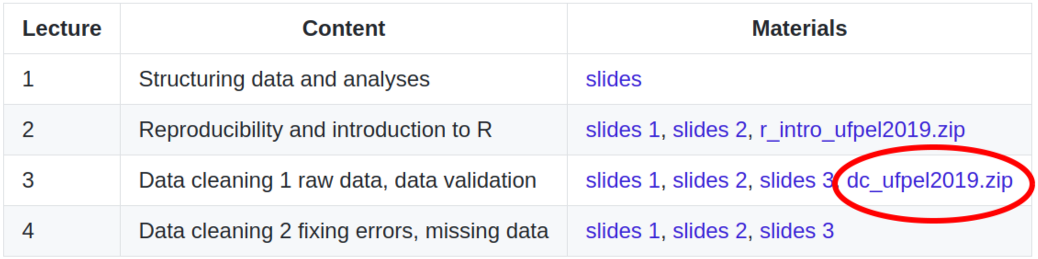
Data Cleaning Tutorial: Transforming Raw Data

Mark van der Loo

Please download and unzip

`github.com/markvanderloo/UFPEL2019`



Lecture	Content	Materials
1	Structuring data and analyses	slides
2	Reproducibility and introduction to R	slides 1, slides 2, r_intro_ufpel2019.zip
3	Data cleaning 1 raw data, data validation	slides 1, slides 2, slides 3, dc_ufpel2019.zip
4	Data cleaning 2 fixing errors, missing data	slides 1, slides 2, slides 3

Try the code together with your neighbour

`02input/json_cleanup.R`

Reading Dirty Data

Reading raw data is about uniforming the technical layout of the data.

We saw the following issues in the example:

- Storage formats
- Data types
- Locales...
- Unit suffixes.
- Non-uniform formatting

Storage formats

Your raw data is often stored in external formats:

- A database (dump).
- Scraped data in html format.
- Data retrieved from an api (JSON/XML).
- csv / txt/ spss / stata / sas / etc.

Action:

Turn the data into a tabular format

Data types

Most important reading action is setting data types:

- numeric columns often contain units (“42%”, “42 $”, “42 km”, etc.)
- numeric columns often contain locale dependent formatting: “4.2” vs “4,2” or “4.200,42” vs “4,200.42”
- date columns (not in example): “4/2/1942” vs “42/2/4” vs “1942-4-2” etc.
- encoding of strings (for non english): “forty two” vs “tweeënveertig”
- footnotes/annotations[1] in values may corrupt data type: 42^

Cleaning action

Fix the columns: remove non-relevant info and set data type

[1] Footnotes can be essential to data interpretation, but desastrous for technical parsing.

Uniforming values

Some text variables are structured, but contain variations:

- phone numbers: “+311234567890” vs “0031-12-34567890” vs “012-34567890”
- soc. sec. numbers: “1234 5678” vs “123456789”
- bank account numbers, etc.
- zip codes: e.g Dutch zip codes: “1234AB” vs “1234 AB” vs “1234 ab”.

Cleaning action

Reformat values into standard format

String manipulations

Replace text in columns

```r
library(stringr)
str_replace(c("3,4", "4,5"), ",", ".") # or use gsub
```

Change casing of characters

```r
str_to_lower("HUGE")
```

Tip: use `readr::parse_*` functions

```r
library(readr)
parse_number(c("3,4", "5,6"), locale = locale(decimal_mark = ","))
```

```
## [1] 3.4 5.6
```

Encoding

```r
bands <- c("Motörhead", "Iron Maiden")
Encoding(bands)
```

```
## [1] "unknown" "unknown"
```

```r
Encoding(bands) <- "latin1"
print(bands)
```

```
## [1] "MotÃ¶rhead"  "Iron Maiden"
```

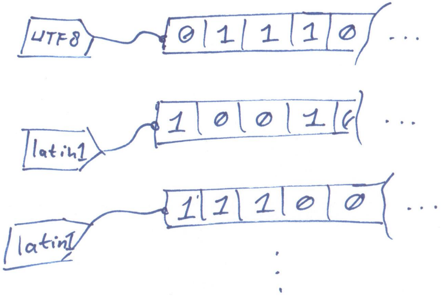
```r
Encoding(bands) <- "UTF-8"
print(bands)
```

```
## [1] "Motörhead"   "Iron Maiden"
```

Character vector x

Encoding(x) <- "UTF-8"

only changes the labels, not the data



Use iconv() to change the encoding.

Figure 1:

Assigment

Read in the “01raw/backbone.xml” file and **reformat the zip code** into 4 digits, no space, two uppercase letters. “1234 ab” -> “1234AB”

```r
library(XML)
backbone_xml <- XML::xmlParse("01raw/backbone.xml")
backbone <- xmlToDataFrame(backbone_xml)

# Fix zip code
...

# and save the result
write.csv( backbone, "01raw/my_backbone.csv"
         , row.names = FALSE, na="")
```