Inhaltsverzeichnis

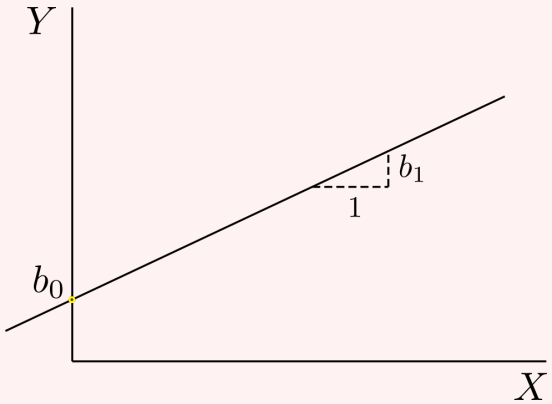
1 Einfache Lineare Regression in R

StatsReminder

Eine zentrale Aufgabe der psychologischen Forschung ist die Erklärung und Vorhersage von Zusammenhängen. Für diese Aufgabe wird die einfache lineare Regression häufig eingesetzt. Die einfache lineare Regression ist ein statistisches Verfahren, mit welchem man den Zusammenhang zweier Variablen durch eine Gerade beschreiben kann (deshalb **lineare** Regression). Vielleicht erinnern Sie sich noch an die Formel einer Geraden $Y = mX + c$. Diese Formel ist auch die Grundlage für die einfache lineare Regression, wobei die Parameter der Formel neue Symbole erhalten:

$$\widehat{Y} = b_0 + b_1 X$$

Hierbei stehen b_0 und b_1 für Zahlen, die die lineare Beziehung zwischen Y und X beschreiben. Y wird als *abhängige Variable* und X als *unabhängige Variable* bezeichnet. Die abhängige Variable ist die Zielgröße, die es durch die unabhängige Variable zu erklären gilt. Diese Formel lässt sich in einem Koordinatensystem grafisch darstellen, indem man die Werte der abhängigen Variable auf der Y-Achse und die Werte der unabhängigen Variable auf der X-Achse abträgt. Aus der Grafik kann man leicht die Funktion von b_0 (Y-Achsenabschnitt der Gerade oder *Intercept*) und b_1 (Steigung der Gerade oder *Slope*) ablesen.



Im nächsten Kapitel werden diese Konzepte auf die multiple Regression erweitert. Mit dieser können mehr als nur eine unabhängige Variable mit in das Modell aufgenommen werden.

1.1 Vorbereitung

Wir laden erneut den `erstis`-Datensatz:

```
load("dat/erstis_neu.RData")
```

Es soll nun eine einfache lineare Regression zur Vorhersage von `lz.1` (Lebenszufriedenheit) durch die `gs.1` (habituelle gute Stimmung) bestimmt werden.

```
sub <- na.omit(erstis[, c("lz.1", "gs.1", "wm.1")])
```

- Der Datensatz wird hier auf die Fälle reduziert, die gültige Werte haben (das dient der Verkürzung der kommenden Befehle)
  - die `wm.1` Variable fließt später in die Analyse ein

Eine kurze Exploration des Datensatzes:

```
nrow(sub)
```

```
[1] 189
```

- Der Datensatz enthält 189 Personen

```
round(cor(sub), digits = 2)
```

```
     lz.1 gs.1 wm.1
lz.1 1.00 0.63 0.16
gs.1 0.63 1.00 0.43
wm.1 0.16 0.43 1.00
```

- Aus der Korrelationsmatrix lässt sich ein starker positiver linearer Zusammenhang zwischen `lz.1` und `gs.1` ablesen ($\rho_{lz.1,gs.1} = 0.63$)

1.2 `lm()`

Um ein einfaches lineares Regressionsmodell in R anzufordern, kann die `lm()`-Funktion (engl., *linear model*) verwendet werden. Die `lm()`-Funktion kann vielseitig eingesetzt werden und daher ist deren Hilfeseite lang und komplex. Im Augenblick genügt es, die Argumente `formula` und `data` kennenzulernen. Um eine einfache lineare Regression zu rechnen, müssen `lm()` lediglich die Formel (`formula`) und die dazugehörigen Daten (`data`) übergeben werden. Die Formel definiert die Beziehung zwischen den Variablen (hier wird die abhängige und unabhängige Variable zugewiesen).

```
mod <- lm(formula = lz.1 ~ gs.1, data = sub)
```

- Dem `formula`-Argument wird ein Objekt übergeben, welches die von uns zu untersuchende Beziehung zwischen den Variablen beschreiben soll
  - links des Tilde-Zeichens (`~`) steht die abhängige Variable, während rechts die unabhängige Variable abgetragen wird
  - `lz.1` entspricht der abhängigen Variable; `gs.1` entspricht der unabhängigen Variable
  - gelesen wird die gesamte Formel als: Lebenszufriedenheit (`lz.1`) in Abhängigkeit (`~`) von guter Stimmung (`gs.1`)
- Mit `data = sub` werden die zur Berechnung der Regression benötigten Daten übergeben bzw. die Variablennamen aus der Formel werden mit den Spalten des Data Frames `sub` verglichen und zugeordnet (falls diese vorhanden sind, falls nicht erscheint eine Fehlermeldung)
- Auch hier kann das resultierende Objekt abgespeichert werden (`mod`)
  - oft nennt man das resultierende Objekt *Modellobjekt*

Wie bei einem Datensatz/Datenobjekt kann auch bei einem sogenannten Modellobjekt die Struktur mit `str()` genauer unter die Lupe genommen werden.

```
str(mod)
```

- Auf einzelne Elemente der Liste kann mit der Schreibweise `objekt$Element` zugegriffen werden (hier nicht ausgegeben, da der Output sehr lang ist)

Wenn man sich lediglich für die Koeffizienten (b_0 und b_1) des Regressionsmodells interessiert, dann reicht es, wenn man das Objekt, in welchem die Resultate der `lm()`-Funktion abgespeichert sind, in der *Console* ausführt. In unserem Fall führen wir die Zeile mit dem Objektnamen `mod` einfach aus:

```
mod
```

```
Call:
lm(formula = lz.1 ~ gs.1, data = sub)

Coefficients:
(Intercept)         gs.1
      2.284        5.730
```

- Der Output ist in zwei Teile untergliedert:
  - der sogenannte `call` zeigt, was den Argumenten der `lm()`-Funktion vor dem Ausführen übergeben wurde
  - der zweite Teil besteht aus den Regressionskoeffizienten (`coefficients`)
- Der `Intercept` 2.28 entspricht b_0 in der obigen Regressionsgleichung
  - der `Intercept` entspricht der geschätzten Lebenszufriedenheitswert (`lz.1`) für Personen mit einem Wert von 0 auf guter Stimmung (`gs.1`)
  - **Achtung:** die Interpretation des `Intercepts` macht nicht automatisch Sinn! 0 ist kein Wert, den die Variable `gs.1` annehmen kann
- Der Wert 5.73 entspricht b_1 in der Regressionsgleichung
  - b_1 entspricht dem Steigungsparameter und ist daher definiert als der Unterschied in der erwarteten/vorhergesagten Lebenszufriedenheit zwischen zwei Personen, die sich um eine Einheit in der guten Stimmung unterscheiden

Alternativ können die Koeffizienten mit den Befehlen `coef(mod)` oder `mod$coef` ausgegeben werden:

```
coef(mod)   # alternativ: mod$coef
```

```
(Intercept)        gs.1
   2.283803    5.730025
```

Mithilfe der geschätzten Koeffizienten kann man die daraus resultierende Regressionsgleichung erstellen:

$$\widehat{Y} = 2.28 + 5.73X$$

$\widehat{Y}$ = die geschätzte Lebenszufriedenheit `lz1`
X = die habituelle gute Stimmung `gs.1`

1.2.1 Einfache lineare Regression mit zentrierter unabhängiger Variable

Um eine bessere inhaltliche Interpretation von b_0 zu ermöglichen, kann man die unabhängige Variable `gs.1` zentrieren. Beim Zentrieren subtrahiert man von jedem individuellem Wert in `gs.1` den Mittelwert von `gs.1`. Die Transformation der unabhängigen Variable führt zu einem anderen `Intercept` b_0 in der Regressionsgleichung und daher auch zu einer anderen Interpretation dieses Koeffizienten.

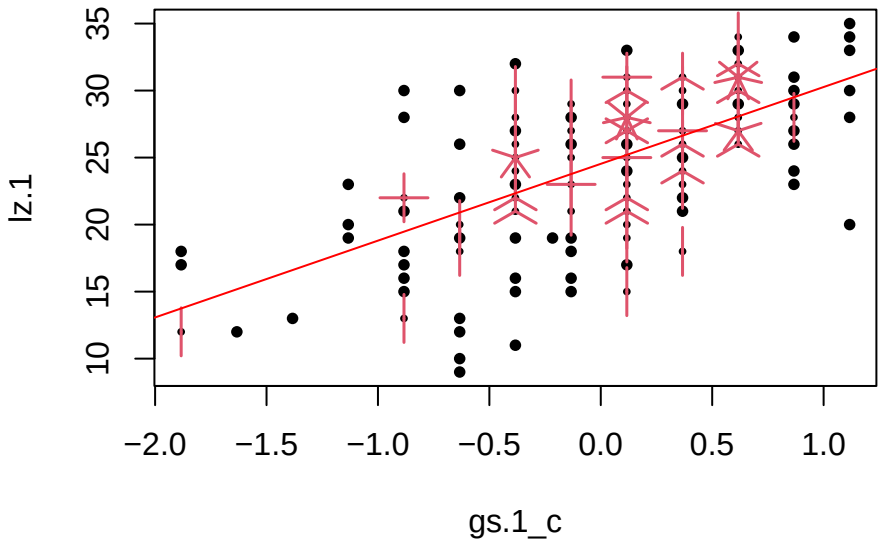
```
sub$gs.1_c <- scale(sub$gs.1, center = TRUE, scale = FALSE)  # zentrieren von gs.1
mod_c <- lm(lz.1 ~ gs.1_c, data = sub)
coef(mod_c)
```

```
(Intercept)      gs.1_c
  24.534392    5.730025
```

- `scale()` wird `scale = FALSE` übergeben, da `gs.1` zentriert aber nicht skaliert werden soll
  - durch das Zentrieren hat die gute Stimmung `gs.1_c` nun einen Mittelwert von 0
- `mod_c` ist das neue Regressionsmodell, das mit dem zentrierten `gs.1_c` berechnet wurde
- b_1 hat sich im Vergleich zum ersten Modell nicht verändert, da sich an der Skalierung der unabhängigen Variablen nichts geändert hat
- b_0 ist nun der geschätzte Lebenszufriedenheitswert für Personen mit einem mittleren Wert auf guter Stimmung

1.2.2 Grafische Veranschaulichung der Regressionsgleichung

```
sunflowerplot(lz.1 ~ gs.1_c, data = sub)
abline(reg = mod_c, col = "red")
```



- Mit `abline()` lassen sich Geraden in eine bestehende Grafik einzeichnen
  - Argument `reg` nutzt die Regressionskoeffizienten aus dem zuvor erstellten Modellobjekt als Y-Achsenabschnitt und Steigung.
- Die rote Gerade nutzt die Regressionskoeffizienten aus dem zuvor erstellten Modellobjekt als Y-Achsenabschnitt und Steigung

1.3 Modellzusammenfassung

Mit dem `summary()` Befehl lassen sich auch statistische Modelle und damit Modellobjekte in R zusammenfassen. Auch hier enthält der Output wieder einige Informationen, die Sie zu diesem Zeitpunkt vermutlich noch nicht verstehen können. Wir fokussieren uns daher auf die Elemente, die Sie bereits gelernt haben und interpretieren können.

```
summary(mod_c)
```

```
Call:
lm(formula = lz.1 ~ gs.1_c, data = sub)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-11.9064  -2.6364   0.3636   2.9311  10.5261

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  24.5344     0.3196   76.75   <2e-16 ***
gs.1_c        5.7300     0.5226   10.96   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.394 on 187 degrees of freedom
Multiple R-squared:  0.3914,    Adjusted R-squared:  0.3881
F-statistic: 120.2 on 1 and 187 DF,  p-value: < 2.2e-16
```

- Der Output ist nun deutlich länger im Vergleich zum Output des Modellobjekts ohne `summary()`-Funktion:
  - hier wird auch die Sektion `Call` als Erstes angezeigt (mit welchen Argumenten `lm()` ausgeführt wurde)
  - unter `Residuals` erhält man Informationen über die Verteilung der Residuen
  - unter `Coefficients` werden die bereits bekannten geschätzten Koeffizienten (`Estimates`) dargestellt. Die anderen Spalten werden Sie im nächsten Semester kennenlernen
  - den Determinationskoeffizienten finden Sie unter `Multiple R-squared` in der vorletzten Zeile. Dieser beträgt hier $R^2 = 0.39$
  - der `Residual standard error` ist der Standardschätzfehler der Residuen

Falls Sie sich lediglich für den Determinationskoeffizienten interessieren, können Sie sich diesen direkt ausgeben lassen:

```
summary(mod_c)$r.squared
```

```
[1] 0.3913531
```

Alternativ kann man den Determinationskoeffizienten *händisch* ausrechnen, indem man die bivariate Korrelation quadriert (nur für den Fall der einfachen linearen Regression):

```
cor(sub$lz.1, sub$gs.1_c)^2
```

```
          [,1]
[1,] 0.3913531
```

1.3.1 Einfache lineare Regression mit standardisierten Variablen

Im Gegensatz zum vorherigen Beispiel, in welchem nur die unabhängige Variable zentriert wurde, sollen nun beide Variablen zentriert und skaliert (das entspricht der z-Transformation/Standardisierung) werden. Im Anschluss wird eine einfache lineare Regression mit den standardisierten Variablen angefordert.

```
std1 <- data.frame(scale(sub))
```

- Alle Variablen im Datensatz `sub` werden standardisiert

```
update(mod, data = std1)
```

```
Call:
lm(formula = lz.1 ~ gs.1, data = std1)

Coefficients:
(Intercept)         gs.1
 -1.939e-16    6.256e-01
```

- Mit dem Befehl `update()` kann man das Regressionsmodell berechnen, ohne die Formel erneut bestimmen zu müssen
  - `update()` nutzt die in `mod` bestimmte Formel und übergibt `lm()` den neuen Datensatz `std1`
  - daraufhin erhalten wir neue Regressionskoeffizienten, da die einfach lineare Regression mit den standardisierten Versionen von `lz.1` und `gs.1` berechnet wurde

Die neue Regressionsgleichung laut den Ergebnissen des neuen Modells:

$$\widehat{Y} = 0 + 0.63X$$

- $b_{0s} = 0$ ist der geschätzte Lebenszufriedenheits-z-Wert für Personen mit mittlerer guter Stimmung (mittlere Lebenszufriedenheit)
- $b_{1s} = 0.63$ ist der Unterschied in den geschätzten z-Werten der Lebenszufriedenheit zwischen zwei Personen, die sich um eine Standardabweichung in der guten Stimmung unterscheiden. Wir erwarten einen Unterschied von 0.63 Standaradbweichungen in der Lebenszufiredenheit zwischen zwei Personen, die sich um eine Standardabweichung in ihrer guten Stimmung voneinander unterscheiden.

1.4 Residuen

Das Residuum ist die Differenz zwischen des tatsächlich beobachteten und auf Basis des Modells vorhergesagten Werts. Es lässt sich also für jede Person im Datensatz ein Residuum berechnen.

```
sub$E <- resid(mod_c)  # alternativ: mod_c$residuals
```

- Mit `resid()` lassen sich die geschätzten Residuen aller Personen ausgeben
- Aufgrund der Länge des Outputs wurde dieser nicht mitgedruckt

Da es 189 Personen im Datensatz gibt, gibt es auch 189 Residuen und dementsprechend kann man sich Deskriptivstatistiken zu den Residuen mit `summary()` ausgeben lassen:

```
summary(sub$E)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-11.9064  -2.6364   0.3636   0.0000   2.9311  10.5261
```

- Der Mittelwert der Residuen ist immer 0; das folgt aus dem benutzten Schätzverfahren für die Regression (Kleinste-Quadrate-Schätzer)
- Die Deskriptivstatistik der Residuen ist auch im `summary()`-Output unter `Residuals` wiederzufinden

```
sd(sub$E)
```

```
[1] 4.382702
```

- Die Standardabweichung der Residuen entspricht dem Standardschätzfehler (`Residual standard error` in der Modellzusammenfassung)
  - der Standardschätzfehler ist ein Maß für die Streuung der Residuen und stellt dar, wie präzise die Schätzung ist

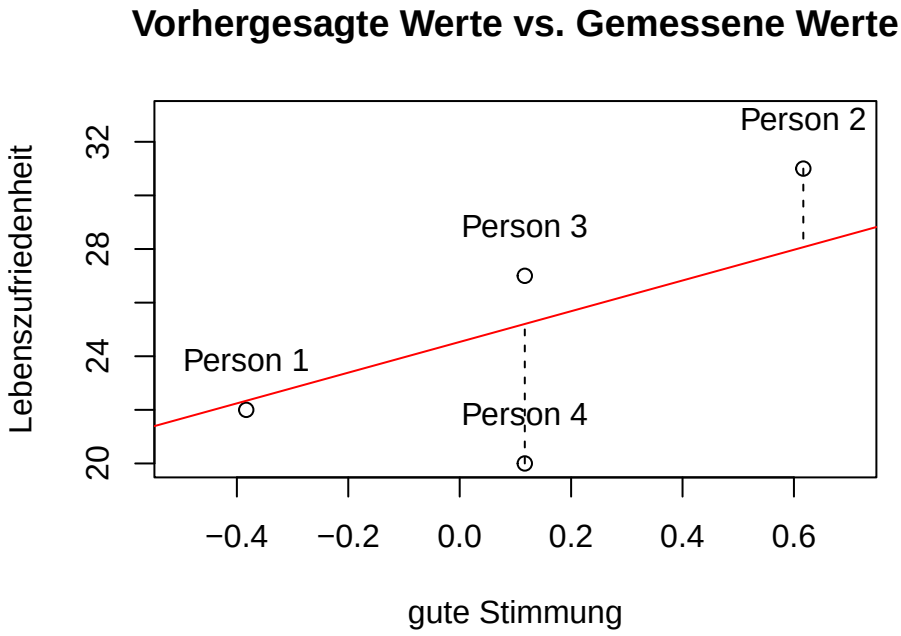
1.5 Vorhergesagte Werte

Mit `predict()` lassen sich die vorhergesagten Werte des Regressionsmodells ausgeben:

```
sub$yDach <- predict(mod_c)
round(head(sub, n = 4), digits = 2)
```

```
  lz.1 gs.1 wm.1 gs.1_c     E yDach
1   22  3.5 2.25  -0.38 -0.34 22.34
2   31  4.5 2.50   0.62  2.93 28.07
3   27  4.0 3.50   0.12  1.80 25.20
4   20  4.0 2.50   0.12 -5.20 25.20
```

- `yDach` ist eine extra Spalte, die wir im Data Frame `sub` für die vorhergesagten Werte der Personen anlegen
  - die Personen in Zeilen 3 und 4 haben denselben Wert auf der unabhängigen Variable `gs.1_c` und daher auch denselben vorhergesagten Wert `yDach` (die geschätzten Werte werden in der Abbildung durch die rote Gerade dargestellt)
  - der beobachtete `lz.1`-Wert von Person 3 wird vom Modell unterschätzt (positives Residuum `E`), von Person 4 überschätzt (negatives Residuum `E`)
  - das Residuum der ersten Person ist klein und damit liegt der geschätzte Wert `yDach` sehr nah am beobachteten Wert



1.6 Varianzzerlegung

Wie in der Vorlesung besprochen, lässt sich die gesamte Streuung der Daten "zerlegen" in die *erklärte* Varianz, welche auf die Modellvorhersage zurückzuführen ist und in die nicht erklärte Varianz (oft auch Residualvarianz genannt).

$$s^2_Y = s^2_{\widehat{Y}} + s^2_E$$

```
var(sub$yDach) + var(sub$E)
```

```
[1] 31.55865
```

- erklärte Varianz + Residualvarianz

```
var(sub$lz.1)
```

```
[1] 31.55865
```

- Gesamtvarianz in `lz.1`

1.7 Korrelationen

Lassen wir uns einmal die Korrelationsmatrix für die Variablen und der Residuen sowie der vorhergesagten Werte des Regressionsmodells ausgeben:

```
round(cor(sub), digits = 2)
```

```
       lz.1 gs.1  wm.1 gs.1_c     E yDach
lz.1   1.00 0.63  0.16   0.63  0.78  0.63
gs.1   0.63 1.00  0.43   1.00  0.00  1.00
wm.1   0.16 0.43  1.00   0.43 -0.14  0.43
gs.1_c 0.63 1.00  0.43   1.00  0.00  1.00
E      0.78 0.00 -0.14   0.00  1.00  0.00
yDach  0.63 1.00  0.43   1.00  0.00  1.00
```

Wir können folgende Beobachtungen machen:

- Unzentrierter und zentrierter Prädiktor korrelieren perfekt
  - die Zentrierung einer Variable ändert nichts an den Korrelationen (invariant gegenüber linearen Transformationen)
  - siehe `gs.1_c` und `gs.1`
- Prädiktor und vorhergesagte Werte korrelieren perfekt
- Prädiktor und Residuum sind unkorreliert
- Die Korrelation zwischen Kriterium und vorhergesagten Werten entspricht der Korrelation zwischen Kriterium und Prädiktor
- Kriterium und Residuen korrelieren hoch (zu $1 - R^2$)

Gleichen Sie diese Beobachtungen doch einmal mit den theoretischen Überlegungen zur linearen Regression in den Vorlesungsfolien ab.

1.8 Partialkorrelation

StatsReminder

Die partielle Korrelation bietet eine Möglichkeit, den Zusammenhang zwischen zwei Variablen, X und Y, zu *bereinigen*. Bereinigen heißt, dass der Einfluss einer Drittvariable Z kontrolliert wird. Dafür werden die Residuen der Regressionen von X auf Z und Y auf Z berechnet. Die Korrelation dieser Residuen entspricht der um Z bereinigten Korrelation zwischen X und Y. Man sagt auch, dass Z herauspartialisiert wurde.

Mithilfe des Pakets `ppcor` lassen sich Partialkorrelationen berechnen:

```
# install.packages("ppcor")
library(ppcor)
```

Angenommen, wir interessieren uns für den Zusammenhang von Lebenszufriedenheit (`lz.1`) und wacher-müder Stimmung (`wm.1`) unter Berücksichtigung des Einflusses der Variable *gute-schlechte Stimmung* (`gs.1`), dann können wir dafür die Funktion `pcor()` aus dem Paket `ppcor` nutzen. **Achtung!** Die Funktion `pcor()` kann mit Datensätzen, die fehlende Werte enthalten, nicht umgehen. Da wir die fehlende Werte im Datensatz `sub` bereits entfernt haben, können wir diesen `pcor()` übergeben.

```
pcor(sub[, c("lz.1", "gs.1", "wm.1")])$estimate
```

```
           lz.1      gs.1       wm.1
lz.1  1.0000000 0.6248441 -0.1540567
gs.1  0.6248441 1.0000000  0.4260447
wm.1 -0.1540567 0.4260447  1.0000000
```

- Als Argument werden die entsprechenden Variablen im Data Frame-Format übergeben

- Die Indexierung `$estimate` am Ende des Befehls beschränkt den Output auf die für uns relevanten Partialkorrelationen
- Der Wert -0.15 entspricht der gesuchten Partialkorrelation zwischen Lebenszufriedenheit und wacher-müder Stimmung, wenn die gute-schlechte Stimmung herauspartialisiert wurde
  - verglichen mit der normalen Korrelation dreht sich das Vorzeichen um (siehe oben unter Abschnitt Korrelationen)
- Die resultierende Korrelationsmatrix ist symmetrisch, d. h., dass die Werte in der Matrix spiegelsymmetrisch im Bezug auf die Diagonale sind (Diagonale von links oben nach rechts unten)
  - es spielt also keine Rolle, ob Sie bspw. den Wert -0.15 aus der Zelle der 1. Spalte und 3. Zeile oder der 3. Spalte und 1. Zeile ablesen
- Allgemein lesen sich die Einträge der Korrelationsmatrix als Korrelation zwischen den Variablen der entsprechende Spalte bzw. Zeile, die aber um den Zusammenhang der dritten Variable bereinigt wurde

Alternativ: Dieselbe Partialkorrelation lässt sich *händisch* bestimmen, indem man sich die Korrelation der Residuen zwischen der Regression von Lebenszufriedenheit auf gute Stimmung und der Regression von wache-müde Stimmung auf gute Stimmung ausgeben lässt.

```
mod1 <- lm(lz.1 ~ gs.1_c, data = sub)
mod2 <- lm(wm.1 ~ gs.1_c, data = sub)
sub$mod1E <- resid(mod1)
sub$mod2E <- resid(mod2)
cor(sub$mod1E, sub$mod2E)
```

```
[1] -0.1540567
```

1.8.1 Semipartialkorrelation

Wie groß ist der Zusammenhang, wenn der Einfluss von guter vs. schlechter Stimmung nur auf die Lebenszufriedenheit (aber nicht auf die wache vs. müde Stimmung) berücksichtigt wird? Um so eine Semipartialkorrelation zu berechnen nutzen wir `spcor()` (ebenfalls aus dem Paket `ppcor`).

```
spcor(sub[, c("lz.1", "gs.1", "wm.1")])$estimate
```

```
           lz.1      gs.1       wm.1
lz.1  1.0000000 0.6169172 -0.1201886
gs.1  0.5648697 1.0000000  0.3323823
wm.1 -0.1392699 0.4206398  1.0000000
```

- Dieser Output unterscheidet sich ein wenig vom Output des `pcor` Befehls
  - die Korrelationsmatrix ist nun nicht mehr symmetrisch, da die Drittvariable nun nur aus einer der beiden Variablen auspartialisiert wird.
- Der Wert -0.14 ist hier die gesuchte Semipartialkorrelation (gute-schlechte Stimmung nur aus Lebenszufriedenheit herauspartialisert)
- Auch `spcor()` kann nur mit Datensätzen umgehen, die um ihre fehlenden Werte bereinigt wurden
- Im Gegensatz zur Partialkorrelation ist die resultierende Semipartialkorrelationsmatrix nicht mehr symmetrisch. Die Drittvariable wird immer aus der Variable, die in der entsprechenden Spalte angegeben ist, herauspartialisiert
  - die Semipartialkorrelation, in welcher die gute-schlechte Stimmung nur aus der wachen-müden Stimmung herauspartialisiert wurde, finden wir daher in der Zelle der ersten Zeile und dritten Spalte (hier ist `wm.1` in der Spalte und `lz.1` in der Zeile abgetragen)

2 Übersicht

2.1 Neue wichtige Konzepte

- **Einfache lineare Regression**
- **Vorhersagen durch die einfache lineare Regression**
- **Varianzzerlegung**
- **Partialkorrelation & Semipartialkorrelation**

2.2 Neue wichtige Befehle, Argumente, Operatoren

Funktion	Verwendung
`mod <- lm(Y ~ X, data = daten)`	Legt das Ergebnis einer Regression zur Vorhersage von Y durch X in Objekt mod ab
`coef(mod)`	Ausgabe der Regressionskoeffizienten
`scale(x, center = TRUE, scale = TRUE)`	Je nach Einstellung: Zentriert oder standardisiert die Variablen x
`summary(mod)`	Gibt vollständigen Output der Regressionsanalyse aus
`update(mod, data = neu)`	Model `mod` wird erneut mit Datensatz *neu* geschätzt (gleiche Variablennamen erforderlich)
`resid(mod)`	Gibt Residuen aus
`predict(mod)`	Gibt vorhergesagte Werte aus
`pcor() spcor()`	Berechnet die Partialkorrelationen bzw. die Semipartialkorrelationen