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. Using a simple example this paper demonstrates how to combine raw data, scripts of various languages, and the describing text of a paper in one single file.

Categories and Subject Descriptors

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

Keywords

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

1. INTRODUCTION AND RELATED WORK

The editor Emacs[1] provides a modular system that can be extended in various ways. One example of an extension is the Org-mode[2][2] which resembles a large set of features: outlining, task and project management, personal wiki system, export to a broad variety of formats, integration of external tools, and much more.

Using the Babel subsystem to integrate external data sources – such as source code and execution results from these source code snippets – allows for creating meta-documents. This method can be applied for Literate Programming[3] or Reproducible Research[1, 4, 5].

Reproducible Research can be seen as the ultimate aim of Open Science and Open Data. People do not only get access to the data behind a research paper but also to the complete process of deriving the paper from the raw data up to the PDF file of the white paper.

This paper is a short demonstration of the general work flow from the very easy to write Org-mode text-file – which contains descriptive text and processing scripts – to the final PDF document in exact ACM template format[3].

Using a real world example, this paper develops a data processing path starting from raw data within CSV files up to visualizing plots. Different computer languages are part of this process: shell script tools, Python[4], and R[5]. The data is generated or processed using one computer language and transferred to the next one which is different from the previous. With this technique, it is possible to process the data with the most suitable command language appropriate for this single processing step.

2. EVALUATING A FORMAL EXPERIMENT

In [6] the authors describe a formal experiment conducted with 18 test persons in the field of information retrieval. The original data set is available online[6].

We are using this public available data set in the following sections to demonstrate the analysis and visualization of the research question: to what extent differs the test persons average re-find task times within the usual folder hierarchy from average re-find task times using the newly proposed

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

[1] http://www.gnu.org/software/emacs/

[2] http://orgmode.org

[3] http://www.acm.org/sigs/publications/proceedings-templates

[4] Programming language, http://www.python.org/

[5] Statistical software, http://www.r-project.org/

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

research software called *tagstore*[7].

2.1 Reading in Data from Files

The analysis starts with reading in raw data from external files. In this first external CSV file, the re-finding performance of all test persons is stored. Since we are only interested in the task times, we want to filter out all other values that are irrelevant for this analysis. We are using GNU shell commands and tools to accomplish this task.

The following shell commands

1. reads in data from a CSV file,
2. removes all values before the character ";" (thus removing all values related to number of mouse clicks),
3. removes all incomplete lines (containing the string "TC"),
4. 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 table above resembles the direct output of the command provided above. Using Org-mode the user can decide, if the commands, or the output, or both gets printed in the exported file.

2.2 Generating Mean Values

Now that we have the raw task times of the test persons for the *tagstore* condition, we do want to derive the arithmetic mean values for further processing.

In the next step, the mean values will be calculated using the programming language Python. We take the result data from the last command from the previous section and use it as input data for the following source code snippet:

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

This time, the output list consisting of 15 mean values is being suppressed for layout purposes.

2.3 Sorting Values

To demonstrate the combination of different command languages, we are now using GNU sed and sort to transform the list of mean values in a column format of mean values and sort it reverse numerical:

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

We get the sorted list of mean values of the task times for all test persons in the *tagstore* condition.

2.4 Process Folder Values

In this section we repeat the previous steps with the task data for the second test condition, the folder condition.

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

```
 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
```

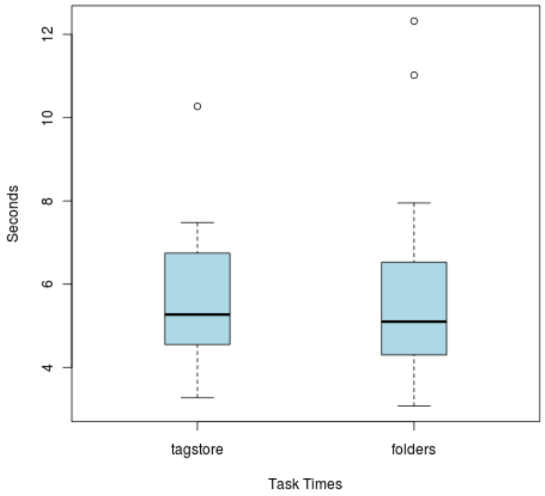
We do now have the second list of average task times which relates to the folder condition of the formal experiment.

2.5 Plotting Data

The open source statistical software R offers many different kind of commands to process data. To visualize the two sets of data using a boxplot graph, we are using the following R script:

[7] http://tagstore.org

```
png(’my_boxplot_data.png’)
## following two commands compensate a bug
## in the Babel method which should be fixed soon:
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"))
```



The resulting graph visualizes the overall result of the two test conditions: the average task times in the *tagstore* condition and the folders condition.

Every single step from the raw data from the CSV file to the graph can be completely checked and re-created by anybody who wants to do so. This method of Reproducible Research guarantees the highest level of transparency and confirmability.

3. OVERVIEW

To sum up the different data paths, we conclude in Table 1 the different steps. We have read in raw data from external CSV files and filtered them using shell tools.

Then we processed the data using a Python program in order to compute the average mean values of the times. Using the tool “sort” we got the sorted list of mean values which were visualized using R into boxplot graphs.

To summarize, the method of writing research papers within Org-mode can simplify the writing process itself, create additional value by having an easy to invoke data processing method, and it illustrates the author’s work in the most transparent way possible.

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

Input	Using	Output
`refinding_tagstore.csv`	shell	TS task time values
TS task time values	Python	TS average time values
TS average time values	shell	TS sorted numbers
`refinding_folder.csv`	shell	F task time values
F task time values	Python	F average time values
F average time values	shell	F sorted numbers
average time values TS + F	R	boxplot of times

4. 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] D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

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

[5] 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.

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