B1 - Data Processing

Kolb

07 August 2018

Inhalt dieses Abschnitts

- Wie bekommt man einen Überblick über die Daten
- Indizieren von Vektoren, Datensätzen und Listen
- Wie geht man mit fehlenden Werten um
- Schleifen und Funktionen
- Zusammenhänge zwischen Variablen

data.frame's

- Beispieldaten importieren:

```
library("readstata13")
dat <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
```

```
typeof(dat)
```

```
## [1] "list"
```

```
head(names(dat))
```

```
## [1] "z000001z" "z000002z" "z000003z" "z000005z" "a11c019a"
```

Anzahl Zeilen und Spalten

- Anzahl der Zeilen/Spalten ermitteln

```
nrow(gpdat) # Zeilen
```

```
## [1] 1222
```

```
ncol(gpdat) # Spalten
```

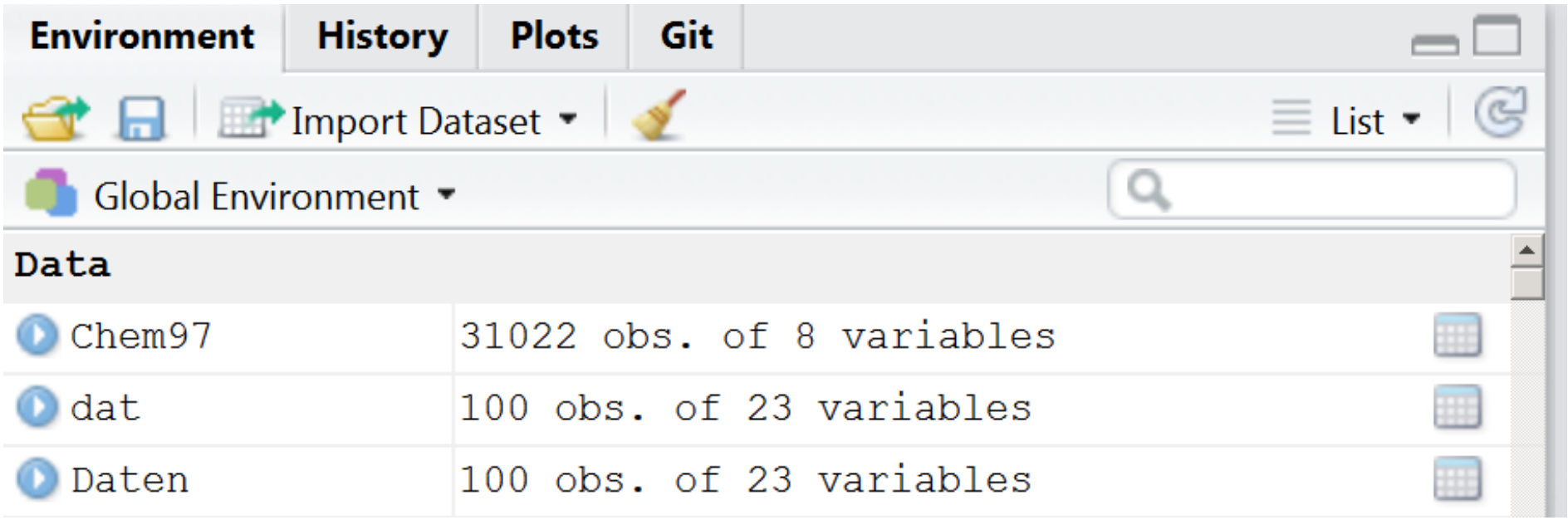
```
## [1] 1192
```

Die Daten ansehen

- Die ersten Zeilen sehen:

```
head(gpdat) # erste Zeilen
tail(gpdat) # letzte Zeilen
```

- Einen Überblick mit Rstudio bekommen:



Indizierung eines data.frame

```
dat[1,1] # das Element oben links bekommen
```

```
## [1] 198431880
```

```
dat[2,] # nur die zweite Zeile sehen
```

```
##    z000001z z000002z        z000003z        z000005z
## 2 436122330   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
```

```
dat[,1] # sich nur die erste Spalte anzeigen lassen
```

```
## [1] 198431880 436122330 856844220 117346660 943433330 26558
```

Indizierung eines data.frame II

```
dat[1:2,] # getting the first two rows
```

```
##    z000001z z000002z         z000003z        z000005z
## 1 198431880   ZA5666 1-0-0 2017-06-20 10.4232/1.12749 Sehr
## 2 436122330   ZA5666 1-0-0 2017-06-20 10.4232/1.12749 Sehr
##         a11c020a       a11c021a       a11c022a
## 1 Sehr zufrieden Sehr zufrieden Stimme eher zu Stimme eher
## 2 Sehr zufrieden Sehr zufrieden Stimme eher zu Stimme eher
##         a11c024a                    a11c025a          a11c0
## 1 Stimme eher zu Eher höheren Lebensstandard Mehrmals im Mo
## 2 Stimme eher zu    Denselben Lebensstandard           Tägl
##                    a11c027a a11c028a          a11c029a a1
## 1 Mindestens einmal im Monat  Täglich           Täglich  T
## 2                   Täglich  Täglich Mehrmals die Woche  T
##              a11c031a          a11c032a          a11c033a
## 1 Mehrmals die Woche           Seltener          Seltener
```

Indizierung

- Das Dollarzeichen kann auch zur Adressierung einzelner Spalten verwendet werden.

```
head(datf$a11c019a)
```

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

```
datf$a11c019a[1:10]
```

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

Zugriff auf Spalten

- Wie bereits beschrieben, können Sie über Zahlen auf die Spalten zugreifen.

```
head(datf[,5])
```

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

```
head(datf[,"a11c019a"]) # dasselbe Ergebnis
```

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

Logische Operatoren

```
(a <- 1:7) # Beispieldaten - numerisch
```

```
## [1] 1 2 3 4 5 6 7
```

```
a>4
```

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

```
a>=4
```

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

```
a<3
```

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

Logische Operatoren II

```
(b <- letters[1:7]) # Beispieldaten - Strings
```

```
## [1] "a" "b" "c" "d" "e" "f" "g"
```

```
b=="e"
```

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

```
b %in% c("e","f")
```

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

GESIS Panel Variable - Estimated duration (bazq020a)

Wie lange haben Sie den Fragebogen ausgefüllt?

```
duration <- as.numeric(datf$bazq020a)
```

```
summary(duration)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -99.00   10.00   15.00   11.81   20.00 1440.00      23
```

Missing values

- Fehlende Werte sind in R als `NA` definiert
- Bei mathematische Funktionen gibt es in der Regel eine Möglichkeit, fehlende Werte auszuschließen.
- Bei `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` und `max()` gibt es das Argument `na.rm`.

```
mean(duration)
```

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

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

```
## [1] 11.81234
```

Die fehlenden Werte finden

```
is.na(head(duration))
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
which(is.na(duration))
```

```
##   [1]   30   63  103  182  184  258  415  424  441  527  588
## [15]  766  861  917  923  962  995 1026 1037 1062
```

```
table(is.na(duration))
```

```
##
## FALSE  TRUE
##  1199    23
```

Die fehlenden Werte rekodieren

```
summary(duration)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -99.00   10.00   15.00   11.81   20.00 1440.00      23
```

```
gpdat$bazq020a[gpdat$bazq020a==-99] <- NA
```

- **Quick-R zu fehlenden Werten**
- **Fehlende Werte rekodieren**

Eine einfache Funktion schreiben

```
tail(duration,n=10)
```

```
##  [1]  36  30 -33  10  20  15  17  15  10  15
```

```
transform_miss <- function(x){
  x[x==-99] <- NA
  return(x)
}
```

```
duration <- transform_miss(duration)
tail(duration,n=10)
```

```
##  [1]  36  30 -33  10  20  15  17  15  10  15
```

B1A Aufgabe - eine Funktion erweitern

- Erweitern Sie die Funktion so, dass sie auch dann ihren Zweck erfüllt, wenn die Value Labels ausgegeben werden (Item nonresponse, Missing by filter, etc.).

Der Befehl `complete.cases()`

Beispiel Datensatz

```
mydata <- data.frame(A=c(1,NA,9,6),B=c("A","B",1,NA))
```

Der Befehl `complete.cases()`

- gibt einen logischen Vektor zurück, der angibt, welche Fälle vollständig sind.

```
# Datenzeilen mit fehlenden Werten auflisten
mydata[complete.cases(mydata),]
```

```
##   A B
## 1 1 A
## 3 9 1
```

Verschiedene Arten von fehlenden Werten (NAs) spezifizieren

- Spezifiziere verschiedene Arten von Fehlern mit dem Paket `memisc`.
- Benutze dazu den Befehl `include.missings()`

```
library(memisc)
?include.missings
```

- Es ist auch möglich, Codebuch-Einträge mit `memisc` zu erstellen.

```
codebook(dat$a11c019a)
```

Datensatz indizieren

```
SEX <- gpdat$a11d054a
table(SEX)
```

```
## SEX
## Männlich Weiblich
##      596      626
```

```
gpdat[SEX=="Männlich",]
# same result:
gpdat[SEX!="Weiblich",]
```

Weitere wichtige Optionen

- Speichern des Ergebnisses in einem Objekt

```
subDat <- gpdat[duration>20,]
```

- mehrere Bedingungen können mit & verknüpft werden

```
gpdat[duration>18 & SEX=="Männlich",]
```

- das oder das Argument - eine der beiden Bedingungen muss erfüllt sein

```
gpdat[duration>18 | SEX=="Männlich",]
```

Die Verwendung von Sequenzen bei der Indizierung

```
library("readstata13")
datf <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta",
                   convert.factors = F)
```

```
datf[15:23,10:14]
```

```
##    a11c024a a11c025a a11c026a a11c027a a11c028a
## 15        1        2        5        5        1
## 16        3        2        4        1        1
## 17        1        1        5        2        4
## 18        2        2        3        3        1
## 19        2        1        4        5        1
## 20        1        2        3        2        1
## 21        2        2        5        5        1
## 22        1        3        4        3        2
## 23        2        2        2        3        1
```

Variablen Labels

```
library(foreign)
dat <- read.dta("../data/ZA5666_v1-0-0_Stata12.dta")
```

```
attributes(dat)
```

```
var.labels <- attr(dat,"var.labels")
```

- Das Gleiche gilt für das haven-Paket

```
library(haven)
dat2 <- read_dta("../data/ZA5666_v1-0-0_Stata14.dta")
var.labels2 <- attr(dat,"var.labels")
```

Umbenennen der Spaltennamen

- Mit dem Befehl `Colnames` erhält man die Spaltennamen

```
colnames(dat)
```

- Wir können die Spaltennamen umbenennen:

```
colnames(dat) <- var.labels
```

- Das gleiche gilt für die Zeilennamen

```
rownames(dat)
```

Private Internetnutzung (a11c034a)

Das Internet gewinnt eine immer größere Bedeutung in der Gesellschaft. Deshalb interessiert uns, ob Sie selbst zumindest gelegentlich das Internet für private Zwecke nutzen?

```
table(dat$a11c034a)
```

```
##
##                                  Item nonresponse
##                                                 0
##          Ja, nutzt Internet für private Zwecke
##                                              1044
## Nein, nutzt Internet nicht für private Zwecke
##                                               177
##                                       Weiß nicht
##                                                 1
```

Faktorstufen

```
str(dat$a11c034a)
```

```
##  Factor w/ 4 levels "Item nonresponse",..: 2 2 2 2 3 2 2 2
```

```
levels(dat$a11c034a)
```

```
## [1] "Item nonresponse"
## [2] "Ja, nutzt Internet für private Zwecke"
## [3] "Nein, nutzt Internet nicht für private Zwecke"
## [4] "Weiß nicht"
```

```
levels(dat$a11c034a)[2:4] <- c("yes","no","don`t know")
levels(dat$a11c034a)
```

```
## [1] "Item nonresponse" "yes"                    "no"
## [4] "don`t know"
```

Exkurs - Wie man Labels verwendet

Werkzeuge für das Arbeiten mit kategorischen Variablen (Faktoren)

```
library("forcats")
```

- `fct_collapse` - um Faktorstufen zu verdichten
- `fct_count` - um die Einträge in einem Faktor zu zählen
- `fct_drop` - Entferne unbenutzte Levels

Der Befehl `fct_count`

Freizeit Häufigkeit: Bücher lesen (a11c026a)

```
fct_count(f = dat$a11c026a)
```

```
## # A tibble: 8 x 2
##   f                                n
##   <fct>                        <int>
## 1 Item nonresponse                 0
## 2 Täglich                        239
## 3 Mehrmals die Woche             204
## 4 Mehrmals im Monat              154
## 5 Mindestens einmal im Monat      97
## 6 Seltener                       347
## 7 Nie                            181
## 8 Weiß nicht                       0
```

Der Befehl `fct_collapse`

```
a11c026a <- fct_collapse(.f = dat$a11c026a,
    Mehrmals=c("Mehrmals die Woche","Mehrmals im Monat"))
```

```
fct_count(a11c026a)
```

```
## # A tibble: 7 x 2
##   f                                n
##   <fct>                        <int>
## 1 Item nonresponse                 0
## 2 Täglich                        239
## 3 Mehrmals                       358
## 4 Mindestens einmal im Monat      97
## 5 Seltener                       347
## 6 Nie                            181
## 7 Weiß nicht                       0
```

recode Befehl im Paket car

```
library(car)
```

```
head(dat$a11c020a)
```

```
## [1] Sehr zufrieden Sehr zufrieden Eher zufrieden Sehr zufri
## [5] Sehr zufrieden Sehr zufrieden
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
head(recode(dat$a11c020a,"'Eher unzufrieden'='A';else='B'"))
```

```
## [1] B B B B B B
## Levels: A B
```

B1B Aufgabe - Value Labels neu kodieren

- Übersetze die Deutschen Werte Labels der Variablen `bbzc022a` ins Englische (Man kann dafür https://www.deepl.com/ verwenden).
- Kodiere die GESIS-Panel-Variable, die amn als englisches Value Label erhält.

Loops in R

- the command `for()` indicates the start of a loop
- in brackets, we have a index and the number of runs (in this case the loop runs from 1 until the number of columns in `dat`)
- in the curly brakets `{}` it is speciefied what happens for one iteration

```
for (i in 1:ncol(dat)){
  dat[,i] <- as.character(dat[,i])
}
```

Loops - keeping results

- we can save the results in a container
- that can be a vector or a list

```
erg1 <- vector()
erg2 <- list()

for (i in 1:ncol(dat)){
  tab <- table(dat[,i])
  erg[i] <- length(tab)
  erg[[i]] <- tab
  cat(i, "\n")
}
```

B1C Exercise - Scripting loops

Exercise from www.r-exercises.com

- The `repeat{}` loop processes a block of code until the condition specified by the `break` statement, (that is mandatory within the `repeat{}` loop), is met.
- The structure of a `repeat{}` loop is:

```
repeat {
commands
if(condition) {
break
}
}
```

- For this exercise, write a `repeat{}` loop that prints all the even numbers from 2 – 10, via incrementing the variable, `i <- 0`.

B1D Exercise - A while loop

Exercise from www.r-exercises.com

`while()` loop will repeat a group of commands until the condition ceases to apply. The structure of a `while()` loop is:

```
while(condition) {
commands
}
```

- With, `i <- 1`, write a `while()` loop that prints the odd numbers from 1 through 7.

The apply family

```
(ApplyDat <- cbind(1:4,runif(4),rnorm(4))) # Example data set
```

```
##      [,1]      [,2]       [,3]
## [1,]    1 0.4518330 -0.6437858
## [2,]    2 0.3713149 -0.5650281
## [3,]    3 0.1791734 -0.1634185
## [4,]    4 0.9031327  0.6804515
```

```
apply(ApplyDat,1,mean)
```

```
## [1] 0.2693491 0.6020956 1.0052516 1.8611947
```

```
apply(ApplyDat,2,mean)
```

```
## [1]  2.5000000  0.4763635 -0.1729452
```

The command apply()

```
apply(ApplyDat,1,var)
```

```
## [1] 0.7004832 1.6847871 3.0136081 3.4432627
```

```
apply(ApplyDat,1,sd)
```

```
## [1] 0.8369488 1.2979935 1.7359747 1.8556031
```

```
apply(X = ApplyDat,MARGIN = 1,FUN = range)
```

```
##            [,1]       [,2]       [,3]      [,4]
## [1,] -0.6437858 -0.5650281 -0.1634185 0.6804515
## [2,]  1.0000000  2.0000000  3.0000000 4.0000000
```

The arguments of the command `apply()`

- If `MARGIN=1` the function `mean` is applied for rows,
- If `MARGIN=2` the function `mean` is applied for columns,
- Instead of `mean` you could also use `var`, `sd` or `length`.

The command tapply()

```
ApplyDat <- data.frame(Income=rnorm(5,1400,200),
                       Sex=sample(c(1,2),5,replace=T))
```

Example command tapply()

```
tapply(ApplyDat$Income,
       ApplyDat$Sex,function(x)x)
```

```
## $`1`
## [1] 1412.540 1267.989 1602.736
##
## $`2`
## [1] 1339.458 1622.268
```

- Other commands can also be used. . . . also self-scripted commands

B1E Exercise - using the `tapply()` command

- Find out which variable contains information about age
- Calculate the average duration (variable `bfzq020a`) by age group

The reshape package

Example dataset

```
(mydata <- data.frame(id=rep(1:2,each=2), # sample dataset
                      time=rep(c(1,2),2),
                      x1 = c(5,3,6,2),
                      x2 = c(6,5,1,4)))
```

```
##   id time x1 x2
## 1  1    1  5  6
## 2  1    2  3  5
## 3  2    1  6  1
## 4  2    2  2  4
```

Example of command `melt`

```
library(reshape)
melt(mydata, id=c("id","time")) #
```

```
##    id time variable value
## 1   1    1       x1     5
## 2   1    2       x1     3
## 3   2    1       x1     6
## 4   2    2       x1     2
## 5   1    1       x2     6
## 6   1    2       x2     5
## 7   2    1       x2     1
## 8   2    2       x2     4
```

The package tibble

Difference between tibble and data.frame

- There are three key differences between tibbles and data frames: printing, subsetting, and recycling rules.

```
library(tibble)
(gpanel1 <- as_tibble(dat))
```

```
## # A tibble: 1,222 x 1,192
##    z000001z z000002z z000003z z000005z a11c019a a11c020a a1
##  *    <int> <chr>    <chr>    <chr>    <fct>    <fct>    <f
##  1   1.98e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Se
##  2   4.36e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Se
##  3   8.57e8 ZA5666   1-0-0 2~ 10.4232~ Eher zu~ Eher zu~ Eh
##  4   1.17e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Se
##  5   9.43e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Eh
##  6   2.66e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Se
```

Edgar Anderson's Iris dataset

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

- petal length and width
- sepal length and width
- **Wikipedia Article for the IRIS dataset**

Relationship between continuous variables

```
# Pearson correlation coefficient
cor(iris$Sepal.Length,iris$Petal.Length)
```

```
## [1] 0.8717538
```

- Correlation between petal length and petal length 0.87
- The Pearson's correlation coefficient is the default method in `cor()`.

Various correlation coefficients

```
# Pearson correlation coefficient
cor(iris[,1:4])
```

```
##                Sepal.Length Sepal.Width Petal.Length Petal.Wi
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000
```

```
# Kendall's tau (rank correlation)
cor(iris[,1:4], method = "kendall")
```

```
##                Sepal.Length Sepal.Width Petal.Length Petal.Wi
## Sepal.Length   1.00000000 -0.07699679    0.7185159   0.6553
## Sepal.Width   -0.07699679  1.00000000   -0.1859944  -0.1571
## Petal.Length   0.71851593 -0.18599442    1.0000000   0.8068
```

Relationship between categorial variables

- `chisq.test()` tests whether two categorial features are stochastically independent.
- The test is performed against the null hypothesis of equal distribution

B1F Exercise - make a interactive table

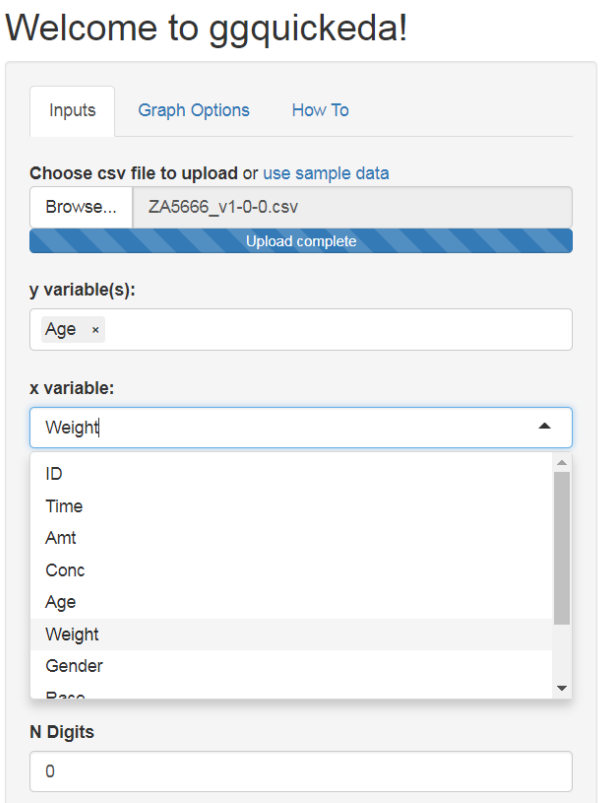
- Download the dataset `dat_cf2.RData` from ILIAS
- Import dataset into R
- Create an interactive table using the following commands

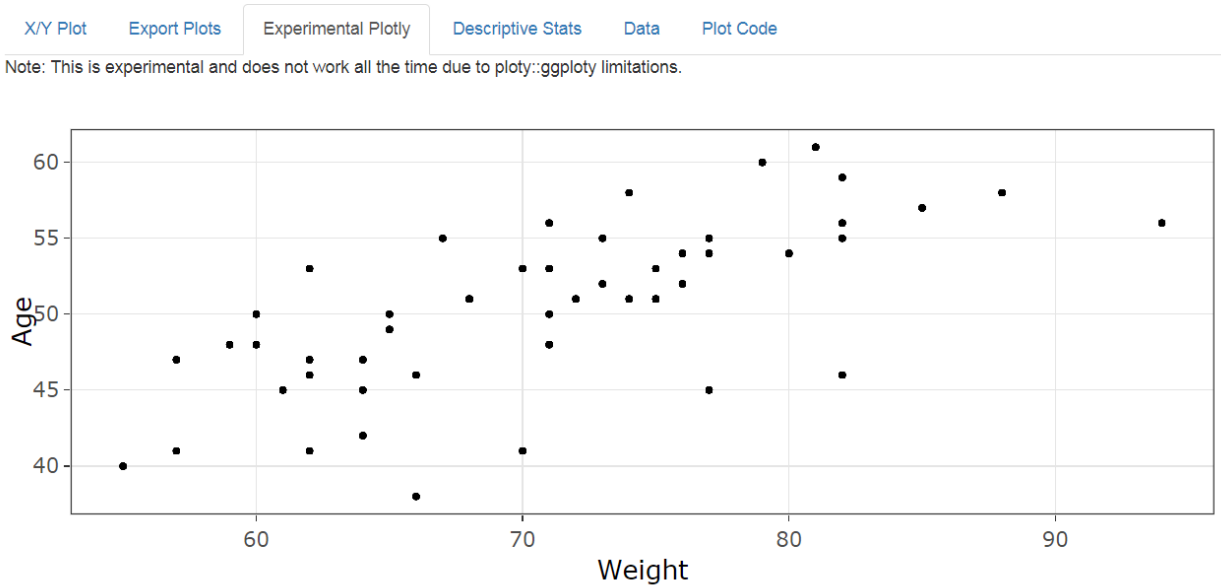
```
library(DT)
DT::datatable(dat_cf2)
```

- See which additional arguments of the function `datatable` are valuable

Shiny App for quick explorative data analysis

https://pharmacometrics.shinyapps.io/ggplotwithyourdata/





Further Links

- **Tidy data** - the package `tidyr`
- Homepage for **the `tidyverse` collection**
- **Data wrangling with R and RStudio**
- Hadley Wickham - **Tidy Data**
- Hadley Wickham - **Advanced R**
- Colin Gillespie and Robin Lovelace **Efficient R programming**