Project 3 - R and Python (reticulate-r) Code

Isaac Horwitz (inh2102)

12/14/2020

```
library(reticulate)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## PYTHON
#py_install('newsapi-python',pip=TRUE)
from newsapi import NewsApiClient
import pandas as pd
newsapi = NewsApiClient(api_key='fcbe5d3b724d4b4c9f4b12a1e2a22c3b')
source_choice = 'bbc-news'
headlines = newsapi.get_top_headlines(sources=source_choice,language='en')
headlines = headlines['articles']
headlines = pd.DataFrame.from_dict(headlines)
```

```
py$headlines %>% tibble() %>% mutate(source=unlist(source)[[2]]) %>% select(-source,-author)
```

```
url <- "https://api.nytimes.com/svc/archive/v1/2020/12.json?api-key=0mmGvRm2j0WuaypYAEP9zlE8Zra6lXcn"
nytimes <- jsonlite::fromJSON(url,flatten=TRUE) %>% as.data.frame()
nytimes <- tibble(nytimes[!duplicated(nytimes$response.docs._id),])
nytimes %>% select(response.docs.headline.main,response.docs.pub_date)
```

```
py$headlines %>% tibble() %>% mutate(source=unlist(source)[[2]]) %>% .[3,] %>% select(title)
# When initially running this, this produced "Inside a vaccine cold storage room"
```

```
nytimes$response.docs.headline.main[1852]
# When initially running this, this produced "The C.D.C. formally approves allowing people 16 and up to
```

```
library(tidyverse,warn.conflicts=FALSE)
library(scales,warn.conflicts=FALSE)
library(grid,warn.conflicts=FALSE)
library(gridExtra,warn.conflicts=FALSE)
options(dplyr.summarise.inform = FALSE)
setwd("~/Downloads")
set.seed(111820)
news <- read.csv("News_Final.csv") %>% tibble() # DATA ON NEWS ITEMS
```

```
private <- news %>% filter(Source %in% c('Bloomberg','Reuters','ABC News','New York Times','Business Ins
                                         'Washington Post','CNN','Wall Street Journal','CNBC','Huffingt
                                         'Reuters via Yahoo! Finance','The Hill','Financial Times','USA
                                         'Daily Caller','Los Angeles Times','Fortune','Politico','The Wa
                                         'The Verge'))
private <- sample_n(private,794)

public_list <- c('NPR','PBS NewsHour','Democracy Now!','Texas Tribune','KPBS','MinnPost','Mother Jones'
                 'KPBS San Diego','PolitiFact','The Forward','Salt Lake Tribune','The Salt Lake Tribune

public <- news %>% filter(Source %in% public_list)
```

```
## SAMPLE PRIVATE AND PUBLIC

set.seed(111820)
p_sample <- sample_n(private,400)
pub_sample <- sample_n(public,400)
setwd("~/Downloads")
pub_sample_labeled <- read_csv("public_sample.csv")
pub_sample_labeled$sentiment %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1.00   -1.00    0.00   -0.18    0.00    1.00
```

```
pub_sample_labeled$episodic %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00    0.47    1.00    1.00
```

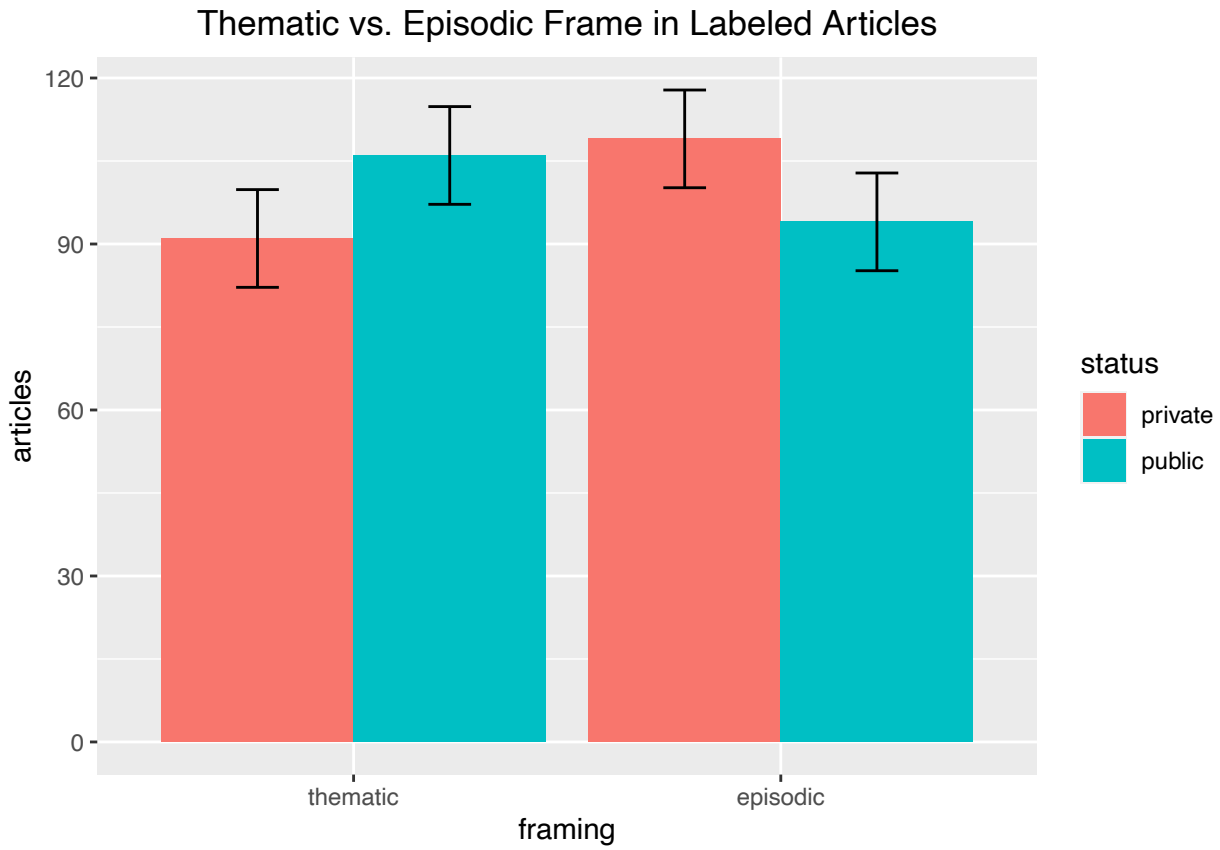
```
priv_sample_labeled <- read_csv("private_sample.csv") %>% filter(!is.na(sentiment))
priv_sample_labeled$sentiment %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1.00   -1.00    0.00   -0.19    0.00    1.00
```

```
priv_sample_labeled$episodic %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.545   1.000   1.000
```

```
episodic <- bind_rows(pub_sample_labeled,priv_sample_labeled) %>% count(episodic,status)
ggplot(data=episodic,aes(x=factor(episodic),y=n,fill=status)) +
  geom_bar(position='dodge',stat='identity') +
  labs(x='framing',y='articles',title='Thematic vs. Episodic Frame in Labeled Articles') +
  scale_x_discrete(breaks=c("0","1"),labels=c("thematic","episodic")) +
  theme(plot.title=element_text(hjust=0.5)) +
  geom_errorbar(aes(ymin=n-sd(n),ymax=n+sd(n)),
                width=0.2,position=position_dodge(0.9))
```




```
pub_and_private <- bind_rows(pub_sample_labeled,priv_sample_labeled)
m1 <- glm(data=pub_and_private,episodic~status+SentimentTitle+Topic,family='binomial')
m2 <- glm(data=pub_and_private,episodic~status,family='binomial')
chisq <- anova(m1,test='Chisq') # run for m2 as well
nullmod <- glm(data=pub_and_private,episodic~1, family="binomial")
pseduo.Rsquared <- 1-logLik(m1)/logLik(nullmod) # run for m2 as well
pseduo.Rsquared
```

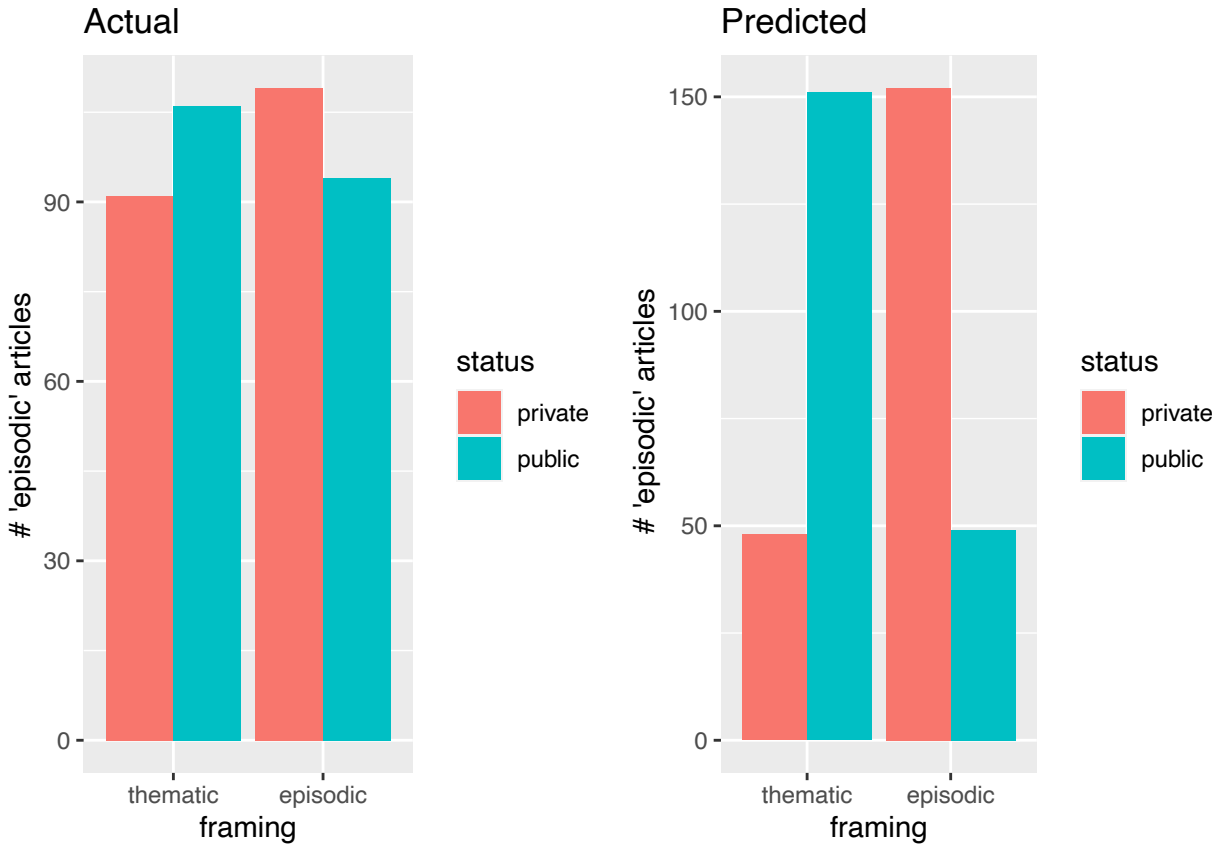
```
## 'log Lik.' 0.01974172 (df=6)
```

```
pub_and_private$pred <- predict(m1,type = "response",newdata=pub_and_private) # run for m2 as well
pub_and_private$pred_binary <- ifelse(pub_and_private$pred>=0.5,1,0) # run for m2 as well

with(pub_and_private,prop.table(table(pred_binary==episodic)))
```

```
##
## FALSE  TRUE
## 0.475 0.525
```

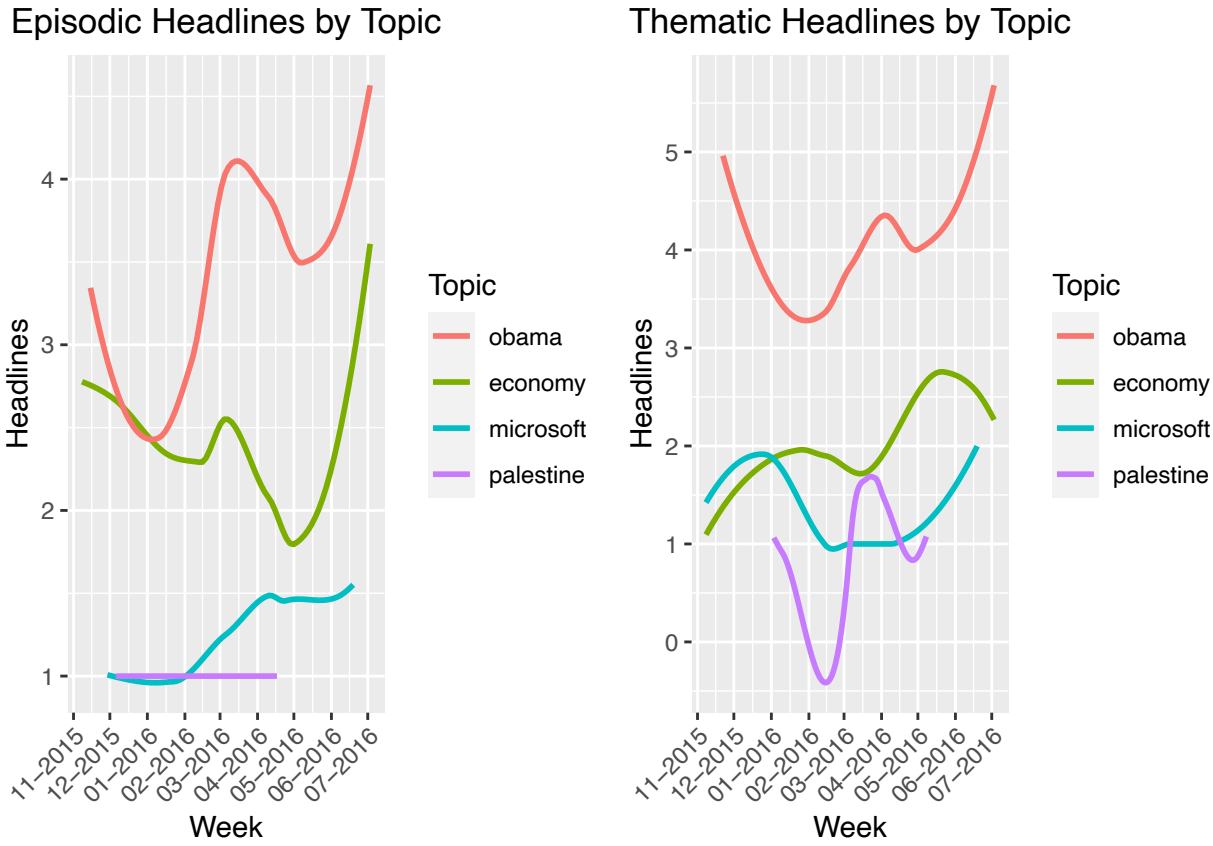
```
episodic <- pub_and_private %>% count(episodic,status)
p1 <- ggplot(data=episodic,aes(x=factor(episodic),y=n,fill=status)) +
  geom_bar(position='dodge',stat='identity') +
  labs(x='framing',y="# 'episodic' articles",title='Actual') +
  scale_x_discrete(breaks=c("0","1"),labels=c("thematic","episodic"))

pred_episodic <- pub_and_private %>% count(pred_binary,status)
p2 <- ggplot(data=pred_episodic,aes(x=factor(pred_binary),y=n,fill=status)) +
  geom_bar(position='dodge',stat='identity') +
```

```
  labs(x='framing',y=" # 'episodic' articles",title='Predicted') +
  scale_x_discrete(breaks=c("0","1"),labels=c("thematic","episodic"))
gridExtra::grid.arrange(p1,p2,nrow=1)
```

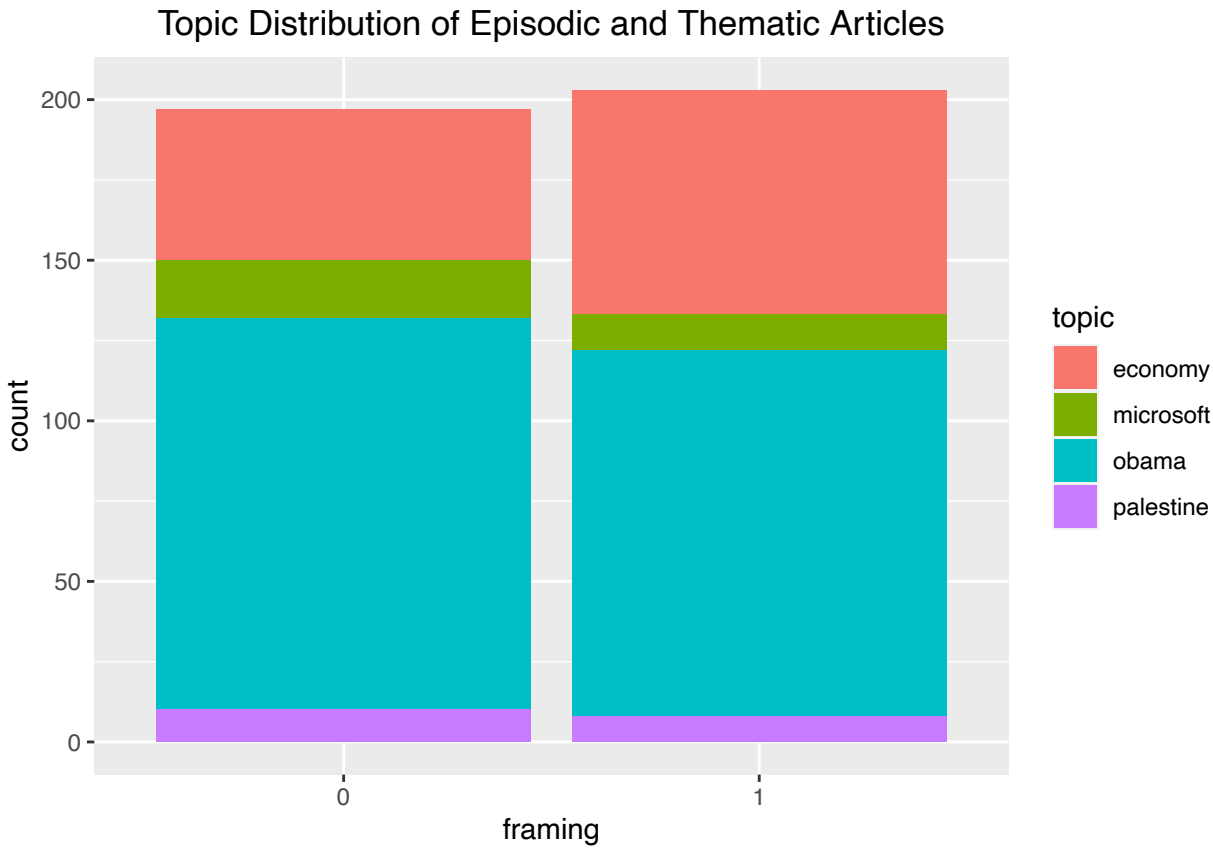



```
news["Hour"] <- substr(news$PublishDate,12,13)
news$Hour <- as.numeric(news$Hour)

episodic <- pub_and_private[pub_and_private$episodic==1,]
thematic <- pub_and_private[pub_and_private$episodic==0,]

### NEWS PER TOPIC/DAY

episodic_news_topic <- episodic[,c("PublishDate","Topic")]
episodic_news_topic$PublishDate <- lubridate::parse_date_time(episodic_news_topic$PublishDate,orders='m

thematic_news_topic <- thematic[,c("PublishDate","Topic")]
thematic_news_topic$PublishDate <- lubridate::parse_date_time(thematic_news_topic$PublishDate,orders='m

## Graph with number of news per topic per day
episodic_news_topic$Week <- lubridate::week(episodic_news_topic$PublishDate)
week <- rep(NA,nrow(episodic_news_topic))
for (i in 1:length(episodic_news_topic$Week)) {
  if (grepl("2016",episodic_news_topic$PublishDate[i])) {
  week[i] <- as.character(MMWRweek::MMWRweek2Date(MMWRyear = 2016,MMWRweek = episodic_news_topic$Week[i]
  }
  if (grepl("2015",episodic_news_topic$PublishDate[i])) {
    week[i] <- as.character(MMWRweek::MMWRweek2Date(MMWRyear = 2015,MMWRweek = episodic_news_topic$Week
```

```
  }
}
week <- as.Date(week)
episodic_news_topic$Week <- week

thematic_news_topic$Week <- lubridate::week(thematic_news_topic$PublishDate)
week <- rep(NA,nrow(thematic_news_topic))
for (i in 1:length(thematic_news_topic$Week)) {
  if (grepl("2016",thematic_news_topic$PublishDate[i])) {
  week[i] <- as.character(MMWRweek::MMWRweek2Date(MMWRyear = 2016,MMWRweek = thematic_news_topic$Week[i]
  }
  if (grepl("2015",thematic_news_topic$PublishDate[i])) {
    week[i] <- as.character(MMWRweek::MMWRweek2Date(MMWRyear = 2015,MMWRweek = thematic_news_topic$Week
  }
}
week <- as.Date(week)
thematic_news_topic$Week <- week

nrEpisodicNewsTopicDay <- episodic_news_topic %>% group_by(Topic,Week) %>% filter(PublishDate > "2015-1
nrEpisodicNewsTopicDay$Week <- as.Date(nrEpisodicNewsTopicDay$Week,origin='1970-01-01')

plot.nrEpisodicNews_daily <- ggplot(nrEpisodicNewsTopicDay,aes(x=Week,y=nrNews,group=Topic,color=fct_rec
  geom_smooth(se=F) + scale_x_date(labels=date_format("%m-%Y"),breaks=date_breaks("month")) + theme(axis
  scale_color_discrete(name = "Topic") + labs(y="Headlines",title="Episodic Headlines by Topic") + scale
  theme(plot.title=element_text(hjust=0.5))

nrThematicNewsTopicDay <- thematic_news_topic %>% group_by(Topic,Week) %>% filter(PublishDate > "2015-1
nrThematicNewsTopicDay$Week <- as.Date(nrThematicNewsTopicDay$Week,origin='1970-01-01')

plot.nrThematicNews_daily <- ggplot(nrThematicNewsTopicDay,aes(x=Week,y=nrNews,group=Topic,color=fct_rec
  geom_smooth(se=F) + scale_x_date(labels=date_format("%m-%Y"),breaks=date_breaks("month")) + theme(axis
  scale_color_discrete(name = "Topic") + labs(y="Headlines",title="Thematic Headlines by Topic") + scale
  theme(plot.title=element_text(hjust=0.5))

grid.arrange(plot.nrEpisodicNews_daily,
             plot.nrThematicNews_daily,
             ncol=2)
```




```
# very slightly modified from this for paper
plot.data <- bind_rows(pub_sample_labeled,priv_sample_labeled) %>% count(episodic,Topic)
ggplot(data=plot.data,aes(x=factor(episodic),y=n,fill=Topic)) +
  geom_bar(position='stack',stat='identity') +
  labs(x='framing',y='count',title='Topic Distribution of Episodic and Thematic Articles',fill='topic') +
  scale_x_discrete(breaks=c("0","1"),labels=c("0","1")) +
  theme(plot.title=element_text(hjust=0.5))
```




```
summarytools::freq(pub_sample_labeled$Source, report.nas = FALSE, totals = FALSE,
     cumul = FALSE, headings = FALSE,order='freq',display.type=FALSE)
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```
##
##                          Freq       %
## ----------------------- ------ -------
##                      NPR    121   60.50
##            PBS NewsHour     27   13.50
##            Mother Jones     14    7.00
##              PolitiFact     12    6.00
##       Salt Lake Tribune      9    4.50
##             The Forward      7    3.50
##          Democracy Now!      6    3.00
##           Texas Tribune      2    1.00
##                    KPBS      1    0.50
##          KPBS San Diego      1    0.50
```

```
summarytools::freq(priv_sample_labeled$Source, report.nas = FALSE, totals = FALSE,
     cumul = FALSE, headings = FALSE,order='freq',display.type=FALSE)
```

```
##
##                                 Freq       %
## ------------------------------ ------ -------
##                      Bloomberg     27   13.50
```

```
##                    Economic Times     13    6.50
##                            Forbes     13    6.50
##                           Reuters     13    6.50
##                          ABC News     12    6.00
##                     New York Times     11    5.50
##                   Business Insider     10    5.00
##                      New York Post     10    5.00
##                    Huffington Post      9    4.50
##                               CNBC      8    4.00
##                                CNN      7    3.50
##                          USA TODAY      7    3.50
##                Wall Street Journal      7    3.50
##                    Washington Post      7    3.50
##                           Politico      6    3.00
##                          The Verge      6    3.00
##                    Financial Times      5    2.50
##                             Fortune      5    2.50
##                  Los Angeles Times      5    2.50
##                           The Hill      5    2.50
##                     Breitbart News      4    2.00
##                       Daily Caller      3    1.50
##                   Washington Times      3    1.50
##         Reuters via Yahoo! Finance      2    1.00
##            The Wall Street Journal      2    1.00
```

```r
library(tidytext,warn.conflicts=FALSE)
library(tm,warn.conflicts=FALSE)
library(quanteda,warn.conflicts=FALSE)
library(doMC,warn.conflicts=FALSE)
library(rtweet,warn.conflicts=FALSE)
library(text2vec,warn.conflicts=FALSE)
library(glmnet,warn.conflicts=FALSE)

train_ind <- sample(seq_len(nrow(pub_and_private)),size=0.8*nrow(pub_and_private))

pub_and_private <- pub_and_private[train_ind,]
testing <- pub_and_private[-train_ind,]

pub_priv_clean <- pub_and_private %>%
  mutate(Title = tolower(Title),
         Title = gsub("[[:punct:]]", "",Title),
         Title = gsub("\\r|\\n"," ",Title),
         Title = plain_tweets(Title))
registerDoMC(cores=3)
stem_tokenizer =function(x) {
lapply(word_tokenizer(x), SnowballC::wordStem, language="en")
}
stopwords <- stopwords::stopwords()
toks <- itoken(pub_priv_clean$Title,
               tokenizer = stem_tokenizer,
               ids = pub_priv_clean$IDLink,
               progressbar = FALSE)
vocab <- create_vocabulary(toks,ngram = c(1L, 2L),stopwords=stopwords)
```

```
vectorizer = vocab_vectorizer(vocab)
dtm = create_dtm(toks, vectorizer)
dim(dtm)
```

```
## [1]  320 2663
```

```
glmnet_classifier = cv.glmnet(x = dtm, y = pub_and_private$episodic,
                              family = 'binomial',
                              # L1 penalty
                              alpha = 1,
                              # interested in the area under ROC curve
                              type.measure = "auc",
                              # 5-fold cross-validation
                              nfolds = 4,
                              # high value is less accurate, but has faster training
                              thresh = 1e-3,
                              # again lower number of iterations for faster training
                              maxit = 1e3)

coef(glmnet_classifier,glmnet_classifier$lambda.min)  %>% tidy() %>% arrange(value)-> coef_df

test_clean <- testing %>%
  mutate(Title = tolower(Title),
         Title = gsub("[[:punct:]]", "",Title),
         Title = gsub("\\r|\\n"," ",Title),
         Title = plain_tweets(Title))

test_toks <- itoken(test_clean$Title,
             tokenizer = stem_tokenizer,
             ids = test_clean$IDLink,
             progressbar = FALSE)

test_dtm = create_dtm(test_toks, vectorizer)

preds = predict(glmnet_classifier, test_dtm, type = 'class')[,1] %>% as.numeric()
testing$text_pred <- preds
real_dummy <- testing$episodic
caret::confusionMatrix(factor(real_dummy),factor(preds))
```

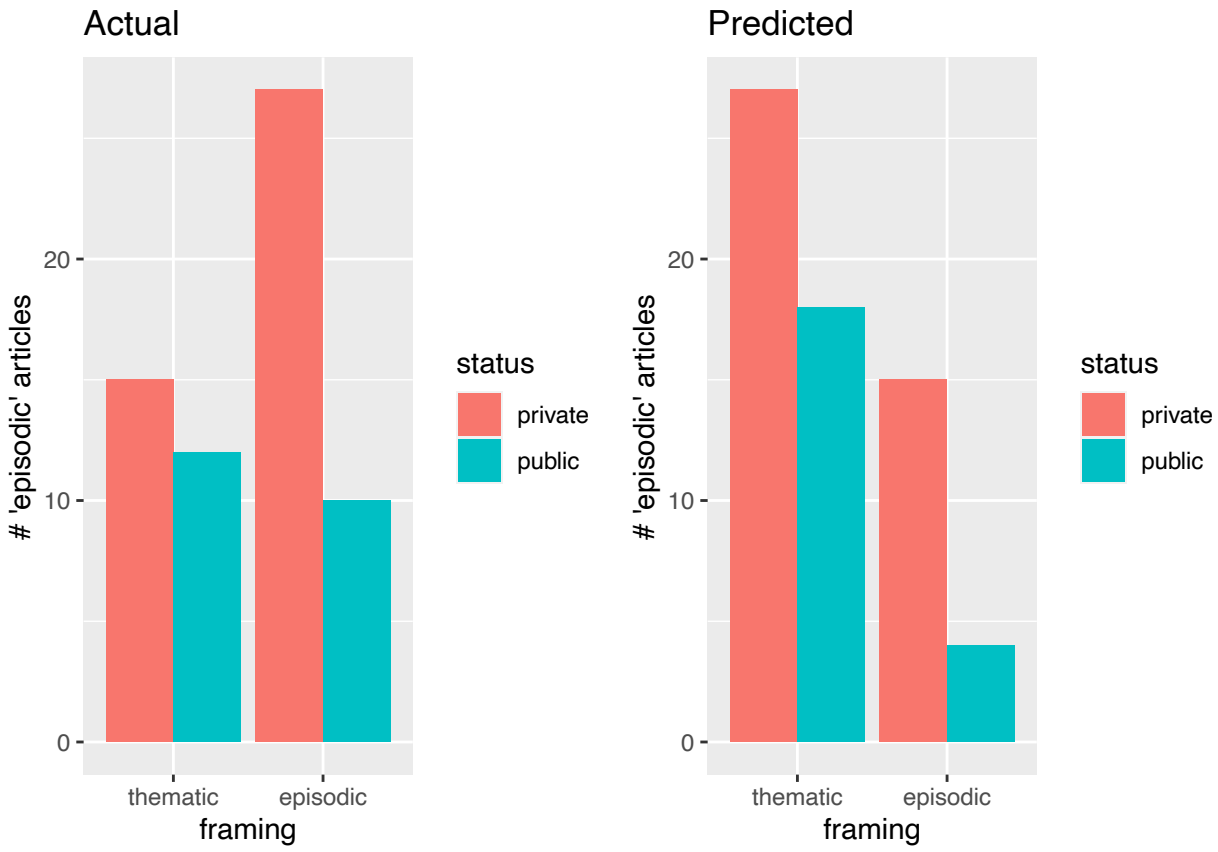
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26  1
##          1 19 18
##
##                Accuracy : 0.6875
##                  95% CI : (0.5594, 0.7976)
##     No Information Rate : 0.7031
##     P-Value [Acc > NIR] : 0.6647323
##
##                   Kappa : 0.4123
##
##  Mcnemar's Test P-Value : 0.0001439
```

```
##
##             Sensitivity : 0.5778
##             Specificity : 0.9474
##          Pos Pred Value : 0.9630
##          Neg Pred Value : 0.4865
##              Prevalence : 0.7031
##          Detection Rate : 0.4062
##    Detection Prevalence : 0.4219
##       Balanced Accuracy : 0.7626
##
##        'Positive' Class : 0
##
```

```
prop.table(table(real_dummy==preds))
```

```
##
##  FALSE   TRUE
## 0.3125 0.6875
```

```
episodic <- testing %>% count(episodic,status)
p1 <- ggplot(data=episodic,aes(x=factor(episodic),y=n,fill=status)) +
  geom_bar(position='dodge',stat='identity') +
  labs(x='framing',y="# 'episodic' articles",title='Actual') +
  scale_x_discrete(breaks=c("0","1"),labels=c("thematic","episodic"))

pred_episodic <- testing %>% count(text_pred,status)
p2 <- ggplot(data=pred_episodic,aes(x=factor(text_pred),y=n,fill=status)) +
  geom_bar(position='dodge',stat='identity') +
  labs(x='framing',y="# 'episodic' articles",title='Predicted') +
  scale_x_discrete(breaks=c("0","1"),labels=c("thematic","episodic"))
gridExtra::grid.arrange(p1,p2,nrow=1)
```


Actual
'episodic' articles
20
10
0
thematic
episodic
framing
status
private
public
Predicted
'episodic' articles
20
10
0
thematic
episodic
framing
status
private
public