Programming for Big Data– Continuous Assessment 4 (B8IT105)

Lecturer: Darren Redmond

Author: Maitiú Baxter

Student Number: 10349393

DBS Email: 10349393@mydbs.ie

# Assessment Task

### Assignment 4 is based on transforming a large dataset in text format - over 5000 lines of text.

### You will need to scrub (clean) the data and place it into the relevant holder/container objects.

### Once in these objects you will see that there are 422 different sets of commit objects.

### So your task will be to analyse these 422 objects that are in a list and come up with 3 interesting statistical pieces of information for this dataset with supporting evidence of "interestingness'

### You code for calculating the analysis should be documented and tested.

### Test should be in a separate file runnable from the command line.

### Your statistical analytics conclusions should be in a word document explaining in approximately 500 words the information that you have gleamed from the dataset.

### You will be required to submit your code via github along with all documentation and tests.

A template python file was created in class to separate the various elements of a provided log file. The code for both this and the test script modified to test the functions therein can be seen in Appendix 1 and 2: *change\_processor.py* and *test\_changes.py*.

The key additional information added which was not discussed in class was the importing of the csv library to enable us to create and write to a csv file. This code was included in the *process\_commits* function (contained in the *ChangeProcessor* class.) A list type container was created called *commits;* where each item within the list contained the respective elements of the revision made for that commit:

* Revision,
* Author,
* Timestamp,
* Lines in Comment,
* Commits,
* Comment

Once each commit from the provided log file was processed, and the list container was finalised, the csv library write function was used to write each element to a newly created csv file, namely ‘*ttest.csv’*.

RStudio was then utilised to begin the analysis of the data extracted from the log file. A working directory was established to ensure we were reviewing the correct csv file, and then the csv file was ‘read’ into memory as a data frame and the respective columns were named as above to identify them.

Two initial analysis functions were executed to help us identify early paths of investigation: the *str* function and the *summary* function. Results of this analysis can be seen below:

> str(dat)

'data.frame': 422 obs. of 10 variables:

$ Revision : Factor w/ 422 levels "r1491146","r1491229",..: 1 2 3 4 5 6 7 8 9 10 ...

$ Author : Factor w/ 10 levels "ajon0002","Alan",..: 9 6 2 9 6 6 6 9 9 9 ...

$ Timestamp : Factor w/ 422 levels "2015-07-13 09:21:48 +0100 (Mon, 13 Jul 2015)",..: 1 2 3 4 5 6 7 8 9 10 ...

$ Lines in Comment: Factor w/ 7 levels "1 line","2 lines",..: 1 1 1 1 1 1 1 1 1 1 ...

$ Commits : Factor w/ 289 levels "['Changed paths:', 'A /cloud/personal/client-international/android/branches/android-15.2-solutions/clients/clie"| \_\_truncated\_\_,..: 49 176 6 98 85 102 260 189 61 127 ...

$ Comment : Factor w/ 346 levels "['- fix a mime-type issue when uploading a file', '- use the latest Cloud SDK 1.0.8 branch version to fix issue"| \_\_truncated\_\_,..: 288 17 129 23 47 11 67 267 267 267 ...

> summary(dat)

Revision Author

r1491146: 1 Thomas :191

r1491229: 1 Jimmy :152

r1491254: 1 Vincent : 26

r1491272: 1 Domain Control Validated: 24

r1491315: 1 ajon0002 : 9

r1491323: 1 Freddie : 7

(Other) :416 (Other) : 13

Timestamp Lines in Comment

2015-07-13 09:21:48 +0100 (Mon, 13 Jul 2015): 1 1 line :348

2015-07-13 11:03:48 +0100 (Mon, 13 Jul 2015): 1 2 lines: 45

2015-07-13 11:32:02 +0100 (Mon, 13 Jul 2015): 1 3 lines: 11

2015-07-13 11:52:09 +0100 (Mon, 13 Jul 2015): 1 4 lines: 9

2015-07-13 13:06:21 +0100 (Mon, 13 Jul 2015): 1 5 lines: 7

2015-07-13 13:48:50 +0100 (Mon, 13 Jul 2015): 1 6 lines: 1

(Other) :416 7 lines: 1

Commits

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/clients/client-att/res/values/strings.xml'] : 29

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/version.properties'] : 28

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/build.gradle'] : 18

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/settings.gradle'] : 13

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/clients/common/mvn-manifest/AndroidManifest.xml']: 5

['Changed paths:', 'M /cloud/personal/client-international/android/branches/android-15.2-solutions/clients/common/lint.xml'] : 4

(Other) :325

Comment

['[gradle-release] prepare for next development iteration']: 24

['Lint fix'] : 8

['Partial code cleanup'] : 6

['Added strings as translated for phraseapp'] : 4

['Legacy app uninstall'] : 4

['Lint modification'] : 4

(Other) :372

Reviewing this data it was determined that the 3 most interesting items were

1. the number of commits per author
2. the day per week when most commits are performed
3. determining the least busy month for programmers

The items listed above have been plotted in R and can be seen in Figures 1, 2 and 3 below:

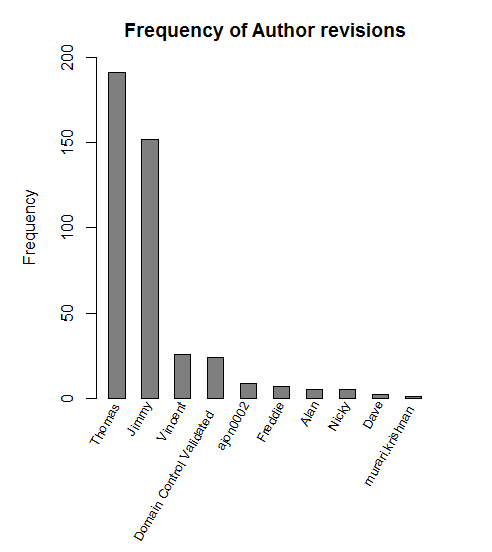


Figure - Commits by author

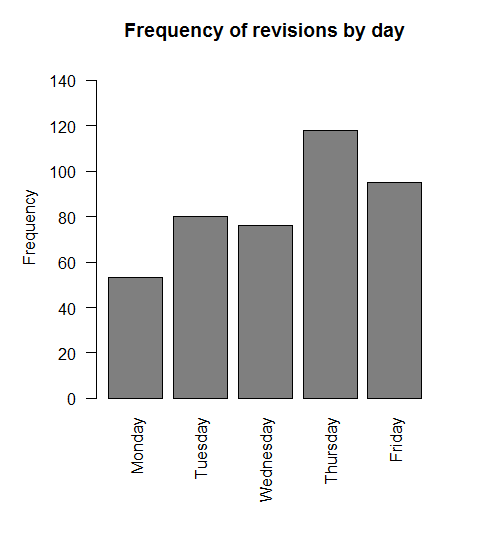


Figure - Commits by day

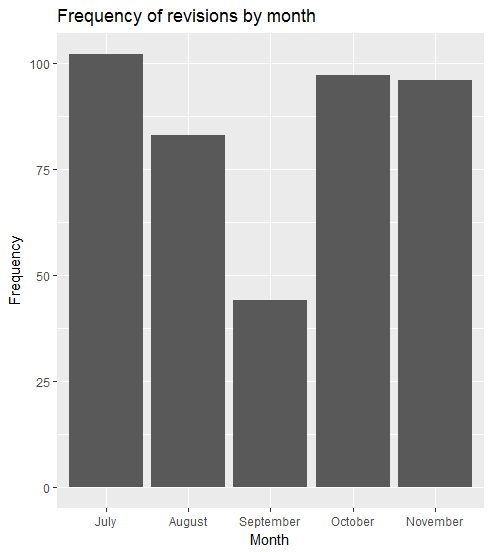


Figure - Commits by month

# Discussion

It can be seen that Thomas is the most productive (or cautious) of all the programmers, with frequent commits, closely followed by Jimmy. Both of them are significantly ahead of their peers for commits overall. Further analysis here could be conducted to determine the type of commit each are doing, to see if trends could be spotted. That is outside the scope of the current assignment. To note however that “[*gradle-release] prepare for next development iteration*” is the most common remark made for a commit.

As is to be expected, the frequency of commits increases towards the end of the week, as programmers complete tasks and commit their changes. This peaks on a Thursday.

September is the least active month for the programmers, with on average almost half the number of commits performed in this month as in any of the others. This may be driven by projects ending or a lull in commits driven by other activities in the workspace.

# Appendix 1 : change\_processor.py code

import csv

sep = 72\*'-'

# the class to process changes

class ChangeProcessor(object):

def \_\_init\_\_(self, file\_name):

self.read\_file(file\_name)

# open the file - and read all of the lines.

# use strip to strip out spaces and trim the line.

def read\_file(self, file\_name):

self.data = [line.strip() for line in open(file\_name, 'r')]

self.process\_commits()

return len(self.data)

def process\_commits(self):

self.commits = []

self.authors = {}

current\_commit = None

index = 0

while True:

try:

# parse each of the commits and put them into a list of commits

details = self.data[index + 1].split('|')

details = [column.strip() for column in details]

# the file changes with spaces at end removed.

details.append(self.data[index+2:self.data.index('',index+1)])

# add details to the list of commits.

self.commits.append(details)

index = self.data.index(sep, index + 1)

if details[1] not in self.authors: #if author in authors

self.authors[details[1]] = 1

else:

self.authors[details[1]] = self.authors[details[1]] + 1

comment\_line\_count = int(details[3].strip().split(' ')[0])

# add the comments to the details.

details.append(self.data[index-comment\_line\_count:index])

except IndexError:

break

# return the order of the file to the same sequence it was created

self.commits.reverse()

# create a writable file to output our container to

ofile = open('ttest.csv', "wb")

writer = csv.writer(ofile, delimiter=',')

# write to the file and use ',' as a delimitor - csv

for index in self.commits:

writer.writerow(index)

# close the opened file so it can be accessed

ofile.close()

def get\_author(self, index):

return self.commits[index][1]

def get\_date(self, index):

return self.commits[index][2]

def get\_number\_of\_comment\_lines(self, index):

return self.commits[index][3]

def get\_revision(self, index):

return self.commits[index][0]

def get\_file\_changes(self, index):

return self.commits[index][4]

def get\_comment(self, index):

return self.commits[index][5]

def get\_number\_of\_commits(self):

return len(self.commits)

def get\_number\_of\_authors(self):

return len(self.authors)

def get\_number\_of\_commit\_by\_author(self, author):

return self.authors[author]

# enable so that this is only called when the script run from the command line

if \_\_name\_\_ == '\_\_main\_\_':

fname = 'changes\_python.log'

changes = ChangeProcessor(fname)

#print(changes.commits)

#print(changes.authors)

#print(changes.get\_author(5))

#print(changes.get\_comment(102))

# Appendix 2: test\_changes code

import unittest

from change\_processor import ChangeProcessor

# test the changes functionality

class TestChanges(unittest.TestCase):

def setUp(self):

self.change\_processor = ChangeProcessor('test\_changes\_1.txt')

self.change\_processor2 = ChangeProcessor('test\_changes\_2.txt')

self.change\_processor3 = ChangeProcessor('changes\_python.log')

# this tests the length of data file is read correctly.

def test\_changes\_length\_of\_data(self):

result = len(self.change\_processor.data)

self.assertEqual(10, result)

result = len(self.change\_processor2.data)

self.assertEqual(99, result)

result = len(self.change\_processor3.data)

self.assertEqual(5255, result)

# this tests the number of commits by author

def test\_changes\_get\_number\_of\_commit\_by\_author(self):

result = self.change\_processor3.get\_number\_of\_commit\_by\_author('Thomas')

self.assertEqual(191, result)

# this tests the number of authors

def test\_changes\_number\_of\_authors(self):

result = self.change\_processor.get\_number\_of\_authors()

self.assertEqual(1, result)

result = self.change\_processor2.get\_number\_of\_authors()

self.assertEqual(3, result)

result = self.change\_processor3.get\_number\_of\_authors()

self.assertEqual(10, result)

# this tests the number of commits

def test\_changes\_number\_of\_commits(self):

result = self.change\_processor.get\_number\_of\_commits()

self.assertEqual(1, result)

result = self.change\_processor2.get\_number\_of\_commits()

self.assertEqual(9, result)

result = self.change\_processor3.get\_number\_of\_commits()

self.assertEqual(422, result)

# this tests the author

def test\_changes\_get\_author(self):

result = self.change\_processor.get\_author(0)

self.assertEqual('viacheslav.vdovenko', result)

result = self.change\_processor2.get\_author(6)

self.assertEqual('vnai0001', result)

result = self.change\_processor3.get\_author(420)

self.assertEqual('Thomas', result)

# this tests the first revision

def test\_changes\_get\_revision(self):

result = self.change\_processor3.get\_revision(0)

self.assertEqual('r1491146', result)

# this tests the date

def test\_changes\_get\_date(self):

result = self.change\_processor3.get\_date(0)

self.assertEqual('2015-07-13 09:21:48 +0100 (Mon, 13 Jul 2015)', result)

# this tests the comment

def test\_changes\_get\_comment(self):

result = self.change\_processor.get\_comment(0)

self.assertEqual(['Renamed folder to the correct name'], result)

result = self.change\_processor2.get\_comment(5)

self.assertEqual(['Chnaged jira url to htps'], result)

result = self.change\_processor3.get\_comment(420)

self.assertEqual(['Removed unused webview.plan.management and webview\_plan\_management properties'], result)

result = self.change\_processor3.get\_comment(421)

self.assertEqual(['Renamed folder to the correct name'], result)

# this tests the number of comment lines

def test\_changes\_get\_number\_of\_comment\_lines(self):

result = self.change\_processor3.get\_number\_of\_comment\_lines(0)

self.assertEqual('1 line', result)

# this tests the file changes of first revision

def test\_changes\_get\_file\_changes(self):

result = self.change\_processor3.get\_file\_changes(0)

self.assertEqual(['Changed paths:',\

'D /cloud/personal/client-international/android/branches/15-2-solutions',\

'A /cloud/personal/client-international/android/branches/android-15.2-solutions (from /cloud/personal/client-international/android/branches/15-2-solutions:1491145)'], result)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

# Appendix 3 : analyse\_in\_r.R code

#### set the working directory here ######

setwd("C:/Users/macan/programming\_for\_big\_data\_10349393/CA4")

####### end wd #############

library(ggplot2)

# read in csv file

dat <- read.csv(file="ttest.csv", sep=",", header = F)

# name the columns

names(dat) <- c("Revision", "Author", "Timestamp","Lines in Comment","Commits","Comment")

# rename a long 'author' name with something slightly more meaningful

dat$Author <- factor(gsub("/OU=Domain Control Validated/CN=svn.company.net", "Domain Control Validated", dat$Author))

# some informative data about our dataframe. Helps to identify early points of analysis

str(dat)

summary(dat)

##### interesting statistical observation No. 1 ####

authors.freq <- sort(table(dat$Author),decreasing = TRUE)

# barplot with perpendicular x-axis labels. 60deg looks better, see below

#barplot(authors.freq,main="Frequency of Author revisions",ylab="Frequency",xlab="Author Name",las=3)

par(mar = c(7, 4, 2, 2) + 1) #add room for the rotated labels

end\_point = .5 + nrow(authors.freq) + nrow(authors.freq)-1 #this is the line which does the trick (together with barplot "space = 1" parameter)

barplot(authors.freq, col="grey50",

main="Frequency of Author revisions",

ylab="Frequency", ylim=c(0,200),

xlab = "",

xaxt = "n",

space=1)

#rotate 60 degrees, srt=60

text(seq(1.5,end\_point,by=2), par("usr")[3]-0.25,

srt = 60, adj= 1.1, xpd = TRUE,

labels = paste(rownames(authors.freq)), cex=0.8)

####### add some extra columns using the timestamp ########

dat$Date <- as.Date(dat$Timestamp)

dat$Day <- weekdays(dat$Date)

dat$Month <- format(dat$Date, "%B")

dat$Time <- format(as.POSIXct(dat$Timestamp),format = "%H:%M:%S")

month.freq <- summary(factor(dat$Month, levels = c("July", "August", "September", "October", "November")))

day.freq <- summary(factor(dat$Day, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")))

##### interesting statistical observation No. 2 ####

## commit occurances by day ##

barplot(day.freq, col="grey50",

main="Frequency of revisions by day",

ylab="Frequency", ylim=c(0,150),

xlab = "",

las = 2)

##### interesting statistical observation No. 3 ####

## commit occurances by month ##

df <- as.data.frame(month.freq)

names(df) <- c("Frequency")

df$names <- factor(rownames(df), as.character(rownames(df)))

ggplot(df, aes(x=names, y=Frequency)) +

geom\_bar(aes(x=names), data=df, stat="identity") +

labs(y="Frequency",x="Month",title="Frequency of revisions by month")