

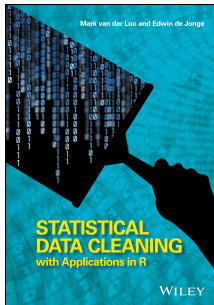
# Monitoring Changes In Data

Mark van der Loo and Edwin de Jonge

Statistics Netherlands Research & Development

Twitter: @markvdloo @edwindjonge

Tokyo | Institute of Statistical Mathematics | 2020



@markvdloo @edwindjonge #datacleaning

Try the code

03input/monitoring.R



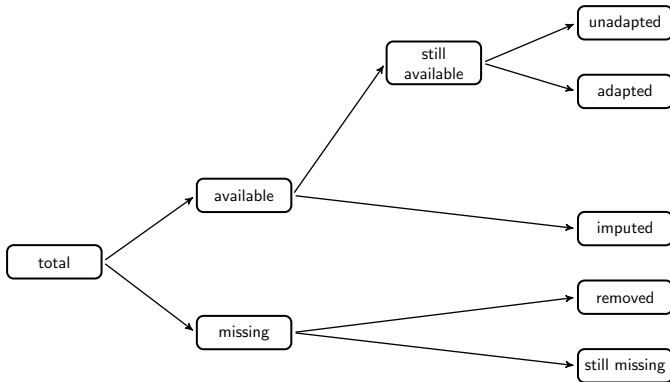
# How to monitor changes

**It depends . . .**

- Cell-by-cell changes?
- Count changes?
- Count changes in satisfying rules?
- Measure changes in aggregates?



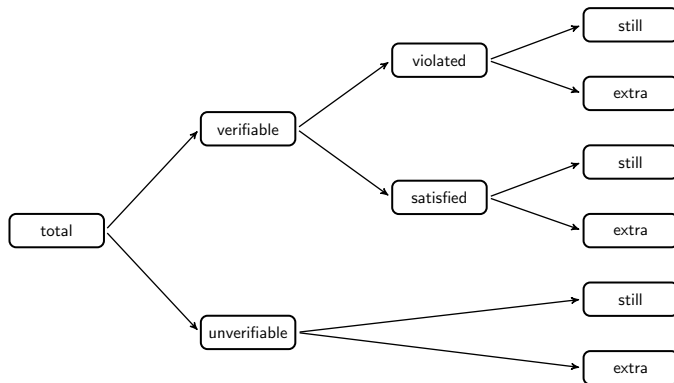
# Decomposing the number of changes in cells



```
validate::cells(start=dataset1, step1=dataset2, step3=dataset3, ...)
```



# Decomposing the number of changes in validation results



```
validate::compare(rules, start=dataset1, step1=dataset2, ...)
```



# Assignment

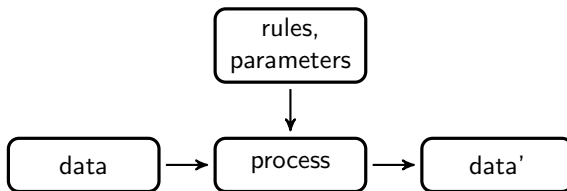
1. Read all the versions of datasets created today
2. Create plots showing the progress in the cell counts and rule counts methods.



## More on monitoring

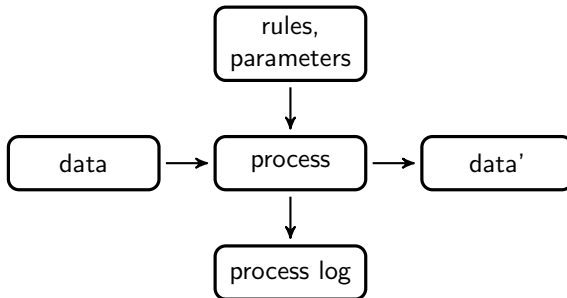


# Process overview

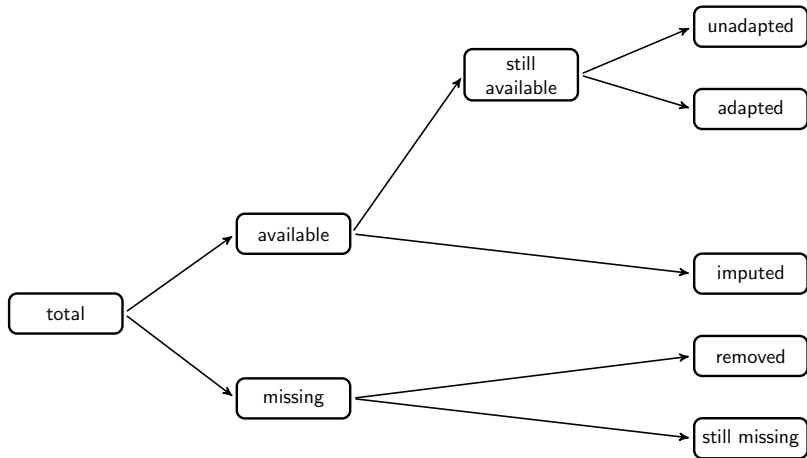




# Process overview



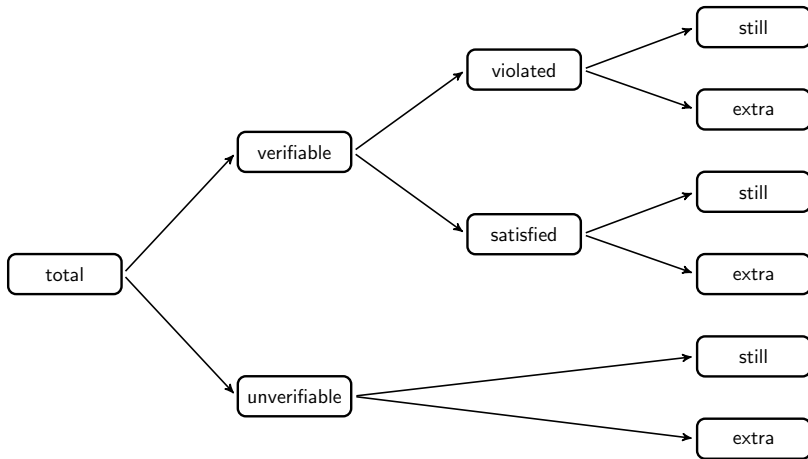
# How to measure changes? 1. Track cells



`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 program 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>%  
  simulation::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 2020-02-14 20:08:47 CET errorlocate::replace_errors(rules) RET01
## 2      1 2020-02-14 20:08:47 CET errorlocate::replace_errors(rules) RET03
## 3      1 2020-02-14 20:08:47 CET errorlocate::replace_errors(rules) RET03
##      variable  old new
## 1 total.rev 1130  NA
## 2 other.rev  -33  NA
## 3   profit   426  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)  
}
```

# But there is more

## As of lumberjack 1.0.0

1. Add the following line to an existing R script, e.g. cleanup.R

```
start_log(SBS2000, logger=cellwise$new(key="id"))
```

2. Run the file from the lumberjack package.

```
library(lumberjack)  
lumberjack::run("cleanup.R")
```

*and everything is done for you.*

