Introduction to data validation and `validate`

Mark van der Loo

Statistics Netherlands | Research & Development
uRos2024, Athense, Greece

Getting Started

```
git clone https://github.com/markvanderloo/2024uRos
```

- Create an RStudio project in `2024uRos`
- Open the file `01_validate.R`

Goals of this presentation

- Define data validation formally
- Insight into the workings of validate and its syntax

Data Validation

Some examples from a survey amongst the ESS member states

- If a respondents has *income from other activities*, fields under *other activities* must be filled.
- *Yield per area* must be between 40 and 60 metric tons
- A person of *age* under 15 cannot *take part in an economic activity*
- The field *type of ownership* (of a building) may not be empty
- The *regional code* must be in the code list.
- The *current average price* divided by *last period's average price* must lie between 0.9 and 1.1.

Definition (European Statistical System)

Definition of data validation

Data Validation is an activity verifying whether or not a combination of values is a member of a set of acceptable combinations.

Di Zio *et al* Methodology of Data Validation (ESS Handbook, 2016)

Formal definition (sort of)

A *data validation function* is a function that accepts a data set and returns `True` (valid) or `False` (invalid)

Notes

- We skip some fine-print, such as the domain of a data validation function
- There is no real difference between data validation functions and data validation rules:

$$\text{Age} \geq 0 \iff f(\text{Age}) = \begin{cases} \texttt{True} \text{ if } \text{Age} \geq 0 \\ \texttt{False} \text{ otherwise} \end{cases}$$

Examples of validation rules

Single variable; multiple variables

$Age \geq 0;\ Age < 15 \Rightarrow Has\_Job = \texttt{no}$

Multiple entities

$\mathsf{mean}(Profit) \geq 10$

Multiple times or domains

$0.9 < \mathsf{mean}(Profit_{2018})/\mathsf{mean}(Profit_{2017}) < 1.1$

Assessing the complexity of data validaton rules

1. Answer with s or m: do we need values from
   - a $s$ingle, or $m$ultiple entity types (populations) U?
   - a $s$ingle, or $m$ultiple measurements τ?
   - a $s$ingle, or $m$ultiple population units u?
   - a $s$ingle, or $m$ultiple variables X?
2. The *complexity label* is the 4-tuple of s's and m's
3. The *complexity level* is the number of m's counted

MPJ van der Loo, E de Jonge (2020). Data Validation. In Wiley StatsRef: Statistics Reference Online

Examples: $U\tau uX$ classification of data validation rules

Single variable; multiple variables

- $Age \geq 0$: $ssss$, level 0
- $Age < 15 \Rightarrow Has\_Job = \texttt{no}$: $ssssm$, level 1

Multiple entities

- mean$(Profit) \geq 10$ $ssms$, level 1

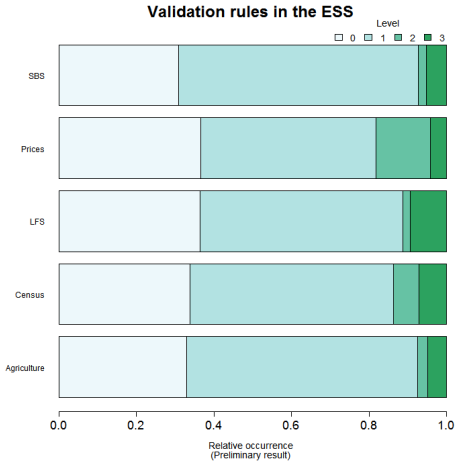
Multiple times or domains

- $0.9 <$ mean$(Profit_{2018})/$mean$(Profit_{2017}) < 1.1$: $smss$, level 1

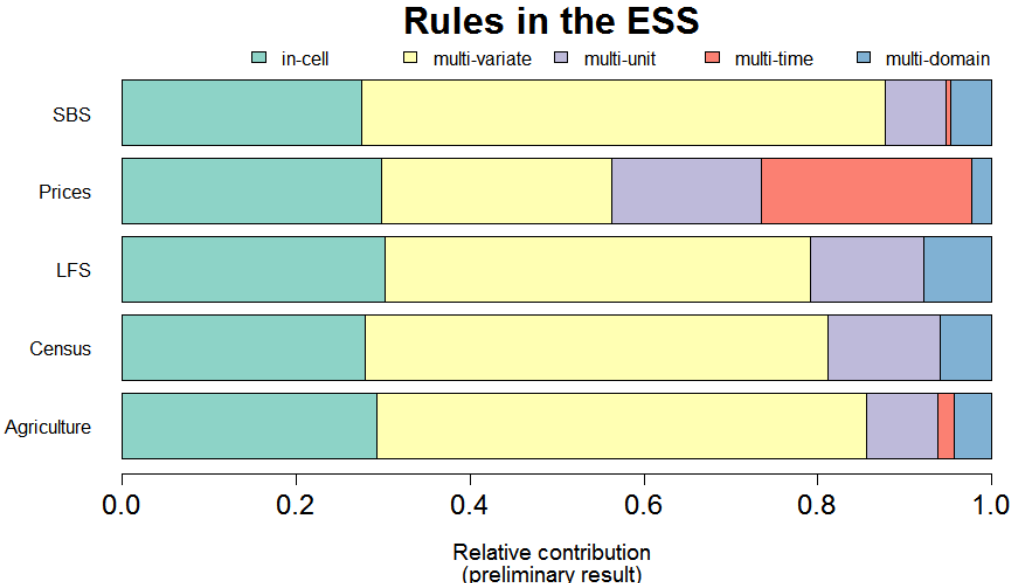
Possible validation rule classes

Validation level				
0	1	2	3	4
ssss	sssm	ssmm	smmm	mmmm
	ssms	smsm	msmm	
	smss	smms		

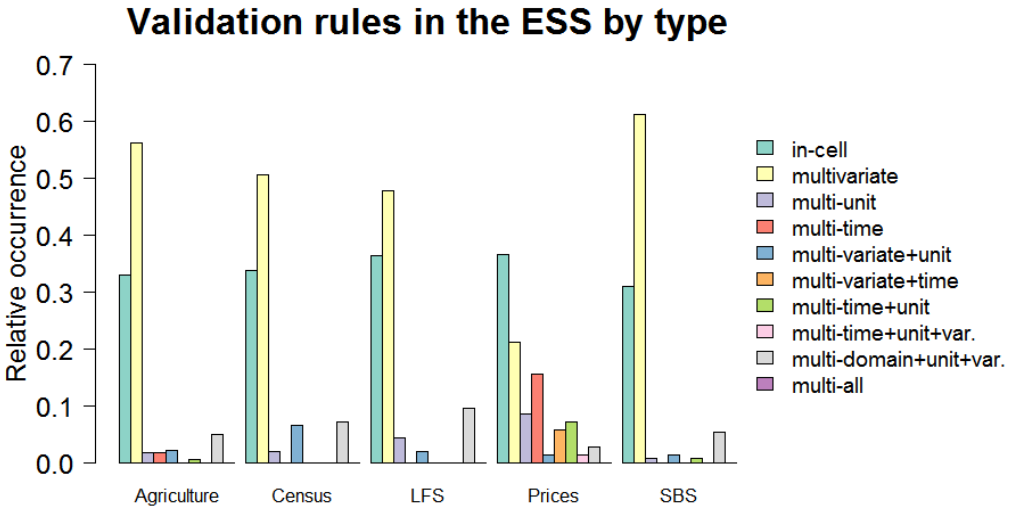
Validation rules in the ESS (1/3)



Validation rules in the ESS (2/3)



Validation rules in the ESS (3/3)



Quizz (1)

What is the $U\tau uX$ $s$ingle/$m$ulti classification of the following rule?

$$\text{mean}(price) \geq 1$$

Quizz (2)

What is the $U\tau uX$ $s$ingle/$m$ulti classification of the following rule?

$$\frac{\text{mean}(price_{2018})}{\text{mean}(price_{2017})} \leq 1.1$$

Quizz (3)

What is the $U\tau uX$ $s$ingle/$m$ulti classification of the following rule?

$$\max\left(\frac{x}{\text{median}(X)}, \frac{\text{median}(X)}{x}\right) < 10$$

Quizz (4)

What is the $U\tau uX$ $s$ingle/$m$ulti classification of the following rule?

$$\underbrace{COE + GOS + GMI + T_{P\&M} - S_{P\&M}}_{\text{GDP, Income approach}} = \underbrace{C + G + I + (X - M)}_{\text{GDP, expenditure approach}}$$

- COE: Compensation of employees
- GOS: Gross operating surplus
- GMI: Gross mixed income
- $T_{P\&M} - S_{P\&M}$: Taxes minus subsidies on production and import
- C: Consumption by households
- G: Government consumption & investment
- I: Gross private domestic investment
- $X - M$: Export minus Imports of goods and services

The `validate` R package

`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.

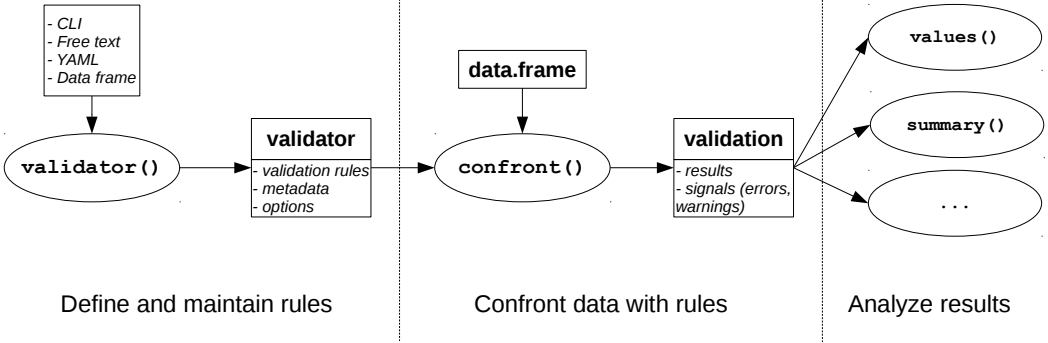
Rules as first-class citizens

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

Validate data

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

The validate package



Reading rules from file

```
### myrulez.R

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

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

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%`

Boolean operations

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

Text search

`grepl`

Functional dependencies (Armstrong)

`city + zipcode ~ streetname`

Refer to the dataset with .

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

SDMX support

```
global_codelist("CL_AGE") # lats version from global SDMX registry
```

```
## [1] "Y" "M" "W" "D" "H"
```

Use SDMX code lists straight in your validation rules

```
Age %in% global_codelist("CL_AGE")
```

Notes

- Needs internet connection
- Uses per-session caching of codelists
- Access any SDMX endpoint with `sdmx_codelist()`
- Get all rules from a DSD with `validator_from_dsd()`

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

```
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
```

```
errors(out)
```

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

Assignment (1)

Preparation

Open the data validation cookbook: data.cleaning.github.io/validate/

Assignment (1)

1. Create a new textfile
2. Define *at least* 10 rules for the `retailers`. Include at least the following checks:
   - `zipcode` is 4 figures and two capitals
   - `id` consists of 4 integers
   - range checks, balance checks
3. Provide rationales in comments
4. Read, and `confront` rules with data
5. Summarize and plot the results.
6. Use `as.data.frame` and `View` to convert and display the results.
7. Make a `plot` of the `validator` object.

If you are familiar with `rmarkdown`: create a data quality report.

More on the `validate` DSL

There are many utilities, including for

- Field format, ranges, completeness, code lists
- Time series (ranges, completeness, aggregates)
- Data in ‘long’ format
- Properties relying on grouped data
- Comparing datasets with respect to a rule set
- import/export
- metadata for rules (e.g. from YAML files)
- ...

Assignment (2)

Preparation

Open the data validation cookbook: data.cleaning.github.io/validate/

In groups (you will be assigned)

- Take 15 minutes to go through the cookbook
- Design 2 multiple-choice (a-b-c) questions about the validate package

Documentation

- Data Validation Cookbook: data.cleaning.github.io/validate/
- MPJ van der Loo, E de Jonge (2021). Data Validation Infrastructure for R. Journal of Statistical Software 1–22 97.
- Statistical Data Cleaning with Applications in R (Wiley, 2018)

Packages

- `validatetools` (Edwin de Jonge): find inconsistencies, redundancies in data validation rules
- `validatesuggest` (Edwin de Jonge): Derive rules from example data