# milestone03_yujiaochen_brianho_jonjay

*Jonathan Jay*

*4/19/2017*

In this analysis we build upon our analysis using word count features, by attempting to add the features of movie titles. In Part I, we find that it's possible to train a model to predict between two distant classes (romance and horror) using only simple features within the title, such as character/word length, presence of grammatical symbols, and sentiment. In Parts II and III, we attempt to predict among three classes–first using title alone, then adding our established model for predicting on description word counts. We find that title feature analysis can't do much in this problem, particularly using a relatively small sample (n = 370).

# Part I: can titles differentiate between two genres?

This analysis explores the possibility of predicting genre based on features of its title. The features here are identified based on a priori consideration of the elements of a title that might provide genre clues. Here we attempted to start simple, predicting only on romance vs. horror movies, because these genres are so distant from each other in our prior mapping.

```
library(stringr)
library(e1071)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(RSentiment)

#load our dataset of the top 10000 movies across many genres
movies <- read.csv("~/Documents/DrPH_open/Data Science 2/Final project/Milestone 3/top10
00.csv")
movies$genres <- gsub("\\[|\\]", "", movies$genre_ids)
```

```
#filter to movies that are horror or romance, but not both
romance <- movies[grepl("10749", movies$genres), ]
romance <- romance[!grepl("27", romance$genres), ]
romance$genre <- 1 #needs numeric for svm later

horror <- movies[grepl("27", movies$genres), ]
horror <- horror[!grepl("10749", horror$genres), ]
horror$genre <- 0

df <- rbind(romance, horror)
df <- df[, c("genre", "title")]

#yields 1329 horror and 1337 romance

df$title <- as.character(df$title)
```

```
#### create title-based features
# basics
df$nwords <- sapply(gregexpr("\\S+", df$title), length) #number of words
df$num <- ifelse(grepl("\\d", df$title), 1, 0) #does it include digits
df$numloc <- regexpr("\\d", df$title) + 1 #where are those digits located in string
df$char <- nchar(df$title) # number of characters

#grammatical features
df$apost <- ifelse(grepl("'", df$title), 1, 0)
df$quest <- ifelse(grepl("\\?", df$title), 1, 0)
df$colon <- ifelse(grepl(":", df$title), 1, 0)
df$hyph <- ifelse(grepl("-", df$title), 1, 0)
df$excl <- ifelse(grepl("!", df$title), 1, 0)
df$amper <- ifelse(grepl("&", df$title), 1, 0)

#other grammatical structure features
df$the_st <- ifelse(substr(df$title, 1, 3)=="The", 1, 0) #does it start with "The"?

#introduce sentiment score on title (from RSentiment package)
# i.e., does the title sound "positive" or "negative"?
df$sentscore <- calculate_score(df$title)
df[df$sentscore==99, "sentscore"] <- 0 # drop "sarcasm" score

#save dataset
write.csv(df, file = "Romances-horrors w title features.csv")
```

This preliminary exercise yielded 12 custom features. Next is exploratory data analysis using these features:

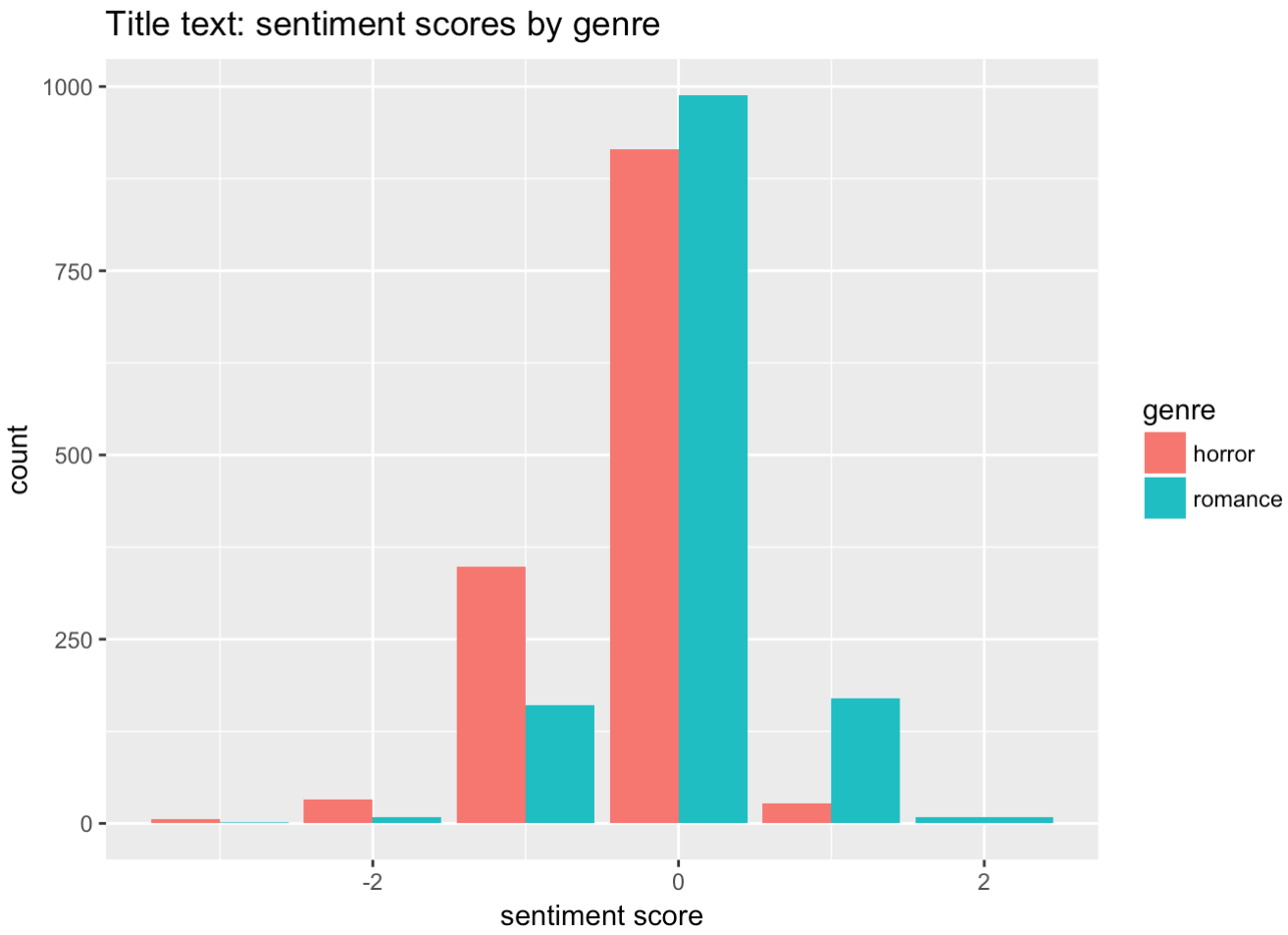
```
df$genre <- factor(df$genre, levels = c(0, 1), labels = c("horror", "romance"))

## how does title sentiment vary by genre?
summary(df[df$genre==1, "sentscore"])
```

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

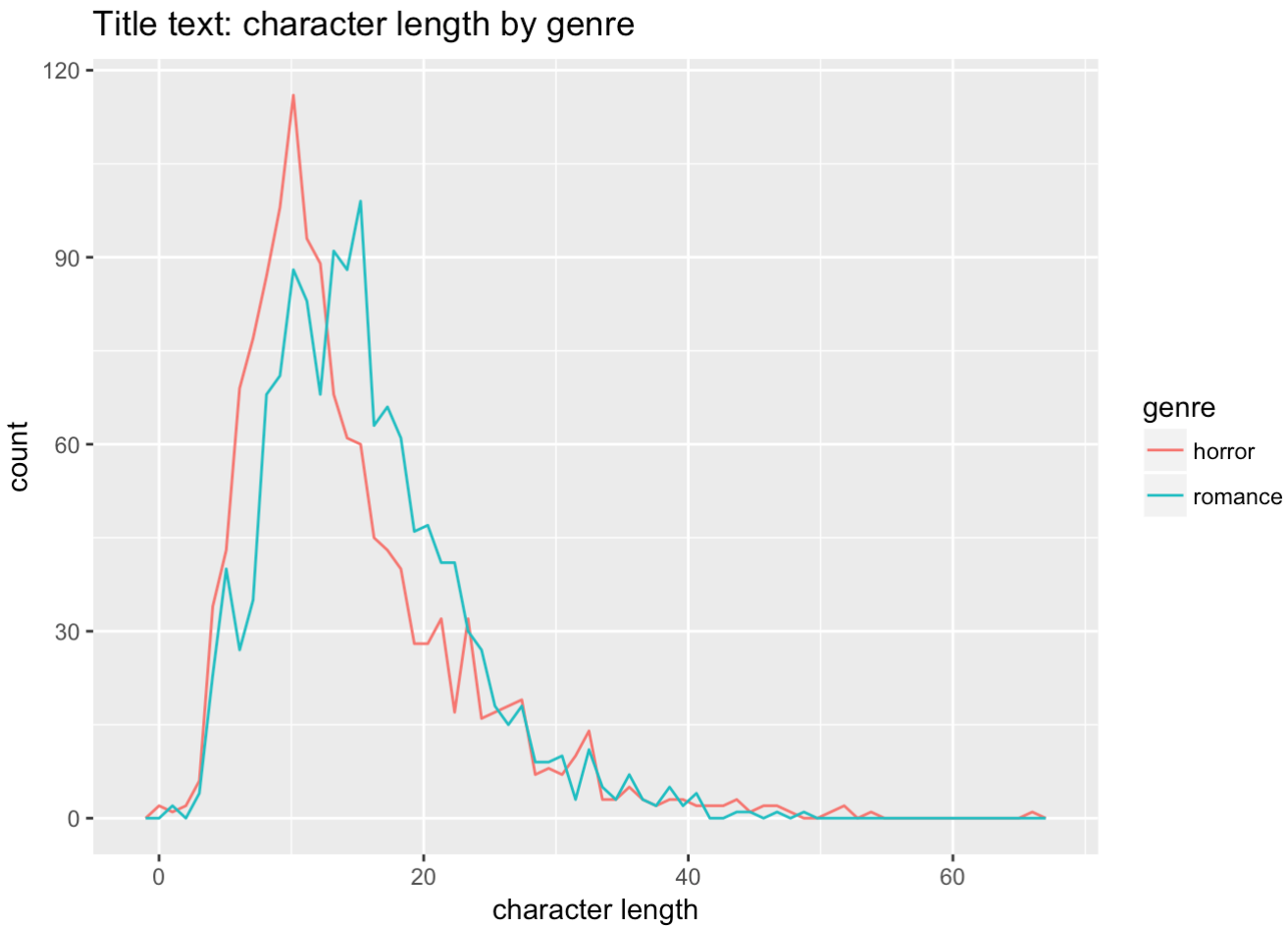
```
summary(df[df$genre==0, "sentscore"])
```

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

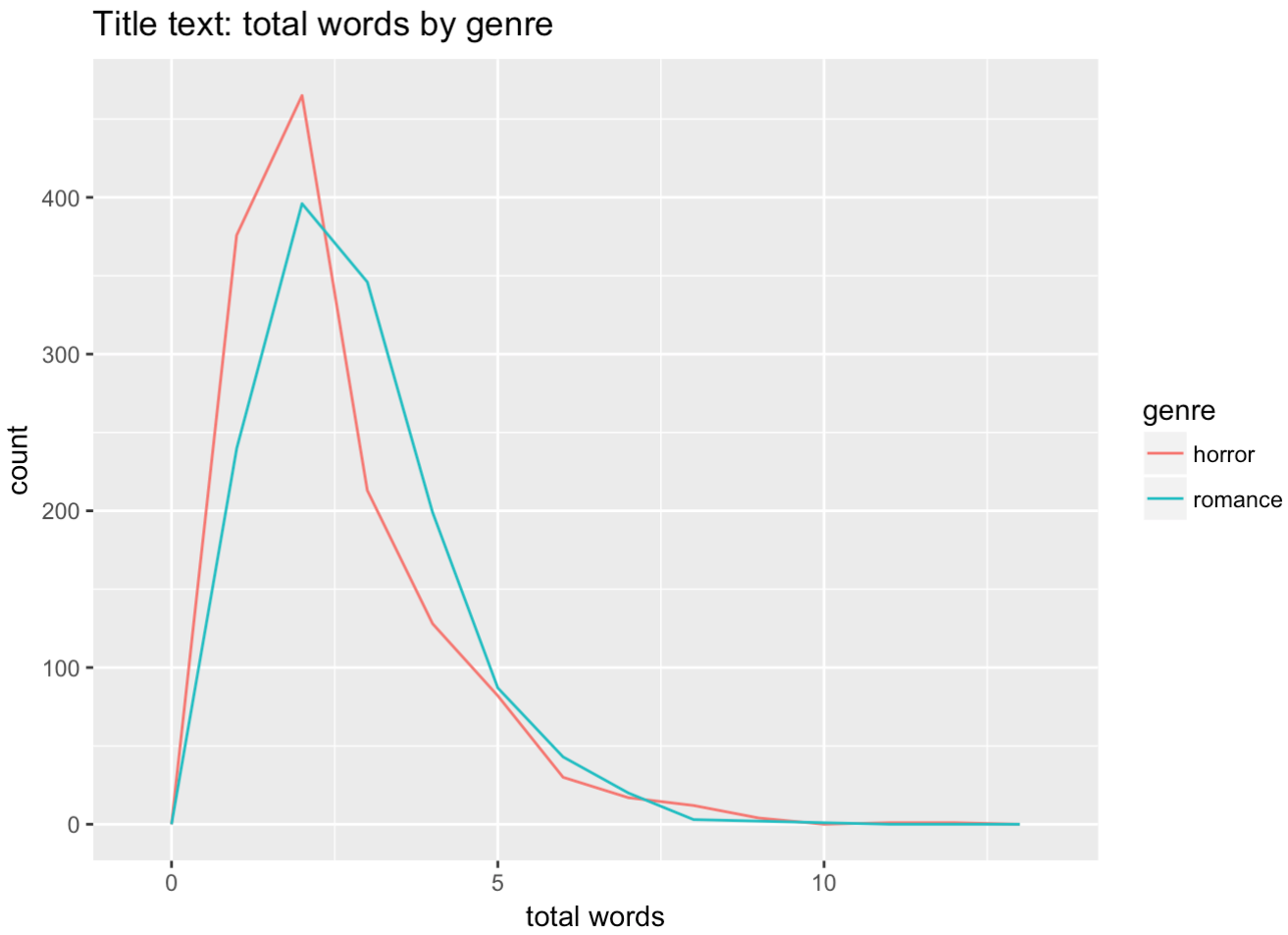
```
##compare:
# sentiment scores
ggplot(df, aes(x=sentscore, fill=genre)) + geom_bar(stat= "count", position = "dodge") +
  labs(x="sentiment score", title="Title text: sentiment scores by genre")
```



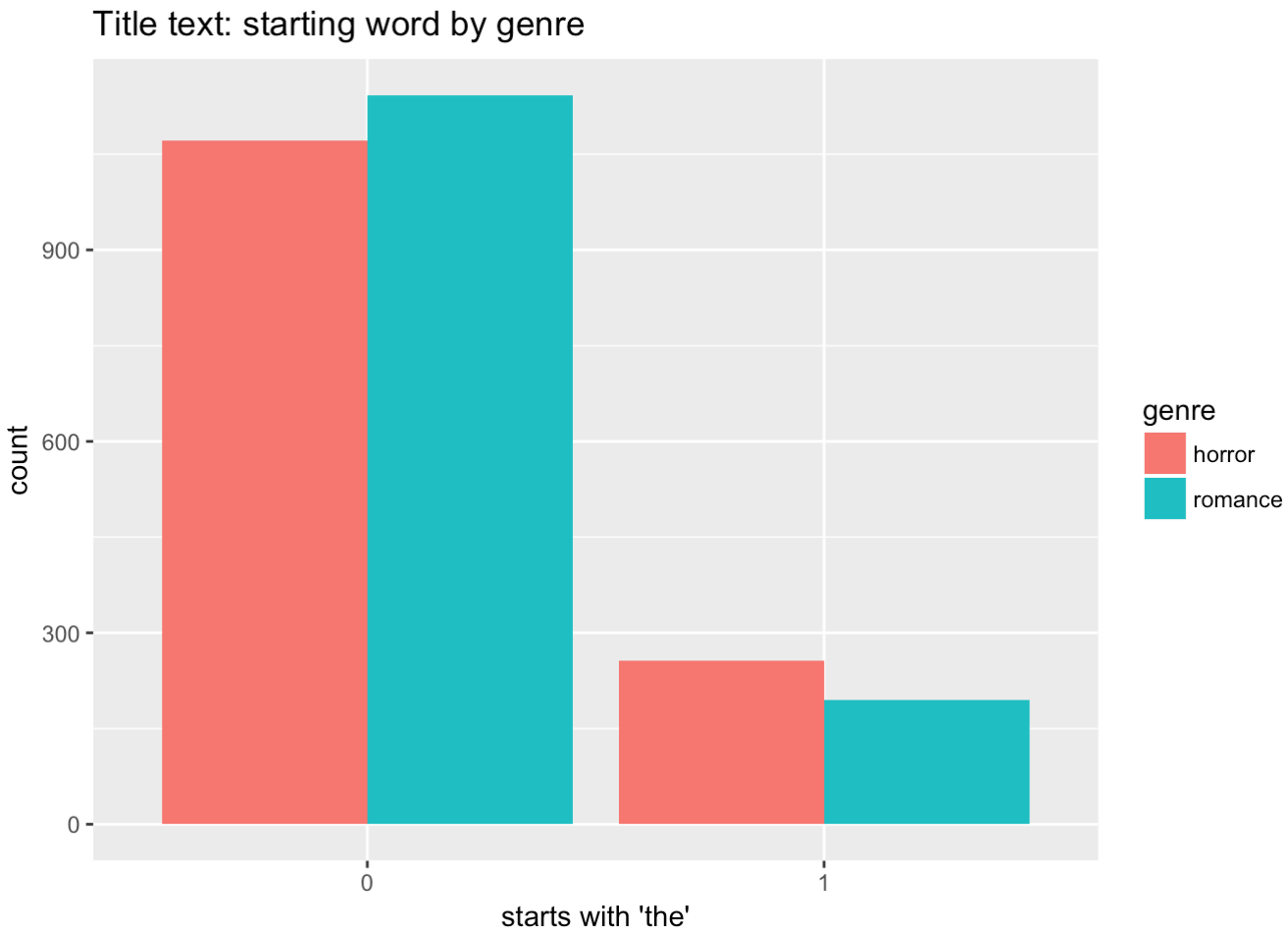

```
# character length
ggplot(df, aes(x=char, color=genre)) + geom_freqpoly(bins=max(df$char)) +
  labs(x="character length", title="Title text: character length by genre")
```



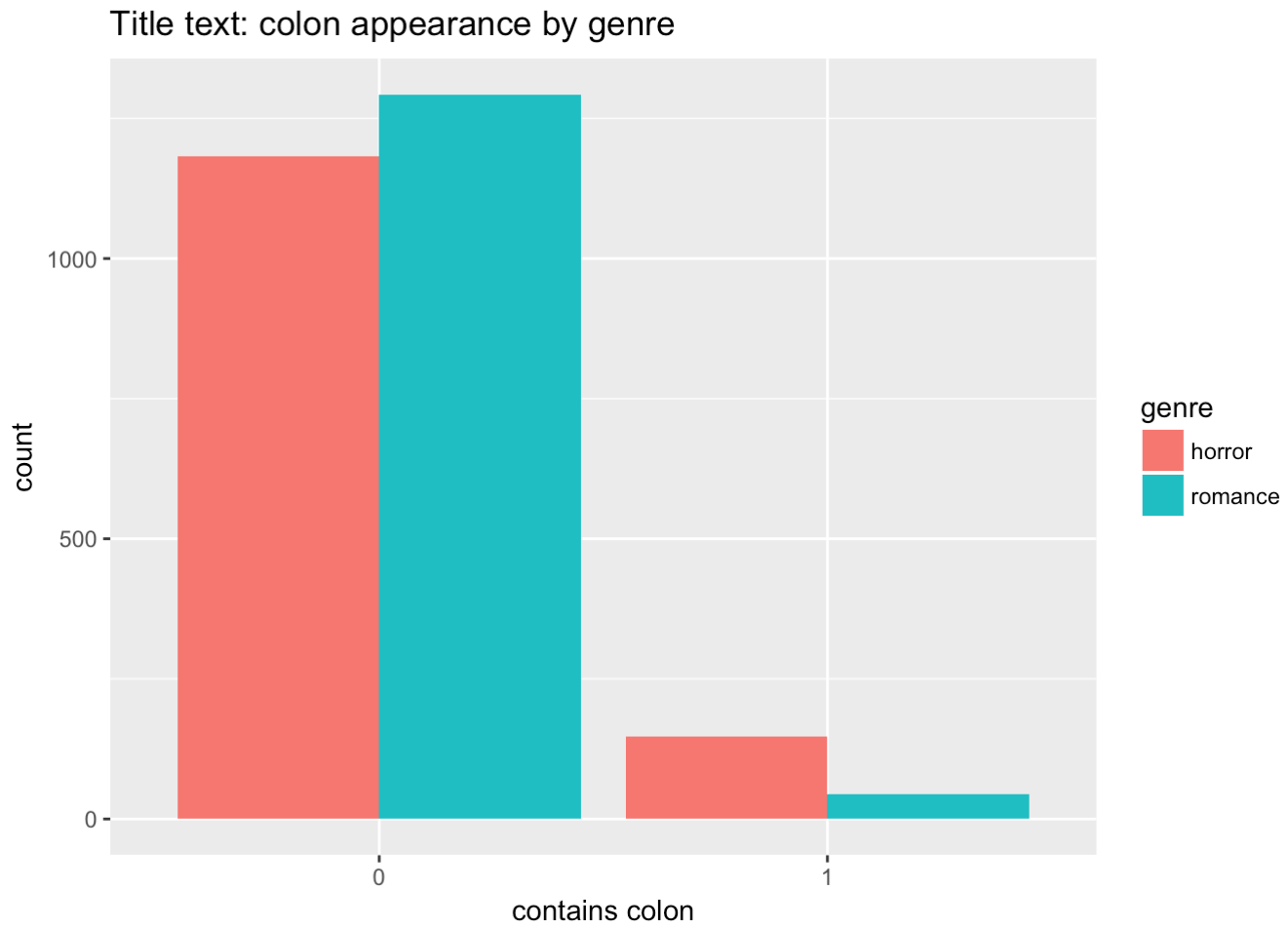

```
# word count
ggplot(df, aes(x=nwords, color=genre)) + geom_freqpoly(bins=max(df$nwords)) +
  labs(x="total words", title="Title text: total words by genre")
```




```
# begins with "The" [i.e., "The Something"]
ggplot(df, aes(x=factor(the_st), fill=genre)) + geom_bar(stat= "count", position = "dodg
e") +
  labs(x="starts with 'the'", title="Title text: starting word by genre")
```




```
# includes a (grammatical) colon
ggplot(df, aes(x=factor(colon), fill=genre)) + geom_bar(stat= "count", position = "dodg
e") +
  labs(x="contains colon", title="Title text: colon appearance by genre")
```




These exploratory analyses show some trends that could help predict genre based on title structure. As expected, it appears that horror movies are more likely to have negative-sentiment titles (e.g. “Wrong Turn 2: Dead End”), although many romance movies have these titles as well (e.g. “Lazy Hazy Crazy”). Romance movies are far more likely to have positive-sentiment names (e.g. “The Most Wonderful Time of the Year”) and very few horror movies do (just a handful, incl. ironic names like “The Perfect Husband”). For this analysis we used the out-of-the-box RSentiment package, which identifies positive and negative words and looks for negation and considers parts of speech.

For word count and character length, the most notable difference appears to be at the low end of the distribution, where horror movies are more likely to have very short titles. They also populate more of the very high end of the distribution.

As hypothesized, horror movies titles were somewhat more likely to have structures such as “The Something” and “Main Title 3: the Subtitle”, as the counts for the-start and colon features demonstrate.

Next, we experiment with SVM models (linear and RBF kernel) to see whether these associations are strong enough to predict genre.

```
###prepare for modeling
df$genre <- as.numeric(df$genre) - 1 # convert back to numeric

#drop title
df <- df[, -2]

# divide test and train
index <- sample(1:nrow(df), 500)
train <- df[-index, ]
test <- df[index, ]

scalevals <- preProcess(train[, 2:ncol(train)], method = c("center", "scale"))

train[, 2:ncol(train)] <- predict(scalevals, train[, 2:ncol(train)])
test[, 2:ncol(test)] <- predict(scalevals, test[, 2:ncol(test)])

################### model 1: SVM
tuneResult.linear <- tune(svm, genre ~ ., data = train, kernel = "linear",
                          ranges = list(cost = exp(-7:0)),
                          tunecontrol = tune.control(cross=5))
tuneResult.RBF <- tune(svm, genre ~ ., data = train, kernel = "radial",
                       ranges = list(gamma = exp(-7:0), cost = exp(-7:2)),
                       tunecontrol = tune.control(cross=5))

print(tuneResult.linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##         cost
##  0.000911882
##
## - best performance: 0.225815
```

```
print(tuneResult.RBF)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##       gamma       cost
##  0.01831564 0.04978707
##
## - best performance: 0.2227781
```

```
train$svm.lk <- ifelse(tuneResult.linear$best.model$fitted >= 0.5, 1, 0)
test$svm.lk <- ifelse(predict(tuneResult.linear$best.model, test) >= 0.5, 1, 0)

### Assess predictive accuracy
## for linear kernel svm model
# training set accuracy
confusionMatrix(train$genre, train$svm.lk)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 843 240
##          1 517 566
##
##                Accuracy : 0.6505
##                  95% CI : (0.63, 0.6706)
##     No Information Rate : 0.6279
##     P-Value [Acc > NIR] : 0.01526
##
##                   Kappa : 0.301
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.6199
##             Specificity : 0.7022
##          Pos Pred Value : 0.7784
##          Neg Pred Value : 0.5226
##              Prevalence : 0.6279
##          Detection Rate : 0.3892
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.6610
##
##        'Positive' Class : 0
##
```

```
# testing set accuracy
confusionMatrix(test$genre, test$svm.lk) #66% accurate in my test without sentiment; no
 improvement with it
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 180  66
##          1 105 149
##
##                Accuracy : 0.658
##                  95% CI : (0.6146, 0.6995)
##     No Information Rate : 0.57
##     P-Value [Acc > NIR] : 3.584e-05
##
##                   Kappa : 0.3175
##  Mcnemar's Test P-Value : 0.003662
##
##             Sensitivity : 0.6316
##             Specificity : 0.6930
##          Pos Pred Value : 0.7317
##          Neg Pred Value : 0.5866
##              Prevalence : 0.5700
##          Detection Rate : 0.3600
##    Detection Prevalence : 0.4920
##       Balanced Accuracy : 0.6623
##
##        'Positive' Class : 0
##
```

```
## for rbf kernel svm model
train$svm.rbf <- ifelse(tuneResult.RBF$best.model$fitted >= 0.5, 1, 0)
test$svm.rbf <- ifelse(predict(tuneResult.RBF$best.model, test) >= 0.5, 1, 0)

### Assess predictive accuracy
## for linear kernel svm model
# training set accuracy
confusionMatrix(train$genre, train$svm.rbf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 817 266
##          1 462 621
##
##                Accuracy : 0.6639
##                  95% CI : (0.6436, 0.6838)
##     No Information Rate : 0.5905
##     P-Value [Acc > NIR] : 1.258e-12
##
##                   Kappa : 0.3278
##  Mcnemar's Test P-Value : 4.931e-13
##
##             Sensitivity : 0.6388
##             Specificity : 0.7001
##          Pos Pred Value : 0.7544
##          Neg Pred Value : 0.5734
##              Prevalence : 0.5905
##          Detection Rate : 0.3772
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.6694
##
##        'Positive' Class : 0
##
```

```
# testing set accuracy
confusionMatrix(test$genre, test$svm.rbf) #63% accurate in my test without sentiment; no
 improvement with it
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 176  70
##          1  91 163
##
##                Accuracy : 0.678
##                  95% CI : (0.6351, 0.7188)
##     No Information Rate : 0.534
##     P-Value [Acc > NIR] : 4.084e-11
##
##                   Kappa : 0.3567
##  Mcnemar's Test P-Value : 0.115
##
##             Sensitivity : 0.6592
##             Specificity : 0.6996
##          Pos Pred Value : 0.7154
##          Neg Pred Value : 0.6417
##              Prevalence : 0.5340
##          Detection Rate : 0.3520
##    Detection Prevalence : 0.4920
##       Balanced Accuracy : 0.6794
##
##        'Positive' Class : 0
##
```

On various runs through this analysis, the classification accuracy was typically between 60-65%. Interestingly, the model performed equally well with the sentiment analysis omitted, perhaps because only a subset of observations receive non-neutral scores, and only positive scores seem to differentiate clearly between the classes.

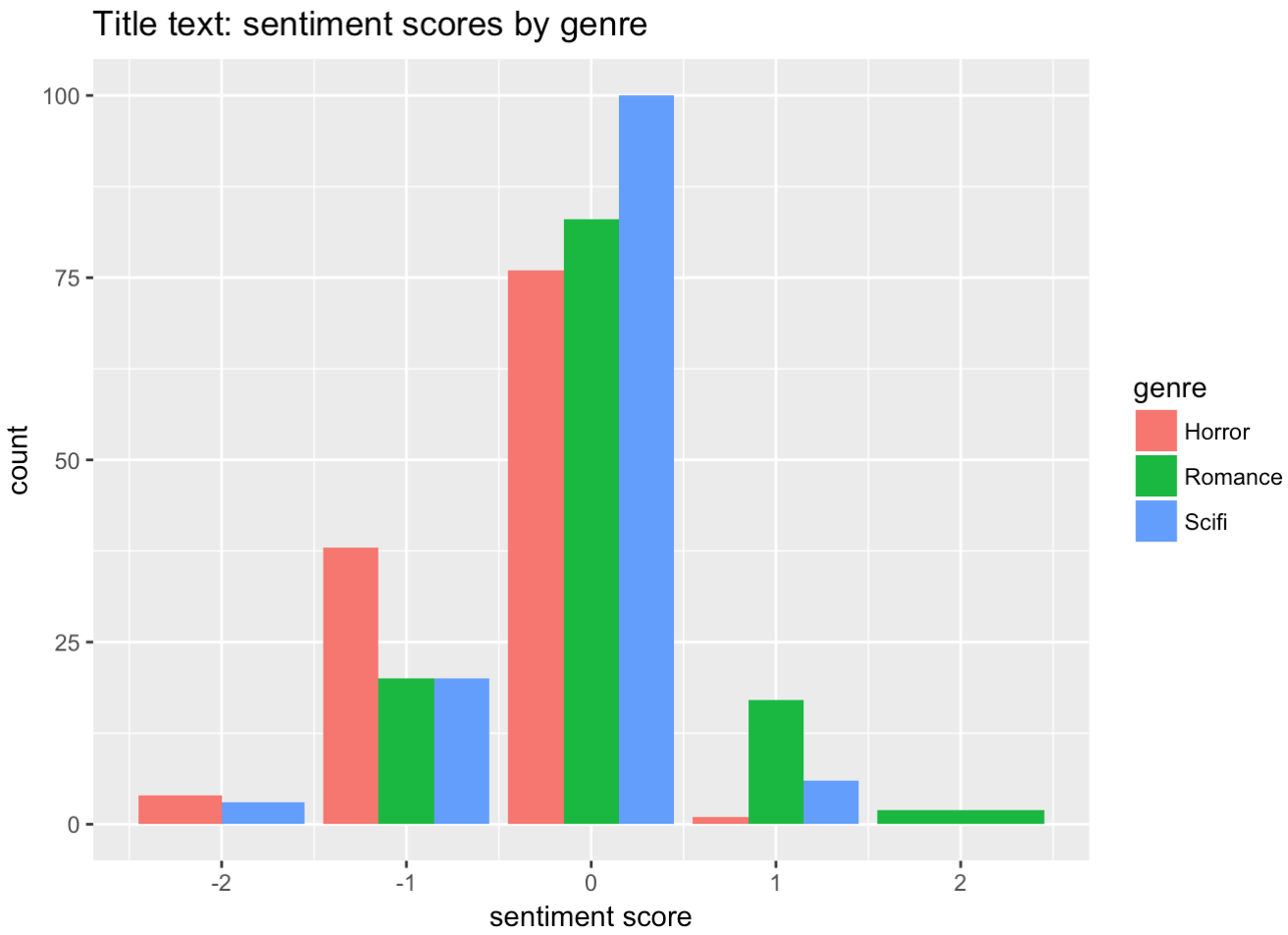
# Part II: can titles differentiate among three genres?

In this part, we adapt the model above to try to predict across three classes, in preparation for combining this approach with our PCA-SVM approach.

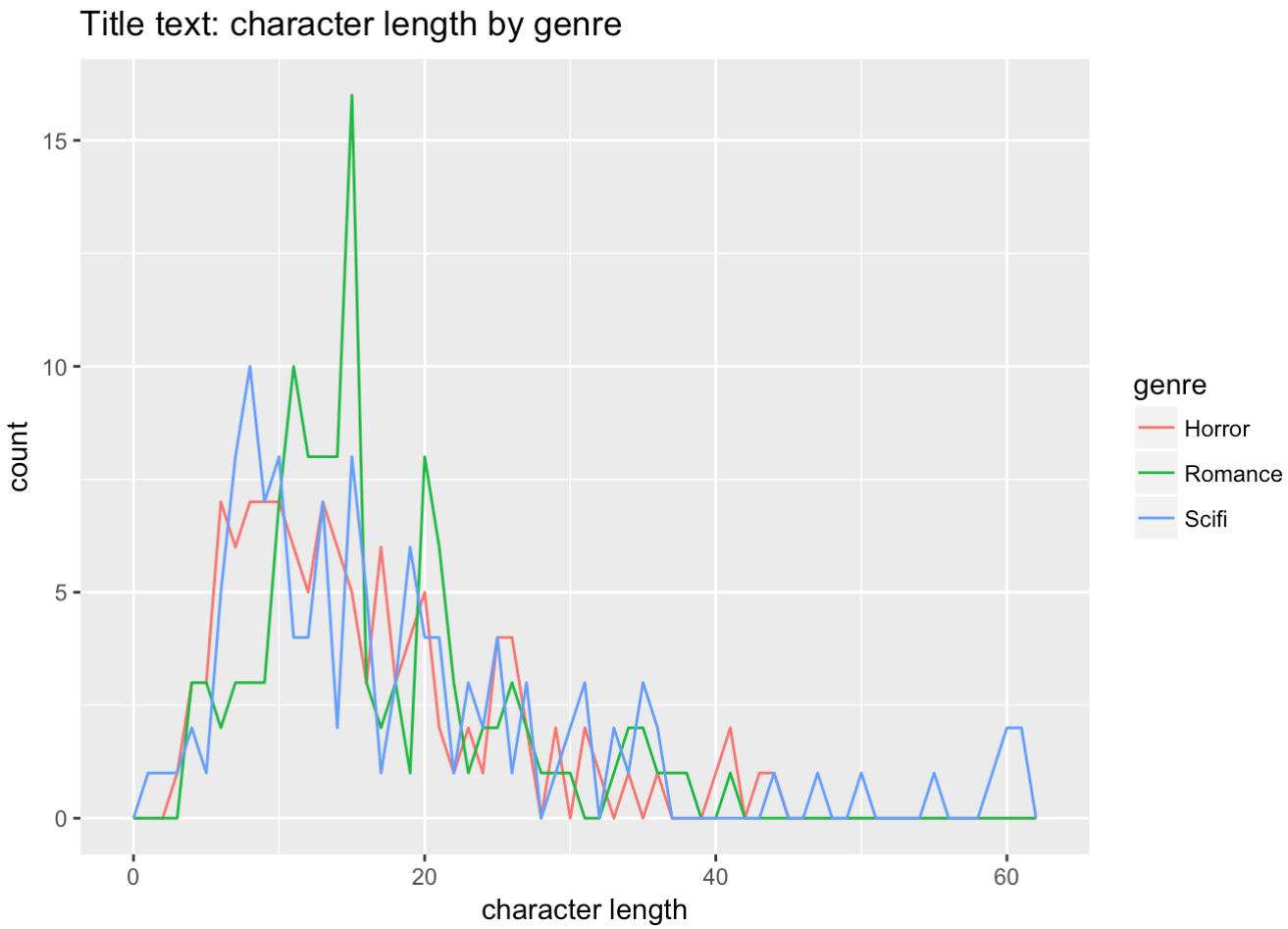
```
genre <- read.csv("~/Documents/DrPH_open/Data Science 2/Final project/Milestone 3/genre.
csv", header=F)
title <- read.csv("~/Documents/DrPH_open/Data Science 2/Final project/Milestone 3/title.
csv", header=F)

df <- data.frame(genre = genre$V2, title = title$V2)
df$title <- as.character(df$title)

## fix weird title strings
df$title <- gsub("\\<[^\\]]*\\>", "", df$title, perl=TRUE)

# apply features from above
df$nwords <- sapply(gregexpr("\\S+", df$title), length) #number of words
df$num <- ifelse(grepl("\\d", df$title), 1, 0) #does it include digits
df$numloc <- regexpr("\\d", df$title) + 1 #where are those digits located in string
df$char <- nchar(df$title) # number of characters

#grammatical features
df$apost <- ifelse(grepl("'", df$title), 1, 0)
df$quest <- ifelse(grepl("\\?", df$title), 1, 0)
df$colon <- ifelse(grepl(":", df$title), 1, 0)
df$hyph <- ifelse(grepl("-", df$title), 1, 0)
df$excl <- ifelse(grepl("!", df$title), 1, 0)
df$amper <- ifelse(grepl("&", df$title), 1, 0)

#other grammatical structure features
df$the_st <- ifelse(substr(df$title, 1, 3)=="The", 1, 0) #does it start with "The"?

#introduce sentiment score on title (from RSentiment package)
# i.e., does the title sound "positive" or "negative"?
df$sentscore <- calculate_score(df$title)
df[df$sentscore==99, "sentscore"] <- 0 # drop "sarcasm" score
```

Check EDA, same as above:

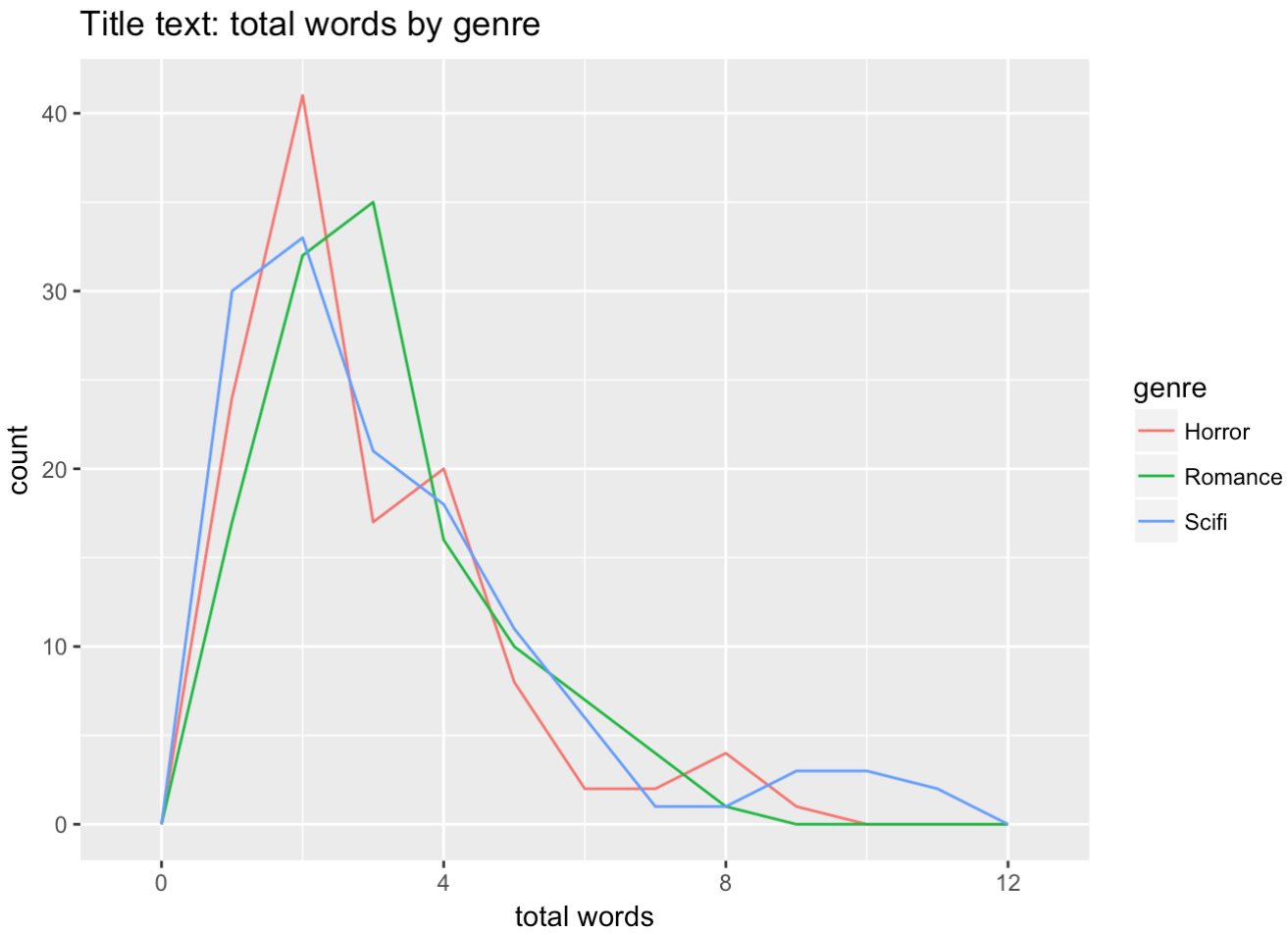
```
##compare:
# sentiment scores
ggplot(df, aes(x=sentscore, fill=genre)) + geom_bar(stat= "count", position = "dodge") +
  labs(x="sentiment score", title="Title text: sentiment scores by genre")
```



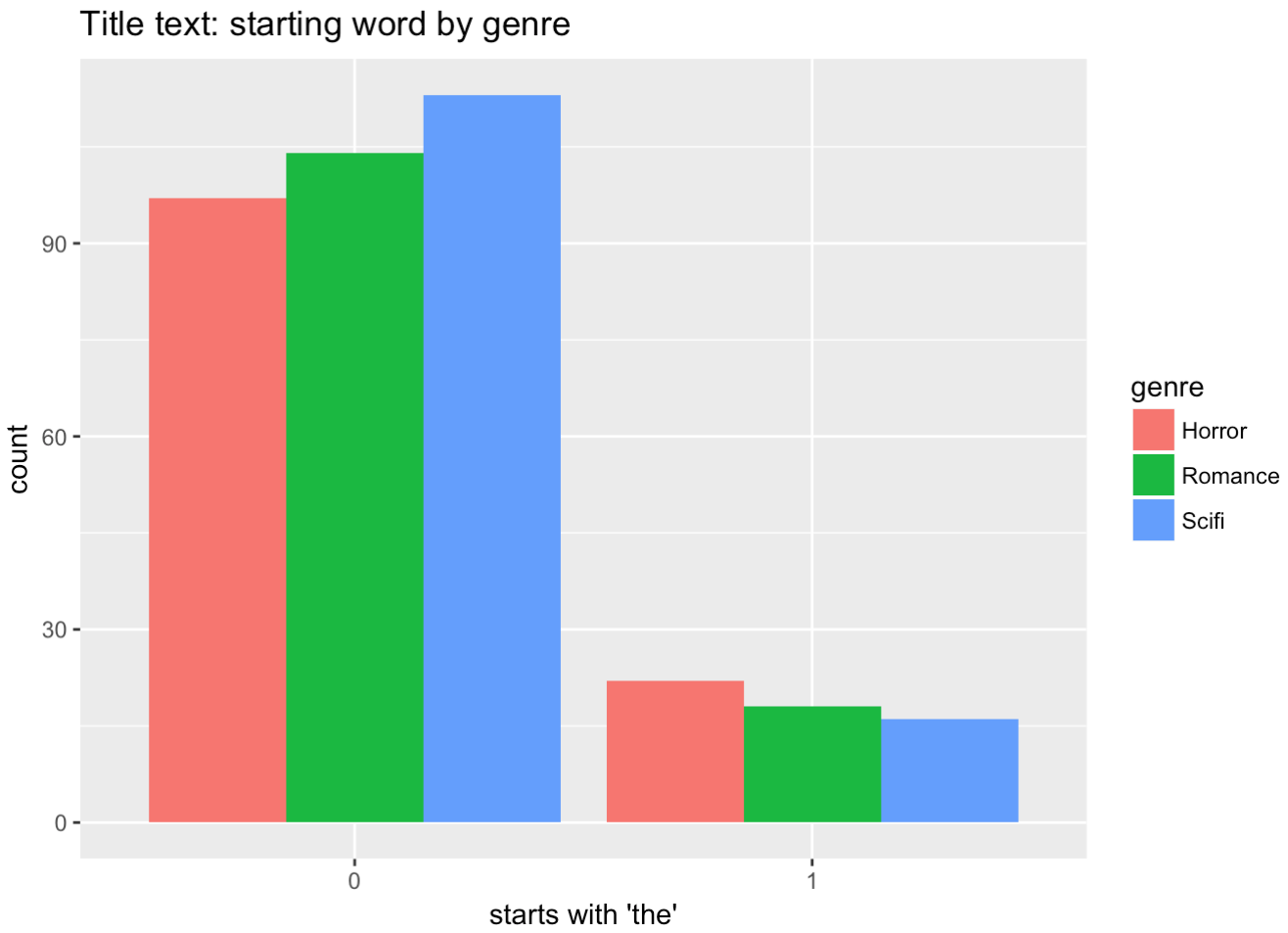

```
# character length
ggplot(df, aes(x=char, color=genre)) + geom_freqpoly(bins=max(df$char)) +
  labs(x="character length", title="Title text: character length by genre")
```



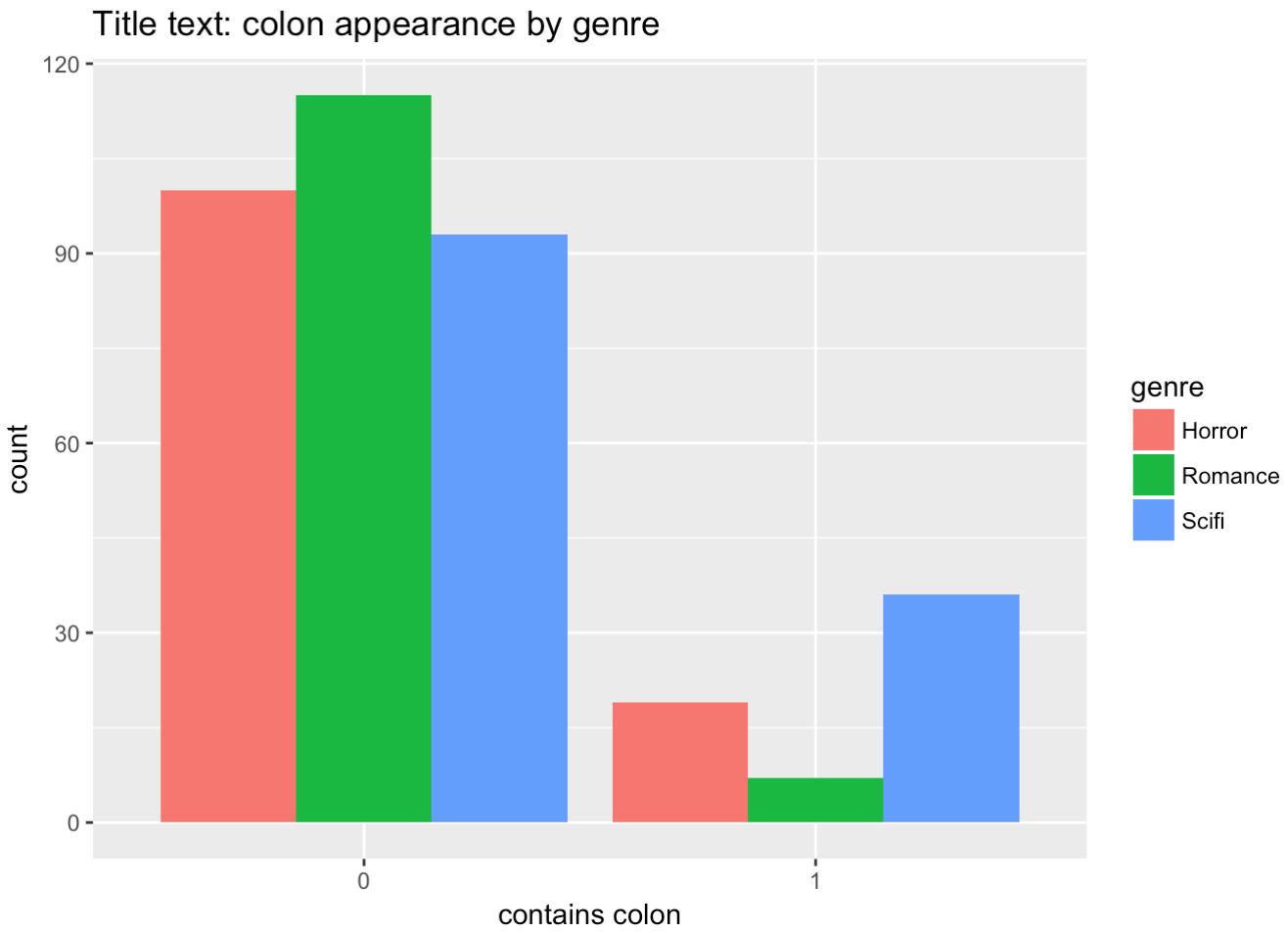

```
# word count
ggplot(df, aes(x=nwords, color=genre)) + geom_freqpoly(bins=max(df$nwords)) +
  labs(x="total words", title="Title text: total words by genre")
```




```
# begins with "The" [i.e., "The Something"]
ggplot(df, aes(x=factor(the_st), fill=genre)) + geom_bar(stat= "count", position = "dodg
e") +
  labs(x="starts with 'the'", title="Title text: starting word by genre")
```




```
# includes a (grammatical) colon
ggplot(df, aes(x=factor(colon), fill=genre)) + geom_bar(stat= "count", position = "dodg
e") +
  labs(x="contains colon", title="Title text: colon appearance by genre")
```




As expected, the inclusion of the sci-fi class makes the classification task much harder: across all of these features, scifi "looks" a lot like horror. We wouldn't expect our classifier to perform well here, but let's try with linear kernel SVM on an exploratory basis:

```
#drop title
df <- df[, -2]

# divide test and train
index <- sample(1:nrow(df), 50)
train <- df[-index, c(1:6, 8, 12:13)] #dropping the grammatical features yielding very l
ittle variation here
test <- df[index,  c(1:6, 8, 12:13)]

scalevals <- preProcess(train[, 2:ncol(train)], method = c("center", "scale"))

train[, 2:ncol(train)] <- predict(scalevals, train[, 2:ncol(train)])
test[, 2:ncol(test)] <- predict(scalevals, test[, 2:ncol(test)])

################### model 1: SVM linear kernel
tuneResult.linear <- tune(svm, genre ~ ., data = train, kernel = "linear",
                          ranges = list(cost = exp(-7:0)),
                          tunecontrol = tune.control(cross=5))
print(tuneResult.linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.578125
```

```
tuneResult.RBF <- tune(svm, genre ~ ., data = train, kernel = "radial",
                       ranges = list(gamma = exp(-7:0), cost = exp(-7:2)),
                       tunecontrol = tune.control(cross=5))
print(tuneResult.RBF)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##      gamma     cost
##  0.3678794 7.389056
##
## - best performance: 0.534375
```

```
train$svm.lk <- tuneResult.linear$best.model$fitted
test$svm.lk <- predict(tuneResult.linear$best.model, newdata = test)

train$svm.rbf <- tuneResult.RBF$best.model$fitted
test$svm.rbf <- predict(tuneResult.RBF$best.model, newdata = test)

### Assess predictive accuracy
## for linear kernel svm model
# training set accuracy
confusionMatrix(train$genre, train$svm.lk)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Horror Romance Scifi
##    Horror      45      52     7
##    Romance     20      82     6
##    Scifi       20      62    26
##
## Overall Statistics
##
##                Accuracy : 0.4781
##                  95% CI : (0.4223, 0.5344)
##     No Information Rate : 0.6125
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2162
##  Mcnemar's Test P-Value : 2.282e-14
##
## Statistics by Class:
##
##                      Class: Horror Class: Romance Class: Scifi
## Sensitivity                 0.5294         0.4184      0.66667
## Specificity                 0.7489         0.7903      0.70819
## Pos Pred Value              0.4327         0.7593      0.24074
## Neg Pred Value              0.8148         0.4623      0.93868
## Prevalence                  0.2656         0.6125      0.12187
## Detection Rate              0.1406         0.2562      0.08125
## Detection Prevalence        0.3250         0.3375      0.33750
## Balanced Accuracy           0.6392         0.6043      0.68743
```

```
# testing set accuracy
confusionMatrix(test$genre, test$svm.lk)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Horror Romance Scifi
##    Horror       7       8     0
##    Romance      2      11     1
##    Scifi        6      12     3
##
## Overall Statistics
##
##                Accuracy : 0.42
##                  95% CI : (0.2819, 0.5679)
##     No Information Rate : 0.62
##     P-Value [Acc > NIR] : 0.9986792
##
##                   Kappa : 0.1747
##  Mcnemar's Test P-Value : 0.0002857
##
## Statistics by Class:
##
##                      Class: Horror Class: Romance Class: Scifi
## Sensitivity                 0.4667         0.3548       0.7500
## Specificity                 0.7714         0.8421       0.6087
## Pos Pred Value              0.4667         0.7857       0.1429
## Neg Pred Value              0.7714         0.4444       0.9655
## Prevalence                  0.3000         0.6200       0.0800
## Detection Rate              0.1400         0.2200       0.0600
## Detection Prevalence        0.3000         0.2800       0.4200
## Balanced Accuracy           0.6190         0.5985       0.6793
```

```
# training set accuracy
confusionMatrix(train$genre, train$svm.rbf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Horror Romance Scifi
##    Horror      47      29    28
##    Romance      8      86    14
##    Scifi       11      29    68
##
## Overall Statistics
##
##                Accuracy : 0.6281
##                  95% CI : (0.5726, 0.6812)
##     No Information Rate : 0.45
##     P-Value [Acc > NIR] : 1.146e-10
##
##                   Kappa : 0.4409
##  Mcnemar's Test P-Value : 1.907e-05
##
## Statistics by Class:
##
##                      Class: Horror Class: Romance Class: Scifi
## Sensitivity                 0.7121         0.5972       0.6182
## Specificity                 0.7756         0.8750       0.8095
## Pos Pred Value              0.4519         0.7963       0.6296
## Neg Pred Value              0.9120         0.7264       0.8019
## Prevalence                  0.2062         0.4500       0.3438
## Detection Rate              0.1469         0.2687       0.2125
## Detection Prevalence        0.3250         0.3375       0.3375
## Balanced Accuracy           0.7439         0.7361       0.7139
```

```
# testing set accuracy
confusionMatrix(test$genre, test$svm.rbf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Horror Romance Scifi
##    Horror       7       6     2
##    Romance      0       7     7
##    Scifi        7       7     7
##
## Overall Statistics
##
##                Accuracy : 0.42
##                  95% CI : (0.2819, 0.5679)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 0.4390
##
##                   Kappa : 0.1338
##  Mcnemar's Test P-Value : 0.0324
##
## Statistics by Class:
##
##                      Class: Horror Class: Romance Class: Scifi
## Sensitivity                 0.5000         0.3500       0.4375
## Specificity                 0.7778         0.7667       0.5882
## Pos Pred Value              0.4667         0.5000       0.3333
## Neg Pred Value              0.8000         0.6389       0.6897
## Prevalence                  0.2800         0.4000       0.3200
## Detection Rate              0.1400         0.1400       0.1400
## Detection Prevalence        0.3000         0.2800       0.4200
## Balanced Accuracy           0.6389         0.5583       0.5129
```

Ouch, not so good! As predicted, the classifier struggled with this tricky and small dataset.

# Part III: can title features improve our already-good PCA classifier?

In this part we revisit the analysis we have been refining, where we take the word corpus from movie descriptions, perform PCA, and classify using SVM. Our benchmark for this approach is 80% predictive accuracy. Can we do better by adding our title features before PCA?

```
feature <- read.csv("~/Documents/DrPH_open/Data Science 2/Final project/Milestone 3/feat
ure.csv")

#### add title features
feature <- cbind(feature, df[, c(2:6, 8, 12:13)])

######## from here down, code is identical to what's reported elsewhere
set.seed(1)
index <- sample(1:nrow(feature),50)

test.genre <- genre[index, "V2"]
test.feature <- feature[index,]

train.genre <- genre[-index, "V2"]
train.feature <- feature[-index,]

PCA.train <- prcomp(train.feature)

PCA.vectors <- PCA.train$rotation[,1:100]

PCA.score.train <- PCA.train$x[,1:100]

test.scaled <-
    scale(test.feature, center=PCA.train$center, scale=PCA.train$scale)

PCA.score.test <-
    test.scaled %*% PCA.train$rotation[,1:100]

train.df <- data.frame(train.genre,PCA.score.train)
colnames(train.df)[1]<- c("Labels")

test.df <- data.frame(test.genre,PCA.score.test)
colnames(test.df)[1]<- c("Labels")

tuned.params <-
    tune(svm, Labels~., kernel="radial", data=train.df, ranges=
             list(gamma=10^(-6:-3),cost=10^(1:4)))

gamma <- tuned.params$best.parameters$gamma
cost <- tuned.params$best.parameters$cost

model <- svm(Labels~., kernel="radial", data=train.df, gamma=gamma, cost=cost)

preds <- predict(model, newdata=test.df)

table(test.df$Labels,preds)
```

```
##          preds
##           Horror Romance Scifi
##   Horror      10       3     3
##   Romance      2      15     0
##   Scifi        3       3    11
```

```
mean(test.df$Labels==preds)
```

```
## [1] 0.72
```

It doesn't look like adding title features improved the fit–in fact, it may have hindered it. Given the troubles in predicting three classes based on title features, especially with this small dataset, it's not a big surprise!