Inhaltsverzeichnis

Die Deskriptivstatistik ist ein wichtiger Bestandteil einer statistischen Analyse. In den Kapiteln zur uni- und bivariaten Deskriptivstatistik werden Sie Werkzeuge kennenlernen, die R Ihnen bietet, um Daten durch Tabellen, Kennzahlen oder Grafiken zusammenzufassen und darzustellen. Die univariate Deskriptivstatistik bezieht sich dabei auf das Beschreiben und Zusammenfassen einzelner Variablen innerhalb eines Datensatzes.

1 Nominal- und ordinalskalierte Variablen

StatsReminder

Eine Variable (Merkmal) besitzt ein sog. **Skalenniveau**. Das Skalenniveau bestimmt, welche Analysemethoden auf diese Variable angewendet werden dürfen. Die wichtigsten Skalenniveaus sind die Nominal-, Ordinal-, Intervall- und die Verhältnisskala, wobei Variablen, die intervall- oder verhältnisskaliert durch den Begriff *metrisch* zusammengefasst werden.

Eine nominalskalierte Variable wäre bspw. das Geschlecht. In einem Datensatz können die Kategorien der Geschlechtsvariable numerisch mit 0, 1 und 2 kodiert werden (0 für männlich, 1 für weiblich und 2 für divers). In R können wir Häufigkeitstabellen für nominalskalierte Variablen erstellen!
Eine ordinalskalierte Variable folgt einer Rangreihe. Ordinalskalierte Daten könnten bspw. Schulnoten oder ein Item eines Fragebogens (Likert-Skala) sein.

1.1 Kennwerte der zentralen Tendenz ab Nominalskalenniveau

Kennwerte der zentralen Tendenz sind sogenannte Lagemaße, welche das Zentrum einer Verteilung beschreiben (anhand eines Wertes quantifizieren).

1.1.1 Modus

Der Modus (oder Modalwert) ist die Merkmalsausprägung, die in den Daten am häufigsten vorkommt. Da nominalskalierte Daten den niedrigsten Informationsgehalt haben, kann der Modus auch für Daten anderer Skalenniveaus berechnet werden. In R kann man sich den Modus mithilfe von `table()` ausgeben lassen:

```
load("dat/erstis_neu.RData")    # laden von Daten
table(erstis$wohnort.alt)       # erstellen einer Häufigkeitstabelle
```

```
alte BL neue BL  Berlin Ausland
     32      26      86      18
```

- diese Häufigkeitstabelle ist eine **absolute** Häufigkeitstabelle; die absolute Häufigkeitstabelle gibt wieder, wie häufig die verschiedenen Ausprägungen der Variable im Datensatz vorkommen
- in der *Console* kann man nun sehen, wie viele der Teilnehmenden an welchem Standort vor 12 Monaten gelebt haben
  - hier lässt sich nun einfach und schnell der Modus ablesen (hier: Berlin)

Alternativ kann das Paket `DescTools` verwendet werden. Hier ist der Befehl zum Berechnen des Modus direkt implementiert:

```
library(DescTools)
```

```
Mode(erstis$wohnort.alt, na.rm = T)
```

```
[1] Berlin
attr(,"freq")
[1] 86
Levels: alte BL neue BL Berlin Ausland
```

- im Gegensatz zu `table()` erhalten Sie hier lediglich die am häufigsten besetzte Kategorie
- damit Sie den korrekten Modus erhalten, muss `Mode` zusätzlich das Argument `na.rm = T` übergeben werden
- beim `table`-Befehl ist das per Voreinstellung (`useNA = "no"`) geregelt!

1.2 Kennwerte der zentralen Tendenz ab Ordinalskalenniveau

1.2.1 Median

Der Median (Zentralwert) ist der Wert, der die Beobachtungen innerhalb einer Stichprobe in zwei gleich große Hälften teilt, also der Median befindet sich genau in der *Mitte* der Daten. Der Median setzt **ordinalskalierte** Daten voraus und ist damit **nicht** auf das obige Wohnortbeispiel anwendbar. Wir berechnen den Median daher für die Variable `lz13` (*"In den meisten Bereichen entspricht mein Leben meinen Idealvorstellungen."; 1 = stimme gar nicht zu - 7 = stimme völlig zu*):

```
median(erstis$lz13, na.rm = TRUE)
```

```
[1] 5
```

- Der Wert, der die Variable in zwei Hälften teilt, ist 5
- Der Median ist gleich dem 50. Quantil und kann dementsprechend interpretiert werden
  - Mindestens 50 % der Studierenden haben einen Wert von 5 oder kleiner angekreuzt.

1.2.2 Quantil

Quantile können als Verallgemeinerung des Medians verstanden werden und dienen dazu, sich ein genaueres Bild über die Lage einer Verteilung zu machen. **Zur Erinnerung**: Ein p-Quantil ist ein Wert x_p $(0 < p < 1)$, für den gilt, dass mindestens ein Anteil $p \cdot 100\%$ der Daten kleiner oder gleich x_p und mindestens ein Anteil $(1 - p) \cdot 100\%$ der Daten größer oder gleich x_p ist. So ist beispielsweise der Median das 50. Quantil (auch Perzentil), da der Median die Daten genau dort teilt, wo 50 % der Werte unter und über dem Median liegen.

Ein Quartil ist ein weiteres wichtiges Quantil, welches die Verteilung einer Variable in vier gleich häufig besetzte Kategorien unterteilt. In R kann man mit `quantile()` beliebige Quantile bestimmen:

```
quantile(erstis$lz13, type = 5, na.rm = TRUE)
```

```
  0%  25%  50%  75% 100%
   1    4    5    6    7
```

- standardmäßig teilt die `quantile()`-Funktion die Daten in vier Abschnitte ein, also in sogenannte Quartile
- Interpretation des 75 % Quantils (3. Quartils): mindestens 75 % der Studierenden haben einen Wert von 6 oder kleiner angekreuzt
- das `type` Argument legt fest, welche Formel zur Berechnung der Quantile genutzt wird
  - `type = 5` entspricht der Formel aus der Vorlesung

Mit Quartilen lässt sich außerdem der empirische Interquartilsbereich, ein Streuungsmaß, bestimmen. Der Interquartilsbereich gibt die Breite des Intervalls der mittleren 50 % der Personen im Datensatz an. In unserem Beispiel erhalten wir einen empirischen Interquartilsbereich von $x_{0.75} - x_{0.25} = 6 - 4 = 2$

Mit dem `probs` Argument lassen sich die Quantile manuell festlegen:

```
quantile(erstis$lz13, type = 5, na.rm = TRUE, probs = c(.1, .9))
```

```
10% 90%
  3   6
```

- mindestens 10 % der Studierenden haben einen Wert von 3 oder kleiner angekreuzt

1.3 Häufigkeitstabellen für Rangklassen

Genau wie für nominalskalierte Variablen lassen sich Häufigkeitstabellen auch für ordinalskalierte Variablen mit dem `table()`-Befehl ausgeben:

```
absolut <- table(erstis$lz13)  # absolute Häufigkeiten
absolut
```

```
 1  2  3  4  5  6  7
 4  9 19 34 53 60 10
```

Um die Daten als relative Häufigkeiten darzustellen, wird das Objekt `absolut` (also die Tabelle der absoluten Häufigkeiten) an die `prop.table()`-Funktion übergeben:

```
relativ <- prop.table(absolut) # relative Häufigkeiten
relativ
```

```
         1          2          3          4          5          6          7
0.02116402 0.04761905 0.10052910 0.17989418 0.28042328 0.31746032 0.05291005
```

- das Objekt `relativ` beinhaltet die relativen Häufigkeiten
  - relative Häufigkeiten beschreiben die Anteile der Fragebogenteilnehmenden in den verschiedenen Kategorien im Verhältnis zu allen Studierenden

```
prozent <- 100 * relativ       # Tabelle mit Prozentangaben
prozent
```

```
        1         2         3         4         5         6         7
 2.116402  4.761905 10.052910 17.989418 28.042328 31.746032  5.291005
```

- damit wir die relativen Häufigkeiten direkt als Prozente interpretieren können, müssen wir lediglich das Objekt `relativ` mit 100 multiplizieren

```
kumuliert <- cumsum(prozent)
kumuliert
```

```
        1         2         3         4         5         6         7
 2.116402  6.878307 16.931217 34.920635 62.962963 94.708995 100.000000
```

- mit `cumsum()` kann man die kumulierten Summen der einzelnen Elemente bilden
- an dieser Tabelle können wir die kumulierten Prozentwerte ablesen; z. B., dass nur 6,88 % aller Studierenden in Kategorie 1 oder 2 geantwortet, und dass ca. 34,92 % aller Studierenden in Kategorie 4 oder einer kleineren Kategorie geantwortet hat.

```
tabelle <- cbind(absolut, relativ, prozent, kumuliert)
tabelle
```

```
  absolut    relativ   prozent  kumuliert
1       4 0.02116402  2.116402   2.116402
2       9 0.04761905  4.761905   6.878307
3      19 0.10052910 10.052910  16.931217
4      34 0.17989418 17.989418  34.920635
5      53 0.28042328 28.042328  62.962963
6      60 0.31746032 31.746032  94.708995
7      10 0.05291005  5.291005 100.000000
```

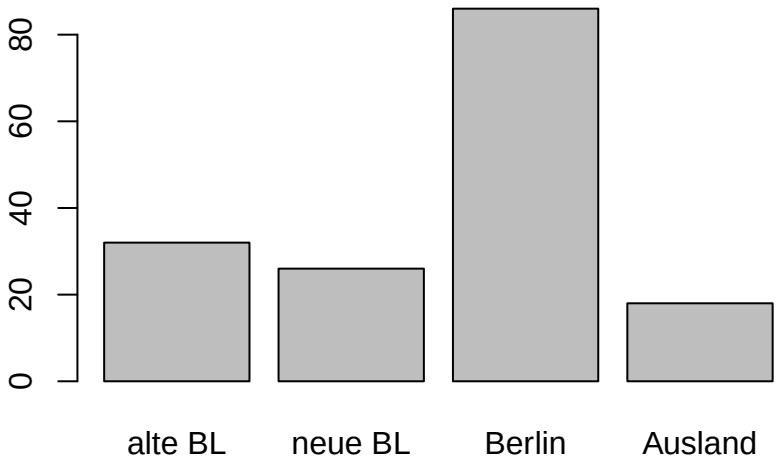
- mit `cbind()` lassen sich die Tabellen in einem Data Frame zusammenfassen
  - `cbind()` steht für *columnbind*, d. h. die verschiedenen Objekte die `cbind()` als Argumente übergeben werden, werden spaltenweise aneinandergefügt

2 Grafiken

2.1 Nominalskalierte Variablen

Zur Darstellung von absoluten Häufigkeiten in einem Säulendiagramm kann man den `barplot()`-Befehl verwenden.

```
barplot(table(erstis$wohnort.alt))
```



- der `barplot()`-Befehl verlangt nach den entsprechenden Daten (d. h., absoluten oder relativen Häufigkeiten pro Ausprägung) im Tabellenformat
  - daher muss für die entsprechende Variable (hier: `wohnort.alt`) zuerst mithilfe von `table` eine Häufigkeitstabelle erstellt werden
  - nur mit dieser Tabelle kann dann mit `barplot` ein Säulendiagramm dargestellt werden

Alternativ kann die Tabelle erst einem Objekt zugewiesen werden und dieses Objekt dann `barplot` übergeben werden:

```
hf_tab <- table(erstis$wohnort.alt)
barplot(hf_tab)
```

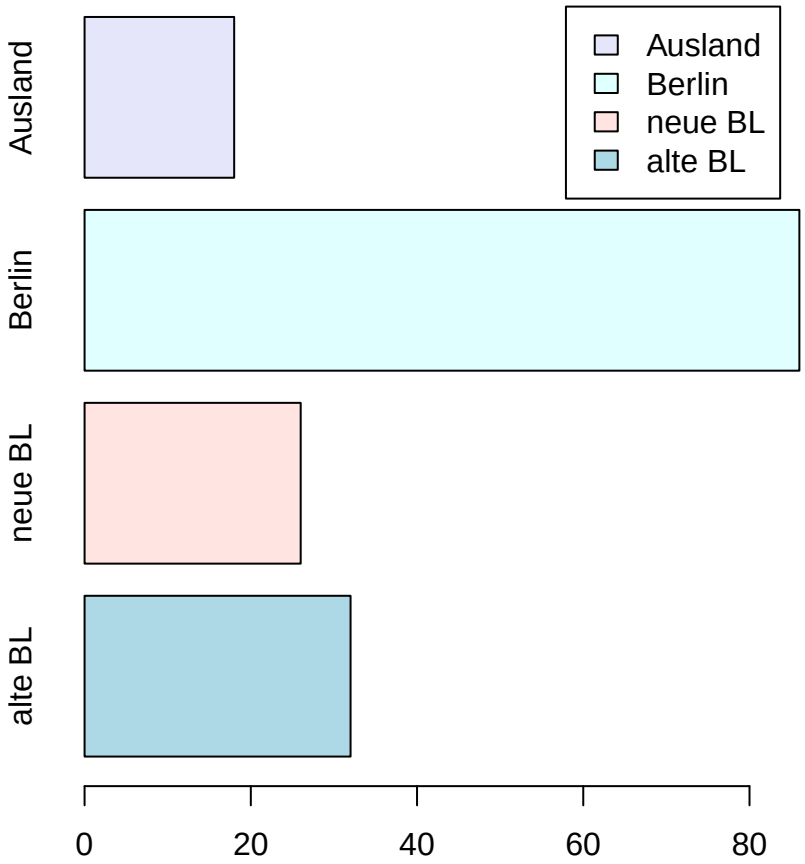
Es gibt eine Vielzahl an Möglichkeiten, Grafiken anschaulicher zu machen und zu individualisieren. So kann man mit dem `horiz` Argument die Achsen des Säulendiagramms tauschen oder mit `col` die Balken einfärben, um die Kategorien farblich zu unterscheiden. Mit `legend = TRUE` ist es möglich, sich eine Legende ausgeben zu lassen. R erstellt dann automatisch einen kleinen Kasten, der die verschiedenen Kategorien mit den benutzten Farben identifiziert.

```
barplot(table(erstis$wohnort.alt),
        horiz = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan", "lavender"),
        legend = TRUE)
```

- für das Tauschen der Achsen muss das `horiz`-Argument auf `TRUE` gesetzt werden
- um die Balken einzufärben muss das `col`-Argument verwendet werden
  - dem Argument wird ein Vektor (`c()`) übergeben, der die erwünschten Farbnamen beinhaltet
  - hierbei muss beachtet werden, dass die Balken in derselben Reihenfolge eingefärbt werden, wie sie dem Argument übergeben werden (probiert es einmal selber aus: ersetzt `"lavender"` durch

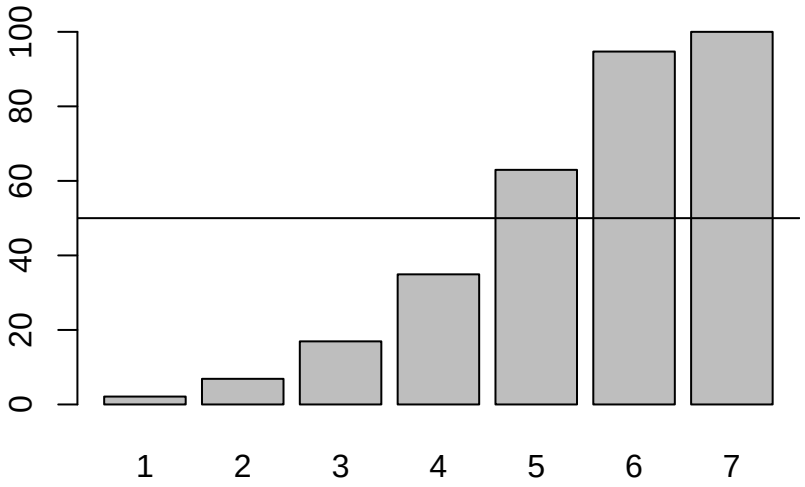
`"red"`, was könnt Ihr beobachten?)

- mit `legend = TRUE` wird automatisch eine Legende für die Grafik erstellt



2.2 Ordinalskalierte Variablen

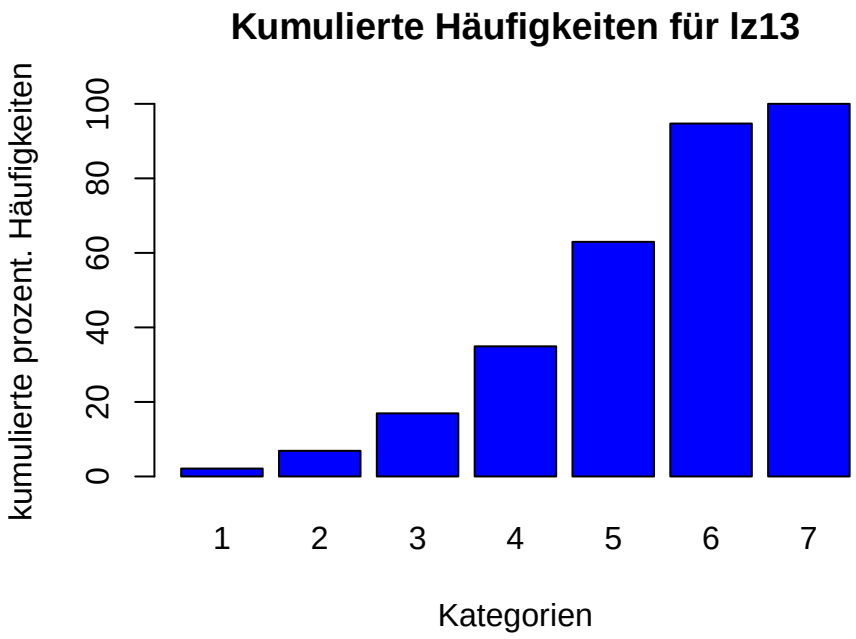
```
barplot(kumuliert)
abline(h = 50)
```



- mit `abline()` lassen sich Geraden in eine bereits offene Grafik hineinzeichnen.
  - `h` steht hier für einen Wert auf der y-Achse; in unserem Beispiel zeichen wir eine horizontale Linie auf dem Wert `50` (`h = 50`)
  - für eine vertikale Linie würde man das Argument `v` nutzen
  - für das Zeichnen von Geraden mit einer Steigung $\neq 0$, kann man die Argumente `a` und `b` verwenden. Wobei `a` für den y-Achsenabschnitt steht und `b` für die Steigung
  - wichtig: `a` und `b` müssen immer gemeinsam spezifiziert werden

In R lassen sich Grafiken mit einer Palette von Argumenten anpassen.
Ein Beispiel:

```
barplot(kumuliert,
        main = "Kumulierte Häufigkeiten für lz13",
        xlab = "Kategorien",
        ylab = "kumulierte prozent. Häufigkeiten",
        col = "blue")
```



- `main`: Titel der Grafik
- `xlab`: Bezeichnung der x-Achse
- `ylab`: Bezeichnung der y-Achse
- `col`: Farbe der Säulen

3 Metrische Variablen

Zusätzlich zu den Lage- und Streuungsmaßen, die schon ab Nominalskalen- bzw. ab Ordinalskalenniveau sinnvoll bestimmt werden können, können bei metrischen Variablen weitere Kennwerte berechnet werden.

3.1 Kennwerte der zentralen Tendenz

3.1.1 Arithmetischer Mittelwert

Das arithmetische Mittel ist das gebräuchlichste Maß der zentralen Tendenz bei metrischen Variablen. Es wird berechnet, indem man die Summe der beobachteten Merkmalswerte x_i durch die Anzahl der Beobachtungen n teilt:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Die Berechnung des arithmetischen Mittelwerts setzt voraus, dass das untersuchte Merkmal mindestens intervallskaliert ist.

```
mean(erstis$lz.1, na.rm = TRUE)
```

```
[1] 24.53439
```

3.1.2 Minimum & Maximum

Das Minimum ist der kleinste Wert in einer Datenreihe; das Maximum der größte Wert.

In R kann man das Minimum und Maximum wie folgt ausgeben lassen:

```
min(erstis$lz13, na.rm = TRUE)
```

```
[1] 1
```

```
max(erstis$lz13, na.rm = TRUE)
```

```
[1] 7
```

Alternativ kann man den `range()`-Befehl nutzen:

```
range(erstis$lz13, na.rm = TRUE)
```

```
[1] 1 7
```

3.2 Streuungsmaße (oder Dispersionsmaße)

Kennwerte der zentralen Tendenz können für verschiedene Variablen ähnliche oder sogar gleiche Werte annehmen. Aber heißt das auch, dass die *Verteilungen* dieser Variablen ähnlich oder gleich sein müssen? Beispielsweise wäre es möglich, dass die Variablen "Wohnort vor 12 Monaten" und "Wohnort jetzt" denselben Modus haben. Das ist aber kein Indikator dafür, dass die Ausprägungen der Merkmalsträger sich bei beiden Variablen gleich über die Kategorien verteilen. Das gilt natürlich auch für alle anderen Lagemaße! Nur weil die Verteilungen von zwei Variablen den gleichen Median oder arithmetischen Mittelwert aufweisen, heißt das nicht, dass die Verteilungen identisch aussehen. Streuungsmaße geben Aufschluss darüber, wie die Daten um den Median oder arithmetischen Mittelwert streuen, d. h., wie sich die Beobachtungen über die verschiedenen Kategorien verteilen / wie große die Unterschiedlichkeit der Ausprägungen über die Merkmalsträger ist.

3.2.1 Varianz & Standardabweichung

Die Varianz ist ein Streuungsmaß für metrische Variablen. In der Vorlesung wurde die empirische Varianz s_x^2 vorgestellt, welche den quadrierten Abstand jedes Werts zum arithmetischen Mittelwert berechnet. Die Formel, welche in R verwendet wird, unterscheidet sich leicht von der in der Vorlesung vorgestellten Formel. Die Gründe dafür werden Sie erst später im Semester erfahren. Im Augenblick reicht es, wenn sie die R-Befehle zur Bestimmung der empirischen Varianz nutzen.

```
var(erstis$lz.1, na.rm = TRUE)   # geschätzte Populationsvarianz
```

```
[1] 31.55865
```

```
sd(erstis$lz.1, na.rm = TRUE)    # geschätzte Populationsstandardabweichung
```

```
[1] 5.617709
```

```
IQR(erstis$lz.1, na.rm = TRUE, type = 5)   # Interquartilsabstand
```

```
[1] 8
```

- der Interquartilsabstand kann auch mit der Funktion `IQR` berechnet werden
  - Achtet bei der Berechnung auf die Benutzung der richtigen Formel, um gleich Ergebnisse wie in der Vorlesung zu erhalten (`type` Argument)

3.2.2 Spannweite

Mit dem Streubereich lässt sich ein weiteres Streuungsmaß berechnen, die Spannweite:

$$R = x_{max} - x_{min}$$

```
max(erstis$lz13, na.rm = TRUE) - min(erstis$lz13, na.rm = TRUE)
```

```
[1] 6
```

3.3 describe()

Im psych Paket gibt eine nützliche Funktion zur Ausgabe einer Vielzahl von Deskriptivstatistiken. Mit describe() lassen sich alle bisher besprochenen Statistiken in einer Zeile ausgeben:

```
# library(psych)
describe(erstis$lz.1, IQR = FALSE)
```

```
   vars   n  mean   sd median trimmed  mad min max range  skew kurtosis   se
X1    1 189 24.53 5.62     26   24.94 5.93   9  35    26 -0.57    -0.27 0.41
```

- n: Anzahl gültiger Fälle auf der Variable
- mean: arithmetisches Mittel
- sd: (Geschätzte Populations-)standardabweichung
- trimmed: getrimmtes Mittel (Voreinstellung: 0.1; das bedeutet, dass vor der Berechung des arithmetischen Mittels, 10 % der Daten an beiden *Rändern* abgeschnitten werden)
- min: Minimum
- max: Maximum
- range: Range
- skew: Schiefe
- kurtosis: Wölbung
- IQR: Interquartilsabstand

3.4 z-Transformation

Mit scale() lassen sich Variablen z-transformieren. Die daraus resultierenden z-Werte haben einen Mittelwert von 0 und eine Standardabweichung von 1.

```
lz.1_z <- scale(erstis$lz.1)

# Kontrolle: Ist Mittelwert = 0 und Standardabweichung = 1?
mean(lz.1_z,  na.rm = TRUE)
```

```
[1] -8.459301e-17
```

```
sd(lz.1_z, na.rm = TRUE)
```

```
[1] 1
```

Falls man die Variable nur zentrieren möchte, kann man dies händisch machen. Dafür ziehen von jedem Wert der Variable lz.1 den Mittelwert von lz.1 ab und speichern das Resultat in einem neuen Objekt lz.1_zent:

```
lz.1_zent <- erstis$lz.1 - mean(erstis$lz.1, na.rm = TRUE)
mean(lz.1_zent, na.rm = TRUE)
```

```
[1] -4.697705e-16
```

Etwas einfacher geht es auch mit der scale()-Funktion:

```
lz.1_zent2 <- scale(erstis$lz.1, scale = FALSE)
mean(lz.1_zent2, na.rm = TRUE)
```

```
[1] -4.697705e-16
```

- mit scale = FALSE verhindern wir, dass die Differenz der Werte und dem Mittelwert durch die Standardabweichung geteilt wird

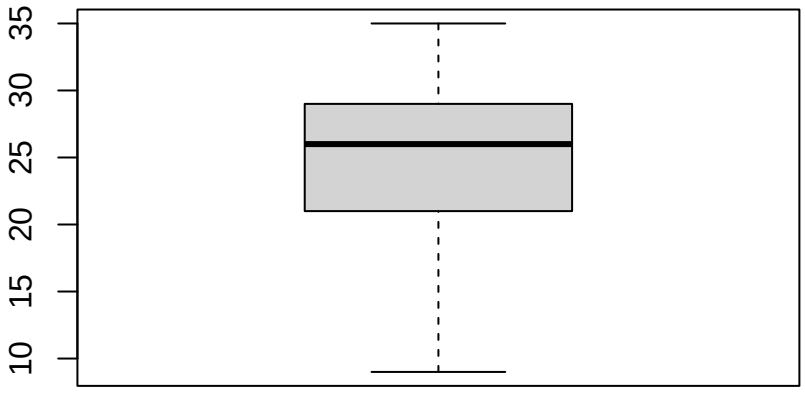
4 Grafiken

4.1 Metrische Variablen

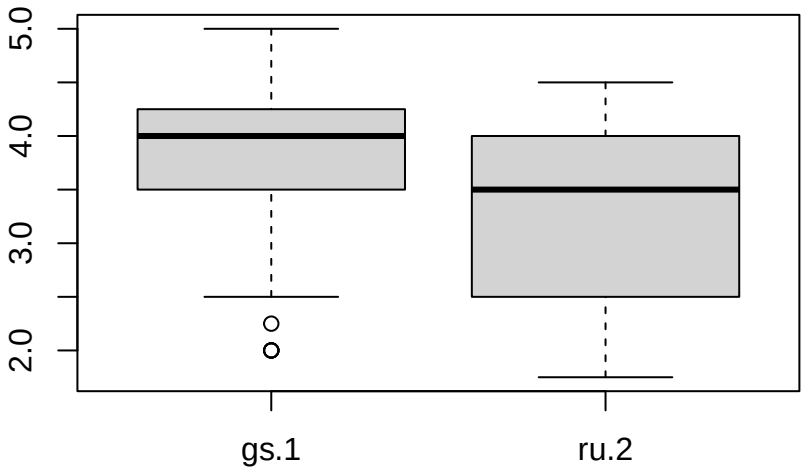
Für metrische Variablen können sogenannte Box-Plots erstellt werden. Der Box-Plot fasst verschiedene Lage- und Streuungsmaße in einer Grafik zusammen. Die Box entspricht immer dem Bereich, in dem die mittleren 50 % der Daten liegen. Die Höhe der Box entspricht dem Interquartilsabstand (engl., *interquartile range*, IQR), welcher als die Differenz zwischen dem 3. und 1. Quartil definiert ist. Der fett gedruckte Strich in der Box entspricht meist dem Median; manchmal wird auch das arithmetische Mittel anstatt des Medians abgetragen. Die "Whisker" oder Antennen stellen Werte dar, die außerhalb der Box liegen. Für die Berechnung der Länge der Whiskers wird meist der IQR mit 1,5 multipliziert. In der Regel wird kein Datenpunkt direkt auf das Ende des Wishkers fallen. Die Whiskers werden dann soweit verkürzt, bis ein Datenpunkt gefunden wurde. Punkte außerhalb der Whiskers werden als Punkte in der Grafik abgetragen und als Ausreißer deklariert. In unserem Beispiel gibt es keine Ausreißer:

Box-Plot in R:

```
boxplot(erstis$lz.1)
```



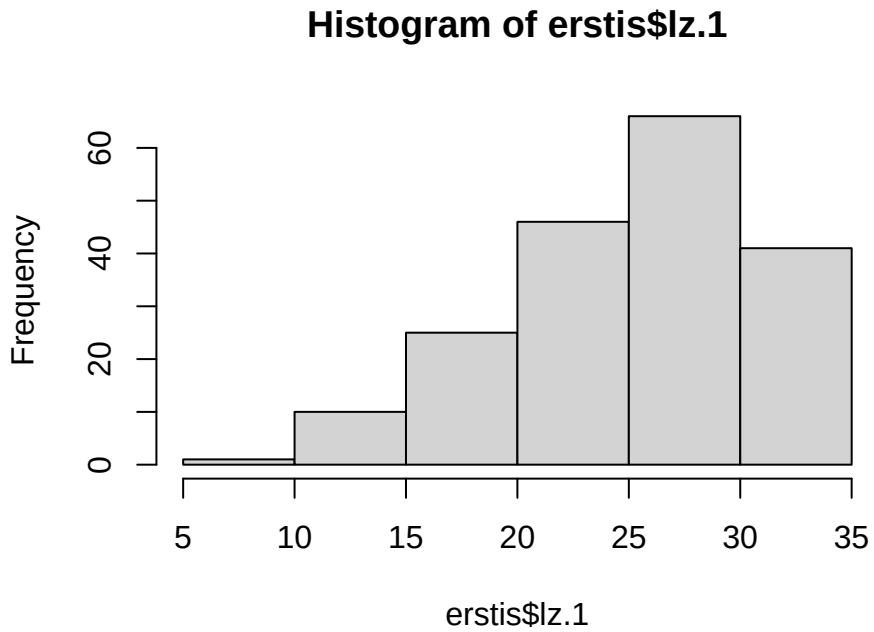

```
boxplot(erstis[, c("gs.1","ru.2")])  # es lassen sich mehrere Box-Plots in einer Grafik darstellen
```



- **Achtung**: Der Vergleich von Box-Plots macht nur Sinn, wenn die Variablen auch den gleichen Wertebereich und die gleiche Skalierung haben

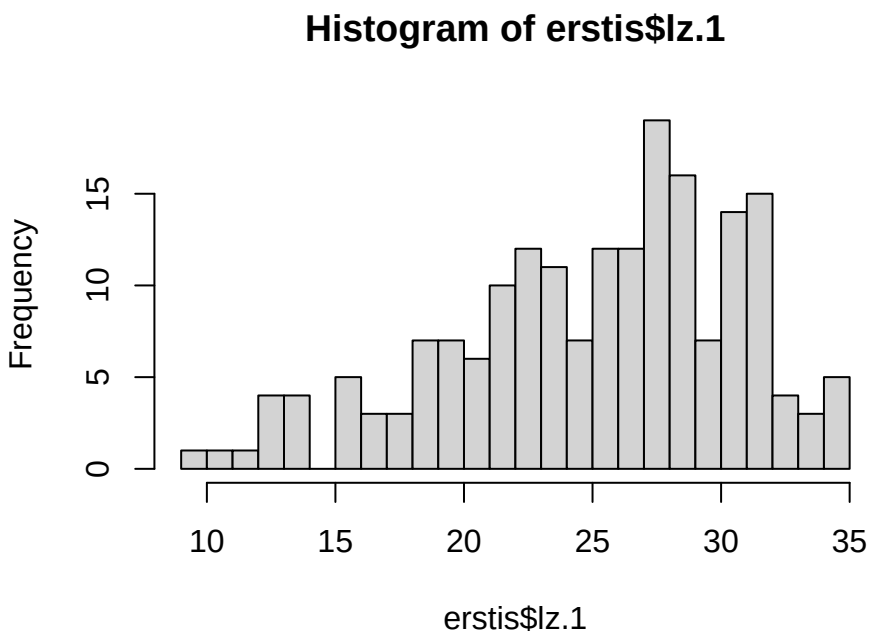
Ein Histogramm ist eine grafische Darstellung von metrischen Variablen, für welche die Werte der metrischen Variablen zunächst in Kategorien / Intervallen zusammengefasst werden. Die Darstellung ist der eines Säulendiagramms nicht unähnlich, allerdings ist im Vergleich zu nominalskalierten Variablen die Kategorisierung nicht natürlich. Hier wird vom Nutzer (oder der Voreinstellung des Programms) eine Einteilung der Daten in sogenannte "bins" festgelegt. Es liegt bei der Erstellung eines Histogramms immer auch ein Informationsverlust vor.

```
hist(erstis$lz.1, right = FALSE)
```



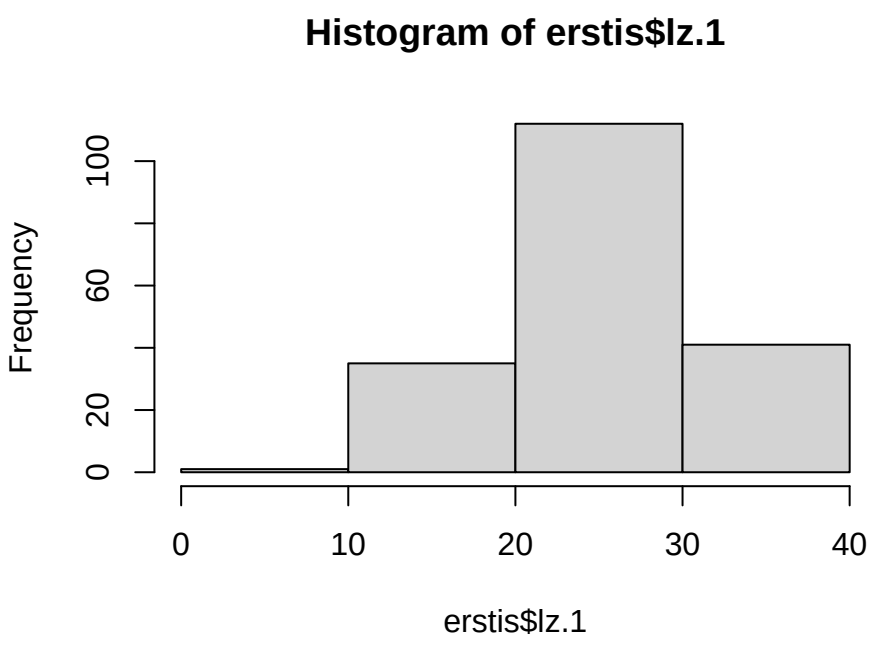
- `right = FALSE` führt zu linksschließenden Intervallen [a,b) (siehe Vorlesung)

```
hist(erstis$lz.1,
     right = FALSE,
     breaks = 30)
```



- `breaks` dient als Vorschlag des Nutzers für die Anzahl der Kategorien (*bins*)
- die endgültige Zahl an bins bleibt R überlassen
  - falls hier kein Wert gesetzt wird, versucht R den *besten* Wert, also die beste Anzahl an bins, zu finden

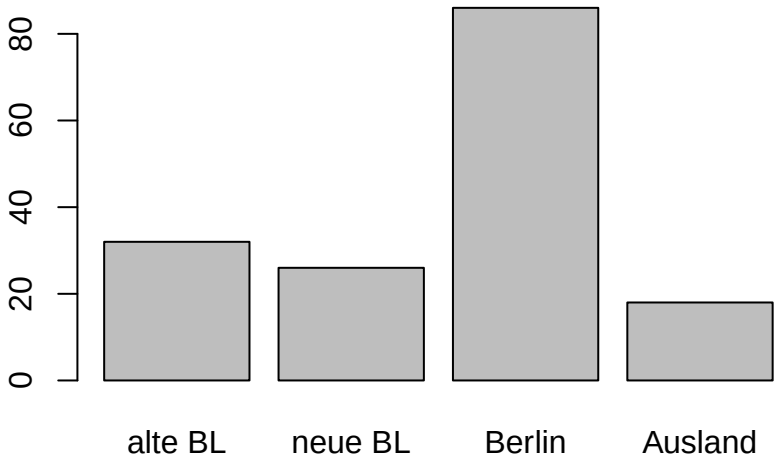
```
hist(erstis$lz.1,
     right = FALSE,
     breaks = c(0, 10, 20, 30, 40))
```



- wenn dem `breaks` Argument ein Vektor mit Werten übergeben wird, dann werden anstatt der Anzahl der *bins* die Intervallgrenzen auf der Variable selbst festgelegt

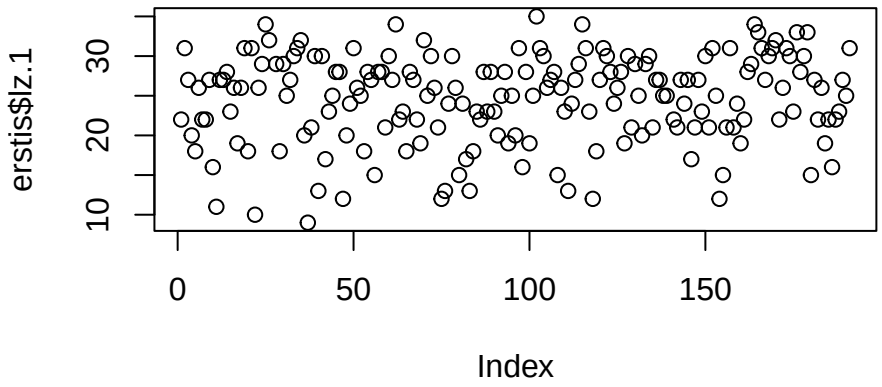
`plot()` ist eine allgemeine Funktion für das Erstellen von Grafiken. *Allgemein* heißt hier, dass `plot()` für das Erstellen von einer Vielzahl von Grafiken eingesetzt werden kann. In R werden solche Funktionen *generisch* genannt. Sie können je nach Typ des übergebenen Objekts erkennen, welche Grafik (Streudiagramm, Box-Plot etc.) am geeignetsten ist. Wenn wir beispielsweise einen Faktor an `plot()` übergeben, dann folgert R, dass ein Säulendiagramm eine gute grafische Darstellung der Daten ist:

```
plot(erstis$wohnort.alt)
```



Bei der Übergabe einer metrischen Variable erstellt R einen sogenannten Indexplot:

```
plot(erstis$lz.1)
```



- auf x-Achse sind die Personen abgetragen

5 Übersicht

5.1 Neue wichtige Konzepte

- **Erstellen von Univariaten Deskriptivstatistiken passend zum Skalenniveau**
- **Erstellen von Grafiken passend zum Skalenniveau**
- **z-Transformation**

5.2 Neue wichtige Befehle, Argumente, Operatoren

Funktion	Verwendung
`table(x)`	Erstellt eine Tabelle mit absoluten Häufigkeiten für die Variable x
`prop.table(table(x))`	Erstellt eine Tabelle mit relativen Häufigkeiten für die Variable x
`100*prop.table(table(x))`	Erstellt eine Tabelle mit den prozentualen Häufigkeiten für die Variable x
`cumsum(table(x))`	Erstellt eine Tabelle mit den kumulierten Häufigkeiten für die Variable x
`median(x, na.rm=TRUE)`	Bestimmt den Median der Variable x.
`quantile(x, na.rm=TRUE)`	Liefert Quartile (25 %, 50 %, 75 %) sowie Min (0 %) und Max (100 %) der Variable x
`IQR(x, na.rm=TRUE)`	Liefert den Interquartilsabstand
`mean(x, na.rm=TRUE)`	Bestimmt das arithmetische Mittel der Variable x
`var(x, na.rm=TRUE)`	Bestimmt die (geschätzte Populations-) Varianz der Variable x
`sd(x, na.rm=TRUE)`	Bestimmt die (geschätzte Populations-) Standardabweichung der Variable x
`range(x, na.rm=TRUE)`	Gibt den Streubereich der Variable x an
`min(x, na.rm=TRUE)`	Gibt den niedrigsten Wert der Variable x an
`max(x, na.rm=TRUE)`	Gibt den höchsten Wert der Variable x an
`barplot(table(x))`	Erstellt ein Säulendiagramm für den Faktor (oder Vektor) x
`hist(x, right = FALSE)`	Erstellt ein Histogramm für den Vektor x mit linksschließenden Intervallen
`boxplot(x)`	Erstellt einen Boxplot für den Vektor x
`plot(objekt)`	Erstellt eine Grafik abhängig von Objektklasse

6 Appendix

6.1 Relativer Informationsgehalt

Der relative Informationsgehalt H ist ein weiteres Streuungsmaß, das man bereits für nominalskalierte Daten verwenden kann. Wenn die Ausprägungen (meist Kategorien) einer Variable alle gleich häufig sind, nimmt H den Wert 1 an. Wenn nur eine Ausprägung vorkommt, nimmt H den Wert 0 an. Der relative Informationsgehalt ist definiert als:

$$H = -\frac{1}{\ln k} \sum_{j=1}^{k} h_j \ln h_j$$

In R muss man diesen selber implementieren:

```
H <- -(1/log(length(h_j))) * sum(h_j*log(h_j))
```

- h_j steht hier stellvertretend für den Vektor mit den relativen Häufigkeiten, für den man den relativen Informationsgehalt berechnen möchte

6.2 ggplot2

Man kann mit R sehr professionelle Grafiken erstellen. Üblicherweise werden dafür aber Pakete eingesetzt, die das *Grafikarsenal* R's erweitern. Ein sehr populäres Grafikpaket heißt `ggplot2` und kann mit `install.packages("ggplot2")` installiert werden. Der Autor des Pakets hat eine eigene Syntax für die Erstellung von Grafiken entwickelt. Deshalb unterscheidet sich das Erstellen von Grafiken mit `ggplot2` deutlich von dem Vorgehen mit `R Base Graphics`.

Das übliche Vorgehen beim Erstellen einer Grafik mit `ggplot2` kann in mehrere Schritte unterteilt werden.

1. Es wird ein Grafikobjekt mit der Funktion `ggplot()` erstellt
2. Es wird festgelegt, welche Variablen wo (z. B., x- oder y-Achse) auf der Grafik erscheinen sollen. Hier wird auch festgelegt, wie die Variablen eingefärbt oder gruppiert werden sollen.
3. Im letzten Schritt wird bestimmt wie die Variablen gezeichnet werden. Zum Beispiel als Histogramm oder Balkendiagramm.

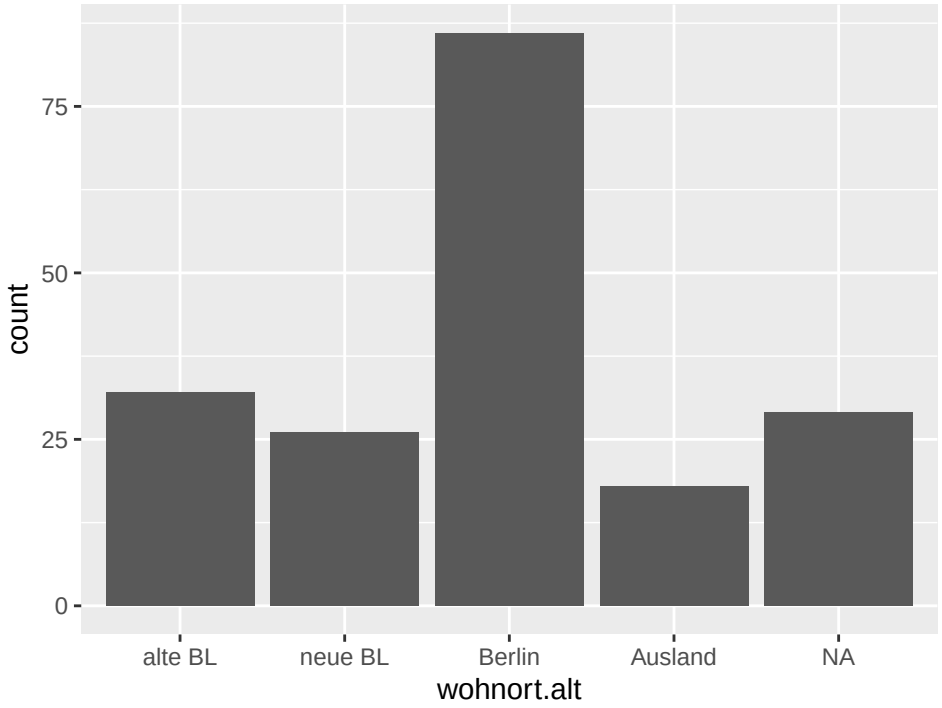
```
library(ggplot2)
```

- zuerst muss das Paket initialisiert werden

```
p <- ggplot(data = erstis,
            mapping = aes(x = wohnort.alt))
```

- Schritt 1: Wir erstellen ein Grafikobjekt `p` mit der Funktion `ggplot()`
  - mit `data` bestimmen wir den Datensatz, für welchen wir eine Grafik erstellen möchten (hier: `erstis`)
- Schritt 2: Wir bestimmen die Variablen, die wir grafisch darstellen möchten und *wo* diese in der Grafik dargestellt werden sollen

```
p + geom_bar()
```



- Schritt 3: Mit `geom_bar()` erstellen wir ein Säulendiagramm

Für ein kurzes Tutorial folgen Sie diesem Link. Für eine umfassende Einführung in `ggplot2` folgen Sie diesem Link.