yN_lie, sentiment, & text.   R

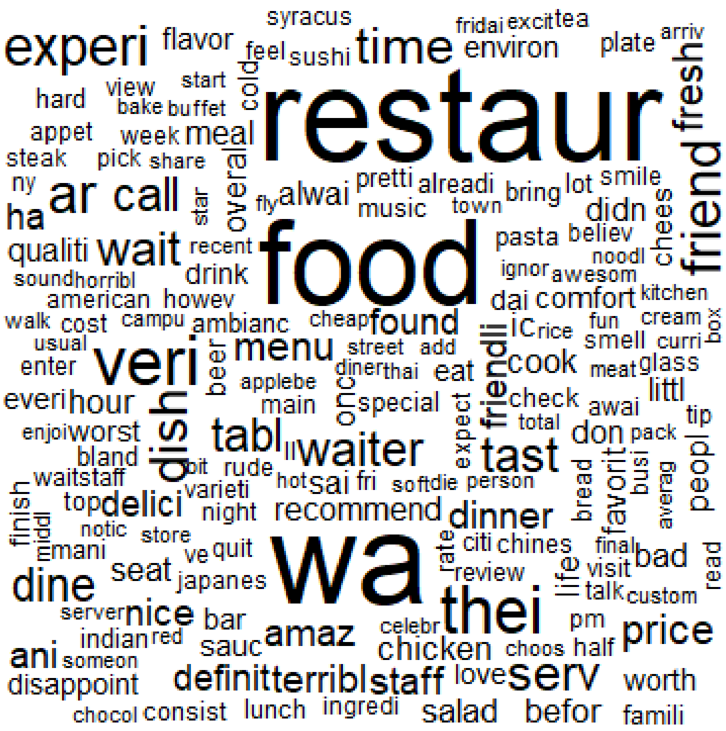
- b.hogan@snhu.edu
- data = https://github.com/bbe2/data/blob/master/a_vs_b_data_deception.csv

R Markdown

```
library(tidytext,warn.conflicts = FALSE, quietly = TRUE)
library(stringr,warn.conflicts = FALSE, quietly = TRUE)
library(dplyr,warn.conflicts = FALSE, quietly = TRUE)
library(tidyr,warn.conflicts = FALSE, quietly = TRUE)
library(wordcloud,warn.conflicts = FALSE, quietly = TRUE)
library(ggplot2,warn.conflicts = FALSE, quietly = TRUE)
options(warn= (-1) )
#===========================================================================
#==> Part 0: PreProcessing + Tokenization + Stemming + Lemmatization ====
#===========================================================================
hw8df0 <- read.csv(
  "C://Users//17574//Desktop//data_it304//a_vs_b_data_deception.csv",
                   stringsAsFactors = FALSE)
hw8df1 <- data.frame(hw8df0)
dim(hw8df1) #92x24: ## [1] 92 24

df_text <-data.frame(1:92)  #blank df
colnames(df_text) <- c("text")
as.character(df_text) ## [1] "1:92"

xtext <-as.character()
x <- 1
while (x <=92)  #put all the text & un-nessary characters into a vector
  {   yt1 <- as.character(0);yt2 <- as.character(0);yt3 <- as.character(0)
      yt4 <- as.character(0);yt4 <- as.character(0);yt6 <- as.character(0)
      yt7 <- as.character(0);yt8 <- as.character(0);yt9 <- as.character(0)
      yt10 <- as.character(0);yt11 <- as.character(0);yt12 <- as.character(0)
      yt13 <- as.character(0);yt14 <- as.character(0);yt15 <- as.character(0)
      yt16 <- as.character(0);yt17 <- as.character(0);yt18 <- as.character(0)
      yt19 <- as.character(0);yt20 <- as.character(0);yt21 <- as.character(0)
      yt22 <- as.character(0)
      yt1 <- hw8df1[x+1,3]; yt2 <- hw8df1[x+1,4]; yt3 <- hw8df1[x+1,5]
      yt4 <- hw8df1[x+1,6]; yt5 <- hw8df1[x+1,7]; yt6 <- hw8df1[x+1,8]
      yt7 <- hw8df1[x+1,9]; yt8 <- hw8df1[x+1,10]; yt9 <- hw8df1[x+1,11]
      yt10 <- hw8df1[x+1,12]; yt11 <- hw8df1[x+1,13]; yt12 <- hw8df1[x+1,14]
      yt13 <- hw8df1[x+1,15]; yt14 <- hw8df1[x+1,16]; yt15 <- hw8df1[x+1,17]
      yt16 <- hw8df1[x+1,18]; yt17 <- hw8df1[x+1,19]; yt18 <- hw8df1[x+1,20]
      yt19 <- hw8df1[x+1,21]; yt20<- hw8df1[x+1,22]; yt21 <- hw8df1[x+1,23]
      yt22 <- hw8df1[x+1,24]
      xtext <-as.character(xtext)
      xtext <- paste(yt1,yt2,yt3,yt4,yt5,yt6,yt7,yt8,yt9,yt10,yt11,yt12,yt13,
                     yt14,yt15,yt16,yt17,yt18,yt19,yt20,yt21,yt22)
      df_text[x,1] <- xtext
    x <- x+1  }
```

```
#dataframe with lie. sentiment, text
list1 <- c(1:nrow(df_text))  #create analysis numeric ID
lie_value <- rep(99, length(list1))
sent_value <- rep(99, length(list1))
reviewID <- rep(1:length(list1))
df_text <- cbind(hw8df1[,1],hw8df1[,2],df_text,reviewID,lie_value,
                 sent_value) #add lie & sentiment back
colnames(df_text) <- c("lie","sentiment","text","reviewID", "lievalue","sentv
alue")
remove(list1, lie_value, sent_value, reviewID, x)
i <- 1
while (i <=length(df_text))  #put all text & un-nessary chrs in vector
  { if (df_text[i,1]=="t") {df_text[i,5]=1}
    if (df_text[i,1]=="f") {df_text[i,5]=0}
    if (df_text[i,2]=="n") {df_text[i,6]=0}
    if (df_text[i,2]=="p") {df_text[i,6]=1}
    i <- i+1  }
#===========================================================================
#==> Part 1: STEMMING, LEMMATIZATION, AND FREQUENCY INSPECTION
#===========================================================================
review_words <- df_text %>%  #==> Tokenization
  select(lie, sentiment, text, reviewID) %>%
  unnest_tokens(word, text, to_lower=TRUE) %>%
  count(lie, sentiment, reviewID, word, sort=FALSE) %>%
  bind_tf_idf(word,reviewID,n)
# spread(key=word, value=tf_idf)
head(review_words,7)  #review of 10 words from tokens

## # A tibble: 7 x 8
##   lie   sentiment reviewID word         n     tf   idf  tf_idf
##   <fct> <fct>        <int> <chr>    <int>  <dbl>  <dbl>   <dbl>
## 1 f     n                1 a            2 0.0328 0.280  0.00919
## 2 f     n                1 also         3 0.0492 2.42   0.119
## 3 f     n                1 american     1 0.0164 3.11   0.0510
## 4 f     n                1 and          2 0.0328 0.0810 0.00265
## 5 f     n                1 are          1 0.0164 1.50   0.0247
## 6 f     n                1 area         1 0.0164 3.11   0.0510
## 7 f     n                1 back         1 0.0164 1.86   0.0305

#==>STEMMING & LEMMATIZATION
library(SnowballC)
review_words$word <- wordStem(review_words$word)
#==> REMOVE STOP WORDS ===> (6991-2941)= 4050 words removed
nrow(review_words)  #6991

## [1] 4833

review_words <- review_words %>%
  filter(!word %in% stop_words$word, str_detect(word,"^[a-z']+$"))
nrow(review_words)  #2941 <high-ish>

## [1] 2448
```

```
#==> VISUALIZATION frequency
review_words %>%
  count(word) %>%
  with(wordcloud(word, n, max.words=200))  #well that is pretty darn cool
```



```
#title("WordCloud Inspection w Data Stemming, Lemmatization & Stop Words Remo
ved")
```

```r
#============================================================================
#==>Chapter 2:TidyText - CAST_DTM=> Tidy=> Cast-Sparse=> Matrix=> DataFrame !
#============================================================================
library(tidytext,warn.conflicts = FALSE, quietly = TRUE)
review_words$reviewID <- as.character(review_words$reviewID)
mydata <- data.frame(review_words)
str(mydata)

## 'data.frame':    2448 obs. of  8 variables:
##  $ lie      : Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sentiment: Factor w/ 2 levels "n","p": 1 1 1 1 1 1 1 1 1 1 ...
##  $ reviewID : chr  "1" "1" "1" "1" ...
##  $ word     : chr  "american" "ar" "buffet" "cheap" ...
##  $ n        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ tf       : num  0.0164 0.0164 0.0164 0.0164 0.0164 ...
##  $ idf      : num  3.11 1.5 3.4 3.4 2.89 ...
##  $ tf_idf   : num  0.051 0.0247 0.0558 0.0558 0.0474 ...

mydtm <- cast_dtm(mydata, reviewID, word, n) #row is doc, col are words
df_tidy <- tidy(mydtm)   #create df and tidy
df_cast <- cast_sparse(df_tidy,document,term,count) # create matrix
dfx <- as.matrix(df_cast)
dfxx <- data.frame(dfx)  # word cube for ml
#str(dfxx)
#============================================================================
#==>Chapter 3: Machine Learn PreProcess: merge lie/sentim. Create Datasets
#============================================================================
library(caret,warn.conflicts = FALSE, quietly = TRUE) #multiple...algorithms
set.seed(199)
#use row names to merge back lie & sentiment lables
dfxx <- cbind(reviewID = rownames(dfxx),dfxx)  #fix & sort row names
dfxx$reviewID <- as.numeric(dfxx$reviewID)  #make back numeric
df_labels <- data.frame(df_text[,c(1,4)]) #1=lie, 2=sentiment, 4=reviewID
df_lie <-merge(dfxx,df_labels,key="reviewID") #only merge targe
df_labels <- data.frame(df_text[,c(2,4)]) #1=lie, 2=sentiment, 4=reviewID
df_sentiment <-merge(dfxx,df_labels,key="reviewID") #lie
labels_ID <-df_lie[,1]  #need to remove ID for running models
df_lie <- df_lie[,c(-1)]  #now reove ID as data all merged
df_sentiment <- df_sentiment[,c(-1)]  #now reove ID as data all merged
write.csv(df_lie,"today.csv")  #makes sure no duplicates on merge/funny
#Create the datasets
train_index <- createDataPartition(df_lie$lie, p=0.7, list=FALSE)
df_lie_train <-df_lie[train_index,]
df_lie_test <- df_lie[-train_index,]  #split data set training...
df_lie_test_labels <-df_lie_test$lie
df_lie_test$sentiment <- as.factor(c("?"))
df_sentiment_train <-df_sentiment[train_index,]
df_sentiment_test <- df_sentiment[-train_index,]  #split data set training...
df_sentiment_test_labels <-df_sentiment_test$sentiment
df_sentiment_test$sentiment <- as.factor(c("?"))
#str(df_sentiment_test);dim(df_sentiment_test)
```

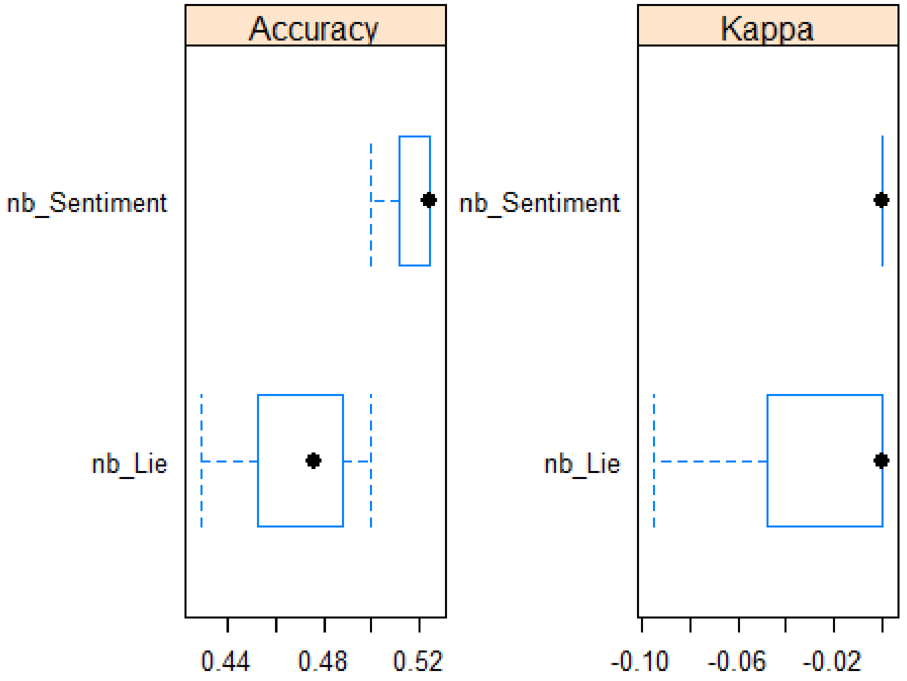
```
#===========================================================================
#==> Multinomial NB Analysis ==================
#===========================================================================
library(klaR,warn.conflicts = FALSE, quietly = TRUE)
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

mlie_nb1 <- train(lie~.,data=df_lie_train, method="nb",
               trControl = trainControl(method="cv", number=3),
           tuneGrid=expand.grid(fL=1:3,usekernel=c(TRUE, FALSE),adjust=1:3))
                                    #laplance=fL; usekernal=smoothing
mlie_predict_nb1 <- predict(mlie_nb1, newdata=df_lie_test, type="raw")
confusionMatrix(mlie_predict_nb1,df_lie_test$lie)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   f  t
##          f 13 13
##          t  0  0
##
##                Accuracy : 0.5                  =========> yikes!
##                  95% CI : (0.2993, 0.7007)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.5774905
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 1.0
##             Specificity : 0.0
##          Pos Pred Value : 0.5
##          Neg Pred Value : NaN
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 1.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : f

#sentiment
msentiment_nb1 <- train(sentiment~.,data=df_sentiment_train, method="nb",
                  trControl = trainControl(method="cv", number=3),
                  tuneGrid=expand.grid(fL=1:3,usekernel=c(TRUE, FALSE),adjust
=1:3))
msentiment_predict_nb1 <- predict(msentiment_nb1, newdata=df_sentiment_test,t
ype="raw")
```

```
#============================================================================
#==>  SVM: Preproces(NOrmalize)
#============================================================================
#normalize and get the lie target variable back in dataframes
normalize <- function(x) { (x - mean(x))/sd(x)  } #z-score transformation
ndf_lie_train <- as.data.frame(lapply(df_lie_train[,1:1013],normalize)) #remo
ve row 1
ndf_lie_train[is.na(ndf_lie_train)] <-0  #remove NaN resulting in rows w zero
x <- data.frame(df_lie_train[,1014])
colnames(x) <- c("lie")
ndf_lie_train <- data.frame(ndf_lie_train,x)#normalize lie test data frames
ndf_lie_test <- as.data.frame(lapply(df_lie_test[,1:1013],normalize))
ndf_lie_test[is.na(ndf_lie_test)] <-0  #remove NaN resulting in rows w zeros
x <- data.frame(df_lie_test[,1014])
colnames(x) <- c("lie")
ndf_lie_test <- data.frame(ndf_lie_test,x)
#write.csv(ndf_sentiment_test,"today.csv")
#==============>sentiment
ndf_sentiment_train <- as.data.frame(lapply(df_sentiment_train[,1:1013],norma
lize)) #remove row 1
ndf_sentiment_train[is.na(ndf_sentiment_train)] <-0  #remove NaN resulting in
rows
x <- data.frame(df_sentiment_train[,1014])
colnames(x) <- c("sentiment")
ndf_sentiment_train <- data.frame(ndf_sentiment_train,x)#normalize test df
ndf_sentiment_test <- as.data.frame(lapply(df_sentiment_test[,1:1013],normali
ze))
ndf_sentiment_test[is.na(ndf_sentiment_test)] <-0  #remove NaN in rows w zero
x <- data.frame(df_sentiment_test[,1014])
colnames(x) <- c("sentiment")
ndf_sentiment_test <- data.frame(ndf_sentiment_test,x)
remove(x)      #write.csv(ndf_sentiment_test,"today.csv")

#============================================================================
#==>  SVM: LIE MOdels
#============================================================================
=
library(gmodels,warn.conflicts = FALSE, quietly = TRUE) #for chi-square
set.seed(1984)
mlie_svm <-train(lie ~., data=ndf_lie_train, method= "svmLinear",
                         tuneGrid = expand.grid(C=seq(0,1,0.05)),
                         trControl = trainControl(method = "boot",
                                                  number=10))
#mlie_svm                          #==>  C (cost)
mlie_predict_svm_linear <- predict(mlie_svm, newdata=ndf_lie_test)
confusionMatrix(mlie_predict_svm_linear, ndf_lie_test$lie)
```

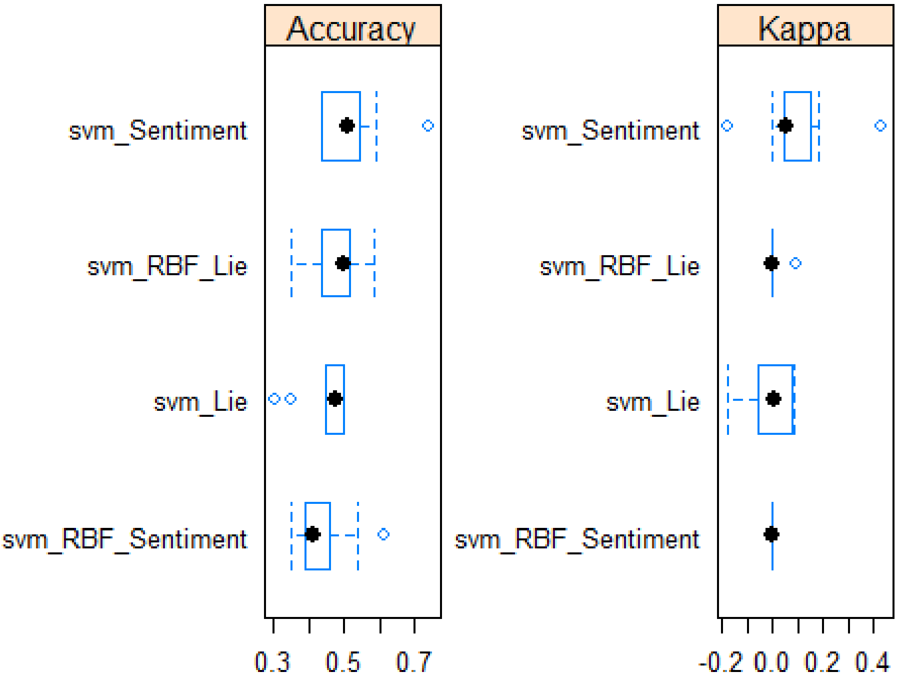
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction f t
##          f 5 7
##           t 8 6
## 
##                Accuracy : 0.4231
##                  95% CI : (0.2335, 0.6308)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.8365
## 
##                   Kappa : -0.1538
## 
##  Mcnemar's Test P-Value : 1.0000
## 
##             Sensitivity : 0.3846
##             Specificity : 0.4615
##          Pos Pred Value : 0.4167
##          Neg Pred Value : 0.4286
##              Prevalence : 0.5000
##          Detection Rate : 0.1923
##    Detection Prevalence : 0.4615
##       Balanced Accuracy : 0.4231
## 
##        'Positive' Class : f
## 

#chi-square
CrossTable(mlie_predict_svm_linear,df_lie_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, dnn=c('predicted','actual'))
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
## 
## Total Observations in Table:  26
## 
## 
##              | actual
##    predicted |         f |         t | Row Total |
## -------------|-----------|-----------|-----------|
##            f |         5 |         7 |        12 |
##              |     0.417 |     0.583 |     0.462 |
##              |     0.385 |     0.538 |           |
## -------------|-----------|-----------|-----------|
##            t |         8 |         6 |        14 |
##              |     0.571 |     0.429 |     0.538 |
##              |     0.615 |     0.462 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        13 |        13 |        26 |
##              |     0.500 |     0.500 |           |
## -------------|-----------|-----------|-----------|
#preceision & recall
# precision_mlie_svm <-posPredValue(mlie_predict_svm_linear,ndf_lie_test$lie,
positive="yes")
# recall_mlie_svm <-sensitivity(mlie_predict_svm_linear,ndf_lie_test$lie,posi
tive="yes")
# precision_mlie_svm;recall_mlie_svm

#==> SVM w non-linear Kernel RBF
mlie_svm_rbf <- train(lie ~., data=ndf_lie_train, method= "svmRadial",
                       tuneGrid = expand.grid(sigma=seq(0,1,0.1),
                                              C=seq(0,1,0.1)),
                       trControl = trainControl(method = "boot",
                                                number=10))
#mlie_svm_rbf
lie_predict_svm_rbf <- predict(mlie_svm_rbf, newdata=ndf_lie_test)
#chi-square
CrossTable(lie_predict_svm_rbf,df_lie_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, dnn=c('predicted','actual'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |-------------------------|
## 
## 
## Total Observations in Table:  26 
## 
## 
##                     | df_lie_test_labels 
## lie_predict_svm_rbf |         f |         t | Row Total | 
## --------------------|-----------|-----------|-----------|
##                   f |        13 |        13 |        26 | 
## --------------------|-----------|-----------|-----------|
##        Column Total |        13 |        13 |        26 | 
## --------------------|-----------|-----------|-----------|
## 
```

```
#===========================================================================
#==>  SVM: SENTIMENT MODELS
#==========================================================================
set.seed(1984)
m_sentiment_svm <-train(sentiment ~., data=ndf_sentiment_train, method= "svmL
inear",
                 tuneGrid = expand.grid(C=seq(0,1,0.05)),
                 trControl = trainControl(method = "boot",
                                          number=10))
#m_sentiment_svm                        #==>  C (cost)
m_sentiment_predict_svm_linear <- predict(m_sentiment_svm, newdata=ndf_sentim
ent_test)
#Chi-square
CrossTable(m_sentiment_predict_svm_linear,df_sentiment_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, dnn=c('predicted','actual'))

#============================================================================
#==>  MODEL COMPARISON GRAPHs
#============================================================================
m_nb_compare <- resamples(list(nb_Lie=mlie_nb1, nb_Sentiment=msentiment_nb1))
summary(m_nb_compare)

## 
## Call:
## summary.resamples(object = m_nb_compare)
## 
## Models: nb_Lie, nb_Sentiment 
## Number of resamples: 3 
## 
## Accuracy 
##                   Min.   1st Qu.    Median     Mean   3rd Qu.      Max.
## nb_Lie       0.4285714 0.4523810 0.4761905 0.468254 0.4880952 0.5000000
## nb_Sentiment 0.5000000 0.5119048 0.5238095 0.515873 0.5238095 0.5238095
##              NA's
## nb_Lie          0
## nb_Sentiment    0
## 
## Kappa 
##                     Min.     1st Qu. Median        Mean 3rd Qu. Max. NA's
## nb_Lie       -0.09565217 -0.04782609      0 -0.03188406       0    0    0
## nb_Sentiment  0.00000000  0.00000000      0  0.00000000       0    0    0

scales <- list(x=list(relation = "free"),
               y=list(relation = "free"))
bwplot(m_nb_compare, scales = scales)
```



```
m_SVM_compare <- resamples(list(svm_Lie=mlie_svm, svm_RBF_Lie=mlie_svm_rbf,
     svm_Sentiment=m_sentiment_svm, svm_RBF_Sentiment=m_sentiment_svm_rb
f ))
summary(m_SVM_compare)

##
## Call:
## summary.resamples(object = m_SVM_compare)
##
## Models: svm_Lie, svm_RBF_Lie, svm_Sentiment, svm_RBF_Sentiment
## Number of resamples: 10
##
## Accuracy
##                        Min.   1st Qu.    Median      Mean   3rd Qu.
## svm_Lie           0.3043478 0.4498433 0.4782609 0.4491517 0.4950000
## svm_RBF_Lie       0.3478261 0.4360870 0.5000000 0.4824461 0.5138889
## svm_Sentiment     0.4333333 0.4460870 0.5086207 0.5197373 0.5395257
## svm_RBF_Sentiment 0.3478261 0.3934783 0.4120370 0.4401471 0.4524457
##                        Max. NA's
## svm_Lie           0.5000000    0
## svm_RBF_Lie       0.5833333    0
## svm_Sentiment     0.7391304    0
## svm_RBF_Sentiment 0.6153846    0
##
## Kappa
##                         Min.     1st Qu.      Median         Mean
## svm_Lie           -0.1774744 -0.05376623 0.003759398 -0.007211927
## svm_RBF_Lie        0.0000000  0.00000000 0.000000000  0.009448819
## svm_Sentiment     -0.1767442  0.04543807 0.056782334  0.087421632
## svm_RBF_Sentiment  0.0000000  0.00000000 0.000000000  0.000000000
##                      3rd Qu.       Max. NA's
## svm_Lie            0.06515499 0.08906883    0
## svm_RBF_Lie        0.00000000 0.09448819    0
## svm_Sentiment      0.13380404 0.42975207    0
## svm_RBF_Sentiment 0.00000000 0.00000000    0

scales <- list(x=list(relation = "free"),
               y=list(relation = "free"))
bwplot(m_SVM_compare, scales = scales)
```


Accuracy
Kappa
svm_Sentiment
svm_RBF_Lie
svm_Lie
svm_RBF_Sentiment
svm_Sentiment
svm_RBF_Lie
svm_Lie
svm_RBF_Sentiment