Introduction to R

Data Processing

Jan-Philipp Kolb

10 März, 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

First things we want to know:

Dimensions

Data types (i.e. character, integer, factor, etc.)

Missing values

Summary statistics

What are some functions to extract this information?

What we want to learn

- 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()`

Example data - The Ames Housing data

Also available on kaggle:

tial_High_Density	80	11622	Pave	No_Alley_Access	Regular
tial_Low_Density	81	14267	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	93	[illegible]	Pave	No_Alley_Access	Regular
tial_Low_Density	74	13830	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	[illegible]	9978	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	41	4920	Pave	No_Alley_Access	Regular
tial_Low_					Irregular
tial_Low_					Irregular
tial_Low_					
tial_Low_Density	75	10000	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	0	7980	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	63	8402	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	85	10176	Pave	No_Alley_Access	Regular
tial_Low_Density	0	6820	Pave	No_Alley_Access	Slightly_Irregular
tial_Low_Density	47	53504	Pave	No_Alley_Access	Moderately_Irregu
tial_Low_Density	152	12134	Pave	No_Alley_Access	Slightly_Irregular

Ames housing data

```
install.packages("AmesHousing")
```

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

```
names(ames_df)
```

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

Number of rows/columns

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

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

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

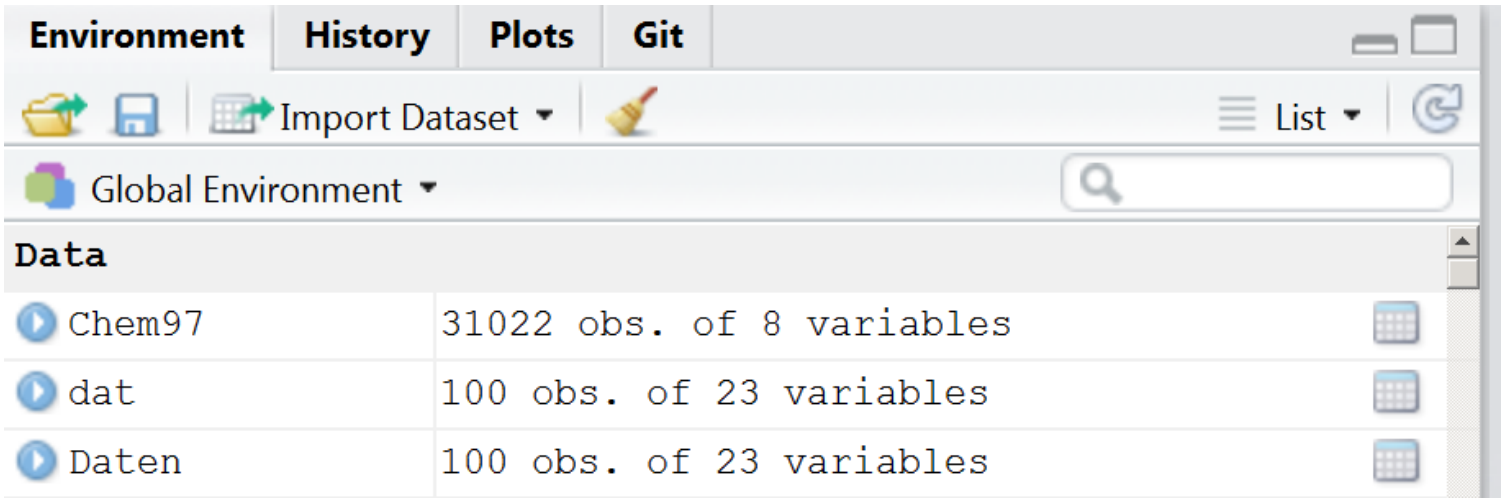
```
## [1] 81
```

View the data

See some lines:

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

Overview with Rstudio:



Indexing

Accessing Columns with the dollar sign

- 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[1:8]
```

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

Accessing Columns with the number or the name

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

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

Subsetting dataset

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

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

Logical operations and indexing

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

Logical operations in indexing

Selecting a subset due to criteria

Select only homes with a area size bigger than 9000

```
ames_df[ames_df$Lot_Area>9000,]
```

Lot_Area	Street	Alley
31770	Pave	No_Alley_Access
11622	Pave	No_Alley_Access
14267	Pave	No_Alley_Access
11160	Pave	No_Alley_Access
13830	Pave	No_Alley_Access
9978	Pave	No_Alley_Access
10000	Pave	No_Alley_Access
10176	Pave	No_Alley_Access

Exercise: Compute the price per area

- Compute the Price (`Sale_Price`) per Area (`Lot_Area` - Lot size in square feet)
- Find the maximum of Price per Area

<!--

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.

-->

The `airquality` data

```
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)

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  45
## [16]  46  52  53  54  55  56  57  58  59  60  61  65  72  75  83
## [31]  84 102 103 107 115 119 150
```

```
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()`

- returns a logical vector indicating which cases are complete.

```
nrow(airquality)
```

```
## [1] 153
```

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

```
## [1] 111
```

A shorthand alternative

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

```
airq_comp <- na.omit(airquality)
nrow(airq_comp)
```

```
## [1] 111
```

Very simple imputation

```
airq <- airquality
airq$Ozone[is.na(airq$Ozone)] <-mean(airq$Ozone,na.rm = T)
```

Comparing mean and variance

```
mean(airquality$Ozone, na.rm = T) ; mean(airq$Ozone)
```

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

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

```
var(airquality$Ozone, na.rm = T)
```

```
## [1] 1088.201
```

```
var(airq$Ozone)
```

```
## [1] 823.3096
```

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(airquality))
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       0       0       0       0
```

```
colSums(is.na(airq))
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##       0       7       0       0       0       0
```

CRAN Task View: Missing Data

`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?

Rename the column names

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

```
colnames(airquality)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

- We can rename the column names:

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

```
## [1] "var1"    "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

- The same applies to the row names

```
rownames(airquality)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"
##  [11] "11"  "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"
##  [21] "21"  "22"  "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"
##  [31] "31"  "32"  "33"  "34"  "35"  "36"  "37"  "38"  "39"  "40"
##  [41] "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"  "49"  "50"
```

Excursus: loops

create a dublette

```
airq2 <- airquality
```

The `for` loop

```
for (i in 1:ncol(airq)){
  airq2[,i] <- as.character(airq2[,i])
}
```

The result of the for loop

```
str(airquality)
```

```
## 'data.frame':    153 obs.
##  $ var1   : int  41 36 12
##  $ Solar.R: int  190 118 
##  $ Wind   : num  7.4 8 12
##  $ Temp   : int  67 72 74
##  $ Month  : int  5 5 5 5 
##  $ Day    : int  1 2 3 4 
```

```
str(airq2)
```

```
## 'data.frame':    153 obs. of  6 varia
##  $ var1   : chr  "41" "36" "12" "18"
##  $ Solar.R: chr  "190" "118" "149" "3
##  $ Wind   : chr  "7.4" "8" "12.6" "11
##  $ Temp   : chr  "67" "72" "74" "62"
##  $ Month  : chr  "5" "5" "5" "5" ...
##  $ Day    : chr  "1" "2" "3" "4" ...
```

The `apply` family

```
apply(airq,2,mean)
```

```
##     Ozone   Solar.R      Wind      Temp     Month       Day
## 42.129310        NA  9.957516 77.882353  6.993464 15.803922
```

```
apply(airq,2,mean,na.rm=T)
```

```
##      Ozone    Solar.R       Wind       Temp      Month        Day
##  42.129310 185.931507   9.957516  77.882353   6.993464  15.803922
```

```
# the following is possible but doesn`t make sense
# for this case:
apply(airq,1,mean)
```

The command `apply()`

```
apply(airq,2,var)
```

```
##      Ozone    Solar.R       Wind       Temp      Month        Day
## 823.309608         NA  12.411539  89.591331   2.006536  78.579721
```

```
apply(airq,2,sd)
```

```
##     Ozone   Solar.R      Wind      Temp     Month       Day
## 28.693372        NA  3.523001  9.465270  1.416522  8.864520
```

```
apply(X = airq,MARGIN = 2,FUN = range)
```

```
##      Ozone Solar.R Wind Temp Month Day
## [1,]     1      NA  1.7   56     5   1
## [2,]   168      NA 20.7   97     9  31
```

The arguments of `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`.

Example command `tapply()`

```
tapply(airq$Wind, airq$Month,mean)
```

```
##         5         6         7         8         9
## 11.622581 10.266667  8.941935  8.793548 10.180000
```

- 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.

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**