DATA CLEANING ISSUES

dr Maria Kubara, WNE UW

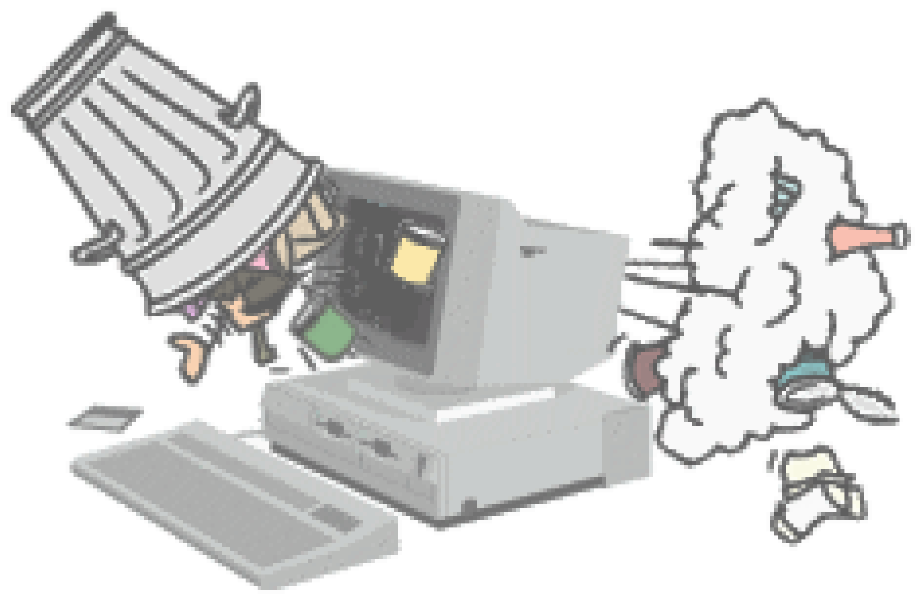
DATA PREPARATION

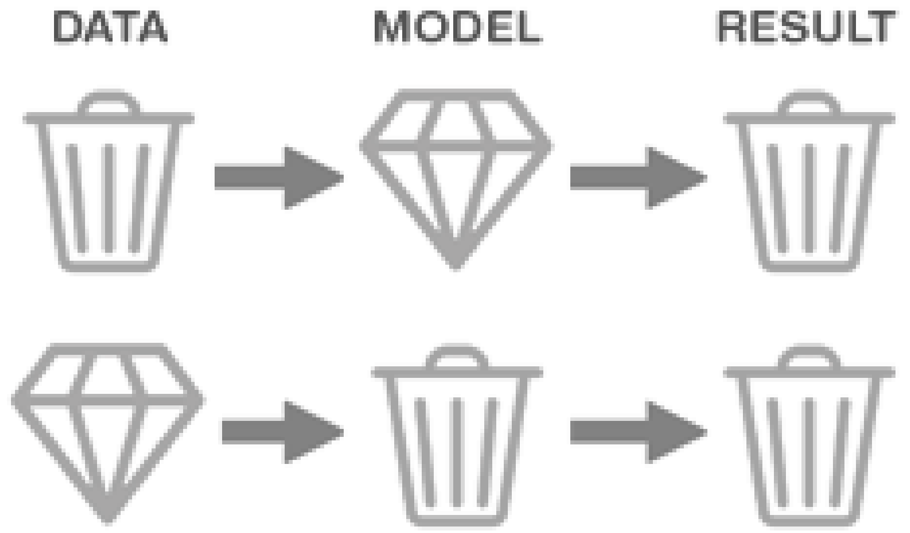
Data preparation is an important step prior to the actual data analysis. The process involves transforming the raw data into a shape that can be understood by a computer and algorithms (e.g. unsupervised learning).

Raw data is usually unstructured (messy data). They may contain errors, be incomplete or be stored without a consistent pattern (e.g. in the gender variable: male, man, 1...).

The computer processes data best in a numerical format. When analysing other types of data (e.g. text data), care should be taken to ensure that they are in the correct form (e.g. conversion to factor variables, division into similar groups, etc.).

GARBAGE IN — GARBAGE OUT





DATA PREPARATION

1. Quality assessment

2. Data cleaning

3. Data transformation

4. Data reduction

QUALITY ASSESSMENT

The step of:

- ❑ carefully reviewing the loaded data,
- ❑ getting to know their structure (how many variables are there, in which formats?),
- ❑ assessing the relevance in the context of the analysis to be carried out (perhaps some variables are not needed?),
- ❑ assessing the consistency of the data (are the data stored in a uniform way?).

QUALITY ASSESSMENT

- **unstructured data types** - values within variables should be comparable (e.g. should we check whether earnings values in different countries are presented in the same currency? Are the values presented in absolute form or, for example, in thousands?)
- **mixed data values** - check whether variables are coded uniformly for all observations (e.g. gender variable - is it stored everywhere as 1-0, or do text values appear, different terms - male, M, male, etc.)
- **outlier observations** - it may already be possible at this stage to discover observations with completely outlier results (e.g. with test results, perhaps someone handed in a blank sheet and scored 0 points - such observations can very seriously affect the average of all results)
- **missing observations** - check where there are empty fields in the data (e.g. questions in a survey left unanswered, missing values next to a variable indicating earnings, etc.).

DATA CLEANING

The process of filling in missing data, correcting data, removing discrepancies and discarding information irrelevant to further analysis.

DATA CLEANING – MISSING DATA

Missing data – lacking information in your dataset

Possible steps:

- remove observations with missing values from the dataset. Suitable if the missing data are a small part of a large dataset. Useful if there are observations with many incomplete fields (e.g. a particular respondent who was unwilling to answer the survey questions),
- Filling in missing data - the choice of filling-in technique depends on the analyst (e.g. using the mean, using prediction or regression, filling in with zeros or NA values). Better solution for small data sets.

DATA CLEANING – NOISY DATA

Noisy data - data containing too many elements (excessive text: e.g. 64GB storage capacity, instead of 64), unevenly distributed (e.g. earnings - it is often useful to group such data into groups of lower, average and higher earnings, instead of storing the exact value), variables containing information that is not relevant for further analysis, etc.

Possible steps:

- cleansing textual data of excess information (removing prefixes, sorting out unit information)
- grouping numerical data into broader categories (binning)
- dividing data into groups (e.g. by cluster analysis, clustering)
- removing redundant variables (e.g. index, duplicate columns, etc.)

DATA TRANSFORMATION

Data transformation involves organising the types of variables - transforming the data into the formats required for further analysis.

This step also includes elements related to the ordering of the scale of values - normalisation and standardisation.

DATA TRANSFORMATION

- **Data format ordering**: making sure variables are in a consistent, intended format.
- **Standardisation**: scaling the data to a certain range (to allow for comparison). The choice of scale is up to the researcher, often scaling values to a range of (-1,1) or (0,1) is used. Normalisation is also possible: transforming data according to a normal distribution.
- **Feature selection**: deciding which variables are most important for subsequent analysis. An important point from a machine learning perspective - the smaller the set of features, the faster we get results (sometimes the results are also more accurate - we reduce excess 'noise' in the data and base inference on the most important features.
- **Discretisation**: a process analogous to binning, but usually performed on cleaned data. It can involve combining information from several variables and transforming them into intervals (e.g. instead of storing exercise time in exact minutes and seconds, you can use a generalised variable in the form of intervals of 0-15 minutes, 15-30 minutes, etc.).

DATA REDUCTION

The more data, the more difficult the analysis (even after cleaning and transformation). Depending on the analytical task, there may be much more data than is needed (e.g. in text analysis, perhaps we do not need full statements, but rather keywords?).

Dimension reduction not only makes analysis easier and more accurate. Storing reduced data is far less burdensome.

DATA REDUCTION

- **Attribute selection:** a process similar to discretisation, it allows data to be fitted into smaller groups; it involves combining single, fragmented attributes into more general groups: e.g. having the variables 'gender' (levels 'female' and 'male') and is actor' (with a single level of 'yes'), these can be combined into the attributes actor', ‘actress'.
- **Numerosity reduction:** a process that supports the subsequent storage and processing of data; it involves selecting only those variables that are relevant for subsequent analysis.
- **Dimensionality reduction:** the process of reducing the amount of data used for later analysis; often involves using algorithms (k-nearest neighbours or PCA) to recognise patterns in the data and combine similar variables into more general grouping attributes.

READING DATA INTO R

DIFFERENT WAYS TO READ DATA IN R

R supports the analysis of various types of data - primarily data acquired from external sources.

Loading data into R is done using:

- general functions, derived from the base R
- extension functions, derived from additional packages installed in the local version of R

READING DATA – BASE FUNCTIONS

1. Localising the file and its absolute path

   (eg. „C:\Users\user1\files\file.extension”)

2. Convert the path to a form recognisable by Linux-based systems (R has many assumptions based on Linux). In practice, this means converting a path from Windows by changing the characters \ to / or \\. Paths extracted from Linux or macOS are given in a form readable by R.

3. Loading the data using functions built into R with the indication of the path to the file.

WORKING DIRECTORY

It is good practice to store data and scripts (used in a single project) in a collective directory. This keeps the files in order and makes analytical work easier.

The working folder can be declared in R. In this way, the files to be read can be pointed to using a short relative path, already starting from the location of the declared working directory.

`setwd()` – command to indicate the working directory

`getwd()` – check the path of the current working directory

When working with RStudio projects, the working directory is automatically set to the indicated project location.

READING DATA – EXTENSION FUNCTIONS

Loading of less common data types is done in R using functions from additional packages. For example, data from STATA, SAS, SPSS or Excel can be loaded in this way.

In this case, we must additionally install and load the package from which the function needed to load this type of data comes. The subsequent steps remain the same.

`install.packages("package")` – package installation

`library("package")` – loading the package in the current R session

Working on a given device and the selected version of R, packages only need to be installed once. The loading of packages should be repeated in each new R session when the need to use them arises. The code that loads packages is usually placed at the beginning of the script.

BASE FUNCTIONS

Data type	Extension	Function

R data files (RData)	rdata, rda	Function: `load()`
		
		`load("survey.rdata")`
		`load("survey.rda")`

R data files, with one variable only (RDS)	rds	Function: readRDS() dataRDS <- readRDS("survey.rds")

Separated data (with commas, spaces, semicolons, etc.)	txt, csv, dat	Function: `read.table()` or `read.csv()`
		
		Important parameters: sep = " " (default, space), sep = "\t" (tab), sep = "," (comma), sep = ";" (semicolon), header = TRUE (if there is a header).
		
		`dataSpace <- read.table("survey1.csv", header=TRUE)`
		`dataComma <- read.csv("survey2.csv", header=TRUE)`

EXTENSION FUNCTIONS

Data source	Package	Function

SPSS	foreign	Function: `read.spss()`
		
		`library(foreign)`
		`dataSPSS <-`
		`read.spss("C:/daa/survey.save",`
		`to.data.frame=TRUE)`

STATA	foreign	Function: `read.data()`
		
		`library(foreign)`
		`dataStata <- read.dta("survey.dta")`

SAS	foreign	Function: `read.xport()`
		
		`library(foreign)`
		`dataSAS <- read.xport("C:/data/survey")`

EXCEL	readxl	Function: `read_excel()`
		
		`library(readxl)`
		`dataEXCEL <- read_excel("survey.xlsx",`
		`sheet = 1)`
		`dataEXCEL <- read_excel("survey.xlsx",`
		`sheet = "sheetname")`

SAVING YOUR WORK TO R FILES

Save a single object to a file:

```
saveRDS(object, "save1.rds")
```

To restore it under a different name:

```
my_data <- readRDS("save1.rds")
```

Saving an object in RData format

```
save(data1, file = "save2.RData")
```

Save multiple objects and load them:

```
save(data1, data2, file =
    "save2.RData")

load("save2.RData")
```

Saving your entire session can be helpful if you are working on bigger analytical project and want to resume your work in the next day exactly as you left it. To do that you need to save the entire workspace image:

```
save.image(file = "image.RData")
```

To restore your workspace, type this:

```
load("image.RData")
```