#+name: author-list #+header: :var authors=authorlist #+header: :var add-authors=additional-authors #+header: :results latex #+header: :exports results

Using Emacs Org-mode to Create Reproducible Research

[Demo][*]

Karl Voit[†]
Institute for Software Technology
Graz University of Technology
Austria
Karl.Voit@IST.TUGraz.at

Thomas S. Dye[‡]
Thomas S. Dye & Colleagues
735 Bishop St, Suite 315
Honolulu
tsd@tsdye.com

ABSTRACT

One important aspect of open science is the ability to reproduce results using the published data set. For this purpose it is crucial to use similar methods and tools as the original author producing the same result set. Reproducible research is a movement that tries to bridge this gap: within one single set of data one can not only find the raw data but also the methods and tools to process the data. The ultimate discipline is to complete this cycle from the raw data up to the presentation in the derived paper. This paper demonstrates using a simple example how to combine raw data, scripts of various languages, and the describing text of a paper in one single file.
#+name: ACM-categories #+header: :var c=categories #+header: :results latex #+header: :exports results

Categories and Subject Descriptors

I.7.1 [**DOCUMENT AND TEXT PROCESSING**]: Document and Text Editing—*Emacs*; H.4.1 [**INFORMATION SYSTEMS APPLICATIONS**]: Office Automation—*Word processing*; D.2.3 [**SOFTWARE ENGINEERING**]: Coding Tools and Techniques; I.7.1 [**DOCUMENT AND TEXT PROCESSING**]: Document Preparation; I.7.4 [**DOCUMENT AND TEXT PROCESSING**]: Electronic Publishing; D.4.9 [**OPERATING SYSTEMS**]: Systems Programs and Utilities; E.2 [**DATA STORAGE REPRESENTATIONS**]: Linked representations

General Terms

Keywords

Open Science, Reproducible Research, Org-mode, Emacs, Tools

1. EMACS ORG-MODE

FIXXME: Add references [2]

[1]

[3]

[4]

1.1 Formal Experiment

In [5] the authors describe a formal experiment conducted with 18 test persons in the field of information retrieval. The original data set is available online[1]. This paper here demonstrates FIXXMEFIXXME: add introduction text

1.1.1 Reading in refinding tagstore values

Reading in raw data related to seconds per task from CSV file:

The following shell commands reads in a CSV file, removes all values before the character ";" (thus removing all values related to number of mouse clicks), removes all incomplete lines (containing the string "TC"), and removes the header line as well (using the `tail` command):

```
sed 's/.*;//' refinding_tagstore.csv | \
   grep -v "TC" | \
   tail -n +2
```

5.7	3.8	4.3	2.4	4.3	3.2
6.0	2.9	4.3	4.6	4.4	3.3
5.4	3.2	6.1	6.5	5.7	4.4
4.3	15.7	6.2	4.9	3.1	3.0
9.7	3.7	3.0	3.9	2.7	8.6
21.8	2.6	11.4	3.0	17.1	5.7
6.6	5.6	5.0	4.1	4.5	2.0
7.0	2.6	10.0	3.7	9.5	3.0
4.8	2.5	4.3	2.3	1.8	3.9
2.4	2.7	7.1	6.2	3.8	5.1
3.7	3.9	7.4	2.0	3.1	7.2
5.5	5.5	8.3	11.4	3.2	7.6
28.3	4.0	3.9	2.0	2.4	4.3
5.0	5.6	6.0	14.2	2.0	6.6
6.7	5.6	7.2	12.3	5.1	6.6

[*] The full source code of this paper is available on github `https://github.com/novoid/orgmode-iKNOW2012`

[†]

[‡]

[1] https://github.com/novoid/2011-01-tagstore-formal-experiment

1.1.2 Generating mean values

In the next step, the mean values per test person will be calculated using the programming language Python:

```
import numpy
return [round(numpy.average(row),2) for row in mytable]
```

This time, the (long) output list of mean values is being suppressed for layout purposes.

1.1.3 Sorting values

```
echo ${myvalues} | sed 's/ /\n/g' | sort -nr
```

#+RESULTS: TS-sort-mean-values

1.1.4 Process folder values

```
sed 's/.*;//' refinding_folders.csv | \
   grep -v "TC" | \
   tail -n +2
```

#+RESULTS: F-time-per-task

```
 6.7  5.4   2.4   3.9  3.6  3.8
 5.4  3.1   3.4   3.5  3.3  3.6
 6.5  6.6   4.0   4.4  4.0  5.1
 3.0  3.3   3.7   7.1  2.8  4.3
 6.6  3.6  10.3   4.6  5.4  3.7
 2.7  3.2   9.4  18.0  4.7  3.8
 7.0  3.7   8.1   4.9  5.2  5.2
34.1  2.8   8.9   8.9  3.1  8.3
 4.0  2.9   3.6   5.7  5.0  5.5
 4.8  1.4   3.5   3.5  3.3  1.9
42.9  1.9  12.3   5.8  7.6  3.4
 7.0  5.2   5.0   3.8  5.1  4.2
19.3  1.6  11.9   7.0  3.9  4.0
 6.6  6.6   4.6   7.5  3.8  5.2
 6.0  3.2   5.1   4.4  5.9  4.0
 4.6  1.6   3.4   4.1  4.4  3.8
 7.1  4.5   7.0   7.6  5.5  7.5
```

```
import numpy
return [round(numpy.average(row),2) for row in mytable]
```

```
echo ${myvalues} | sed 's/ /\n/g' | sort -nr
```

#+RESULTS: F-sort-mean-values

1.1.5 Plotting data

```
png('my_boxplot_data.png')
mFdata=c(4.3, 3.72, 5.1, 4.03, 5.7, 6.97, 5.68, 11.02, 4.45, 3.07, 12.32, 5.05, 7.95, 5.72, 4.77, 3.65, 6.53)
mTSdata=c(3.95, 4.25, 5.22, 6.2, 5.27, 10.27, 4.63, 5.97, 3.27, 4.55, 4.55, 6.92, 7.48, 6.57, 7.25)
#par(mai=c(0.8,0.8,0,0), omd=c(0,0.5,0,1))
# bot, lef, top, rig
boxplot( list(mTSdata, mFdata),
    names=c("tagstore", "folders"),
    xlab="Task Times", ylab="Seconds",
    pars = list(boxwex = 0.3, staplewex = 0.5,
    boxfill="lightblue"))
```

#+RESULTS: draw-histogram

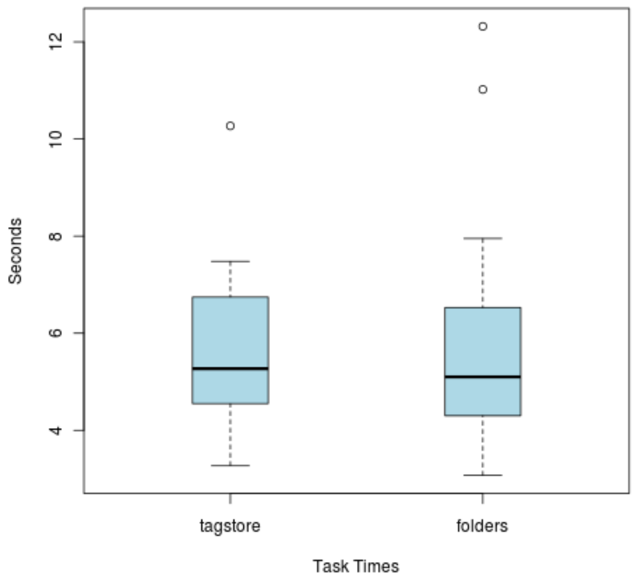

Figure 1: Comparison of the two task conditions for re-finding: tagstore and folders. There is no significant difference between the two conditions.

```
import numpy as np
import matplotlib.pyplot as plt

#n, bins, patches = plt.hist(myvalues, histtype="bar")
plt.xlabel('Sorted Average Task Times')
plt.ylabel('Seconds')
plt.bar(range(1,len(myvalues)+1), myvalues)

plt.savefig("my_hist.png", format="png")
```

1.2 boxplot-test

```
png('my_boxplot_test.png')
#lmts <- range(x1,x2,y1,y2)
par(mfrow = c(1, 2))
boxplot(mydata, mydata, xlab="x")
```

#+RESULTS: boxplot-test

1.3 Overview

Table 1: Overview of the input values, execution languages, and output values.

Input	**Language**	**Output**
`refinding_tagstore.csv`	shell	task time values
task time values	Python	average time values
average time values	shell	sorted numbers
average time values	R	boxplot of times

1.4 End

2. REFERENCES

[1] M. Delescluse, R. Franconville, S. Joucla, T. Lieury, and C. Pouzat. Making neurophysiological data analysis reproducible. why and how? *Journal of Physiology Paris*, (0), Aug. 2011.

[2] C. Dominik. *The Org-Mode 7 Reference Manual: Organize Your Life with GNU Emacs*. Network Theory, UK, 2010. with contributions by David O'Toole, Bastien Guerry, Philip Rooke, Dan Davison, Eric Schulte, and Thomas Dye.

[3] E. Schulte and D. Davison. Active documents with org-mode. *Computing in Science Engineering*, 13(3):66 –73, June 2011.

[4] E. Schulte, D. Davison, T. Dye, and C. Dominik. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, 46(3):1–24, 1 2012.

[5] K. Voit, K. Andrews, and W. Slany. TagTree: Storing and re-finding files using tags. In *Proc. 7^{th} Conference of the Austrian Computer Society Workgroup: Human-Computer Interaction (Usab 2011)*, volume 7058 of *LNCS*, pages 471–481. Springer, Nov. 2011.