A5 - Data Processing

Jan-Philipp Kolb

21 Februar, 2020

FIRST THINGS TO DO

Don't try to kiss your data on the first date; rather, you just want to get to know the data:

1. Import the data
2. Review the codebook
3. **Learn about the data**
4. Quick (visual) understanding of the data

LEARN ABOUT THE DATA

So what are the first things we want to know about our data?

- dimensions
- data types (i.e. character, integer, factor, etc.)
- missing values
- summary statistics

What are some functions to extract this information?

LEARN ABOUT THE DATA

- So what are the first things we want to know about our data?
- dimensions: `dim()`, `ncol()`, `nrow()`, `names()`
- data types: `str()`, `class()`, `is.`, `as.`
- missing values: `is.na()`, `sum(is.na())`, `colSums(is.na())`
- summary statistics: `summary()`, `quantile()`, `var()`, `sd()`, `table()`

Data Frames

Example data:

```
ames_data <- AmesHousing::make_ames()
```

```
typeof(ames_data)
```

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

```
head(names(ames_data))
```

```
## [1] "Lot_Area"     "Street"        "Alley"         "Lot_Shape"
```

Transfer to data.frame

- Transfer data to a `data.frame`:

```
ames_df <- data.frame(ames_data)
```

Number of rows/columns

- Find out the number of rows/columns

```
nrow(ames_df) # rows
```

```
## [1] 2930
```

```
ncol(ames_df) # columns
```

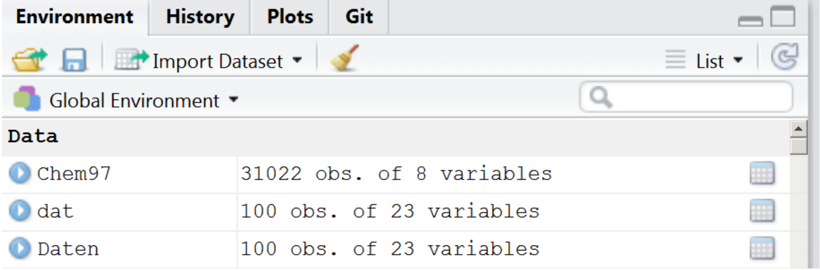
```
## [1] 81
```

View the data

- See the some lines:

```
head(ames_df) # first lines
tail(ames_df) # last lines
```

- Overview with Rstudio:



What is the difference?

vector

```
0.70 0.86 0.95 0.25 0.52 0.37 0.27 0.80 0.60 0.26
```

matrix

```
     [,1] [,2] [,3] [,4]
[1,] 0.70 0.37 0.70 0.37
[2,] 0.86 0.27 0.86 0.27
[3,] 0.95 0.80 0.95 0.80
[4,] 0.25 0.60 0.25 0.60
[5,] 0.52 0.26 0.52 0.26
```

data frame

```
   Sepal.Length Sepal.Width Petal.Width Species
1           5.1         3.5         0.2  setosa
2           4.9         3.0         0.2  setosa
3           4.7         3.2         0.2  setosa
4           4.6         3.1         0.2  setosa
5           5.0         3.6         0.2  setosa
6           5.4         3.9         0.4  setosa
7           4.6         3.4         0.3  setosa
8           5.0         3.4         0.2  setosa
9           4.4         2.9         0.2  setosa
10          4.9         3.1         0.1  setosa
```

list

```
$item1
[1] 1 2 3

$item2
[1] "a" "b" "c" "d" "e"

$item3
[1]  TRUE FALSE  TRUE  TRUE

$item4
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Indexing

The principle of indexing

```
vector[element]
data.frame[rows, columns]
matrix[rows, columns]
list[component]
list[[component]]
list$component
```

Accessing Columns

- The dollar sign can also be used to address individual columns

```
head(ames_df$Lot_Area)
```

```
## [1] 31770 11622 14267 11160 13830  9978
```

```
ames_df$Lot_Area[1:10]
```

```
##  [1] 31770 11622 14267 11160 13830  9978  4920  5005  5389  7
```

- As already described, you can use numbers to access the columns

```
head(ames_df[,5])
head(ames_df[,"Street"]) # the same result
```

Exercise: Vectors and Indexing

Assume that we have registered the height and weight for four people: Heights in cm are 180, 165, 160, 193; weights in kg are 87, 58, 65, 100. Make two vectors, height and weight, with the data. The bodymass index (BMI) is defined as

$$\frac{\text{weight in kg}}{(\text{height in m})^2}$$

Make a vector with the BMI values for the four people, and a vector with the natural logarithm to the BMI values. Finally make a vector with the weights for those people who have a BMI larger than 25.

Subsetting dataset

```r
Street <- ames_df$Street
table(Street)
```

```
## Street
## Grvl Pave
##   12 2918
```

```r
ames_df[Street=="Grvl",]
# same result:
ames_df[Street!="Pave",]
```

Get stata attributes

```r
att_dat <- attributes(ames_df)
head(names(att_dat))
```

```
## [1] "names"     "class"     "row.names"
```

Example: the variable names

```r
head(att_dat$names)
```

```
## [1] "MS_SubClass"   "MS_Zoning"     "Lot_Frontage"  "Lot_Area"
## [6] "Alley"
```

THE AIRQUALITY DATA

```r
data(airquality)
Ozone <- airquality$Ozone
```

airquality {datasets} R Documentation

New York Air Quality Measurements

Description

Daily air quality measurements in New York, May to September 1973.

Usage

```
airquality
```

Format

A data frame with 154 observations on 6 variables.

[,1]	Ozone	numeric Ozone (ppb)
[,2]	Solar.R	numeric Solar R (lang)
[,3]	Wind	numeric Wind (mph)
[,4]	Temp	numeric Temperature (degrees F)
[,5]	Month	numeric Month (1--12)
[,6]	Day	numeric Day of month (1--31)

Details

Daily readings of the following air quality values for May 1, 1973 (a Tuesday) to September 30, 1973.

Other important options

- save result to an object

```
subDat <- airquality[Ozone>30,]
```

- multiple conditions can be linked with &

```
airquality[Ozone>18 & airquality$Month==5,]
```

- the or argument - one of the two conditions must be fullfilled

```
airquality[Ozone>18 | airquality$Month==5,]
```

Missing values

- Missing values are defined as `NA` in R
- Math functions usually have a way to exclude missing values in their calculations.
- `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` and `max()` all take the `na.rm` argument.

```
mean(Ozone)
```

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

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

```
## [1] 42.12931
```

Find the missing values:

```
head(is.na(Ozone))
```

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

```
which(is.na(Ozone))
```

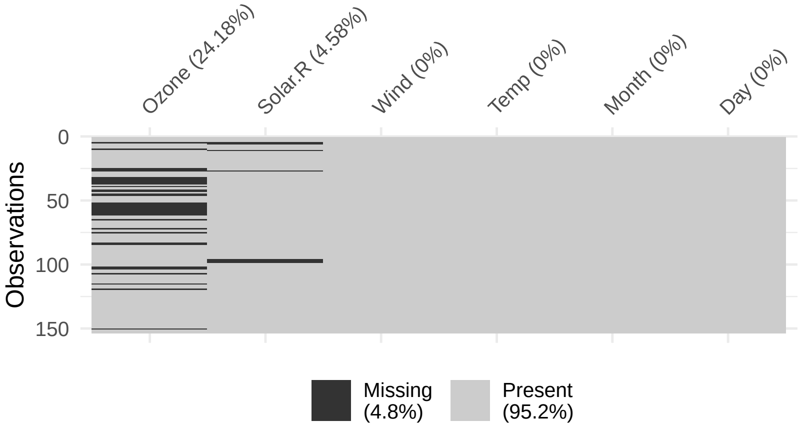
```
##  [1]   5  10  25  26  27  32  33  34  35  36  37  39  42  43
## [20]  55  56  57  58  59  60  61  65  72  75  83  84 102 103
```

```
table(is.na(Ozone))
```

```
## 
## FALSE  TRUE 
##   116    37
```

Missing Data Visualisations

```
# Data Structures, Summaries, and Visualisations
# for Missing Data
library(naniar)
vis_miss(airquality)
```



The command complete.cases()

The command complete.cases()

- returns a logical vector indicating which cases are complete.

```
nrow(gpdat)
# list rows of data without missing values
gpdat_comp <- gpdat[complete.cases(gpdat),]
nrow(gpdat_comp)
```

An shorthand alternative

- An shorthand alternative is to simply use `na.omit()` to omit all rows containing missing values.

```
gpdat_comp <- na.omit(gpdat)
nrow(gpdat_comp)
```

```
## NULL
```

Very simple imputation

```
# data frame with missing data
df <- data.frame(col1 = c(1:3, NA),
                 col2 = c("this", NA,"is", "text"),
                 col3 = c(TRUE, FALSE, TRUE, TRUE),
                 col4 = c(2.5, 4.2, 3.2, NA),
                 stringsAsFactors = FALSE)

(df$col4[is.na(df$col4)] <- mean(df$col4, na.rm = TRUE))
```

```
## [1] 3.3
```

NAs per column

- For data frames, a convenient shortcut to compute the total missing values in each column is to use `colSums()`:

```
colSums(is.na(df))
```

```
## col1 col2 col3 col4
##    1    1    0    0
```

CRAN Task View: Missing Data

Maintainer:	Julie Josse, Nicholas Tierney and Nathalie Vialaneix (r-miss-tastic team)
Contact:	r-miss-tastic at clementine.wf
Version:	2019-07-02
URL:	https://CRAN.R-project.org/view=MissingData

Missing data are very frequently found in datasets. Base R provides a few options to handle them using computations that involve only observed data (`na.rm = TRUE` in functions `mean`, `var`, ... or `use = complete.obs|na.or.complete|pairwise.complete.obs` in functions `cov`, `cor`, ...). The base package stats also contains the generic function `na.action` that extracts information of the `NA` action used to create an object.

These basic options are complemented by many packages on CRAN, which we structure into main topics:

- Exploration of missing data
- Likelihood based approaches
- Single imputation
- Multiple imputation
- Weighting methods
- Specific types of data
- Specific application fields

Exercise: Missing values

1. How many missing values are in the built-in data set airquality?
2. Which variables are the missing values concentrated in?
3. How would you impute the mean or median for these values?
4. How would you omit all rows containing missing values?

Codebook entries

- It is also possible to create stata like codebook entries with `memisc`.

```
codebook(gpdat$a11c019a)
```

Rename the column names

- With the command `colnames` you get the column names

```
colnames(airquality)
```

- We can rename the column names:

```
colnames(airquality)[1] <- "var1"
```

- The same applies to the row names

```
rownames(airquality)
```

Private internet usage (a11c034a)

The Internet is constantly growing in significance for society. Therefore, we are interested whether you yourself use the Internet at least occasionally for private purposes?

```
table(gpdat$a11c034a)
```

```
## < table of extent 0 >
```

```
ind <- which(names(att_dat$label.table)=="a11c034a")
att_dat$label.table[ind]
```

```
## NULL
```

```
let <- as.factor(c(LETTERS[1:5],LETTERS[1:3]))
str(let)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 2 3 4 5 1 2 3
```

```
levels(let)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
levels(let)[2:4] <- c("first","second","third")
levels(let)
```

```
## [1] "A"      "first"  "second" "third"  "E"
```

Factor

A factor- type vector contains a set of numeric codes with character-valued levels.

Example

- a family of two girls (1) and four boys (0),

```
(kids <- factor(c(1,0,1,0,0,0),levels= c(0,1),
                labels= c("boy","girl")))
```

```
## [1] girl boy  girl boy  boy  boy
## Levels: boy girl
```

```
class(kids)
```

```
## [1] "factor"
```

```
mode(kids)
```

```
## [1] "numeric"
```

A factor

```
kids + 1
```

```
## Warning in Ops.factor(kids, 1): '+' not meaningful for factor
## [1] NA NA NA NA NA NA
```

```
as.numeric(kids)
```

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

```
1 + as.numeric(kids)
```

```
## [1] 3 2 3 2 2 2
```

Excursus - how to use labels

Tools for Working with Categorical Variables (Factors)

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

- `fct_collapse` - to summarize factor levels
- `fct_count` - to count the entries in a factor
- `fct_drop` - Take out unused levels

The command fct_count

Leisure time frequency: Read books (a11c026a)

```
fct_count(f = let)
```

```
## # A tibble: 5 x 2
##   f          n
##   <fct>  <int>
## 1 A          2
## 2 first      2
## 3 second     2
## 4 third      1
## 5 E          1
```

The command fct_collapse

```r
letb <- fct_collapse(.f = let,
    important=c("first","second"))
```

```r
fct_count(letb)
```

```
## # A tibble: 4 x 2
##   f             n
##   <fct>     <int>
## 1 A             2
## 2 important     4
## 3 third         1
## 4 E             1
```

RECODE COMMAND IN PACKAGE CAR

```r
library(car)
```

```r
head(let)
```

```
## [1] A      first  second third  E      A
## Levels: A first second third E
```

```r
head(recode(let,"'first'='A';else='B'"))
```

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

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

```
##      [,1]      [,2]       [,3]
## [1,]    1 0.7908896 -0.4159306
## [2,]    2 0.9998150  0.2702666
## [3,]    3 0.7794650 -0.4790617
## [4,]    4 0.2667414  0.7045150
```

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

```
## [1] 0.4583197 1.0900272 1.1001344 1.6570855
```

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

```
## [1] 2.50000000 0.70922774 0.01994732
```

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

```
## [1] 0.5841669 0.7540981 3.1030893 4.1648478
```

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

```
## [1] 0.7643081 0.8683882 1.7615588 2.0407959
```

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

```
##            [,1]      [,2]       [,3]      [,4]
## [1,] -0.4159306 0.2702666 -0.4790617 0.2667414
## [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] 1558.331 1101.663 1593.510 1344.209
##
## $`2`
## [1] 1540.743
```

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

Exercise: using the `tapply()` command

- Calculate the average ozone value by month using the airquality dataset and the tapply command.

Example dataset

```r
(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

```r
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
```

```r
load("../data/merge_example_data.RData")
```

```
> authorN
  nationality deceased     name
1          US      yes    Tukey
2   Australia       no Venables
3          US       no  Tierney
4          UK       no   Ripley
5   Australia       no   McNeil
```

```
> books
      name                         title     other.author
1    Tukey     Exploratory Data Analysis             <NA>
2 Venables Modern Applied Statistics ...           Ripley
3  Tierney                     LISP-STAT             <NA>
4   Ripley            Spatial Statistics             <NA>
5   Ripley         Stochastic Simulation             <NA>
6   McNeil     Interactive Data Analysis             <NA>
7   R Core          An Introduction to R Venables & Smith
```

MERGE DATA

- these two give the same results:

```
(m0 <- merge(authorN, books))
```

```
##       name nationality deceased                        title
## 1   McNeil   Australia       no    Interactive Data Analysis
## 2   Ripley          UK       no           Spatial Statistics
## 3   Ripley          UK       no        Stochastic Simulation
## 4  Tierney          US       no                    LISP-STAT
## 5    Tukey          US      yes    Exploratory Data Analysis
## 6 Venables   Australia       no Modern Applied Statistics ...
```

```
(m1 <- merge(authors, books, by.x = "surname",
             by.y = "name"))
```

```
##    surname nationality deceased                        title
## 1   McNeil   Australia       no    Interactive Data Analysis
## 2   Ripley          UK       no           Spatial Statistics
## 3   Ripley          UK       no        Stochastic Simulation
## 4  Tierney          US       no                    LISP-STAT
## 5    Tukey          US      yes    Exploratory Data Analysis
```

Links and resources

- **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**
- **Quick-R about missing values**
- **Recode missing values**