Contents

8.1 Analysing the Relationship Between Friends and Followers for Twitter Users

8.1.1 Retrieve the posts from Twitter

relevant posts can be retrieved from twitter by utilising the `rtweet` package, packages can be loaded for use in **R** thusly:

The `rtweet` API will search for tweets that contain all the words of a query regardless of uppercase or lowercase usage [5].

In order to leverage the *Twitter* API it is necessary to use tokens provided through a *Twitter* developer account:

and hence all tweets containing a mention of *Ubisoft* can be returned and saved to disk as shown in listing 3:

8.2.2 Count of Followers and Friends

In order to identify the number of users that are contained in the *tweets* the `unique()` function can be used to return a vector of names which can then be passed as an index to the vector of counts as shown in listing 4, this provides that 81.7% of the tweets are by unique users.

```r
# Load Packages -----------------------------------------------------------
setwd("~/Dropbox/Notes/DataSci/Social_Web_Analytics/SWA-Project/scripts/")

if (require("pacman")) {
  library(pacman)
} else{
  install.packages("pacman")
  library(pacman)
}

pacman::p_load(xts, sp, gstat, ggplot2, rmarkdown, reshape2,
               ggmap, parallel, dplyr, plotly, tidyverse,
               reticulate, UsingR, Rmpfr, swirl, corrplot,
               gridExtra, mise, latex2exp, tree, rpart,
               lattice, coin, primes, epitools, maps, clipr,
               ggmap, twitteR, ROAuth, tm, rtweet, base64enc,
               httpuv, SnowballC, RColorBrewer, wordcloud,
               ggwordcloud, tidyverse, boot)
```

Listing 1: Load the Packages for ***R***

8.1.3 Summary Statistics

The average number of friends and followers from users who posted tweets mentioning *Ubisoft* can be returned using the `mean()` as shown in listing 5 this provides that on average each user has 586 friends and 63,620 followers.

8.1.4 Above Average Followers

Each user can be compared to the average number of followers, by using a logical operator on the vector (e.g. `y > ybar`), this will return an output of logical values. ***R*** will coerce logicals into 1/0 values meaning that the mean value will return the proportion of `TRUE` responses as shown in listing 6. This provides that:

- 2.4% of the have identified have an above average **number of followers**.
- 20.6% of the users identified have an above average **number of friends**.

```r
# Set up Tokens ============================================================
 
options(RCurlOptions = list(
  verbose = FALSE,
  capath = system.file("CurlSSL", "cacert.pem", package = "RCurl"),
  ssl.verifypeer = FALSE
))

setup_twitter_oauth(
  consumer_key = "*************************",
  consumer_secret = "**************************************************",
  access_token = "**************************************************",
  access_secret = "*********************************************"
)

# rtweet ===================================================================
tk <-    rtweet::create_token(
  app = "SWA",
  consumer_key    = "*************************",
  consumer_secret = "**************************************************",
  access_token    = "**************************************************",
  access_secret   = "*********************************************",
  set_renv        = FALSE
```

Listing 2: Import the twitter tokens (redacted)

```r
 n <- 1000
 tweets.company <- search_tweets(q = 'ubisoft', n = n, token = tk,
                                 include_rts = FALSE)
 save(tweets.company[,], file = "resources/Download_1.Rdata")
```

Listing 3: Save the Tweets to the HDD as an rdata file

```
(users <- unique(tweets.company$name)) %>% length()
x <- tweets.company$followers_count[duplicated(tweets.company$name)]
y <- tweets.company$friends_count[duplicated(tweets.company$name)]

## > [1] 817
```

Listing 4: Return follower count of twitter posts

```
x<- rnorm(090)
y<- rnorm(090)
(xbar <- mean(x))
(ybar <- mean(y))

## > [1] 4295.195
## > [1] 435.9449
```

Listing 5: Determine the average number of friends and followers

```
(px_hat <- mean(x>xbar))
(py_hat <- mean(y>ybar))

## > [1] 0.0244798
## > [1] 0.2729498
```

Listing 6: Calculate the proportion of users with above average follower counts

8.1.5 Bootstrap confidence intervals

a/b.) Generate a bootsrap distribution

A bootstrap assumes that the population is an infinitely large repetition of the sample and may be produces with respect to follower counts by resampling with replacement/repetition and plotted using the `ggplot2` library as deomonstrated in listings 7 and .1 and shown in figure 1.

This shows that the population follower counts is a non-normal skew-right distribution, which is expected because the number of friends is an integer value bound by zero [6].

```r
## Resample the Data
(bt_pop <- sample(x, size = 10^6, replace = TRUE)) %>% head()

## > [1]   7 515 262 309 186 166
```

Listing 7: Bootstrapping a population from the sample.

```r
## Make the Population
bt_pop_data <- tibble("Followers" = bt_pop)
ggplot(data = bt_pop_data, aes(x = Followers)) +
  geom_histogram(aes(y = ..density..), fill = "lightblue", bins = 35,
  ↪  col = "pink") +
  geom_density(col = "violetred2") +
  scale_x_continuous(limits = c(1, 800)) +
  theme_bw() +
  labs(x = "Number of Followers", y = "Density",
       title = "Bootstrapped population of Follower Numbers")
```

c.) Estimate a Confidence Interval for the population mean Follower Counts

In order to perform a bootrap for the population mean value of follower counts it is necessary to:

1. Resample the data with replacement
   - i.e. randomly select values from the sample allowing for repetition
2. Measure the statistic of concern
3. Replicate this a sufficient number of times
   - i.e. Greater than or equal to 1000 times [2, Ch. 5]

This is equivalent to drawing a sample from a population that is infinitely large and constructed of repetitions of the sample. This can be performed in ***R*** as shown in listing 8.

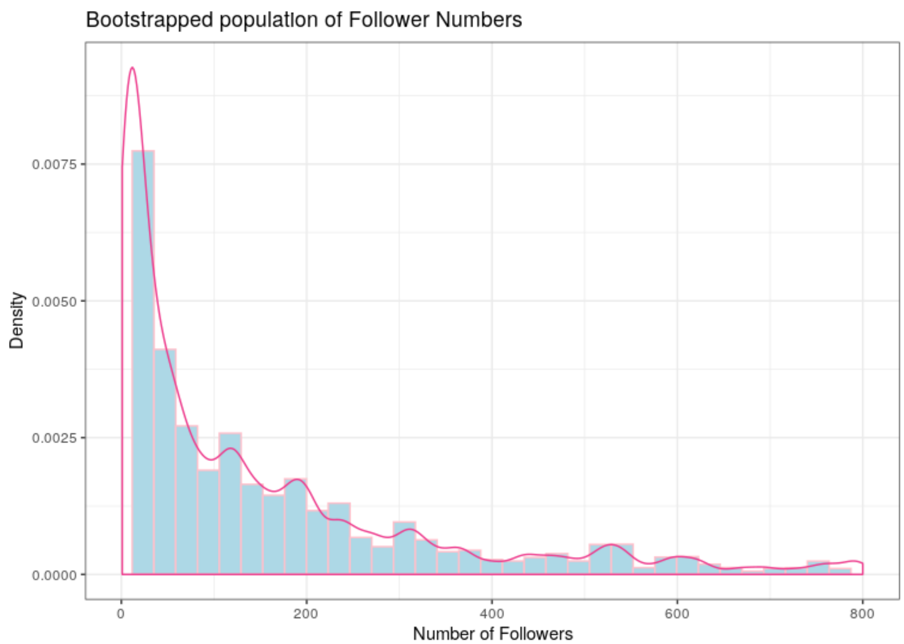

Figure 1: Histogram of the bootrapped population of follower counts

```r
xbar_boot_loop <- replicate(10^3, {
  s <- sample(x, replace = TRUE)
  mean(s)
  })
quantile(xbar_boot_loop, c((1-0.97)/2, (1+0.97)/2))

##        1.5%      98.5%
##    588.4189 10228.7352
```

Listing 8: Confidence Interval of Mean Follower Count in Population

A 97% probability interval is such that a sample drawn from a population will contain the population mean in that interval 97% of the time, this means that it may be concluded with a high degree of certainty that the true population mean lies between 588 and 10228.

1. Alternative Approaches If this data was normally distributed it may have been appropriate to consider bootstrapping the standard error and using a t distribution, however it is more appropriate to use a percentile interval for skewed data such as this, in saying that however this method is not considered to be very accurate in the literature and is often too narrow. [3, Section 4.1]

   - It's worth noting that the normal t value bootstrap offers no advantage over using a t distribution (other than being illustrative of bootstrapping generally) [3, Section 4.1]

   The boot package is a bootstrapping library common among authors in the data science sphere [4, p. 295] [8, p. 237] that implements confidence intervals consistent with work by Davison and Hinkley [7] in there texbook *Bootstrap Methods and their Application*. In this work it is provided that the BC_a method of constructing confidence intervals is superior to mere percentile methods in terms of accuracy [2, Ch. 5], a sentiment echoed in the literature. [1, 2, Ch. 5]

   Such methods can be implemented in ***R*** by passing a function to the the boot call as shown in listing 9. This provides a broader interval, providing that the true confidence interval could lie between 1079 and 16227 followers.

```r
xbar_boot <- boot(data = x, statistic = mean_val, R = 10^3)
boot.ci(xbar_boot, conf = 0.97, type = "bca", index = 1)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = xbar_boot, conf = 0.97, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 97%   ( 1079, 16227 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
## Warning message:
## In norm.inter(t, adj.alpha) : extreme order statistics used as
↪  endpoints
```

Listing 9: Bootstrap of population mean follower count implementing the BC_a method

references

d.) Estimate a Confidence Interval for the population mean Friend Counts

A Confidence interval for the population mean friend counts may be constructed in a like wise fashion as shown in listings 10. This provides that the 97% confidence interval for the population mean friend count is between 384 and 502 (or 387 and 496 if the BC_a method used, they're quite close and so the more conservative percentile method will be accepted).

FIXME 8.1.6 Estimate a 97% Confidence Interval for the High Friend Count Proportion

In order to bootstrap a confidence interval for the proportion of users with above average follower counts, repeteadly draw random samples from an infinitely large population composed entirely of the sample, and record the sampled proportion. this can be acheived by resampling the observations of above and below as shown in listing 11.

This provides that:

- The 97% confidence interval for the population proportion of users that have an above average number of friends is between 0.24 and 0.31.
    - i.e. The probability of any given sample containing the population mean within this interval would be 97%, although that doesn't however mean that there is a 97% probability that this interval contains the value, merely that we may be 97% *confident*

8.1.7 Is the Number of Friends Independent to the Number of Followers

One method to determine whether or not the number of followers is independent of the number of friends is to bin the counts and determine whether or not the distribution of users across those counts is consistent with the hypothesis of independence.

Bin the Follower and Friend Categories

The counts may be binned by performing a logical interval test as shown in listing 12.

Find the Group frequency

These values may be tabluated in order to count the occurence of users among these categories as shown in listing 13 and table 1.

Table 1: Table of Binned Friend and Follower counts, transposed relative to code.

	Followers	***Friends***
Tens	421	262
Hundreds	317	476
1 - Thousands	39	47
2 - Thousands	11	15
3 - Thousands	9	6
4 - Thousands	2	9
5 Thousand or More	18	2

```r
# d.) Estimate a Confidence Interval for the populattion mean Friend Count ===
# Using a Percentile Method #####################################################
ybar_boot_loop <- replicate(10^3, {
  s <- sample(y, replace = TRUE)
  mean(s)
  })
quantile(ybar_boot_loop, c(0.015, 0.985)

# Using BCA Method ############################################################
mean_val <- function(data, index) {
  X = data[index]
  return(mean(X))
}

xbar_boot <- boot(data = y, statistic = mean_val, R = 10^3)
boot.ci(xbar_boot, conf = 0.97, type = "bca", index = 1)


##      1.5%    98.5%
## 383.7619 501.5903
##
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = xbar_boot, conf = 0.97, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 97%   (386.8, 496.7 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

Listing 10: Bootstrap of population mean follower count

```r
# 8.1.6 High Friend Count Proportion
 ↪  -------------------------------------------
prop <- factor(c("Below", "Above"))
## 1 is above average, 2 is below
py_hat_bt <- replicate(10^3, {
  rs      <- sample(c("Below", "Above"),
                    size = length(y),
                    prob = c(py_hat, 1-py_hat),
                    replace = TRUE)
isabove <- rs == "Above"
mean(isabove)
})
quantile(py_hat_bt, c(0.015, 0.985))


##      1.5%     98.5%
## 0.2399021 0.3072215
## > > > . + > > >
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = py_hat_boot, conf = 0.97, type = "bca")
##
## Intervals :
## Level       BCa
## 97%   ( 0.2399,  0.3072 )
## Calculations and Intervals on Original Scale
```

Listing 11: Bootstrap of Proportion of Friends above average

```r
## Assign Categories
x_df <- data.frame(x)
x_df$cat[0       <= x_df$x & x_df$x < 100] <- "Tens"
x_df$cat[100     <= x_df$x & x_df$x < 1000] <- "Hundreds"
x_df$cat[1000    <= x_df$x & x_df$x < 2000] <- "1Thousands"
x_df$cat[2000    <= x_df$x & x_df$x < 3000] <- "2Thousands"
x_df$cat[3000    <= x_df$x & x_df$x < 4000] <- "3Thousands"
x_df$cat[4000    <= x_df$x & x_df$x < 5000] <- "4Thousands"
x_df$cat[5000    <= x_df$x & x_df$x < Inf] <- "5ThousandOrMore"

### Make a factor
x_df$cat <- factor(x_df$cat, levels = var_levels, ordered = TRUE)

### Determine Frequencies
(x_freq <- table(x_df$cat) %>% as.matrix())

## ** b) Find the Friend Count Frequency ===========================================
## Assign Categories
y_df <- data.frame(y)
y_df$cat[0       <= y_df$y & y_df$y < 100] <- "Tens"
y_df$cat[100     <= y_df$y & y_df$y < 1000] <- "Hundreds"
y_df$cat[1000    <= y_df$y & y_df$y < 2000] <- "1Thousands"
y_df$cat[2000    <= y_df$y & y_df$y < 3000] <- "2Thousands"
y_df$cat[3000    <= y_df$y & y_df$y < 4000] <- "3Thousands"
y_df$cat[4000    <= y_df$y & y_df$y < 5000] <- "4Thousands"
y_df$cat[5000    <= y_df$y & y_df$y < Inf]  <- "5ThousandOrMore"

### Make a factor
y_df$cat <- factor(y_df$cat, levels = var_levels, ordered = TRUE)

### Determine Frequencies
(y_freq <- table(y_df$cat) %>% as.matrix())
```

Listing 12: Use Logical Test to Assign observations into bins

```
vals <- t(cbind(x_freq, y_freq))
rownames(vals) <- c("Followers.x", "followers.y")
vals

##               Tens Hundreds 1Thousands 2Thousands 3Thousands 4Thousands
## Followers.x    421      317         39         11          9          2
## followers.y    262      476         47         15          6          9
##               5ThousandOrMore
## Followers.x                18
## followers.y                 2
```

Listing 13: Tabulate the binned counts for the distribution of users among among amount and status.

Find the Expected Counts under each group and test for independence

The expected count of each cell, under the assumption that the two metrics are independent, will be the proportion users per bracket multiplied by the number of users in that status group. This implies that any cell will be:

- the product of the row sum, multiplied by the column sum divided by the number of counts.

This can be equivalently expressed as an outer product as shown in equation (1), in ***R*** this operation is denoted by the %o% operator, which is shorthand for the `outer()` function, this and other summary statistics may be evaluated as shown in listing 14.

The outer product is such that:

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^{\mathsf{T}} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}.$$

This means the matrix of expected frequencies can be expressed as an outer product thusly:

$$\tilde{\mathbf{e}} = \frac{1}{n} \times \begin{bmatrix} \sum_{j=1}^{n} [o_{1j}] \\ \sum_{j=1}^{n} [o_{2j}] \\ \sum_{j=1}^{n} [o_{3j}] \\ \sum_{j=1}^{n} [o_{4j}] \\ \vdots \\ \sum_{j=1}^{n} [o_{nj}] \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{n} [o_{i1}] \\ \sum_{j=1}^{n} [o_{i2}] \\ \sum_{j=1}^{n} [o_{i3}] \\ \cdots \\ \sum_{j=1}^{n} [o_{in}] \end{bmatrix}^{\mathrm{T}} \tag{1}$$

1. Testing Independence In order to test whether or not the distribution of users among brackets is independent of being a follower or friend a χ^2 test may be used, this can be evaluated from a model or simulated, in ***R***, the simulated test is shown in listing 15, this provides a p -value $<$ 0.0005, which means that the hypothesis of independence may be rejected with a high degree of certainty.

```
## ***** Calculate Summary Stats
n <- sum(vals)
bracket_prop <- colSums(vals) / n
metric_prop  <- rowSums(vals) / n
o <- vals
e <- rowSums(vals) %o% colSums(vals) / n
chi_obs <- sum((e-o)^2/e)
```

Listing 14: Calculate Expected frequency of values under the assumption of independence.

```
chisq.test(vals, simulate.p.value = TRUE)


## ^^IPearson's Chi-squared test with simulated p-value (based on 2000
## ^^Ireplicates)
##
## data:  vals
## X-squared = 88.109, df = NA, p-value = 0.0004998
```

Listing 15: Chi-Square testing for independence between friend and follower bin categories.

(a) From First Principles The χ^2 statistic may be performed from first principles by randomly sampling the values at the rate at which they occured, tabulating those counts, measuring the χ^2 -value and then repeating this many times.

    Because the samples are random they must be independent and average number of positives is hence an estimate for the *FPR*, which is in turn an estimate for the p -value. This technique is demonstrated in listing 16, the p-value being returned as 0.0004, this value is consistent with the value produced by ***R***'s built in `chisq.test` function and so is accepted.

FIXME Conclusion

The p -value measures the probability of rejecting the null hypothesis when it is true, i.e. the probability of a detecting a *false positive*, a very small p -value is hence good evidence that the null hypothesis should be rejected (because doing so would unlikely to be a mistake).

In saying that however the p -value is distinct from the *power* statistic, which is a measure of */the probability of accepting the alternative hypothesis* when it is true, a low p -value is not a measurement of the probability of being correct.

Hence me way conclude, with a high degree of certainty, that the follower and friend counts are not independent of one another.

8.2 Finding Themes in tweets

In order to clean the corpus it will be necessary to:

```r
## ***** Create Vectors of factor levels
brackets <- unique(x_df$cat)
metrics <- c("follower", "friend")

## ***** Simulate the data Assuming H_0
## I.e. assuming that the null hypothesis is true in that
## the brackets assigned to followers are independent of the friends
## (this is a symmetric relation)

s <- replicate(10^4,{
  ## Sample the set of Metrics
  m <- sample(metrics, size = n, replace = TRUE, prob = metric_prop)

  ## Sample the set of Brackets (i.e. which performance bracket the
  ↪  user falls in)
  b <- sample(brackets, size = n, replace = TRUE, prob = bracket_prop)

  ## Make a table of results
  o <- table(m, b)
  o

  ## Find What the expected value would be
  e_sim <- t(colSums(e) %o% rowSums(e) / n)

  ## Calculate the Chi Stat
  chi_sim <- sum((e_sim-o)^2/e_sim)
  chi_sim

  ## Is this more extreme, i.e. would we reject null hypothesis?
  chi_sim > chi_obs

})

mean(s)
```

Listing 16: Performing a χ^2 statistic from first principles

1. remove numbers
2. remove punctuation
3. remove whitespace
4. case fold all characters to lower case
5. remove a set of stop words
6. reduce each word to its stem

Should Emoji's be removed

These studies suggest that

Emoji's are variables that can be used as a predictive features [**lecompte2017**] and can improve the performance of Sentiment Analysis Models [**shiha2017**] that use the *Bag of Words Approach*

References

[1] James Carpenter and John Bithell. "Bootstrap Confidence Intervals: When, Which, What? A Practical Guide for Medical Statisticians". en. In: *Statistics in Medicine* 19.9 (2000), pp. 1141–1164. ISSN: 1097-0258. DOI: 10.1002/(SICI)1097-0258(20000515)19:9<1141::AID-SIM479>3.0.CO;2-F. URL: https://doi-org.ezproxy.uws.edu.au/10.1002/(SICI)1097-0258(20000515)19:9%3C1141::AID-SIM479%3E3.0.CO;2-F (visited on 04/27/2020) (cit. on p. 6).

[2] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge ; New York, NY, USA: Cambridge University Press, 1997. ISBN: 978-0-521-57391-7 978-0-521-57471-6 (cit. on pp. 5, 6).

[3] Tim C. Hesterberg. "What Teachers Should Know About the Bootstrap: Resampling in the Undergraduate Statistics Curriculum". In: *The American Statistician* 69.4 (Oct. 2015), pp. 371–386. ISSN: 0003-1305. DOI: 10.1080/00031305.2015.1089789. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4784504/ (visited on 04/26/2020) (cit. on p. 6).

[4] Gareth James et al., eds. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics 103. OCLC: ocn828488009. New York: Springer, 2013. ISBN: 978-1-4614-7137-0 (cit. on p. 6).

[5] Michael Kearney. *Get Tweets Data on Statuses Identified via Search Query. Search_tweets*. en. Manual. 2019. URL: https://rtweet.info/reference/search_tweets.html (visited on 04/26/2020) (cit. on p. 1).

[6] NIST. *1.3.3.14.6. Histogram Interpretation: Skewed (Non-Normal) Right*. Oct. 2013. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/histogr6.htm (visited on 04/26/2020) (cit. on p. 4).

[7] Brian Ripley. *Boot.Ci Function | R Documentation*. Apr. 2020. URL: https://www.rdocumentation.org/packages/boot/versions/1.3-25/topics/boot.ci (visited on 04/27/2020) (cit. on p. 6).

[8] Matt Wiley and Joshua Wiley. *Advanced R Statistical Programming and Data Models*. New York, NY: Springer Berlin Heidelberg, 2019. ISBN: 978-1-4842-2871-5 (cit. on p. 6).