Monitoring data cleaning processes

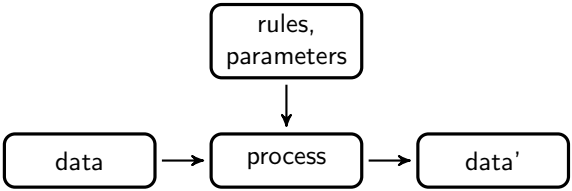
Mark van der Loo, Statistics Netherlands

CBS, Department of Methodology

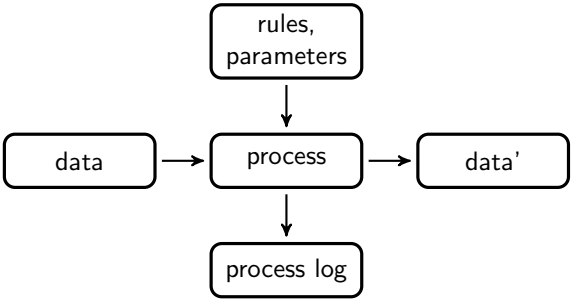
Complutense University of Madrid, Spring 2019

UNIVERSIDAD COMPLUTENSE MADRID

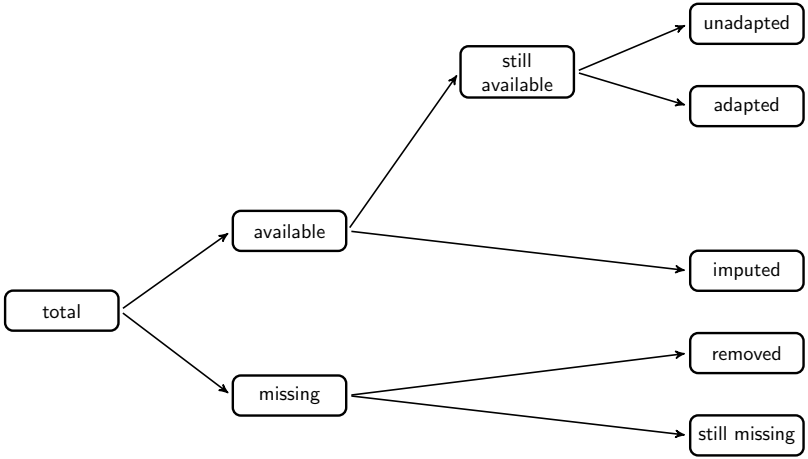
Process overview



Process overview

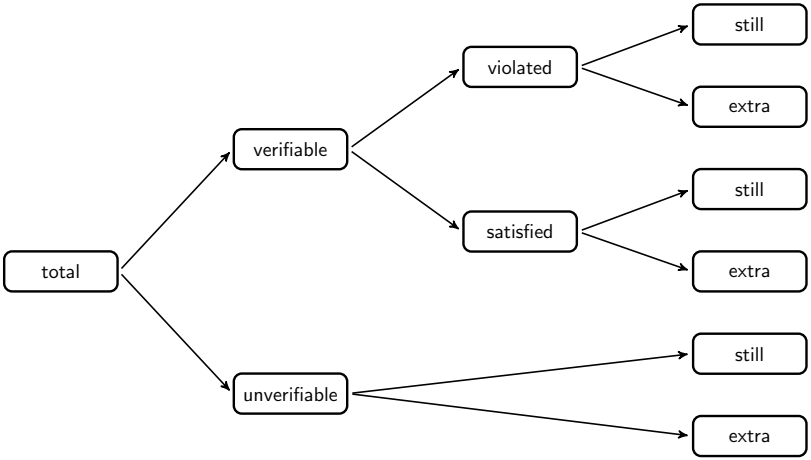


How to measure changes? 1. Track cells




```
validate::cells()
```

How to measure changes? 2. Track validations




```
validate::compare()
```

How to measure changes between `data` and `data'`?

Many ways

- List every change (record, variable, old, new) ('diff')
- Count differences in cells, or validation
- ⋮
- Note if something has changed (`TRUE`/`FALSE`)

Needs

Logging framework

- Supporting any type of comparison of `data` and `data'`
- Supporting any kind of transformation between `data` and `data'`
- Without demanding changes in the transforming functions
- That does not get in the way of the user

Logging framework

Idea

- A data cleaning procedure is implemented as a sequence of expressions (a script).
- These expressions are *composed* into a programe when you run the script (`source()`)
- To obtain a logging framework that is not intrusive for the user, we can *change the way expressions are composed*.

The `lumberjack` package: preparation

```
dat <- read.csv("SBS2000.csv", stringsAsFactors = FALSE)
head(dat,3)
```

```
##      id size incl.prob staff turnover other.rev total.rev staff.costs
## 1 RET01  sc0      0.02    75       NA        NA      1130          NA
## 2 RET02  sc3      0.14     9     1607        NA      1607         131
## 3 RET03  sc3      0.14    NA     6886       -33      6919         324
##   total.costs profit vat
## 1       18915  20045  NA
## 2        1544     63  NA
## 3        6493    426  NA
```

```
library(validate)
rules <- validator(.file="ruleset.R")

library(lumberjack)
logger <- cellwise$new(key="id")
```

The lumberjack package: clean up

```
dat %L>%
  lumberjack::start_log(logger) %L>%
  errorlocate::replace_errors(rules) %L>%
  rspa::tag_missing() %L>%
  simputation::impute_rhd(. ~ 1, backend="VIM") %L>%
  rspa::match_restrictions(rules) %L>%
  lumberjack::dump_log() -> dat_out
```

```
## Dumped a log at cellwise.csv
```

Read the log:

```
read.csv("cellwise.csv") %L>% head(3)
```

```
##   step                    time                          expression   key
## 1    1 2019-05-23 09:36:15 CEST errorlocate::replace_errors(rules) RET01
## 2    1 2019-05-23 09:36:15 CEST errorlocate::replace_errors(rules) RET03
## 3    1 2019-05-23 09:36:15 CEST errorlocate::replace_errors(rules) RET03
##    variable   old new
## 1    profit 20045  NA
## 2 other.rev   -33  NA
## 3  turnover  6886  NA
```

Background

The pipe is a sort of *function composition* operator.

```
# Pseudocode:
`%>%` <- function(x, fun){
  return( fun(x) )
}
```

The `lumberjack` does some extra things:

```
# Pseudocode
%L>% <- function(x, fun){
  y <- fun(x)
  if ( logger_attached_to(x) ){
    logger <- get_logger(x)
    logger$add_difference(x,y)
  }
  return(y)
}
```