Daff

Edwin de Jonge (@edwindjonge), Gregory Warnes

UseR!2017, July 6, 2017

Who am I (Edwin)

- Statistical consultant / methodologist
- Statistics Netherlands (CBS): all official statistics of NL
- twitter: @edwindjonge
- github: `https://github.com/edwindj`

Stepping in for Greg Warnes

What is daff?

Short version

Daff is a diff for $data.frames$

- Detect changes: **`diff_data`**, **`differs_from`**
- Store and restore diff: **`write_diff`**, **`read_diff`**
- Patch updated data: **`patch_data`**, **`merge_data`**
- Render a diff: **`render_diff`**

Diff?

`diff`

- Command-line utility for comparing text files.
- Used in all source code version control systems.
- `diff` checks *lines*:
  - which lines have changed, removed or added.

`daff`

- Utility for comparing tables
- `daff` compares *records* and *columns*:
  - which values changed
  - rows added/removed
  - columns added/removed.

Why o why?

- Support version control of `data.frame`.
- To log changes of data.
- e.g. subsequent steps in data process: what did these steps do?
- Supports monitoring external data changes beyond your control.
- Makes even manual editing reproducible (*Note: manual editing is really bad*).

Use case: data update

Raw data update

- You have build a nice R script:
  - takes raw data as input
  - removes errors
  - fits a model
  - calculates output

You get an updated raw data file: what are the changes?

- Did output change?
- Also input: should the script be adapted, e.g. data cleaning?

Use case: manual editing

- **bad practice**, but it happens: e.g. implausible values, manual data correction.

Manual editing

- Compare the input and output
- Make the manual step *reproducible*: all process steps can be re-executed:
  - data + changes = new data
  - in `diff` parlor: version1 + patch = version2

Daff protocol

Highlighter diff format

- highlighter diff format: `http://dataprotocols.org/tabular-diff-format`
- diff protocol for tabular data.
- shows rows/columns that changed.
- supports patching data.
- format itself is in tabular format (nifty!)
- can be stored in txt (csv) or db.

Detecting changes

`daff` detects the following changes:

- changing a value.
- adding a row.
- removing a row.
- adding a column.
- removing a column.
- changing type of a column (partially)
  - `daff` supports it, but highlighter format not

diff_data: value was changed

```r
library(daff)
x         <- data.frame(A=1, B=  1)
x_changed <- data.frame(A=1, B=100)
patch <- diff_data(x, x_changed)
print(patch)
```

```
## Daff Comparison: 'x' vs. 'x_changed'
##    A B
## -> 1 1->100
```

patch_data: apply the change

```
x
```

```
##   A B
## 1 1 1
```

```
patch_data(x, patch)
```

```
##   A   B
## 1 1 100
```

replay‘ the change on original data:

- when org. data is updated, same procedure!

diff_data: row was added

```r
x         <- data.frame(A=1  , B=1)
x_changed <- data.frame(A=1:2, B=1:2)
diff_data(x,x_changed)
```

```
## Daff Comparison: 'x' vs. 'x_changed'
##      A B
##      1 1
## +++ 2 2
```

diff_data: row was deleted

```r
x         <- data.frame(A=1:2, B=1:2)
x_changed <- data.frame(A=1  , B=1)
diff_data(x,x_changed)
```

```
## Daff Comparison: 'x' vs. 'x_changed'
##     A B
##     1 1
## --- 2 2
```

diff_data: column was added

```r
x         <- data.frame(A=1, B=1)
x_changed <- data.frame(A=1, B=1, C=1)
diff_data(x,x_changed)
```

```
## Daff Comparison: 'x' vs. 'x_changed'
##         +++
## @@ A B C
## +  1 1 1
```

diff_data: column was removed

```r
x         <- data.frame(A=1, B=1, C=1)
x_changed <- data.frame(A=1, B=1)
diff_data(x,x_changed)
```

```
## Daff Comparison: 'x' vs. 'x_changed'
##         ---
## @@ A B C
```

diff_data options

```
diff_data( data_ref, data
         , always_show_header        = TRUE
         , always_show_order         = FALSE
         , columns_to_ignore         = c()
         , count_like_a_spreadsheet = TRUE
         , ids                       = c()
         , ignore_whitespace         = FALSE
         , never_show_order          = FALSE
         , ordered                   = TRUE
         , ...
         )
```

differs_from

- Pipe-friendly version of `diff_data`

```
x_changed %>%
  differs_from(x)

# same as

diff_data(x, x_changed)
```

Merging

- Combine two derived `data.frames` from a common parent.

```r
x   <- data.frame(A =   1, B=  1)
# two changes were made in parallel
x_a <- data.frame(A = 100, B=  1)
x_b <- data.frame(A =   1, B=100)
merge_data(x, x_a, x_b)
```

```
##     A   B
## 1 100 100
```

Reading and writing table diffs

```r
x          <- data.frame(A = 1, B =   1)
x_changed <- data.frame(A = 1, B = 100)
diff <- diff_data(x, x_changed)
write_diff(diff, "diff.csv")
diff2 <- read_diff("diff.csv")
patch_data(x, diff2)
```

```
##   A   B
## 1 1 100
```

Render diff

```
x         <- data.frame(A = 1:2, B = 1:2)
x_changed <- data.frame(         B = 2  , C = 1)

x_changed %>%
  differs_from(x) %>%
  render_diff(use.DataTable=FALSE)
```

‘x’ vs. ‘x_changed’

2017-07-06 00:42:16

	#	Modified	Reordered	Deleted	Added
Rows	2 → 1	0	0	1	0
Columns	2	0	0	1	1

!	---		+++
@@	**A**	**B**	**C**
---	1	1	null
+	2	2	1

Implementation

- Wraps the excellent library `daff` javascript, by Paul Fitzpatrick (@fitzyfitzyfitzy).
- library actually written in Haxe, which compiles to js, python, C++
- Uses R package `V8` to run javascript (Thanks Jeroen Ooms!)

Thank you for your attention!

Interested?

```
install.packages("daff")`
```

or visit:

- http://github.com/edwindj/daff