# Validate your data

Mark van der Loo and Edwin de Jonge

Statistics Netherlands Research & Development
@markvdloo @edwindjonge

useR!2021

## `validate`: *data validation infrastructure for R*

### A domain-specific language for rule definition

Define *any* check on your data, using the *full power* of the R language.

### Rules as first-class citizens

- CRUD operations (create, read, update, delete)
- Summarize, plot, investigate rules
- Rich metadata

### Validate data

- Confront data with rules
- CRUD on results, summarize, plot
- Export to ESS standard reporting format (upcoming)

## Assignment 1

Try the following code.

```r
library(validate)
library(magrittr)
data(retailers)
head(retailers)
retailers %>%
  check_that(turnover + other.rev == total.rev
             , turnover > 0, other.rev > 0 ) %>%
  summary()
```

## Assignment 1

```r
library(validate)
library(magrittr)
data(retailers)
retailers %>%
  check_that(turnover + other.rev == total.rev
             , turnover > 0, other.rev > 0 ) %>%
  summary()
```

```
##   name items passes fails nNA error warning
## 1   V1    60     19     4  37 FALSE   FALSE
## 2   V2    60     56     0   4 FALSE   FALSE
## 3   V3    60     23     1  36 FALSE   FALSE
##                                      expression
## 1 abs(turnover + other.rev - total.rev) <= 1e-08
## 2                                  turnover > 0
## 3                                 other.rev > 0
```

## Data validation with `validate`

```r
library(validate)
data(retailers)
head(retailers,3)[3:7]
```

```
##   staff turnover other.rev total.rev staff.costs
## 1    75       NA        NA      1130          NA
## 2     9     1607        NA      1607         131
## 3    NA     6886       -33      6919         324
```

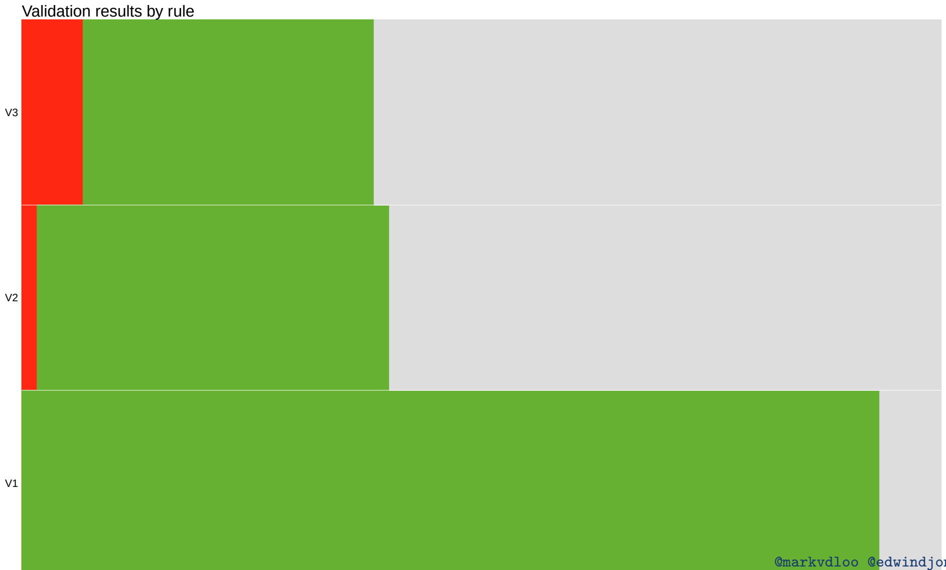
## Data validation with `validate`

```r
rules <- validator(
   turnover >= 0
  , other.rev >= 0
  , turnover + other.rev == total.rev
)

out <- confront(retailers, rules)
summary(out)
```

## Plotting output

```
plot(out)
```



## Reading rules from file

```
### myrulez.txt

# some basic checks
staff >= 0
turnover >= 0
other.rev >= 0
# account balance checks
turnover + other.rev == total.rev
# other common sense stuff
if (staff >= 1) staff.costs >= 1
```

```
rulez <- validator(.file="myrulez.txt")
```

## Domain Specific Language

### Validation DSL

Any R statement resulting in a `logical`.

### Examples

```
# Range checks
has_job %in% c('yes','no')
turnover >= 0
# Multivariate checks
abs(profit) <= 0.6 * turnover
# Multi-row checks
mean(profit) > 10
# Logical implications
if (staff > 0) staff.costs > 0
```

## Validation DSL

### Comparisons

`>, >=,==, <=, <, %in%`

### Completeness

`is.complete`

### Boolean operations

`!, all(), any(), &, &&, |, ||, if () else`

### Text formatting

`grepl`, `field_length`, `field_format`

### Functional dependencies (Armstrong)

`city + zipcode ~ streetname`

### Refer to the dataset with .

`nrow(.) == 40`, `"turnover" %in% names(.)`

## Transient assignments (macros) using :=

### Example 1

$$\max\left(\frac{x}{x^*}, \frac{x^*}{x}\right) \leq 10$$

```
med := median(turnover,na.rm=TRUE)
hb := pmax(turnover/med, med/turnover, na.rm=TRUE)
hb <= 10
```

### Example 2

```
beta_2 := coefficients(lm(turnover ~ profit))[2]
beta_2 >= 0
```

## Variable groups

### Many variables, same rule

```
G := var_group(staff, turnover, other.rev, total.costs)
G >= 0
```

## Error handling

```r
out <- check_that(women, hite > 0, weight>0)
out
```

```
## Object of class 'validation'
## Call:
##     check_that(women, hite > 0, weight > 0)
##
## Rules confronted: 2
##    With fails    : 0
##    With missings: 0
##    Threw warning: 0
##    Threw error   : 1
```

```r
errors(out)
```

```
## $V1
## [1] "object 'hite' not found"
```

## Naming rules

```r
rules <- validator(
  to_pos = turnover >= 0
  , or_pos = other.rev >= 0
  , balance = turnover + other.rev == total.rev)
rules
```

```
## Object of class 'validator' with 3 elements:
##  to_pos : turnover >= 0
##  or_pos : other.rev >= 0
##  balance: turnover + other.rev == total.rev
```

## Rule selection

```r
rules[1:2]
```

```
## Object of class 'validator' with 2 elements:
##  to_pos: turnover >= 0
##  or_pos: other.rev >= 0
## Rules are evaluated using locally defined options
```

```r
rules["balance"]
```

```
## Object of class 'validator' with 1 elements:
##  balance: turnover + other.rev == total.rev
## Rules are evaluated using locally defined options
```

## Rule metadata

```r
rules[[3]]
```

```
##
## Object of class rule.
##  expr       : turnover + other.rev == total.rev
##  name       : balance
##  label      :
##  description:
##  origin     : command-line
##  created    : 2021-07-07 12:59:27
##  meta       : language<chr>, severity<chr>
```

## More manipulation: combining rule sets

```r
validator(x > 0) + validator(x <= 1)
```

```
## Object of class 'validator' with 2 elements:
##  V1  : x > 0
##  V1.1: x <= 1
```

## Export rules & metadata to and import from `data.frame`

### Create data frame

```r
rules_df <- as.data.frame(rules)
```

### Read from data frame

```r
myrules <- validator(.data = rules_df)
```

## Setting options

### Global options

```
# stop at error instead of catching
voptions(raise="all")
```

### Options per object

```
# value to replace NA outcomes
voptions(rules, na.value=FALSE)
```

### When confronting data with rules

```
out <- confront(retailers, rules
          , lin.eq.eps=1e-2 )
```

## validatedb

- Sometimes data is big and stored in a database
- `validatedb` executes `validate` checks on a database.
- `checks` are translated into SQL code.