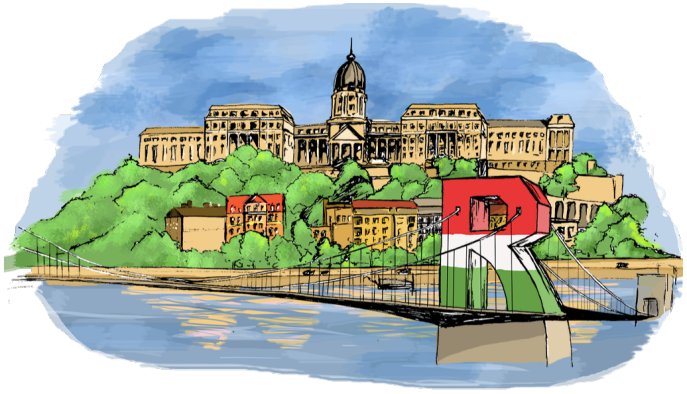
Logging changes in data with lumberjack

Mark van der Loo, Statistics Netherlands

@markvdloo | github.com/markvanderloo



The next 15 minutes

- Motivation
- How to do it
- Why it works
- Examples

Example

```
# 'retailers' dataset from the 'validate' package
head(dat,3)
```

```
##      Id turnover other.rev total.rev
## 1 RET01       NA        NA      1130
## 2 RET02     1607        NA      1607
## 3 RET03     6886       -33      6919
```

Computing task

Estimate mean(other.rev)/mean(turnover)

Clean up and compute result

```
library(dcmodify); library(simputation); library(dplyr)
dat %>%
  modify_so(if (other.rev < 0)
    other.rev <- -1*other.rev) %>%
  impute_const(other.rev ~ 0) %>%
  impute_rlm(turnover ~ total.rev) %>%
  impute_median(turnover ~ 1) %>%
  summarize(result = mean(other.rev)/mean(turnover))
```

```
##       result
## 1 0.08844255
```

Questions

We are using a pretty complex estimator

$$\text{Estimate} = f(\text{input}) = (\text{mean} \circ \text{impute} \circ \text{clean})(\text{input})$$

How important is each step for the final result?

- How many cells are altered by each step of the cleaning process?
- How do e.g. the column means change during the cleaning?
- How about the variance?
- ...

Logging changes in data

Wish list

- Working for all data in/data out functions
- User-definable logging
- Near-zero change in workflow

Using `lumberjack`

```
out <- dat %L>%
  # Tag data for logging; use lumberjack
  start_log( cellwise$new(key="Id") ) %L>%
  # Do your cleanup
  modify_so(if(other.rev < 0) other.rev <- -1*other.rev) %L
  impute_rlm(turnover ~ total.rev) %L>%
  impute_median(turnover ~ 1) %L>%
  impute_const(other.rev ~ 0) %L>%
  # Dump log to file
  dump_log() %L>%
  # continue with analyses
  summarize(result=mean(other.rev)/mean(turnover))
```

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


Check the logging info

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

```
##   step                      time
## 1    1 2018-05-16 10:30:42 CEST
## 2    2 2018-05-16 10:30:42 CEST
## 3    2 2018-05-16 10:30:42 CEST
##                                                    expression   key
## 1 modify_so(if (other.rev < 0) other.rev <- -1 * other.rev) RET03
## 2                          impute_rlm(turnover ~ total.rev) RET01
## 3                          impute_rlm(turnover ~ total.rev) RET05
##    variable old      new
## 1 other.rev -33   33.000
## 2  turnover  NA 1125.608
## 3  turnover  NA 5597.627
```

How it works

`start_log(data, logger)`

Attach a logger object to the data. The data 'wants' to be logged.

Lumberjack: `%L>%`

Check if the data has a logger, if so: use it.

`dump_log(data, stop=TRUE)`

Dump logging info, remove logger (by default)

The lumberjack operator

In stead of this:

```
# not-a-pipe pseudocode
`%>%` <- function(x, f){
  f(x)
}
```

Do this:

```
# lumberjack pseudocode
`%L>%` <- function(x, f){
  input  <- data
  output <- f(x)
  if ( x wants to be logged )
     store logging info based on input and/or output
  output
}
```


Some loggers

In `lumberjack`

- `simple`: test if input is identical to output.
- `filedump`: dump the whole dataset after each operation
- `expression_logger`: log the result of user-defined expressions

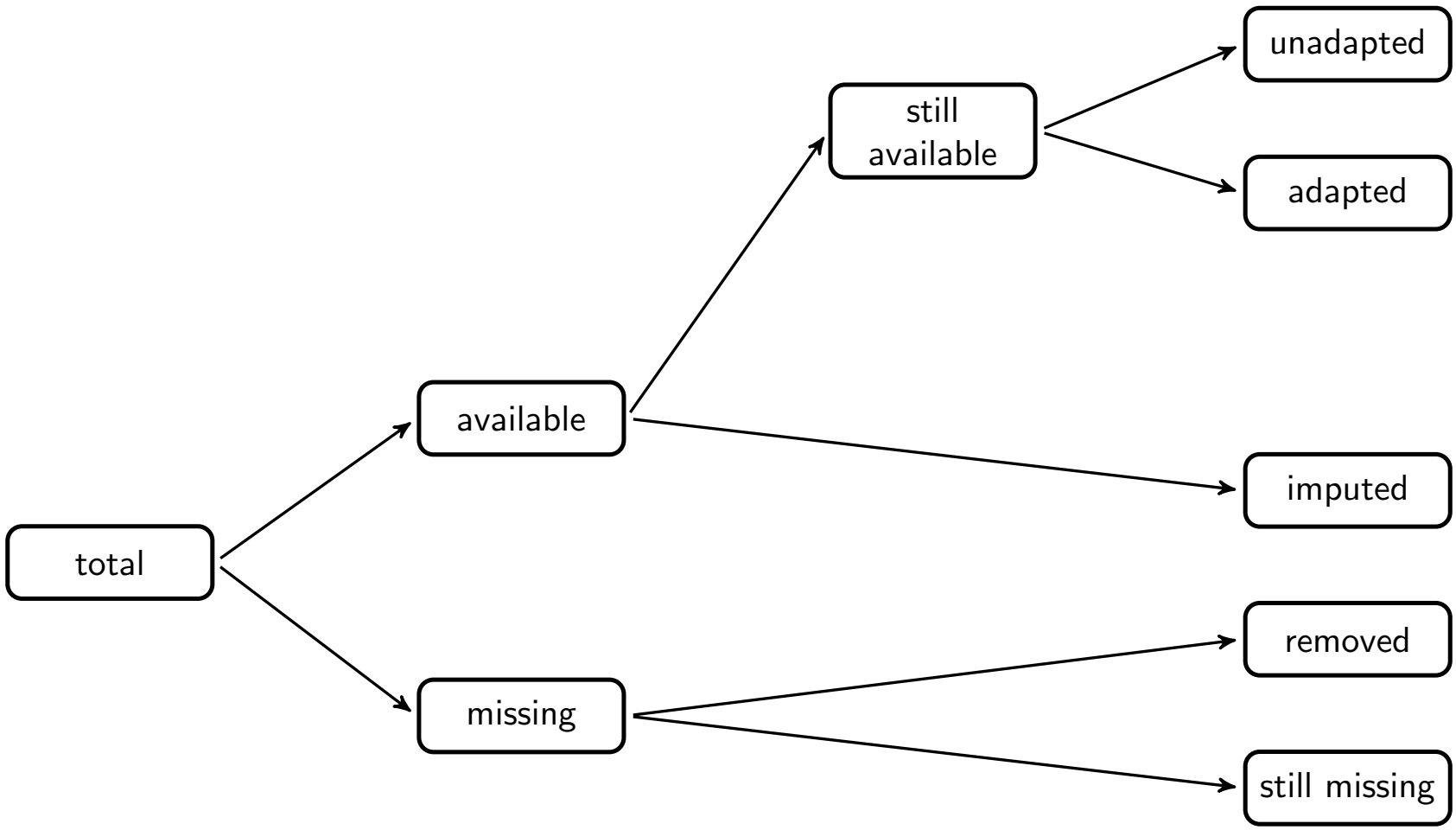
In `validate`

- `lbj_cells`: Summary of cell changes (see next slide)
- `lbj_rules`: Summary of changes in validation rule compliance

In `daff`

- `lbj_daff`: Create a data diff file.

The `lbj_cells` logger: count cells changed



Van der loo and de jonge (2018)

The `lbj_cells` logger

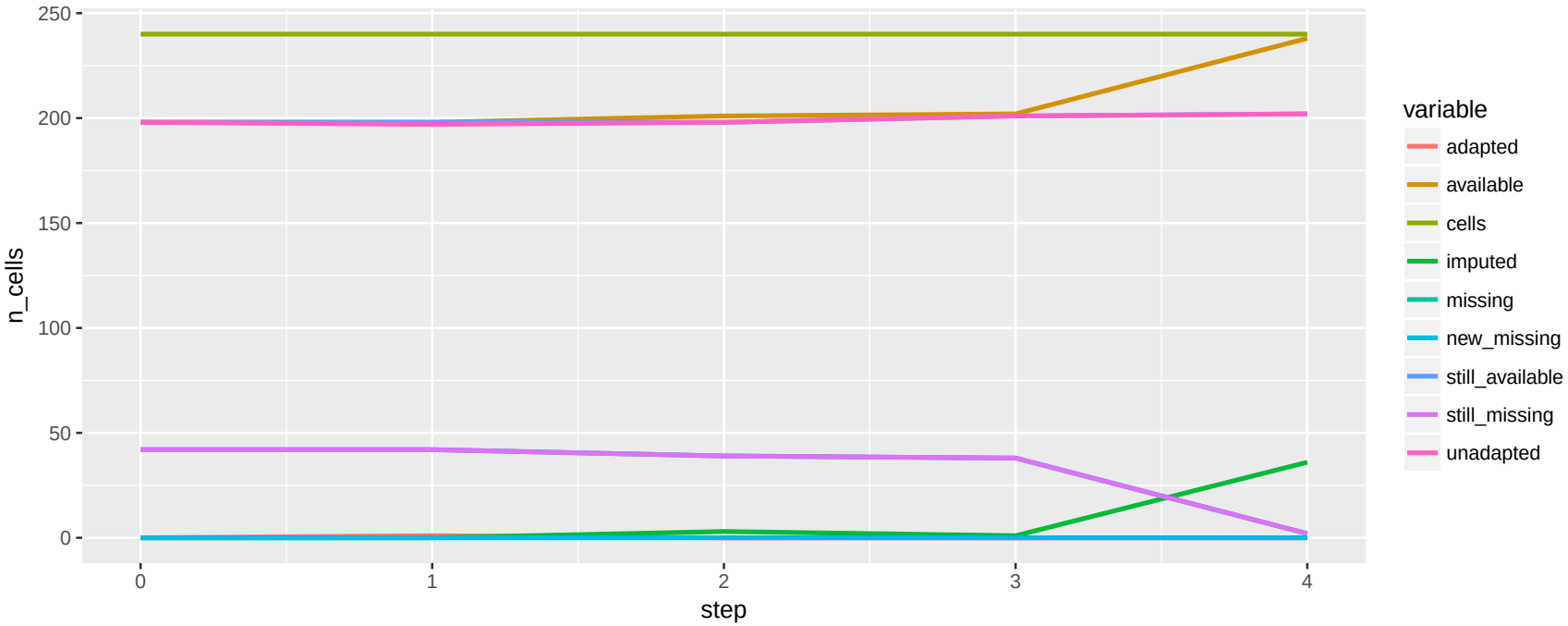
```
dat %L>%
  start_log(validate::lbj_cells()) %L>%
  ...
  dump_log() %L>%
  summarize(result=mean(other.rev)/mean(turnover))
```

```
## Dumped a log at /home/mark/projects/tex/eRum2018/pres/ce
```

```
##       result
## 1 0.08844255
```


The `lbj_cells` logger

```
read.csv("cells.csv") %>%
  gather(variable, n_cells,-step,-time,-expression) %>%
  ggplot(aes(x=step,y=n_cells,color=variable)) + geom_line(size=1)
```



Log any list of expressions (version $\geq$ 0.3.0)

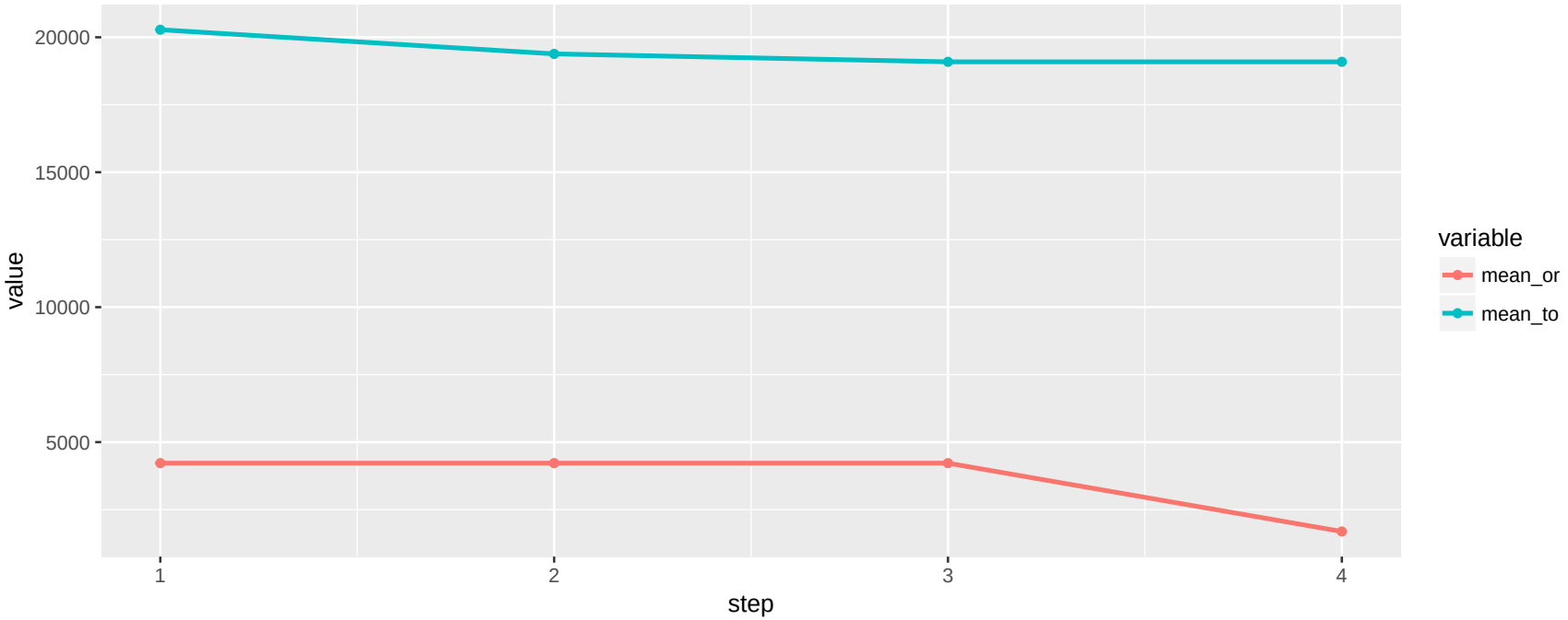
```
logger <- expression_logger$new(
  mean_or = mean(other.rev, na.rm=TRUE)
  , mean_to = mean(turnover, na.rm=TRUE)
)
```

```
dat %L>%
  start_log(logger) %L>%
  ...
  dump_log() %L>%
  summarize(result=mean(other.rev)/mean(turnover))
```

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


Log any list of expressions (version $\geq$ 0.3.0)

```
read.csv("expression_log.csv") %>%
    gather(variable, value, -expression, -step) %>%
    ggplot(aes(x=step,y=value, col=variable)) +
    geom_line(size=1) + geom_point()
```



Logger API: create your own loggers

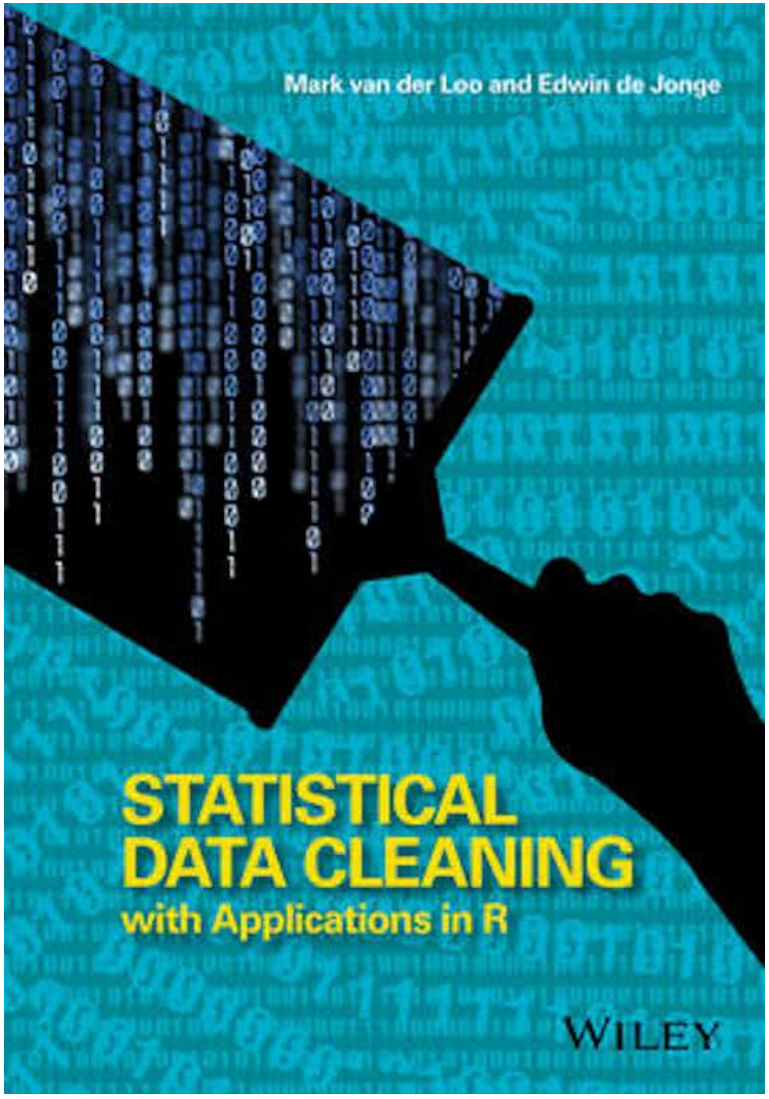
A logger is a `R6` or `RC` object with at least:

- `$add(meta, input, output)`
  - `meta`: `list(expr, src)` (expression and source)
  - `input`: input data
  - `output`: output data
- `$dump()` This function dumps the logged information

For package authors

You can `Extend` the `lumberjack` pkg (see vignette).

More information



SDCR

M. van der Loo and E. de Jonge (2018) *Statistical Data Cleaning with applications in R* Wiley, Inc.

lumberjack 0.2.0

- Available on CRAN

Vignettes

- Getting started
- Creating loggers