EDV2

Max Brede

2022-05-24

Contents

Chapter 1

Vorwort

Dieses mit `bookdown` erstellte Dokument ist das Skript zur Übung "PSY_B_12-2: Computerunterstützte Datenanalyse II" der CAU zu Kiel.

Chapter 2

Lehrplan

2.1 Semesterplan

2.2 Übungsformat

Die Übung soll zur Hälfte in 45-minütigen Sitzungen im Vorlesungsformat zur Vorstellung der Funktionen und zur anderen Hälfte als 45-minütige praktische Übung stattfinden. Es wird pro Übungs-Sitzung ein Übungszettel ausgegeben, der mit Hilfe der in der Zugehörigen Vorlesung besprochenen Funktionen bearbeitet werden können soll. Diese Zettel sollen nach der jeweiligen Vorlesung für die Übungen vorbereitet werden, in denen der Zettel dann besprochen und mögliche Fragen geklärt werden. Nach den Übungssitzungen haben die Studierenden dann eine Woche Zeit, zusätzliche Hausaufgaben zu bearbeiten.

Eine Ausnahme von diesem Ablauf ist die erste Sitzung, in der organisatorisches und Grundlagen in 90 minütigem Vorlesungsstil besprochen werden sollen. Auch nach dieser Sitzung werden aber Übungszettel und Hausaufgaben ausgegeben.

2.3 Lehrziele für jede Sitzung

Die Studierenden können nach dem Absolvieren der Übung…

Einheit 1

- Gründe für deskriptive Statistik nennen.
- für verschiedene Ausgangssituationen entscheiden, welche Darstellungsform angemessen ist.
- Verfahren zum Umgang mit fehlenden Werten nennen und anwenden.
- Verfahren zum Umgang mit Ausreißern nennen und anwenden.

Einheit 2

- R `formulas` lesen und verwenden.
- Korrelationsanalysen in R durchführen und die Ergebnisse interpretieren.
- einfache lineare Regressionen in R durchführen und die Ergebnisse interpretieren.

Einheit 3

- multiple lineare Regressionen in R durchführen und die Ergebnisse interpretieren.

Einheit 4

- t-Tests in R durchführen und die Ergebnisse interpretieren.
- einfaktorielle Varianzanalysen in R durchführen und die Ergebnisse interpretieren.

Einheit 5

- zweifaktorielle Varianzanalysen in R durchführen und die Ergebnisse interpretieren.

Einheit 6

- beliebige Linearkontraste in R durchführen und die Ergebnisse interpretieren.
- paarweise post-hoc t-Tests in R durchführen und die Ergebnisse interpretieren.

Einheit 7

- erfolgreich an der Klausur teilnehmen.

2.4 Prüfungsleistung

Die Studierenden **müssen** während des Semesters die nach den Übungssitzungen ausgegebenen Hausaufgaben innerhalb einer Woche sinnvoll bearbeitet abgeben.

Mit maximal einer nicht sinnvoll bearbeiteten Serie werden die Studierenden zur Gruppenarbeit am Ende des Semesters zugelassen.

Chapter 3

Deskriptive Statistik und Data Cleaning

3.1 Organisatorisches

3.1.1 Übungsablauf

Es wird alle zwei Wochen eine 45-minütige Vorlesung geben und dazu alternierend online-Übungsstunden in mit je einem Kurs. (Eine Ausnahme ist die erste Woche, in der wir nur eine 90-minütige Vorlesung haben.)

3.1.2 Übungsablauf

051232	Mo	14:15 - 15:00	n.V.	Dirk Bosy
Kurs Gruppe 8				
051216	Mo	15:00 - 15:45	n.V.	Dirk Bosy
Kurs Gruppe 5				
051212	Mo	16:15 - 17:00	n.V.	Maike Splittgerber
Kurs Gruppe 6				
051224	Mo	17:00 - 17:45	n.V.	Maike Splittgerber
Kurs Gruppe 7				
051227	Di	16:15 - 17:00	n.V.	Max Brede
Kurs Gruppe 3				
051215	Di	17:00 - 17:45	n.V.	Dirk Bosy
Kurs Gruppe 4				
051231	Do	16:15 - 17:00	n.V.	Ronja Kleine
Kurs Gruppe 9				
051217	Do	17:00 - 17:45	n.V.	Ronja Kleine
Kurs Gruppe 10				
051213	Fr	10:00 - 10:45	n.V.	Dirk Bosy
Kurs Gruppe 1				
051214	Fr	11:00 - 11:45	n.V.	Dirk Bosy
Kurs Gruppe 2				

3.1.3 Prüfungsleistung

- Gruppenarbeit(4-5 Personen) über 3 Wochen
- Termin für die Klausur entweder vor oder mit Anfang in der Klausurphase.

Genauere Informationen gibt es über das Olat.

Als Zulassung für die Gruppenarbeit ist wieder die sinnvolle Bearbeitung von allen bis auf eine Hausaufgabenserien nötig.

3.2 Deskriptive Statistik

3.2.1 Wozu brauche ich das?

Im Gegensatz zur Inferenzstatistik ist das erklärte Ziel der deskriptiven Statistik, (wie der Name schon sagt) beschreibende Aussagen über die vorliegende Stichprobe zu treffen.

Wir wollen uns also möglichst genau angucken, wie unsere Stichprobe aussieht.

Wozu könnte das gut sein?

3.2.2 Gründe für deskriptive Statistik

- Indikatoren zur externen Validität(Verteilung von Organismusvariablen, Demografie,...)
- Aussagen über Verteilungseigenschaften
  - Schnell zu erfassende Präsentationen von zentraler Tendenz und Streuungen
  - Hinweise auf ungewöhnliche Werte (Ausreißer, fehlende Werte,...)
- Übersicht über Effekte und Zusammenhänge, inklusive solcher, die möglicherweise nicht a-priori erwartet wurden

3.2.3 deskriptive Statistik

Für einen ganz einfachen, schnellen und umfassenden Überblick über die Daten funktioniert die `skim`-Funktion aus dem `skimr`-Paket gut:

```
skimr::skim(df)
```

3.2.4 Demografie

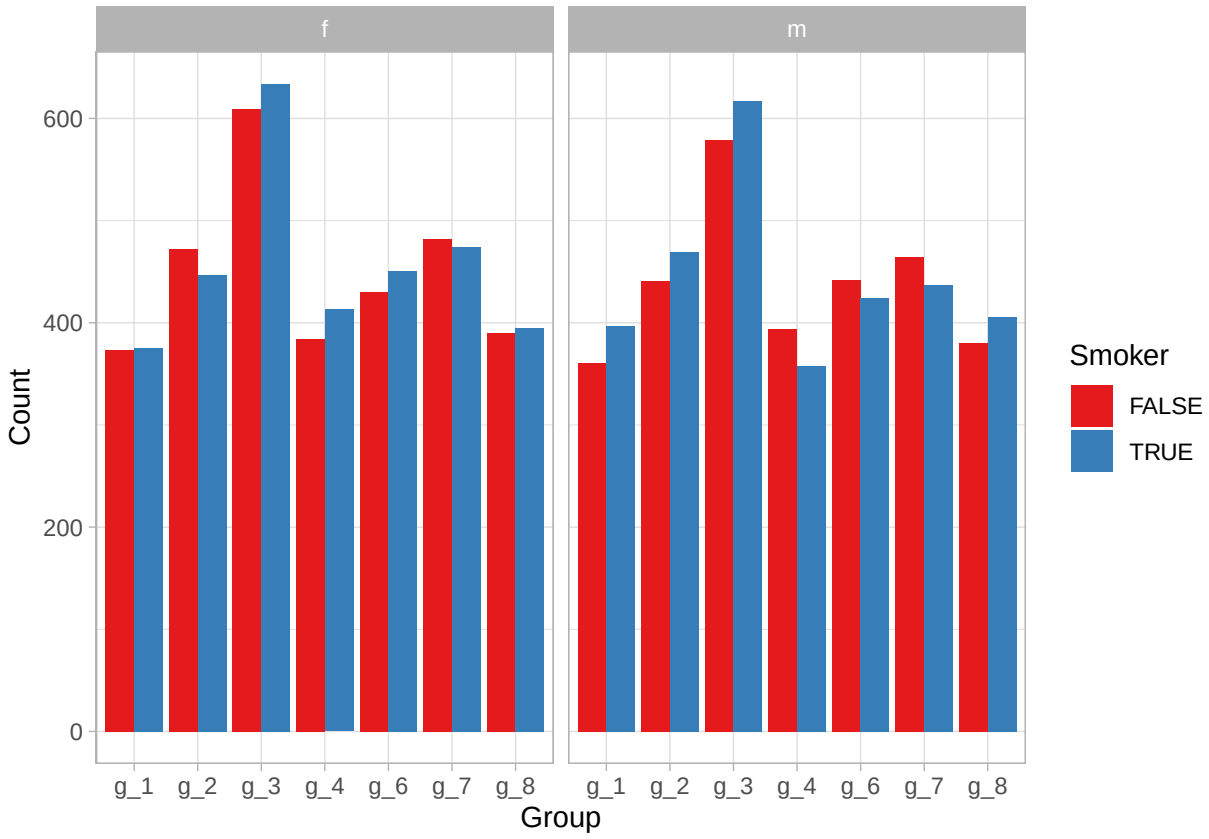
Einfache, schnell zu erfassende Beschreibung der Stichprobe zum Beispiel über eine Tabelle:

```
df %>%
  count(sex, smoker, group) %>%
  pivot_wider(names_from = group,
              values_from = n)
```

```
## # A tibble: 4 x 9
##   sex   smoker   g_1   g_2   g_3   g_4   g_6   g_7
##   <chr> <lgl>  <int> <int> <int> <int> <int> <int>
## 1 f     FALSE    373   472   609   384   430   482
## 2 f     TRUE     375   447   634   413   451   474
## 3 m     FALSE    361   441   579   394   442   464
## 4 m     TRUE     397   469   617   358   424   437
##     g_8
##   <int>
## 1   390
## 2   395
## 3   380
## 4   406
```

Aber vielleicht ein bisschen übersichtlicher in einer Grafik:

```
df %>%
  count(sex, smoker, group) %>%
  ggplot(aes(x = group, fill = smoker, y = n)) +
  geom_col(position = 'dodge') +
  facet_wrap(~sex) +
  labs(x = 'Group',
       y = 'Count',
       fill = 'Smoker')
```



Man sieht auf einen Blick, dass Geschlecht und Raucher ungefähr gleich aufgeteilt wurden, die Gruppen aber wesentlich unterschiedliche Größen aufweisen!

3.2.5 Aussagen über Verteilungseigenschaften

In der `skim`-Ausgabe haben wir ja schon sehen können, dass keine fehlenden Werte vorliegen (`n_missing` war 0).

Wir könnten uns aber noch die Frage stellen, ob Extremwerte in den Gruppen auftauchen, außerdem wollen wir möglichst übersichtlich unsere Verteilungseigenschaften präsentieren.

Das hilft uns zum Einen, um einen besseren Überblick über die Daten zu erhalten, die wir auswerten wollen, zum Anderen hilft es bei einem späteren Bereicht der Leserin, unsere Statistik einzuschätzen. Dazu können wir uns entweder eine Tabelle mit den Verteilungsparametern der numerischen Variablen ausgeben lassen:

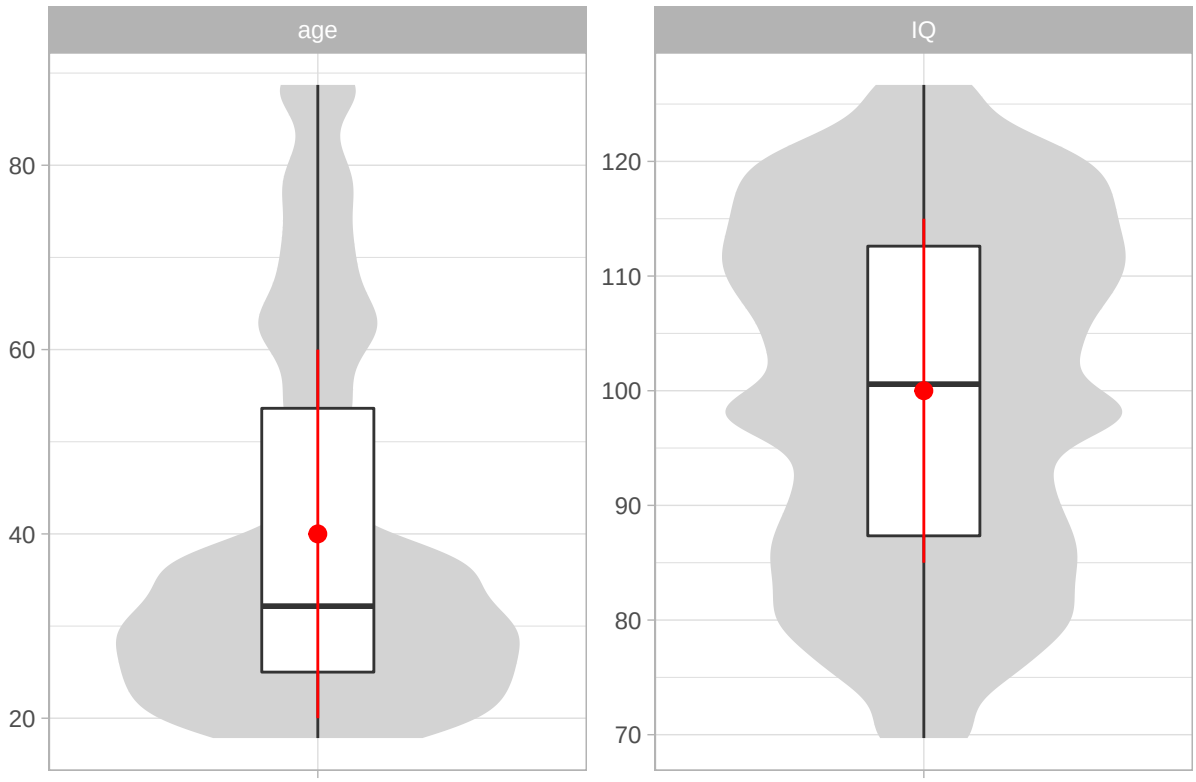
```
df %>%
  pivot_longer(where(is.numeric),
               names_to = 'variable') %>%
  group_by(variable) %>%
  summarise(across(
    value,
    list(
      mean = ~ mean(.),
      sd = ~ sd(.),
      min = ~ min(.),
      q1 = ~ quantile(., .25),
      median = ~ median(.),
      q3 = ~ quantile(., .75),
      max = ~ max(.)
    ),
    .names = '{.fn}'
  )) %>%
  mutate(across(where(is.numeric),
                ~round(.,2)))
```

```
## # A tibble: 2 x 8
##   variable  mean    sd   min    q1 median    q3   max
##   <chr>    <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 age         40    20  17.8  25     32.2  53.6  88.7
## 2 IQ         100    15  69.7  87.3  101.  113.  127.
```

Oder, wieder ein bisschen übersichtlicher, in einem Diagramm. So könnten wir die ganzen Infos gerade zum Beispiel in einem Boxplot mit eingezeichneten Mittelwerten und Streuungsbalken darstellen:

```
df %>%
  pivot_longer(where(is.numeric),
               names_to = 'variable') %>%
  ggplot(aes(x = '', y = value)) +
  geom_violin(fill = 'lightgrey', color = NA) +
  geom_boxplot(width = .25) +
  stat_summary(fun = mean,
               fun.min = function(x)mean(x) - sd(x),
```

```
              fun.max = function(x)mean(x) + sd(x),
              color = 'red') +
facet_wrap(~variable, scales = 'free') +
labs(x = '',
     y = '')
```



Das Alter ist eindeutig schief verteilt, der IQ dafür mehrgipflig.

Nach der von Tukey aufgestellten Regel (Tukey, 1977) haben wir auch keine Ausreißer (dazu auch später mehr).

3.2.6 Darstellung von Effekten und Zusammenhängen

3.2.6.1 Unterschiede

Da wir eine Reihe von Gruppierungsvariablen haben, könnte der erste Impuls sein, sich die Variablen nach diesen Gruppen aufgeteilt darstellen zu lassen, um mögliche Gruppenunterschiede zu verdeutlichen.

Auch hier können wir uns Tabellen erstellen:

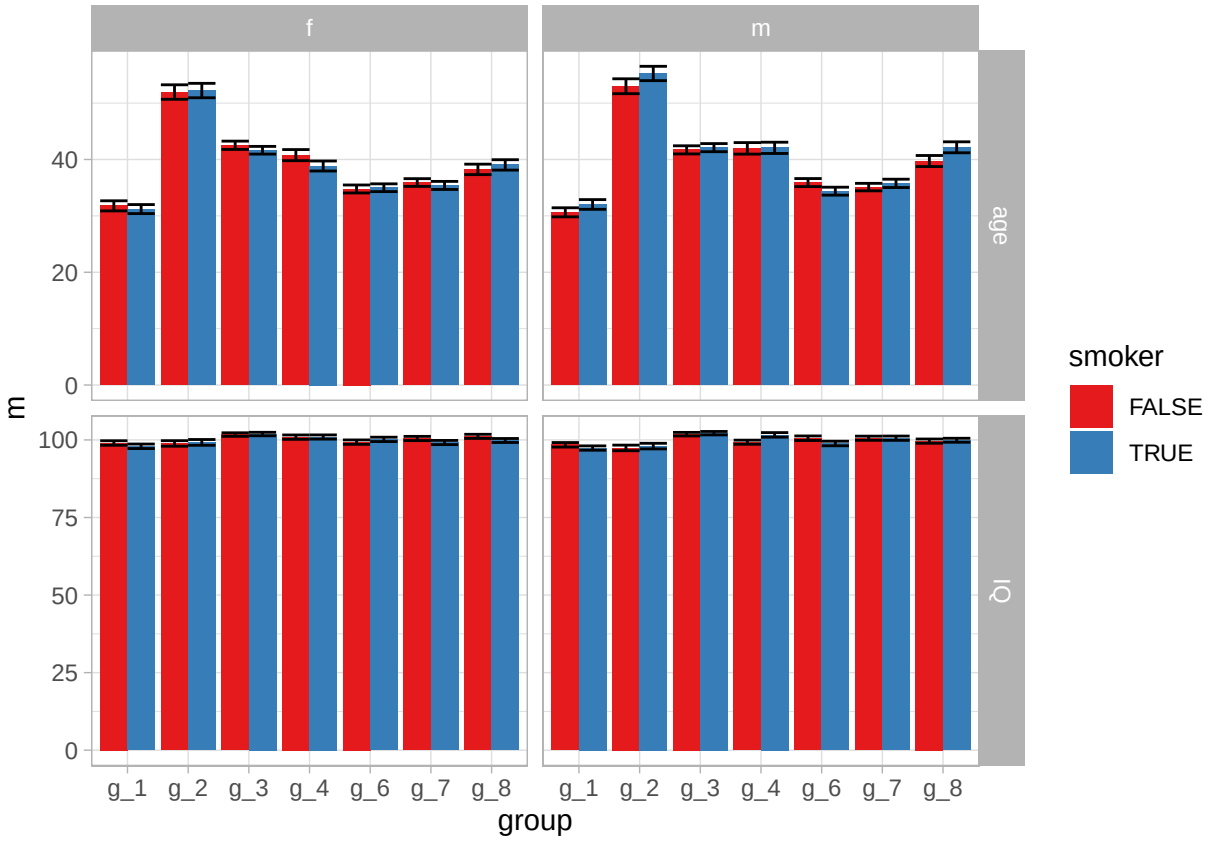
```
df %>%
  group_by(smoker, group, sex) %>%
  summarise(across(where(is.numeric),
                   .fns = list(mean = ~mean(.),
                               sd = ~sd(.),
                               n = ~n()))) %>%
  mutate(across(where(is.numeric),
                ~round(.,2)))
```

```
## # A tibble: 28 x 9
## # Groups:   smoker, group [14]
##    smoker group sex   IQ_mean IQ_sd  IQ_n age_mean
##    <lgl>  <chr> <chr>   <dbl> <dbl> <dbl>    <dbl>
##  1 FALSE  g_1   f        99.0  13.9   373     31.8
##  2 FALSE  g_1   m        98.4  13.9   361     30.6
##  3 FALSE  g_2   f        98.9  19.3   472     52.0
##  4 FALSE  g_2   m        97.4  19.0   441     53.0
##  5 FALSE  g_3   f       102.   14.0   609     42.5
##  6 FALSE  g_3   m       102.   13.7   579     41.7
##  7 FALSE  g_4   f       101.   13.8   384     40.8
##  8 FALSE  g_4   m        99.3  13.1   394     42.0
##  9 FALSE  g_6   f        99.3  14.6   430     34.8
## 10 FALSE  g_6   m       101.   15.4   442     35.9
##    age_sd age_n
##     <dbl> <dbl>
##  1   17.3   373
##  2   15.3   361
##  3   27.9   472
##  4   27.7   441
##  5   18.2   609
##  6   17.4   579
##  7   19.3   384
##  8   20.3   394
##  9   14.6   430
## 10   14.9   442
## # ... with 18 more rows
```

Oder Plots erstellen um die möglichen Unterschiede darzustellen:

```
df %>%
  pivot_longer(cols = where(is.numeric),
               names_to = 'variable') %>%
  group_by(variable, smoker, sex, group) %>%
  summarise(m = mean(value),
            sem = sqrt(var(value)/n()),
            upper = m + sem,
```

```
            lower = m - sem) %>%
ggplot(aes(x = group,
           y = m,
           fill = smoker)) +
geom_col(position = 'dodge') +
geom_errorbar(aes(ymin = lower,
                  ymax = upper),
              position = 'dodge')+
facet_grid( variable ~ sex , scales = 'free')
```



Das wird zwar ein bisschen unübersichtlich (wenn man das wirklich sinnvoll betreiben wollen würde sollte man sich Gedanken dazu machen, welche Variablen tatsächlich von Relevanz sind), man könnte aber zu dem Schluss kommen dass die IQs relativ ähnlich sind, die Altersgruppen aber nicht.

3.2.6.2 Zusammenhänge

Zuletzt wollen wir noch gucken, ob in den Daten irgendwelche (linearen) Zusammenhänge direkt ersichtlich sind. Dazu können wir zuerst Korrelationen berechnen, zum Beispiel einmal für die gesamte Stichprobe und einmal für die Untergruppen:

```
library(magrittr)

df %$%
  cor(age,
      IQ)
```

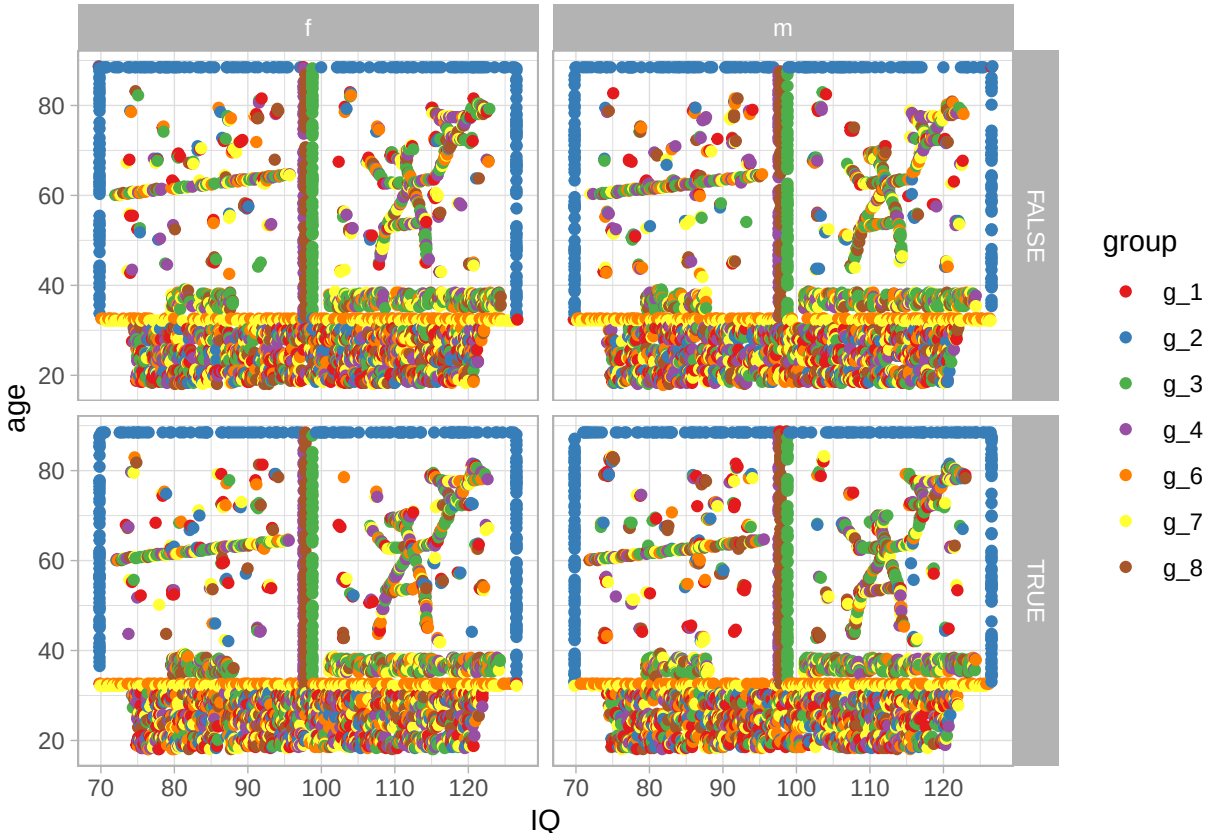
```
## [1] 0.06659135
```

```
df %>%
  group_by(group) %>%
  summarise(r = cor(age, IQ))
```

```
## # A tibble: 7 x 2
##   group       r
##   <chr>   <dbl>
## 1 g_1   0.0540
## 2 g_2   0.00747
## 3 g_3   0.110
## 4 g_4   0.0952
## 5 g_6   0.111
## 6 g_7   0.139
## 7 g_8   0.0636
```

Hier ist so weit nichts auffällig. Ein letzter zu überprüfender Aspekt sind die nicht-linearen Zusammenhänge, zum Beispiel über angemessene Plots. Dies können wir zum Einen für die Untergruppen überprüfen wollen:

```
df %>%
  ggplot(aes(x = IQ,
             y = age,
             color = group)) +
  geom_point() +
  facet_grid(smoker ~ sex)
```



Zum Anderen für die gesamte Stichprobe:

```
df %>%
  ggplot(aes(IQ, age, color = group)) +
  geom_point(size = .001) +
  scale_color_grey()
```

3.3 Data Cleaning

3.3.1 Umgang mit fehlenden Werten

`NA`s sind das in R zur Codierung von fehlenden Werten genutzte Datenformat.

Sie können in Vektoren (und damit auch `tibble`-Spalten) jeden Datenformats auftreten:

```
c(T,NA,F)
```

```
## [1]  TRUE    NA FALSE
```

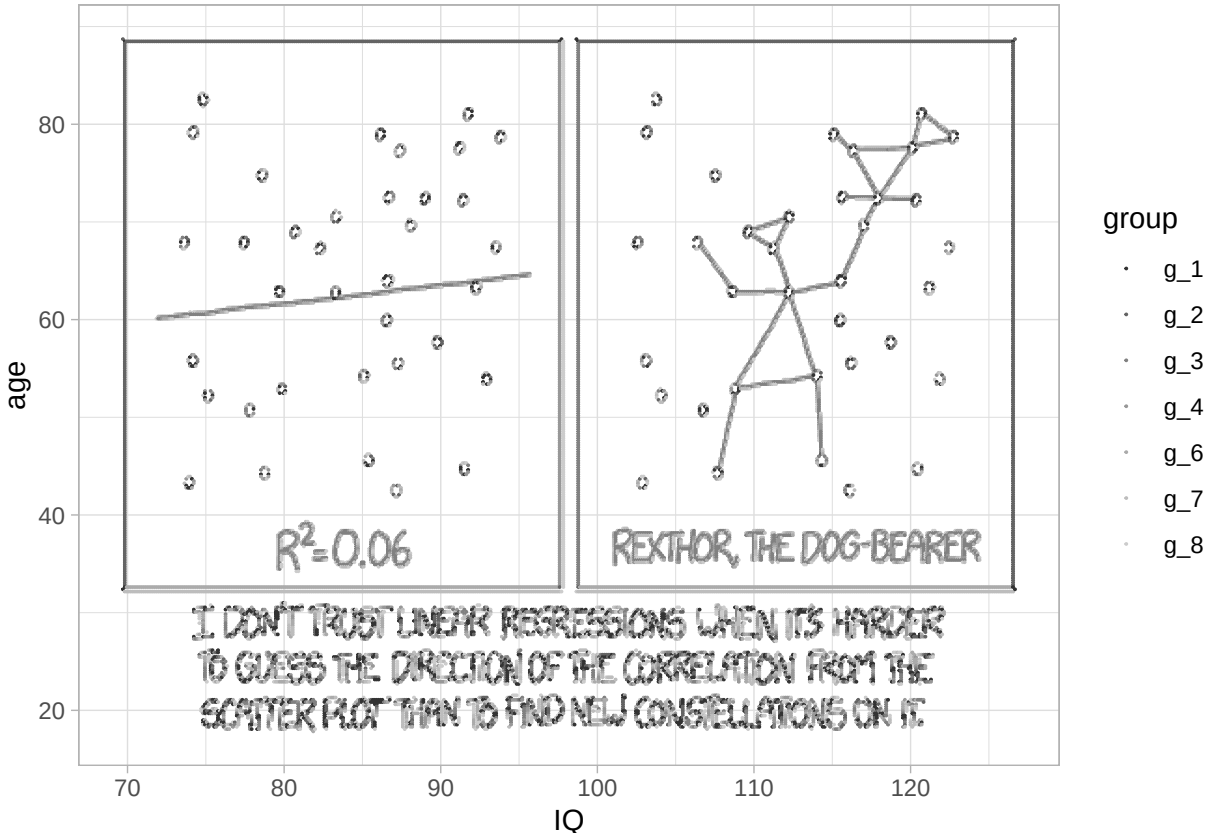


Figure 3.1: Original-Comic von [xkcd](https://xkcd.com/1725/)

```
c(1,NA,2)
```

```
## [1]  1 NA  2
```

```
c('a',NA,'b')
```

```
## [1] "a" NA  "b"
```

Wenn wir versuche mit einem Vektor zu rechnen, der `NA`s beinhaltet, können wir auf Probleme stoßen:

```
aVector <- c(NA,1,2,5,NA,6,8)
length(aVector[aVector > 3])
```

```
## [1] 5
```

```
mean(aVector)
```

```
## [1] NA
```

Mit der `is.na()`-Funktion können wir uns einen logischen Vektor ausgeben lassen, der fehlende Werte codiert. Den können wir dann wie gewohnt benutzen:

```
sum(is.na(aVector))
```

```
## [1] 2
```

```
any(is.na(aVector))
```

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

3.3.2 fehlende Werte und einfache Kennwerte

Die meisten Funktionen im `base R` Umfang haben ein `na.rm`-Argument mit dem wir fehlende Werte von Berechnungen ausschließen können. Das kann an vielen Stellen schon eine sinnvolle Lösung sein, zum Beispiel wenn wir die Infos über die Anzahl fehlender Werte nicht verlieren wollen.

Das könnte dann zum Beispiel so aussehen:

```
mean(aVector)
```

```
## [1] NA
```

```
mean(aVector, na.rm = T)
```

```
## [1] 4.4
```

Dieses Argument können wir auch in die gewohnten Pipelines einsetzen.

Als Beispiel nehmen wir diesen kleinen (unrealistisch unvollständigen) Datensatz `df_2`:

```
(df_2 <- read_csv('data/small_nas.csv'))
```

```
## # A tibble: 12 x 4
##       VP group    t_1    t_2
##    <dbl> <dbl>  <dbl>  <dbl>
##  1     1     1  3.66  -1.53
##  2     2     2 NA      3.32
##  3     3     3  2.23  NA
##  4     4     4  4.82   4.14
##  5     5     1  0.142  0.537
##  6     6     2  1.86   5.04
##  7     7     3 NA      1.46
##  8     8     4  3.60   3.50
##  9     9     1  0.353 NA
## 10    10     2 NA     NA
## 11    11     3  3.79   2.57
## 12    12     4  5.37   4.95
```

Wir könnten jetzt die Pipeline für die Gruppenunterschiede von eben nochmal benutzen, aber um eine Angabe zur Anzahl der fehlenden Werte ergänzen:

```
df_2 %>%
  group_by(group) %>%
  summarise(across(matches('t_'),
                   .fns = list(mean = ~mean(., na.rm = T),
                               sd = ~sd(., na.rm = T),
                               n = ~n(),
                               missing = ~sum(is.na(.))))) %>%
  mutate(across(where(is.numeric),
                ~round(.,2)))
```

```
## # A tibble: 4 x 9
##   group t_1_mean t_1_sd t_1_n t_1_missing t_2_mean
##   <dbl>    <dbl>  <dbl> <dbl>       <dbl>    <dbl>
## 1     1     1.39   1.97     3           0    -0.5
## 2     2     1.86  NA        3           2     4.18
## 3     3     3.01   1.1      3           1     2.01
## 4     4     4.6    0.91     3           0     4.2
##   t_2_sd t_2_n t_2_missing
##    <dbl> <dbl>       <dbl>
## 1   1.46     3           1
## 2   1.22     3           1
## 3   0.79     3           1
## 4   0.73     3           0
```

3.3.3 `NA`-Bereinigung von Datensätzen

Vor statistischen Auswertungen kann es aber einfacher sein, den Datensatz komplett von fehlenden Werten zu bereinigen.

Je nach dem Fall und der Person die man fragt, gibt es verschiedene Vorgehensweisen. Wir gucken uns hier genauer den fallweisen Ausschluss und das ersetzen durch Werte der zentralen Tendenz an.

Die Entscheidung für das Auffüllen oder das Ausschließen muss von Fall zu Fall gefällt werden!

Wenn wir zum Beispiel unseren `df_2` nochmal angucken, fehlt ein Viertel der Werte. Hier die Fälle aufzufüllen und so zu tun als würde man mit 133% der Werte arbeiten, die tatsächlich vorlagen, ist offensichtlich schwierig. Gleichzeitig wird oft das Argument vorgebracht, dass insbesondere diejenigen Versuchspersonen, die in bestimmten Bedingungen keine Antwort produziert haben ein wichtiger Teil der Stichprobe sind und das Auslassen an sich als Form der Antwort betrachtet werden kann. Wenn wir diese Versuchspersonen ausschließen, verzerren wir nach diesem Argument also systematisch unsere Stichprobe.

Wichtig ist also vor jeder Bereinigung, Überlegungen darüber anzustellen, was im gegebenen Fall gerade die angemessenste Lösung darstellt. Bei unserem Datensatz df_2 sind beide Methoden nicht wirklich gut, der Datensatz ist aber auch extrem.

3.3.3.1 Fallweiser Ausschluss

Die radikalste Methode ist der Fallweise Ausschluss, also der Ausschluss aller Eintragungen einer Versuchsperson, die mindestens einen fehlenden Wert vorliegen hat.

Als Erinnerung, hier nochmal unser Datensatz `df_2`:

VP	group	t_1	t_2
1	1	3.6612191	-1.5337696
2	2	NA	3.3223479
3	3	2.2250924	NA
4	4	4.8187796	4.1387727
5	1	0.1419190	0.5373972
6	2	1.8635821	5.0435449
7	3	NA	1.4552116
8	4	3.6001703	3.5013055
9	1	0.3530444	NA
10	2	NA	NA
11	3	3.7877141	2.5676322
12	4	5.3706695	4.9490249

Wir müssten also die Versuchspersonen ausschließen.

Die einfachste Variante dafür ist, den Datensatz ins `wide`-Format zu überführen (wie er es in unserem Fall schon vorliegt) und mit `drop_na` diejenigen Zeilen auszuschließen, die fehlende Werte beinhalten:

```
df_2 %>%
  drop_na()
```

```
## # A tibble: 7 x 4
##      VP group   t_1    t_2
##   <dbl> <dbl> <dbl>  <dbl>
## 1     1     1 3.66  -1.53
## 2     4     4 4.82   4.14
## 3     5     1 0.142  0.537
## 4     6     2 1.86   5.04
## 5     8     4 3.60   3.50
## 6    11     3 3.79   2.57
## 7    12     4 5.37   4.95
```

3.3.3.2 Ersetzen fehlender Werte

Statt radikal alle Fälle auszuschließen, die mindestens einen fehlenden Wert beinhalten, gibt es auch Ansätze, diese aufzufüllen. Gängige Verfahren hier sind die fehlenden Werte hypothesenunabhängig (also nicht gruppenweise) mit dem (getrimmten) Mittelwert, dem Median oder dem Modus der Gesamtstichprobe aufzufüllen. Die Umsetzung in R sieht dann immer gleich aus, die einzige Änderung findet im Kennwert statt, den man zur Ergänzung wählt.

Hier mal ein Beispiel mit dem getrimmten Mittelwert:

```
df_2 %>%
  mutate(across(where(is.numeric),
                ~replace_na(.,
                            mean(.,
                                 na.rm=T,
                                 trim = .05))))
```

```
## # A tibble: 12 x 4
##       VP group   t_1    t_2
##    <dbl> <dbl> <dbl>  <dbl>
##  1     1     1 3.66  -1.53
##  2     2     2 2.87   3.32
##  3     3     3 2.23   2.66
##  4     4     4 4.82   4.14
##  5     5     1 0.142  0.537
##  6     6     2 1.86   5.04
##  7     7     3 2.87   1.46
##  8     8     4 3.60   3.50
##  9     9     1 0.353  2.66
## 10    10     2 2.87   2.66
## 11    11     3 3.79   2.57
## 12    12     4 5.37   4.95
```

3.3.4 Umgang mit Ausreißern

Ausreißerbereinigung sind ein komplexes Thema, über das viel diskutiert werden kann und auch muss.

Da wir uns hier aber im Rahmen einer praktischen Übung befinden sparen wir uns das und nutzen die weit verbreitete Regel, die Tukey (1977) formuliert hat:

Diejenigen Werte sind als Ausreißer zu betrachten, die außerhalb des Intervals

$$Q_1 - 1.5 \cdot \text{IQR} \leq x \leq Q_3 + 1.5 \cdot \text{IQR}$$

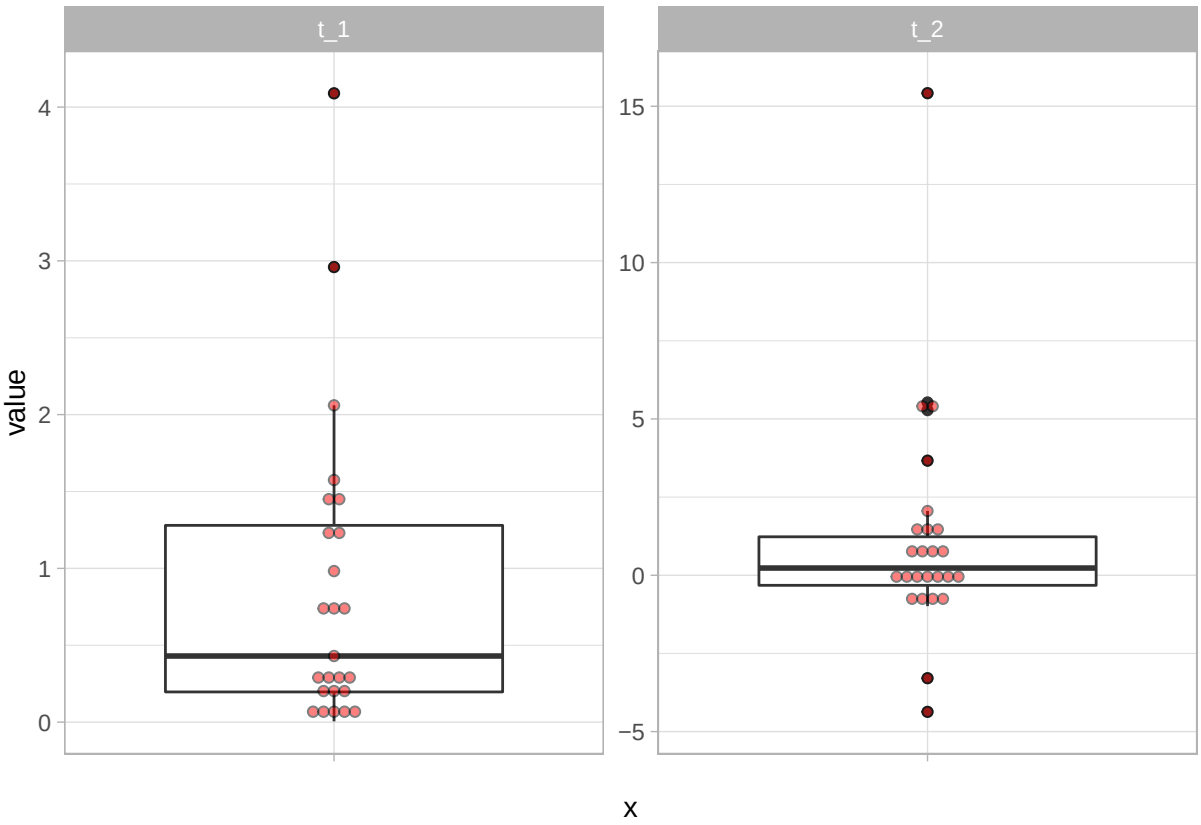
liegen. Diese Regel ist auch der in `geom_boxplot` implementierte Standard, den wir ja auch schon zumindest vom Sehen kennen.

Die Frage ist nun, was wir mit eventuell detektierten Ausreißern machen.

Dafür betrachten wir den folgenden Datensatz `df_3`:

```
df_3 %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(y = value, x = '')) +
  geom_boxplot() +
  geom_dotplot(binaxis = 'y',
               stackdir = 'center',
               alpha = .5,
```

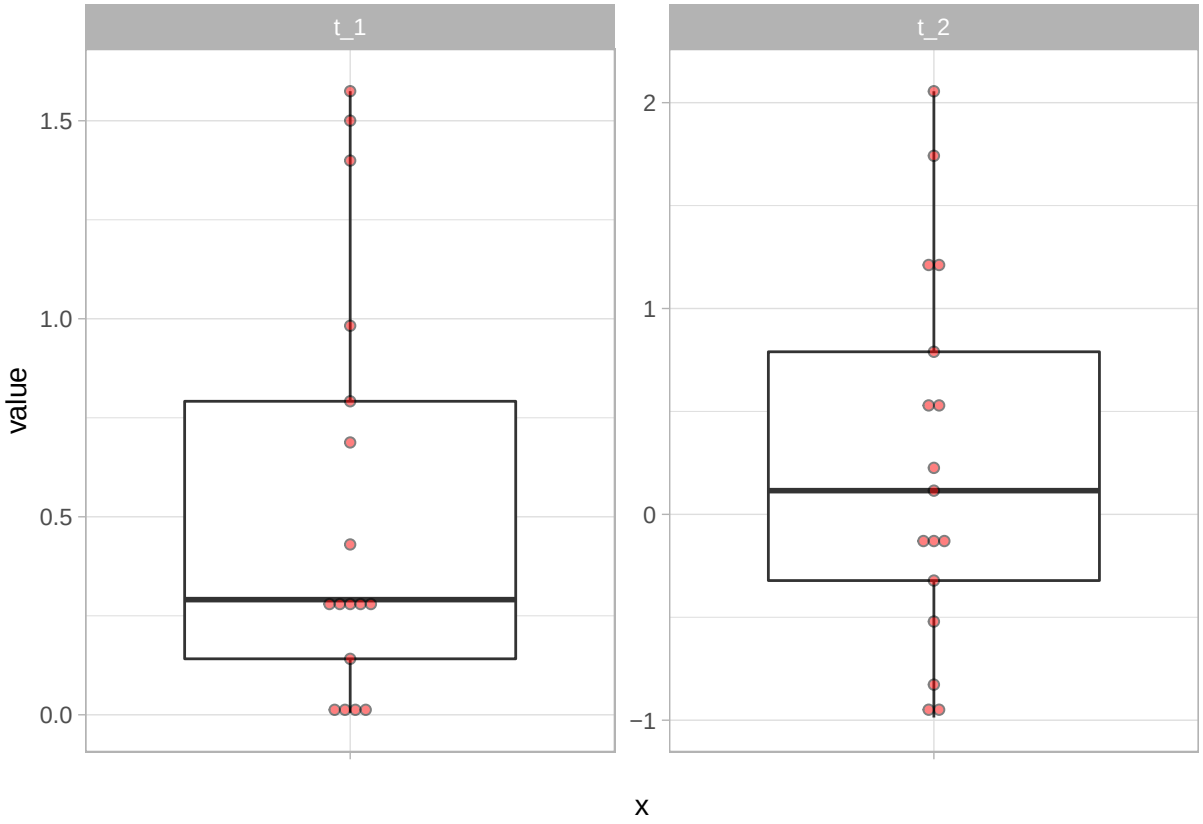
```
                fill = 'red',
                dotsize = .5) +
  facet_wrap(~name,scales = 'free_y')
```



Auch hier können wir wieder zum radikalen Ausschluss greifen und den Datensatz einfach danach filtern, dass unsere Variablen zwischen den "Tukey Fences" liegen:

```
df_3 %>%
  filter(between(t_1,
                 quantile(t_1,.25) - 1.5 * IQR(t_1),
                 quantile(t_1,.75) + 1.5 * IQR(t_1)) &
           between(t_2,
                 quantile(t_2,.25) - 1.5 * IQR(t_2),
                 quantile(t_2,.75) + 1.5 * IQR(t_2))) %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(y = value, x = '')) +
  geom_boxplot() +
  geom_dotplot(binaxis = 'y',
```

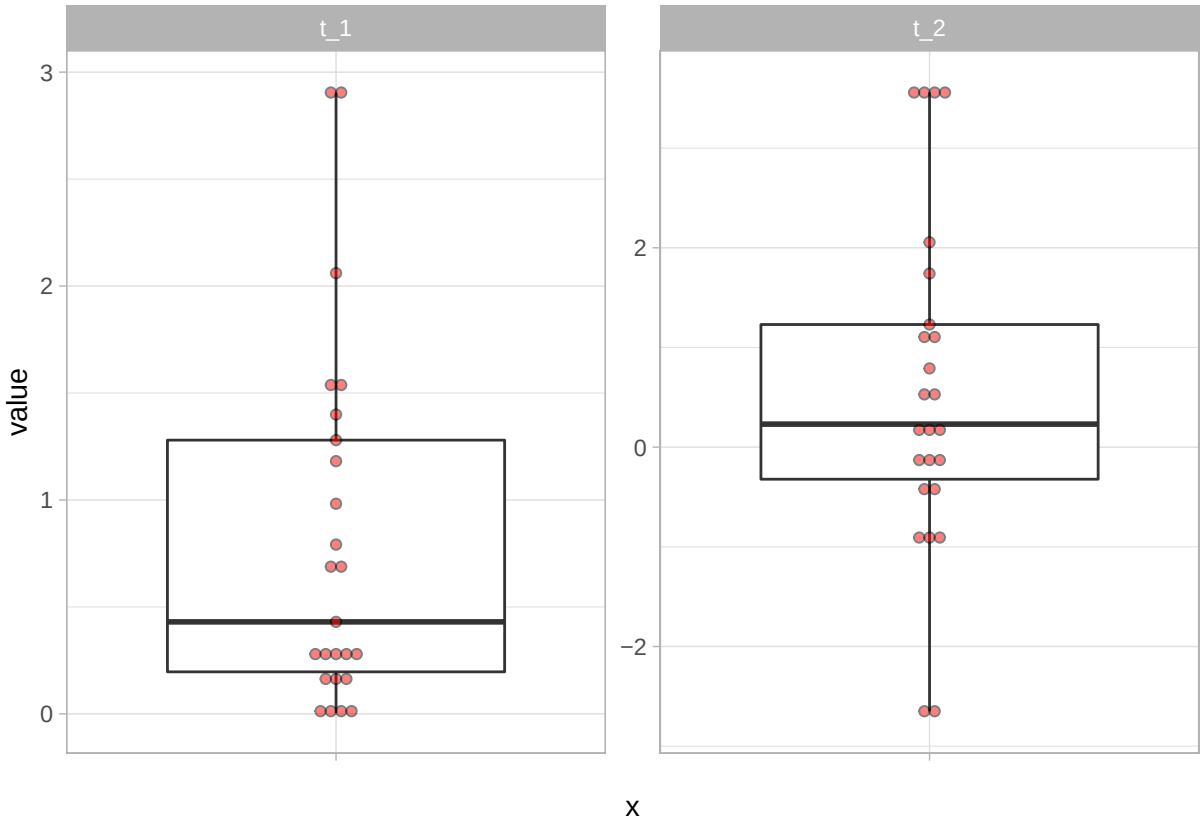
```
                stackdir = 'center',
                alpha = .5,
                fill = 'red',
                dotsize = .5) +
  facet_wrap(~name,scales = 'free_y')
```



Oder wir 'winsorieren' unsere Ausreißer, indem wir sie durch den Wert der jeweiligen Grenze ersetzen. Wir sagen also, dass diejenigen Werte, die außerhalb der Fences liegen auf den Wert der jeweiligen Grenze gesetzt werden sollen:

```
df_3 %>%
  mutate(across(everything(),
                ~ifelse(. >  quantile(.,.75) + 1.5 * IQR(.),
                        quantile(.,.75) + 1.5 * IQR(.),
                        .)),
         across(everything(),
                ~ifelse(. <  quantile(.,.25) - 1.5 * IQR(.),
                        quantile(.,.25) - 1.5 * IQR(.),
                        .))) %>%
```

```
pivot_longer(cols = everything()) %>%
  ggplot(aes(y = value, x = '')) +
  geom_boxplot() +
  geom_dotplot(binaxis = 'y',
               stackdir = 'center',
               alpha = .5,
               fill = 'red',
               dotsize = .5) +
  facet_wrap(~name,scales = 'free_y')
```



3.4 Organisationsformen von Datensätzen

3.4.1 long vs. wide format

3.4.1.0.1 long-format

Name	RT	Bedingung
Snake Müller	2624	1
Snake Müller	3902	2
Snake Müller	6293	3
Vera Baum	1252	1
Vera Baum	2346	2
Vera Baum	4321	3

3.4.1.0.2 wide-format

Name	RT_1	RT_2	RT_3
Snake Müller	2624	3902	6293
Vera Baum	1252	2346	4321

Die `pivot`-Funktionen `pivot_longer` und `pivot_wider` bieten die Möglichkeit, einen Datensatz von einem in das andere Format zu konvertieren.

```
longFormat
```

```
##           Name   RT Bedingung
## 1 Snake Müller 2624         1
## 2 Snake Müller 3902         2
## 3 Snake Müller 6293         3
## 4    Vera Baum 1252         1
## 5    Vera Baum 2346         2
## 6    Vera Baum 4321         3
```

3.4.1.1 long to wide

```
wideFormat <- longFormat %>%
  pivot_wider(names_from = 'Bedingung',
              values_from = 'RT',
              names_prefix = 'RT_')
wideFormat
```

```
## # A tibble: 2 x 4
##   Name          RT_1  RT_2  RT_3
##   <chr>        <dbl> <dbl> <dbl>
## 1 Snake Müller  2624  3902  6293
## 2 Vera Baum     1252  2346  4321
```

3.4.1.2 wide to long

```
longFormat <- wideFormat %>%
  pivot_longer(cols = -1,
               names_prefix = 'RT_',
```

```
               names_to = 'Bedingung',
               values_to = 'RT')
longFormat
```

```
## # A tibble: 6 x 3
##   Name         Bedingung    RT
##   <chr>        <chr>     <dbl>
## 1 Snake Müller 1          2624
## 2 Snake Müller 2          3902
## 3 Snake Müller 3          6293
## 4 Vera Baum    1          1252
## 5 Vera Baum    2          2346
## 6 Vera Baum    3          4321
```

Chapter 4

Hilfsmittel für die Inferenzstatistik

4.1 Hilfsmittel für die Inferenzstatistik

4.1.1 Modellterme

Alle inferenzstatistischen Verfahren im `base-R`-Umfang und viele andere aus Zusatzpaketen nutzen die sogenannte *Formelschreibweise* um Modelle zu definieren. Am Anfang ist die Syntax ein bisschen ungewohnt, am Ende resultiert aus dieser Schreibweise aber eine sehr übersichtliche und schnell erfassbare Modell-Formulierung.

Die Formulierung folgt dabei grundsätzlich dem folgenden System, das sich am Besten analog zu einer mathematischen Funktionsgleichung vorgestellt werden kann. Da das `=` aber schon für Zuweisungen belegt ist, wird es in `formula`-Schreibweise durch eine Tilde (~) ersetzt:

Table 4.1

	modellierte Variable(n)	~	Modellformel
Regression:	Kriterium	~	Prädiktor(en)
Varianzanalyse:	AV	~	UV(s als Faktor(en))

4.1.1.1 Modell-Term

Der Modell-Term auf der rechten Seite der Tilde wird dabei aus einer Reihe von Variablen und Kombinationsoperatoren zusammengesetzt. Zuerst etwas unintuitiv sind diese Operatoren im normalen R-Kontext mit anderen Bedeutungen belegt, in `formula`s funktionieren sie aber so *nicht* Die Operatoren sind die folgenden:

Operator	übliche Bedeutung	Bedeutung in 'formula's
+	Addition	Vorhersageterm hinzufügen
-	Subtraktion	Vorhersageterm ausschließen
<A> : <B>	Sequenz	Interaktion AxB
<A> * <B>	Multiplikation	Effekt von A, B und AxB

Anhand von einer Reihe von Beispielen wird die Formulierung deutlich, dafür führen wir noch kurz eine Hand voll Notationen ein, die meisten davon sind wahrscheinlich nicht überraschend:

Abkürzung	Bedeutung
H_0	Nullhypothese eines statistischen Tests
H_1	Alternativhypothese eines statistischen Tests
UV	unabhängige Variable
AV	abhängige Variable
X_i / Y_i	numerische (Zufalls-) Variable
F_i	kategoriale Varable (Faktor)

4.1.1.2 Regressionsmodelle

`Y ~ X1`: einfache lineare Regression von `Y` auf `X1`

`Y ~ X1 + X2`: multiple lineare Regression von `Y` auf `X1` und `X2`

`Y ~ X1+X2+X1:X2`: multiple lineare Regression von `Y` auf `X1` und `X2` sowie auf den Interaktionsterm von `X1` und `X2`

`Y ~ X1*X2`: multiple lineare Regression von `Y` auf `X1` und `X2` sowie auf den Interaktionsterm von `X1` und `X2`

4.1.1.3 Varianzanalytische Modelle

`Y ~ F1`: einfaktorielle Varianzanalyse

`Y ~ F1 + F2 + F1:F2`: zweifaktorielle Varianzanalyse mit beiden Haupteffekten und der Interaktion

`Y ~ F1 * F2`: auch zweifaktorielle Varianzanalyse mit beiden Haupteffekten und der Interaktion

`Y ~ X1 + F1`: Kovarianzanalyse mit Kovariate X1 und Faktor F1

Innerhalb einer Modellformel können die Terme selbst das Ergebnis der Anwendung von Funktionen auf Variablen sein:

$$\texttt{log}(Y) \sim \texttt{scale}(X)$$

Wenn wir die für die Formulierung genutzten Operatoren für arithmetische Operationen in der Modellformel verwenden wollen, müssen sie mit `I()` eingeschlossen werden um den Kontext klarzumachen:

$$Y \sim \texttt{I}(2 * X)$$

4.1.2 Aufgabe

Welche Hypothese(n) pass(t/en) zu folgender Modellformel:

```
IQ ~ Geschlecht + Raucher
```

- A: Es gibt einen Unterschied zwischen der Intelligenz von Rauchern und Nichtrauchern und zwischen der von Frauen und Männern.
- B: Es gibt einen Unterschied zwischen der Intelligenz von Rauchern und Nichtrauchern und zwischen der von Frauen und Männern sowie einen Unterschied in der Intelligenz zwischen Rauchern und Nichtrauchern, der sich in der Ausprägung zwischen den Geschlechtern unterscheidet.
- C: Es gibt einen Unterschied in der Intelligenz zwischen Rauchern und Nichtrauchern, der sich in der Ausprägung zwischen den Geschlechtern unterscheidet.
- D: Es gibt einen Zusammenhang zwischen Rauchen und Geschlecht auf der einen und Intelligenz auf der anderen Seite.

Lösung

A ist richtig.

Chapter 5

Einfache lineare Zusammenhänge

5.0.1 Datensatz

Für die Tests auf linearen Zusammenhänge werden wir den Datensatz `df_wide` mit den folgenden Variablen benutzen:

Variable	Inhalt
'group'	Treatment-Gruppe
'pre_skill'	motorischer Skill vor dem Treatment
'post_skill'	motorischer Skill nach dem Treatment
'hawie_iq'	Intelligenz-Quotient aus HAWIE
'hawie_wahr_log'	Skalenwert Wahrnehmungsgebundenes logisches Denken aus HAWIE

5.1 Test auf Korrelation

5.1.1 Test-Hintergrund

Die empirische Korrelation zweier gemeinsam normalverteilter Variablen lässt sich daraufhin testen, ob sie mit der H_0 'kein linearer Zusammenhang' (H_0 : $\rho_{X,Y} = 0$) verträglich ist.

Dabei wird genutzt, dass bei Multinormalverteilung und Unkorreliertheit der Variablen X und Y die Teststatistik $t_r = r_{x,y}\sqrt{\frac{n-2}{1-r_{x,y}{}^2}}$ $t-$verteilt ist, mit $n-2$ Freiheitsgraden.

Man testet also die Teststatistik t gegen die t_{n-2} -Verteilung.

Ist der Test signifikant, wird die H_1 angenommen, also dass die 'wahre' Korrelation zwischen X und Y ungleich 0 ist.

Gerichtete Hypothesen lassen sich analog testen.

5.1.2 Test auf Korrelation in R

Man kann den Test in R mit Vektoren als Eingabe...

```
cor.test(df_wide$hawie_iq, df_wide$hawie_wahr_log)
```

```
##
##  Pearson's product-moment correlation
##
## data:  df_wide$hawie_iq and df_wide$hawie_wahr_log
## t = 8.0781, df = 48, p-value = 1.678e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6094958 0.8564580
## sample estimates:
##       cor
## 0.7590665
```

... und mit Modellformel als Eingabe aufrufen.

```
cor.test(~ hawie_iq + hawie_wahr_log, data = df_wide)
```

```
##
##  Pearson's product-moment correlation
##
## data:  hawie_iq and hawie_wahr_log
## t = 8.0781, df = 48, p-value = 1.678e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6094958 0.8564580
## sample estimates:
##       cor
## 0.7590665
```

Das `alternative`-Argument bietet die Möglichkeit, die Richtung des Signifikanztests anzugeben.

Dabei steht `'greater'` für einen rechtsseitigen, `'lesser'` für einen linksseitigen und der Standard `'two.sided'` für einen zweiseitigen Test.

```
cor.test(~ hawie_iq + hawie_wahr_log,
         data = df_wide,
         alternative='greater')
```

```
##
```

```
##  Pearson's product-moment correlation
##
## data:  hawie_iq and hawie_wahr_log
## t = 8.0781, df = 48, p-value = 8.392e-11
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
##  0.6375781 1.0000000
## sample estimates:
##       cor
## 0.7590665
```

Der Output lässt sich noch ein bisschen schicker mit der `tidy`-Funktion aus dem `broom`-Paket darstellen (ist auch im `tidyverse` enthalten):

```
cor.test(~ hawie_iq + hawie_wahr_log,
         data = df_wide,
         alternative='greater') %>%
  broom::tidy()
```

```
## # A tibble: 1 x 8
##   estimate statistic  p.value parameter conf.low
##      <dbl>     <dbl>    <dbl>     <int>    <dbl>
## 1    0.759      8.08 8.39e-11        48    0.638
##   conf.high method
##       <dbl> <chr>
## 1         1 Pearson's product-moment correlation
##   alternative
##   <chr>
## 1 greater
```

5.1.3 Aufgabe

Wie kann ich das Ergebnis interpretieren?

```
cor.test(~ hawie_iq + hawie_wahr_log,
         data = df_wide,
         alternative='greater') %>%
  broom::tidy()
```

```
## # A tibble: 1 x 8
##   estimate statistic  p.value parameter conf.low
##      <dbl>     <dbl>    <dbl>     <int>    <dbl>
## 1    0.759      8.08 8.39e-11        48    0.638
##   conf.high method
##       <dbl> <chr>
## 1         1 Pearson's product-moment correlation
##   alternative
##   <chr>
```

```
## 1 greater
```

- A: Die Logik-Leitung beeinflusst den IQ signifikant positiv.
- B: Es gibt keine Korrelation zwischen Logik-Leistung und IQ.
- C: Die Logik-Leistung und der IQ sind signifikant von Null unterschiedlich korreliert.
- D: Es gibt einen signifikanten, positiv linearen Zusammenhang zwischen Logik-Leistung und IQ.

Lösung

C und D könnte man so sagen, D hat aber natürlich mehr Informationsgehalt.

5.2 Einfache lineare Regression

5.2.1 Modellanpassung

Bei der einfachen linearen Regression werden anhand der paarweise vorhandenen Daten zweier Variablen X und Y die Parameter a und b der Vorhersagegleichung $\hat{Y} = bX + a$ so bestimmt, dass die Werte von Y (dem Kriterium) bestmöglich mit der Vorhersage $\hat{Y}$ aus den Werten von X (dem Prädiktor) übereinstimmen.

Als Maß für die Güte der Vorhersage wird die Summe der quadrierten Residuen $E = Y - \hat{Y}$, also der Abweichungen von vorhergesagten und Kriteriumswerten herangezogen.

Lineare Modelle wie das der Regression lassen sich mit lm() anpassen und so die Parameter a und b schätzen.

```
lm(formula= <Modellformel> , data=<Datensatz>)
```

Ein von lm() zurückgegebenes Objekt stellt ein deskriptives Modell der Daten dar, das in anderen Funktionen weiter verwendet werden kann.

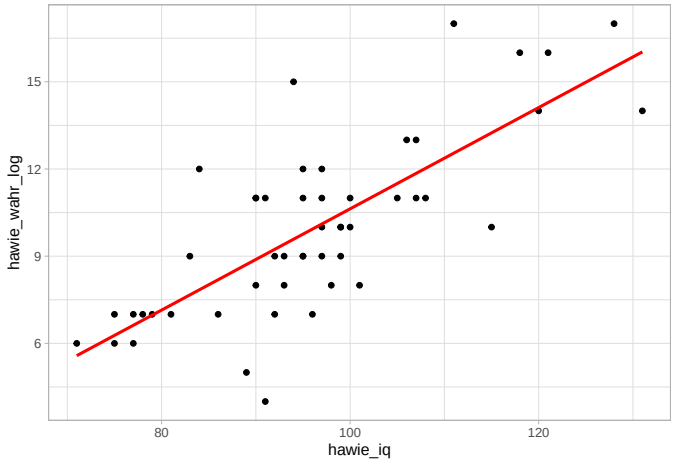
5.2.1.1 Beispiel für deskriptive Modellanpassung

Als Beispiel soll die Leistung auf der Skala zum wahrnehmungsgebundenen logischen Denken als Kriterium mit dem IQ als Prädiktor vorhergesagt werden.

```
(fitI <- lm(hawie_wahr_log ~ hawie_iq, data = df_wide))
```

```
##
## Call:
## lm(formula = hawie_wahr_log ~ hawie_iq, data = df_wide)
##
## Coefficients:
## (Intercept)     hawie_iq
##     -6.7909       0.1742
```

```
ggplot(df_wide, aes(x = hawie_iq, y = hawie_wahr_log)) +
  geom_point() +
  geom_smooth(formula =  y ~ x ,
              method = 'lm',col ='red',se = F)
```



5.2.2 β-Gewicht

Will man statt des b-Gewichtes das standardisierte β-Gewicht angeben, muss in der Modellformel z-transformiert werden.

Dafür könenn wir entweder alle Teile der `formula` `scale`n:

```
(fitZ <- lm(scale(hawie_wahr_log) ~ scale(hawie_iq),
            data = df_wide))
```

```
##
## Call:
## lm(formula = scale(hawie_wahr_log) ~ scale(hawie_iq), data = df_wide)
##
## Coefficients:
##     (Intercept)  scale(hawie_iq)
##      -4.971e-16        7.591e-01
```

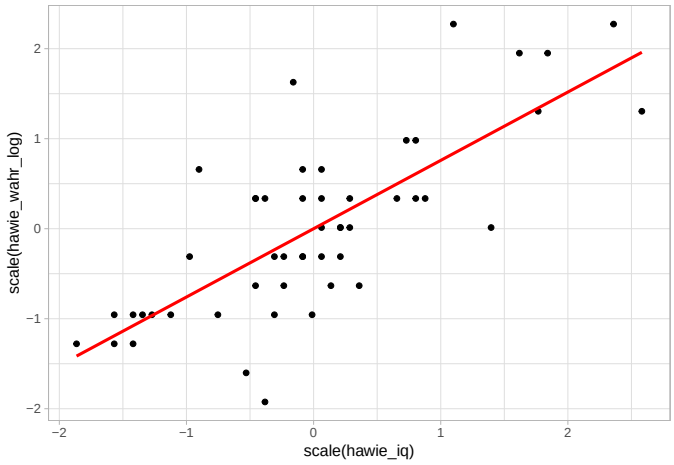
Oder wir benutzen die Index-pipe `%$%` aus dem `magrittr`-Paket und ein zwischengeschaltetes `mutate`, um die Skalierung ein bisschen übersichtlicher zu gestalten:

```
library(magrittr)
fitZ <- df_wide %>%
  mutate(hawie_wahr_log = scale(hawie_wahr_log),
         hawie_iq = scale(hawie_iq)) %$%
  lm(hawie_wahr_log ~ hawie_iq)
```

```
fitZ
```

```
##
## Call:
## lm(formula = hawie_wahr_log ~ hawie_iq)
##
## Coefficients:
## (Intercept)     hawie_iq
##  -4.971e-16    7.591e-01
```

```
df_wide %>%
  mutate(hawie_wahr_log = scale(hawie_wahr_log),
         hawie_iq = scale(hawie_iq)) %>%
  ggplot(aes(x = scale(hawie_iq),
                       y = scale(hawie_wahr_log))) +
    geom_point() +
    geom_smooth(formula =  y ~ x ,
                method = 'lm',col ='red',se = F)
```



5.2.3 weitere Parameter

`lm()` gibt eine Liste zurück, die ein deskriptives Modell der Daten darstellt.

R bietet weitere Funktionen um einzelne Parameter dieses Outputs auszulesen.

Zum Beispiel: `residuals()` zum Anzeigen der Residuen, `coef()` zur Ausgabe der geschätzten Modellparameter und `fitted()` für die vorhergesagten Werte.

```
residuals(fitZ)
```

```
##           1           2           3           4
## -0.132244120  0.553125355 -0.286404915  0.288579211
```

```
##             5             6             7             8
##  1.748985446 -0.244745860 -0.946823542  0.371897320
##             9            10            11            12
##  0.611492617 -0.075993249 -0.036450587 -0.357247601
##            13            14            15            16
## -0.034334195 -0.736411877  1.438547463 -0.469749341
##            17            18            19            20
## -0.146835935  0.682335303 -0.244745860 -0.905164488
##            21            22            23            24
## -1.634309409 -0.330180362 -0.200970413 -0.161427751
##            25            26            27            28
##  0.121942993  0.723994358 -0.721820061  0.682335303
##            29            30            31            32
## -0.146835935  0.136534808  0.721877965 -0.088468673
##            33            34            35            36
## -0.455157526  0.430264583 -1.198894262 -0.655210160
##            37            38            39            40
## -0.203086806 -0.103060488  0.234444733 -0.273929492
##            41            42            43            44
##  0.119826600  0.401080952 -0.384314840  0.065692122
##            45            46            47            48
##  1.342753931 -1.046849860  0.626084433  0.009441252
##            49            50
##  0.482282669  0.428148191
## attr(,"scaled:center")
## [1] 9.96
## attr(,"scaled:scale")
## [1] 3.096805
```

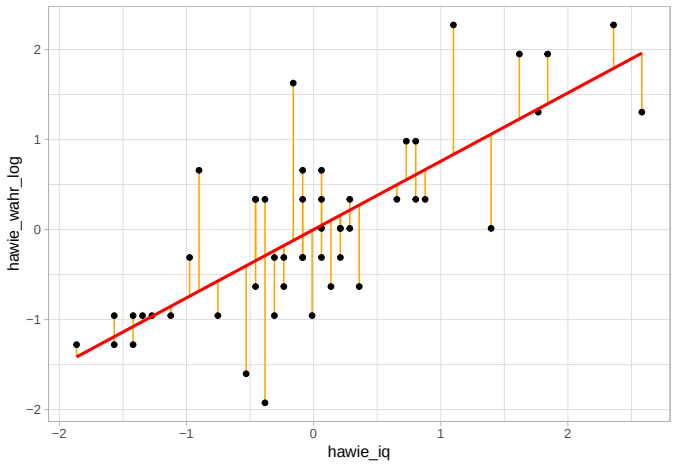
Im `broom`-Paket gibt es außerdem die `augment`-Funktion, die uns den zum Fitten genutzten Datensatz mit einer Reihe von Zusatzinfos ausgibt.

```
broom::augment(fitZ)
```

```
## # A tibble: 50 x 8
##    hawie_wahr_log[,1] hawie_iq[,1]  .fitted  .resid
##                 <dbl>        <dbl>    <dbl>   <dbl>
##  1             -0.310      -0.234  -0.178   -0.132
##  2              1.95        1.84    1.40     0.553
##  3             -0.633      -0.456  -0.347   -0.286
##  4              0.336       0.0622  0.0473   0.289
##  5              1.63       -0.160  -0.122    1.75
##  6             -0.310      -0.0860 -0.0653  -0.245
##  7             -0.956      -0.0119 -0.00900 -0.947
##  8              0.982       0.803   0.610    0.372
##  9              0.659       0.0622  0.0473   0.611
## 10             -0.310      -0.308  -0.234   -0.0760
```

```
##      .hat .sigma  .cooksd .std.resid
##     <dbl>  <dbl>    <dbl>      <dbl>
##  1 0.0211  0.664 0.000445     -0.203
##  2 0.0892  0.659 0.0380        0.881
##  3 0.0243  0.663 0.00241      -0.441
##  4 0.0201  0.663 0.00201       0.443
##  5 0.0205  0.613 0.0756        2.69
##  6 0.0202  0.664 0.00145      -0.376
##  7 0.0200  0.650 0.0216       -1.45
##  8 0.0332  0.662 0.00567       0.575
##  9 0.0201  0.659 0.00904       0.939
## 10 0.0219  0.665 0.000153     -0.117
## # ... with 40 more rows
```

```
fitZ %>%
  broom::augment() %>%
  ggplot(aes(x = hawie_iq,
             y = hawie_wahr_log))+
  geom_linerange(aes(ymin = .fitted, ymax = hawie_wahr_log), col = 'orange') +
    geom_point() +
    geom_smooth(formula =  y ~ x ,
                method = 'lm',col ='red',se = F)
```



5.3 Regressionsanalyse

5.3.1 Test-Hintergrund

Unter Voraussetzungen von Varianzhomogenität und Normalverteilung der Y-Werte für jeden möglichen Wert von X können Regressionskoeffizienten ähnlich wie Korrelationskoeffizienten auf Unterschiedlichkeit von 0 getestet werden.

Dazu wird genutzt, dass der Term $t = \frac{b}{\frac{s_{Y|X}}{s_X\sqrt{N-1}}}$ t_{N-1}-verteilt ist, wenn das tatsächliche b^* nicht unterschiedlich von 0 ist und für jeden Wert von X Y normalverteilt ist mit $\mu = b^*X + a^*$ und einer Varianz σ^2. Die Nullhypothese ist also $H_0 : b^* = 0$.

5.3.2 Test in R

Um zusätzliche Informationen (insbesondere inferenzstatistische Kennwerte) eine mit `lm()` erstellten Regressions-Modells zu erhalten, kann einfach `summary()` verwendet werden.

```
fitZ %>%
  summary()
```

```
##
## Call:
## lm(formula = hawie_wahr_log ~ hawie_iq)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.63431 -0.31924 -0.08223  0.42138  1.74899
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.971e-16  9.302e-02   0.000        1
## hawie_iq     7.591e-01  9.397e-02   8.078 1.68e-10
##
## (Intercept)
## hawie_iq    ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6578 on 48 degrees of freedom
## Multiple R-squared:  0.5762, Adjusted R-squared:  0.5674
## F-statistic: 65.26 on 1 and 48 DF,  p-value: 1.678e-10
```

Außerdem gibt es auch hier einen hübschen `broom`-Output, für den wir mit `conf.int` angeben können, Konfidenzintervalle für die Regressionsgewichte ausgeben lassen zu wollen:

```
fitZ %>%
  broom::tidy(conf.int = T)
```

```
## # A tibble: 2 x 7
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
```

```
## 1 (Intercept) -4.97e-16    0.0930 -5.34e-15 1.00e+ 0
## 2 hawie_iq     7.59e- 1    0.0940  8.08e+ 0 1.68e-10
##   conf.low conf.high
##      <dbl>     <dbl>
## 1   -0.187     0.187
## 2    0.570     0.948
```

5.3.3 Aufgabe

```
fitZ %>%
  broom::tidy(conf.int = T)
```

```
## # A tibble: 2 x 7
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) -4.97e-16    0.0930 -5.34e-15 1.00e+ 0
## 2 hawie_iq     7.59e- 1    0.0940  8.08e+ 0 1.68e-10
##   conf.low conf.high
##      <dbl>     <dbl>
## 1   -0.187     0.187
## 2    0.570     0.948
```

Wie lässt sich das Ergebnis interpretieren?

- A: Höhere IQ-Werte hängen mit höheren Logik-Leistungswerten zusammen.
- B: Es gibt keine Korrelation zwischen IQ-Werten und Logik-Leistung.
- C: Mit jedem Anstieg des IQ um eine Streuungs-Einheit, steigt die Vorhersage um 0.7590665 Streuungs-Einheiten.
- D: Man kann wegen der unzureichenden Berücksichtigung nicht-linearer Zusammenhänge keine Aussage treffen.

Lösung

C könnte man so sagen.

Chapter 6

Lineare Zusammenhänge II

6.0.1 Datensatz

Wir benutzen wieder den Datensatz `df_wide` aus der letzten Woche. Hier nochmal eine Übersicht:

Variable	Inhalt
'group'	Treatment-Gruppe
'pre_skill'	motorischer Skill vor dem Treatment
'post_skill'	motorischer Skill nach dem Treatment
'hawie_iq'	Intelligenz-Quotient aus HAWIE
'hawie_wahr_log'	Skalenwert wahrnehmungsgebundenes logisches Denken aus HAWIE

6.1 Multiple Lineare Regression

6.1.1 Verfahren

Bei der multiplen linearen Regression dienen mehrere quantitative oder dichotome Variablen X_j als Prädiktoren zur Vorhersage des quantitativen Kriteriums Y. Die Vorhersagegleichung hat hier die Form $\hat{Y} = a + b_1X_1 + ... + b_jX_j + ... + b_pX_p$, wobei die Koeffizienten a und b_j auf Basis der empirischen Daten zu ermitteln sind.

Als R-`formula` sieht das wie folgt aus:

$$\text{Kriterium} \sim \text{Prädiktor}_1 + \cdots + \text{Prädiktor}_p$$

Man versucht also, eine stetige Variable durch eine Linearkombination mehrerer Variablen vorherzusagen. Dabei ist aber meistens eher das Ausmaß des Zusammenhangs als die tatsächliche Vorhersage für neue Werte interessant.

6.1.2 Deskriptive Modellanpassung und Regressionsanalyse

Regression vom motorischen Skill nach dem Training auf IQ und wahrnehmungsgebundenes logisches Denken:

```
fit_post_il <- df_wide %>%
   lm(post_skill~hawie_iq+hawie_wahr_log,
      data=.)

fit_post_il
```

```
##
## Call:
## lm(formula = post_skill ~ hawie_iq + hawie_wahr_log, data = .)
##
## Coefficients:
##    (Intercept)        hawie_iq  hawie_wahr_log
##        5.75745        -0.05013         0.42397
```

Für standardisierte Gewichte wie vorher:

```
library(magrittr)
fit_post_il_z <- df_wide %>%
  mutate(across(where(is.numeric), ~scale(.))) %$%
   lm(post_skill~hawie_iq+hawie_wahr_log)
```

6.1.3 Darstellung

Mit der Funktion `scatter3d()` aus dem Paket `car` lassen sich die Daten dann dreidimensional mit Residuen plotten.

```
library(car)

scatter3d(post_skill~hawie_iq+hawie_wahr_log,
          data=df_wide)
```

Ein bisschen komplizierter geht das auch mit dem Paket `plotly`. Dafür muss zuerst ein Datensatz erstellt werden, der den gesamten Raum der im darzustellenden Datensatz vorliegenden Prädiktoren abdeckt:

```
dummy <-  df_wide %>%
  select(hawie_iq, hawie_wahr_log) %>%
  expand.grid()
```

Für alle diese Kombinationen müssen wir jetzt mit unserem Modell eine Vorhersage treffen:

```
dummy$post_skill <- predict(fit_post_il, dummy)
```

Und diesen Datensatz können wir jetzt mit `plotly` darstellen:

```
library(plotly)

plot_ly(data = df_wide,
        x = ~hawie_iq,
        y = ~hawie_wahr_log,
        z = ~post_skill,
        type = 'scatter3d',
        mode = 'markers') %>%
  add_mesh(data = dummy)
```

6.2 Regressionsdiagnostik

6.2.1 Regressionsdiagnostik

Regressionsgleichungen sind sehr anfällig für verschiedene Klassen von Ausreißern. Ein Extremwert kann unter bestimmten Bedingungen die errechneten Koeffizienten extrem beeinflussen und verzerren.

Es gibt drei Klassen von diagnostischen Werten, die unterschiedliche Aspekte möglicher Verfälschung beleuchten.

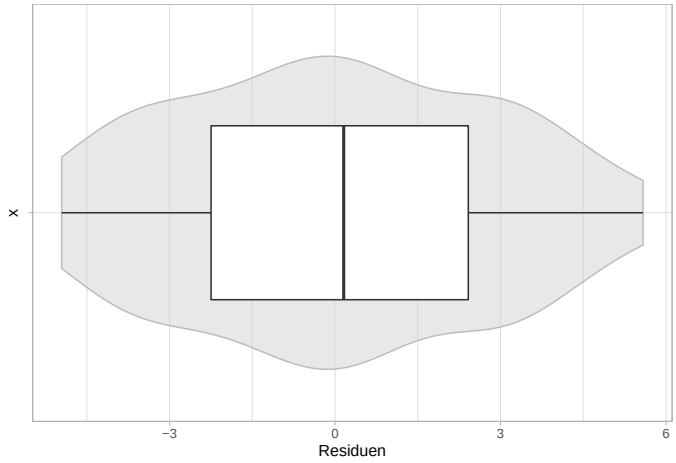
1. Abstand (mögliche Ausreißer im Wertebereich des Kriteriums)
2. Hebelwirkung (mögliche Ausreißer im Wertebereich der Prädiktoren)
3. Einfluss (Kombination von Abstand und Hebelwirkung)

http://omaymas.github.io/InfluenceAnalysis/ gibt es eine Shiny-App, mit der man daran rumspielen kann.

6.2.2 Abstand

Die Plausibilität des Abstandes lässt sich am Besten mit Hilfe der Residuen der Regression überprüfen. Dafür benutzen wir hier eine grafische Darstellung, um einen Überblick über deren Verteilung zu erlangen.

```
df_wide %>%
  mutate(Residuen = residuals(fit_post_il)) %>%
ggplot(aes(y = Residuen, x = '') ) +
  geom_violin(color = 'grey', fill = 'lightgrey',
              alpha = .5) +
  geom_boxplot(width = .5) +
  coord_flip()
```



6.2.3 Hebelwirkung

Hebelwirkung ist das Ausmaß, in dem ein Prädiktor-Wert ungewöhnlich in Bezug auf die restlichen Prädiktorwerte ist. Das für Aussagen darüber genutzte Maß sind die Hebelwerte. Bei der einfachen Regression sind Hebelwerte die Abweichung der einzelnen Prädiktorwerte von deren Mittelwert. Bei der multiplen Regression ist das nicht ganz so einfach.

In beiden Fällen bewegt sich der Hebelwert aber zwischen $\frac{1}{N}$ und 1.

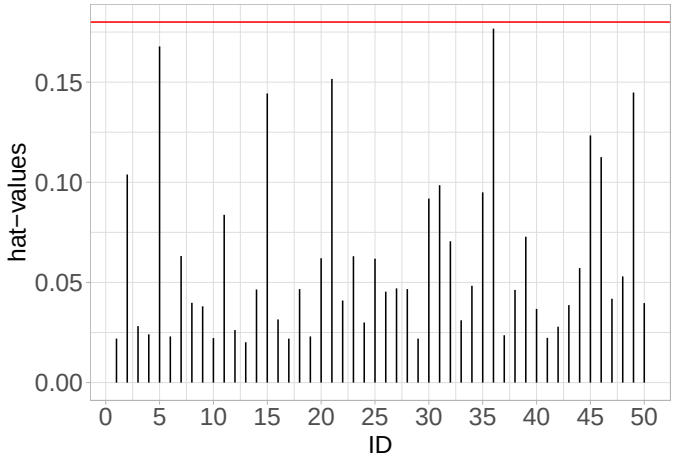
Diese Hebelwerte werden mit `hatvalues()` berechnet.

Faustregel: Bei p Einflussgrößen (Prädiktoren) und N Beobachtungen sind Fälle mit Hebelwerten von größer als $3 \cdot \frac{p+1}{N}$ problematisch.

```
h <- hatvalues(fit_post_il)
```

Mit einem Spikeplot lassen sich diese dann veranschaulichen.

```
tibble(hats = h,
       ID = 1:50) %>%
  ggplot(aes(x= ID, ymax = h, ymin= 0)) +
  geom_linerange() +
  geom_hline(yintercept = (3*(2+1)/50),col='red') +
  scale_x_continuous(breaks = seq(0,50,5)) +
  labs(y = 'hat-values')
```

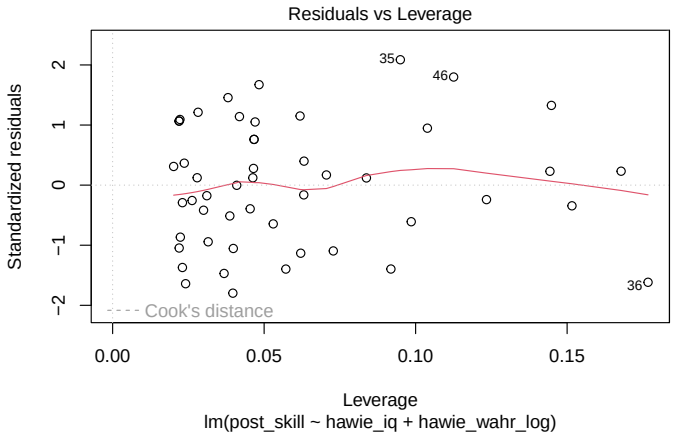


6.2.4 Einfluss

Unter Einfluss oder *influence* versteht man den Einfluss eines einzelnen Datenpunktes auf die gesamte Vorhersage. Er stellt also eine Kombination der vorher genannten Parameter dar.

Am besten lässt sich dieser mit folgender Funktion grafisch überprüfen:

```
plot(fit_post_il, which = 5)
```



Ganz hübsche Alternativen bieten auch die Funktion `influencePlot()` aus dem `car`-Paket:

```
influencePlot(fit_post_il)
```

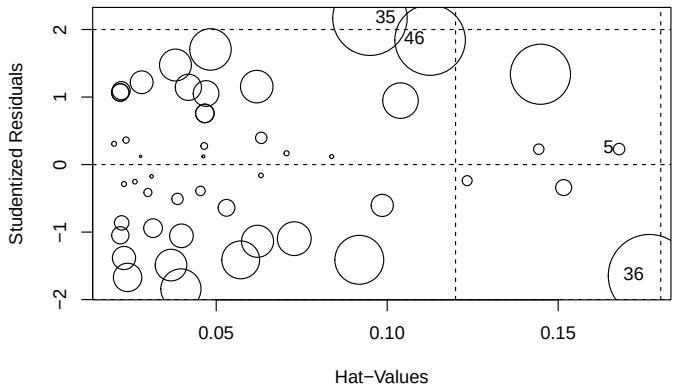


Table 6.1

StudRes	Hat	CookD
0.229	0.168	0.00361
2.17	0.095	0.152
-1.65	0.177	0.187
1.84	0.113	0.137

Außerdem können wir uns auch hier das `broom`-Paket benutzen, diesmal um uns alle diagnostischen Werte in einem praktischen Datensatz ausgeben zu lassen:

```
fit_post_il %>%
  broom::augment()
```

6.2.5 Test der Regressionskoeffizienten

Auch bei der multiplen linearen Regression lassen sich die Regressionskoeffizienten der einzelnen Prädiktoren jeweils auf die Nullhypothese testen, dass die jeweiligen "wahren Koeffizienten" ${b_i}^*$ gleich 0 sind. Die Schätzung der Streuung der Koeffizienten ist ein bisschen komplizierter als im einfachen Fall, deswegen sei hier nur erwähnt, dass es geht.

Die für jeden Prädiktoren gebildete Teststatistik $t = \frac{b_i}{s_{b_i}}$ ist dann bei Gültigkeit der Nullhypothese ($H_0 : {b_i}^* = 0$) t_{N-p-1}-verteilt, wobei p die Gesamtzahl der Prädiktoren und N die Gesamtzahl der Beobachtungen ist.

Der Test der Parameter auf Signifikanz läuft wie im einfachen Fall mit der Funktion `summary()`

```
summary(fit_post_il_z)
```

```
##
## Call:
## lm(formula = post_skill ~ hawie_iq + hawie_wahr_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.70753 -0.77446  0.05423  0.83241  1.92405
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.812e-17  1.371e-01   0.000   1.0000
## hawie_iq       -2.331e-01  2.127e-01  -1.096   0.2787
## hawie_wahr_log  4.524e-01  2.127e-01   2.127   0.0387
##
## (Intercept)
## hawie_iq
## hawie_wahr_log *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9692 on 47 degrees of freedom
## Multiple R-squared:  0.09891,    Adjusted R-squared:  0.06057
## F-statistic:  2.58 on 2 and 47 DF,  p-value: 0.0865
```

Oder auch wieder mit `broom::tidy()`:

```
broom::tidy(fit_post_il_z)
```

6.2.5.1 Aufgabe

Wie lässt sich das Ergebnis interpretieren?

1. Der IQ liefert einen signifikanten Beitrag zur Vorhersage des motorischen Skills nach dem Training.
2. Es gibt keine Korrelation zwischen IQ-Werten und dem motorischen Skill nach dem Training.
3. Größere Werte auf der Skala für wahrnehmungsgebundenes logisches Denken bewirken eine signifikante Steigerung des motorischen Skills nach dem Training.
4. Die Werte der Vorhersage steigen mit denen des wahrnehmungsgebundenen logischen Denkens und die Werte für wahrnehmungsgebundenes logisches Denken leisten einen signifikanten Beitrag zur Vorhersage des motorischen Skills nach der Intervention.

Antwort

Viertens kann man so sagen.

Gegen 3 spricht die Kausalinterpretation, 1 ist verkehrt (keine Signifikanz) und über 2 können wir mit dem Ergebnis des Gesamtmodells direkt keine Aussage treffen.

6.2.6 Test der Signifikanz von R^2

Man kann sich die Frage stellen, ob das Modell mit den gewählten Prädiktoren insgesamt das Kriterium gut vorhersagt. Das lässt sich am einfachsten bewerkstelligen, indem man den Determinationskoeffizienten R^2 auf die H_0 testet, dass der 'wahre' Koeffizient R^* der Population gleich 0 ist. ("$H_0 : R^* = 0$")

Getestet wird diese Hypothese mit der Teststatistik $F = \frac{(N-p-1)R^2}{p(1-R^2)}$, die $F_{p,N-p-1}$-verteilt ist. Dabei ist N wieder die Anzahl der Beobachtungen und p die Anzahl der Prädiktoren.

Um diesen Test durchzuführen können wir entweder auf den unteren Teil des `summary`-Outputs für ein Regressionsmodell gucken:

```
## Residual standard error: 0.9692 on 47 degrees of freedom
## Multiple R squared: 0.0989 , Adjusted R-squared: 0.0606
## F-statistic:  2.58 on 2 and 47 DF, p-value: 0.0865
```

Oder die `broom::glance()`-Funktion nutzen:

```
fit_post_il_z %>%
  broom::glance()
```

6.2.6.1 Aufgabe

Wie lässt sich das Ergebnis interpretieren?

1. Unser Modell ist ein sehr gutes Modell, es klärt einen signifikanten Teil der Varianz auf.
2. Unser Modell sagt den motorischen Skill nicht gut voraus.
3. Unser Modell klärt $\sim 10\%$ der Varianz auf.

Antwort

2 und 3 lassen sich so sagen.

6.2.7 Prüfung der Voraussetzungen

Für die Tests auf Signifikanz in der konventionellen Regressionsanalyse gelten die Voraussetzungen der Varianzhomogenität und der Normalverteiltheit der

Residuen für jede einzelne Prädiktor-Kombination. Wir setzen also wieder voraus, dass Y für jede Kombination der X_p normalverteilt ist mit $\mu = b_1^* X_1 + \cdots + b_p^* X_p + a^*$ und einer Varianz σ^2.

Zusätzlich zu den Voraussetzungen ist für die multiple Regression das Ausmaß der Multikollinearität der Prädiktoren relevant.

6.2.8 Grafische Prüfverfahren

Um die Anforderungen an die Messfehler der Regression heuristisch zu überprüfen, lassen sich verschiedene grafische Darstellungen heranziehen. Dazu benutzt man am besten eine standardisierte Form der Residuen. Zwei davon haben sich durchgesetzt, zum einen die

studentischen

$$E_{stud} = \frac{\frac{E}{s}}{(1 - \frac{1}{N} + h)^{\frac{1}{2}}}$$

und zum anderen die *standardisierten*

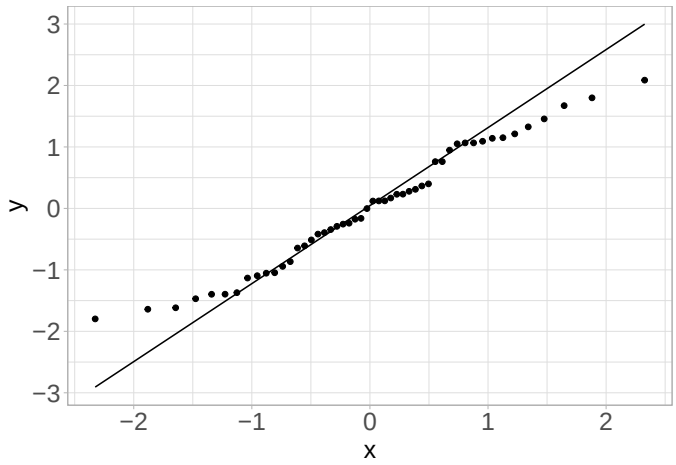
$$E_{stan} = \frac{Y - \hat{Y}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}}$$

Residuen. Die Residuen lassen sich dafür mit `rstudent()` oder `rstandard()` berechnen.

Hier wird die `broom::augment()`-Funktion praktisch, da wir in dem ausgegebenen Datensatz die Residuen und die vorhergesagten Werte praktisch aufbereitet haben

Die Verteilungseigenschaften können wir dann grafisch-heuristisch mit einem qq-Plot überprüfen:

```
fit_post_il %>%
  broom::augment() %>%
  ggplot(aes(sample = .std.resid)) +
  geom_qq() +
  geom_qq_line()
```

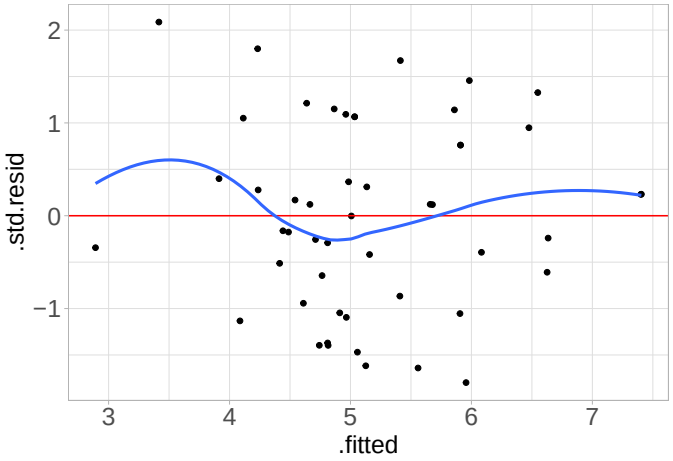


Je weiter die Punkte von der eingezeichneten Gerade abweichen, desto weniger können wir von einer Normalverteiltheit der Residuen ausgehen. Wie groß "noch akzeptable" Abweichung ist, ist ein Stück weit Gefühlssache.

Hier findet man eine kleine shiny-App, mit der an qq-plots rumgespielt werden kann:

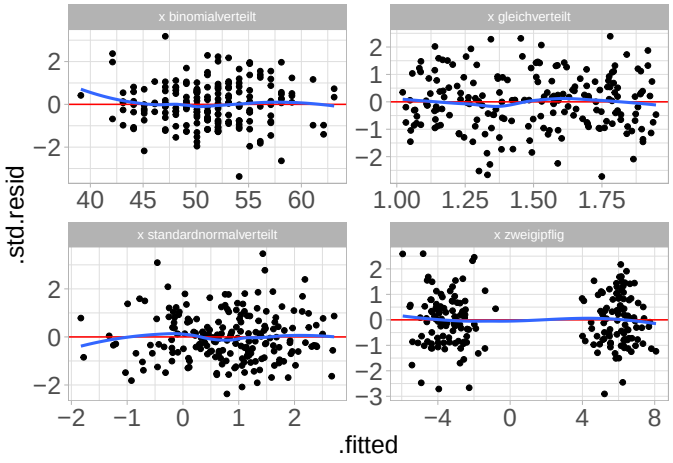
Die Voraussetzung der Varianzhomogenität lässt sich auch so verstehen, dass wir für Y für jede Werte-Kombination unserer Prädiktoren eine gleich große Varianz voraussetzen. Über den Verlauf unserer vorhergesagten Werte sollten wir also um den 0-Punkt ungefähr gleich (breit) streuende Residuen beobachten können. Die genaue Form ist dabei aber natürlich abhängig von der Verteilung der Prädiktoren. Wenn wir diesem Plot jetzt noch einen lokalen Schätzer des Mittelwerts hinzufügen, haben wir einen so genannten Spread-Level-Plot:

```
fit_post_il %>%
  broom::augment() %>%
  ggplot(aes(x = .fitted, y = .std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = 'red') +
  geom_smooth(formula = 'y~x',se = F, method = 'loess')
```

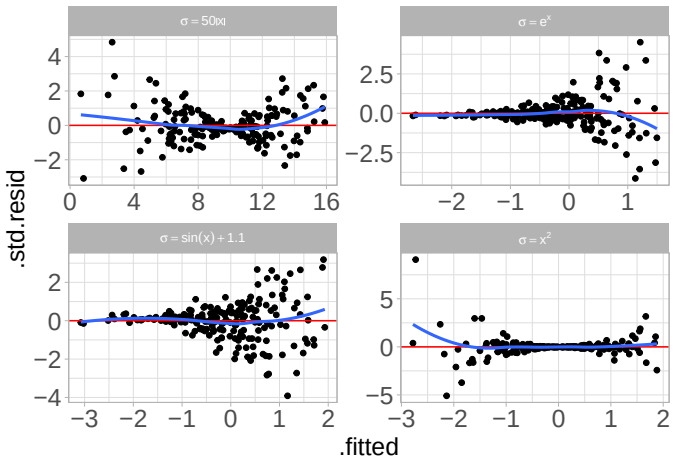


In unserem Fall haben wir scheinbar ein paar Ausreißer (wie vorher auch schon gesehen), sonst ist die Punktwolke aber unproblematisch. Die Abweichung der loess-Regression sieht oft dramatischer aus als es faktisch ist, vor allem bei wenigen Beobachtungen wie bei uns.

Man muss immer im Hinterkopf behalten, dass das genaue Bild stark von der Verteilung der Prädiktoren abhängt. So ist keins der folgenden Muster wirklich eine typische Punktwolke, trotzdem sind die Voraussetzungen überall gegeben:



Wirklich Grund zur Sorge sollten uns Bilder wie die folgenden geben:



6.2.9 Inferenzstatistischer Test der Voraussetzungen

Inferenz-statistisch lässt sich die Normalverteilung der Residuen zum Beispiel mit dem Kolmogorov-Smirnov-Test überprüfen:

```
ks.test(x = rstudent(fit_post_il),
        y = 'pnorm')
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  rstudent(fit_post_il)
## D = 0.093678, p-value = 0.7726
## alternative hypothesis: two-sided
```

6.2.9.1 Aufgabe

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  rstudent(fit_post_il)
## D = 0.093678, p-value = 0.7726
## alternative hypothesis: two-sided
```

Wie lässt sich das Ergebnis interpretieren? 1. $1 - p$ ist kleiner als 30%, deswegen können wir keine Normalverteilung annehmen. 2. p ist größer als 20%; da wir eine Normalverteiltheit nicht ausschließen können, nehmen wir diese Voraussetzung als gegeben an. 3. Wir können gar nichts sagen, der Test aller Residuen auf einmal ergibt keinen Sinn.

Antwort

Zweitens ist die übliche Interpretation, auch wenn dem Herrn Andres hier der Dampf aus den Ohren steigt.

6.2.10 Multikollinearität

Multikollinearität liegt dann vor, wenn sich die Werte eines Prädiktors gut aus einer Linearkombination der übrigen Prädiktoren vorhersagen lassen.

Dies ist insbesondere dann der Fall, wenn Prädiktoren paarweise miteinander hoch korrelieren.

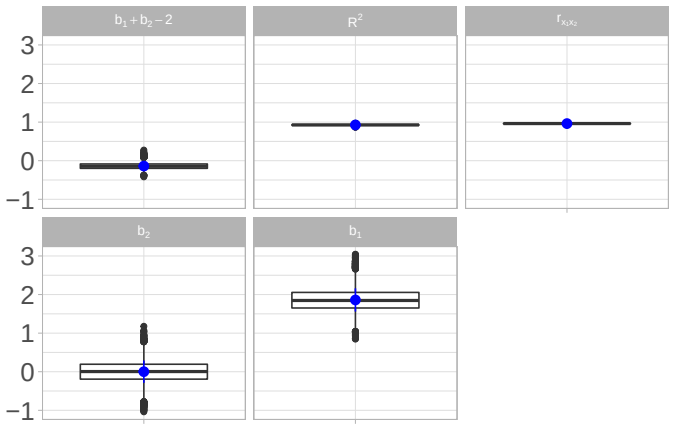
Für die multiple Regression hat dies weniger stabile Schätzungen der Koeffizienten als unerwünschte Konsequenz. Das heißt für die Praxis, dass die Regressionsgewichte schwer interpretierbar werden, sobald die entsprechenden Prädiktoren zu stark korrelieren, da von Stichprobe zu Stichprobe starke Änderungen zu erwarten sind. Modelle mit Multikolliniaren Prädiktoren haben aber meistens relativ stabile Determinationskoeffizienten, wenn uns also so oder so nur das Gesamtmodell interessiert, ist Multikollinearität kein allzu großes Problem.

Als kleines Beispiel sind hier Simulationsergebnisse von 10000 Regressionen mit jeweils korrelierten Prädiktoren:

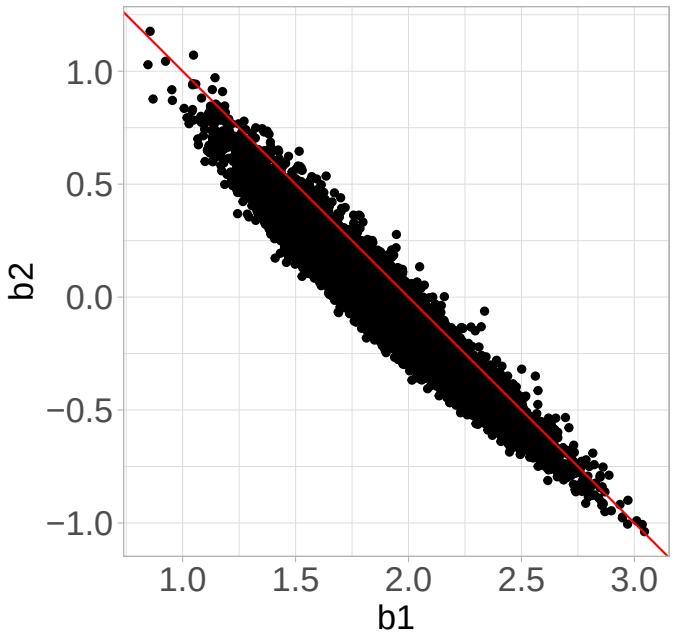
Die Regression wurden jeweils auf einem nach dem folgenden Schema simulierten Datensatz erstellt:

```
tibble(y = rnorm(100),
       x1 = y / 2 * runif(100,.5, 1.5),
       x2 = x1 * runif(100,.5, 1.5))
```

Insgesamt verteilen sich die Ergebnisse wie folgt:



Gewisse Linearkombinationen der Koeffizienten sind deutlich stabiler als die Koeffizienten allein, wie man hier am Beispiel von b_1+b_2+2 gut sehen kann. Welche Linearkombination gerade besonders stabil sein könnte, kann man einfach an einer Punktwolke wie der folgenden ablesen:



Paarweise lineare Zusammenhänge lassen sich anhand der Korrelationsmatrix der Prädiktoren prüfen.

```
df_wide %>%
  select(hawie_iq,
         hawie_wahr_log) %>%
  cor()
```

```
##                 hawie_iq hawie_wahr_log
## hawie_iq       1.0000000      0.7590665
## hawie_wahr_log 0.7590665      1.0000000
```

Faustregel: Korrelationen > 0.8 weisen auf starke Kollinearität hin.

Der `Varianzinflationsfaktor` $\text{VIF}_j = \frac{1}{1-R_j^2}$ jedes Prädiktors j liefert eine weitere Möglichkeit zur Kollinearitätsdiagnostik. Er kann mit der Funktion `vif()` aus dem Paket `car` berechnet werden.

```
library(car)
vif(fit_post_il)
```

```
##       hawie_iq hawie_wahr_log
##       2.359503       2.359503
```

Faustregel: VIF-Faktor $> 4 \rightarrow$ starke Multikollinearität

Table 6.2

post_skill	hawie_iq	hawie_wahr_log	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
2	93	9	4.91	-2.91	0.022	2.81	0.0082	-1.05
9	121	16	6.48	2.52	0.104	2.82	0.0347	0.948
8	90	8	4.64	3.36	0.0282	2.8	0.0142	1.21
1	97	11	5.56	-4.56	0.0241	2.76	0.0221	-1.64
8	94	15	7.41	0.595	0.168	2.84	0.00361	0.232
4	95	9	4.81	-0.811	0.023	2.84	0.000669	-0.292
5	96	7	3.91	1.09	0.0632	2.84	0.00358	0.399
3	107	13	5.91	-2.91	0.0398	2.81	0.0154	-1.05
10	97	12	5.98	4.02	0.0381	2.78	0.028	1.46
8	92	9	4.96	3.04	0.0222	2.81	0.00904	1.09
6	120	14	5.68	0.322	0.0838	2.84	0.000436	0.12
4	97	9	4.71	-0.711	0.0262	2.84	0.000589	-0.256
6	97	10	5.13	0.865	0.0201	2.84	0.000661	0.311
5	98	8	4.24	0.763	0.0465	2.84	0.00125	0.278
8	111	17	7.4	0.599	0.144	2.84	0.00298	0.23
2	99	9	4.61	-2.61	0.0315	2.82	0.00965	-0.943
8	99	10	5.03	2.97	0.0219	2.81	0.0085	1.07
8	90	11	5.91	2.09	0.0467	2.83	0.00945	0.761
1	95	9	4.81	-3.81	0.023	2.79	0.0148	-1.37
1	101	8	4.09	-3.09	0.0621	2.8	0.0283	-1.13
2	91	4	2.89	-0.892	0.152	2.84	0.00706	-0.344
5	108	11	5.01	-0.00743	0.041	2.84	1.03e-07	-0.0027
4	77	6	4.44	-0.442	0.0631	2.84	0.00059	-0.162
4	105	11	5.16	-1.16	0.03	2.84	0.0018	-0.418
8	77	7	4.87	3.13	0.0619	2.8	0.0291	1.15
5	95	12	6.08	-1.08	0.0454	2.84	0.00246	-0.394
7	92	7	4.11	2.89	0.047	2.81	0.0182	1.05
8	90	11	5.91	2.09	0.0467	2.83	0.00945	0.761
8	99	10	5.03	2.97	0.0219	2.81	0.0085	1.07

Table 6.3

term	estimate	std.error	statistic	p.value
(Intercept)	6.81e-17	0.137	4.97e-16	1
hawie_iq	-0.233	0.213	-1.1	0.279
hawie_wahr_log	0.452	0.213	2.13	0.0387

Table 6.4

term	estimate	std.error	statistic	p.value
(Intercept)	6.81e-17	0.137	4.97e-16	1
hawie_iq	-0.233	0.213	-1.1	0.279
hawie_wahr_log	0.452	0.213	2.13	0.0387

Table 6.5

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.0989	0.0606	0.969	2.58	0.0865	2	-67.8	144	151	44.2	47

Table 6.6

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.0989	0.0606	0.969	2.58	0.0865	2	-67.8	144	151	44.2	47

Table 6.7

y	x1	x2
0.193	0.143	0.0999
1.02	0.735	0.816
0.813	0.21	0.186
0.954	0.361	0.218
2.08	0.931	1.01
1.61	0.71	1.02
0.19	0.142	0.152
-0.999	-0.336	-0.251
-0.615	-0.203	-0.142
3.32	1.41	1.54
0.922	0.545	0.431
1.23	0.467	0.32
-0.216	-0.142	-0.115
-1.34	-0.401	-0.443
-0.304	-0.153	-0.163
-0.0779	-0.0267	-0.0381
-0.193	-0.0836	-0.0737
0.123	0.0864	0.0447
-0.279	-0.16	-0.122
0.758	0.306	0.297
0.0606	0.0302	0.0388
-0.165	-0.0743	-0.0917
0.0715	0.0189	0.0136
0.649	0.22	0.261
1.1	0.359	0.369
-1.37	-0.553	-0.538
-0.127	-0.0864	-0.0849
1.08	0.804	1.09
-1.45	-0.736	-0.967

Chapter 7

Gruppenunterschiede I

7.0.1 Datensatz

Wir verwenden den (leicht erweiterten), simulierten Datensatz aus den letzten Wochen:

Variable	Inhalt
‘vp_nr‘	VP-Nummer
‘group‘	Treatment-Gruppe
‘sex‘	Geschlecht
‘hawie_iq‘	Intelligenz-Quotient aus HAWIE
‘hawie_wahr_log‘	Skalenwert wahrnehmungsgebundenes logisches Denken aus HAWIE
‘pre_skill‘	motorischer Skill vor dem Treatment
‘post_skill‘	motorischer Skill nach dem Treatment

7.1 t-Test

7.1.1 Ein-Stichproben-Problem

Beim Ein-Stichproben-t-Test wird der Erwartungswert einer Stichprobe gegen den bekannten Erwartungswert einer Population getestet.($H_0 : \mu = \mu_0$)

Der Test setzt eigentlich voraus, dass die abhängige Variable normalverteilt ist. Wenn die Stichprobe groß genug ist, um eine näherungsweise Normalverteilung des Mittelwertes zu gewährleisten, kann man die t-Statistik aber ohne viele Probleme mit der entsprechenden t-Verteilung (t_{n-1}) verglichen werden.

```
t.test(x=<Vektor>, alternative=c("two.sided", "less", "greater"), mu=0)
```

```
muH0 <- 100
(tResults <- t.test(df_wide$hawie_iq, alternative="two.sided", mu=muH0))
```

```
##
##  One Sample t-test
##
## data:  df_wide$hawie_iq
## t = -4.132, df = 149, p-value = 5.976e-05
## alternative hypothesis: true mean is not equal to 100
## 95 percent confidence interval:
##  91.76135 97.09199
## sample estimates:
## mean of x
##  94.42667
```

7.1.2 Zwei-Stichproben-Problem, abhängiger Fall

Beim abhängigen Zwei-Stichproben-Fall wird im Prinzip derselbe Test durchgeführt, nur dass hier die Differenzwerte der Beobachtungen als Stichprobe genutzt werden. Da die Frage hier nicht ist, ob der Erwartungswert einer Population einem spezifischen Erwartungswert gleich ist, sondern ob sich die Erwartungswerte der jeweiligen Beobachtungen unterscheiden, ist die Nullhypothese hier:

$$\mathrm{H}_0 : \mu_D = \mu_1 - \mu_2 = 0$$

Die Teststatistik ist hier auch t_{n-1}-verteilt. Dabei steht das n hier aber für die Anzahl der Beobachtungs-Paare, nicht die Gesamtzahl der Beobachtungen.

```
t.test(x=df_wide$pre_skill,
       y=df_wide$post_skill,
       alternative="two.sided", paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  df_wide$pre_skill and df_wide$post_skill
## t = 0.9628, df = 149, p-value = 0.3372
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.2946581  0.8546581
## sample estimates:
## mean difference
##            0.28
```

Geht auch für das long-Format und in Formel-Schreibweise:

```
df_wide %>%
  pivot_longer(cols = c(pre_skill,post_skill),
               names_to = 'time',
               values_to = 'skill') %>%
  t.test(skill ~ time, data=.,
         alternative = 'less', paired=T)
```

```
##
##  Paired t-test
##
## data:  skill by time
## t = -0.9628, df = 149, p-value = 0.1686
## alternative hypothesis: true mean difference is less than 0
## 95 percent confidence interval:
##       -Inf 0.2013441
## sample estimates:
## mean difference
##           -0.28
```

Und genau wie bei den Regressionsanalysen können wir auch hier wieder `broom` benutzen, um eine übersichtliche Tabelle zu erhalten:

```
df_wide %$%
  t.test(x=pre_skill,
         y=post_skill,
         alternative="two.sided",
         paired=TRUE) %>%
  broom::tidy()
```

Table 7.1

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
0.28	0.963	0.337	149	-0.295	0.855	Paired t-test	two.sided

7.1.2.1 Aufgabe

```
## [1] "t = -0.9628, df = 149, p-value = 0.1686"
```

Wie interpretieren wir das Ergebnis?

1. Die Skillwerte zwischen Pre- und Post-Zeitpunkt sind nicht unterschiedlich.
2. Wir verwerfen die H_0 und nehmen an, dass die Skillwerte der Population sich zu den zwei Zeitpunkten unterschieden.
3. Wir können die H_0 nicht verwerfen.

4. Weil wir unabhängige Testzeitpunkte haben, ist das der falsche Test, das Ergebnis ist also tendenziell zu liberal.

Antwort

1 ist eine Interpretation der H_0, 2 wäre richtig wäre der Test signifikant, 4 ist ein möchtegern-schlauer Distraktor.

Richtig ist 3.

7.1.3 Zwei-Stichproben-Problem, unabhängiger Fall

Beim unabhängigen Stichproben t-Test wird getestet, ob die Erwartungswerte zweier Stichproben unterschiedlich sind ($H_0 : \mu_1 = \mu_2$). Dabei wird vorausgesetzt, dass die abhängige Variable in beiden Gruppen normalverteilt ist und dieselbe Varianz hat. Gelten diese Voraussetzungen, ist die t-Statistik unter der H_0 $t_{n_1+n_2-2}$-verteilt.

```
t.test(x=<Vektor>, y=<Vektor>, paired=FALSE,
       alternative=c("two.sided", "less", "greater"))
```

Auch als Formelschreibweise:

```
df_wide %>%
  t.test(post_skill ~ sex, data=.,
       alternative="less", paired=FALSE,var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  post_skill by sex
## t = -1.7207, df = 148, p-value = 0.0437
## alternative hypothesis: true difference in means between group f and group m is less
## 95 percent confidence interval:
##         -Inf -0.02840063
## sample estimates:
## mean in group f mean in group m
##        4.626667        5.373333
```

7.1.3.0.1 Ungleiche Varianzen (oder Stichprobengrößen) Im Falle einer Varianzinhomogenität ist die t-Statistik nicht mehr gut mit der $t_{n_1+n_2-2}$-Verteilung zu vergleichen. Die Welch-Korrektur kann in diesem Fall dazu dienen, eine Anzahl von Freiheitsgraden zu schätzen, die besser zu der Verteilung unter der H_0 der Teststatistik passt.

Mit dem `var.equal`-Argument lässt sich logisch angeben, ob die Freiheitsgrade nach Welch geschätzt werden sollen. Dabei ist `FALSE` der Standard.

```
df_wide %>%
  t.test(post_skill ~ sex, data=.,
       alternative="less", paired=FALSE,var.equal=F)
```

```
##
##  Welch Two Sample t-test
##
## data:  post_skill by sex
## t = -1.7207, df = 145.74, p-value = 0.04372
## alternative hypothesis: true difference in means between group f and group m is less than 0
## 95 percent confidence interval:
##         -Inf -0.02833062
## sample estimates:
## mean in group f mean in group m
##        4.626667        5.373333
```

7.1.4 Test von Voraussetzungen

Obwohl t-Tests relativ robust gegen Verletzungen der Normalverteilungsvoraussetzung sind, kann man auf die Idee kommen, auf die Normalverteiltheit der AV zu testen Außerdem kann für den Zwei-Stichproben-Test überlegt werden, ob die Varianzhomogenität überprüft werden soll, um über das genutzte Verfahren zu entscheiden.

7.1.5 Test auf Normalverteilung

Der Kolmogorov-Smirnov-Test auf eine feste Verteilung vergleicht die kumulierten relativen Häufigkeiten von Daten einer stetigen Variable mit einer frei wählbaren Verteilungsfunktion – etwa der einer bestimmten Normalverteilung. Hier wird er aber nur der Vollständigkeit halber erwähnt, da er stark abhängig von der Stichprobengröße schnell zu suboptimalen Handlungen führen kann.

```
ks.test(x=<Vektor> , y="<Name der Verteilungsfunktion>",
        alternative=c("two.sided", "less", "greater"))
```

```
ks.test(df_wide$hawie_iq, 'pnorm', 100, 15, alternative='two.sided')
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  df_wide$hawie_iq
## D = 0.19056, p-value = 3.715e-05
## alternative hypothesis: two-sided
```

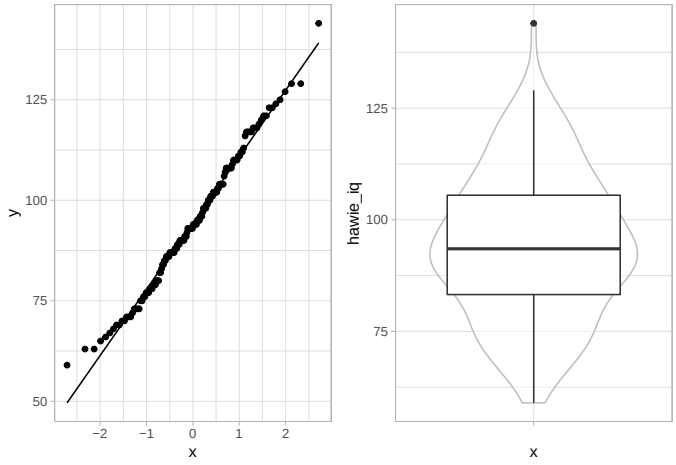
Bei kleinem N kann der Shapiro-Wilk-Test auf unspezifische Normalverteilung bessere Ergebnisse liefern, bei großem N hat dieser aber dasselbe Problem wie

der KS-Test, und wird eigentlich immer signifikant.

```
shapiro.test(df_wide$hawie_iq)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_wide$hawie_iq
## W = 0.99, p-value = 0.3654
```

Außerdem möglich sind grafisch-heuristische Testvarianten:



7.1.5.1 Aufgabe

Was ist daran falsch?

1. Nichts, alle diese Verfahren können so angewendet werden
2. Verfahren zum Testen von Verteilungsvoraussetzungen sind absurd! die Verfahren, die wir meistens benutzen, sind gegen alle Verletzungen der Voraussetzungen robust!
3. Über alle Gruppen hinweg einen Test auf Normalverteilung durchzuführen ist nicht sinnvoll.
4. Grafiken sind deutlich weniger aussagekräftig als inferenzstatistische Tests, die haben hier nichts zu suchen.

Antwort

1. ist falsch, die Av über die Zellen zu poolen, ergibt keinen Sinn.
2. könnte man argumentieren, bei sehr extremen Verletzungen oder der Verletzung mehrerer Voraussetzungen gleichzeitig kann man aber in Schwierigkeiten kommen
3. ist richtig, siehe 1.

4. sehe ich nicht so. Grafisch-heuristische Tests haben ihren heuristischen Entscheidungsanteil wenigstens im Namen.

7.1.6 Test auf Varianzhomogenität

Varianzhomogenität lässt sich mit dem Levene-Test testen, einem Signifikanz-Test, der auf Abweichungen von der Varianzhomogenität in zwei oder mehr Populationen testet. Im Prinzip wird die absolute Abweichung jedes Wertes einer Gruppe vom Zentrum dieser Gruppe berechnet. Über die resultierenden Werte wird dann eine Varianzanalyse gerechnet.

Dafür kann in `R` die Funktion `leveneTest()` aus dem `car`-Paket verwendet werden. Dabei testet der Standard nicht die von Levene ursprünglich aufgestellte Differenz zum Mittelwert, sondern die Version von Brown und Forsythe, bei der die Differenz zum Median gebildet wird.

7.1.6.1 Levene-Test für einen Faktor

```
library(car)
df_wide %>%
  leveneTest(hawie_iq ~ sex, data=.)
```

Table 7.2

Df	F value	Pr(>F)
1	0.341	0.56
148		

7.1.6.2 Levene-Test für mehr als einen Faktor

Mit Verknüpfung zweier Faktoren als Interaktion im Modellterm wird der Test als assoziierte einfaktorielle Varianzanalyse ausgeführt.

```
df_wide %>%
  leveneTest(hawie_iq ~ group * sex, data=.)
```

Table 7.3

Df	F value	Pr(>F)
5	0.489	0.784
144		

7.2 Varianzanalyse - ein unabhängiger Faktor

7.2.1 Ein unabhängiger Faktor

Bei der einfaktoriellen, unabhängigen Varianzanalyse testen wir eine Variable in einer Reihe von Bedingungen darauf, ob die jeweiligen Erwartungswerte unterschiedlich sind. Bei Gültigkeit der $H_0 : \mu_1 = \mu_2 ... = \mu_J$, sowie Unabhängigkeit, Normalverteiltheit und gleicher Varianz der AV in allen Bedingungen ist die Teststatistik $F_{J-1,N-J}$-verteilt.

Für Varianzanalysen gibt es eine ganze Reihe von Paketen und Funktionen, `base-R` hat alleine drei, die man für diesen Zweck nutzen kann. Für die bessere Integration in den `tidyverse`-workflow werden wir hier aber die Funktionen aus dem `ez`-Paket nutzen.

```
ezANOVA(data = <data>,
        dv = <AV>,
        wid = <VP-Code>,
        between = <UV>)
```

Als Beispiel sollen die Werte der zentralen Leistungs-AV zum ersten Messzeitpunkt in der follow-up Phase zwischen den Gruppen verglichen werden.

```
library(ez)
anova_group <-
  df_wide %>%
  ezANOVA(dv = post_skill,
          wid = vp_nr,
          between = group)
```

Ergebnis des Tests:

```
anova_group
```

```
## $ANOVA
##   Effect DFn DFd        F            p p<.05
## 1  group   2 147 17.76712 1.227478e-07     *
##         ges
## 1 0.1946717
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd      SSn   SSd         F         p p<.05
## 1   2 147 2.773333 326.8 0.6237454 0.5373457
```

Der Output besteht aus den Ergebnistabellen für zwei inferenzstatistische Tests. Zum einen ist da das Ergebnis eines Levene-Tests:

```
anova_group$`Levene's Test for Homogeneity of Variance`
```

Mit

Table 7.4

DFn	DFd	SSn	SSd	F	p	p<.05
2	147	2.77	327	0.624	0.537	

- Zähler- und Nennerfreiheitsgraden
- den entsprechenden Quadratsummen
- dem F-Wert
- dem entsprechenden p-Wert und einer Signifikanzaussage

Zum Anderen dem Ergebnis der eigentlichen Varianzanalyse:

```
anova_group$`ANOVA`
```

Table 7.5

Effect	DFn	DFd	F	p	p<.05	ges
group	2	147	17.8	1.23e-07	*	0.195

Mit

- einer Angabe des Effekts, für den die Ergebnisse berichtet werden
- Zähler- und Nennerfreiheitsgraden
- den entsprechenden Quadratsummen
- dem F-Wert
- dem entsprechenden p-Wert und einer Signifikanzaussage
- einem Schätzer für die Effektstärke η^2

7.2.1.1 Aufgabe

Table 7.6

Effect	DFn	DFd	F	p	p<.05	ges
group	2	147	17.8	1.23e-07	*	0.195

Wie interpretieren wir das Ergebnis?

1. Es gibt keinen Unterschied zwischen den Gruppen.
2. Der Test ist signifikant, deswegen nehmen wir an, dass die Interventionen zu größerem Skill führen.

3. Der Test ist nicht signifikant, deswegen können wir keine Aussage treffen. Mit dem mittleren Effekt können wir aber mit G*Power eine Stichprobengröße ausrechnen, die zu Signifikanz führen würde.
4. Der Test ist signifikant, deswegen entscheiden wir uns die Alternativhypothese anzunehmen, dass die Interventionen und die Kontrollgruppe sich in ihren Skill-Werten unterscheiden.

Antwort

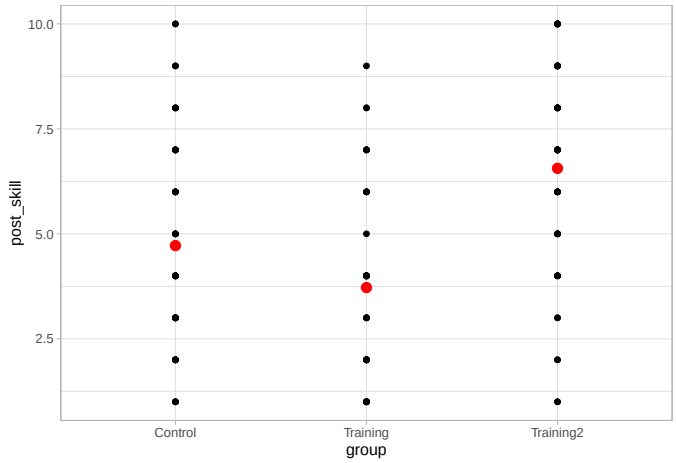
1. ist falsch, der Test ist signifikant
2. ist falsch, da eine Richtung mit-formuliert ist und wir ungerichtet getestet haben
3. ist falsch da zum Einen der Test signifikant ist und zum Anderen so Power nicht funktioniert
4. ist richtig

7.2.2 Grafisch Voraussetzungen prüfen

Die Voraussetzungen der Varianzanalyse lassen sich zu folgender Aussage umformulieren:

Die Fehler müssen unabhängige, normalverteilte Variablen mit Erwartungswert 0 und gleicher Streuung sein.

Diese umformulierten Anforderungen lassen sich dann wieder wie bei der Regressionsanalyse grafisch-heuristisch untersuchen.

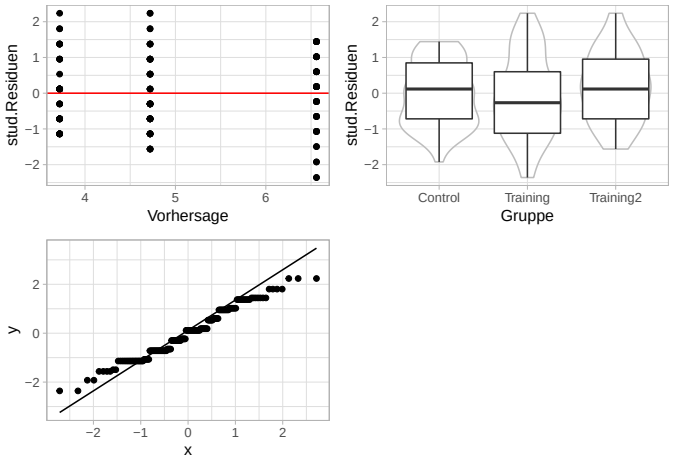


Damit wir die bekannten Funktionen zur Standardisierung der Fehler nutzen können, müssen wir aber noch das `return_aov`-Argument auf `TRUE` setzen.

`aov`-Modelle sind die `base-R` Version, Ergebnisse von Varianzanalysen in einem ähnlichen Format wie die von linearen Modellen abzuspeichern.

```
anova_group <- df_wide %>%
  ezANOVA(dv = post_skill,
          wid = vp_nr,
          between = group,
          return_aov = T)
anova_group$aov
```

```
## Call:
##    aov(formula = formula(aov_formula), data = data)
##
## Terms:
##                   group Residuals
## Sum of Squares  207.52    858.48
## Deg. of Freedom      2       147
##
## Residual standard error: 2.416609
## Estimated effects may be unbalanced
```



7.2.3 Paarvergleiche mit t-Tests und α-Adjustierung

Besteht nach einer ANOVA die Frage, bei welchen Paaren Erwartungswertunterschiede vorliegen, können t-Tests durchgeführt werden, um die Unterschiede jeweils zweier Gruppen auf Signifikanz zu prüfen.

```
with(df_wide, pairwise.t.test(post_skill, group,
                              p.adjust.method="bonferroni", paired=F, pool.sd=T))
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  post_skill and group
```

```
##
##           Control Training
## Training  0.12089 -
## Training2 0.00062 8.1e-08
##
## P value adjustment method: bonferroni
```

Das `p.adjust.method`-Argument übernimmt eine relativ große Zahl von Korrekturen:

```
"holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
"none"
```

Chapter 8

Gruppenunterschiede II

8.0.1 Datensatz

Wir verwenden den (leicht erweiterten), simulierten Datensatz aus den letzten Wochen:

Variable	Inhalt
‘vp_nr‘	VP-Nummer
‘group‘	Treatment-Gruppe
‘sex‘	Geschlecht
‘hawie_iq‘	Intelligenz-Quotient aus HAWIE
‘hawie_wahr_log‘	Skalenwert wahrnehmungsgebundenes \ logisches Denken aus HAWIE
‘pre_skill‘	motorischer Skill vor dem Treatment
‘peri_skill‘	motorischer Skill während des Treatments
‘post_skill‘	motorischer Skill nach dem Treatment
‘substance‘	Substanzgabe vor der jeweiligen Testung

8.1 Varianzanalyse - ein abhängiger Faktor

8.1.1 ein abhängiger Faktor

Die einfaktorielle Varianzanalyse mit abhängigen Gruppen dient dazu, Mittelwerte zu vergleichen, die aus einer Reihe von Beobachtungen stammen, bei denen statistische Abhängigkeit angenommen werden muss. Das ist zum Beispiel bei einem Design mit wiederholten Messungen in jeweils ein und derselben Versuchsperson der Fall.

Wenn die Variablen normalverteilt, die wiederholten Messungen untereinander (zwischen den Versuchspersonen) unabhängig sind und Sphärizität gegeben

ist, ist die Teststatistik im Falle der Gültigkeit der $H_0 : \mu_1 = \mu_2 ... = \mu_k$ $F_{k-1,(k-1)(N-1)}$-verteilt.

Für die Durchführung der Varianzanalyse mit abhängigen Gruppen mit `ezANOVA()` muss

- der Datensatz im LONG-Format vorliegen (zumindest muss die AV über alle Messzeitpunkte in einer Variable abgelegt sein)
- Ein Blockbildungsfaktor, also eine Kennzeichnung des Messzeitpunktes vorliegen
- wie vorher auch eine Variable, die die Versuchsperson codiert, vorliegen.

Im Folgenden wird der Blockbildungsfaktor genannt.

Der aus dem unabhängigen Fall bekannte Funktionsaufruf muss nur so geändert werden, dass statt des vorher mit dem `between`-Argument angegebenen unabhängigen Faktors mit dem `within`-Argument der abhängige oder Blockbildungsfaktor angegeben wird.

```
ezANOVA(data = <data>,
        dv = <AV>,
        wid = <VP-Code>,
        within = <Block>)
```

In unserem Beispiel-Datensatz sieht das so aus:

```
library(ez)
anova_time <-
  df_wide %>%
  pivot_longer(cols = contains('skill'),
               names_to = 'time',
               values_to = 'skill') %>%
  ezANOVA(dv = skill,
          wid = vp_nr,
          within = time)
```

Das Ganze führt dann zu dem folgenden Ergebnis:

```
anova_time
```

```
## $ANOVA
##   Effect DFn DFd        F          p p<.05        ges
## 2   time   2 398 4.329244 0.01380361     * 0.01245873
##
## $`Mauchly's Test for Sphericity`
##   Effect         W            p p<.05
## 2   time 0.8678891 8.090194e-07     *
##
## $`Sphericity Corrections`
##   Effect       GGe      p[GG] p[GG]<.05       HFe
```

```
## 2   time 0.8833057 0.01757486         * 0.8906258
##       p[HF] p[HF]<.05
## 2 0.0173103         *
```

Der Output besteht aus den Ergebnistabellen für zwei inferenzstatistische Tests und einer Ergebnistabelle für die Testergebnisse nach Sphärizitäts-Korrektur. Als Erstes ist da das Ergebnis eines Mauchly-Tests (ein Test auf Sphärizität):

```
anova_time$`Mauchly's Test for Sphericity`
```

Table 8.1

Effect	W	p	p<.05
time	0.868	8.09e-07	*

Mit

- Angaben zu dem/n Faktor(en) für den/die Sphärizität überprüft wurde
- der entsprechenden Teststatistik
- dem entsprechenden p-Wert und einer Signifikanzaussage

Zum Anderen dem Ergebnis der eigentlichen Varianzanalyse:

```
anova_time$`ANOVA`
```

Table 8.2

Effect	DFn	DFd	F	p	p<.05	ges
time	2	398	4.33	0.0138	*	0.0125

Mit

- einer Angabe des Effekts, für den die Ergebnisse berichtet werden
- Zähler- und Nennerfreiheitsgraden
- den entsprechenden Quadratsummen
- dem F-Wert
- dem entsprechenden p-Wert und einer Signifikanzaussage
- einem Schätzer für die Effektstärke η^2

Und zu guter Letzt mit dem Ergebnis der Varianzanalyse nach Greenhouse-Geisser und Hyunh-Feldt-Korrektur

```
anova_time$`Sphericity Corrections`
```

Mit

- Einer Angabe des Effekts, für den die Ergebnisse berichtet werden

Table 8.3

Effect	GGe	p[GG]	p[GG]<.05	HFe	p[HF]	p[HF]<.05
time	0.883	0.0176	*	0.891	0.0173	*

- dem ϵ nach Greenhouse-Geisser
- dem entsprechenden p-Wert und einer Signifikanzaussage
- dem ϵ nach Hyunh-Feldt
- dem entsprechenden p-Wert und einer Signifikanzaussage

8.1.1.1 Aufgabe

```
## $ANOVA
##   Effect DFn DFd        F          p p<.05        ges
## 2   time   2 398 4.329244 0.01380361     * 0.01245873
##
## $`Mauchly's Test for Sphericity`
##   Effect         W            p p<.05
## 2   time 0.8678891 8.090194e-07     *
##
## $`Sphericity Corrections`
##   Effect       GGe      p[GG] p[GG]<.05       HFe
## 2   time 0.8833057 0.01757486         * 0.8906258
##       p[HF] p[HF]<.05
## 2 0.0173103         *
```

Wie interpretieren wir das Ergebnis?

1. Es gibt keinen Unterschied.
2. Da Sphärizität nicht gegeben ist, ist das Ergebnis nicht zu interpretieren.
3. Mit Berücksichtigung des mit Sphärizitäts-Korrektur nach Huynh-Feldt signifikant gewordenen Tests nehmen wir unsere H_1 an, also dass sich die Skillwerte zu den Testzeitpunkten unterscheiden, es also einen Effekt der Treatments gab.
4. Da wir die Normalverteiltheit der Variablen nicht überprüft haben, können wir keine Aussage machen.

Antwort

1. ist falsch, da Interpretation der H_0
2. ist falsch, zwar ist die Voraussetzung der Sphärizität nicht gegeben, da wir aber dafür korrigieren ist dieser Umstand zu vernachlässigen.
3. kann man so sagen

4. Stimmt nicht ganz, aus Robustheits-Gründen wird das Ergebnis in der Regel schon valide sein.

8.2 Varianzanalyse - zwei unabhängige Faktoren

8.2.1 Zwei unabhängige Faktoren

Das Modell der zweifaktoriellen Varianzanalyse erlaubt es, drei Effekte zu testen: den Haupteffekt der ersten sowie der zweiten UV und den Interaktionseffekt.

Statistische H_0en:

$$\begin{array}{lllllllll} \mathrm{H}_{0A}: & \mu_{1\cdot} & = & \mu_{2\cdot} & = & \cdots & = & \mu_{J\cdot} \\ \mathrm{H}_{0B}: & \mu_{\cdot 1} & = & \mu_{\cdot 2} & = & \cdots & = & \mu_{\cdot K} \\ \mathrm{H}_{0I}: & \forall \mu_{jk} & \text{gilt}: & \mu_{jk} & = & \mu_{j\cdot} & +\mu_{\cdot k}- & \mu \end{array}$$

Die H_1 ist dann entsprechend immer "nicht H_0".

Die Auswertung funktioniert dann ganz ähnlich zum einfaktoriellen Fall, wir müssen nur den zusätzlichen between-Faktor angeben. Dabei wird vom `ez`-Paket die folgende etwas ungewöhnliche Syntax zur Verkettung der Faktoren eingeführt:

```
ezANOVA(data = <data>,
        dv = <AV>,
        wid = <VP-Code>,
        between = .(<UV1>,<UV2>))
```

Das `.()` ist eine extra für diesen Zweck eingeführte Liste

Wenn man sich an die ungewöhnliche Notation gewöhnt hat, ist die Anwendung aber ganz leicht:

```
library(ez)
anova_groupXsex <-
  df_wide %>%
  ezANOVA(dv = post_skill,
          wid = vp_nr,
          between = .(group,sex))
```

Der Output ist dann auch nicht weiter überraschend, wir bekommen wieder eine Ergebnistabelle für die ANOVA und eine für den Levene-Test. Die Tabelle für die ANOVA ist jetzt aber natürlich etwas länger[1].

```
anova_groupXsex$ANOVA
```

[1] Und in diesem Beispiel auch nicht mehr ganz richtig. Um zu einer akkuraten Schätzung für das η^2 zu kommen, müsste man eigentlich noch das `observed`-Argument auf `sex` setzen, aber das würde hier jetzt ein bisschen zu weit führen.

Table 8.4

Effect	DFn	DFd	F	p	p<.05	ges
group	3	192	66.5	1.6e-29	*	0.509
sex	1	192	1.59	0.209		0.0082
group:sex	3	192	0.659	0.578		0.0102

8.2.1.1 Aufgabe

```
anova_groupXsex
```

```
## $ANOVA
##      Effect DFn DFd         F            p p<.05
## 1     group   3 192 66.471321 1.602553e-29     *
## 2       sex   1 192  1.586832 2.093085e-01
## 3 group:sex   3 192  0.658699 5.784263e-01
##           ges
## 1 0.509470743
## 2 0.008197002
## 3 0.010187322
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd      SSn      SSd         F         p p<.05
## 1   7 192 9.348968 317.8641 0.8067247 0.5826385
```

Wie interpretieren wir das Ergebnis?

1. Da der Levene-Test nicht signifikant geworden ist können wir keine Varianzhomogenität annehmen. Deswegen können wir die Ergebnisse nicht interpretieren.
2. Nicht alle Effekte sind signifikant geworden, deswegen können wir nichts interpretieren.
3. Da der Test des Haupteffekts der Gruppe signifikant geworden ist, entscheiden wir uns dazu anzunehmen, dass die unterschiedlichen Interventionen zu unterschiedlichen Skillwerten führen.
4. Da der Interaktionseffekt nicht signifikant geworden ist, nehmen wir an, dass der Erwartungswertverlauf parallel ist.

Antwort

1. Blödsinn, genau verkehrt herum
2. Stimmt auch nicht. Dahingegen kann man sich merken, dass der Satz “Der

Interaktionseffekt ist signifikant geworden, deswegen sind die Haupteffekte schwer zu interpretieren" richtig ist.

3. Kann man so sagen
4. Interpretation der H_0

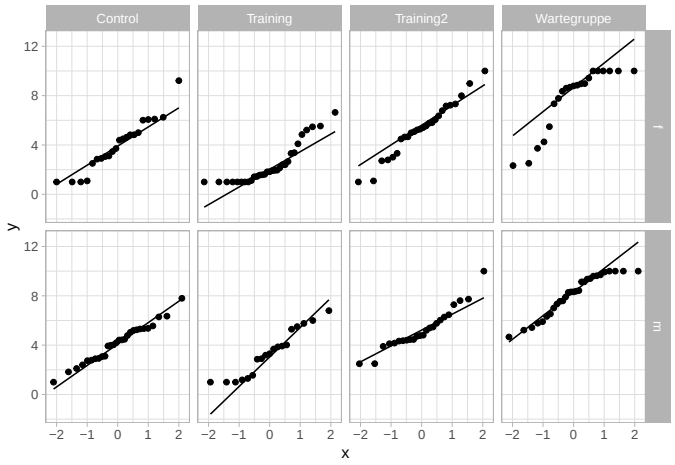
8.2.1.2 Typen von Quadratsummen

8.2.2 Voraussetzungen

Die zweifaktorielle Varianzanalyse hat (ähnlich wie die einfaktorielle) die Voraussetzungen, dass die Zellen Normalverteilt mit gleichen Varianzen sind.

Um das zu überprüfen, können wir einfach die uns schon bekannten Verfahren anwenden. Für die Normalverteiltheit also entweder qq-Plots:

```
df_wide %>%
  ggplot(aes(sample = post_skill)) +
  geom_qq() +
  geom_qq_line()+
  facet_grid(sex ~ group)
```



Oder zellenweise inferenzstatistische Tests, zum Beispiel Shapiro-Wilk:

```
df_wide %>%
  group_by(group, sex) %>%
  summarise(W = shapiro.test(post_skill)$statistic,
            p = shapiro.test(post_skill)$p.value,
            'sign. auf 20%-Niveau' = ifelse(p < .2, '*', ''))
```

```
## # A tibble: 8 x 5
## # Groups:   group [4]
##   group       sex       W         p
```

```
##   <chr>       <chr> <dbl>     <dbl>
## 1 Control     f     0.945 0.246
## 2 Control     m     0.982 0.902
## 3 Training    f     0.810 0.0000782
## 4 Training    m     0.923 0.130
## 5 Training2   f     0.978 0.836
## 6 Training2   m     0.931 0.105
## 7 Wartegruppe f     0.793 0.000508
## 8 Wartegruppe m     0.913 0.0204
##   `sign. auf 20%-Niveau`
##   <chr>
## 1 ""
## 2 ""
## 3 "*"
## 4 "*"
## 5 ""
## 6 "*"
## 7 "*"
## 8 "*"
```

Für die Varianzhomogenität können wir einfach das von `ezANOVA` zurückgegebenen Levene-Test-Ergebnis nutzen:

```
anova_groupXsex$`Levene's Test for Homogeneity of Variance`
```

Table 8.5

DFn	DFd	SSn	SSd	F	p	p<.05
7	192	9.35	318	0.807	0.583	

8.3 Varianzanalyse - zwei abhängige Faktoren

8.3.1 Zwei abhängige Faktoren

Statistische H_0en:

$$\begin{array}{lllllllll} \mathrm{H}_{0A}: & \mu_{1\cdot} & = & \mu_{2\cdot} & = & \cdots & = & \mu_{J\cdot} \\ \mathrm{H}_{0B}: & \mu_{\cdot 1} & = & \mu_{\cdot 2} & = & \cdots & = & \mu_{\cdot K} \\ \mathrm{H}_{0I}: & \forall \mu_{jk} & \text{gilt}: & \mu_{jk} & = & \mu_{j\cdot} & +\mu_{\cdot k}- & \mu \end{array}$$

Für die Durchführung der Zweifaktoriellen Varianzanalyse mit abhängigen Gruppen mit `ezANOVA` müssen natürlich wieder die zu benutzenden Faktoren im `long`-Format vorliegen. Die Erweiterung vom einfaktoriellen Fall funktioniert dann wieder äquivalent zum unabhängigen Fall.

```
ezANOVA(data = <data>,
        dv = <AV>,
        wid = <VP-Code>,
        within = .(<UV1>,<UV2>))
```

Für diese Auswertung passt unser bisheriger Datensatz nicht so ganz. Wir benutzen deswegen kurz ein anderes Beispiel:

In einem (fiktiven) Versuch zum Modell-Lernen wurde 10 Kindern an vier Terminen jeweils ein Video vorgespielt. Die Videos waren Darstellungen des folgenden Designs:

Table 8.6

	Aggressives Modell	Nicht aggressives Modell
Kind als Modell	AV: Fremdurteil Aggressionen	AV: Fremdurteil Aggressionen
Jugendliche:r als Modell	AV: Fremdurteil Aggressionen	AV: Fremdurteil Aggressionen
Erwachsene:r als Modell	AV: Fremdurteil Aggressionen	AV: Fremdurteil Aggressionen

Dabei wurde die AV von einem Beobachter erhoben, der auf einem Fragebogen ein (der Einfachheit halber) normalverteiltes Urteil über das aggressive Verhalten des Kindes nach dem Versuch angegeben hat.

Der zugehörige Datensatz sieht wie folgt aus:

```
aggr_df
```

```
anova_behavXmodel <- aggr_df %>%
  pivot_longer(cols = -1,
               names_to = c('model_behav', 'model', '.value'),
               names_pattern = '(.+)_(.+)_(.+)') %>%
  ezANOVA(dv = aggr,
          wid = Kind_ID,
          within = .(model, model_behav))
```

Das Ergebnis beteht dann wie im einfaktoriellen Fall wieder aus drei Tabellen, einmal dem Ergebnis eines Mauchly-Tests für den ersten Faktor und die Interaktion:

```
anova_behavXmodel$`Mauchly's Test for Sphericity`
```

8.3.1.1 Aufgabe

Warum taucht kein Test für die Sphärizität des Modellverhaltens auf?

Table 8.7

aggrMod_kind_aggr	nAggrMod_kind_aggr	aggrMod_erw_aggr	nAggrMod_erw_aggr	aggrMod_jug_aggr
13.4	8.61	9.08	8.37	11.4
11.4	10.6	4.66	9.11	10.3
12.4	3.22	9.48	10.1	12.5
12.6	5.44	13.6	5.17	9.55
12.4	5.73	15.7	8.51	8.26
11.9	7.27	8.71	1.85	11.9
13.5	5.43	9.23	4.65	9.38
11.9	0.687	4.71	4.45	13.9
14	1.12	11.4	-0.243	10.1
11.9	8.64	8.08	7.11	12.3

Table 8.8

Effect	W	p	p<.05
model	0.895	0.642	
model:model_behav	0.746	0.309	

1. Da das Modellverhalten ein unabhängiger Faktor ist.
2. Bestimmt weil der Test des zweiten Faktors irgendwie in den anderen beiden Tests implizit ist.
3. Da Zirkularität bei einer 2x2-Kovarianzmatrix immer gegeben ist.
4. Weil R einen Fehler gemacht hat.

Antwort

1. ist Unsinn
2. die Voraussetzung gilt für alle drei Variablen, wir könnten auch nicht alle Tests durchführen wollen
3. Stimmt

4. ist bei einem gut etablierten und getesteten Paket eher unwahrscheinlich, aber kann natürlich sein

Als zweites dem Ergebnis der Varianzanalyse, natürlich wieder mit drei Zeilen für Haupteffekte und Interaktion:

```
anova_behavXmodel$ANOVA
```

Table 8.9

Effect	DFn	DFd	F	p	p<.05	ges
model	2	18	1.87	0.183		0.0602
model_behav	1	9	44.7	9e-05	*	0.56
model:model_behav	2	18	2.72	0.0927		0.0769

Und zu guter Letzt dem Ergebnis der Varianzanalyse nach den bekannten Korrekturen:

```
anova_behavXmodel$`Sphericity Corrections`
```

Table 8.10

Effect	GGe	p[GG]	p[GG]<.05	HFe	p[HF]	p[HF]<.05
model	0.905	0.187		1.12	0.183	
model:model_behav	0.797	0.108		0.942	0.0968	

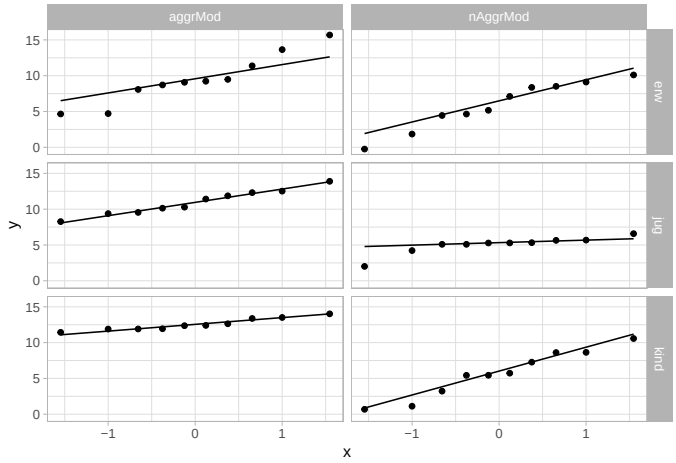
8.3.2 Voraussetzungen

Die abhängige zweifaktorielle Varianzanalyse hat (ähnlich wie die einfaktorielle) die Voraussetzungen, dass die Zellen Normalverteilt sind und die Kovarianzmatrix Sphärizität aufweist (die Varianzen der Differenzen sind gleich).

Um das zu überprüfen, können wir auch hier einfach die uns schon bekannten Verfahren anwenden. Für die Normalverteiltheit also entweder qq-Plots:

```
aggr_df %>%
  pivot_longer(cols = -1,
               names_to = c('model_behav', 'model', '.value'),
               names_pattern = '(.+)_(.+)_(.+)') %>%
  ggplot(aes(sample = aggr)) +
  geom_qq() +
```

```
  geom_qq_line()+
  facet_grid(model ~ model_behav)
```



Oder zellenweise inferenzstatistische Tests, zum Beispiel Shapiro-Wilk:

```
aggr_df %>%
  pivot_longer(cols = -1,
               names_to = c('model_behav', 'model', '.value'),
               names_pattern = '(.+)_(.+)_(.+)') %>%
  group_by(model_behav, model) %>%
  summarise(W = shapiro.test(aggr)$statistic,
            p = shapiro.test(aggr)$p.value,
            'sign. auf 20%-Niveau' = ifelse(p < .2, '*', ''))
```

```
## # A tibble: 6 x 5
## # Groups:   model_behav [2]
##   model_behav model     W      p
##   <chr>       <chr> <dbl>  <dbl>
## 1 aggrMod     erw   0.938 0.528
## 2 aggrMod     jug   0.976 0.938
## 3 aggrMod     kind  0.929 0.435
## 4 nAggrMod    erw   0.940 0.556
## 5 nAggrMod    jug   0.800 0.0144
## 6 nAggrMod    kind  0.951 0.675
##   `sign. auf 20%-Niveau`
##   <chr>
## 1 ""
## 2 ""
## 3 ""
## 4 ""
## 5 "*"
```

```
## 6 ""
```

Für die Sphärizität können wir einfach das von `ezANOVA` zurückgegebenen Mauchly-Test-Ergebnis nutzen:

```
anova_behavXmodel$`Mauchly's Test for Sphericity`
```

Table 8.11

Effect	W	p	p<.05
model	0.895	0.642	
model:model_behav	0.746	0.309	

8.4 Varianzanalyse - unabhängige und abhängige Faktoren

8.4.1 Zweifaktorielle Varianzanalyse mit split-plot-Design

Ein Split-Plot-Design liegt im zweifaktoriellen Fall vor, wenn within- und between-Faktoren kombiniert werden.

Für die Durchführung der Zweifaktoriellen Varianzanalyse mit split-plot-Design über `ezANOVA` müssen natürlich genau wie in den vorangegangenen Fällen die zu benutzenden Faktoren im `long`-Format vorliegen.

Wir wollen die Haupteffekte und die Interaktion vom Testzeitpunkt und dem Treatment untersuchen. Dafür müssen wir natürlich wieder zuerst den Datensatz pivotieren um dann entsprechend das `within` und das `between`-Argument setzen.

```
anova_timeXgroup <- df_wide %>%
  pivot_longer(cols = contains('skill'),
               names_to = 'time',
               values_to = 'skill') %>%
  ezANOVA(dv = skill,
          wid = vp_nr,
          between = group,
          within = time)
```

Die Ergebnisse sind dann wie im abhängigen Fall in die drei Tabellen für den Mauchly-Test

```
anova_timeXgroup$`Mauchly's Test for Sphericity`
```

Die Ergebnisse der ANOVA:

Table 8.12

Effect	W	p	p<.05
time	0.95	0.00707	*
group:time	0.95	0.00707	*

```
anova_timeXgroup$ANOVA
```

Table 8.13

Effect	DFn	DFd	F	p	p<.05	ges
group	3	196	73.2	8.36e-32	*	0.265
time	2	392	5.92	0.00293	*	0.0201
group:time	6	392	25.4	1.79e-25	*	0.209

Und die Sphärizitäts-Korrektur aufgeteilt.

```
anova_timeXgroup$`Sphericity Corrections`
```

Table 8.14

Effect	GGe	p[GG]	p[GG]<.05	HFe	p[HF]	p[HF]<.05
time	0.953	0.00343	*	0.962	0.00333	*
group:time	0.953	2.19e-24	*	0.962	1.35e-24	*

Wie interpretieren wir das?

1. Es gibt eine signifikante Interaktion zwischen Untersuchungszeitpunkt und Experimentalgruppe.
2. Der signifikante Haupteffekt von `time` führt uns zu der Annahme, dass die Interventionen eine Veränderung über den Untersuchungsverlauf auslösen, der signifikante Haupteffekt von `group` führt uns zu der Annahme, dass es einen Unterschied in der Wirkung der Treatments gibt.
3. Da wir die Voraussetzungen nicht überprüft haben, können wir keine Aussage aus den Ergebnissen ziehen.

Table 8.15

Effect	DFn	DFd	F	p	p<.05	ges
group	3	196	73.2	8.36e-32	*	0.265
time	2	392	5.92	0.00293	*	0.0201
group:time	6	392	25.4	1.79e-25	*	0.209

Table 8.16

Effect	GGe	p[GG]	p[GG]<.05	HFe	p[HF]	p[HF]<.05
time	0.953	0.00343	*	0.962	0.00333	*
group:time	0.953	2.19e-24	*	0.962	1.35e-24	*

4. Die Ergebnisse legen einen linearen Zusammenhang zwischen Untersuchungsverlauf, Experimentalgruppe und motorischem Skill nahe.

Antwort

1. Stimmt.

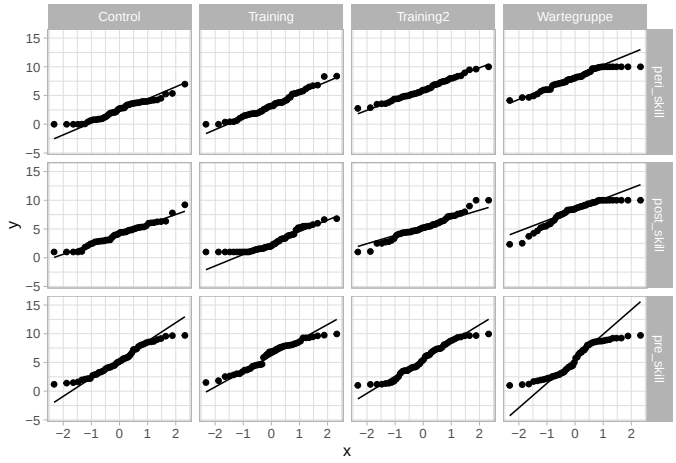
8.4.2 Voraussetzungen

Die zweifaktorielle Varianzanalyse im split-plot-Design hat als Voraussetzung eine Kombination der aus dem jeweiligen einfaktoriellen Fall bekannten. So setzen wir voraus, dass die AV über die unabhängigen Faktor-Stufen Varianzhomogenität aufweist und ihre Kovarinazmatrix zirkulär ist.

Außerdem ist wieder Normalverteiltheit in den Zellen Voraussetzung.

Um das zu überprüfen, können wir auch hier einfach die uns schon bekannten Verfahren anwenden. Für die Normalverteiltheit also wieder entweder qq-Plots:

```
df_wide %>%
  pivot_longer(cols = contains('skill'),
               names_to = 'time',
               values_to = 'skill') %>%
  ggplot(aes(sample = skill)) +
  geom_qq() +
  geom_qq_line()+
  facet_grid(time ~ group)
```



Oder zellenweise inferenzstatistische Tests, zum Beispiel Shapiro-Wilk:

```
df_wide %>%
  pivot_longer(cols = contains('skill'),
               names_to = 'time',
               values_to = 'skill') %>%
  group_by(time, group) %>%
  summarise(W = shapiro.test(skill)$statistic,
            p = shapiro.test(skill)$p.value,
            'sign. auf 20%-Niveau' = ifelse(p < .2, '*', ''))
```

```
## # A tibble: 12 x 5
## # Groups:   time [3]
##    time       group           W         p
##    <chr>      <chr>       <dbl>     <dbl>
##  1 peri_skill Control     0.953 0.0433
##  2 peri_skill Training    0.968 0.184
##  3 peri_skill Training2   0.982 0.653
##  4 peri_skill Wartegruppe 0.928 0.00453
##  5 post_skill Control     0.971 0.247
##  6 post_skill Training    0.870 0.0000567
##  7 post_skill Training2   0.977 0.441
##  8 post_skill Wartegruppe 0.866 0.0000419
##  9 pre_skill  Control     0.953 0.0446
## 10 pre_skill  Training    0.943 0.0169
## 11 pre_skill  Training2   0.944 0.0187
## 12 pre_skill  Wartegruppe 0.897 0.000384
##    `sign. auf 20%-Niveau`
##    <chr>
##  1 "*"
##  2 "*"
```

```
##  3 ""
##  4 "*"
##  5 ""
##  6 "*"
##  7 ""
##  8 "*"
##  9 "*"
## 10 "*"
## 11 "*"
## 12 "*"
```

Für die Sphärizität können wir einfach das von `ezANOVA` zurückgegebenen Mauchly-Test-Ergebnis nutzen:

```
anova_timeXgroup$`Mauchly's Test for Sphericity`
```

Table 8.17

Effect	W	p	p<.05
time	0.95	0.00707	*
group:time	0.95	0.00707	*

8.5 Beliebige Linearkontraste

8.5.1 Kontraste, ein Faktor

Wie schon bei der Varianzanalyse mit einem unabhängigen Faktor erwähnt, können mit `pairwise.t.test` alle Zellvergleiche als α-korrigierte t-Tests durchgeführt werden.

Als Alternative gibt es mit den beliebigen Linearkontrasten eine flexiblere Methode, um spezielle Vergleiche anzustreben. Jeder dieser Kontraste (ψ) ist eine Linearkombination der Gruppenerwartungswerte μ_j mit den Koeffizienten c_j, wobei die Koeffizienten in Summe 0 ergeben. Mit $1 \leq \mathrm{j} \leq \mathrm{J}$ könnte man entsprechend Kontraste wie folgt formulieren:

$$\psi = \sum_{1}^{J} c_j \cdot \mu_j$$

Um diese theoretischen Kontraste zu testen, können wir den Dunns Test für mehrfache Vergleiche nutzen. Dieser nutzt die unter der Gültigkeit der Nullhypothese $H_0 : \psi = 0$ t_{N-J}-verteilte Teststatistik $t = \frac{\hat{\psi}}{\sqrt{\sum_1^J \frac{c_j^2}{n_j} MS_{Fehler}}}$. Dabei

steht N für die Stichprobengröße, MS_w für die mittlere Quadratsumme aus dem Nenner der einfaktoriellen Varianzanalyse, $\hat{\psi}$ für den empirischen Schätzer $\sum_1^J c_j \cdot M_j$ des Kontrasts und n_j für die Zellbesetzung.

Da wir mit einer t-Verteilung testen, können wir natürlich auch gerichtete Hypothesen aufstellen.

In R lässt sich dieser Test mit der `glht` (generalized linear hypothesis test)-Funktion aus dem `multcomp`-Paket durchführen.

Dazu brauchen wir zuerst ein `aov`-Modell für die Quadratsumme, die wir entweder mit der `aov`-Funktion erstellen können…

```
aov_mod <- aov(post_skill ~ group, data = df_wide)
```

…oder uns mit Hilfe des `return_aov`-Arguments der `ezANOVA`-Funktion mit ausgeben lassen können:

```
ez_anova <- df_wide %>%
  ezANOVA(dv = post_skill,
          wid = vp_nr,
          between = group,
          return_aov = T)
ez_anova$aov
```

```
## Call:
##    aov(formula = formula(aov_formula), data = data)
##
## Terms:
##                     group Residuals
## Sum of Squares  738.9439  700.5886
## Deg. of Freedom        3       196
##
## Residual standard error: 1.890617
## Estimated effects may be unbalanced
```

Als nächsten Schritt müssen wir unsere Linearkontraste formulieren. In unserem Fall könnten wir auf die Idee kommen, die Trainings-Bedingungen gegen die Kontrollbedingung testen zu wollen.

Die einfachste (und übersichtlichste) Möglichkeit diesen Kontrast zu formulieren ist es, die dem gewünschten Kontrast entsprechende Linearkombination als Text zu notieren:

```
contr_hypothesis <- '2 * Control - (Training + Training2) <= 0'
```

Man kann die Kontraste unter Anderem zwar auch als Matrix formulieren, die Text-Variante hat aber die entscheidenden Vorteile dass 1. die Ordnung der Faktorstufen nicht bekannt sein muss und 2. auf den ersten Blick erkennbar ist, welche Richtung welcher Hypothese getestet werden soll.

Unsere Nullhypothese ist also:

$$H_0 : 2\mu_{Control} - (\mu_{Training} + \mu_{Training2}) = \psi \leq 0$$

Um diese Hypothese zu testen, müssen wir sie der `mcp`-Funktion auch aus dem `glht`-Paket übergeben. Dabei müssen wir darauf achten, dass wir den Namen des in der Varianzanalyse genutzten between-Faktors als Argument-Namen nutzen:

```
library(multcomp)
cont <- mcp(group = contr_hypothesis)
```

Mit diesen `mcp`-Objekt können wir nun endlich den Kontrast-Test rechnen:

```
glht(ez_anova$aov, cont)
```

```
##
##   General Linear Hypotheses
##
## Multiple Comparisons of Means: User-defined Contrasts
##
##
## Linear Hypotheses:
##                                              Estimate
## 2 * Control - (Training + Training2) <= 0  0.08774
```

Achtung: Das Modell muss ursprünglich mit einem `factor` als Gruppierungsfaktor gerechnet worden sein. Netterweise wandelt `ezANOVA` das aber auch für uns um.

Das Ergebnis von `glht` lässt sich auch wieder mit `broom` darstellen:

```
glht(ez_anova$aov, cont) %>%
  broom::tidy()
```

Table 8.18

term	contrast	null.value	estimate	std.error	statistic	adj.p.value
group	2 * Control - (Training + Training2)	0	0.0877	0.655	0.134	0.447

Mit Setzen des `test` -Arguments können wir außerdem wieder auswählen, wie wir den p-Wert adjustieren, indem wir einen Aufruf der `adjusted`-Funktion übergeben. Bei einem Test müssen wir nicht unbedingt adjustieren, können also wie folgt vorgehen:

```
glht(ez_anova$aov, cont) %>%
  broom::tidy(test = adjusted('none'))
```

Table 8.19

term	contrast	null.value	estimate	std.error	statistic	p.value
group	2 * Control - (Training + Training2)	0	0.0877	0.655	0.134	0.447

Die Möglichkeit der Korrektur weißt schon darauf hin, dass wir diese Methode auch für multiple Vergleiche einsetzen können.

Dafür müssen wir aus der einen Hypothese einfach einen Vektor aus Hypothesen machen:

So könnten wir zusätzlich zum bisherigen Kontrast auch alle Zellen noch mit der Wartegruppe vergleichen wollen. Dafür würden wir das ganze wie folgt formulieren:

```
contr_hypothesis <- c('2 * Control - (Training + Training2) <= 0',
                      'Wartegruppe - Control <= 0',
                      'Wartegruppe - Training <= 0',
                      'Wartegruppe - Training2 <= 0')

glht(ez_anova$aov,mcp(group = contr_hypothesis)) %>%
  broom::tidy(test = adjusted('holm'))
```

Table 8.20

term	contrast	null.value	estimate	std.error	statistic	adj.p.value
group	2 * Control - (Training + Training2)	0	0.0877	0.655	0.134	0.447
group	Wartegruppe - Control	0	3.9	0.378	10.3	0
group	Wartegruppe - Training	0	5.21	0.378	13.8	0
group	Wartegruppe - Training2	0	2.67	0.378	7.06	2.81e-11

8.5.2 Kontraste, mehr als ein Faktor

Um Kontraste über mehr als einen between-Faktor rechnen zu können, müssen wir unser mehrfaktorielles Design in ein assoziiert einfaktorielles umwandeln.

Dazu können wir die `interaction`-Funktion aus dem `base`-Umfang nutzen, die zwei Faktoren zu einem kombiniert.

Als Beispiel betrachten wir nochmal die Unterschiede zwischen den Treatment-Gruppen und den Geschlechtern. Um eine assoziierte einfaktorielle Varianzanal-

yse über diese Faktoren zu rechnen, müssen wir einfach ein `mutate` mit einer `interaction` hinzufügen:

```
df_wide_I <- df_wide %>%
  mutate(groupXsex = interaction(group, sex, sep = '_'))

assoz_anova <- df_wide_I %>%
  ezANOVA(dv = post_skill,
          wid = vp_nr,
          between = groupXsex,
          return_aov = T)
```

Auf dieser Basis können wir dann einfach wie im einfaktoriellen Fall unsere Kontraste formulieren. Zum Beispiel können wir je einen Haupteffekts- und Interaktionseffektskontrast formulieren und testen. Da die Kombinationen der Stufen sehr viele sind, bietet es sich hier an, die Matrix-Schreibweise zu nutzen. Die Textschreibweise würde aber auch funktionieren.

In die Matrix schreiben wir zeilenweise die Gewichte, die getestet werden sollen.

Damit wir genau wissen, welches Gewicht an welche Stelle kommt, können wir uns zuerst die Stufen des Interaktionsfaktors ausgeben lassen:

```
levels(df_wide_I$groupXsex)
```

```
## [1] "Control_f"     "Training_f"    "Training2_f"
## [4] "Wartegruppe_f" "Control_m"     "Training_m"
## [7] "Training2_m"   "Wartegruppe_m"
```

Nun erstellen wir eine Matrix, bei der jede Spalte der Ordnung der Levels entsprechend dem zu gewichtenden Erwartungswert einer Zelle entspricht:

```
contrasts <- matrix(
  c(
    1,1,1,1,-1,-1,-1,-1, # Haupteffekt des Geschlechts
    1,-1,-1,1,1,-1,-1,1, # Haupteffekt des Trainings
    1,-1,-1,1,-1,1,1,-1 # Interaktionseffekt (Umgekehrte Wirkung bei Männern)
  ),
  nrow = 3,
  byrow = T # damit zeilenweise aufgefüllt wird.
)
```

Der Übersicht halber können wir diese Kontraste auch noch für den Output benennen, außerdem müssen wir noch Spaltennamen für die `mcp`-Funktion hinzufügen:

```
colnames(contrasts) <- levels(df_wide_I$groupXsex)
rownames(contrasts) <- c('f > m', 'T1, T2 < W, C', 'Interaktion')
contrasts
```

```
##                Control_f Training_f Training2_f
## f > m                  1          1           1
## T1, T2 < W, C          1         -1          -1
## Interaktion            1         -1          -1
##                Wartegruppe_f Control_m Training_m
## f > m                      1        -1         -1
## T1, T2 < W, C              1         1         -1
## Interaktion                1        -1          1
##                Training2_m Wartegruppe_m
## f > m                   -1            -1
## T1, T2 < W, C           -1             1
## Interaktion              1            -1
```

Und diese Matrix können wir dann wie gewohnt für die Kontrasttests nutzen. Dabei können wir hier jetzt (da wir in der Kontrastmatrix noch keine Richtung vorgegeben haben) wie gewohnt die Testrichtung mit `alternative` festlegen:

```
glht(assoz_anova$aov, mcp(groupXsex = contrasts),
     alternative = 'greater') %>%
  broom::tidy()
```

Table 8.21

term	contrast	null.value	estimate	std.error	statistic	adj.p.value
groupXsex	f > m	0	-1.4	1.08	-1.29	0.999
groupXsex	T1, T2 < W, C	0	7.67	1.08	7.07	3.47e-11
groupXsex	Interaktion	0	0.452	1.08	0.417	0.698

Bibliography

Tukey, J. W. (1977). *Exploratory Data Analysis*, volume 2. Reading, Mass.