Übungszettel Arbeitsumgebung und Datentransformation

M.Psy.205, Dozent: Dr. Peter Zezula

Johannes Brachem (johannes.brachem@stud.uni-goettingen.de)

Links

Übungszettel als PDF-Datei zum Drucken

Hinweise zur Bearbeitung

1. Bitte beantworten Sie die Fragen in einer .Rmd Datei. Sie können Sie über `Datei > Neue Datei > R Markdown...` eine neue R Markdown Datei erstellen. Den Text unter dem *Setup Chunk* (ab Zeile 11) können Sie löschen.
2. Informationen, die Sie für die Bearbeitung benötigen, finden Sie auf der Website der Veranstaltung
3. Zögern Sie nicht, im Internet nach Lösungen zu suchen. Das effektive Suchen nach Lösungen für R-Probleme im Internet ist tatsächlich eine sehr nützliche Fähigkeit, auch Profis arbeiten auf diese Weise. Die beste Anlaufstelle dafür ist der R-Bereich der Programmiererplattform Stackoverflow
4. Auf der Website von R Studio finden Sie sehr hilfreiche Übersichtszettel zu vielen verschiedenen R-bezogenen Themen. Ein guter Anfang ist der Base R Cheat Sheet

Tipp der Woche

Mit der Tastenkombination `ctrl` + `alt` + `i` (Windows) oder `cmd` + `alt` + `i` (Mac) können Sie per Knopfdruck ein neues R-Code-Feld (*Chunk*) in R-Markdown Dateien erstellen.

Aufgabe 1: R Markdown

a) Neue Markdown Datei öffnen

1. Laden Sie unter diesem Link unter diesem Link unsere Vorlage für die Erstellung von R-Markdown Dokumenten zur Bearbeitung unserer Übungszettel herunter.(Hinweis: Sie können auch eine neue .Rmd Datei erstellen, indem Sie auf `Datei > Neue Datei > R Markdown...` klicken. Mit unserer Vorlage ist der Start aber vermutlich einfacher.)
2. Speichern Sie die Datei unter einem sinnvollen Namen in einem sinnvollen Ordner ab. In diesem Ordner sollten Sie bestenfalls in der Folge alle .Rmd-Dateien für die Bearbeitung der Übungszettel speichern. Öffnen Sie nun die Datei.
3. Lassen Sie die Datei mit `Strg` + `Shift` + `K` (Windows) oder `Cmd` + `Shift` + `K` (Mac) rendern. Sie sollten nun im “Viewer” unten rechts eine “schön aufpolierte” Version ihrer Datei sehen.

b) Code-Chunks benutzen

1. Erstellen Sie einen R-Code-Chunk mit `ctrl` + `alt` + `i` (Windows) oder `cmd` + `alt` + `i` (Mac)
2. Tippen Sie `5 + 3` in den grau hinterlegten Code-Chunk. Führen Sie die Zeile aus, indem Sie `strg` + `enter` (Windows) oder `cmd` + `enter` drücken. Das Ergebnis, `8` sollte Ihnen in der Konsole nun angezeigt werden.
3. Tippen Sie `test <- 5 + 3` in eine neue Zeile in den Code-Chunk und führen Sie auch diese Zeile aus. Das Ergebnis sollte Ihnen diesmal nicht direkt angezeigt werden, stattdessen haben Sie im *Workspace* oben rechts ein neues Objekt namens "test".
4. Tippen Sie nun `test` in eine neue Zeile in den Code-Chunk und führen Sie sie aus. Nun sollte Ihnen das Ergebnis wieder in der Konsole (unten links) angezeigt werden.
5. Tippen Sie nun `# test anzeigen` in die **gleiche Zeile wie in der vorherigen Aufgabe** und führen Sie die Zeile erneut aus. Sie können sehen, dass die keinen Effekt hatte: `#` kennzeichnet *Kommentare*, alles was hinter einem `#` steht, wird nicht ausgerechnet. Dies ist nützlich für (kurze!) Erklärungen. Längere Erklärungen sollten außerhalb der *Chunks* als normaler Text geschrieben werden.
6. Klicken Sie auf `Tools > Global Options` und wählen Sie auf der linken Seite "R Markdown" aus. Überprüfen Sie, ob das Häkchen bei "*Show Output inline for all Markdown documents*" gesetzt ist. Wenn ja, **entfernen Sie es**. Das Häkchen sollte **nicht** gesetzt sein.

c) Arbeitsverzeichnis setzen

Wenn Sie Dateien in R einlesen möchten, oder aus R abspeichern möchten, ist es wichtig, dass Sie ein Arbeitsverzeichnis verwenden. Das ist der Ordner auf ihrem Computer, in dem R nach Dateien sucht, und in dem R Dateien abspeichert.

1. Setzen Sie das Arbeitsverzeichnis, indem Sie oben auf `Session > Set Working Directory > To Source File Location` klicken.
2. Betrachten Sie die Konsole (unten links). Sie sehen dort einen Befehl, der ähnlich wie dieser aussieht: `setwd("P:/mv")`
3. Kopieren Sie diesen Befehl und fügen ihn in den dafür vorgesehenen Code-Chunk am Anfang ihrer .Rmd-Datei ein. Nun können Sie immer, wenn Sie die Datei erneut öffnen, diesen Befehl ausführen und dadurch automatisch das richtige Arbeitsverzeichnis setzen. **Achtung**: Das funktioniert natürlich nur, wenn Sie immer den gleichen Ordner auf Ihrem Computer verwenden.

Aufgabe 2: Pakete

1. Installieren Sie das Paket-System `tidyverse`, indem Sie `install.packages("tidyverse")` ausführen. Zu diesem System gehören mehrere sehr nützliche Pakete, z.B. `dplyr` für Datenaufbereitung und `ggplot2` für Plots. Hinweis: Auf den PCs im CIP-Pool müssen Sie diesen Befehl nicht ausführen, die Pakete sind installiert.

2. Laden Sie das Paket-System `tidyverse` mit dem Befehl `library(tidyverse)`. Sie sollten nun diese Anzeige bekommen, die Ihnen sagt, welche Pakete standardmäßig zum `tidyverse` gehören. Diese werden automatisch mitgeladen. Schreiben Sie den Syntaxbefehl zum Laden in den dafür vorgesehenen Code-Chunk am Anfang der .Rmd-Datei.

Aufgabe 3: Mit Objekten umgehen

a) Erstellen von Objekten

Erstellen Sie die folgenden Objekte:

1. Einen Vektor `a1`, der die Zahlen von 1 bis 5 enthält.
2. Einen Vektor `a2` der die Zahlen von 0 bis 4.5 in Schritten von 0.5 enthält (d.h. er beinhaltet 10 Zahlen).
3. Einen Vektor `a3` der Länge 10, der Zahlen von 0 bis 85 in gleichmäßigen abständen enthält.
4. Eine Matrix, die die Zahlen 1 bis 9 in drei Zeilen enthält. Sie sollte so aussehen:

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

5. Einen `tibble`-Datensatz namens `a5`, der die beiden Vektoren `a2` und `a3` enthält.

Hinweis: `tibble`s sind moderne Formen von `data.frame`s. Wir empfehlen, dass Sie immer `tibble`s verwenden. Wenn Sie Daten mit den `readr`-Befehlen `read_csv()` oder `read_delim()` einlesen, werden diese automatisch als `tibble` gespeichert.

6. Einen `tibble`-Datensatz namens `a6`, der die beiden Vektoren `a2` und `a3` enthält. Bei diesem `data frame` sollten Sie bei der Erstellung die beiden Vektoren umbenennen: Nennen Sie `a2` "motivation" und `a3` "result"
7. Eine Liste `a7`, die alle bisher erstellen Elemente beinhaltet.

b) Auf Elemente zugreifen

1. Greifen Sie auf das dritte Element des Vektors `a1` zu.
2. Greifen Sie auf die dritte Spalte der Matrix `a4` zu.
3. Greifen Sie abermals auf die dritte Spalte der Matrix `a4` zu. Sorgen Sie diesmal dafür, dass die Dimensionen der Matrix erhalten bleiben. Das Ergebnis sollte wie folgt aussehen:

```
##      [,1]
## [1,]    3
## [2,]    6
## [3,]    9
```

4. Greifen Sie auf die zweite bis sechste Zeile des data.frame `a6` zu.
5. Greifen Sie auf die Spalte / Variable `result` im data.frame `a6` zu.

c) Objekte manipulieren

1. Addieren Sie `3` zu jedem Element des Vektors `a1` und speichern das Ergebnis unter dem Namen `c1`
2. Erstellen Sie einen neuen tibble `c2`, indem Sie nur die Zeilen 3 und 4 aus dem tibble `a6` auswählen.

Aufgabe 4: Logische Abfragen

1. Greifen Sie auf alle Zeilen des Datensatzes `a6` zu, bei denen `result` größer als 50 ist.
2. Greifen Sie auf alle Zeilen des Datensatzes `a6` zu, bei denen `result` zwischen 30 und 70 liegt.
3. Greifen Sie auf alle Zeilen des Datensatzes `a6` zu, bei denen `result` *kleiner* als 30, oder *größer* als 70 ist.
4. Nutzen Sie sum() und eine logische Abfrage, um die Anzhal von Zeilen in `a6` zu erhalten, bei denen `result` größer als 20 ist.

Aufgabe 5: Daten einlesen

1. Laden Sie den Datensatz *seatbelts.csv* hier herunter. Legen Sie diese Datei in einen Ordner namens "data" in Ihrem Ordner für die Statistik-Übungszettel.
2. Lesen Sie den Datensatz *seatbelts.csv* mit dem Befehl `read_csv()` ein und speichern Sie ihn in R unter dem Namen `seatbelt_data`. Tipp: Wenn Sie die Datei wie in 5.1 beschrieben in einem Unterordner "data" in Ihrem Arbeitsverzeichnis abgespeichert haben, ist der Pfad zur Datei für Sie nun `"data/seatbelts.csv"`.

Hinweis: `read_csv` und `read.csv` sind unterschiedliche Befehle. `read_csv` ist die modernere Variante und wird von uns empfohlen. Mehr können Sie bei Interesse hier erfahren.

3. Schauen Sie sich kurz die Bedeutung der Variablen in diesem Datensatz an. Unter diesem Link finden Sie eine kurze Beschreibung.

Aufgabe 6: Mit der Pipe %>% umgehen.

1. Schreiben Sie den folgenden Befehl um, so dass Sie %>% verwenden, um den Code besser lesbar zu machen.

```
filter(seatbelt_data, year == 1969)
```

2. Schreiben Sie den folgenden Befehl ebenfalls mit der Pipe %>% um, um ihn besser lesbar zu machen.

```
select(filter(seatbelt_data, year == 1969, kms > 10000), DriversKilled)
```

3. Nehmen Sie Ihren Code von Aufgabe 6.1 und nutzen Sie ihn, um ein *Subset* von `seatbelt_data` namens `seatbelt_data_69` zu erzeugen, in dem nur die Daten aus dem Jahr 1969 enthalten sind.

Aufgabe 7: Mit `dplyr` umgehen

Hinweise

- Bitte nutzen Sie wann immer möglich die piping-Schreibweise mit %>% für diese und ähnliche Aufgaben. Dies ist in unser aller Interesse, da so die Lesbarkeit und Nachvollziehbarkeit ihres Codes maximiert wird.

- Lassen Sie sich über `?<funktion>` die Hilfe-Seite zu den vorgeschlagenen Funktionen anzeigen. Dort sehen sie alle Befehle, die Sie in einer Funktion verwenden können. Zum Beispiel zeigt Ihnen `?arrange` die Hilfe-Seite zum Befehl `arrange` an. (Da es mehrere Funktionen namens "arrange" gibt, müssen Sie zunächst per Klick auswählen, zu welcher Sie sich die Hilfe anzeigen lassen wollen.)

Aufgaben

1. Nutzen Sie den Befehl `filter()`, um sich alle Daten aus dem Monat *Januar* im Datensatz `seatbelt_data` anzeigen zu lassen.
2. Nutzen Sie den Befehl `select()`, um sich außerdem nur die Daten zu `DriversKilled`, `law`, `year` und `month` anzeigen zu lassen.
3. Nutzen Sie den Befehl `mutate()`, um eine neue Variable namens `id` zu erstellen, die die Zahlen von 1 bis 192 enthält. Überschreiben Sie `seatbelt_data` mit dem Ergebnis.
4. Nutzen Sie den Befehl `arrange()`, um den Datensatz anhand von `id` *abwärts* zu sortieren.
5. Nutzen Sie den Befehl `group_by()`, um im unbearbeiteten Datensatz `seatbelt_data` die Daten nach Jahren zu gruppieren. Nutzen Sie anschließend `summarize()`, um sich die mittlere Anzahl von Todesfällen pro Jahr anzeigen zu lassen.
6. Nutzen Sie den Befehl `rename()`, um die Variable `DriversKilled` in `drivers_killed` und `PetrolPrice` in `petrol_price` umzubenennen.

Aufgabe 8: Long & Wide Data

Es ist für fast alle Vorgänge in R am besten, wenn Sie Daten im *long*-Format haben. Das liegt einfach an der Art, wie R funktioniet.

Wide-Data sieht so aus:

```
## # A tibble: 4 x 4
##    subj gender    t1    t2
##   <int> <chr>  <dbl> <dbl>
## 1     1 m          2     6
## 2     2 m          3     5
## 3     3 f          4     4
## 4     4 f          5     3
```

Long-Data sieht so aus (dies sind die gleichen Daten):

```
## # A tibble: 8 x 4
##    subj gender time  value
##   <int> <chr>  <chr> <dbl>
## 1     1 m      t1        2
## 2     2 m      t1        3
## 3     3 f      t1        4
## 4     4 f      t1        5
## 5     1 m      t2        6
## 6     2 m      t2        5
## 7     3 f      t2        4
## 8     4 f      t2        3
```

In der Psychologie sind unsere Roh-Daten oft im *wide*-Format. Das Paket `tidyr` bietet die Möglichkeit, diese Daten einfach in das *long*-Format zu bringen. Genaueres dazu können Sie hier finden.

Aufgaben

1. Führen Sie den unten stehenden Befehl aus, um den Datensatz `wide_data` zu erzeugen.

```
wide_data <- tibble(subj = 1:4,
                    gender = c("m", "m", "f", "f"),
                    t1 = c(2,3,4,5),
                    t2 = c(6,5,4,3))
```

2. Nutzen Sie den Befehl `gather()` mit den Optionen `key` und `value`, um einen Datensatz `long_data` zu erzeugen. Dieser sollte genau so aussehen, wie im Beispiel oben.

Aufgabe 9: Rendern (knit)

Lassen Sie die Datei wie in Aufgabe 1 a)3 mit `Strg` + `Shift` + `K` (Windows) oder `Cmd` + `Shift` + `K` (Mac) rendern. Sie sollten nun im "Viewer" unten rechts eine "schön aufpolierte" Version ihrer Datei sehen. Falls das klappt: Herzlichen Glückwunsch! Ihr Code kann vollständig ohne Fehlermeldung gerendert werden. Falls nicht: Nur Mut, das wird schon noch! Vielleicht schaffen wir es ja in der Übung gemeinsam.

Literatur

Anmerkung: Diese Übungszettel basieren zum Teil auf den "Smart Alex" Aufgaben aus dem Lehrbuch *Discovering Statistics Using R* (Field, Miles & Field, 2012). Sie wurden für den Zweck dieser Übung modifiziert, und der verwendete R-Code wurde aktualisiert.

Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R*. London: SAGE Publications Ltd.

Version: 21 April, 2022 08:40