Inhaltsverzeichnis

In diesem Kapitel zur Deskriptivstatistik wird es um die Möglichkeiten gehen, die R Ihnen bietet, um Daten durch Tabellen, bestimmte Kennzahlen oder Grafiken zusammenzufassen und darzustellen. Die bivariate Deskriptivstatistik bezieht sich dabei auf das Betrachten von zwei Variablen.

1 Metrische Variablen

StatsReminder

Metrische Variablen umfassen intervall- und verhältnisskalierte Variablen. Bei der Messung von intervallskalierten Variablen wird die Äquivalenz, die Ordnung und die Verhältnisse von Differenzen verschiedener Ausprägungen berücksichtigt.

Verhältnisskalierte Variablen haben zusätzlich einen natürlichen Nullpunkt. Ein klassisches Beispiel einer verhältnisskalierten Variable ist die Größe einer Person. Bei einer solchen Variable lassen sich zusätzlich Aussagen über die Verhältnisse zwischen Werten der Variable machen. Eine Person, die 1,60 cm groß ist, ist zweimal so groß als eine 80 cm große Person.

1.1 Streudiagramm zweier metrischer Variablen

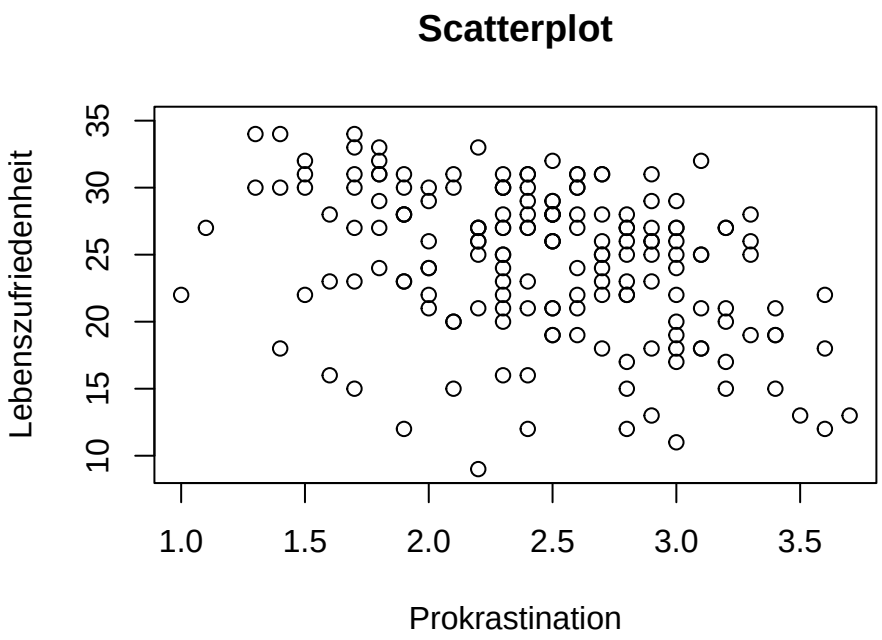
Mit einem Streudiagramm lassen sich Wertepaare zweier Variablen in einer Grafik betrachten. Es ist wichtig, dass diese Variablen an denselben Personen erhoben werden, damit die Wertepaare logisch in das Streudiagramm abgetragen werden können. Oft wird das Streudiagramm auch Punktewolke oder *Scatterplot* genannt. Generell liefert ein Streudiagramm Informationen über:

- Die Form des Zusammenhangs zweier Variablen (z. B. linear, quadratisch, kubisch)
- Die Richtung des Zusammenhangs zweier Variablen (z. B., positiver oder negativer linearer Zusammenhang)
- Stärke des Zusammenhangs zweier Variablen
- Ausreißer in den Daten

Die `plot()`-Funktion wurde bereits im letzten Kapitel eingeführt. Mit dieser lassen sich Streudiagramme schnell und einfach erstellen:

```
plot(x = erstis$prok, y = erstis$lz.1,
     xlab = "Prokrastination",
```

```
     ylab = "Lebenszufriedenheit",
     main = "Scatterplot")
```



- Hier wurde auf der x-Achse Prokrastination und auf der y-Achse Lebenszufriedenheit zum ersten Messzeitpunkt abgetragen
  - die Argumente `x` und `y` entsprechen den Achsen im Koordinatensystem

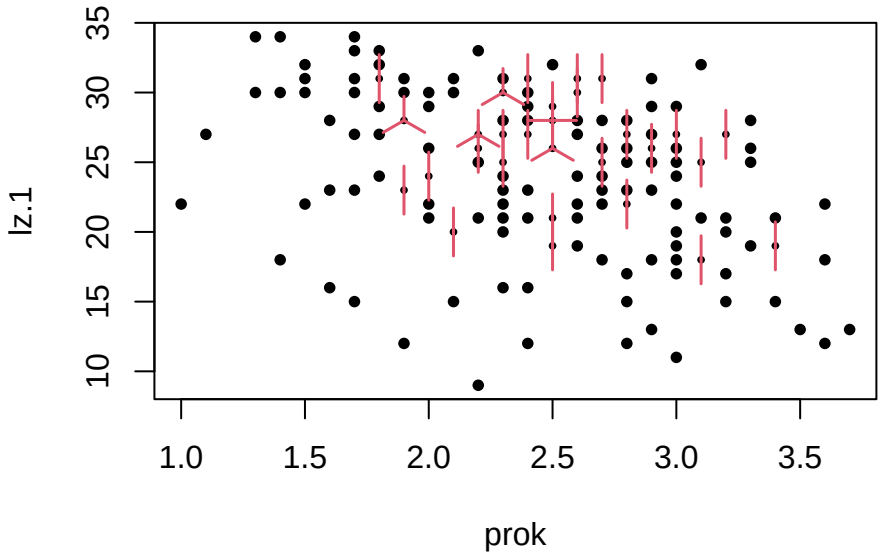
Alternativ:

```
plot(lz.1 ~ prok, data = erstis)
```

Der Ausdruck `lz.1 ~ prok` stellt eine Formel dar. In R werden Ihnen solche Formeln immer wieder begegnen. Insbesondere dann, wenn Grafiken (wie oben) oder statistische Modelle erstellt werden. Die Formel `lz.1 ~ prok` kann wie folgt gelesen werden: ***`lz.1`*** *(Lebenszufriedenheit) wird erklärt durch* ***`prok`*** *(Prokrastination)* oder ***`lz.1`*** *soll in Abhängigkeit von* ***`prok`*** *in einem Streudiagramm dargestellt werden.*
Wichtig: In Formeln kann der $-Operator nicht verwendet werden, daher muss der `plot()`-Funktion das `data`-Argument übergeben werden. Mit dem `data`-Argument teilen wir der `plot()`-Funktion mit, wo die in der Formel verwendeten Variablen zu finden sind.

In solchen Grafiken kann es zu Überlappungen kommen, die in einem regulären Streudiagramm nur schwer erkennbar sind. Um Überlappungen von Wertepaaren visuell erkennbar zu machen, kann man die Funktion `sunflowerplot()` verwenden:

```
sunflowerplot(lz.1 ~ prok, data = erstis)
```



- Wertepaare, die überlappen, werden hier als zusätzliche *Blütenblätter* dargestellt
  - jedes Blütenblatt (roter Strich) signalisiert, dass sich ein weiterer Punkt unter dem sichtbaren Punkt befindet

1.2 Kovarianz und Produkt-Moment-Korrelation

StatsReminder

Kovarianz und Produkt-Moment-Korrelation sind Maße für den linearen Zusammenhang zweier Variablen.
Die Kovarianz wird wie folgt berechnet (siehe Vorlesungsunterlagen):

$$s_{xy} = \frac{\sum_{m=1}^{n}(x_m - \bar{x})(y_m - \bar{y})}{n}$$

Bei einer positiven Kovarianz s_{xy} kann man folgern, dass sich die Werte der Variablen in dieselbe Richtung bewegen. Übersetzt: Umso höher die Ausprägung einer Person auf Variable X, desto höher ist tendenziell auch ihre Ausprägung auf der Variable Y, d. h. hohe Werte auf einer Variablen gehen in der Tendenz mit hohen Werten auf der anderen Variablen einher.
Im Gegensatz zur positiven Kovarianz kann man anhand einer negativen Kovarianz folgern, dass sich die Variablen in verschiedene Richtungen bewegen. Übersetzt: Umso höher die Ausprägung einer Person auf Variable X, desto niedriger ist tendenziell auch ihre Ausprägung auf der Variable Y, d. h. hohe Werte auf einer Variablen gehen in der Tendenz mit niedrigen Werten auf der anderen Variablen einher. Wenn die Kovarianz nahe Null liegt, deutet das darauf hin, dass kein linearer Zusammenhang zwischen den Variablen besteht.

Im Gegensatz zum Vorzeichen der Kovarianz lässt sich der Wert s_{xy} in der Regel nicht oder nur schwer interpretieren. Das liegt daran, dass die Kovarianz ein unstandardisiertes Maß ist. Wenn die Kovarianz standardisiert wird, erhalten wir die Produkt-Moment-Korrelation:

$$r_{xy} = \frac{\sum_{m=1}^{n}(x_m - \bar{x})(y_m - \bar{y})}{s_x s_y}$$

Die Werte der Korrelation lassen sich nun über die Vorzeichen hinaus interpretieren. Sie lassen nun auch Aussagen über die Stärke des Zusammenhangs zu. Die Korrelation kann Zahlen aus dem Intervall $[-1, 1]$ annehmen, wobei ein Werte um 0.1 als klein, Werte um 0.3 als moderat und Werte ab 0.5 als großer Zusammenhang interpretiert werden.

In R erhält man die Kovarianz und die Produkt-Moment-Korrelation respektive mit:

```
cov(erstis$prok, erstis$lz.1, use = "complete")
```

```
[1] -1.133705
```

```
cor(erstis$prok, erstis$lz.1, use = "complete")
```

```
[1] -0.3662255
```

- Die Kovarianz ist kein standardisiertes Zusammenhangsmaß, d.h. wir können aus dem Ergebnis nicht ablesen, ob der Zusammenhang stark oder schwach ist. Wir sehen nur, dass der Wert negativ ist. Für die Berechnung der Kovarianz in R mit dem Befehl cov() gilt der gleiche Hinweis wie für Berechnung der Varianz/Standardabweichung. Die in R berechneten Koeffizienten weichen von den „per Hand“ berechneten Kennzahlen anhand der Formeln aus der VL ab. Das Ergebnis der Korrelation hingegen ist identisch zu der Berechnung „per Hand“.
- Die Korrelation lässt uns auch eine Aussage über die Stärke des Zusammenhangs machen
  - Es gibt einen mittelstarken negativen linearen Zusammenhang zwischen `lz.1` und `prok`
  - Es folgt: Je höher die Lebenszufriedenheit, desto tendenziell geringer die Prokrastination

Auch die Funktionen `cov()` für die Kovarianz und `cor()` für die Produkt-Moment-Korrelation benötigen Angaben über den Umgang mit fehlenden Werten. Das Argument heißt hier `use` und hat verschiedene Optionen. Das Argument `use = "everything"` ist die Voreinstellung und führt zum Ergebnis `NA`, sobald

es für die ausgewählten Variablen Personen mit fehlenden Werten gibt. Das Argument `use = "complete"` bedeutet, dass Personen mit fehlenden Werten auf einer der Variablen ausgeschlossen werden.

Üblicherweise interessiert man sich nicht nur für die Korrelation zweier Variablen in einem Datensatz, sondern für die paarweisen Korrelationen zwischen mehreren Paaren aus Variablen in einem Datensatz. Um die paarweisen Korrelationen zwischen Variablen zu berechnen, indexieren wir unseren Data Frame `erstis` mit einem Zeichenvektor, der die Variablennamen beinhaltet und legen das Resultat in einem neuen Objekt ab. Anschließend kann die Korrelationsmatrix mit dem regulären Befehl `cor()` ausgegeben werden:

```
dat <- erstis[, c("extra", "prok", "lz.1")]   # Vorauswahl der Variablen
cor(dat, use = "complete")
```

```
           extra       prok       lz.1
extra  1.0000000 -0.1186157  0.1669917
prok  -0.1186157  1.0000000 -0.3662255
lz.1   0.1669917 -0.3662255  1.0000000
```

- Diese Tabelle gibt alle bivariaten Korrelationen für Variablen des (Sub-)Datensatzes an
  - man nennt eine Tabelle aus Korrelationen auch *Korrelationsmatrix*
- Auf der Diagonalen liegen die Korrelationen der Variablen mit sich selber

1.3 Umgang mit fehlenden Werten

Bei der Anforderung von bivariaten Korrelationen für mehr als 2 Variablen spielt der Umgang mit fehlenden Werten eine Rolle. Hier kann zwischen dem listenweisen und dem paarweisen Ausschluss von fehlenden Werten unterschieden werden.

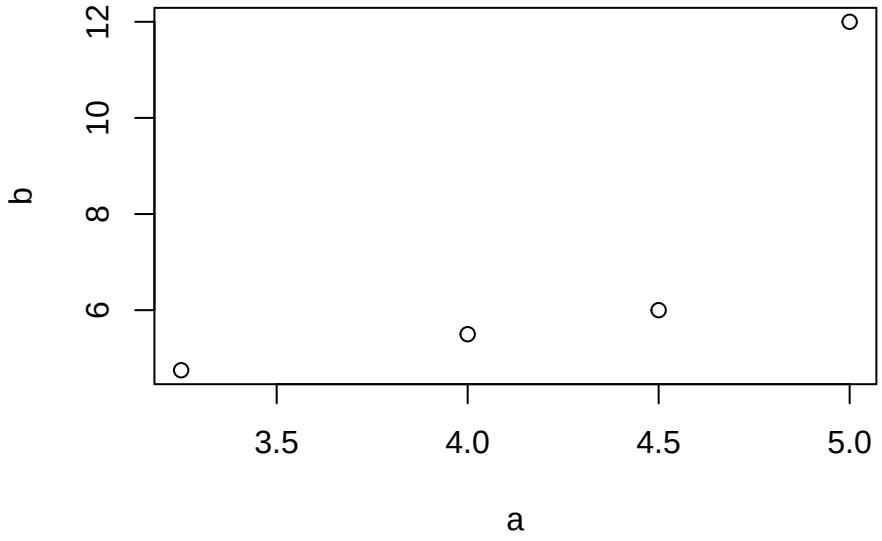
- Listenweiser Fallausschluss
  - entspricht dem strengen Umgang mit fehlenden Werten -> nur Personen mit gültigen Werten auf allen Variablen des Datensatzes werden analysiert
  - `use = "complete"`
- Paarweiser Fallausschluss
  - entspricht dem liberalen Umgang mit fehlenden Werten -> Personen, die *paarweise* gültige Werte haben, werden analysiert (also Werte nur auf den Spalten, die für die Berechnung der Kovarianz/Korrelation benötigt werden)
  - `use = "pairwise"`

1.3.1 Produkt-Moment-Korrelation bei Ausreißern

Die Produkt-Moment-Korrelation reagiert sensitiv auf Ausreißer, d.h. auf einzelne Wertepaare, die aus dem Muster "ausscheren". Ausreißer sind extreme Werte (oder außergewöhnliche Werte im Kontext der Daten). Im statistischen Jargon heißt *sensitiv*, dass die Produkt-Moment-Korrelation nicht robust gegenüber Ausreißern ist. Ausreißer können also das Ergebnis der Produkt-Moment-Korrelation deutlich verändern.

Beispiel:

```
a <- c(4.5, 5, 3.25, 4)
b <- c(6, 12, 4.75, 5.5)
plot(a, b)
```




```
cor(a, b)
```

```
[1] 0.8230172
```

1.3.2 Rangkorrelation nach Spearman

Bei Vorliegen von Ausreißern kann es sinnvoll sein, anstelle der Produkt-Moment-Korrelation die Rangkorrelation nach Spearman zu bestimmen. Sie ist definiert als Produkt-Moment-Korrelation der Ränge auf den beiden Variablen.

```
a <- c(4.5, 5, 3.25, 4)
b <- c(6, 12, 4.75, 5.5)
rank(a)
```

```
[1] 3 4 1 2
```

```
rank(b)
```

```
[1] 3 4 1 2
```

- Die vier Personen haben auf beiden Variablen die gleiche Rangfolge: Die erste Person hat auf Variable `a` und Variable `b` den 3. Rang. Die dritte Person hat auf beiden Variablen die geringste Ausprägung und daher den ersten Rang usw.

```
cor(rank(a), rank(b))
```

```
[1] 1
```

- Die Produkt-Moment-Korrelation der Ränge ist daher 1

```
cor(a,b, method = "spearman")
```

```
[1] 1
```

- Mit dem `method` Argument kann man spezifizieren, welcher Korrelationstyp verwendet werden soll

  - `method = "pearson"` entspricht der Produkt-Moment-Korrelation, welcher der default ist

2 Zwei ordinalskalierte kategoriale Merkmale

Wir erinnern uns, dass ordinalskalierte Daten einer Rangreihe folgen. Ein klassisches Beispiel für ein ordinalskaliertes Merkmal wären Schulnoten. Im `erstis`-Datensatz sind die Stimmungsvariablen "müde" (`stim7`) und "entspannt" (`stim12`) Beispiele für ordinalskalierte Merkmale (Antworten der Personen auf Items mit geordneten Antwortkategorien).

```
table(erstis$stim7, erstis$stim12)
```

```
    1  2  3  4  5
1   1  1  3  2  1
2   2  9 19 26  2
3   3  8 23 22  0
4   6 11 18 19  1
5   1  5  4  2  0
```

2.1 γ-Koeffzient

Der γ-Koeffzient wird für die Berechnung der Korrelation zwischen zwei geordneten ordinalskalierten kategorialen Merkmalen verwendet.

```
library(Hmisc)
rcorr.cens(erstis$stim7, erstis$stim12, outx = TRUE)[2]
```

```
      Dxy
-0.2006142
```

- Für die Berechnung des γ-Koeffizienten benutzt man die `rcorr.cens()`-Funktion, welche im `Hmisc`-Paket zu finden ist
  - der output dieser Funktion bietet viele Informationen; für den Moment reicht es den γ-Koeffzienten ausgeben zu lassen (daher die Indexierung `[2]`)

3 Zwei nominalskalierte kategoriale Merkmale

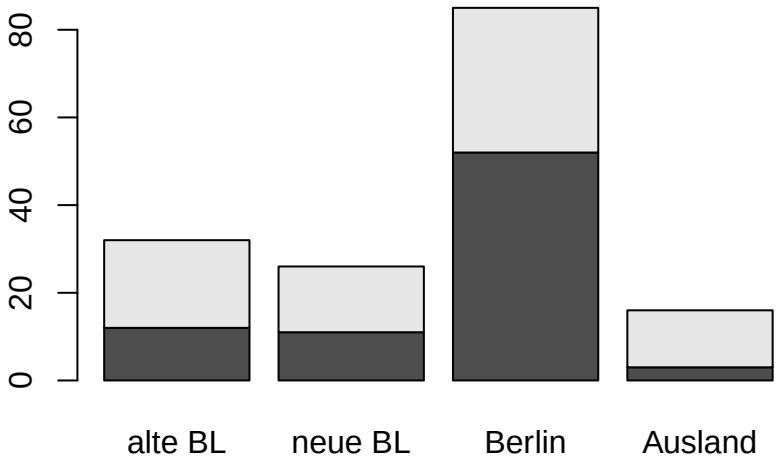
Merkmale sind nominalskaliert, wenn die Ausprägungen des Merkmals voneinander unterscheidbare, ungeordnete Kategorien sind. Im `erstis`-Datensatz sind `job` und `wohnort.alt` nominalskalierte Variablen.

```
table(erstis$job, erstis$wohnort.alt)
```

```
       alte BL neue BL Berlin Ausland
  ja         12      11     52       3
  nein       20      15     33      13
```

Es gibt verschiedene Möglichkeiten, nominalskalierte Variablen graphisch darzustellen.

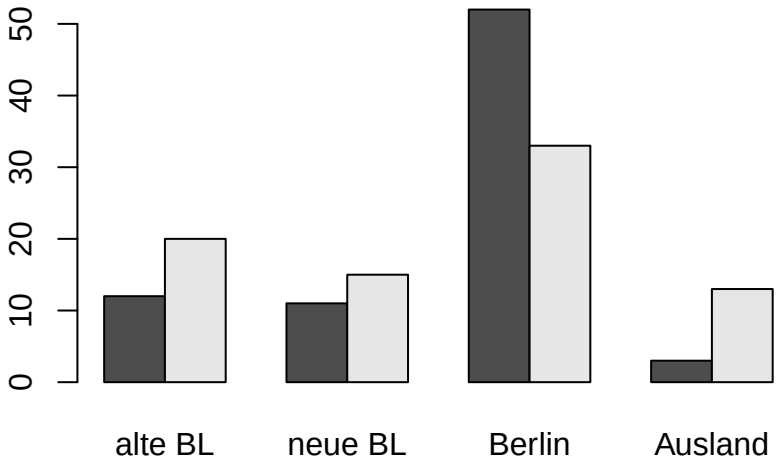
```
barplot(table(erstis$job, erstis$wohnort.alt))
```



- Per Default stellt der Barplot die Kategorien "gestapelt" dar

Eine elegantere Variante ist die Darstellung als gruppiertes Säulendiagramm:

```
barplot(table(erstis$job, erstis$wohnort.alt),
        beside = TRUE)
```

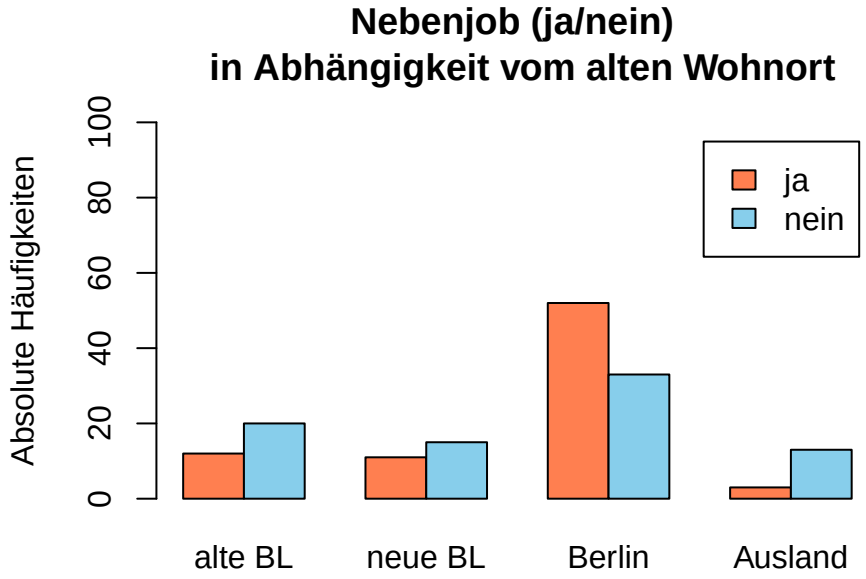


- Die Häufigkeiten sind nun deutlich einfacher zu vergleichen

  - durch das `beside = TRUE` Argument werden die Balken nebeneinander dargestellt

- Diese Darstellung lässt sich natürlich auch ein wenig aufwerten:

```
barplot(table(erstis$job, erstis$wohnort.alt),
        beside = TRUE,
        main = "Nebenjob (ja/nein) \n in Abhängigkeit vom alten Wohnort",
        ylab = "Absolute Häufigkeiten",
        ylim = c(0, 100),
        col = c("coral", "skyblue"),
        legend = TRUE)
```



- Bedeutung der Argumente:
  - `main`: Hier kann die Überschrift als Zeichenvektor übergeben werden (das `\n` beduetet das an dieser Stelle ein Zeilenumbruch eingefügt werden soll)
  - `ylab`: Hier kann die Y-Achsenbeschriftung verändert werden
  - `ylim`: Hier kann der gewünschte Wertebereich der Y-Achse bestimmt werden
  - `col`: Dient der Bestimmung der Säulenfarben
  - `legend`: Hier kann man sich eine Legende zur Beschriftung der Farben ausgeben lassen

Mit `?barplot` kann man sich weitere Informationen zu Argumenten bzgl. der Bearbeitung von Barplots und Grafiken im Allgemeinen anlesen.

4 Spezialfall: Zwei dichotome Merkmale

Wenn beide Variablen dichotom sind, können folgende Koeffizienten bestimmt werden:

- φ-Koeffizient
- Yule's Q
- Odd's Ratio

Für die Berechnung dieser Koeffizienten nutzen wir die Variablen `uni7` (Unisport) und `uni8` (Unipartys).

```
table(erstis$uni7, erstis$uni8)
```

```
      nein ja
  nein   99 20
  ja     21 26
```

```
addmargins(table(erstis$uni7, erstis$uni8))
```

```
      nein  ja Sum
  nein   99  20 119
  ja     21  26  47
  Sum   120  46 166
```

- mit `addmargins()` lassen sich zusätzlich die Randsummen einer Tabelle ausgeben

```
library(psych)    # psych: Funktionen für die Berechnung von Yule's Q und phi-Koeffizient
phi(table(erstis$uni7, erstis$uni8))  # phi-Koeffizient
```

```
[1] 0.39
```

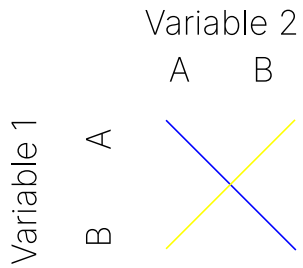
```
Yule(table(erstis$uni7, erstis$uni8)) # Yule's Q
```

```
[1] 0.7194389
```

- mit `phi()` wird der φ-Koeffizient berechnet
- mit `Yule()` wird Yule's Q berechnet
  - an beide Funktionen müssen die Daten der Variablen in Tabellenformat übergeben werden (`table()`)

Interpretation der Koeffizienten: Personen, die am Unisport teilnehmen, gehen auch eher auf Uniparties als Personen, die keinen Unisport machen.

Eine simple Interpretationshilfe kann wie folgt veranschaulicht werden:



Die blaue linie beschreibt einen positiven Zusammenhang: Kategorie A von Variable 1 geht eher mit Kategorie A von Variable 2 einher (und B eher mit B).
Die gelbe Linie beschreibt einen negativen Zusammenhang: Kategorie A von Variable 1 geht eher mit Kategorie B von Variable 2 einher (und B eher mit A).

Die Odd's Ratio lässt sich wie folgt bestimmen:

```
library(DescTools)
OddsRatio(table(erstis$uni7, erstis$uni8))
```

```
[1] 6.128571
```

Interpretation der Odd's Ratio: Die Chance, nicht auf Unipartys zu gehen, ist bei Personen, die nicht am Unisport teilnehmen, gut 6-mal größer als bei Personen, die am Unisport teilnehmen.

5 Ein kategoriales und ein metrisches Merkmal

5.1 Gruppierte Deskriptivstatistiken

Wenn man sich die Verteilungskennwerte eines metrischen Merkmals in Abhängigkeit eines kategorialen Merkmals (also die Verteilungen innerhalb der Gruppen der kategorialen Variable) anschauen möchte, kann `tapply()` verwendet werden. In unserem Beispiel verwenden wir das metrische Merkmal *wache Stimmung* (`wm`) und das kategoriale Merkmal Gruppe (`gruppe`).

```
tapply(erstis$wm.1, erstis$gruppe, mean, na.rm = TRUE)
```

```
       1        2        3        4
3.324468 3.060606 2.856250 3.058511
```

- `tapply()` wendet Funktionen auf Subgruppen an, z. B. die mittlere Wachheit getrennt nach Kursen
  - zuerst wird `erstis$wm.1` übergeben (das Merkmal, für das ein bestimmter Kennwert berechnet werden soll)
  - `erstis$gruppe` ist der Faktor, welcher die Gruppen definiert
  - `mean` ist die anzuwendende Funktion (in diesem Fall der Kennwert), der berechnet werden soll (hier könnten auch andere Funktionen wie bspw. `sd()` verwendet werden)
  - an letzter Stelle können Argumente für die anzuwendende Funktion übergeben werden

Interpretation: Teilnehmer im Kurs 1 scheinen im Mittel etwas wacher zu sein als der Durchschnitt, während Teilnehmer im Kurs 3 weniger wach zu sein scheinen.

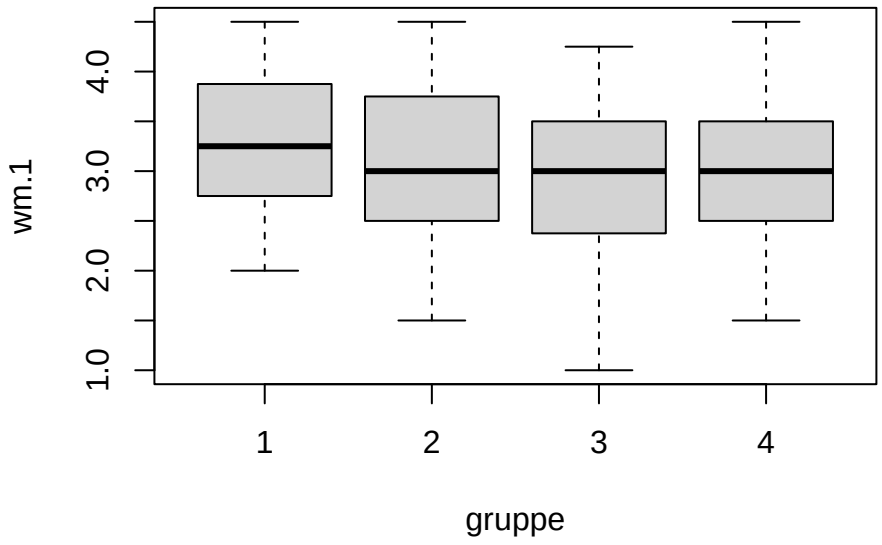
Wenn man eine umfangreiche gruppierte Übersicht zu diversen Statistiken erstellen will, kann man `describeBy()` verwenden.

```
library(psych)
describeBy(erstis$wm.1, erstis$gruppe)
```

```
 Descriptive statistics by group
group: 1
   vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
X1    1 47 3.32 0.59   3.25    3.32 0.74   2 4.5   2.5 0.04    -0.99 0.09
------------------------------------------------------------
group: 2
   vars  n mean   sd median trimmed  mad min max range  skew kurtosis  se
X1    1 55 3.06 0.76      3    3.09 0.74 1.5 4.5     3 -0.36    -0.69 0.1
------------------------------------------------------------
group: 3
   vars  n mean  sd median trimmed  mad min  max range skew kurtosis   se
X1    1 40 2.86 0.8      3     2.9 0.74   1 4.25  3.25 -0.4    -0.77 0.13
------------------------------------------------------------
group: 4
   vars  n mean   sd median trimmed  mad min max range  skew kurtosis  se
X1    1 47 3.06 0.69      3    3.06 0.74 1.5 4.5     3 -0.14    -0.78 0.1
```

Eine grafische Veranschaulichung wäre ein gruppierter Boxplot:

```
boxplot(wm.1 ~ gruppe, data = erstis)
```



- Hier zeigen alle vier Boxplots die Verteilung der wach-müde Stimmung innerhalb der Kategorien des Merkmals `gruppe`

6 Übersicht

6.1 Neue wichtige Konzepte

- **Streudiagramm**
- **Kovarianz & Korrelation**
- **Zusammenhangsmaße von nominalskalierten und ordinalskalierten Variablen**
- **Gruppierte Deskriptivstatistiken**

6.2 Neue wichtige Befehle, Argumente, Operatoren

Funktion	Verwendung
`plot(y ~ x)`	Erstellt ein Streudiagramm für die kontinuierlichen Variablen x und y
`sunflowerplot(y ~ x)`	Streudiagramm, wobei multiple Datenpunkte als zusätzliche "Blätter" dargestellt werden
`cor(x, y)` `cor(Daten)`	Bestimmt die Pearson-Korrelation für die Variablen x und y bzw. alle bivariaten Korrelationen im Datensatz `Daten`.
`rcorr.cens(Daten)`	Bestimmt den γ-Koeffzienten
`phi(table(Daten))`	Bestimmt den φ-Koeffizienten
`Yule(table(Daten))`	Bestimmt den Yule's Q
`addmargins(table(Daten))`	Berechnet die Randsummen einer Tabelle
`OddsRatio(table(Daten))`	Bestimmt die Odd's Ratio