Data validation infrastructure: the **validate** package

Mark van der Loo and Edwin de Jonge

Statistics Netherlands

Validate

Goal

To make checking your data against domain knowledge and technical demands as easy as possible.

Content of this talk

- Basic concepts and workflow
- Examples of possibilities and syntax
- Outlook

Basic concepts and workflow

Data Validation

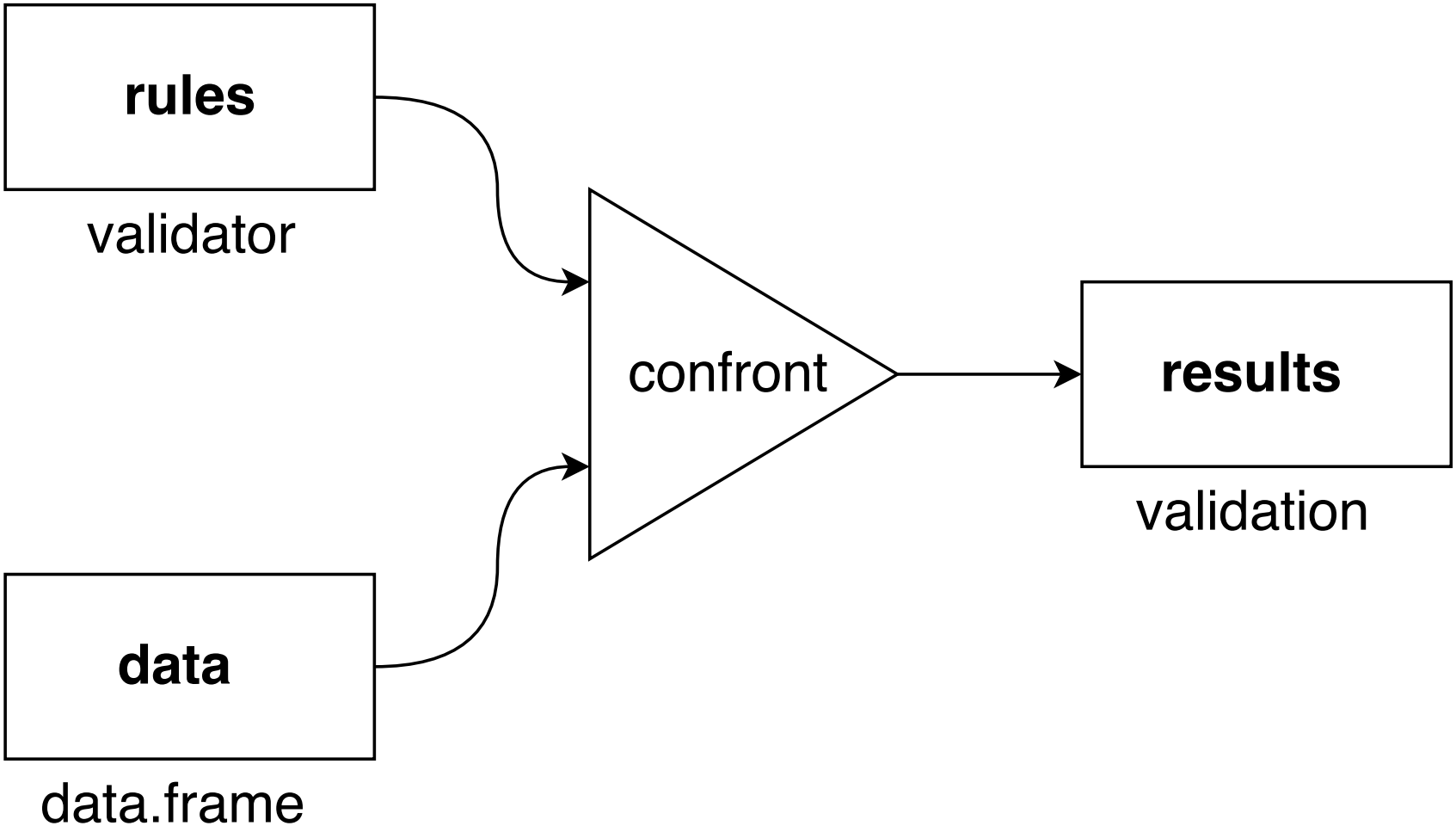
Intuitively

Check whether data holds up to (multivariate) relations that you expect from domain knowledge.

Validation rules

These relations can often be expressed as a set of simple rules.

Basic concepts of the **validate** package



Example: retailers data

```
library(validate)
data(retailers)

dat <- retailers[4:6]
head(dat)
```

```
##   turnover other.rev total.rev
## 1       NA        NA      1130
## 2     1607        NA      1607
## 3     6886       -33      6919
## 4     3861        13      3874
## 5       NA        37      5602
## 6       25        NA        25
```

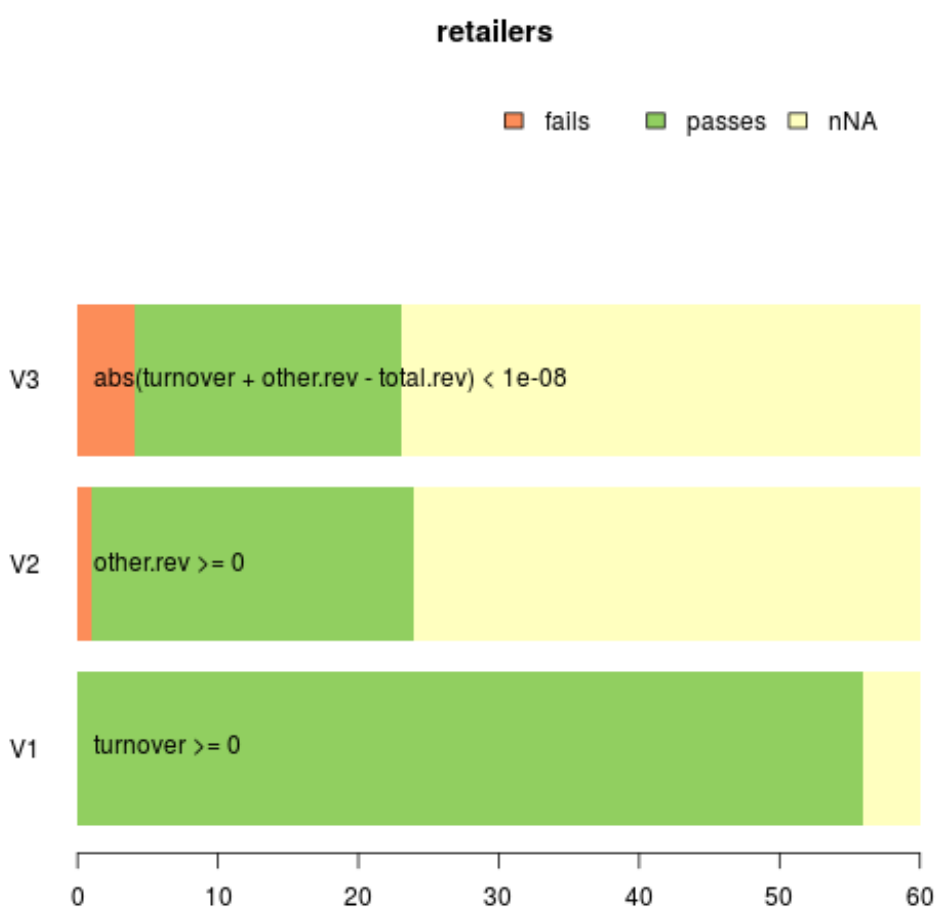
Basic workflow

```
# define rules
v <- validator(turnover >= 0, other.rev >= 0
  , turnover + other.rev == total.rev)
# confront with data
cf <- confront(dat, v)
# analyze results
summary(cf)
```

```
##   rule items passes fails nNA error warning
## 1   V1    60     56     0   4 FALSE   FALSE
## 2   V2    60     23     1  36 FALSE   FALSE
## 3   V3    60     19     4  37 FALSE   FALSE
##                                   expression
## 1                                turnover >= 0
## 2                             other.rev >= 0
```

Plot the validation

```
barplot(cf, main="retailers")
```



Get all results

```
# A value For each item (record) and each rule
head(values(cf))
```

```
##         V1    V2    V3
## [1,]    NA    NA    NA
## [2,]  TRUE    NA    NA
## [3,]  TRUE FALSE FALSE
## [4,]  TRUE  TRUE  TRUE
## [5,]    NA  TRUE    NA
## [6,]  TRUE    NA    NA
```

Shortcut using **check_that()**

```
cf <- check_that(dat
     , turnover >= 0
     , other.rev >= 0
     , turnover + other.rev == total.rev)
```

Or, using the **magrittr** 'not-a-pipe' operator:

```
dat %>%
  check_that(turnover >= 0
     , other.rev >= 0
     , turnover + other.rev == total.rev) %>%
  summary()
```

Read rules from file

```
### myrules.txt

# inequalities
turnover >= 0
other.rev >= 0

# balance rule
turnover + other.rev == total.rev
```

```
v <- validator(.file="myrules.txt")
```

Validation features

Validating aggregates

Rule

Mean turnover must be at least 20% of mean total revenue

Syntax

```
mean(total.rev,na.rm=TRUE) /
  mean(turnover,na.rm=TRUE) >= 0.2
```

Rules that express conditions

Rule

If the total revenue is larger than zero, the turnover must be larger than zero.

Syntax

```
# executed for each row
if (total.rev > 0) turnover > 0
```

Functional dependencies

Rule

Two records with the same zip code, must have the same city and street name.

$$zip \rightarrow city + street$$

```
zip ~ city + street
```

Using transient variables

Rule

The turnover must be between 0.1 and 10 times its median

Syntax

```
# transient variable with the := operator
med := median(turnover,na.rm=TRUE)
turnover > 0.1*med
turnover < 10*med
```

Don't repeat yourself

Rule

Turnover, other revenue, and total revenue must be between 0 and 2000.

Syntax

```
G := var_group(turnover, other.rev, total.rev)

G >= 0
G <= 2000
```

Validating types

Rule

Turnover is a numeric variable

Syntax

```
# any is.-function is valid.
is.numeric(turnover)
```

Validating metadata

Rules

- *The variable **foo** must be present.*
- *The number of rows must be at least 20*

Syntax

```
# use the "." to access the dataset as a whole
"foo" %in% names(.)
nrow(.) >=  20
```

Referencing other datasets

Rule

The mean turnover of this year must not be more than 1.1 times last years mean.

Syntax

```
# use the "$" operator to reference other datasets
v <- validator(
  mean(turnover, na.rm=TRUE) <
    mean(lastyear$turnover,na.rm=TRUE))

cf <- confront(dat, v, ref=list(lastyear=dat_lastyear))
```

Other features

- Rules (**validator** objects)
  - Select from validator objects using `[]`
  - Extract or set rule metadata (label, description, timestamp, ...)
  - Get affected variable names, rule linkage
  - Summarize rules
  - Read/write to `yaml` format
- Confront
  - Control behaviour on `NA`
  - Raise errors, warnings
  - Set machine rounding limit

Outlook

In the works / ideas

- More analyses of rules
- More programmability
- More (interactive) visualisations
- Roxygen-like metadata specification
- More support for reporting
- ...

We'd ♡ to hear your comments, suggestions, bugreports

Validate is just the beginning!

See github.com/data-cleaning

References

- Di Zio et al (2015) Methodology for data validation. ESSNet on validation, deliverable. [pdf]
- Van der Loo (2015) A formal typology of data validation functions. in *United Nations Economic Comission for Europe Work Session on Statistical Data Editing* , Budapest. [pdf]
- Van der Loo, M. and E. de Jonge (2016). Statistical Data Cleaning with Applications in R, *Wiley* (in preparation).

Contact, links

Code, bugreports

- cran.r-project.org/package=validate
- github/data-cleaning/validate

This talk

slideshare.com/markvanderloo

Contact

mark.vanderloo@gmail.com @markvdloo
edwindjonge@gmail.com @edwindjonge