National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Xiyao Xu
Student ID: x18107834

School of Computing
National College of Ireland

Supervisor: XXX

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……xiyao xu……

Student ID: ……x18107834……..……

Programme: ……Msc in Data Analysis…… **Year:** …2018-2019

Module: ……Research paper……..……

Lecturer: ……Bahman Honari……..……

Submission Due Date: …… …December 12th 2019……..……

Project Title: Analysing the impact of social media on online shopping platform sales by using sentiment analysis with text mining……..……

Word Count: ……150…… **Page Count:** ……17……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……xiyao xu……

Date: ……December 12th……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Xiyao Xu
Student ID: x18107834

1 Software Requirement

Data extraction:

The data is extract by data extraction software: bazhuayu. https://www.bazhuayu.com/

Register for the software. Then click on the following icon.

2 Data pre-processing

I used python to do pre-processing. Software used: Pycharm

Python download: https://www.python.org/downloads/release/python-2716/

Pycharm download: https://www.jetbrains.com/pycharm/

Codes for sentiment analysis pre-processing

```
# -*- coding: utf-8 -*-
__author__ = 'Bai Chenjia'

import ...
print "user dictionary loading..."
import sys
reload(sys)
sys.setdefaultencoding("utf-8")
jieba.load_userdict('F://paper/word_dictionary/pos_dic.txt')
jieba.load_userdict('F://paper/word_dictionary/neg_dic.txt')

# segmentation, return list
def segmentation(sentence):
    seg_list = jieba.cut(sentence)
    seg_result = []
    for w in seg_list:
        seg_result.append(w)
    #print seg_result[:]
    return seg_result


def postagger(sentence):
    pos_data = pseg.cut(sentence)
    pos_list = []
    for w in pos_data:
        pos_list.append((w.word, w.flag))
    #print pos_list[:]
    return pos_list

# sentence segment
def cut_sentence(words):
    words = words.decode('utf8')
    start = 0
    i = 0
    token = 'meaningless'
    sents = []
    punt_list = ',.!?;~, 。！？；～… '.decode('utf8')
    #print "punc_list", punt_list
    for word in words:
        #print "word", word
        if word not in punt_list:   # if it is not Apostrophe
```

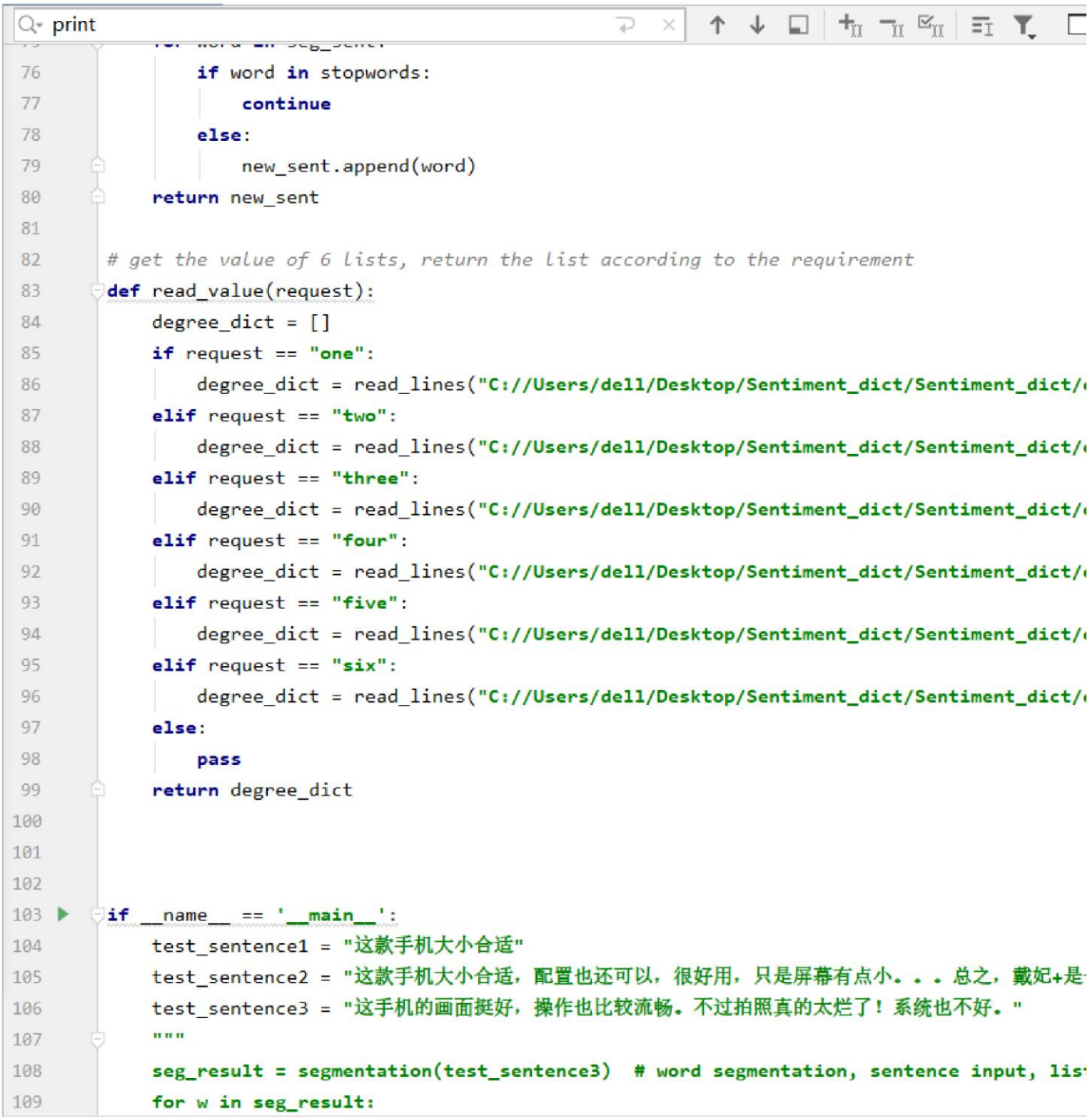
segmentation() › for w in seg_list

```
            #print "word", word
            if word not in punt_list:   # if it is not Apostrophe
                #print "word1", word
                i += 1
                token = list(words[start:i+2]).pop()
                #print "token:", token
            elif word in punt_list and token in punt_list:  # process Ellipsis
                #print "word2", word
                i += 1
                token = list(words[start:i+2]).pop()
                #print "token:", token
            else:
                #print "word3", word
                sents.append(words[start:i+1])   # stop the sentence
                start = i + 1
                i += 1
        if start < len(words):
            sents.append(words[start:])
        return sents

def read_lines(filename):
    fp = open(filename, 'r')
    lines = []
    for line in fp.readlines():
        line = line.strip()
        line = line.decode("utf-8")
        lines.append(line)
    fp.close()
    return lines

# remove stop words
def del_stopwords(seg_sent):
    stopwords = read_lines("F://paper/stop_words.txt")  # read stop word l
    new_sent = []   # stop removed sentence
```

Package pip, jieba, numpy are needed.



3 Data analysis

3.1 Sentiment analysis

Sentiment analysis used python and pycharm, which the same with section2.
Code:

```
# -*- coding: utf-8 -*-
__author__ = 'Bai Chenjia'

import text_process as tp
import numpy as np



print "Begin"
posdict = tp.read_lines("C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/emotion_di
negdict = tp.read_lines("C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/emotion_di

most_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/degree_
very_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/degree_
more_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/degree_
ish_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/degree_
insufficient_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dic
inverse_degree = tp.read_lines('C://Users/dell/Desktop/Sentiment_dict/Sentiment_dict/deg
print "The degree files are loaded"

emotion_level1 = "sad"
emotion_level2 = "angry"
emotion_level3 = "calm"
emotion_level4 = "peaceful"
emotion_level5 = "happy"

emotion_level6 = "Emotion fluctuation is small"
emotion_level7 = "Emotion fluctuation is big"


# 2.Adverb of degree processing, according to the type of degree adverb, multiplied by d
def match(word, sentiment_score):
    if word in most_degree:
        sentiment_score *= 2.0
```

text_process.py × dict_main.py ×

```
35      elif word in very_degree:
36          sentiment_score *= 1.75
37      elif word in more_degree:
38          sentiment_score *= 1.5
39      elif word in ish_degree:
40          sentiment_score *= 1.2
41      elif word in insufficient_degree:
42          sentiment_score *= 0.5
43      elif word in inverse_degree:
44          #print "inversedict", word
45          sentiment_score *= -1
46      return sentiment_score
47
48  # Example: [5, -2] →  [7, 0]; [-4, 8] →  [0, 12]
49  def change_to_positive_number(positive_count, negative_count):
50      pos_count = 0
51      neg_count = 0
52      if positive_count < 0 and negative_count <= 0:
53          neg_count += negative_count - positive_count
54          pos_count = 0
55      elif negative_count < 0 and positive_count < 0:
56          pos_count = positive_count - negative_count
57          neg_count = 0
58      elif positive_count < 0 and negative_count >= 0:
59          neg_count = -positive_count
60          pos_count = -negative_count
61      else:
62          pos_count = positive_count
63          neg_count = negative_count
64      return (pos_count, neg_count)
65
66
67  # Find the total sentiment score of a single Weibo sentence
68  def single_sentence_sentiment_score(weibo_sent):
```

```
68  def single_sentence_sentiment_score(weibo_sent):
69      single_review_senti_score = []
70      cuted_review = tp.cut_sentence(weibo_sent)  #Sentence segmentation, analyzing each s
71
72      for sent in cuted_review:
73          seg_sent = tp.segmentation(sent)   # segmentation
74          seg_sent = tp.del_stopwords(seg_sent)[:]
75          #for w in seg_sent:
76          #   print w,
77          i = 0    # Record the position of the scanned word
78          s = 0    # Record the position of the sentiment word
79          poscount = 0    # Record the positive emotion score in the segmented sentence
80          negcount = 0    # Record the negative emotion score in the segmented sentence
81
82          for word in seg_sent:   # analysis sentence by sentence
83              #print word
84              if word in posdict:
85                  #print "posword:", word
86                  poscount += 1
87                  for w in seg_sent[s:i]:
88                      poscount = match(w, poscount)
89                  #print "poscount:", poscount
90                  s = i + 1
91
92              elif word in negdict:
93                  #print "negword:", word
94                  negcount += 1
95                  for w in seg_sent[s:i]:
96                      negcount = match(w, negcount)
97                  #print "negcount:", negcount
98                  s = i + -1
99
100
101             elif word == "！".decode("utf-8") or word == "!".decode('utf-8'):
```

text_process.py × dict_main.py ×

```
102                 for w2 in seg_sent[::-1]:
103                     if w2 in posdict:
104                         poscount += 2
105                         break
106                     elif w2 in negdict:
107                         negcount += 2
108                         break
109             i += 1
110
111         single_review_senti_score.append(change_to_positive_number(poscount, negcount))
112     pos_result, neg_result = 0, 0   # Record the positive sentiment score and negative s
113     for res1, res2 in single_review_senti_score:
114         pos_result += res1
115         neg_result += res2
116     #print pos_result, neg_result
117     result = pos_result - neg_result   # final score of the sentence
118     result = round(result, 1)
119     return result
120
121 """
122 # 测试
123 weibo_sent = "这手机的画面挺好，操作也比较流畅。不过拍照真的太烂了！系统也不好。"
124 score = single_review_sentiment_score(weibo_sent)
125 print score
126 """
127
128
129 def run_score():
130     fp_test = open('F://paper/code/test_data.txt', 'r')
131     contents = []
132     for content in fp_test.readlines():
133         content = content.strip()
134         content = content.decode("utf-8")
135         contents.append(content)
```

text_process.py × dict_main.py ×

```
136     fp_test.close()
137     results = []
138     for content in contents:
139         score = single_sentence_sentiment_score(content)
140         results.append((score, content))
141     return results
142
143
144 def write_results(results):
145     fp_result = open('test_result.txt', 'w')
146     for result in results:
147         fp_result.write(str(result[0]))
148         fp_result.write(' ')
149         fp_result.write(result[1])
150         fp_result.write('\n')
151     fp_result.close()
152
153
154 def handel_result(results):
155
156     pos_number, neg_number, mid_number, number_ratio = 0, 0, 0, 0
157
158     pos_mean, neg_mean, mean_ratio = 0, 0, 0
159
160     pos_variance, neg_variance, var_ratio = 0, 0, 0
161     pos_list, neg_list, middle_list, total_list = [], [], [], []
162     for result in results:
163         total_list.append(result[0])
164         if result[0] > 0:
165             pos_list.append(result[0])
166         elif result[0] < 0:
167             neg_list.append(result[0])
168         else:
169             middle_list.append(result[0])
```

text_process.py dict_main.py

```
170
171     pos_number = len(pos_list)
172     neg_number = len(neg_list)
173     mid_number = len(middle_list)
174     total_number = pos_number + neg_number + mid_number
175     number_ratio = pos_number/neg_number
176     pos_number_ratio = round(float(pos_number)/float(total_number), 2)
177     neg_number_ratio = round(float(neg_number)/float(total_number), 2)
178     mid_number_ratio = round(float(mid_number)/float(total_number), 2)
179     text_pos_number = "积极微博条数为 " + str(pos_number) + " 条，占全部微博比例的 %" + str(
180     text_neg_number = "消极微博条数为 " + str(neg_number) + " 条，占全部微博比例的 %" + str(
181     text_mid_number = "中性情感微博条数为 " + str(mid_number) + " 条，占全部微博比例的 %" +
182
183     pos_array = np.array(pos_list)
184     neg_array = np.array(neg_list)
185     total_array = np.array(total_list)
186     pos_mean = pos_array.mean()
187     neg_mean = neg_array.mean()
188     total_mean = total_array.mean()
189     mean_ratio = pos_mean/neg_mean
190     if pos_mean <= 6:
191         text_pos_mean = emotion_level4
192     else:
193         text_pos_mean = emotion_level5
194     if neg_mean >= -6:
195         text_neg_mean = emotion_level2
196     else:
197         text_neg_mean = emotion_level1
198     if total_mean <= 6 and total_mean >= -6:
199         text_total_mean = emotion_level3
200     elif total_mean > 6:
201         text_total_mean = emotion_level4
202     else:
203         text_total_mean = emotion_level2
```

```
        text_total_mean = emotion_level2

    pos_variance = pos_array.var(axis=0)
    neg_variance = neg_array.var(axis=0)
    total_variance = total_array.var(axis=0)
    var_ratio = pos_variance/neg_variance
    #print "pos_variance:", pos_variance, "neg_variance:", neg_variance, "var_ration:",
    if total_variance > 10:
        text_total_var = emotion_level7
    else:
        text_total_var = emotion_level6
    ################################
    result_dict = {}
    result_dict['pos_number'] = pos_number
    result_dict['neg_number'] = neg_number
    result_dict['mid_number'] = mid_number
    result_dict['number_ratio'] = round(number_ratio, 1)
    result_dict['pos_mean'] = round(pos_mean, 1)
    result_dict['neg_mean'] = round(neg_mean, 1)
    result_dict['total_mean'] = round(total_mean, 1)
    result_dict['mean_ratio'] = abs(round(mean_ratio, 1))
    result_dict['pos_variance'] = round(pos_variance, 1)
    result_dict['neg_variance'] = round(neg_variance, 1)
    result_dict['total_variance'] = round(total_variance, 1)
    result_dict['var_ratio'] = round(var_ratio, 1)

    result_dict['text_pos_number'] = text_pos_number
    result_dict['text_neg_number'] = text_neg_number
    result_dict['text_mid_number'] = text_mid_number
    result_dict['text_pos_mean'] = text_pos_mean
    result_dict['text_neg_mean'] = text_neg_mean
    result_dict['text_total_mean'] = text_total_mean
    result_dict['text_total_var'] = text_total_var
    """
```

```
if __name__ == '__main__':
    results = run_score()
    write_results(results)
    result_dict = handel_result(results)
```

3.2 TF-IDF

It is processed on the google colab: https://colab.research.google.com/notebooks/welcome.ipynb

```
#server connection
!apt-get install -y -qq software-properties-common python-software-properties modul
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={cre

#direction
!ls

!mkdir -p drive
!google-drive-ocamlfuse drive
!ls

# go the the working directory
import os

os.chdir("drive/paper")
!ls

#keyword extraction

import jieba
import math
import jieba.analyse
```

Code:

```
import  math
import  jieba.analyse

class  TF_IDF:

        def  __init__(self,file,stop_file):
                self.file  =  file
                self.stop_file  =  stop_file
                self.stop_words  =  self.getStopWords()

        def  getStopWords(self):
                swlist=list()
                for  line  in  open(self.stop_file,"r",encoding="utf-8").readlines():
                        swlist.append(line.strip())
                print("stop  words  loaded...")
                return  swlist

        def  loadData(self):
                dMap  =  dict()
                for  line  in  open(self.file,"r",encoding="utf-8").readlines():
                        id,title  =  line.strip().split("\t")
                        dMap.setdefault(id,  [])
                        for  word  in  list(jieba.cut(str(title).replace("  ",""),  cut_all=
                                if  word  not  in  self.stop_words:
                                        dMap[id].append(word)
                print("load  products  and  its  title,  using  jieba  to  segment  and  del
                return  dMap

        def  getFreqWord(self,words):
                freqWord  =  dict()
                for  word  in  words:
                        freqWord.setdefault(word,0)
                        freqWord[word]  +=  1
                return  freqWord
```

3.3 Outlier process:

```
                for key in dMap.keys():
                        if word in dMap[key]:
                                count += 1
                return count

        def getTFIDF(self,words,dMap):
                outDic = dict()
                freqWord = self.getFreqWord(words)
                for word in words:

                        tf = freqWord[word]*1.0 / len(words)

                        idf = math.log(len(dMap)/(self.getCountWordInFile(word,dMap)+1
                        tfidf = tf * idf
                        outDic[word] = tfidf

                orderDic = sorted(outDic.items(), key=lambda x:x[1], reverse=True)
                return orderDic

        def getTag(self,words):

                print(jieba.analyse.extract_tags(words, topK=200, withWeight=True))

if __name__ == "__main__":

        file = "./taobao_tfidf.txt"

        stop_file = "./stop_words.txt"

        tfidf=TF_IDF(file,stop_file)
        tfidf.getTag(open("./taobao_tfidf.txt","r",encoding="utf-8").read(),)
```

Using R and R studio: link: https://cran.r-project.org/bin/windows/base/, https://rstudio.com/products/rstudio/download/

Code:

Winsorizing:

```r
data <- c(756153,534383,458519,24456,31138,2617734,75549,2905154,26520,3
,374737,337104,1164074,261359,1898800,137491,486411,2178952,85561,56796,
,1480950,388998,226727,1529637,48183,1510995,144798,115717,302831,442244
,223083,509471,69630,64380,167594,301439,156588,766846,1258,213744,38219
,450207,209583,84926,861646,888573,1805145,62196,231195,220568,929129,66

data

length(data)
summary(data)
benth <-783632 + 1.5*IQR(data)
benth

data[data > benth]
data[data > benth] <-benth
data
summary(data)
boxplot(data)

```

Robust regression:

```r
install.packages("car")
install.packages("foreign")
install.packages("MASS")
library(foreign)
library(MASS)
require(MASS)
require(foreign)
library(car)
library(carData)

mydata<-read.csv("robost_regression.csv",header = T)
str(mydata)
attach(mydata)
summary(mydata)

plot(cooks.distance(lm(Sales ~ Popularity, data = mydata)))
qqnorm(Sales);qqline(Sales)

summary(ols <- lm(Sales ~ Popularity, data = mydata))

dist <- cooks.distance(ols)
dist<-data.frame(dist)
s <- stdres(ols)
a <- cbind(mydata, dist, s)

sabs <- abs(s)
a <- cbind(mydata, dist, s, sabs)
asorted <- a[order(-sabs), ]
asorted[1:10, ]

summary(rr.huber <- rlm(Sales ~ Popularity, data = mydata))

huber <- data.frame(usage = mydata$Sales, risid = rr.huber$resid, weight = rr.huber$w)
huber2 <- huber[order(rr.huber$w), ]
huber2[1:10, ]


rr.bisquare <- rlm(Sales ~ Popularity, data = mydata, psi = psi.bisquare)
summary(rr.bisquare)
bisqr <- data.frame(Sales = mydata$Sales,resid = rr.bisquare$resid, weight = rr.bisquare$w)
bisqr2 <- bisqr[order(rr.bisquare$w), ]
bisqr2[1:10, ]

```

References:

Csdn.net. (2019). [online] Available at: https://blog.csdn.net/gamer_gyt/article/details/85690389 [Accessed 12 Dec. 2019].

Csdn.net. (2019). [online] Available at: https://blog.csdn.net/chenpe32cp/article/details/77801600 [Accessed 12 Dec. 2019].

Robust Regressions in R. (2018). YouTube. Available at: https://www.youtube.com/watch?v=lU25mxM4mhs [Accessed 12 Dec. 2019].

Outlier Treatment in R - Part 2 - Winsorizing. (2016). YouTube. Available at: https://www.youtube.com/watch?v=JTgLDDyAuF0 [Accessed 12 Dec. 2019].