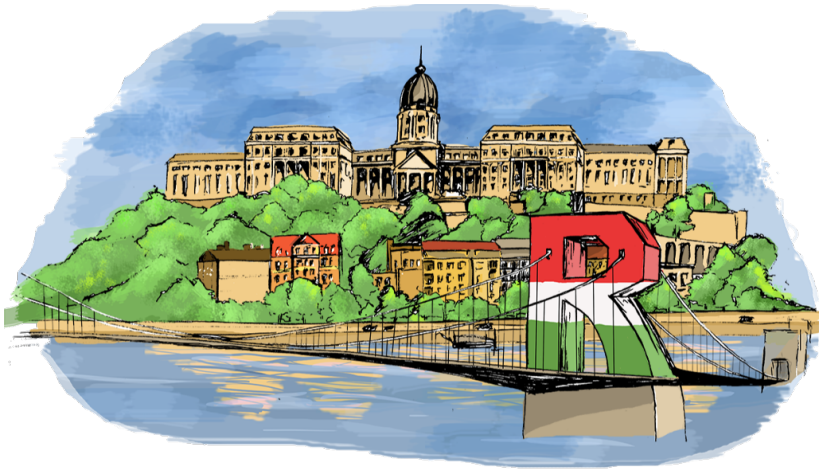
Validatetools

Edwin de Jonge, Statistics Netherlands

@edwindjonge | github.com/edwindj



Who am I?

- Data scientist / Methodologist Statistics Netherlands (aka CBS).
- Author of several R-packages, including `whisker`, `validate`, `errorlocate`, `docopt`, `tableplot`, `chunked`, `ffbase`,...
- Co-author of *Statistical Data Cleaning with applications in R (2018)* (sorry for the plug, but relevant for this talk...)

CAUTION: BAD DATA

BAD DATA QUALITY MAY RESULT IN FRUSTRATION AND LEAD TO DROP KICKING YOUR COMPUTER

Data cleaning...

A large part of your and our job is spent in data-cleaning:

- getting your data in the right shape (e.g. `tidyverse`)
- assessing missing data (e.g. `VIM`)
- checking validity (e.g. `validate`)
- locating and removing errors: `errorlocate`!
- impute values for missing or erroneous data (e.g. `simputation`)


RULES

Cleaning philosophy

- "Explicit is better than implicit".
- Make data knowledge as explicit as possible.
- Store these as validation rules.

Advantages:

- Easy checking of rules
- Data quality statistics: how often is each rule violated?
- Allows for reasoning on rules: which variables are involved in errors?
- Simplifies rule changes and additions.

Rules

- Data rules are solidified domain knowledge.
- Real world knowledge e.g. :
  - age is not negative.
  - human age is less then 150 years.
- Expert knowledge, e.g:
  - `IF profit > 0 THEN turnover > 0`
  - `IF married THEN age > 16`

Rules (2)

A lot of datacleaning packages are based on explicit rules that data must conform to.

- `validate` to check **validity** of data
- `errorlocate` to find **errors**.
- `rspa`, `deductive`, `dcmodify` for **correction** and **imputation** using data rules.

R package `validate`

Package `validate` allows to:

- formulate explicit data rule that data must conform to:

```
library(validate)
check_that( data.frame(age=160, job = "yes", income = 3000)
  age >= 0,
  age < 150,
  job %in% c("yes", "no"),
  if (driver_license == TRUE) age >= 16
)
```


Data checking

- A large part of data quality assurance in Official Statistics is checking data validity:
- n, number of records is high, typically > 0.5M
- p, number of columns is high, typically > 20
- population is diverse, different rules for different subpopulations.
- often many processing steps from input to statistic each checking/using (implicit) domain knowledge.

Result:

- Often many rules, great and small.
- Rules often defined multiple times at different processing steps.
- Rules may partially contradict each other.

Why-o-why `validatetools`?

- We have package `validate`, what is the need?

Because we'd like to...

- clean up rule set (kind of meta-cleaning...).
- detect and resolve problems with rules:
  - redundant rules
  - conflicting rules
  - unintended consequences.
- check the rule set using formal logic (without any data!).
- solve these kind of fun problems :-)

Formally...

Rule set S

A validation rule set S is a conjunction of rules r_i, which applied on record $\boldsymbol{x}$ returns `TRUE` (valid) or `FALSE` (invalid)

$$S(\boldsymbol{x}) = r_1(\boldsymbol{x}) \wedge \cdots \wedge r_n(\boldsymbol{x})$$

Note

- a record has to comply to each rule r_i.
- it is thinkable that two or more r_i are in conflict, making each record invalid.

Formally...

Rule $r_i(x)$

A rule a disjunction of atomic clauses:

$$r_i(x) = \bigvee_j C_i^j(x)$$

with:

$$C_i^j(\boldsymbol{x}) = \begin{cases} \boldsymbol{a}^T \boldsymbol{x} \leq b \\ \boldsymbol{a}^T \boldsymbol{x} = b \\ x_j \in F_{ij} \text{with } F_{ij} \subseteq D_j \\ x_j \notin F_{ij} \text{with } F_{ij} \subseteq D_j \end{cases}$$

Formally, note

- We restrict ourselves to linear, categorical and mixed constraints.
- Rules are not independent.

Mixed Integer Programming

Problem: infeasibility

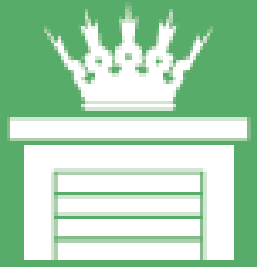
Problem

One or more rules in conflict:

- $S(x)$ always is 'FALSE, **no data correct**
- *happens more often than you think*

```
library(validatetools)
rules <- validator( is_adult = age >=21
                  , is_child = age < 18
                  )
is_infeasible(rules)
```

```
## [1] TRUE
```

KEEP CALM

AND

RESOLVE CONFLICT

Conflict, and now?

```
# Find out which rule would remove the conflict
detect_infeasible_rules(rules)
```

```
## [1] "is_adult"
```

```
# And itś conflicting rule(s)
is_contradicted_by(rules, "is_adult")
```

```
## [1] "is_child"
```

- One of these rules needs to be removed
- Which one? Depends on human assessment...

Detecting and removing redundant rules

- Rule r_A may imply r_B, so r_B can be removed.
- Simplifies rule set!

```
rules <- validator( r_A = age >= 18
                  , r_B = age >= 12
                  )
detect_redundancy(rules)
```

```
##   r_A   r_B
## FALSE  TRUE
```

```
remove_redundancy(rules)
```

```
## Object of class 'validator' with 1 elements:
##  r_A: age >= 18
```


Value substitution

```
rules <- validator( r1 = if (gender == "male") weight > 50
                  , r2 = gender %in% c("male", "female")
                  )

substitute_values(rules, gender = "male")
```

```
## Object of class 'validator' with 2 elements:
##   r1             : weight > 50
##   .const_gender: gender == "male"
```

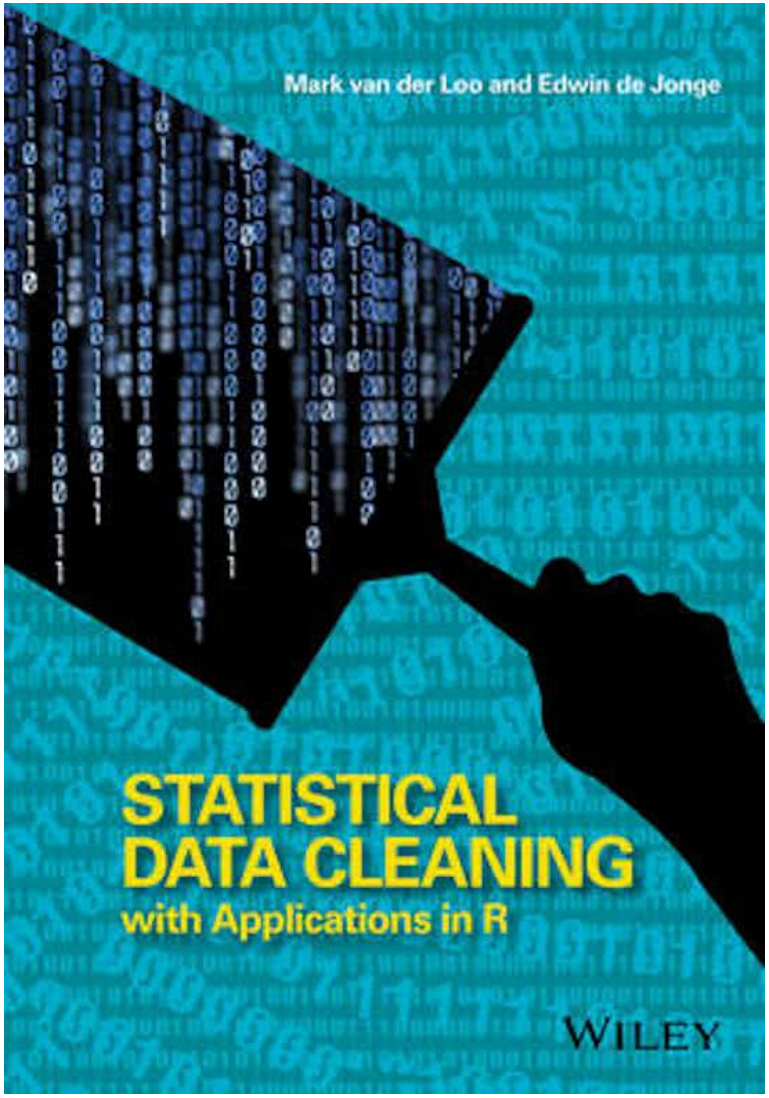

All to gether now:

`simplify_rules` applies all simplification methods to the rule set

```
rules <- validator( r1 = if (age < 16) income == 0
                  , r2 = job %in% c("yes", "no")
                  , r3 = if (job == "yes") income > 0
                  )
simplify_rules(rules, job = "yes")
```

```
## Object of class 'validator' with 3 elements:
##   r1          : age >= 16
##   r3          : income > 0
##   .const_job: job == "yes"
```

Interested?



SDCR

M. van der Loo and E. de Jonge (2018) *Statistical Data Cleaning with applications in R* Wiley, Inc.

validatetools

- Available on CRAN

More theory?

← See book

Thank you for your attention! / Köszönöm a figyelmet!