## Author’s Writing – Multinomial Logistic Regression Analysis

Lisbona, Randall, [rlisbona@mail.smu.edu](mailto:rlisbona@mail.smu.edu) *Southern Methodist University* and Stoffa, Joseph, [jstoffa@mail.smu.edu](mailto:jstoffa@mail.smu.edu)  Southern Methodist University

Introduction

Natural Language Processing (NLP) is an area of study related to developing a computer’s ability to “understand” and process human or “natural” language. NLP packages such as Python’s Natural Language Toolkit provide several tools to parse and analyze text including methods for tokenization, tagging, and frequency calculations. This analysis is a continuation of the research in predicting the authorship of a work(s) based off of (still writing)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 1 - Five Number Summary and Standard Deviation** | | | | |
|  | Per. Small | rather | quite | though |
| Min. | 44.05 | 0.0154 | 0.0164 | 0.0392 |
| 1st Qu. | 45.58 | 0.0379 | 0.0562 | 0.0491 |
| Median | 46.07 | 0.0507 | 0.0804 | 0.0564 |
| Mean | 46.17 | 0.05912 | 0.09509 | 0.09282 |
| 3rd Qu. | 46.81 | 0.0723 | 0.1131 | 0.1409 |
| Max. | 48.67 | 0.1327 | 0.2188 | 0.1855 |
| Std. Dev. | 1.1020 | 0.0313 | 0.0607 | 0.0554 |
|  | Per. Med | hardly | perhaps | without |
| Min. | 27.33 | 0.0017 | 0.0201 | 0.0438 |
| 1st Qu. | 28.72 | 0.0094 | 0.0309 | 0.0632 |
| Median | 29.79 | 0.0275 | 0.0548 | 0.0741 |
| Mean | 30.46 | 0.02809 | 0.05518 | 0.09046 |
| 3rd Qu. | 31.98 | 0.0451 | 0.0686 | 0.1301 |
| Max. | 35.00 | 0.0542 | 0.1096 | 0.1468 |
| Std. Dev. | 2.3769 | 0.0180 | 0.0268 | 0.0374 |
|  | Per. Large | certainly | perfectly | either |
| Min. | 18.91 | 0.0157 | 0.0021 | 0.009 |
| 1st Qu. | 21.22 | 0.218 | 0.0069 | 0.016 |
| Median | 22.91 | 0.0397 | 0.0158 | 0.0292 |
| Mean | 23.37 | 0.03842 | 0.02071 | 0.02878 |
| 3rd Qu. | 25.79 | 0.0514 | 0.0359 | 0.0407 |
| Max. | 27.23 | 0.0642 | 0.0518 | 0.0603 |
| Std. Dev. | 2.7844 | 0.0165 | 0.0159 | 0.0157 |

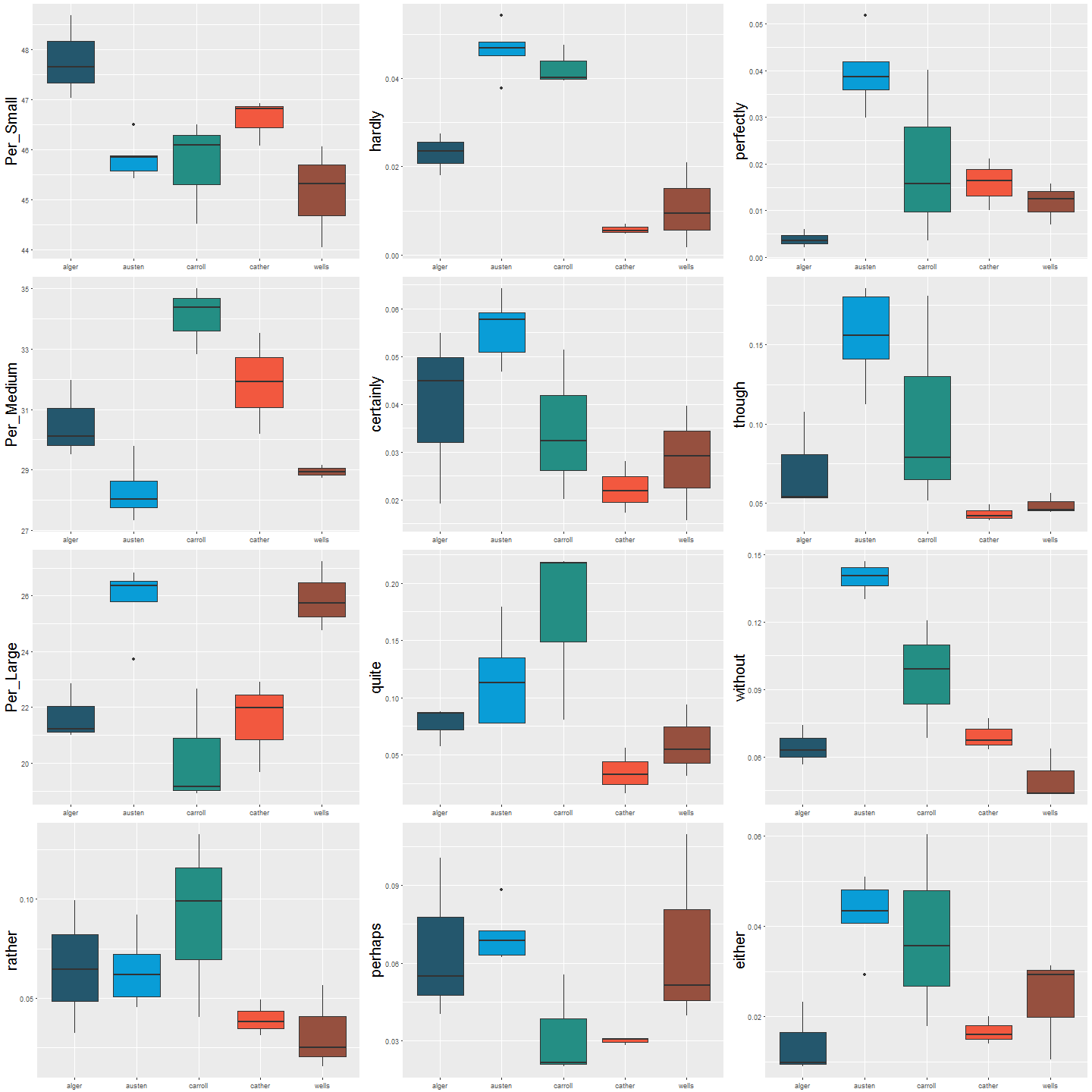
Section I – Building the Dataset

Files were downloaded from TextFile.com. Most the files available contained text written in the 19th Century, therefore, only novels from that period where considered in order maintain a consistent vernacular. Furthermore, only collections of three or more books from the same author were considered to maintain a good block size. Similarly, only texts having at least 300 pages were used to ensure text diversity.

Once downloaded, a Python script systematically read the text from each file into memory as a string, which was then tokenized and the frequency of each word calculated using the freqDist() method from the NLTK library. A set of custom functions then compared the texts to produce a common list of 166 words. A custom dictionary was created by scraping dictionary.com to classify each word in the list by type. The list was then subsetted to include only adverbs five letters or longer. Adverbs were chosen for this study under the assumption that verb, noun, and adjective use would likely correlate with genre which was not accounted for. Furthermore, conjuctions, prepositions, articles etc. where viewed as too generic. The final subset included 24 adverbs. The final dataset consisted of the author’s name, book title, and the percentage of small, medium, large, and each of the 24 adverbs.

Section II – Exploratory Analysis

text

**Figure 1 – Boxplots of Word Frequencies**

among authors. For example, although Cather and Wells use the words “hardly”, “perfectly”, “though”, “certainly”, “quite”, “rather”, and “either” with similar frequency they differ drastically in their use of small and large words in general in addition to “perhaps” and “without”.

Section III – Model Assumptions

Assumptions – I got this.

Section IV – Multinomial Logistic Regressiona

|  |  |
| --- | --- |
| **Table 2 - Bartlett's Test** | |
| Variable | P-Value |
| Percent Small | 0.5156 |
| Percent Medium | 0.3111 |
| Percent Large | 0.869 |
| Rather | 0.3137 |
| Hardly | 0.2483 |
| Certainly | 0.4089 |
| Quite | 0.2676 |
| Perhaps | 0.01994 |
| Perfectly | 0.07681 |
| Though | 0.02037 |
| Without | 0.1866 |
| Either | 0.2117 |

Running a MANOVA test using Wilk’s Lambda in R yields an F-statistic of 0.022 which means we reject the null hypothesis that the frequencies of word use are the same; the frequency of at least one word varies between authors. In other words, at least one of the authors has a word frequency that is different from the others. Looking at the individual F-statistics, Reference Table 3, it can be seen that except for “rather” and “perhaps” the frequency of words differs for at least one author from the others.

To determine if word frequency differs for more than one author a series of contrasts where conducted. Because of the low number of observations contrasts were limited to just five independent variables: “hardly”, “quite”, “perfectly”, “without”, “either”. Looking at Table 4, less than half of the possible contrasts yielded statistically significant results; however, this could be due to the abridged variable lists. For example, if the variable “Per\_Large” is included in the contrast of Austen v. Carroll the F-Statistic drops to 0.0374.

|  |  |
| --- | --- |
| **Table 3 - Individual F-Statics** | |
| Variable | P-Value |
| Percent Small | 0.00852 |
| Percent Medium | 0.00013 |
| Percent Large | 0.0004725 |
| Rather | 0.1357 |
| Hardly | 1.67E-06 |
| Certainly | 0.01745 |
| Quite | 0.01784 |
| Perhaps | 0.08062 |
| Perfectly | 0.001873 |
| Though | 0.004147 |
| Without | 4.16E-06 |
| Either | 0.01996 |

|  |  |
| --- | --- |
| **Table 4 - Contrasts** | |
| Authors | F-Statistic |
| Alger v. Austin | 0.01063 |
| Alger v. Carroll | 0.426 |
| Alger v. Cather | 0.2084 |
| Alger v. Wells | 0.2344 |
| Austin v. Carroll | 0.4966 |
| Austin v. Cather | 0.0014 |
| Austin v. Wells | 0.0059 |
| Carroll v. Cather | 0.115 |
| Carroll v. Wells | 0.271 |
| Cather v. Wells | 0.0365 |

Conclusion

We did maths and good stuffs happened.

Appendix

Python Script for Creating Dataset

import nltk

import os

import json

import requests

from lxml import html

import pandas as pd

pd.set\_option("display.max\_columns", 30) #set to view all columns in dataset

#set directory for saving dataset

#parent = os.path.dirname(\_\_file\_\_)

parent = 'c:\\Users\\anobs\\Documents\\GitHub\\MSDS6372\_Project3'

BookDir = os.path.join(parent, 'Books')

DataDir = os.path.join(parent, 'Data')

print(os.getcwd())

filename = os.path.join(DataDir, 'test.txt')

#scrapes dictionary.com to determine a words type

def lookupType(word):

baseURL = "http://www.dictionary.com/browse/"

url = baseURL + word

removeWords = ["sentence", "idiom", "idioms", "interjection"]

trans\_table = dict.fromkeys(map(ord, ','), None)

page = requests.get(url)

tree = html.fromstring(page.content)

wordTypesRaw = tree.xpath('//header[@class="luna-data-header"]//span[@class="dbox-pg"]/text()')

wordTypes = []

for word in wordTypesRaw:

word = word.lower()

word = word.translate(trans\_table)

wordTypes.append(word.split(" ")[0])

wordTypes = set(wordTypes)

for word in removeWords:

if word in wordTypes:

wordTypes.remove(word)

return(list(wordTypes))

#takes a list of token lists and returns a lists of tokens found in all lists

def findShared(list\_of\_lists):

notin = []

seen = set(list\_of\_lists[0])

for i in range(1, len(list\_of\_lists)):

for w in seen:

if w not in list\_of\_lists[i] and w not in notin:

notin.append(w)

for r in notin:

seen.remove(r)

return(seen)

#returns a dictionary of the number and percents of small, medium, and large

#words

def getWordDist(tokens):

word\_count = len(tokens)

small\_words = []

med\_words = []

large\_words = []

for w in tokens:

if len(w) < 4:

small\_words.append(w)

elif len(w) < 6 and len(w) > 3:

med\_words.append(w)

else:

large\_words.append(w)

small = len(small\_words)

med = len(med\_words)

large = len(large\_words)

return({"Small": small, "Medium": med, "Large": large, \

"PerSmall": format(small/word\_count\*100, ".2f"), \

"PerMed": format(med/word\_count\*100, ".2f"), \

"PerLarge": format(large/word\_count\*100, ".2f"), \

"SmallList": small\_words, "MedList": med\_words, "LargeList": large\_words})

#returns a dictionary that includes: list of common tokens and books. Books is

#a list of books and contains: title, author, and dictionary of words and their

#frequency in the text

def createVocabDistro(directory):

vocab\_dict = {}

files = os.listdir(directory)

list\_of\_lists = []

books = []

for file in files:

book = {}

split\_file = file.split("-")

author = split\_file[0]

title = split\_file[1]

book\_text = open(directory+"\\"+file, "r")

tokens = nltk.word\_tokenize(book\_text.read())

tokens = [w.lower() for w in tokens if w.isalpha()]

fdist = nltk.FreqDist(tokens)

freq\_dict = getWordDist(tokens)

vocabulary = set(tokens)

list\_of\_lists.append(vocabulary)

book["title"] = title

book["author"] = author

book["words"] = len(tokens)

book["frequency"] = fdist

book["perSmall"] = freq\_dict["PerSmall"]

book["perMed"] = freq\_dict["PerMed"]

book["perLarge"] = freq\_dict["PerLarge"]

books.append(book)

book\_text.close()

shared = list(findShared(list\_of\_lists))

vocab\_dict["shared"] = shared

vocab\_dict["books"] = books

return(vocab\_dict)

#write vocabDistro to file or if file already exists load from file

jsonfilename = os.path.join(DataDir, "vocabDistro.json")

if os.path.isfile(jsonfilename):

print("reading stored information from json file")

with open(jsonfilename) as json\_data:

d = json.load(json\_data)

else:

with open(jsonfilename, "w") as fout:

d = createVocabDistro(BookDir)

json.dump(d, fout)

#build dictionary from shared vocab words, if word is not already in

#dictionary.txt, scrape www.dictionary.com for word. After all shared words

#are checked save dictionary to dicionary.txt if applicable. Dictionary.txt

#is created so that www.dictionary.com server(s) aren't accessed more

#than necessary

dictFile = os.path.join(DataDir, "dictionary.json")

if os.path.isfile(dictFile):

textfile = open(dictFile, "r")

dictionary = json.load(textfile)

textfile.close()

else:

with open(dictFile, "w") as fout:

fout.write("{}")

dictionary = "{}"

fout.close()

update = False

for word in d['shared']:

if word not in dictionary:

try:

dictionary[word] = {"type": lookupType(word)}

print("looking up "+word)

update = True

except:

print("error looking up "+word)

if update:

print("\*\*\*updating dictionary\*\*\*")

fout = open(dictFile, "w")

json.dump(dictionary, fout)

fout.close()

print("Adverbs: ", len([w for w in d["shared"] if len(w) > 4 and "adverb"\

in dictionary[w]["type"]]))

#get list of adverbs from saved file if it exists or randomly sample 20 if it

#doesn't

adverb\_filename = os.path.join(DataDir, "chosenAdverbs.txt")

if os.path.isfile(adverb\_filename):

with open(adverb\_filename) as f:

Words = f.read().split(", ")

else:

#randomly chose 8 adverbs longer than 4 letters from word list and save

#list to file

with open(adverb\_filename, "w") as fout:

Words = [w for w in d["shared"] if len(w) > 4 and "adverb"\

in dictionary[w]["type"]]

fout.write(', '.join(Words))

#create dataset from directory of books

Title = []

Author = []

Per\_Small = []

Per\_Medium = []

Per\_Large = []

for book in d["books"]:

Title.append(book["title"])

Author.append(book["author"])

Per\_Small.append(book["perSmall"])

Per\_Medium.append(book["perMed"])

Per\_Large.append(book["perLarge"])

booksDF = pd.DataFrame.from\_items([('Title',Title), ('Author',Author), ('Per\_Small',Per\_Small),\

('Per\_Medium',Per\_Medium), ('Per\_Large',Per\_Large)])

for word in Words:

word\_freq = []

for book in d["books"]:

word\_freq.append(format(book["frequency"][word]/book["words"]\*100, ".4f"))

booksDF[word] = word\_freq

print(Words)

print("shared: ", len(d["shared"]))

df\_textfile = os.path.join(DataDir, "booksDF.csv")

if not (os.path.isfile(df\_textfile)):

print("\*\*\*Writing Dataframe to CSV file\*\*\*")

booksDF.to\_csv(df\_textfile, index=False)

else:

print("Dataframe is already stored as a file.")

R Script for MANOVA: