Deskriptive Statistik

R für empirische Wissenschaften v1.0.3

Jan Philipp Nolte

Inhaltsverzeichnis

1 Einführung

Deskriptive Statistiken sind wichtig für einen ersten Überblick über die Daten und werden fast immer in Papern oder in Abschlussarbeiten verlangt. Auch wird der Umgang mit fehlenden Werten in diesem Kapitel beschrieben.

2 Deskriptive Statistik

Wir nehmen wieder den Datensatz aus dem letzten Kapitel mit den fünf Persönlichkeitsfaktoren. Das `skimr` Package enthält die Funktion `skim()`, die wir wie gewohnt in einer Pipe ausführen können. Zuerst laden wir das Package.

```
library(skimr)
```

Der Funktion muss kein weiteres Argument übergeben werden, weil alle Spalten analysiert werden. Es werden Informationen über Zeilen- und Spaltenzahl, Anzahl an fehlenden Werten, übliche deskriptive Maße sowie ein Histogram über die Verteilung der jeweiligen Werte ausgegeben. Für einen ersten Überblick optimal. Da der Output im Internet und im PDF nicht richtig abgebildet werden kann, wird der Output hier ausgespart. Aber probiere es selbst einmal.

```
big_five %>%
  skim()
```

Natürlich können wir auch hier mit `group_by()` den Datensatz in die jeweiligen Untergruppen unterteilen.

```
big_five %>%
  group_by(Geschlecht) %>%
  skim()
```

Schauen wir uns den Output einmal genauer an. Was ist denn da mit dem Alter los? Maximales Alter ist 1995? Da hat wohl jemand sein Geburtsjahr und nicht sein Alter in Jahren angegeben. Schauen wir mal, ob mehrere unrealistische Alter angegeben wurden.

```
big_five %>%
  filter(Alter > 150)
```

```
## # A tibble: 2 x 8
##   Alter Geschlecht Herkunft Extraversion Neurotizismus Vertraeglichkeit
##   <dbl> <chr>      <chr>           <dbl>         <dbl>            <dbl>
## 1  1964 f          JO                3.2           2.3              3.4
## 2  1995 f          PH                2.5           3.7              3.8
## # ... with 2 more variables: Gewissenhaftigkeit <dbl>, Offenheit <dbl>
```

Wie man sieht, haben nur zwei Personen das Eintragen falsch verstanden. Ändern wir das mit der bereits gelernten `if_else()` Funktion kurz ab. Für alle Personen, die ein Alter größer als

150 eingetragen haben, wird 2018 minus das angegebene Geburtsjahr gerechnet und für die Restlichen nur das richtig eingetragene Alter zurückgegeben.

```
big_five <- big_five %>%
  mutate(Alter = if_else(Alter > 150, 2018 - Alter, Alter))
```

Wir merken sofort, wie eng die Datenvorbereitung mit der deskriptiven Statistik verknüpft ist. In der Regel merkt man erst beim Erkunden des Datensatzes, was noch alles verändert werden muss, damit man die geplanten Rechnung durchführen kann.

Es gibt natürlich auch eine Funktion aus dem `tidyverse` für die deskriptive Statistik. Möchten wir die Anzahl der Personen, den Mittelwert des Alters, deren Standardabweichung und Standardfehler ausgegeben haben, benutzen wir `summarise()`

```
big_five %>%
  summarise(N = n(),
            Mittelwert = mean(Alter),
            SD = sd(Alter),
            SE = SD / sqrt(N))
```

```
## # A tibble: 1 x 4
##       N Mittelwert    SD    SE
##   <int>      <dbl> <dbl> <dbl>
## 1   200       26.5  11.3 0.801
```

mit `n()` für die Anzahl, `mean()` für den Mittelwert, `sd()` für die Standardabweichung und `sqrt()` für die Quadratwurzel. Für die selben Statistiken pro Geschlecht, benutzen wir wieder `group_by()`.

```
big_five %>%
  group_by(Geschlecht) %>%
  summarise(N = n(),
            Mittelwert = mean(Alter),
            SD = sd(Alter),
            SE = SD / sqrt(N))
```

```
## # A tibble: 2 x 5
##   Geschlecht     N Mittelwert    SD    SE
##   <chr>      <int>      <dbl> <dbl> <dbl>
## 1 f            118       25.9  11.3  1.04
## 2 m             82       27.3  11.3  1.25
```

Nun kommen wir zur alles entscheidenen Frage. Wer hat denn nun die Tipprunde gewonnen? Zur schöneren Ausgabe benutzen wir die bereits kennengelernte `arrange()` Funktion.

```
tipp_wm %>%
  group_by(Person) %>%
  summarise(Gesamtpunktzahl = sum(Punkte)) %>%
  arrange(desc(Gesamtpunktzahl))
```

```
## # A tibble: 6 x 2
##   Person       Gesamtpunktzahl
##   <chr>                  <dbl>
## 1 Alice                     50
## 2 Bob                       49
## 3 Charlie                   48
## 4 Thomas_Bayes              48
## 5 Jerzy_Neyman              40
## 6 Egon_Pearson              33
```

Alice hat gewonnen! Juhu!

2.1 Umgang mit fehlenden Werten

Was aber nun, wenn nicht alle Werte vorhanden sind? Im Datensatz zur Tipp WM haben manche Tipper teilweise vergessen, ihren Tipp abzugeben. Solche fehlenden Werte nennt man NAs (not available). NAs können zu Fehlermeldungen führen. Es gibt verschiedene Möglichkeiten mit NAs umzugehen. Welche man verwendet, hängt stark vom Kontext ab.

Möchte man zum Beispiel nur den Mittelwert einer Variable berechnen, kann man bei der Funktion `mean()` das Argument `na.rm` auf `TRUE` setzen.

```
tipp_wm %>%
  summarise(Mittelwert = mean(Punkte, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   Mittelwert
##        <dbl>
## 1      0.698
```

Alternativ kann man mit `na.omit()` auch sämtliche NAs im ganzen Datensatz entfernen. Aber Vorsicht, es werden alle Zeilen entfernt, in denen auch nur ein einziger Wert fehlt. Angenommen man möchte sich zwei untersuchte Variablen genauer anschauen und in den beiden Spalten selbst fehlen kaum Werte. Es kann aber gut sein, dass Du ausserdem eine sozio-demographische Spalte (zum Beispiel Alter) erhoben hast. In dieser Spalte fehlen nun einige Werte. Für die Untersuchung Deiner beiden Variablen spielt diese Spalte aber gar keine Rolle. Die Funktion würde trotzdem alle Variablen entfernen. Im Datensatz `tipp_wm_wide` würden von 64 Zeilen lediglich 47 übrig bleiben.

```
tipp_wm_wide %>%
  na.omit()
```

```
## # A tibble: 47 x 7
##    Thomas_Bayes Egon_Pearson Jerzy_Neyman Alice Bob   Charlie Ergebnis
##    <chr>        <chr>        <chr>        <chr> <chr> <chr>   <chr>
##  1 2:0          1:0          2:1          2:1   3:1   2:1     5:0
##  2 1:3          0:0          0:2          1:2   0:2   0:2     0:1
```

```
##  3 0:0         0:2         1:1         1:1   2:1   1:1     0:1
##  4 1:2         1:1         2:1         2:4   2:2   1:0     3:3
##  5 0:2         1:2         0:2         1:2   2:1   1:2     0:1
##  6 4:0         1:2         3:0         1:0   3:0   4:0     2:0
##  7 2:1         0:0         1:1         1:1   0:1   1:0     0:1
##  8 3:0         1:0         2:0         3:1   3:1   2:0     0:1
##  9 2:1         2:0         1:1         2:1   2:0   3:1     1:1
## 10 1:1         1:0         1:0         2:1   1:0   2:1     1:0
## # ... with 37 more rows
```

Sei Dir dessen also bewusst. Die meisten induktiven Verfahren regeln NAs allerdings von selbst.

2.2 Weitere Funktionen

Andere nicht explizit besprochene Funktionen können nach dem selben Schema wie beispielsweise `mean()` und `sd()` verwendet werden.

- `quantile()`
- `var()`
- `median()`
- `min()`
- `max()`