Handle your data!

Golden rules for data handling

The first three golden steps

1. Look at it
2. Look at it
3. LOOK at it

You should look at your data using any kind of software, try differnt software, look at it with R, with excel, with a text editor. Import it in R look at the types of the variable, look at the variable names, the values, is there some macroscopic property you did not expect, something that looks wrong ?

Use describe(). Use unique().

Look at each column separatly, count the number of NA ntrue(is.na(data$column)) (ntrue is a function of the pipeline package from the lab).

Keep the raw file untouched

You should not modify the raw file, or overwrite it. You load it, and every step of modification should be in a script that you can run anytime without risking overwriting or corrupting your raw data in any way. If it is really not possible, make sure to keep a copy of your initial raw data, make necessary manual modification, and consider this new file as raw data, but this should be kept to a minimum.

Rename your variables

For the lab prefer snake_case_names, and do not use abreviations like rt, use something that makes complete sense without any ambiguity like response_time. Do not fear to be overly clear, for example the mean of task 2 block 1 should not be name mt2b1, or mean_t2_b1, or avg_task2B1, but mean_correct_ratio_task2_block1. You can use the formatter of the pipeline to make sure all your variables are correctly named and documented.

Decide what to do with the values

Should they be formatted ? Normalized ? What to do with NA ?

Do not overwrite raw columns but rather create a new data.frame that is cleaned up and save it.

Example Datasets

https://data.humdata.org/ https://figshare.com/articles/Wellcome_Trust_APC_spend_2012_13_data_file/963054 https://data.humdata.org/dataset/health-outcomes

Two example datasets:

```r
library(readxl)
# A good practice is to have at the begining of your file the path to your data folder
dataFolder = "~/Google Drive/Master Students/courses/introduction_a_r/data"

# or to set manualy your working directory
```

```r
setwd("~/Google Drive/Master Students/courses/introduction_a_r")

# Country health outcome
healthOutcomes = read.csv(paste(c(dataFolder,"/raw/Health-Outcomes.csv"), collapse = ""))

#head(healthOutcomes)

# Somalia flow monitoring
somaliaFlow = read_excel("data/raw/Somalia_DTM-Flow-Monitoring-Dataset-Oct-Dec-2016.xlsx")

#head(somaliaFlow)
```

Data.table

Adapted from: https://www.analyticsvidhya.com/blog/2016/05/data-table-data-frame-work-large-data-sets/

What is a data.table ?

Think of data.table as an advanced version of data.frame. It inherits all the characteristics of a data.frame and works perfectly even when data.frame syntax is applied on data.table. This package is good to use with any other package which accepts data.frame.

DT[selection, action, by=c(columns,. . .)]

- DT is referred to the data table.
- Selection condition on which you will perform the action
- Action: Select the columns, create new columns, or even compute directly some functions of columns
- by: refers to any categorical variable i.e. put the variable on the basis of which the grouping shall be executed.

```r
library(data.table)
```

```r
#creating a dummy data table
DT <- data.table( ID = 1:50,
                Capacity = sample(100:1000, size = 50, replace = F),
                Code = sample(LETTERS[1:4], 50, replace = T),
                State = rep(c("Alabama","Indiana","Texas","Nevada"), 50))

#simple data.table command
DT[Code == "C", mean(Capacity), State]

DT[Code == "D"]
DT[, mean(Capacity), by = State]
DT[Code == "A", mean(Capacity)]

# Create a column
DT[, meanCapacity := mean(Capacity), by = State]

# Get a named computation
DT[, list(meanCapacity=mean(Capacity)),by = State]
```

Some usefull functions

```r
# Change the names of columns
names(DT) = c("id", "capacity", "code", "state", "meanCapacity")
setnames(DT, c("id", "state"), c("ID", "STATE"))

# Change the type of a column
DT[, ID:= as.character(ID)]

# Perform computation on columns
DT[, capacity := capacity - meanCapacity]

DT[, absoluteDiff := capacity - mean(capacity), by=STATE]
```

Preprocess

Names, values, NA

Explore

```r
# data.table
# psych
```