Trolls regression analysis

This notebook walks through the analysis of the ~1.2 million tweet database consisting of tweets from IRA-linked twitter accounts and from accounts belonging to the legitimate news sources from which the IRA drew. The analysis uses meta-data provided by IBM Watson's Natural Language Understanding (NLU) service. In particular, the analysis uses, for each tweet, the sentiment score (a number in the interval [-1,1]), and five emotion scores (anger, disgust, fear, joy, and sadness, each a number in the interval [0,1]).

The analysis here uses cluster-robust standard errors to perform z-tests on each of the regression coefficients. The clusters used for this purpose are the stratification levels, which are location-month pairs. We find that IRA tweets are associated with lower sentiment, lower joy, higher anger, higher disgust, higher fear, and higher sadness than the control tweets.

First, we clear the workspace and define some helper functions which will be used to transform the Watson NLU scores from their bounded intervals to unbounded real numbers. `sentiment01()` will convert the sentiment score from the interval [-1,1] to the interval [0,1]. `logit_transform()` will be used to perform the logit transform $x \to \log\left(\frac{x}{1-x}\right)$ on each of the transformed sentiment score and the five emotion scores. In addition, prior to performing the logit transform, `logit_transform` shrinks each score from the interval [0,1] to the interval $[\epsilon, 1-\epsilon]$, where ϵ is the machine epsilon. This ensures that all resulting logit-transformed values are finite.

```r
rm(list=ls()) # clear workspace

# Helper functions
logit_transform <- function(p){
  # This function performs a logit transform after shrinking a
  # [0,1] input to [epsilon,1-epsilon] for tiny epsilon.
  # The logit transform is to remove the boundaries of the input,
  # so that it is not confined to [0,1]. Thus it more closely
  # approximates the normal.
  # The shrinking is because logit(0) and logit(1) give -Inf, Inf.
  epsilon <- .Machine$double.eps
  shrunkp <- p * (1-2*epsilon) + epsilon
  return(log(shrunkp/(1-shrunkp)))
}
sentiment01 <- function(s) {(s+1)/2} # Put sent. score on 0-1 scale
```

Loading and examining the data

Next we load in the data and take a look at it.

```r
# Load in the data
dat <- read.csv('./2020-02-26_tweets-and-nlu-results_complete-set.csv',
  colClasses=c("tweet_id"="character"))
head(dat)
```

```
##              tweet_id is_ira                date year month twitter_account
## 1 671478900036210690      1 2015-12-01 00:00:00 2015    12    nyc_everyday
## 2 671479367487217665      1 2015-12-01 00:02:00 2015    12    nyc_everyday
## 3 671482584908488704      1 2015-12-01 00:14:00 2015    12    nyc_everyday
## 4 671495873369518080      1 2015-12-01 01:07:00 2015    12    nyc_everyday
## 5 671500883214000129      1 2015-12-01 01:27:00 2015    12    nyc_everyday
## 6 671503389717479424      1 2015-12-01 01:37:00 2015    12    nyc_everyday
##                                                                           text
## 1             NYC seeks new vendor to operate McCarren Park ice rink  #NewYork
## 2                                            Celebrity honeymoon photos  #showbiz
## 3 The Neediest Cases: A Harlem Manâ\200\231s First Steps Toward Self-Reliance  #news
## 4                   #politics Paris climate talks creates big carbon footprint
## 5                   Syrian Family of 7 Is Quickly Settled in New Jersey  #news
## 6                         #politics Ala. lawmaker warns of non-American stores
##   stratify nlu_status language n_char sentiment_label sentiment_score
## 1       15       pass       en     64         neutral        0.000000
## 2       15       pass       en     36        positive        0.698621
## 3       15       pass       en     74         neutral        0.000000
## 4       15       pass       en     58         neutral        0.000000
## 5       15       pass       en     58         neutral        0.000000
## 6       15       pass       en     52         neutral        0.000000
##   emotion_anger emotion_disgust emotion_fear emotion_joy emotion_sadness
## 1      0.011416        0.026089     0.071247    0.699065        0.129425
## 2      0.027618        0.283126     0.021730    0.520235        0.142678
## 3      0.100427        0.226664     0.082296    0.296920        0.287729
## 4      0.094829        0.106908     0.265659    0.110157        0.294323
## 5      0.042851        0.218382     0.191767    0.352556        0.199474
## 6      0.239378        0.329806     0.109264    0.004859        0.395757
```

Next we check for missing values.

```
# Check for NAs
data.frame(names(dat),apply(dat,2,function(x) sum(is.na(x))))
```

```
##                      names.dat. apply.dat..2..function.x..sum.is.na.x...
## tweet_id               tweet_id                                        0
## is_ira                   is_ira                                        0
## date                       date                                        0
## year                       year                                        0
## month                     month                                        0
## twitter_account twitter_account                                        0
## text                       text                                        0
## stratify               stratify                                        0
## nlu_status           nlu_status                                        0
## language               language                                        0
## n_char                   n_char                                      813
## sentiment_label sentiment_label                                        0
## sentiment_score sentiment_score                                      932
## emotion_anger     emotion_anger                                   102537
## emotion_disgust emotion_disgust                                   102537
## emotion_fear       emotion_fear                                   102537
## emotion_joy         emotion_joy                                   102537
## emotion_sadness emotion_sadness                                   102537
```

We see that sentiment score is missing from 932 cases, and emotion scores are missing from 102,537 cases. We now investigate these cases further. In the following code block, we examine the language of these tweets with missing emotion data, on the theory that Watson's failure to supply emotion scores may be due to these being in a language with which Watson is not compatible. And indeed we see that many of these tweets are non-english, and many of them are such that Watson was unable to determine their language. However, the vast of them are in English.

```
table(dat$language[is.na(dat$emotion_anger)])
```

```
##
##              ar      de      en      es      fr      it      ja      nl      pt      ru
##    813        1      53  101473      71      42      24       2      24      33       1
```

Watson's output includes `nlu_status`, which is an indicator for whether Watson NLU was successfully completed on the tweet. We therefore check `nlu_status` for the tweets with missing emotion scores:

```
table(dat$nlu_status[is.na(dat$emotion_anger)])
```

```
##
##   fail   pass
##    813 101724
```

We see that many of these cases of missing emotion scores correspond to failed Watson NLU (the same number, in fact, as were not assigned a language), but the overwhelming majority of them are scored as `pass`. It is unclear why there would be missing emotion scores for these tweets, given that they pass Watson NLU.

As a final method of examinining these tweets in English with `nlu_status = pass` and missing emotion scores, we can read a sample of them directly, in the below code:

```
print(head(dat$text[is.na(dat$emotion_anger) & dat$language=='en' & dat$nlu_status=='pass']))
```

```
## [1] George W. Bush campaigns for brother @JebBush https://t.co/5knMF5ryZc https://t.co/OkurYITFX8
## [2] Star Wars: Episode VIII starts filming: https://t.co/MO9Zn4m9ue
## [3] #FIRSTALERT: Track the snow and freezing rain with our LIVE RADAR: https://t.co/McEopk2Bkm https
## [4] #FIRSTALERT: Be careful on roads, freezing rain falling in Philly area, creating slippery condit
## [5] Antique firearms stolen from Pa. Civil War museum: https://t.co/pXjNz90HQB
## [6] .@Pontifex celebrates Mexico's indigenous people: https://t.co/5uDy4fyzYr https://t.co/yGL1NAmkk
## 1199444 Levels: \177I remember being sent to my room for being bad..I was so happy..no one could both
```

```
print(tail(dat$text[is.na(dat$emotion_anger) & dat$language=='en' & dat$nlu_status=='pass']))
```

```
## [1] 7 facts about the 2016 summer solstice, the official start of summer https://t.co/1Kx2WJj0JH http
## [2] 7 facts about the 2016 summer solstice, the official start of summer https://t.co/1Kx2WJj0JH http
## [3] Woman killed after car hits tree on Garden State Parkway, police said https://t.co/qCiYkl23CL ht
## [4] Woman killed after car hits tree on Garden State Parkway, police said https://t.co/qCiYkl23CL ht
## [5] Brick Township H.S. seniors go out on a high (PHOTOS) https://t.co/Gx80KMV1fA https://t.co/xsnSu
## [6] Brick Township H.S. seniors go out on a high (PHOTOS) https://t.co/Gx80KMV1fA https://t.co/xsnSu
## 1199444 Levels: \177I remember being sent to my room for being bad..I was so happy..no one could both
```

Preprocessing

There is no obvious reason why these tweets should lack emotion scores. It remains mysterious why these tweets lack emotion scores. They amount to roughly 8% of the total tweets. For now, we remove them. It would be preferable to explain their lack of emotion scores.

```
dat <- na.omit(dat)
```

The data frame contains many columns we don't need, which require a great deal of computer memory. To ease the later computations, we delete some unneeded columns:

```
keepers <- c("tweet_id","is_ira","stratify","sentiment_score",
             "emotion_anger","emotion_disgust","emotion_fear",
             "emotion_joy","emotion_sadness")
dat <- dat[,keepers]
```

Some tweets are in multiple stratify groups. We can see how many using the following code.

```r
print(dim(dat)[1]-dim(unique(dat[,names(dat)!='stratify']))[1])
```

```
## [1] 61036
```

If every tweet were in only a single stratify group, we could use ordinary one-hot encoding for the stratify feature. Since some tweets are in multiple groups, we will instead use "n-hot encoding", as follows. We will create a column in our data frame for each of the stratify groups. For each tweet, for each stratify column, the value of the element will be 1 if the tweet is in the corresponding stratify group, and 0 otherwise. Note that this means that tweets that are in multiple stratify groups will have 1 entries in multiple of these stratify columns. The following code performs this "n-hot encoding", after first saving the stratify labels in a separate `dat.stratify`.

```r
# Save the stratify labels and tweet_id
dat.stratify <- dat[,c('tweet_id','stratify')]

# Create all the dummy columns. This takes a while
library(dummies)
dat <- dummy.data.frame(dat,names="stratify",sep="_",verbose=TRUE)

# Aggregate stratify columns by sum, grouped by tweet_id
library(dplyr)

dat <- dat %>% group_by(tweet_id,is_ira,sentiment_score,
                              emotion_anger,emotion_disgust,emotion_fear,
                              emotion_joy,emotion_sadness) %>%
  summarise_at( names(dat)[grepl("stratify",names(dat))] ,sum)
```

Earlier attempts at analysis ran afoul of what was eventually found to be a duplicate stratify label. Existence of duplicate proven here, after which we correct the problems by simply deleting one of the two columns in question.

```r
ds226 <- dat$tweet_id[dat$stratify_226==1]
ds240 <- dat$tweet_id[dat$stratify_240==1]
print(all(ds226==ds240)) # TRUE
```

```
## [1] TRUE
```

```r
rm(ds226,ds240)

# Remove group 240
dat <- dat[,!names(dat)=='stratify_240']
```

Since we're storing stratify group labels in `dat.stratify`, we need to update that data frame as well, so that it matches row-by-row with the transformed version of `dat` that we created earlier. We do that now:

```r
dat.stratify <- dat.stratify %>% group_by(tweet_id) %>%
  summarise_at( "stratify" , first)
# Make sure we didn't pick up any 240's:
print(sum(dat.stratify$stratify==240))
```

```
## [1] 0
```

Now we perform the transformations of sentiment score and the emotion scores from their original [-1,1] and [0,1] bounds, to have support over all real numbers. See above description of the logit transform for details.

```r
# Convert sentiment score to logit scale
dat$sentiment_score <-
  logit_transform(sentiment01(dat$sentiment_score))
```

```
# Convert each emotion score to logit scale
dat$emotion_anger <- logit_transform(dat$emotion_anger)
dat$emotion_disgust <- logit_transform(dat$emotion_disgust)
dat$emotion_fear <- logit_transform(dat$emotion_fear)
dat$emotion_joy <- logit_transform(dat$emotion_joy)
dat$emotion_sadness <- logit_transform(dat$emotion_sadness)
head(as.data.frame(dat[,1:8]))
```

```
##             tweet_id is_ira sentiment_score emotion_anger emotion_disgust
## 1 671478846592520192      0        0.000000    -2.1441516     -0.95196409
## 2 671478880612454400      0        0.000000    -0.8210350     -3.28357448
## 3 671478881572880386      0       -3.338508    -1.8959988     -1.64909722
## 4 671478900036210690      1        0.000000    -4.4612577     -3.61980615
## 5 671478915848781824      0       -1.118226    -0.5174744     -4.06737733
## 6 671478960253829120      1       -2.234521    -3.1310622      0.01844052
##   emotion_fear emotion_joy emotion_sadness
## 1   -1.2186222  -1.9053691      -0.1404705
## 2   -2.1185802  -1.7306927      -0.6873807
## 3   -0.8613709  -2.8683008       0.3038893
## 4   -2.5676901   0.8428494      -1.9060524
## 5   -0.9539507  -1.1139458      -0.9262703
## 6   -3.6507983  -3.5941065      -0.3701449
```

The analysis

Now we prepare to perform the regression analysis by defining our dependent variables (`targets`) and our covariates (`is_ira`, plus all the stratify dummy variables). We also load the packages needed to perform and view the analysis. Package `biglm` is necessary due to the size of the data set, which is otherwise computationally difficult to analyze. Using `biglm`, we can fit the OLS model piecemeal, using a bit of the data at a time. This produces an `biglm` object which cannot be used with standard `R` tools for cluster robust standard errors, and as a result, we also load the `bigcluster_sandwich` tools for performing cluster robust standard error analysis on a `biglm` object.

```
# Perform the analysis
targets <- c("sentiment_score","emotion_anger","emotion_disgust",
             "emotion_fear","emotion_joy","emotion_sadness")
covariates <- c(names(dat)[grep('stratify',names(dat))],"is_ira")

library(tidyr)
library(broom)
library(biglm)
library(lmtest)
source('../Downloads/bigcluster_sandwich.R')
```

Finally, the below loop performs the regression analysis. For each of the six dependent variables, the loop fits a `biglm` OLS model, then uses `BigCluster` to get the cluster robust standard errors of the regression coefficients. We use the 'HC0' estimator to obtain the cluster robust standard errors. That is: where e_i is the residual for the i^{th} observation, the estimated covariance of the regression coefficients $\widehat{\beta}$ is given by:

$$\text{Var}(\widehat{\beta}) = (X^T X)^{-1} X^T \widehat{\Omega} X (X^T X)^{-1}, \quad \widehat{\Omega} = \text{diag}(E_i, i = 1, \ldots, k)$$

where k is the number of clusters and E_i is the sample covariance of the residuals from the i^{th} cluster (for more information, see MacKinnon, J. G., & White, H. (1985)). Then `coeftest` is used, along with the output of `BigCluster`, to find the p-value for the hypothesis that the regression coefficient for the covariate `is_ira` is zero. Finally, the (very conservative) Bonferroni correction is applied to account for the fact that we are performing six simultaneous tests. Note that the clusters used here are the stratify groups themselves, which are location-month pairs.

```r
for (target in targets) {
  reg_formula = paste0(target,' ~ ',paste(covariates,collapse='+'))
  # Fit model. Too big, have to do it in chunks.
  maxrows <- 100000
  mod <- biglm(data = dat[1:maxrows,c(target,covariates)],
               as.formula(reg_formula))
  for (i in 2:floor(dim(dat)[1]/maxrows) ) {
    mod <- update(mod, dat[ ((i-1)*maxrows+1):(i*maxrows) ,
                            c(target,covariates)])
  }
  mod <- update(mod, dat[(i*maxrows+1):(dim(dat)[1]),
                         c(target,covariates)])

  # Now get vcov matric for cluster robust se's:
  bigvcov <- BigCluster(mod, as.data.frame(dat.stratify$stratify),
                        dataFrame=as.data.frame(
                          dat[,c(target,covariates)]), sumMethod = "slow")

  # Get Bonferroni correction of is_ira p-values

  cat(paste(paste(rep('=',60),collapse = ""),
            "\n\nREGRESSION RESULTS FOR ",target,":\n",
            paste(rep('=',60),collapse = ""),"\n",sep=""))
  coef_isira <- coef(mod)[['is_ira']]
  cat(paste("REGRESSION COEFFICIENT FOR is_ira: ",
            signif(coef_isira,digits = 3),"\n",sep=""))
  pval <- coeftest(mod,bigvcov)['is_ira','Pr(>|z|)']
  bfcpval <- p.adjust(pval,"bonferroni",6)
  # Print adjusted p-values
  cat(paste("BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: ",
            signif(bfcpval,digits = 3),"\n",sep=""))


}
```

```
## ============================================================
##
## REGRESSION RESULTS FOR sentiment_score:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: -0.0507
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 0.046
## ============================================================
##
## REGRESSION RESULTS FOR emotion_anger:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: 0.245
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 1.54e-32
## ============================================================
##
## REGRESSION RESULTS FOR emotion_disgust:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: 0.465
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 6.51e-51
## ============================================================
##
## REGRESSION RESULTS FOR emotion_fear:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: 0.0971
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 0.00021
## ============================================================
##
## REGRESSION RESULTS FOR emotion_joy:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: -0.128
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 0.000349
## ============================================================
##
## REGRESSION RESULTS FOR emotion_sadness:
## ============================================================
## REGRESSION COEFFICIENT FOR is_ira: 0.271
## BONFERRONI CORRECTED, CLUSTER ROBUST p-value FOR is_ira: 1.54e-27
```

From the results, we see that `is_ira` is significant for each of our six independent variables, such that IRA tweets have lower sentiment and joy, and higher anger, disgust, fear, and sadness than the control tweets.