ProjectTemplate Demo

Ivan Hanigan

April 17, 2012

Outline

The Compendium concept

My goal is to develop data analysis projects along the lines of the Compendium concept of Gentleman and Temple Lang (2007) [1]. Compendia are dynamic documents containing text, code and data. Transformations are applied to the compendium to view its various aspects.

- Code Extraction (Tangle): source code
- Export (Weave): LaTeX, HTML, etc
- Code Evaluation

I'm also following the orgmode technique of Schulte et al (2012) [2]

ProjectTemplate

This is a simple demo of the R package *ProjectTemplate* http://projecttemplate.net/ which is aimed at standardising the structure and general development of data analysis projects in R. A primary aim is to allow analysts to quickly get a project loaded up and ready to:

- reproduce or
- create new data analyses.

Why?

It has been recognised on the R blogosphere that it

- is "meant to handle very complex research projects" (http://bryer.org/2012/maker-an-r-package-for-managing-document-building-and-versioning)
  and
- is considered as being amongst the best approaches to the workflow for doing data analysis with R (http://blog.revolutionanalytics.com/2010/10/a-workflow-for-r.html)

The Reichian load, clean, func, do approach

The already mentioned blog post
http://blog.revolutionanalytics.com/2010/10/a-workflow-for-r.html
also links to another‘best’ approach, the:

- *Reichian load, clean, func, do* approach
  http://stackoverflow.com/a/1434424.

Which I’ve also followed to prepare this demo using the tutorial and data from the package website
http://projecttemplate.net/getting_started.html

Init the project

First we want to initialise the project directory.

```
####
# init
require('ProjectTemplate')
create.project('analysis',minimal=TRUE)
```

dir()

```
####
# init dir
dir('analysis')
```

cache
config
data
munge
README
src

The reports directory

I've added the reports directory manually and asked the package author if this is generic enough to be in the defaults for

```
minimal = TRUE
```

I believe it may be as the *Getting Started* guidebook states:

> *'It's meant to contain the sort of written descriptions of the results of your analyses that you'd* ***publish in a scientific paper.***
> *With that report written . . . , we've gone through* ***the simplest sort of analysis you might run with ProjectTemplate****.*

```
####
# init reports
dir.create('analysis/reports')
```

Do the analysis

```
####
# this is the start of the analysis,
# assumes the init.r file has been run
if(file.exists('analysis')) setwd('analysis')
Sys.Date()
# keep a track of the dates the analysis is rerun
getwd()
# may want to keep a reference of the directory
# the project is in so we can track the history
```

Get the projecttemplate tutorial data

Get the data from http://projecttemplate.net/letters.csv.bz2 (I downloaded on 13-4-2012) Put it in the data directory for auto loading.

```
####
# analysis get tutorial data
download.file('http://projecttemplate.net/letters.csv.bz2',
  destfile = 'data/letters.csv.bz2', mode = 'wb')
```

Tools

Edit the *config/global.dcf* file to make sure that the load_libraries setting is turned on

Load the analysis data

```
####
# analysis load
require(ProjectTemplate)
load.project()
```

check the analysis data

```
tail(letters)
```

zyryan	z	y
zythem	z	y
zythia	z	y
zythum	z	y
zyzomys	z	y
zyzzogeton	z	y

Develop munge code

Edit the *munge/01-A.R* script so that it contains the following two lines of code:

```
# For our current analysis, we're interested in the total
# number of occurrences of each letter in the first and
# second letter positions and not in the words themselves.
# compute aggregates
first.letter.counts <- ddply(letters, c('FirstLetter'),
  nrow)
second.letter.counts <- ddply(letters, c('SecondLetter'),
  nrow)
```

Now if we run with

```
load.project()
```

all munging will happen automatically. However...

To munge or not to munge?

As you'll see on the website, once the data munging is completed and outputs cached, load.project() will keep recomputing work over and over. The author suggests we manually edit our configuration file.

```
 # edit the config file and turn munge on
 # load.project()
 # edit the config file and turn munge off
 # or my preference
 source('munge/01-A.r')
# which can be included in our first analysis script
# but subsequent analysis scripts can just call load.project
# without touching the config file
```

Cache

Once munging is complete we cache the results

```
cache('first.letter.counts')
cache('second.letter.counts')
```

Plot first and second letter counts

Produce some simple density plots to see the shape of the first and second letter counts.

- Create *src/generate_plots.R*. Use the src directory to store any analyses that you run.
- The convention is that every analysis script starts with load.project() and then goes on to do something original with the data.

Do generate plots

Write the first analysis script into a file in **src**

```
require('ProjectTemplate')
load.project()
plot1 <- ggplot(first.letter.counts, aes(x = V1)) +
  geom_density()
ggsave(file.path('reports', 'plot1.pdf'))

plot2 <- ggplot(second.letter.counts, aes(x = V1)) +
  geom_density()
ggsave(file.path('reports', 'plot2.pdf'))
```

And now run it (I do this from a main 'overview' script).

```
source('src/generate_plots.R')
```

First letter

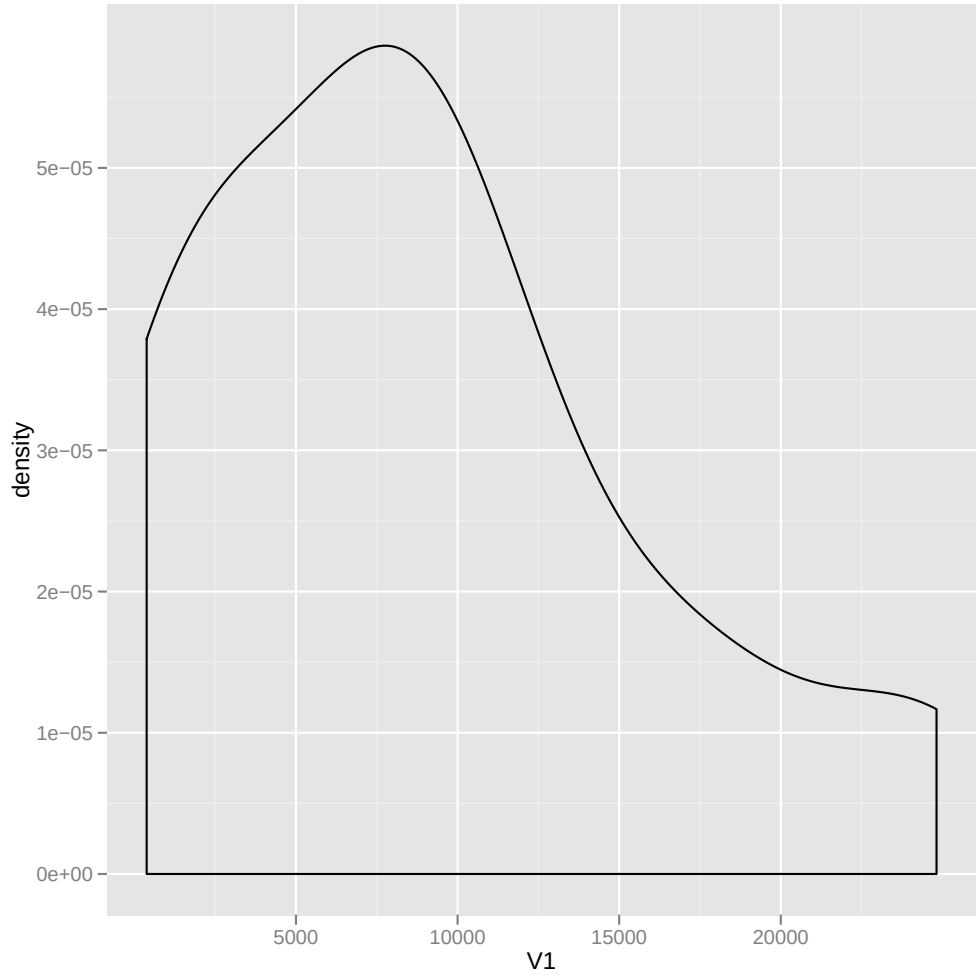


Figure: plot1.pdf

Second letter

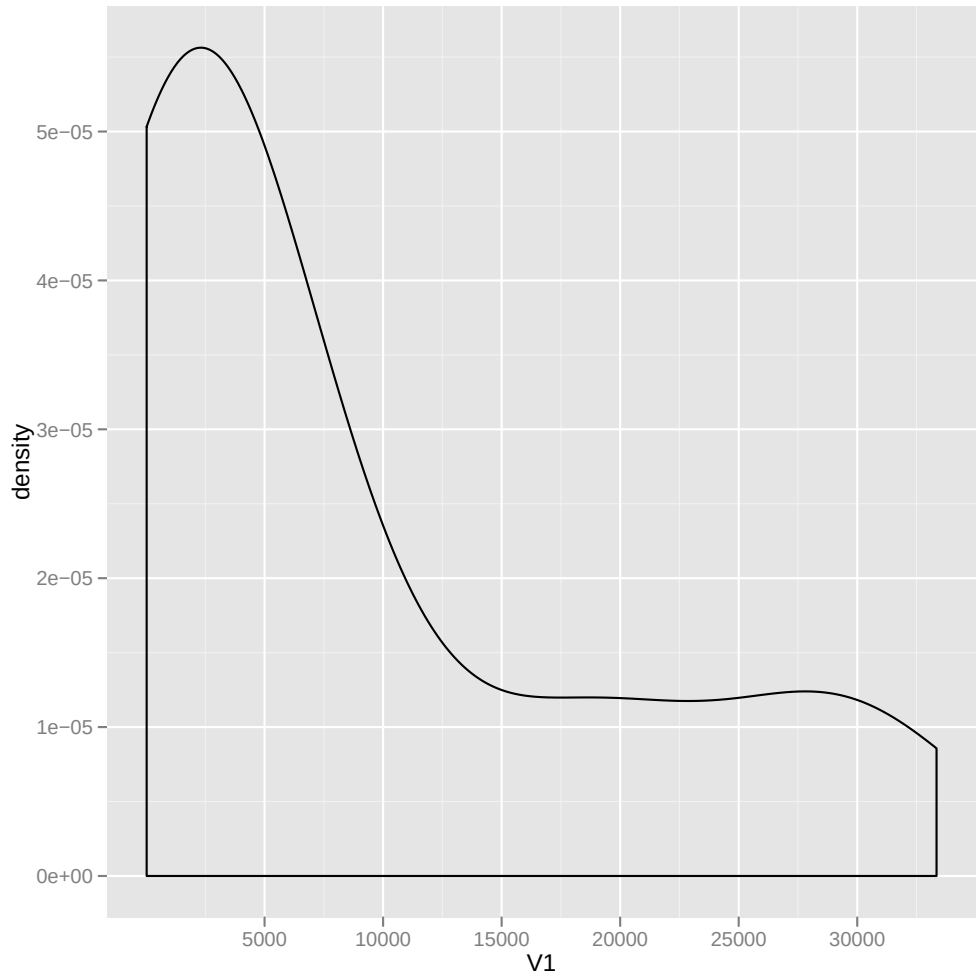


Figure: plot2.pdf

Report results

We see that both the first and second letter distributions are very skewed. To make a note of this for posterity, we can write up our discovery in a text file that we store in the reports directory.

```
\documentclass[a4paper]{article}
\title{Letters analysis}
\author{Ivan Hanigan}
\begin{document}
\maketitle
blah blah blah
\end{document}
```

Produce final report

```
# now run LaTeX on the file in reports/letters.tex
```

Personalised project management directories

```
####
# init additional directories for project management
analysisTemplate()

dir()
```

admin
analysis
data
document
init.r
metadata
ProjectTemplateDemo.org
references
tools
versions

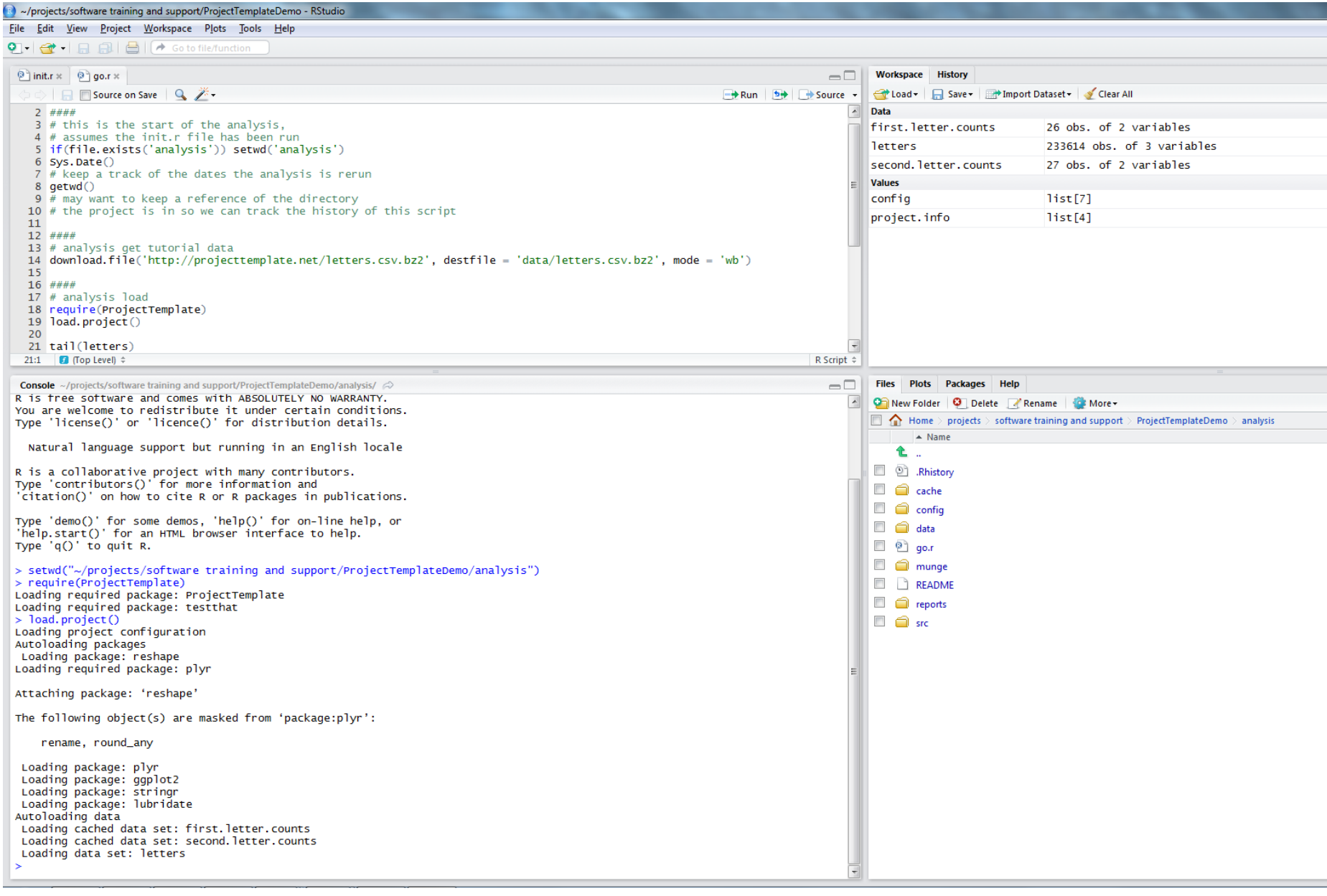
Navigating using other code editors

Emacs is not for everyone.

> *a great operating system, lacking only a decent editor*

http://upsilon.cc/žack/blog/posts/2008/10/from_Vim_to_Emacs_-_part_1/
Let's take a look at the project using RStudio.

RStudio


~/projects/software training and support/ProjectTemplateDemo - RStudio
File Edit View Project Workspace Plots Tools Help
Go to file/function
init.r
go.r
Source on Save
Run
Source
2 ####
3 # this is the start of the analysis,
4 # assumes the init.r file has been run
5 if(file.exists('analysis')) setwd('analysis')
6 Sys.Date()
7 # keep a track of the dates the analysis is rerun
8 getwd()
9 # may want to keep a reference of the directory
10 # the project is in so we can track the history of this script
11
12 ####
13 # analysis get tutorial data
14 download.file('http://projecttemplate.net/letters.csv.bz2', destfile = 'data/letters.csv.bz2', mode = 'wb')
15
16 ####
17 # analysis load
18 require(ProjectTemplate)
19 load.project()
20
21 tail(letters)
21:1 (Top Level)
R Script
Workspace
History
Load
Save
Import Dataset
Clear All
Data
first.letter.counts 26 obs. of 2 variables
letters 233614 obs. of 3 variables
second.letter.counts 27 obs. of 2 variables
Values
config list[7]
project.info list[4]
Console ~/projects/software training and support/ProjectTemplateDemo/analysis/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> setwd("~/projects/software training and support/ProjectTemplateDemo/analysis")
> require(ProjectTemplate)
Loading required package: ProjectTemplate
Loading required package: testthat
> load.project()
Loading project configuration
Autoloading packages
Loading package: reshape
Loading required package: plyr
Attaching package: 'reshape'
The following object(s) are masked from 'package:plyr':
rename, round_any
Loading package: plyr
Loading package: ggplot2
Loading package: stringr
Loading package: lubridate
Autoloading data
Loading cached data set: first.letter.counts
Loading cached data set: second.letter.counts
Loading data set: letters
>
Files
Plots
Packages
Help
New Folder
Delete
Rename
More
Home projects software training and support ProjectTemplateDemo analysis
Name
..
.Rhistory
cache
config
data
go.r
munge
README
reports
src

Conclusions

The Emacs Orgmode file is the compendium from which the whole analysis can be re-created. The upshot is that once I have developed the project's main ***.org*** file and completed the analysis I can send it (and only it) to another analyst and if they run it (using Emacs) it should get the project to exactly the same state that it was in when I left it, ready for reproduction or extension.

THANKS for listening!
To see a copy of the org file for this demo go to
https://github.com/ivanhanigan/ProjectTemplateDemo

References


Robert Gentleman and Duncan Temple Lang.
Statistical Analyses and Reproducible Research.
Journal of Computational and Graphical Statistics, 16(1):1–23, March 2007.


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), 2012.