nun werden zunächst die Faktoren konvertiert und Levels gesetzt. Anschließend wird die Accuracy für die Tranings-, sowie für die Testdaten berechnet.

```
library(caret)
```

```
Warning: Paket 'caret' wurde unter R Version 4.3.2 erstellt
```

```
Lade nötiges Paket: lattice
```

```
Warning: Paket 'lattice' wurde unter R Version 4.3.2 erstellt
```

```
Attache Paket: 'caret'
```

```
Die folgenden Objekte sind maskiert von 'package:yardstick':

    precision, recall, sensitivity, specificity
```

```
Das folgende Objekt ist maskiert 'package:purrr':

    lift
```

```
compare_train7$Manager <- factor(compare_train7$Manager)
compare_train7$pred_train <- factor(compare_train7$pred_train, levels = levels(compare_tra

compare_test7$Manager <- factor(compare_test7$Manager)
compare_test7$pred_test <- factor(compare_test7$pred_test, levels = levels(compare_test7$M

accuracy_train <- confusionMatrix(compare_train7$Manager, compare_train7$pred_train)
print(accuracy_train)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 199   0
         1   0 174
```

```
               Accuracy : 1
                 95% CI : (0.9902, 1)
    No Information Rate : 0.5335
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5335
         Detection Rate : 0.5335
   Detection Prevalence : 0.5335
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

```r
compare_test7$pred_test <- factor(compare_test7$pred_test, levels = levels(compare_train7$

accuracy_test <- confusionMatrix(compare_test7$Manager, compare_test7$pred_test)
print(accuracy_test)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 60  0
         1  0 40

               Accuracy : 1
                 95% CI : (0.9638, 1)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA
```

```
            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.6
         Detection Rate : 0.6
   Detection Prevalence : 0.6
      Balanced Accuracy : 1.0

       'Positive' Class : 0
```

```
# library(caret)
# compare_train7$pred_train <- factor(compare_train7$pred_train, levels = levels(compare_t
#
# accuracy_train <- confusionMatrix(compare_train7$Manager, compare_train7$pred_train)
# print(accuracy_train)
#
# compare_test7$pred_test <- factor(compare_test7$pred_test, levels = levels(compare_test7
#
# accuracy_test <- confusionMatrix(compare_test7$Manager, compare_test7$pred_test)
# print(accuracy_test)
```

Zusatz: Zuvor wurde der obrige Codechunk verwendet. Zum 7.12.2023 hat dieser Problemlos funktioniert. Zum 11.12.2023 hat dieser trotz identischen Codes (Gesamte Datei unverändert laut Github) nicht mehr funktioniert. Evtl. wurde die Library Caret in seiner Funktion geupdated (Und vorher eine veraltete Version genutzt) (Quelle 1). Es musste zusätzlich der Factor des "wahren" Werts von Manager auf sich selbst angepasst werden. Also von Char zu Faktor. Wobei ein Faktor hier den Zustand Manager abbildet mit dem Level = 1 = Manager (Quelle 2).

Quelle 1: https://stackoverflow.com/questions/51548908/error-data-and-reference-should-be-factors-with-the-same-levels

Quelle 2: https://bjoernwalther.com/variablen-in-r-als-faktor-definieren/

Die Confusion Matrix und die Statistiken werden verwendet, um die Leistung eines Klassifikationsmodells zu bewerten. In diesem Fall handelt es sich um eine binäre Klassifikation mit den Klassen 0 und 1.

Für die Confusion Matrix:

7.3.7.1 Trainingsdaten:

- Es gibt insgesamt 373 Beobachtungen in den Trainingsdaten.
- Von diesen wurden 199 korrekt als Klasse 0 und 174 korrekt als Klasse 1 vorhergesagt.
- Es gab keine falsch vorhergesagten Werte für Klasse 0 oder Klasse 1.
- Die Genauigkeit (Accuracy) beträgt 100%, was bedeutet, dass alle Vorhersagen korrekt waren.
- Sensitivität, Spezifität, Positiver und Negativer Vorhersagewert sind alle 1, was bedeutet, dass alle Metriken perfekt sind.

7.3.7.2 Testdaten:

- Es gibt insgesamt 100 Beobachtungen in den Testdaten.
- Von diesen wurden 60 korrekt als Klasse 0 und 40 korrekt als Klasse 1 vorhergesagt.
- Es gab keine falsch vorhergesagten Werte für Klasse 0 oder Klasse 1.
- Auch hier beträgt die Genauigkeit 100%, was bedeutet, dass alle Vorhersagen korrekt waren.
- Sensitivität, Spezifität, Positiver und Negativer Vorhersagewert sind alle 1, was auf perfekte Metriken hinweist.

In beiden Trainings- und Testdaten zeigt das Modell eine perfekte Vorhersagegenauigkeit und erzielt in allen Metriken (Sensitivität, Spezifität, etc.) die bestmöglichen Werte. Dies könnte darauf hindeuten, dass das Modell möglicherweise überangepasst (overfitted) ist, da es sowohl auf den Trainings- als auch auf den Testdaten perfekt funktioniert. Es ist wichtig, das Modell auf Daten zu bewerten, die es bisher nicht gesehen hat, um sicherzustellen, dass es generalisiert und nicht überangepasst ist.

8 8. Abschluss

8.1 8.1.: Kritik

Folgende Kritikpunkte und offene Fragen sind nach Beendigung der Datenanalyse aufgetreten:

1. Berücksichtigung von Years of Experience und Education Level beim Vergleichen:

   War es angemessen, alle Betrachtungen unter der Gleichstellung von Years of Experience und Education Level in Länder- und Geschlechtsvergleichen zu berücksichtigen? Hätte dies generell in die Datenaufbereitung aufgenommen werden sollen? Dies bedarf genauerer Untersuchung

2. Numerische Transformation nicht-numerischer Werte:

   Wäre es möglich gewesen, alle nicht-numerischen Werte in numerische zu transformieren, um potenziell andere Korrelationen und Regressionen zu identifizieren?

3. Daten aus China:

   Gibt es Anzeichen dafür, dass in China nur Berichte von westlichen Firmen vorliegen, die dort hohe Gehälter zahlen, insbesondere vor dem Hintergrund des niedrigen BIP pro Person?

4. Entscheidungsbaum als reine Klassifikation:

   Hätte der Entscheidungsbaum möglicherweise effektiver als reines Klassifikationsmodell angegangen werden sollen?

5. Beschränkung auf weniger Jobs:

   Hätte es Sinn gemacht, sich in der gesamten Analyse auf einige wenige Jobs zu beschränken, um präzisere Ergebnisse mit weniger Ausreißern zu erzielen?

6. Prüfung der Zuverlässigkeit des Entscheidungsbaumes:

   Existieren alternative Methoden zur Prüfung der Zuverlässigkeit des Entscheidungsbaumes? Falls ja wäre es ratsam gewesen, mehrere dieser Methoden anzuwenden, um mehrere Faktoren zu beurteilen.

7. Data Frames:

   Mit mehr Zeit hätten wir gerne eine übersichtlichere und einheitlichere Transformation der Data Frames angestrebt. Nun haben wir circa 10 verschiedene Data Frames mit teilweise unübersichtlicher Benennung, was nicht nur die Durchlaufzeit erhöht, sondern auch manchmal zu Problemen beim Coden geführt hat.