University of Applied Sciences and Arts Northwestern Switzerland
School of Business

member of
swissuniversities

AACSB
ACCREDITED

Data Science - Exercises

Holger Wache

Exercise A

The (build-in) Data Set “mtcars”

```
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
…
```

First Insights into Data Set “mtcars”

```
> str(mtcars)
'data.frame':    32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Compute the mean, median, and mode of column “wt”

```
> mean(mtcars$wt)
[1] 3.21725
```

```
> median(mtcars$wt)
[1] 3.325
```

```
> mode(mtcars$wt)
[1] "numeric"
```

Not correct; mode is another function

```
> y <- table(mtcars$wt)
> y

1.513 1.615 1.835 1.935  2.14   2.2  2.32 2.465  2.62  2.77  2.78 2.875
    1     1     1     1     1     1     1     1     1     1     1     1
3.15  3.17  3.19 3.215 3.435  3.44  3.46  3.52  3.57  3.73  3.78  3.84
   1     1     1     1     1     3     1     1     2     1     1     1
3.845  4.07  5.25 5.345 5.424
    1     1     1     1     1
> names(y)[which(y==max(y))]
[1] "3.44"
```

That would be the correct (statistical) mode

.. Or everything in one command for “mtcars”

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930
       wt             qsec             vs              am
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000
      gear            carb
 Min.   :3.000   Min.   :1.000
 1st Qu.:3.000   1st Qu.:2.000
 Median :4.000   Median :2.000
 Mean   :3.688   Mean   :2.812
 3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :5.000   Max.   :8.000
```

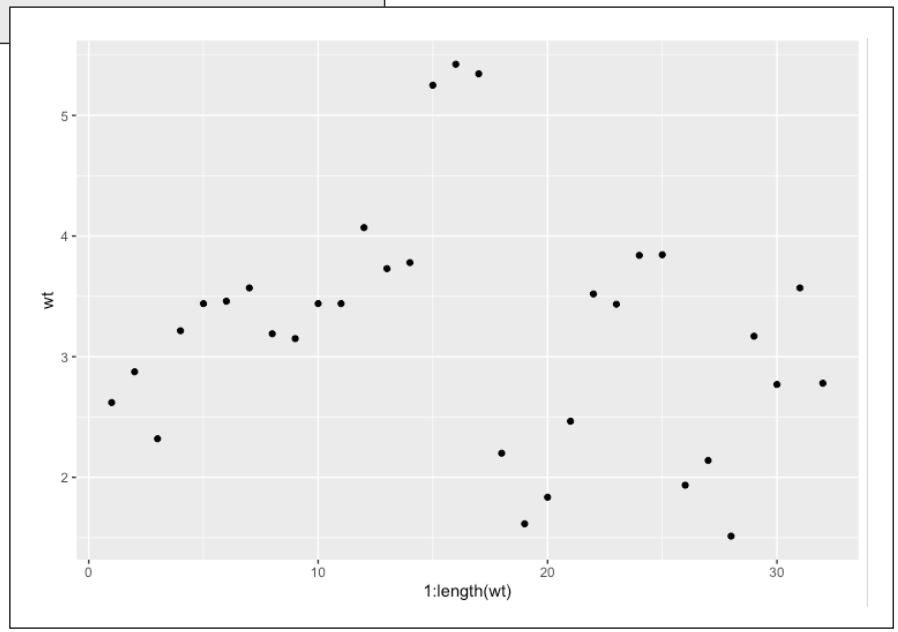
Draw (plot) the column “wt”

```
> ggplot(mtcars) + geom_point(mapping = aes(x = 1:length(wt), y = wt))
```

Advanced plotting of data mtcars

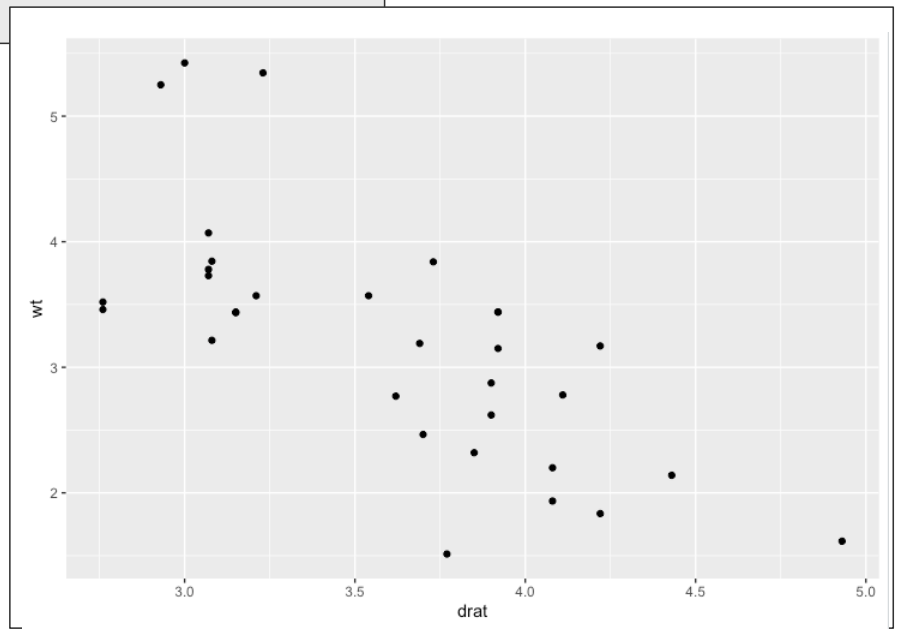
Adding a layer

Draw points a the x and y coordinates



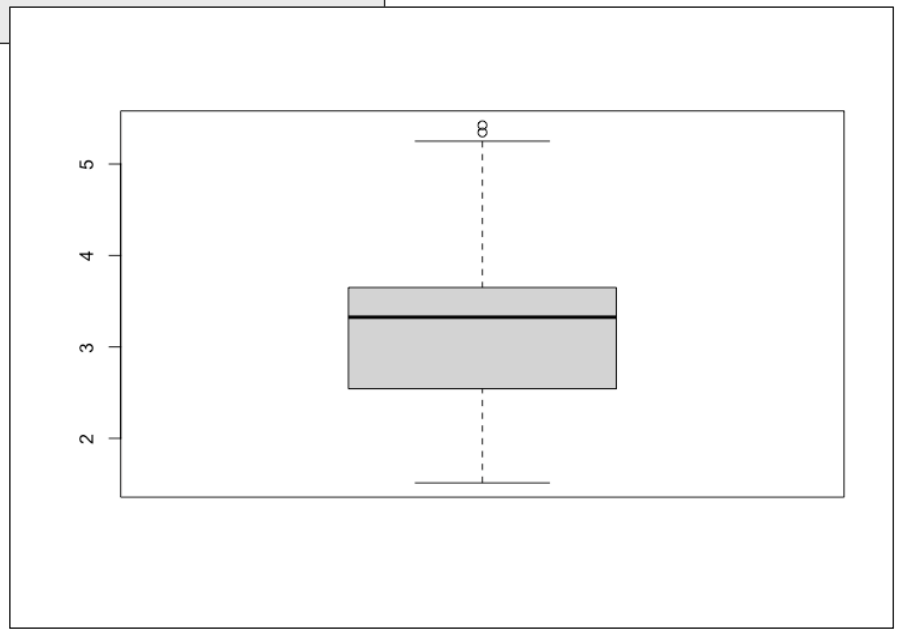
Draw (plot) the column “wt” against column “drat”

```
> ggplot(mtcars) + geom_point(mapping = aes(x = drat, y = wt))
```



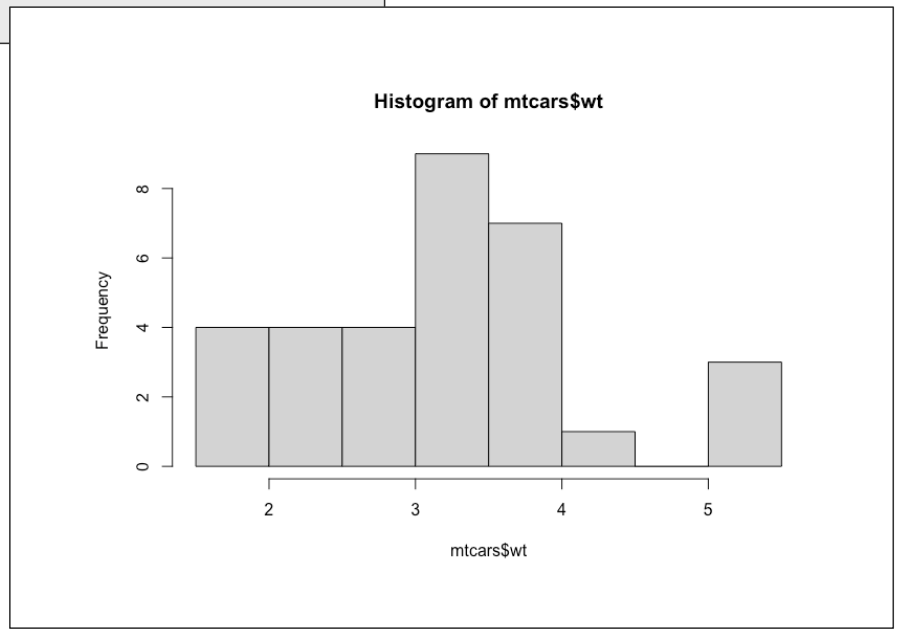
Boxplot of column “wt”

```
> boxplot(mtcars$wt)
```



Histogram of column “wt” and help for hist()

```
> hist(mtcars$wt)
```




```
> ?hist
```

hist {graphics} R Documentation

Histograms

Description

The generic function `hist` computes a histogram of the given data values. If `plot = TRUE`, the resulting object of class `"histogram"` is plotted by `plot.histogram`, before it is returned.

Usage

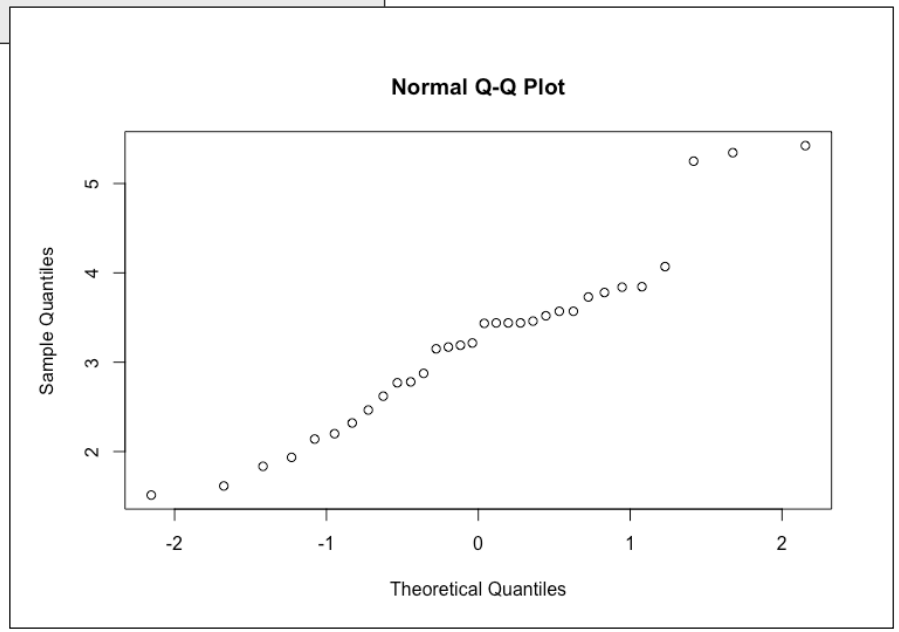
```
hist(x, ...)

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = "lightgray", border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, warn.unused = TRUE, ...)
```

Arguments

x	a vector of values for which the histogram is desired.

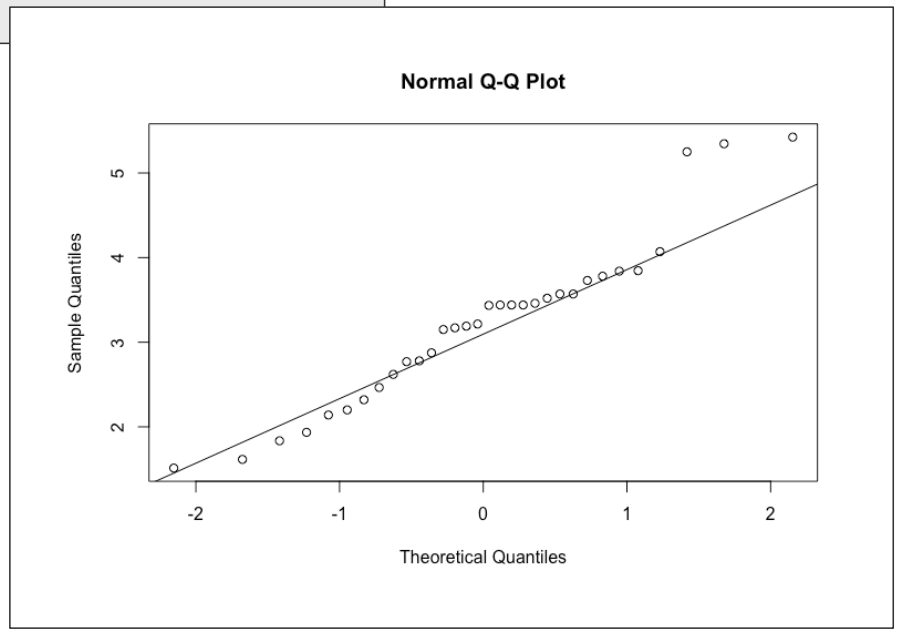
Q-Q Plot of column “wt”

```
> qqnorm(mtcars$wt)
```



Q-Q Plot with assumed line of column “wt”

```
> qqline(mtcars$wt)
```

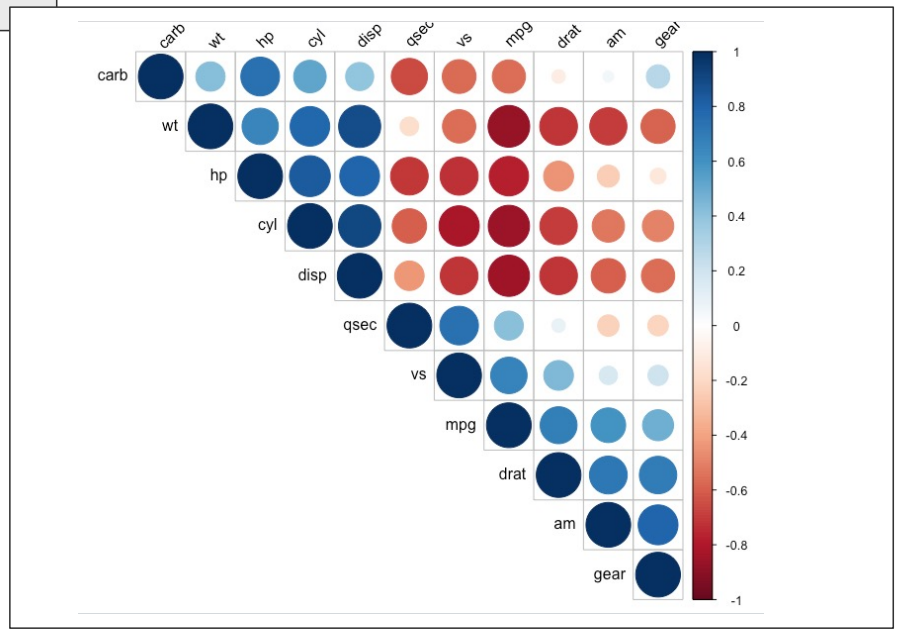


Correlation of columns of data set “mtcars”

```
> cor(mtcars)
…
> round(cor(mtcars),2)
       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

Plot of the correlation of columns of data set “mtcars”

```
> install.packages("corrplot")
> library(corrplot)
> corrplot(cor(mtcars), type = "upper", order = "hclust",
tl.col = "black", tl.srt = 45)
```



University of Applied Sciences and Arts Northwestern Switzerland
School of Business

member of
swissuniversities

AACSB
ACCREDITED

Exercise B

Data Cleaning

Exercise B

Missing Values

Prerequisites

First, install additional packages

1. an additional package containing flights and
2. an additional package containing nice functions

```
install.packages("nycflights13")
install.packages("tidyverse")
```

After the installation these packages need to be "activated"

```
library(nycflights13)
library(tidyverse)
```

See the data “flights”

```
> flights
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight
tailnum
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1      517            515         2      830            819        11 UA        1545 N14228
 2  2013     1     1      533            529         4      850            830        20 UA        1714 N24211
 3  2013     1     1      542            540         2      923            850        33 AA        1141 N619AA
 4  2013     1     1      544            545        -1     1004           1022       -18 B6          725 N804JB
 5  2013     1     1      554            600        -6      812            837       -25 DL          461 N668DN
 6  2013     1     1      554            558        -4      740            728        12 UA        1696 N39463
 7  2013     1     1      555            600        -5      913            854        19 B6          507 N516JB
 8  2013     1     1      557            600        -3      709            723       -14 EV        5708 N829AS
 9  2013     1     1      557            600        -3      838            846        -8 B6          79 N593JB
10  2013     1     1      558            600        -2      753            745         8 AA          301 N3ALAA
# … with 336,766 more rows, and 7 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>

> count(flights)
# A tibble: 1 x 1
      n
  <int>
1 336776
```

Filtering the data

The function “filter()” allows to select some data lines (sets) with respect to some conditions.
For example all flights in the first month:

```
> filter(flights, month == 1)
...
```

Several conditions can also be combined:

```
> filter(flights, month == 1 & day == 1)
...
```

Identifying Missing Values: Deleting the data row

You can also simply remove the rows with missing values with the function “filter()” . Just negate the “is,na()” test, i.e. select all data lines which DON’T contain a missing value:

```
> filter(flights, ! is.na(dep_time))
...
```

… and counting them

```
count(filter(flights, ! is.na(dep_time)))
# A tibble: 1 x 1
      n
  <int>
1 328521
```

Remembering the data set

You can save the data set and can make it available with a (different) name

```
> my_flights <- filter(flights, ! is.na(dep_time))
```

The variable “my_flights” now contains the data set where the rows with missing values in the ”dep_time” are removed.

Identifying Missing Values: Replacing them (1/4)

Instead of deleting the rows with missing values you can also replace the missing values.

For example for the “dep_time”, there is a scheduled departure time “sched_dep_time” given. That value can be used for replacement.
First make a copy of “flights”

```
> flights_with_replaced_dep_time <- flights
```

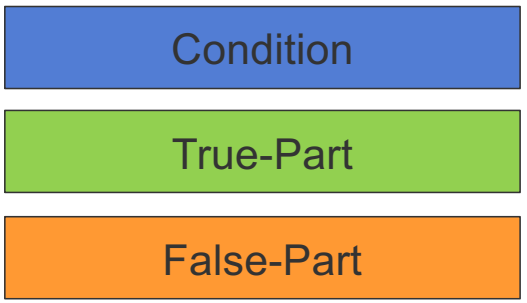
Identifying Missing Values: Replacing them (2/4)

Then address the attribute departure time `dep_time` in `flights_with_replaced_dep_time`, i.e. `flights_with_replaced_dep_time$dep_time`.

The function “ifelse”

- test a condition (here: `is.na(flights$dep_time)`).
- If the test succeeds, then the value from `flights$sched_dep_time` taken.
- Otherwise the original `flights$dep_time`

```
flights_with_replaced_dep_time$dep_time <-
        ifelse(is.na(flights$dep_time),
                flights$sched_dep_time,
                flights$dep_time)
```



Identifying Missing Values: Replacing them (3/4)

Check it with the following example:

```
> filter(flights, tailnum == "N18120")
# A tibble: 134 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1     1842           1422       260     1958           1535       263 EV        4633 N18120
 2  2013     1     1       NA           1630        NA       NA           1815        NA EV        4308 N18120
 3  2013     1     2      836            751        45     1059           1001        58 EV        4420 N18120
```

That would be with all NA's. But in

```
> filter(flights_with_replaced_dep_time, tailnum == "N18120")
# A tibble: 134 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1     1842           1422       260     1958           1535       263 EV        4633 N18120
 2  2013     1     1     1630           1630        NA       NA           1815        NA EV        4308 N18120
 3  2013     1     2      836            751        45     1059           1001        58 EV        4420 N18120
```

… we replaced one NA.

Identifying Missing Values: Replacing them (4/4)

You can also replace NA's with other values, e.g. 12:00

```
replacement <- 1200

flights_with_replaced_dep_time$dep_time <-
  ifelse(is.na(flights$dep_time),
         replacement,
         flights$dep_time)
```

… or replacing it by the mean (remove NA: "na.rm = TRUE"; convert the mean to an integer: "as.integer"):

```
replacement <- as.integer(mean(flights$dep_time, na.rm = TRUE))

flights_with_replaced_dep_time$dep_time <-
  ifelse(is.na(flights$dep_time),
         replacement,
         flights$dep_time)
```

Exercise B

Outliers

Identifying and Eliminating Outliers (1/2)

Lets analyse the departure delay:

```
> ggplot(flights) + geom_point(mapping = aes(x = flight, y = dep_delay))
```

There are some flights which really depart earlier than scheduled.

```
> arrange(flights, dep_delay)
year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013    12     7     2040           2123       -43       40           2352        48 B6          97 N592JB
 2  2013     2     3     2022           2055       -33     2240           2338       -58 DL        1715 N612DL
 3  2013    11    10     1408           1440       -32     1549           1559       -10 EV        5713 N825AS
 4  2013     1    11     1900           1930       -30     2233           2243       -10 DL        1435 N934DL
 5  2013     1    29     1703           1730       -27     1947           1957       -10 F9         837 N208FR
```

We catch them:

```
> minus_delay <- filter(flights, dep_delay <= 0)
```

Identifying and Eliminating Outliers (2/2)

Analyse the distribution of the negative departure delay:

```
> boxplot(minus_delay$dep_delay)
```

Some values are out of range. We remove all lines with have a lower negative departure delay than 29:

```
> my_flights <- filter(flights, dep_delay > -29)
```

University of Applied Sciences and Arts Northwestern Switzerland
School of Business

member of
swissuniversities

AACSB
ACCREDITED

Exercise C

Transformation and Normalization

Prerequisites

If not installed so far, please install the following additional packages

1. an additional package containing flights and
2. an additional package containing nice functions

```
install.packages("nycflights13")
install.packages("tidyverse")
```

After the installation these packages need to be "activated", also when you start R (or R-Studio)

```
library(nycflights13)
library(tidyverse)
```

Deleting columns

Deleting a column, e.g. dep_delay

```
> my_flight <- subset(flights,select=-dep_delay)
> my_flights
# A tibble: 336,776 × 18
    year month   day dep_time sched_dep_time arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <int>          <int>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1      517            515      830            819        11 UA        1545 N14228
 2  2013     1     1      533            529      850            830        20 UA        1714 N24211
 3  2013     1     1      542            540      923            850        33 AA        1141 N619AA
 ...
```

Deleting several columns, e.g. dep_delay and flight

```
> my_flight <- subset(flights,select=-c(dep_delay,flight))
> my_flights
# A tibble: 336,776 × 17
    year month   day dep_time sched_dep_time arr_time sched_arr_time arr_delay carrier tailnum origin
   <int> <int> <int>    <int>          <int>    <int>          <int>     <dbl> <chr>   <chr>   <chr>
 1  2013     1     1      517            515      830            819        11 UA      N14228  EWR
 2  2013     1     1      533            529      850            830        20 UA      N24211  LGA
 3  2013     1     1      542            540      923            850        33 AA      N619AA  JFK
 ...
```

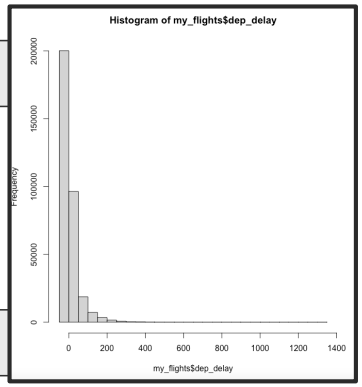
Transforming departure delay (1/2)

First remove all rows with missing values and remove rows with extreme negative delay

```
> my_flights <- filter(flights, ! is.na(dep_time))
> my_flights <- filter(my_flights, dep_delay > -29)
```

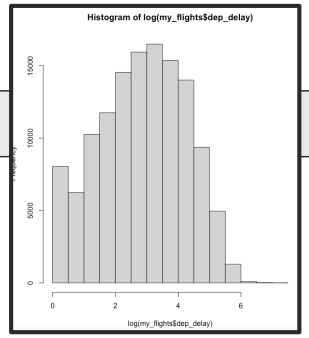
We now analyse the distribution

```
> hist(my_flights$dep_delay)
```



Looks imbalanced, looks like a logarithmic distribution; converting it to a more uniform distribution ..

```
> hist(log(my_flights$dep_delay))
```

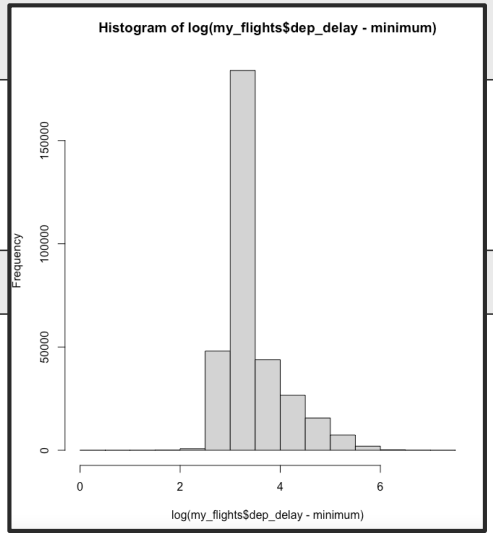


Now it looks better, but we produce NA (for the negative delays; log is not defined for negative inputs)

Transforming departure delay (2/2)

In order to remove negative values – and don't want to delete them – we simply shift the delays by the most negative value (i.e. the minimum). Then all values are positive.

```
> minimum <- min(my_flights$dep_delay,na.rm = TRUE)
> hist(log(my_flights$dep_delay - minimum))
```



Looks better now; we keep it!

```
> my_flights$dep_delay <- log(my_flights$dep_delay - minimum)
```

Normalising departure time (1/2)

First remove all rows with missing values

```
> my_flights <- filter(flights, ! is.na(dep_time))
```

We apply the min-max normalisation to “dep_time” (assuming a range of 0000-2359).

The new min is 0 and the new max is 1. Then $\frac{v-0000}{(2359-0000)} * (1-0) + 0 = \frac{v}{2359}$

```
> my_flights$dep_time <- my_flights$dep_time / 2359
> my_flights
# A tibble: 328,521 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <dbl>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1    0.219            515         2      830            819        11 UA        1545 N14228
 2  2013     1     1    0.226            529         4      850            830        20 UA        1714 N24211
 3  2013     1     1    0.230            540         2      923            850        33 AA        1141 N619AA
...
```

Normalising departure time (2/2)

However the coding of time in integer is not continuous. E.g. 1178 would never exists. We need a (self-defined) conversion function "time_conversion", which translates that into continuous numbers

```
> time_conversion <- function(x) {
h <- trunc(x/100,0)
m <- x-(h*100)
r <- m+(h*60)
return(r)
}
```

```
> my_flights$dep_time <- time_conversion(my_flights$dep_time) / (24*60)
> my_flights
# A tibble: 328,521 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <dbl>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1    0.220            515         2      830            819        11 UA        1545 N14228
 2  2013     1     1    0.231            529         4      850            830        20 UA        1714 N24211
 3  2013     1     1    0.238            540         2      923            850        33 AA        1141 N619AA
...
```

Exercise D

Variable Transformation

Prerequisites

In this example we use a different data set which has much more qualitative data:

```
> titanic <- data.frame(Titanic)
> titanic
   Class    Sex   Age Survived Freq
1    1st   Male Child       No    0
2    2nd   Male Child       No    0
3    3rd   Male Child       No   35
4   Crew   Male Child       No    0
5    1st Female Child       No    0
6    2nd Female Child       No    0
7    3rd Female Child       No   17
…
```

As usual we make a copy…

```
my_titanic <- titanic
```

Transforming the categorial (nominal) variable Survived (1/2)

First we need really to transform the column/variable ”Survived” into a factor:

```
> f <- factor(titanic$Survived)
```

what are the possible values of the factor f?

```
> levels(f)
 [1] "Male"   "Female"
```

The factor f is internally already a number (an integer, in order to be precise)

```
> typeof(f)
[1] "integer"
```

Transforming the categorial (nominal) variable Survived (2/2)

Now we only need to transform f to an integer …

```
> as.integer(f)
```

… and write it into the column/variable “Survived”

```
> my_titanic$Survived <- as.integer(f)
```

Or everything in one line

```
> my_titanic$Survived <- as.integer(factor(titanic$Survived))
> my_titanic
Class    Sex   Age Survived Freq
1    1st   Male Child        1    0
2    2nd   Male Child        1    0
3    3rd   Male Child        1   35
4   Crew   Male Child        1    0
```

Transforming the categorial (nominal) variable Sex

The column/variable ”Sex” is NOT ordered. Therefore we can not transform it into a factor. But we can create a unique (boolean) column for each value.

We use a special package which supports us in this task

```
> install.packages("fastDummies")
> library(fastDummies)
```

“dummy_cols” selects the column “Sex”, remove it, and add for each value a new (0/1) columns, i.e. two colums “Sex_Male” and “Sex_Female”.

```
> my_titanic <- dummy_cols(my_titanic, select_columns="Sex", remove_selected_columns = TRUE)
> my_titanic
   Class   Age Survived Freq Sex_Male Sex_Female
1    1st Child        1    0        1          0
2    2nd Child        1    0        1          0
3    3rd Child        1   35        1          0
4   Crew Child        1    0        1          0
```

Transforming the ordinal variable Age (1/2)

Age is qualitative but ordered. This time we would like to influence how the different values are translated into number. An Adult is older than a child. Therefore Child = 1, Adult = 2

```
> ordered(my_titanic$Age, levels= c("Child", "Adult"))
 [1] Child Child Child Child Child Child Child Child Adult Adult Adult Adult Adult Adult Adult Adult
[17] Child Child Child Child Child Child Child Child Adult Adult Adult Adult Adult Adult Adult Adult
Levels: Child < Adult
```

Converting it into integers:

```
> as.integer(ordered(my_titanic$Age, levels= c("Child", "Adult")))
 [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
```

Transforming the ordinal variable Age (2/2)

Now we can replace the values in columns Age

```
> my_titanic$Age <- as.integer(ordered(my_titanic$Age, levels= c("Child", "Adult")))
```

… resulting into:

```
> my_titanic
   Class Age Survived Freq Sex_Male Sex_Female
1    1st   1        1    0        1          0
2    2nd   1        1    0        1          0
3    3rd   1        1   35        1          0
4   Crew   1        1    0        1          0
```

Transforming the (partly) ordinal variable Class (1/2)

Class is qualitative but partly ordered. While the three values (”1st”, “2nd”, “3rd”) are obviously ordered, is the value “Crew” a little bit separated from that. Therefore we need to have a Boolean column for “Crew” but an ordered column for the other three values.

Create a column just for the Crew:

```
> ifelse(my_titanic$Class =="Crew",1,0)
```

Add an additional column to the data set. Now the crew is separated:

```
> my_titanic$Class_Crew <- ifelse(my_titanic$Class =="Crew",1,0)
```

Transforming the (partly) ordinal variable Class (2/2)

Now order Class :

```
> ordered(my_titanic$Class, levels= c("Crew", "3rd", "2nd","1st"))
[1] 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew
[21] 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew 1st  2nd  3rd  Crew
Levels: Crew < 3rd < 2nd < 1st
```

Converting it into integers:

```
> as.integer(ordered(my_titanic$Class, levels= c("Crew", "3rd", "2nd","1st")))
[1] 4 3 2 1 4 3 2 1 4 3 2 1 4 3 2 1 4 3 2 1 4 3 2 1 4 3 2 1 4 3 2 1
```

The right ‘Class numbers’ (i.e. subtract 1)

```
> as.integer(ordered(my_titanic$Class, levels= c("Crew", "3rd", "2nd","1st")))-1
[1] 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
```

```
> my_titanic$Class <-
      as.integer(ordered(my_titanic$Class, levels= c("Crew", "3rd", "2nd","1st")))-1
[1] 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
```

The resulting data set

```
> my_titanic
   Class Age Survived Freq Sex_Male Sex_Female Class_Crew
1      3   1        1    0        1          0          0
2      2   1        1    0        1          0          0
3      1   1        1   35        1          0          0
4      0   1        1    0        1          0          1
5      3   1        1    0        0          1          0
6      2   1        1    0        0          1          0
7      1   1        1   17        0          1          0
…
```