Übungszettel Indexerstellung

M.Psy.205, Dozent: Dr. Peter Zezula

Joshua Driesen (joshua.driesen@stud.uni-goettingen.de)

#German

Links

Übungszettel als PDF-Datei zum Drucken

Übungszettel mit Lösungen:

Lösungszettel als PDF-Datei zum Drucken

Der gesamte Übugszettel als .Rmd-Datei (Zum Downloaden: Rechtsklick > Speichern unter…)

Hinweise zur Bearbeitung

1. Bitte beantworten Sie die Fragen in einer .Rmd Datei. Sie können Sie über `Datei > Neue Datei > R Markdown...` eine neue R Markdown Datei erstellen. Den Text unter dem *Setup Chunk* (ab Zeile 11) können Sie löschen. Unter diesem Link können Sie auch unsere Vorlage-Datei herunterladen (Rechtsklick > Speichern unter…).
2. Informationen, die Sie für die Bearbeitung benötigen, finden Sie auf der Website der Veranstaltung
3. Zögern Sie nicht, im Internet nach Lösungen zu suchen. Das effektive Suchen nach Lösungen für R-Probleme im Internet ist tatsächlich eine sehr nützliche Fähigkeit, auch Profis arbeiten auf diese Weise. Die beste Anlaufstelle dafür ist der R-Bereich der Programmiererplattform Stackoverflow
4. Auf der Website von R Studio finden Sie sehr hilfreiche Übersichtszettel zu vielen verschiedenen R-bezogenen Themen. Ein guter Anfang ist der Base R Cheat Sheet

Aufgabe 1: SerienmörderInnen

Erste Schritte

Setzen Sie das gewünschte Arbeitsverzeichnis, und laden Sie das `tidyverse`-Paket. Laden Sie im Anschluss den Datensatz `https://pzezula.pages.gwdg.de/data/serial_killers.txt` (Quelle) herunter, oder lesen Sie ihn direkt aus der URL ein. Machen Sie sich die Struktur des Datensatzes klar.

Index Erstellen

Erstellen Sie einen Index, in dem Sie die Häufigkeit von SerienmörderInnen innerhalb eines Landes mit der Bevölkerung des Landes in Beziehung setzen. Skalieren Sie diesen so, dass ein Land, das genauso viele SerienmörderInnen hat, wie bei der Größe seiner Bevölkerung zu erwarten wäre, den Wert 10 erhält, und höhere Werte einer größeren Häufigkeit entsprechen. (Hinweis: Hierzu dürfen Sie voraussetzen, dass aus

Ländern, die nicht im Datensatz vorkommen, keine Serienmorde bekannt sind. In Wahrheit verteilen sich "nur" 93,2% aller SerienmörderInnen auf die aufgeführten Länder.)

Kritik

Reflektieren Sie kritisch über die verwendeten Daten und mögliche Folgen für den von Ihnen erstellten Index. Gehen Sie hierbei insbesondere auf mögliche Konfundierungen ein.

Lösung

Erste Schritte

```
setwd("P:/MV")
library(tidyverse)

serial_killers <- read_delim("https://pzezula.pages.gwdg.de/data/serial_killers.txt",
                             delim = "\t")
```

Index erstellen

Zuerst ohne dplyr.

```
serial_killers$Share_of_killers <- serial_killers$Number_of_known_serial_killers /
                                sum(serial_killers$Number_of_known_serial_killers)
```

Hier wird zunächst der Anteil des Landes an allen bekannten SerienmörderInnen berechnet. Entspricht dieser Anteil gleichzeitig dem Anteil des Landes an der Weltbevölkerung, hat dieses Land so viele Fälle wie von seiner Bevölkerungszahl her zu erwarten war.

```
serial_killers$UnstandardisierterIndex <- (serial_killers$Share_of_killers /
                                 serial_killers$Share_of_world_population_in_percent)
```

`Share_of_killers` ist als Dezimalanteil angegeben, `Share_of_world_population` dagegen in Prozent. Bei einem Land mit genau der erwarteten Anzahl wäre das Verhältnis der beiden 1:100, also 0.01. Um diesen Wert nun auf 10 zu standardisieren, reicht eine Multiplikation mit 1000:

```
serial_killers$Index <- serial_killers$UnstandardisierterIndex * 1000
```

Dann noch einmal mit dplyr, sowie mit anderer Vorgehensreihenfolge:

```
serial_killers <- mutate(serial_killers,
                         killers_expected =
                            (Share_of_world_population_in_percent / 100) *
                            sum(Number_of_known_serial_killers)) %>%
                  mutate(Index2 =
                            (Number_of_known_serial_killers /
                            killers_expected)
                            * 10)
```

Hier wurde zunächst die zu erwartende Killerzahl explizit berechnet, und anschließend mit der tatsächlichen Anzahl verglichen. Das hier gebildete Verhältnis von realer und erwarteter Zahl musste dann nur noch mit 10 multipliziert werden, um auf die gewünschte Verteilung zu kommen.

Kritik

Das größte Problem bei den verwendeten Daten liegt in einer vermutlich recht starken Konfundierung mit anderen Eigenschaften der Länder, insbesondere mit der dortigen Strafverfolung. Zwei mögliche Konfundierungsquellen wären hier:

- good cop: Die Polizei in dem entsprechenden Land ist so gut, dass Sie den/die TäterIn bereits nach der ersten Tat dingfest machen können, sodass es gar nicht erst zu einer Mordserie kommt.
- bad cop: Die Polizei in dem entsprechenden Land ist so schlecht, dass Sie die einzelnen Taten nicht miteinander in Verbindung bringt, und so eine Mordserie nicht als solche erkennt.

Aufgabe 2: Geschlechterverhältnisse deutscher Städte

Datensatz einlesen

Lesen Sie den Datensatz `https://pzezula.pages.gwdg.de/data/Staedte.txt` ein. (Quelle: Statistisches Bundesamt, 2016) Hierin befindet sich eine Liste aller deutscher Städte, sortiert nach der Bevölkerungszahl. Die Bevölkerung finden Sie auch im Datensatz, einmal insgesamt, und einmal aufgeschlüsselt nach Geschlecht.

Index erstellen

Erstellen Sie einen Index, der das Geschlechterverhältnis der deutschen Städte abbildet. Dabei soll der Wertebereich *symmetrisch* um null herum verteilt sein, sprich: Der Betrag des Indexes soll für zwei gleich große, entgegengerichtete Ungleichverteilungen gleich groß sein.

Tipp: schauen Sie sich die mögliche Verteilung Ihres Index über den Wertebereich an.

Extremwerte verstehen

Suchen Sie anhand des von Ihnen erstellten Index die drei Städte mit dem größten Ungleichgewicht der beiden Geschlechter. Versuchen Sie mit einer Internetrecherche erste Ansatzpunkte für mögliche Gründe für diese Extremwerte zu ermitteln.

Mit der Zeit gehen

Die Daten wurden vor der Einführung des sog. dritten Geschlechts erhoben, und spiegeln daher die binäre Kategorisierung damaliger Rechtssprechung wieder:

```
all(Staedte$weiblich + Staedte$maennlich == Staedte$Gesamtbevoelkerung)
```

```
## [1] TRUE
```

Überprüfen Sie den von Ihnen erstellten Index darauf, ob er gegenüber der Einführung einer dritten Geschlechtskategorie robust ist, also bei gleich vielen Frauen und Männern trotzdem den Wert 0 annimmt. Falls nicht, überlegen Sie sich auch eine mögliche Korrektur!

Bonusaufgabe: Zipf's Law

Das Zipfsche Gesetz (Wikipedia) stammt ursprünglich aus der quantitativen Linguistik, und besagt: "Wenn die Elemente einer Menge – beispielsweise die Wörter eines Textes – nach ihrer Häufigkeit geordnet werden, ist die Wahrscheinlichkeit *p* ihres Auftretens umgekehrt proportional zur Position *n* innerhalb der Rangfolge: $p(n) \sim \frac{1}{n}$."

Es gibt einige Hinweise darauf, dass auch Städte innerhalb eines Landes diesem Gesetz folgen. Konkret heißt das: Die zweitgrößte Stadt hat halb so viele Einwohner wie die größte, die drittgrößte ein Drittel so viele, die viertgrößte ein Viertel, und so weiter.

Generieren Sie in Ihrem Datensatz eine neue Variable, in der sich die gemäß Zipf erwarteten Einwohnerzahlen der Städte befinden. Nutzen Sie im Anschluss einen Chi-Quadrat-Test (`chisq.test()`), um zu überprüfen, ob sich die tatsächliche und die erwartete Einwohnerverteilung signifikant voneinander unterscheiden.

Lösung

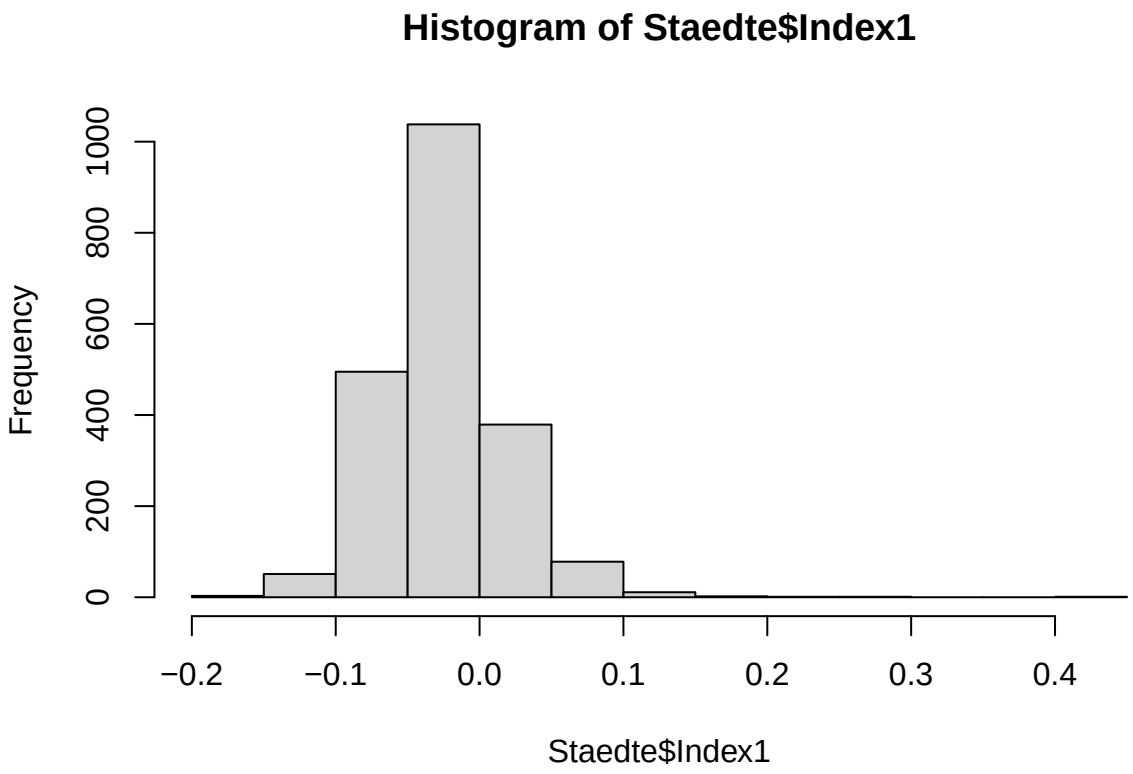
Datensatz einlesen

```
Staedte <- read_delim("https://pzezula.pages.gwdg.de/data/Staedte.txt",
                              delim = "\t")
```

Index erstellen

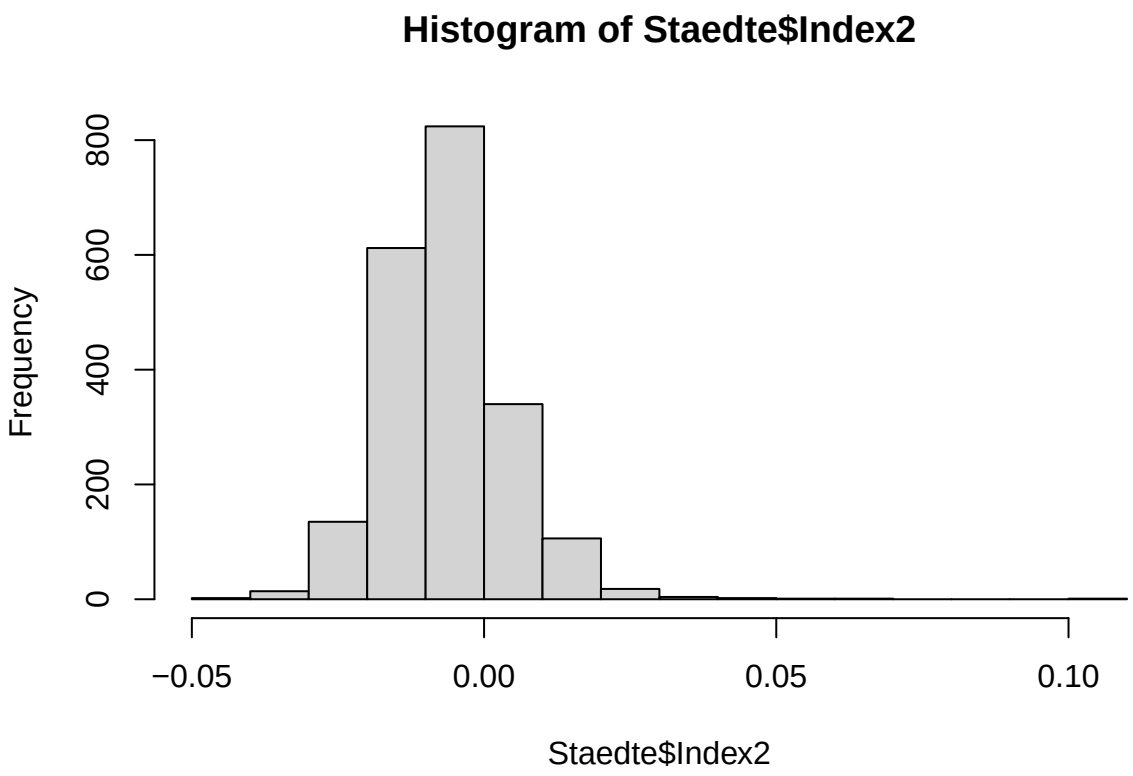
Einen guten Ansatzpunkt liefert das Verhältnis der beiden Geschlechter. Dies ist aber weder symmetrisch (0,5 ist numerisch näher an 1 als die 2, obwohl beide Zahlen einem gleich großen Ungleichgewicht entsprechen), noch um null zentriert. Hierfür bietet Logarithmieren eine praktische Lösung: Der Betrag des Logarithmus von 0,5 und 2 ist derselbe, und der Logarithmus von 1 ist 0. Dieser Ansatz wird hier ohne `dplyr` gerechnet:

```
Staedte$Index1 <- log(Staedte$maennlich / Staedte$weiblich)

hist(Staedte$Index1)
```



Alternativ kann auch die Abweichung eines GeschlechterANTEILS von 50% als Index verwendet werden. Hier wird der Frauenanteil verwendet, damit positive Werte wie bei `Index1` einem höheren Männeranteil entsprechen. Für diesen Ansatz wird hier `dplyr` verwendet:

```
Staedte <- mutate(Staedte, Index2 = 0.5 -
                    (weiblich / Gesamtbevoelkerung))

hist(Staedte$Index2)
```



Extremwerte verstehen

Die Histogramme legen bereits nahe, dass Extremwerte eher im positiven Bereich, also bei Männerübergewicht vorliegen. In der `View(Staedte)`-Ansicht könnte man diese Werte finden, indem man nach dem erstellten Index sortiert, und einfach die obersten drei Zeilen ansieht. R kann einem diese Arbeit aber natürlich auch abnehmen:

```
ExtremStaedte <- arrange(Staedte, #nimmt den Datensatz Staedte, und sortiert ihn
                         desc( #in ABSTEIGENDER (descending) Reihenfolge
                           abs( #nach dem ABSOLUT-Betrag
                             Index1))) %>% #von Index1
                 slice(1:3) #und behält nur die obersten drei Einträge darin.

ExtremStaedte
```

```
## # A tibble: 3 x 9
##    Rang Bundesland Stadt                                    Flaeche Gesamtbevoelker~ maennlich weibl
##   <dbl>      <dbl> <chr>                                      <dbl>            <dbl>     <dbl>  <d
## 1  2035          6 Schwarzenborn, Stadt                       26.9             1408       848
## 2  2026         16 Hohenleuben, Stadt                          9.53            1537       865
## 3   824          3 Clausthal-Zellerfeld, Berg- und Universi~  43.7            15818      8826     6
```

Die drei Städte mit dem größten Geschlechterungleichgewicht sind Schwarzenborn, Hohenleuben und Clausthal-Zellerfeld. Eine kurze Recherche erklärt die Geschlechterverteilung: In allen drei Städten sind

die Hauptarbeitgeber aus Branchen, die traditionell eher männliche Mitarbeiter haben: die Bundeswehr in Schwarzenborn, eine Justizvollzugsanstalt in Hohenleuben und Technische Universität/Bergbau in Clausthal-Zellerfeld.

Mit der Zeit gehen

`Index1` betrachtet nur das Verhältnis der beiden Geschlechter, und ist somit gegen die Einführung einer weiteren Kategorie robust. `Index2` dagegen würde durch diese Einführung seine Validität verlieren. Man könnte jedoch eine Korrektur einführen, wenn man den Anteil des "dritten Geschlechts" kennt, indem man die Hälfte dieses Anteils von dem Index abzieht. Ein Beispiel:

Bei 45% Frauen, 45% Männern und 10% "Diversen" (Eine schreckliche Wortwahl übrigens) würde der unkorrigierte `Index2` den Wert `0.50 - 0.45 = 0.05` annehmen. Zieht man von nun `0.10 / 2 = 0.05` davon ab, wird der Wert null, und zeigt somit wieder korrekt das genaue Gleichgewicht von Frauen und Männern an.

Bonusaufgabe: Zipf's Law

```
Staedte$Zipf_prediction <- (1 / Staedte$Rang) * max(Staedte$Gesamtbevoelkerung)

chisq.test(Staedte$Gesamtbevoelkerung, Staedte$Zipf_prediction)
```

```
## Warning in chisq.test(Staedte$Gesamtbevoelkerung, Staedte$Zipf_prediction): Chi-squared approximatio
```

```
##
##  Pearson's Chi-squared test
##
## data:  Staedte$Gesamtbevoelkerung and Staedte$Zipf_prediction
## X-squared = 4093220, df = 4091233, p-value = 0.2436
```

Deutsche Städte scheinen nicht signifikant von der durch das Zipf'sche Gesetz vorhergesagten Verteilung abzuweichen.

Aufgabe 3: Bachelorbewerbungen

Datensatz einlesen

Lesen Sie den Datensatz `https://pzezula.pages.gwdg.de/data/bsc_bewerbung.txt` ein. In diesem finden Sie die Abiturnoten von 200 (fiktiven) BewerberInnen auf einen Bachelor-Studienplatz am GEMI. Machen Sie sich die Struktur des Datensatzes klar.

Notendurchschnitt erstellen

Erzeugen Sie zwei neue Variablen in dem Datensatz, in denen der Notendurchschnitt vermerkt wird. Behalten Sie für die erste Variable die gegebene Skalierung mit 0 bis 15 Punkten bei. Für die zweite Variable skalieren Sie bitte auf klassische Schulnoten von 1 bis 6 um. Wichtig: Achten Sie darauf, die Bewerbendennummer in der ersten Spalte *nicht* in den Notendurchschnitt miteinzubeziehen!

GEMI-Score

Bachelorstudienplätze am GEMI werden nicht bloß auf Basis der Abinote vergeben, sondern anhand einer eigenen Notengewichtungsformel. In diese geht die Gesamtnote zu 80%, die Englisch-Note zu 10%, und Deutsch und Mathe zu je 5% ein. Bilden Sie diesen Score in einer neuen Variable nach. Transformieren Sie diesen Score dabei auf den Wertebereich 0 bis 100!

Eigener Score

Reflektieren Sie über Ihr eigenes Bachelorstudium: Welche Fähigkeiten und welches Vorwissen erwies sich hierfür als nützlich? Erstellen Sie anhand dieser Überlegungen Ihren eigenen gewichteten Bewerbungsscore, ebenfalls mit dem Wertebereich 0 bis 100. Schreiben Sie ggf. ein paar Worte zur Begründung Ihrer Entscheidungen!

Zulassung

Leider können Sie von den 200 BewerberInnen nur 100 für ein Bachelorstudium zulassen. Erzeugen Sie sowohl für den GEMI-Score, als auch für Ihren selbst entwickelten Score eine neue Variable, in der die Zulassungsentscheidung anhand des jeweiligen Scores kodiert wird. Hierfür bietet sich der Variablentyp `logical` an.

Score-Vergleich

Analysieren Sie, in wieviel Prozent der Fälle der von Ihnen entwickelte und der GEMI-Score zu verschiedenen Ergebnissen kommen!

Lösung

Datensatz einlesen

```
bsc_bewerbung <- read_delim("https://pzezula.pages.gwdg.de/data/bsc_bewerbung.txt",
                            delim = "\t")
```

Notendurchschnitt erstellen

```
bsc_bewerbung$DurchschnittPunkte <- rowMeans(bsc_bewerbung[,-1])
#[,-1] = erste Spalte nicht aufnehmen

bsc_bewerbung <- mutate(bsc_bewerbung,
                        Durchschnitt_1bis6 =
                          (17 - DurchschnittPunkte) / 3)
#Eine Korrektur, die bei 0 Punkten die Note 6 vergibt,
#ist bei unseren Daten unnötig.
```

GEMI-Score erstellen

```
bsc_bewerbung$GEMI_score <- ((100 / 15) * #Wertebereich auf 0 bis 100
          ((0.80 * bsc_bewerbung$DurchschnittPunkte) +
           (0.10 * bsc_bewerbung$Englisch) +
           (0.05 * bsc_bewerbung$Deutsch) +
           (0.05 * bsc_bewerbung$Mathe)))
```

Eigener Score

```
bsc_bewerbung$eigener_score <- ((100 / 15) *
          ((0.10 * bsc_bewerbung$Bio) + #Immer mehr Neuro-Fokus
           (0.10 * bsc_bewerbung$Mathe) + #Statistik
           (0.10 * bsc_bewerbung$Englisch) + #Schnell Paper verstehen
           (0.70 * bsc_bewerbung$DurchschnittPunkte)))
```

Zulassung

Da wir genau die Hälfte unserer BewerberInnen zulassen wollen, bietet sich der Median als Cut-Off an, hier am Beispiel GEMI_score:

```
bsc_bewerbung$Zulassung_GEMI <- bsc_bewerbung$GEMI_score >
                                median(bsc_bewerbung$GEMI_score)

sum(bsc_bewerbung$Zulassung_GEMI)
```

```
## [1] 100
```

```
#Genau 100, weil keine Rangbindung beim Median
```

Einen flexiblerer Ansatz, der auch bei anderen Zulassungsraten funktioniert, bietet die `rank()`-Funktion. Hier für den eigenen Score:

```
bsc_bewerbung$Zulassung_eigen <- rank(bsc_bewerbung$eigener_score) > 100

sum(bsc_bewerbung$Zulassung_eigen)
```

```
## [1] 100
```

Score-Vergleich

```
bsc_bewerbung$Unterschiedliche_Entscheidung <-
                                      bsc_bewerbung$Zulassung_GEMI !=
                                      bsc_bewerbung$Zulassung_eigen
```

```r
Anzahl_Disagreements <- sum(bsc_bewerbung$Unterschiedliche_Entscheidung)

Prozent_Disagreement <- (Anzahl_Disagreements / nrow(bsc_bewerbung)) * 100

Prozent_Disagreement
```

```
## [1] 10
```

English

Links

Exercise sheet in PDF

Exercise sheet with solutions included

Exercise sheet with solutions included as PDF

The source code of this sheet as .Rmd (Right click and "store as" to download …)

Some hints

1. Please give your answers in a .Rmd file. You may generate one from scratch using the file menu: 'File > new file > R Markdown …' Delete the text below *Setup Chunk* (starting from line 11). Alternatively you may use this sample Rmd by donloading it.
2. You may find the informations useful that you can find on the start page of this course.
3. Don't hesitate to google for solutions. Effective web searches to find solutions for R-problems is a very useful ability, professionals to that too … A really good starting point might be the R area of the programmers platform Stackoverflow
4. You can find very useful cheat sheets for various R-related topics. A good starting point is the Base R Cheat Sheet.

Task 1: Serial killers

First steps

Load the tidyverse and download the data frame at `https://pzezula.pages.gwdg.de/data/serial_killers.txt` (Source). You can also read in the data frame directly from the URL. Take a look at the the data structure.

Index creation

Create an index that puts the frequency of serial killers in a country in relation to the population of that country. It should be scaled in a way that gives a value of 10 to countries having exactly as many serial killers as would be expected from their population size. Higher values should indicate an elevated frequency. (Note: For this task, you can assume that countries not listed here don't have any known serial killers. In reality, 'only' 93.2% of all known serial killers come from the countries listed here.)

Critique

Reflect critically on the data used here and possible consequences for the index you created. Take special note of possible confounders.

Solution

First steps

```
setwd("P:/MV")
library(tidyverse)

serial_killers <- read_delim("https://pzezula.pages.gwdg.de/data/serial_killers.txt",
                             delim = "\t")
```

Index creation

First without dplyr.

```
serial_killers$Share_of_killers <- serial_killers$Number_of_known_serial_killers /
                                sum(serial_killers$Number_of_known_serial_killers)
```

Here, we first calculate the countries' share of known serial killers. If this share is equal to a country's share of the world population, then that country has as many cases as would be expected from its population.

```
serial_killers$UnstandardisierterIndex <- (serial_killers$Share_of_killers /
                                 serial_killers$Share_of_world_population_in_percent)
#UnstandardisierterIndex - non-standardized index
```

`Share_of_killers` is given as a decimal, `Share_of_world_population` in percent. In a country with matching shares in serial killers and population, their ratio would be 1:100 = 0.01. To standardize this to the value 10, we can simply multiply it by 1000:

```
serial_killers$Index <- serial_killers$UnstandardisierterIndex * 1000
```

The same again, using dplyr and a different order of steps:

```
serial_killers <- mutate(serial_killers,
                         killers_expected =
                            (Share_of_world_population_in_percent / 100) *
                            sum(Number_of_known_serial_killers)) %>%
                  mutate(Index2 =
                            (Number_of_known_serial_killers /
                            killers_expected)
                            * 10)
```

Here, we first explicitly calculated the expected number of serial killers, and then compared that to the real number of cases. The ratio of real and expected case numbers just needed to be multiplied with 10 to get to the scaling we needed.

Critique

The biggest problem with the data we used is a high chance of confounding with other characteristics of the countries studied, especially with differences in local law enforcement. Two examples of that could be:

- good cop: Police in a given country is so good that most potential serial killers are caught after their first offense, thus keeping them from becoming serial killers at all.
- bad cop: Police in a given country is so bad that they fail to connect the different offenses to one single offender, thus failing to recognise a serial killer as such.

Task 2: Gender ratios of German cities

Read data

Read in the data frame `https://pzezula.pages.gwdg.de/data/Staedte.txt`. (Source: Statistisches Bundesamt, 2016) In it, you'll find a list of all German cities, sorted by size of population. The population's in the data frame, too, once as total and once broken down by sex.

German	English
Staedte	cities
Rang	rank
Bundesland	federal state
Stadt	city
Flaeche	area
Gesamtbevoelkerung	total population
maennlich	male
weiblich	female

Index creation

Create an index representing the ratio of the two sexes. The scale has to be *symmetrical* around zero, i.e. the absolute value of the score of two inequalities in opposing directions, but of the same size, should be the same.

Understanding extreme values

Use your index to find the three cities with the biggest inequalities between male and female inhabitants. Try researching the web to find clues how these inequalities come to be. (Note: this might prove impossible without German Wikipedia for the smaller cities. If so, skip to the next task.)

Keeping up with the times

The present data were aggregated before the legal introduction of the so-called 'third gender', which is why they still contain the binary gender categorization:

```
all(Staedte$weiblich + Staedte$maennlich == Staedte$Gesamtbevoelkerung)
```

```
## [1] TRUE
```

Check whether your index could handle the introduction of a third gender categorie, i.e. if it would still be zero when there are the same number of men and women. If not, try to fix that!

Bonus task: Zipf's Law

Zipf's Law (Wikipedia) originally stems from quantitative linguistics and states: If you sort the elements of a set - e.g. words from a text - by their frequency, then the probability of one element's occurence *p* is inversely proportianal to its position *n* in the ranked order: $p(n) \sim \frac{1}{n}$."

There's some evidence that cities within one country follow this law as well. In practice, this means: There's half as many inhabitants in the second biggest city as in the biggest, a third as many in the third biggest as in the biggest, a fourth as many in the fourth biggest etc.

Add a new variable to your data frame containing the population size predictions based on Zipf's law. Next, use a chi-square-test (`chisq.test()`) to check whether the real and the expected distributions of population differ significantly.

Solution

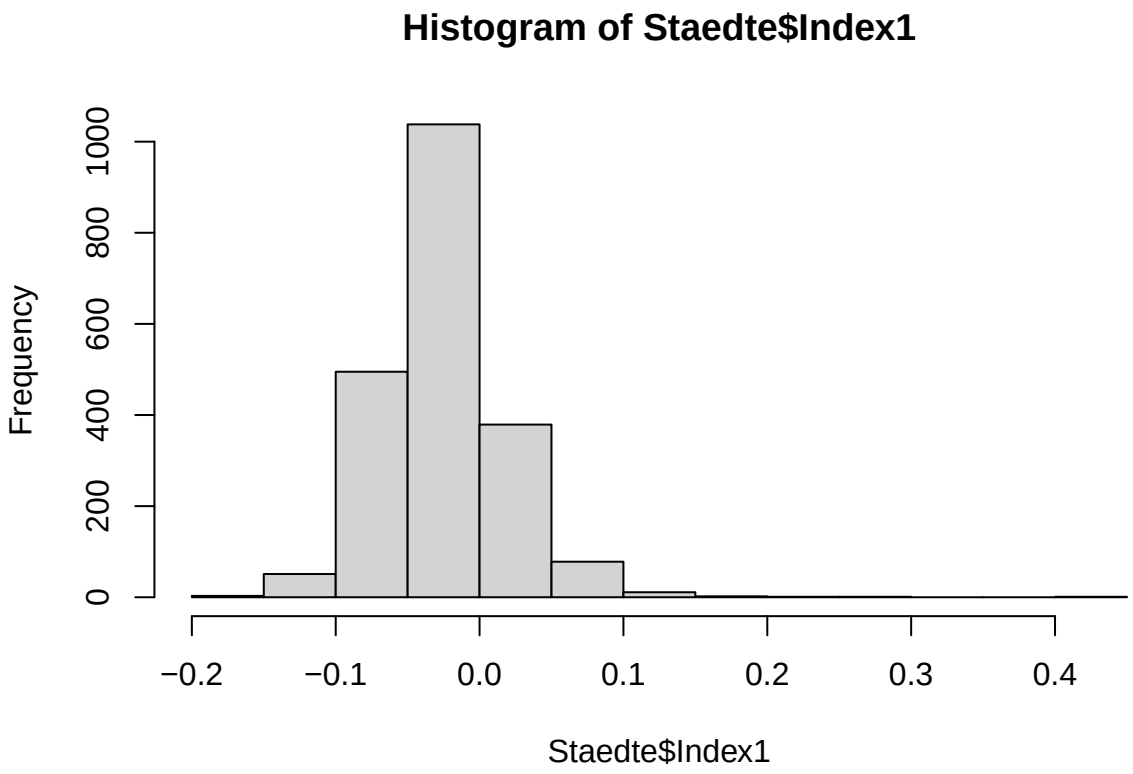
Read data

```
Staedte <- read_delim("https://pzezula.pages.gwdg.de/data/Staedte.txt",
                              delim = "\t")
```

Index creation

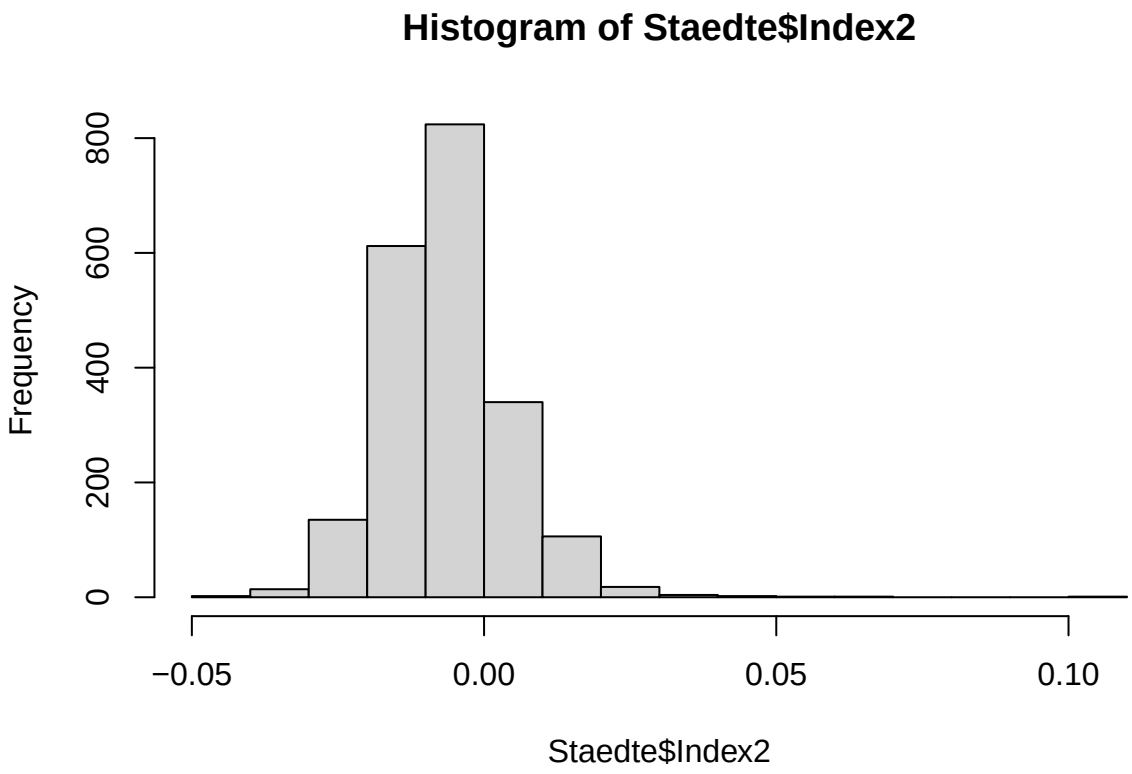
A good starting point is the ratio of the two sexes. Unfortunately, though, it isn't symmetrical. (0.5 is numerically closer to 1 than 2 is, even though both numbers correspond to the same strength of inequality.) The ratio also isn't centered around zero. But there is an easy fix for that: the absolute value of the *logarithm* of 0.5 and 2 is the same, and the logarithm of 1 is zero. Here, we'll do this without `dplyr`:

```
Staedte$Index1 <- log(Staedte$maennlich / Staedte$weiblich)

hist(Staedte$Index1)
```



As an alternative, we could also use the deviation of on sex's SHARE of the population from 50% as our index. Here, we'll use the share of women, so that positive values indicate an elevated male population size, like with `Index1`. This, we'll do in `dplyr`:

```
Staedte <- mutate(Staedte, Index2 = 0.5 -
                    (weiblich / Gesamtbevoelkerung))

hist(Staedte$Index2)
```



Understanding extreme values

The histogram already indicated that extreme values tend to be positive, indicating male preponderance. You can find these values using `View(Staedte)`, then sorting by your index and looking at the first three rows. Of course, R can also do this job for you:

```
Extreme_Staedte <- arrange(Staedte, #takes Staedte data frame and sorts it
                           desc( #in descending order
                             abs( #using the absolute value
                               Index1))) %>% #of Index1
                   slice(1:3) #and only keeps the top three entries.

Extreme_Staedte
```

```
## # A tibble: 3 x 9
##    Rang Bundesland Stadt                                    Flaeche Gesamtbevoelker~ maennlich weibl
##   <dbl>      <dbl> <chr>                                      <dbl>            <dbl>     <dbl>  <d
## 1  2035          6 Schwarzenborn, Stadt                       26.9             1408       848
## 2  2026         16 Hohenleuben, Stadt                          9.53            1537       865
## 3   824          3 Clausthal-Zellerfeld, Berg- und Universi~  43.7            15818      8826     6
```

The three biggest inequalities are in Schwarzenborn, Hohenleuben and Clausthal-Zellerfeld. A web search finds the reasons: All three cities have big employers from sectors with traditionally high male employment: The army in Schwarzenborn, a prison in Hohenleuben and a technical university and mining in Clausthal-Zellerfeld.

Keeping up with the times

`Index1` only uses the ratio of the two sexes, so it's robust with regard to adding a third category. However, `Index2` would lose its validity with the introduction of the third gender. You could however subtract half of the share of that third gender from the index to correct for it - but you would have to know that share to begin with. Example:

Say there's 45% women, 45% men an 10% 'diverse' (a terrible choice of words btw). Then the uncorrected `Index2` would be `0.50 - 0.45 = 0.05`. If you subtract `0.10 / 2 = 0.05`, you'll get zero again, correctly indicating that there are as many men as women.

Bonus task: Zipf's Law

```
Staedte$Zipf_prediction <- (1 / Staedte$Rang) * max(Staedte$Gesamtbevoelkerung)

chisq.test(Staedte$Gesamtbevoelkerung, Staedte$Zipf_prediction)
```

```
## Warning in chisq.test(Staedte$Gesamtbevoelkerung, Staedte$Zipf_prediction): Chi-squared approximatio
```

```
##
##  Pearson's Chi-squared test
##
## data:  Staedte$Gesamtbevoelkerung and Staedte$Zipf_prediction
## X-squared = 4093220, df = 4091233, p-value = 0.2436
```

German cities don't seem to significantly differ from Zipf's Law.

Task 3: Bachelor applications

Read data

Read in the data at `https://pzezula.pages.gwdg.de/data/bsc_bewerbung.txt`. In it, you'll find the "Abitur" (high school diploma) grades from 200 (fictional) applicants for the psychology bachelor at the GEMI. Check out the structure of the data.

German	English
bsc_bewerbung	bachelor application
Nr	no.
Deutsch	German
Englisch	English
ZweiteFremdsprache	SecondForeignLanguage
Mathe	Maths
Bio	biology
Chemie	chemistry
Physik	physics
Geschichte	history
Erdkunde	geography
Kunst	art
Sport	P.E.

Create grade average

Create two new variables for the grade average. In the first, keep the original scaling from 0 to 15 points. For the second, rescale it to classic school grades from 1 to 6. Note: Take care *not* to include the applicant number in the first column into the grade average!

GEMI-Score

At the GEMI, bachelor spots are not distributed by the grade average alone, but by a special weighted average. This contains 80% overall average, an extra 10% for the English grade, and 5% for German and for maths, respectively. Add a new variable for this score. Rescale the score so that its theoretical range is 0 to 100!

Your own Score

Reflect on your own bachelor's study: Which skills were useful? Which pre-existing knowledge helped you? Use these considerations to create your own weighted sum score, again scaling it to 0 to 100. You might want to write down a few points explaining your reasoning.

Admission

Unfortunately, you can only accept 100 of the 200 applicants. Create a new variable for the GEMI score and another for your own score in which the admission or non-admission of the applicants is recorded. The variable-type `logical` is probably best for this.

Comparing the scores

Analyse in how many percent your own score and the GEMI score would lead to different decisions!

Solution

Read data

```
bsc_bewerbung <- read_delim("https://pzezula.pages.gwdg.de/data/bsc_bewerbung.txt",
                            delim = "\t")
```

Create grade average

```
bsc_bewerbung$DurchschnittPunkte <- rowMeans(bsc_bewerbung[,-1])
#[,-1] = don't include first column
#DurchschnittPunkte - PointsAverage

bsc_bewerbung <- mutate(bsc_bewerbung,
                        Durchschnitt_1bis6 =
                          (17 - DurchschnittPunkte) / 3)
```

```
#Durchschnitt_1bis6 - average_1to6
#A correction where zero points yields a 6 is not necessary for our data
```

Create GEMI score

```
bsc_bewerbung$GEMI_score <- ((100 / 15) * #Range 0 to 100
          ((0.80 * bsc_bewerbung$DurchschnittPunkte) +
           (0.10 * bsc_bewerbung$Englisch) +
           (0.05 * bsc_bewerbung$Deutsch) +
           (0.05 * bsc_bewerbung$Mathe)))
```

Own Score

```
bsc_bewerbung$eigener_score <- ((100 / 15) * #eigener_score - own_score
          ((0.10 * bsc_bewerbung$Bio) + #Increasing focus on neuro topics
           (0.10 * bsc_bewerbung$Mathe) + #statistics
           (0.10 * bsc_bewerbung$Englisch) + #Quickly grasp papers
           (0.70 * bsc_bewerbung$DurchschnittPunkte)))
```

Admission

Since we'll admit exactly half of the applicants, we can simply use the median as a cut-off value. Done here for the GEMI score:

```
bsc_bewerbung$Zulassung_GEMI <- bsc_bewerbung$GEMI_score >
                                median(bsc_bewerbung$GEMI_score)
#Zulassung_GEMI - admission_GEMI

sum(bsc_bewerbung$Zulassung_GEMI)
```

```
## [1] 100
```

```
#exactly 100 because there were no two applicants with median scores
```

A more flexible approach that also works with different admission rates would use the `rank()`-function. Done here for my own score:

```
bsc_bewerbung$Zulassung_eigen <- rank(bsc_bewerbung$eigener_score) > 100
#Zulassung_eigen - admission_own

sum(bsc_bewerbung$Zulassung_eigen)
```

```
## [1] 100
```

Comparing the scores

```
bsc_bewerbung$Unterschiedliche_Entscheidung <-
                                    bsc_bewerbung$Zulassung_GEMI !=
                                    bsc_bewerbung$Zulassung_eigen
#Unterschiedliche_Entscheidung - Different_decision

Anzahl_Disagreements <- sum(bsc_bewerbung$Unterschiedliche_Entscheidung)
#Anzahl_Disagreements - Number_of_disagreements

Prozent_Disagreement <- (Anzahl_Disagreements / nrow(bsc_bewerbung)) * 100
#Prozent - percent

Prozent_Disagreement
```

```
## [1] 10
```