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)
library(dcmmodify); 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

```r
out <- dat

# Tag data for logging; use lumberjack
start_log( cellwise$new(key="Id") )

# Do your cleanup
modify_so(if(other.rev < 0) other.rev <- -1*other.rev)

impute_rlm(turnover ~ total.rev)

impute_median(turnover ~ 1)

impute_const(other.rev ~ 0)

# Dump log to file
dump_log()

# continue with analyses
summarize(result=mean(other.rev)/mean(turnover))

## Dumped a log at cellwise.csv
```
Check the logging info

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

<table>
<thead>
<tr>
<th>step</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018-05-16 10:30:42 CEST</td>
</tr>
<tr>
<td>2</td>
<td>2018-05-16 10:30:42 CEST</td>
</tr>
<tr>
<td>3</td>
<td>2018-05-16 10:30:42 CEST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>expression</th>
<th>key</th>
</tr>
</thead>
<tbody>
<tr>
<td>modify_so(if (other.rev &lt; 0) other.rev &lt;- -1 * other.rev)</td>
<td>RET03</td>
</tr>
<tr>
<td>impute_rlm(turnover ~ total.rev)</td>
<td>RET01</td>
</tr>
<tr>
<td>impute_rlm(turnover ~ total.rev)</td>
<td>RET05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>variable</th>
<th>old</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>other.rev</td>
<td>-33</td>
<td>33.000</td>
</tr>
<tr>
<td>turnover</td>
<td>NA</td>
<td>1125.608</td>
</tr>
<tr>
<td>turnover</td>
<td>NA</td>
<td>5597.627</td>
</tr>
</tbody>
</table>
How it works

\texttt{start\_log(data, logger)}

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

\textbf{Lumberjack:} \texttt{\%L>}%

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

\texttt{dump\_log(data, stop=TRUE)}

Dump logging info, remove logger (by default)
The lumberjack operator

In stead of this:

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

Do this:

```r
# 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
}
```

For package authors
You can Enhance the lumberjack package by providing a logger. (see 'Extending lumberjack' vignette)
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 **lbg_cells** logger: count cells changed

- **unadapted**
- **adapted**
- **imputed**
- **removed**
- **still missing**

Van der loo and de jonge (2018)
The `lbj_cells` logger

dat

```
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/cells.csv

## result

```
## 1 0.08844255
```
The `lbj_cells` logger

```r
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 $\geq 0.3.0$)

```r
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 $\geq 0.3.0$)

```r
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 \texttt{R6} or \texttt{RC} object with at least:

- \texttt{$\text{add(meta, input, output)}$}
  - \texttt{meta: list(expr, src)} (expression and source)
  - \texttt{input: input data}
  - \texttt{output: output data}

- \texttt{$\text{dump()}$} This function dumps the logged information

For package authors

You can Extend the lumberjack pkg (see vignette).
More information

SDCR

lumberjack 0.2.0

▶ Available on CRAN

Vignettes

▶ Getting started
▶ Creating loggers