LV zu den Übungen zu Data Analytics II (inkl. Probeklausur)

Prof. Dr. Wolfgang Bischof

HS Augsburg

10. Oktober 2022

Prof. Dr. Wolfgang Bischof
HS Augsburg
Wirtschaftspsychologie

9. Übungsblatt zur Vorlesung „Data Analytics II“

Aufgabe 9.1 *(Strings, Strings, Strings))*

(i) Lesen Sie die Datei „Kfz_Policen.xlsx“, die Sie in unserem Moodle-Kursraum finden, in den Tibble `Kfz_Policen` ein. Wie viele Policen sind in der Datei aufgelistet?

Lösung:

```
library(tidyverse)
# über Oberfläche
library(readxl)
Kfz_Policen <- read_excel("Kfz_Policen.xlsx")
nrow(Kfz_Policen) # 1500 Policen

## [1] 1500
```

(ii) Geben Sie alle Namen, die mindestens zweimal den Buchstaben „ü“ enthalten, aus. Was fällt Ihnen auf? Wie viele Namen sind betroffen?

Lösung:

```
library(stringr)
Kfz_Policen %>% filter( str_detect( Name, ".*ü.*ü.*" ) )

## # A tibble: 56 x 2
##    Name                              Prämie
##    <chr>                              <dbl>
##  1 Gülsalin Inönü                      736.
##  2 Atilgan Akgündüz                    923.
##  3 Gülbani Sezer Akgündüz              792.
##  4 Dr. Fermuta Pesent Aksu Akgündüz    985.
##  5 Maside Korutürk Yüksel              797.
##  6 Tümkurt Güçlü                       555.
##  7 Bayan Müsüre Hançer Durdu           891.
##  8 Dr. Ülküdes Tuyug Güçlü            1013.
##  9 Bayan Çalim Paye Akgündüz           741.
## 10 Günsel Sümerya Durmus               705.
## # ... with 46 more rows

# es handelt sich um 56 überwiegend, vielleicht sogar ausschließlich, türkische
# Namen
```

(iii) Die Aktuare Ihrer Versicherung haben herausgefunden, dass Menschen, die mindestens zwei „ü“ im Namen tragen, besonders selten in Unfälle verwickelt sind. Daher möchten die Versicherung all diesen Personen einen Rabatt in Höhe von fünf Prozent gewähren. Passen Sie die Policen im Tibble `Kfz_Policen` entsprechend an.

Lösung:

```
Kfz_Policen <- Kfz_Policen %>%
  mutate( Prämie = ifelse( str_detect( Name, ".*ü.*ü.*"), 0.95*Prämie, Prämie ))
```

(iv) Sie können sich nur noch ungefähr an den Namen eines gestrigen Anrufers erinnern, nämlich, dass der Namen „uk“ enthält und mit „ic“ endet. Nun will der Chef aber wissen, wie hoch dessen Prämie ist. Wie finden Sie diese Prämie heraus?

Lösung:

```
Kfz_Policen %>% filter( str_detect(Name, ".*uk.*ic$") )

## # A tibble: 1 x 2
##   Name           Prämie
##   <chr>           <dbl>
## 1 Markus Vukovic   769.
```

(v) Der Fehlerteufel hat zugeschlagen. Alle Personen mit Nachnamen „Schmidt“ heißen in Wirklichkeit „Schmitt“. Dies betrifft aber nicht Personen mit Doppelnamen wie z.B. „Schmidt-Stiffel“ oder „Lebkuchen-Schmidt“. Wie korrigieren Sie den Fehler mit einem geschickten Befehl?

Hinweis: Die Liste händisch zu durchsuchen, mag bei 1500 Verträgen noch sinnvoll sein, aber denken Sie an Datenbanken, die Millionen Verträge enthalten.

Lösung:

```
Kfz_Policen <- Kfz_Policen %>%
  mutate( Name = str_replace(Name, "( Schmidt$)|( Schmidt )", "Schmitt" ) )
# Check
Kfz_Policen %>% filter( str_detect(Name, "Schmidt") ) # passt

## # A tibble: 8 x 2
##   Name                                  Prämie
##   <chr>                                  <dbl>
## 1 Liselotte Schmidt-Stiffel               767.
## 2 Friedericke Löchel-Schmidt              762.
## 3 Ing. Klaus-Dieter Siering-Schmidt B.A.   836.
## 4 Dr. Theresia Lebkuchen-Schmidt          713.
## 5 Trudel Hornig-Schmidtecke               732.
## 6 Univ.Prof. Saban Schmidtke B.A.         741.
## 7 Rosa Schmidt-Mielcarek                  716.
## 8 Ernestine Schmidtke                     818.
```

(vi) Lesen Sie die Datei „pi-10-million.txt“ mit dem Befehl `scan` in den Vektor `pi_str` ein, wobei Sie die Option `what = character()` setzen, damit die Zahl π als durchsuchbare Zeichenkette vorliegt. Die Datei enthält die ersten zehn Millionen Nachkommastellen von π. Sie ist in unserem Moodle-Kursraum zu finden. Lernen Sie diese Nachkommastellen bis zur Klausur auswendig.[1] Ab welcher Nachkommastelle wird erstmals eine Folge von sechs gleichen Ziffern vollendet[2]? Geben Sie zusätzlich die Folge der sechs Ziffer aus.

Lösung:

```
pi_str <- scan(file="pi-10-million.txt", what = character())
str_locate(pi_str, "(\\d)\\1{5,}")  # bereits ab der 762-ten Nachkommastelle

##      start end
## [1,]   762 767

                                    # treten sechs gleiche Ziffern auf
str_extract(pi_str, "(\\d)\\1{5,}") # liefert die erste Folge von sechs gleichen
```

[1] Das war ein kleiner Scherz. Ich empfehle lediglich zu jedem Geburtstag so viele Nachkommastellen von π auswendig zu lernen, wie man Jahre alt geworden ist. Das hält geistig fit.

[2] Das ist der sogenannte Feynman-Punkt. Wer Genaueres wissen will, der lese: `https://de.wikipedia.org/wiki/Feynman-Punkt`

```
## [1] "999999"
```

```
                                        # Ziffern
```

(vii) Bei welcher Nachkommastelle wird erstmals Ihr Geburtsdatum in der Form `TTMMJJ` vollendet, also z.B. die Folge 2,3,0,3,0,1 wenn Sie am 23.3.2001 geboren wurden)?

Lösung:

```
# Bsp.: Geburtsdatum: 13.8.2001
str_locate(pi_str, "130801")
```

```
##       start    end
## [1,] 163273 163278
```

Aufgabe 9.2 *(Textanalysen für Fortgeschrittene)*

(i) Lesen Sie die Dateien „BewerbungEgoist.txt“ und „BewerbungMuster.txt“ mit dem `scan`-Befehl ein. Berechnen Sie für beide Texte die „Ich-Quote“, das ist die relative Häufigkeit des Wortes „ich“ (klein oder groß geschrieben) an allen Wörtern, wobei auch Zahlen als Worte zählen. Was fällt Ihnen auf? Ist der Unterschied signifikant?

Hinweise: Achten Sie beim Einlesen auf das richtige Encoding. Um die Satzzeichen zu eliminieren, fügen sie die Zeichenketten zunächst mit dem Befehl `str_c` *zusammen und zerlegen Sie nachher wieder mit dem Befehl* `str_split`*, wobei Sie die Option* `pattern` *geeignet setzen, um die Worte voneinander zu trennen.*

Lösung:

```
bewerbung_str <- scan(file="BewerbungMuster.txt", what = character(),
                      encoding = "UTF-8")
# zu einer Zeichenkette zusammenfügen
bewerbung_str <- str_c(bewerbung_str, collapse = " ")
# und wieder zerhacken
worte <- str_split(bewerbung_str, pattern = boundary("word"))[[1]]
# Ich-Quote
sum( tolower(worte)=="ich" ) / length(worte)
```

```
## [1] 0.04608295
```

```
# nun dasselbe für den Egoisten
bewerbung_ego_str <- scan(file="BewerbungEgoist.txt", what = character(),
                          encoding = "UTF-8")
# zu einer Zeichenkette zusammenfügen
bewerbung_ego_str <- str_c(bewerbung_ego_str, collapse = " ")
# und wieder zerhacken
worte_ego <- str_split(bewerbung_ego_str, pattern = boundary("word"))[[1]]
# Ich-Quote
sum( tolower(worte_ego)=="ich" ) / length(worte_ego)
```

```
## [1] 0.0660793
```

```
# signifikant?
fisher.test(matrix(c(sum( tolower(worte)=="ich" ), length(worte),
                     sum( tolower(worte_ego)=="ich" ), length(worte_ego)),
                   nrow = 2))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  matrix(c(sum(tolower(worte) == "ich"), length(worte), sum(tolower(worte_ego) == "ich"),
## p-value = 0.4178
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.2740483 1.7020629
## sample estimates:
## odds ratio
##  0.6979213

# nein (p-Wert: 42%)
```

(ii) Lesen Sie die Datei „Faust_I.txt“ mit dem `scan`-Befehl ein und erstellen Sie ein Tibble, in dem die absoluten Häufigkeiten der zehn am meisten benutzen Wörter sowie die Häufigkeit der sonstigen Wörter stehen. Ignorieren Sie dabei Groß-/Kleinschreibung.

Hinweis: Sehen Sie sich zuvor den Text von Goethe an. Welche Worte sollten wir bei der Analyse ausschließen?

Lösung:

```
faust_str <- scan(file="Faust_I.txt", what = character(), encoding = "UTF-8")
# zu einer Zeichenkette zusammenfügen
faust_str <- str_c(faust_str, collapse = " ")
# und wieder zerhacken
worte_faust <- str_split(faust_str, pattern = boundary("word"))[[1]]
# Wörter in Großbuchstaben löschen
worte_tbl <- tibble( Worte = worte_faust)
worte_tbl <- worte_tbl %>% filter( toupper(Worte) != Worte)
# alle Wörter in Kleinbuchstaben schreiben
worte_tbl <- worte_tbl %>% mutate(Worte = tolower(Worte))
# Worte zählen
top10words_faust <- head(worte_tbl %>% count( Worte ) %>% arrange(desc(n)),10)
# sonstige Wörter
top10words_faust <- top10words_faust %>%
  add_row(Worte = "Sonstige", n=nrow(worte_tbl)-sum(top10words_faust$n))
# und alphabetisch sortieren
( top10words_faust <- top10words_faust %>% arrange( Worte ) )
```

```
## # A tibble: 11 x 2
##    Worte        n
##    <chr>    <int>
##  1 das        402
##  2 der        584
##  3 die        642
##  4 du         312
##  5 ein        395
##  6 ich        691
##  7 ist        362
##  8 nicht      426
##  9 Sonstige 24462
## 10 und        917
## 11 zu         379
```

(iii) Unser Ziel ist es nun, nur aufgrund von Worthäufigkeiten erkennen zu können, ob es sich bei einem fremden Text um einen Auszug aus Faust II handelt oder um einen anderen Text.

Lesen Sie dazu die beiden Dateien „Faust_II_Lustgarten_100.txt“ und „HSA-Qualitaetskodex-Novelle-2020.txt“ ein. Ermitteln Sie, wie oft die in (ii) ermittelten Wörter auftreten. Führen Sie anschließen Chi-Quadrat-Anpassungstests durch. Ist ein Unterschied zwischen den beiden Texten ersichtlich?

Hinweis: Sollte ein Wort aus der Liste aus (ii) in einem der Texte gar nicht auftreten, müssen Sie es vor Durchführung des Chi-Quadrattests mit dem Befehl `add_row()` *zum Tibble hinzufügen.*

Quellen zu (ii) & (iii):

- Goethe-Texte: `http://www.digbib.org`
- HSA-Qualitätskodex-Novelle `https://www.hs-augsburg.de/`

Lösung:

```
# Check mit Lustgarten aus "Faust II"
lustgarten_str <- scan(file="Faust_II_Lustgarten_100.txt", what = character(),
                       encoding = "UTF-8")
# zu einer Zeichenkette zusammenfügen
lustgarten_str <- str_c(lustgarten_str, collapse = " ")
# und wieder zerhacken
worte_lustgarten <- str_split(lustgarten_str, pattern = boundary("word"))[[1]]
# Wörter in Großbuchstaben löschen
worte_tbl <- tibble( Worte = worte_lustgarten)
worte_tbl <- worte_tbl %>% filter( toupper(Worte) != Worte)
# nur die Top-10 aus Faust zählen
worte_tbl <- tibble( Worte = tolower(worte_lustgarten))
worte_lustgarten <- worte_tbl %>% filter( Worte %in% top10words_faust$Worte )
# Wörter zählen
top10words_lustgarten <- worte_lustgarten %>% count( Worte )
# sonstige Wörter
top10words_lustgarten <- top10words_lustgarten %>%
  add_row(Worte = "Sonstige", n=nrow(worte_tbl)-sum(top10words_lustgarten$n)) %>%
  arrange( Worte )

# Testen, ob Unterschied signifikant ist
chisq.test( top10words_lustgarten$n, p=top10words_faust$n, rescale.p = TRUE)


##
##  Chi-squared test for given probabilities
##
## data:  top10words_lustgarten$n
## X-squared = 11.382, df = 10, p-value = 0.3285

# kein signifikanter Untwerschied

# Check mit HSA-Qualitaetskodex-Novelle-2020
novelle_str <- scan(file="HSA-Qualitaetskodex-Novelle-2020.txt",
                        what = character(), encoding = "UTF-8")
# zu einer Zeichenkette zusammenfügen
novelle_str <- str_c(novelle_str, collapse = " ")
# und wieder zerhacken
worte_novelle <- str_split(novelle_str, pattern = boundary("word"))[[1]]
# nur die Top-10 aus Faust zählen
worte_tbl <- tibble( Worte = tolower(worte_novelle))
worte_novelle <- worte_tbl %>% filter( Worte %in% top10words_faust$Worte )
# Wörter zählen
top10words_novelle <- worte_novelle %>% count( Worte )
# sonstige Wörter sowie  "du" und "ich" ergänzen, weil die in der Novelle nicht
# vorkommen
top10words_novelle <- top10words_novelle %>%
```

```
  add_row(Worte = c("Sonstige","ich","du"),
          n=c(nrow(worte_tbl)-sum(top10words_novelle$n),0,0)) %>%
  arrange( Worte )

# Testen, ob Unterschied signifikant ist
chisq.test( top10words_novelle$n, p=top10words_faust$n, rescale.p = TRUE)

##
##  Chi-squared test for given probabilities
##
## data:  top10words_novelle$n
## X-squared = 157.04, df = 10, p-value < 2.2e-16

# yep: signifikanter Unterschied => Goethe war nicht in der Kommission
# zur Erstellung des HSA-Qualitätskodexes
```

Aufgabe 9.3 *(Kategoriale Größen in „General Social Survey“))*

Betrachten Sie den Datensatz `gss_cat` aus der `tidyverse`-Bibliothek.

(i) Untersuchen Sie die Verteilung von `rincome` (= reported income = gemeldetes Einkommen). Warum ist das Standard-Säulendiagramm schwer zu verstehen? Wie können Sie die Darstellung verbessern?

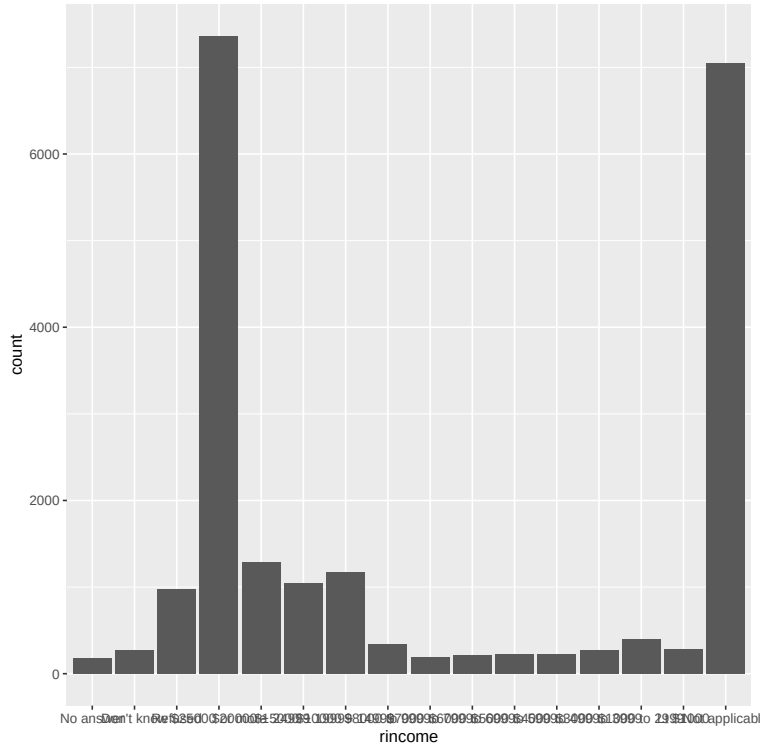
Lösung:

```
library(tidyverse)

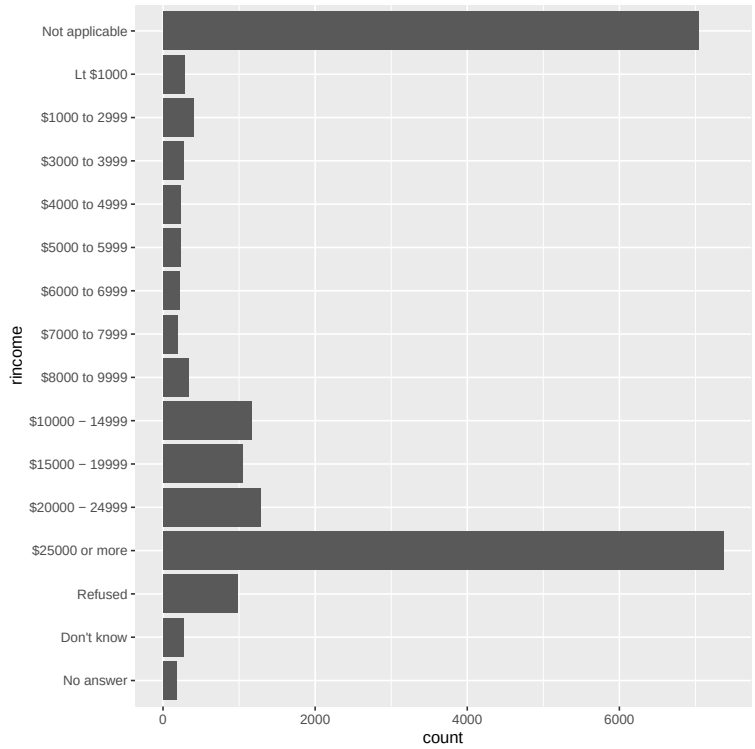
summary(gss_cat) # für einen ersten Überblick

##       year                marital           age                     race
##  Min.   :2000   No answer    :   17   Min.   :18.00   Other         : 1959
##  1st Qu.:2002   Never married: 5416   1st Qu.:33.00   Black         : 3129
##  Median :2006   Separated    :  743   Median :46.00   White         :16395
##  Mean   :2007   Divorced     : 3383   Mean   :47.18   Not applicable:    0
##  3rd Qu.:2010   Widowed      : 1807   3rd Qu.:59.00
##  Max.   :2014   Married      :10117   Max.   :89.00
##                                       NA's   :76
##            rincome                   partyid            relig
##  $25000 or more:7363   Independent       :4119   Protestant:10846
##  Not applicable:7043   Not str democrat  :3690   Catholic  : 5124
##  $20000 - 24999:1283   Strong democrat   :3490   None      : 3523
##  $10000 - 14999:1168   Not str republican:3032   Christian :  689
##  $15000 - 19999:1048   Ind,near dem      :2499   Jewish    :  388
##  Refused       : 975   Strong republican :2314   Other     :  224
##  (Other)       :2603   (Other)           :2339   (Other)   :  689
##                denom          tvhours
##  Not applicable  :10072   Min.   : 0.000
##  Other           : 2534   1st Qu.: 1.000
##  No denomination : 1683   Median : 2.000
##  Southern baptist: 1536   Mean   : 2.981
##  Baptist-dk which: 1457   3rd Qu.: 4.000
##  United methodist: 1067   Max.   :24.000
##  (Other)         : 3134   NA's   :10146

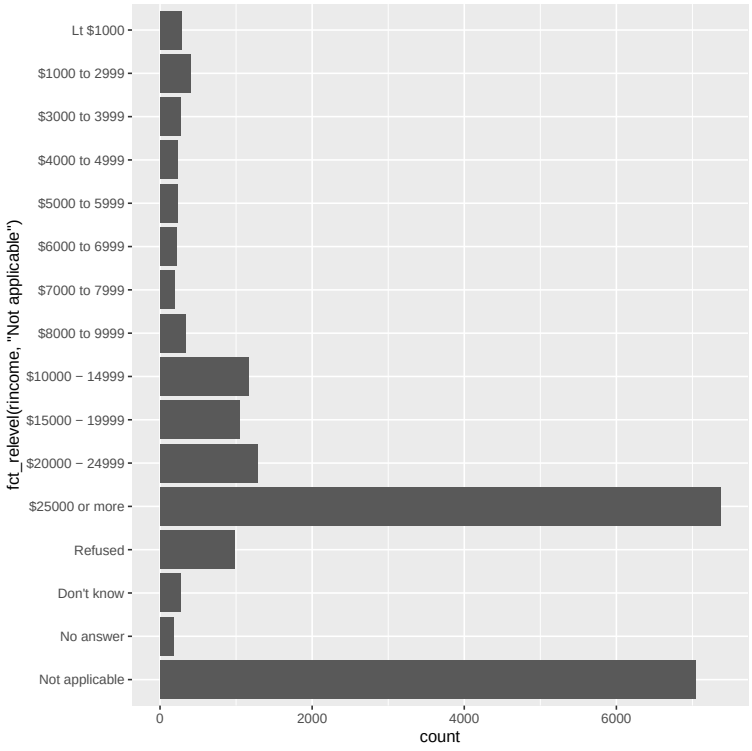
ggplot(gss_cat) + geom_bar(aes(x=rincome))
```




```
# die x-Achsenbeschriftungen lassen sich nicht gut lesen, deshalb drehen wir
# das Diagramm um 90 Grad
ggplot(gss_cat) + geom_bar(aes(y=rincome))
```



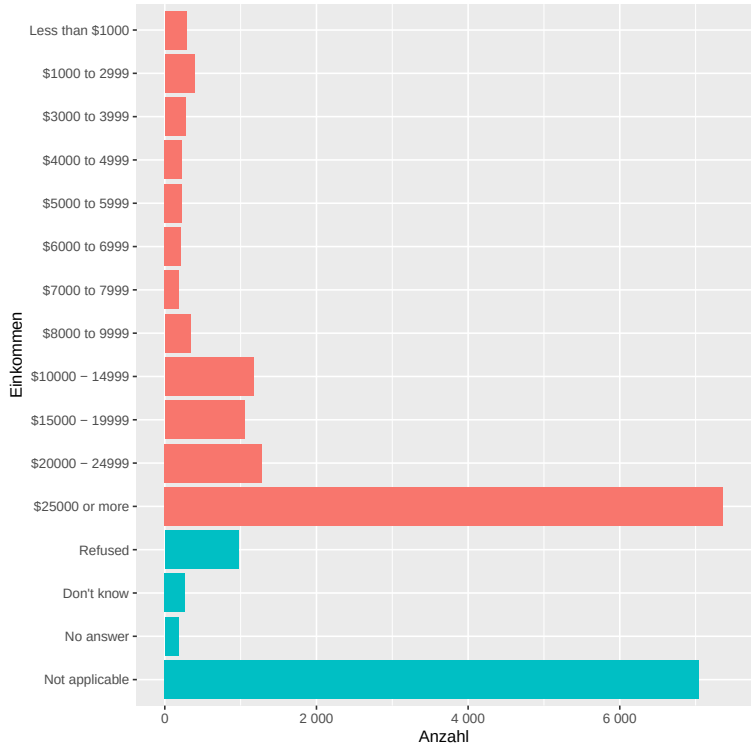

```
# "Not applicable" sollte nach unten sortiert werden
ggplot(gss_cat) + geom_bar(aes(y=fct_relevel(rincome, "Not applicable")))
```



Wenn man möchte, kann man die Grafik noch weiter aufhübschen:

- „Less than $1000“ statt „Lt $1000“
- andere Farbe für die Kategorien, bei denen die Angabe des Einkommens fehlt (’Refused’, ’Don’t know’, ’No answer’, and ’Not applicable’)
- aussagekräftige Achsenbezeichnungen
- schmale Leerzeichen als Tausendertrennzeichen
- keine Legende

```
aux_blank <- function(x) { return (scales::comma(x,big.mark = " "))}
gss_cat %>%
  mutate(rincome = fct_recode(rincome,
    "Less than $1000" = "Lt $1000"
  )) %>%
  mutate(rincome_na = rincome %in% c("Refused", "Don’t know", "No answer",
                                     "Not applicable")) %>%
  ggplot() + geom_bar(aes(y=fct_relevel(rincome, "Not applicable"),
                          fill = rincome_na)) +
  scale_x_continuous("Anzahl", labels = aux_blank) +
  scale_y_discrete("Einkommen")+
  theme(legend.position = "None")
```



(ii) Welches ist die häufigste Religion in dieser Umfrage? Welches ist die häufigste `partyid`?

Lösung:

```
gss_cat %>% group_by(relig) %>% summarise( Anzahl = n()) %>%
  filter( Anzahl == max(Anzahl) )

## # A tibble: 1 x 2
##   relig      Anzahl
##   <fct>       <int>
## 1 Protestant  10846

# für die zweite Teilaufgabe wählen wir eine andere Alternative
gss_cat %>% count(partyid) %>% arrange(desc(n)) %>% head(1)

## # A tibble: 1 x 2
##   partyid         n
##   <fct>       <int>
## 1 Independent  4119
```

(iii) Für welche der Faktoren in `gss_cat` gibt es eine natürliche Reihenfolge der Stufen? Wie können die anderen Faktoren in einem Säulendiagramm sinnvoll sortiert werden?

Lösung:
Nur für `rincome`, dessen Stufen man nach auf- oder absteigenden Einkommen sortieren sollte. Für die anderen Faktoren bietet es sich an, nach auf- oder absteigender Anzahl an Beobachtungen zu sortieren. Stufen, die zu fehlenden Angaben gehören, sollten in allen Fällen an den Anfang oder ans Ende sortiert werden.

(iv) Wie haben sich die Anteile der Menschen, die sich als Demokraten, Republikaner und Unabhängige ausgeben, im Laufe der Zeit verändert? Erstellen Sie eine aussagekräftige und gut lesbare Grafik.

Lösung:

```
# Stufen zu den Parteien Demokraten, Republikaner und Unabhängige
# zusammenfassen
gss_cat %>%
```

```
  mutate(partyid = fct_collapse( partyid,
    other = c("No answer", "Don't know", "Other party"),
    rep = c("Strong republican", "Not str republican"),
    ind = c("Ind,near rep", "Independent", "Ind,near dem"),
    dem = c("Not str democrat", "Strong democrat") )) %>%
  count(year, partyid) %>%
  group_by(year) %>%
  mutate( Anteil = n / sum(n) ) %>%
  ggplot( aes(x=year, y=Anteil, color=fct_reorder2(partyid, year, Anteil)) ) +
  geom_point() +
  geom_line() +
  scale_x_discrete("Jahr") +
  labs(color = "Party ID")
```

