R data validation course Part 1

by Statistics Iceland, for/with our colleagues from:
Armenia, Azerbaijan, Belarus, Georgia, Kazakhstan, Kyrgyzstan,
Moldova, Tajikistan, Turkmenistan, Ukraine, Uzbekistan

March 2021

Focus

- *Rule* based and *advanced* data validation for official statistics, using *R*
- Principles
  - Reproducibility
  - Transparency
  - Flexibility
  - Peer review & Scientific/Statistical knowledge
  - Collaboration

Organization

- Lectures (on **22**, 24, 26 of March 2021, from 9:00 GMT to 12:00 GMT, via interprefy platform)
- Practical sessions (23 and 25 of March 2021, from 9:00 GMT to 11:00 GMT, via TEAMS):
  - students use, adapt and report, based on two template Rmd-files (provided by the instructors, one for each sesssion) and individual datasets,
    similar to the examples shown at:
    https://github.com/violetacln/revaliew/blob/master/rmd_reports/
- Useful sharing at: https://github.com/violetacln/learnRval

Prerequisites

- To have Rstudio installed
- Desirable: basic knowledge of R (data from/into R, data frames)
- R-packages frequently used:

```
# install.packages("tidyverse",
#                  "validate", "validatetools",
#                  "errorlocate", "dcmodify", "simputation",
#                  "DataExplorer", "funModeling",
#                  "caret", "tidymodels",  # optional
#                  dependencies=TRUE)
```

- at least plan for the future to use some version control / git or similar

References

- MPJ van der Loo and E de Jonge (2020). Data Validation Infrastructure for R. Journal of Statistical Software, Accepted for publication. https://arxiv.org/abs/1912.09759
- MPJ van der Loo (2020) The Data Validation Cookbook version 1.0.1. https://data-cleaning.github.io/validate
- the great collection of open code for official statistics, at: https://github.com/SNStatComp/awesome-official-statistics-software
- Russian resource, more general: http://herba.msu.ru/shipunov/software/r/r-ru.htm

Get to know each other

Round of table short discussion? or slido survey?

- what is your field?
- your favorite example of R application?

Part 1:

Reproducible, reviewed, improved

data processing and analysis

with validation, as stages of GSBPM, using *R*

The simplest motivating example which does "everything"!

Before any theoretical considerations, run the following code! (added to a piece adapted from: https://www.r-bloggers.com/2016/03/easy-data-validation-with-the-validate-package/)

```
d <- iris
v <- validate::validator(
        ratio = Sepal.Width > 0.5 * Sepal.Length
      , mean = mean(Sepal.Width) > 0)
result <- validate::confront(d,v)
summary(result)
validate::plot(result)
# may add
DataExplorer::create_report(d) # check your browser!
correlation::correlation(d, bayesian=TRUE)
funModeling::df_status(d)
```

Imagine now that:

you want to do the same with other type of software,

but want to change:

- the data set or
- some rules or
- some exploratory components

1.1. Introduction:

- **motivation**: easy to maintain R – workflows for reproducible data processing and analysis
- validation and *advanced* **validation**: definitions
- open code **repositories** and version control: good practice
- data in R: reminders about **input/output** using files/databases
- data sets: the **examples** used for this course

1.2. Error detection at primary level:

- variable **checks**
- multivariate checks (records)
- statistical checks (basic)
- simple **exploratory** data analysis

1.1.1. **motivation**:

easy to maintain, review, revise
R – workflows for reproducible *trusted* data processing and analysis

Fun Film 1, watch if possible in class, from:
https://www.youtube.com/watch?v=s3JldKoA0zw

Dimensions of reproducibility:

- Computational
- **Empirical**
- Statistical

Main goals:

- review, replicate, confirm, open data analysis

Role of **validation** in this process

The final workflow

Read_data → Read_validation_rules →

Start_logging →

Verify_rule_consistency → Confront_data_rules →

Exploration_data_analysis →

Apply_modifier_rules →

Locate_errors → Impute_corrected_values →

Repeat_confront_and_explore →

Output

Main ingredients

i0. Knowledge of: statistics, subject matter, coding

i1. Data of official statistics - ideally, from a database

i2. Validation rules - ideally, from a database

i3. Several R-packages

1.1.2. validation and *advanced* **validation**

Definition:
whether *or not* a combination of data values belongs to a set of *acceptable* combinations!

If not -> incorrect
If yes -> plausible *

from *Methodology for data validation handbook*,
https://ec.europa.eu/eurostat/cros/system/files/ess_handbook_-_methodology_for_data_validation_v2.0_-_rev2018_0.pdf

Where in the statistical production? Input/Process/Output

Type of work? Automated versus Manual, Reproducible versus non

How? Rule based versus statistical and data analysis

What about? Content (logical, statistical consistency) versus Structure (IT requirements)

Where in the business process (GSBPM stage)?

- Specify
- Design production and data collection
- Build production
- Data collection
- **Data processing**
- **Analysis**
- **Dissemination**
- Evaluation

Validation Levels

Validation level 0: consistency with the expected IT structural requirements

Validation level 1: consistency within the data set

Validation level 2: consistency with other data sets within the same domain and within the same data source

Validation level 3: consistency within the same domain between different data sources

Validation level 4: consistency between separate domains in the same data provider

Validation level 5: consistency with data of other data providers

Eurostat Tools:

- STRUVAL
- CONVAL

Rule based validation -> Languages:

- VTL
- R, Python, ...
- SQL

1.1.3. open code **repositories** and version control

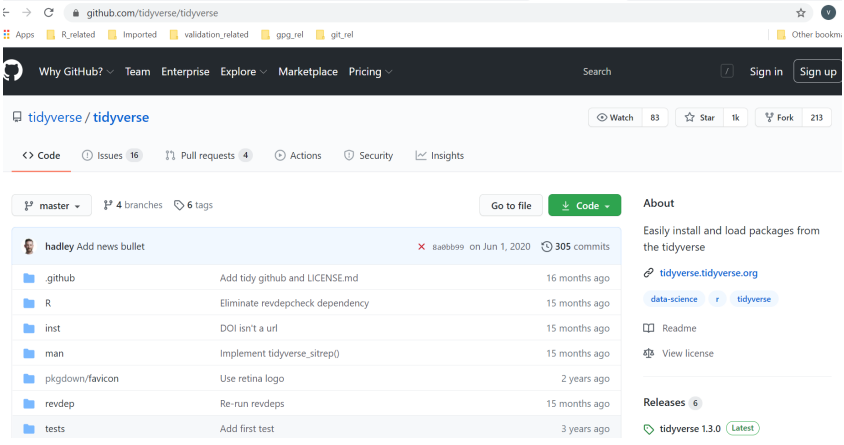
Good practice: learn from software developers! Stages.
Use: *R-Projects* and *packrat* to define the whole ensemble of: code and packages

Why: multiple computers/people/versions/directions for the same project
Main structure: repositories (databases of changes) and working copies

Examples of software: Git, CVS, Subversion, Mercurial, Bazaar

Examples of repositories: Gitlab, Github, Sourceforge, FreeCode, CRAN, CTAN

Visit a page!


github.com/tidyverse/tidyverse
Apps
R_related
Imported
validation_related
gpg_rel
git_rel
Other bookm
Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign up
tidyverse / tidyverse
Watch 83
Star 1k
Fork 213
Code
Issues 16
Pull requests 4
Actions
Security
Insights
master
4 branches
6 tags
Go to file
Code
hadley Add news bullet
8a0bb99 on Jun 1, 2020
305 commits
.github
Add tidy github and LICENSE.md
16 months ago
R
Eliminate revdepcheck dependency
15 months ago
inst
DOI isn't a url
15 months ago
man
Implement tidyverse_sitrep()
15 months ago
pkgdown/favicon
Use retina logo
2 years ago
revdep
Re-run revdeps
15 months ago
tests
Add first test
3 years ago
About
Easily install and load packages from the tidyverse
tidyverse.tidyverse.org
data-science
r
tidyverse
Readme
View license
Releases 6
tidyverse 1.3.0
Latest

1.1.4. Reminders about data **input/output** in/from R

Input/Output data:

- included in R-packages
- from (csv and other types of) files
- from databases
- and more!

Example: Input data from an R-package

```
library(dplyr)
data(storms)
?storms
head(storms,3)

library(validate)
data(package = "validate")
data(SBS2000)
head(SBS2000,3)

data(package = .packages(all.available = TRUE))
```

Example: Input/Output csv(or other...) -files

```
# if needed, one may use "file.choose()" instead of "path_name_file" in:

df00 <- read.csv("path_name_file")

# or using the readr-package of tidyverse:

df01 <- readr::read_csv("path_name_file")

# and OK ... there might be some excel files somewhere...

df0 <- readxl::read_excel("path_name_file")

write.csv(df, "path_and_name_file", row.names = FALSE)
```

Example 1: Input/Output DataBase (if on *windows*)

when using *RODBC*

```
library(RODBC)
#connect
dbhandle <- odbcDriverConnect('driver={SQLServer};
            server=server_name ; database=database_name;
            trusted_connection=true')
#read
df <- sqlQuery(dbhandle, "select * from database_table")

# or
df <- sqlFetch(dbhandle,  database_table_name)

close(dbhandle) # or odbc(close)

#write
sqlSave( dbhandle, data_frame_name, table_name, append=FALSE)
```

using *odbc* and *DBI*:

```
library(odbc)
library(DBI)

con_dev <- dbConnect(odbc(),
                     Driver = "FreeTDS",
                     Server = "myserver",
                     port=XXXX,   #depending on server
                     Database = "mydatabase",
                     UID = "myname",
                     PWD = rstudioapi::askForPassword("Database password")
)

result <- dbSendQuery(con_dev, "SELECT * FROM schema.table")

dfnew <- dbFetch(result)
```

1.1.5. data sets: **examples** used for this course

your favorite data set,
cars, storms, SBS2000, iris, diamonds . . . any reasonable data set from

```
data(package = .packages(all.available = TRUE))
```

1.2. Error detection at primary level

- Rule based validation

– following Mark van der Loo's validate-cookbook (and book on Data editing!)
– using Mark van der Loo's and Edwin de Jonge's R-packages

- and simplest data mining based - validation

following many sources, such as
– R for data science, at https://r4ds.had.co.nz/
– DataExplorer, see https://boxuancui.github.io/DataExplorer/ and on CRAN
– funModeling, see https://blog.datascienceheroes.com/exploratory-data-analysis-data-preparation-with-funmodeling/ and on CRAN

1.2.1. primary rules about independent **variables** and **records**

- Variables:

– type: *is.numeric(), is.character()*
– format: *number_format()*, regex functions
– range / list: *in_range()* and *%in% valid_codes_list*
– missing values: *!is.na*

- Records:

– uniquely identified: *is_unique*(Y_1, Y_2, Y_5, Y_9)
– available, conditionally on some variables: *contains_at_least(keys,by)*
– missing records (e.g. gaps in time series): *in_linear_sequence(series, blocks, begin, end)*

Examples

```
#may combine the is.na() with any() and all() functions
library(validate)
data("SBS2000")
v1 <- validator(!any(is.na(incl.prob)))
v2 <- validator(all(is.na(vat)))
v3 <- validator(is.character(size))
v4 <- validator(is.numeric(turnover))
summary(confront(SBS2000, v1+v2+v3+v4))
```

```r
#library(validate)
#data("SBS2000")
v5 <- validator(TO = turnover >= 0)
v6 <- validator(PR = in_range(incl.prob, min=0, max=1))
summary(confront(SBS2000, v5+v6, lin.ineq.eps=0))
# see rounding errors condition above
```

```r
valid_code_list <- c("sc0", "sc1")
## or read it from a database or a file
v7 <- validator(size %in% valid_codes_list)
summary(confront(SBS2000, v7))

#simple testing of number format
number_format("12,123", min_dig=2, dec=",")

##more general format of fields: use grepl() or
##use field_format() with regex
v8 <- validator(
          grepl("^sc[0-9]$", size)
        , field_format(id, "^RET\\d{2}$" , type="regex") )
summary(confront(SBS2000, v8))
```

```r
data("samplonomy")
vr1 <- validator(is_unique(region, period, measure))
summary(confront(samplonomy, vr1)) #[1:7] # try with and without it

vr2 <- validator(
      contains_at_least(
      keys = data.frame(period = as.character(2014:2019))
    , by=list(region, measure) ))
summary(confront(samplonomy, vr2))[1:7]

vr3 <- validator(
          in_linear_sequence(period
            , by = list(region, freq, measure)))
summary( confront(samplonomy, vr3))[1:7]
```

1.2.2. **multivariate** primary rules

- Multivariate combinations (balance)

$$F(Y_1, Y_2, ...) \geq 0$$

- Conditional restrictions

$$Cond_1(Y_1, Y_2, ...) \rightarrow Cond_2(Y_1, Y_2, ...))$$

- Forbidden multivariate combinations (non-existing patterns)

$$\nexists(Y_1^a, Y_2^b ...)$$

- Missing values, by record

Templates

```
rule_ex1 <- validator(F>= 0)
```

```
rule_ex2 <- validator(if (Cond1) Cond2)
```

```
rule_ex3 <- validator( does_not_contain(glob(forbidden_keys))) )
```

```
is_complete($Y_1$, ...)  # result: one T/F per line
all_complete(Y)          # result: one T/F
```

Exercise:
propose/run F, Cond1, Cond2, forbidden_keys for SBS2000 or samplonomy data

Solution

https://cran.r-project.org/web/packages/validate/vignettes/cookbook.html#4_Multivariate_checks

1.2.3. **statistical** primary rules

Key functions for defining the rules:

- Conditions on statistical summary measures: use any *base R stat* functions
- Rules for aggregated data: *rule(Cond,by)*
- Hierarchical aggregation structures: *hierarchy()*

Templates

```
vs1 <- validator(F(statFun(data, ...)) >=0)  # or other eq/ineq

vs2 <- validator(F(Y1, do_by(data, by=Y2, fun=statFun, ...))>=0)

vha <- validator(hierarchy(YtoAgg, Ygroup, hierarchy,by=Y,aggregator, ...))

#?hierarchy()
```

Exercise:
propose/run F, Cond1, Cond2, forbidden_keys for SBS2000 or samplonomy data

Solution

https://cran.r-project.org/web/packages/validate/vignettes/cookbook.html#5_Statistical_checks

1.2.4. simple **exploratory** data analysis : *to run!*

```
df <- SBS2000   # or iris, or anything you like!
##variable types
dnames <-names( DataExplorer::split_columns(df)$discrete)
cnames <- names( DataExplorer::split_columns(df)$continuous)
##short overview
  Hmisc::describe(df)
##data univariate plots
  DataExplorer::plot_intro(df)
  DataExplorer::plot_missing(df)
  funModeling::freq(df)  #DataExplorer::plot_bar(df, maxcat=450)
  funModeling::plot_num(df) #DataExplorer::plot_histogram(df)
##qq-plots, overall and conditional
  DataExplorer::plot_qq(df)
  DataExplorer::plot_qq(df, by="size")   ## Species, if use df <- iris
```

To follow

Part 2: Workflow for *error detection*
based on expert rules and statistical analysis

Part 3: Workflow for *analysis and dissemination of results*
with validation, as stages of GSBPM, using R

Assignment 1 (to be done mostly during class):

Download the Rmarkdown file *RvalPractice1.Rmd*.

Run, for a data set of your choice, each chunk of R-code included in the file.

Knit the whole file for producing a report in your favorite format: Word, html, pdf.

Thank you!

Discussion