Black Friday Analysis

Sanata Sy-Sahande

October 22, 2018

In this document, I first do some exploratory data analysis on the Black Friday dataset. I then run several prediction models to predict the amount purchased by a customer, and classification models to predict the product category of the purchase.

Exploratory analysis

The dataset includes information on about 5800+ customers, for about 3600+ products.

```
length(unique(bf$Product_ID)) #3k+ products
```

```
## [1] 3623
```

```
length(unique(bf$User_ID)) #6K customers
```

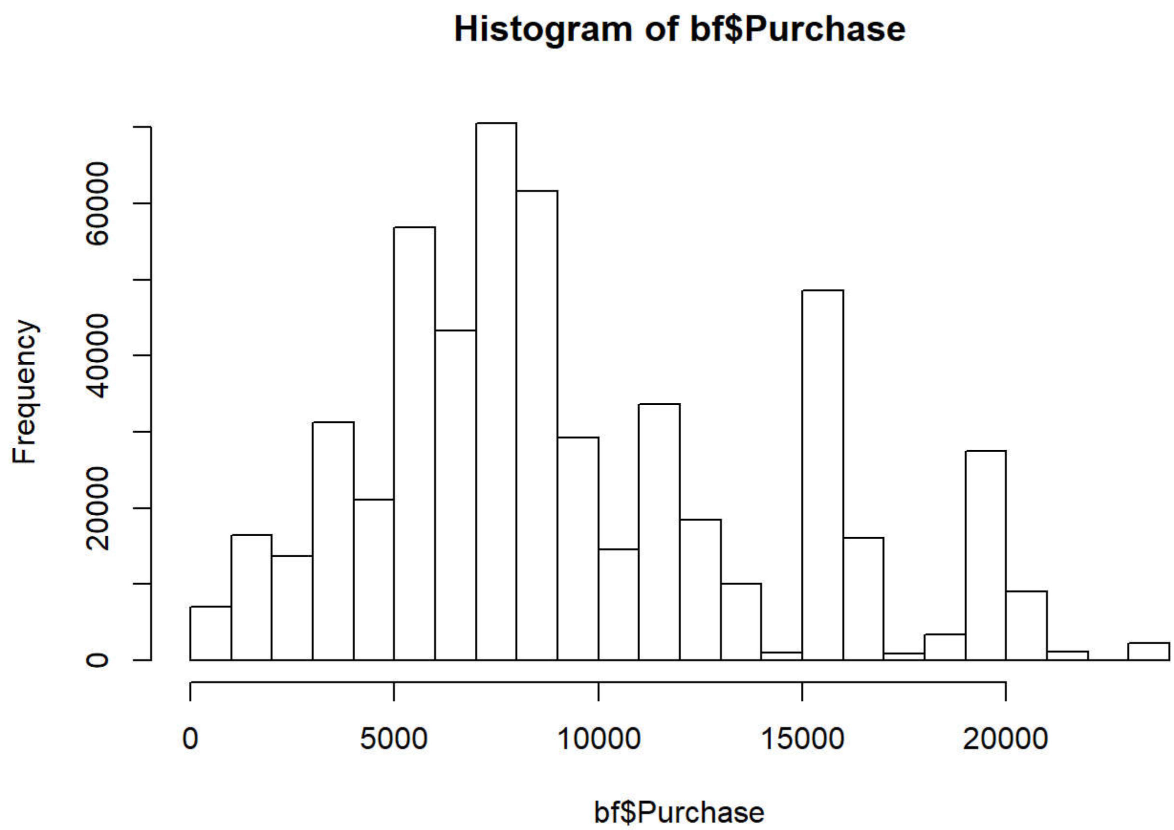
```
## [1] 5891
```

The average purchase is about $9300, and the variable is normally distributed, with spikes at $15000 and $20000.

```
mean(bf$Purchase)
```

```
## [1] 9333.86
```

```
hist(bf$Purchase)
```

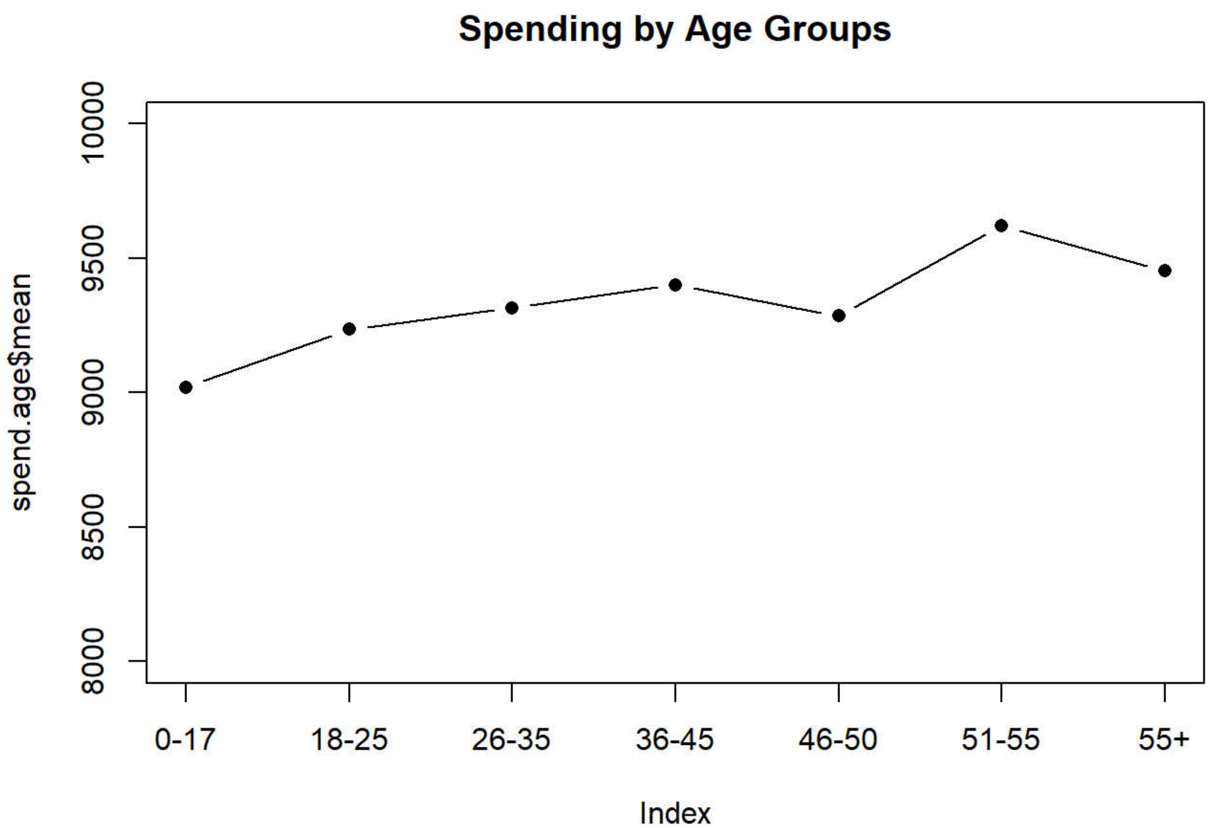


Purchases as a function of features

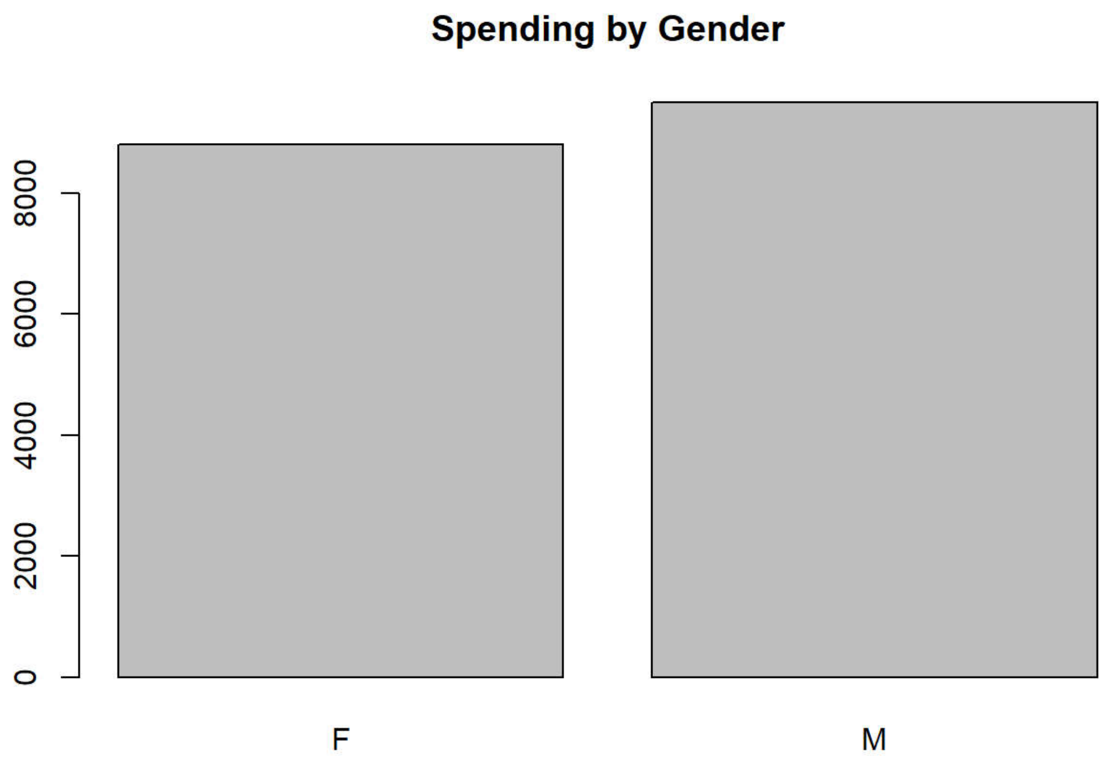
The next series of plots show how purchase amounts vary by the other variables in the dataset.

At first glance age, gender (men), and product category seem to be the best predictors of amount purchased.

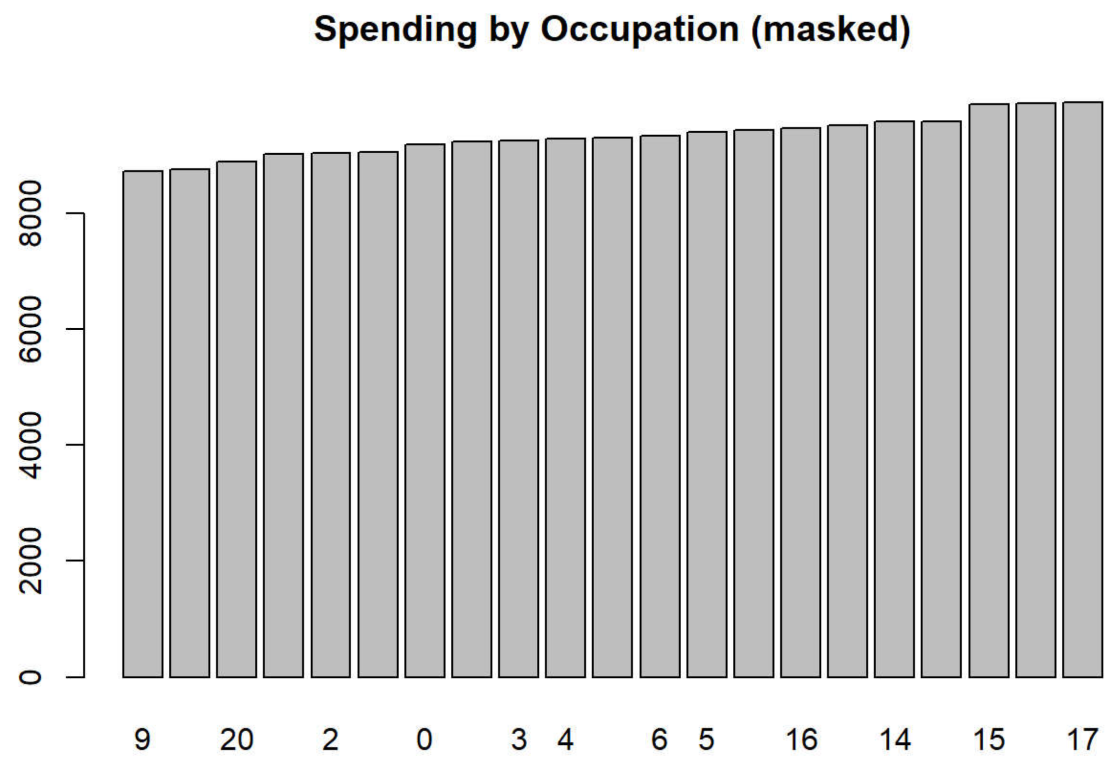
```
#by age
spend.age <- bf %>% group_by(Age) %>%
              summarise(mean = mean(Purchase))

plot(spend.age$mean, xaxt="n", pch=16, ylim=c(8000, 10000), type='b',
     main = "Spending by Age Groups")
axis(1, 1:7, labels=spend.age$Age)
```



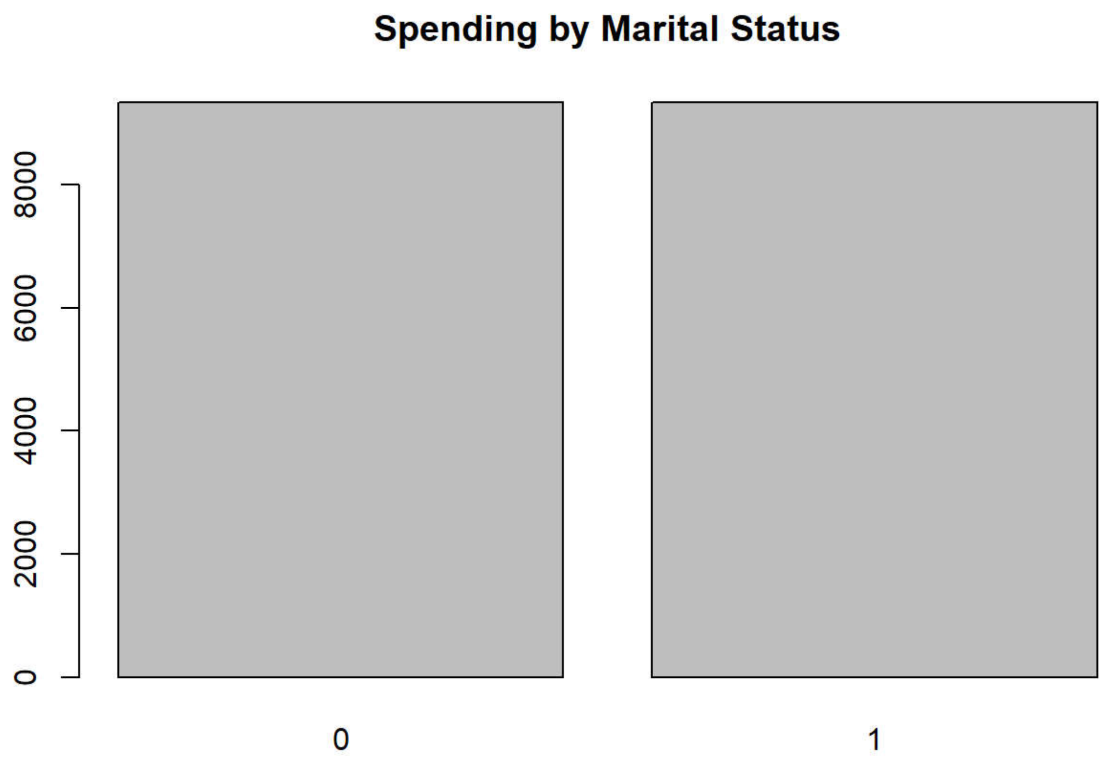

```
#by gender
spend.gender <- bf %>% group_by(Gender) %>%
                    summarise(mean = mean(Purchase))
barplot(spend.gender$mean, names=spend.gender$Gender,
        main = "Spending by Gender")
```



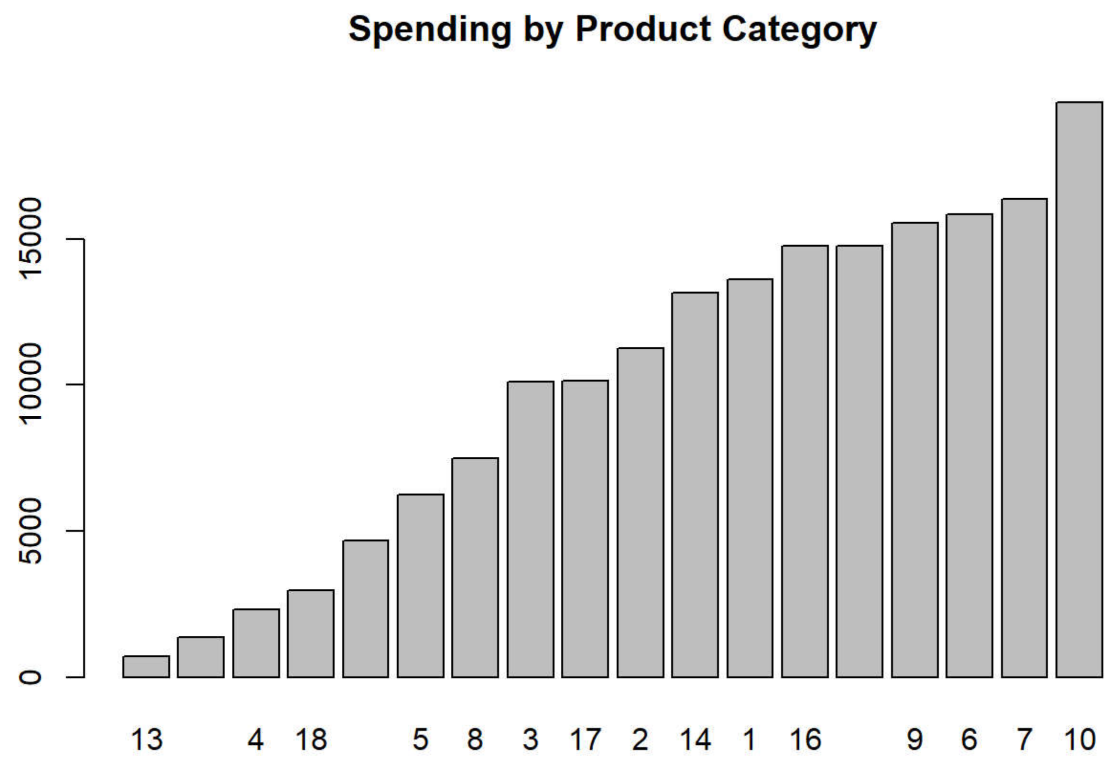

```r
#by occupation
spend.occ <- bf %>% group_by(Occupation) %>%
                 summarise(mean = mean(Purchase)) %>% arrange(mean)
barplot(spend.occ$mean, names=spend.occ$Occupation,
        main = "Spending by Occupation (masked)")
```



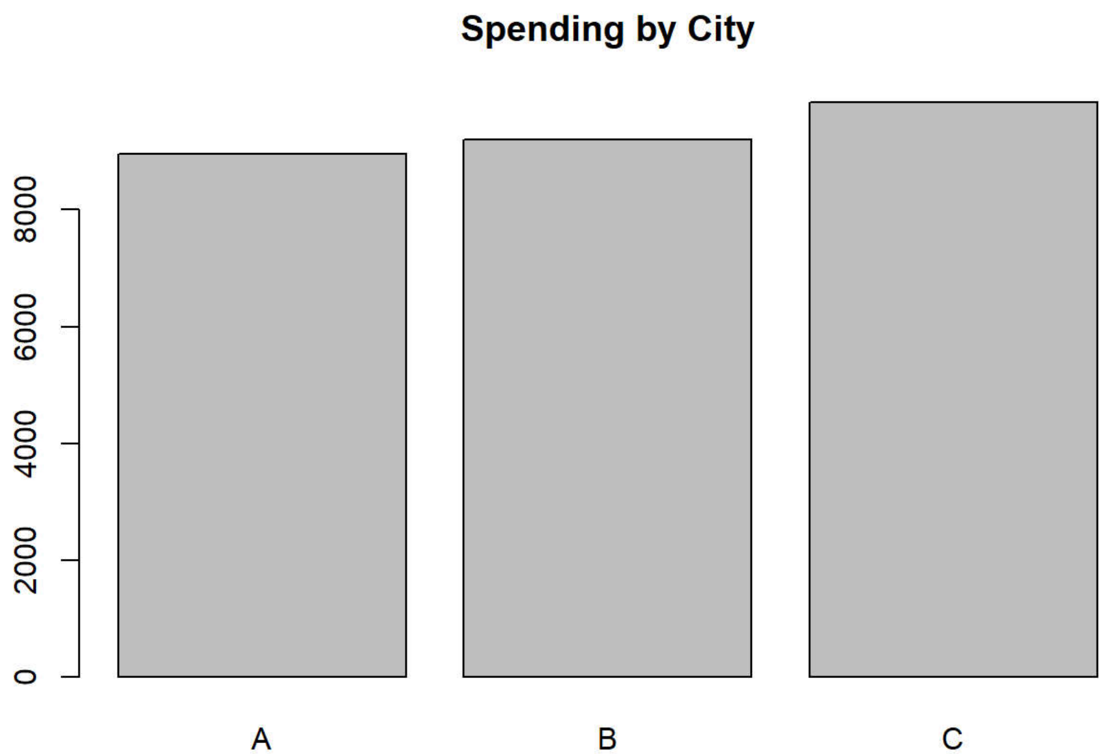

```
#by marital status
spend.mar <- bf %>% group_by(Marital_Status) %>%
              summarise(mean = mean(Purchase)) %>% arrange(mean)
barplot(spend.mar$mean, names=spend.mar$Marital_Status,
        main = "Spending by Marital Status")
```




```
#by category
spend.cat <- bf %>% group_by(Product_Category_1) %>%
              summarise(mean = mean(Purchase)) %>% arrange(mean)
barplot(spend.cat$mean, names=spend.cat$Product_Category_1,
        main = "Spending by Product Category")
```

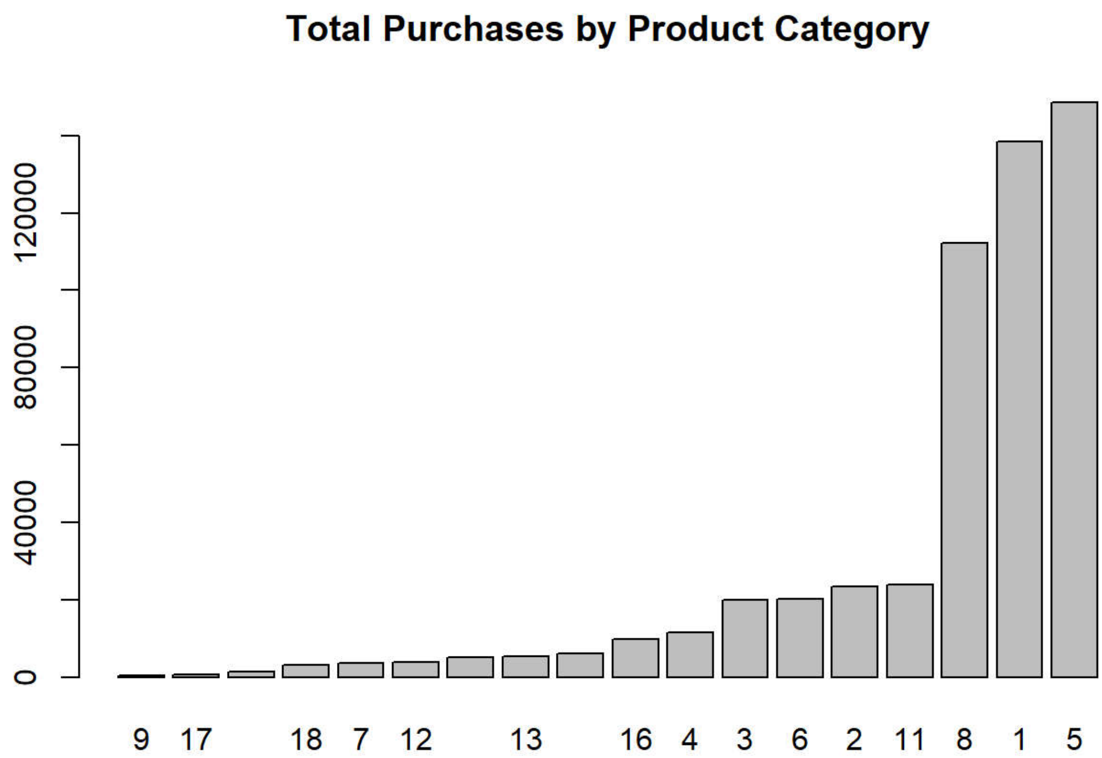



```
#by city
spend.city <- bf %>% group_by(City_Category) %>%
                summarise(mean = mean(Purchase)) %>% arrange(mean)
barplot(spend.city$mean, names=spend.city$City_Category,
        main = "Spending by City")
```



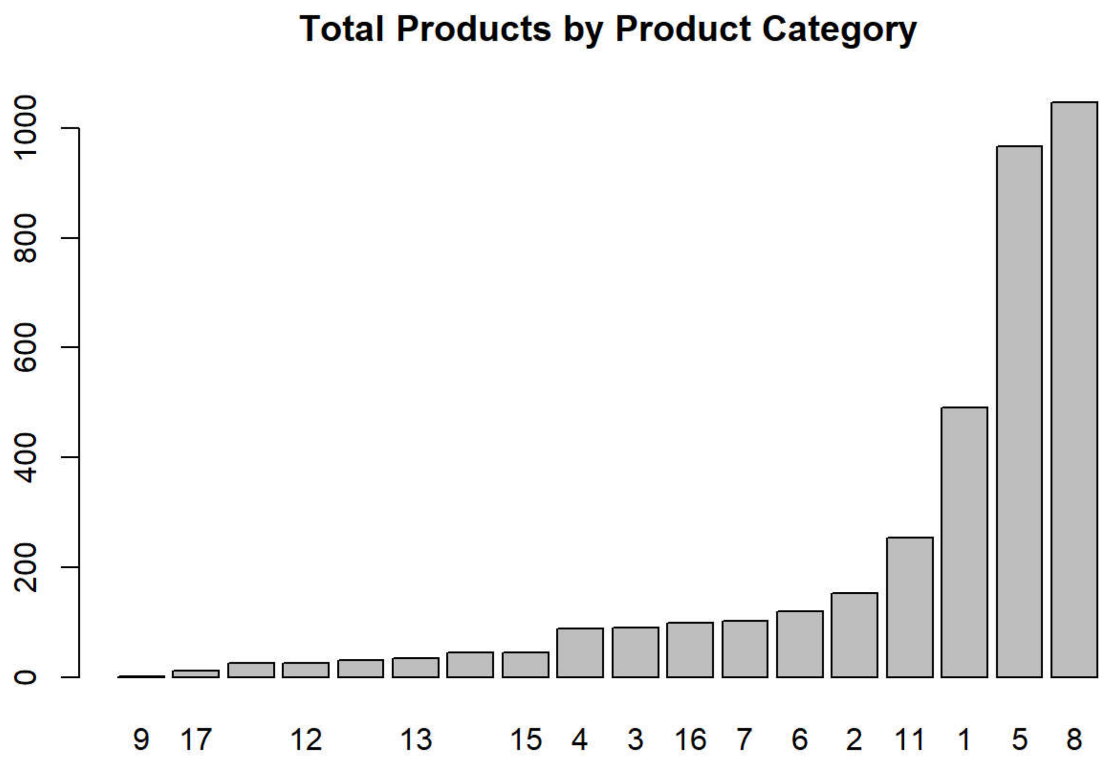
Next, I quickly check which product categories are most popular. Categories 1, 5, and 8 outnumber all other categories. This issue will come up later in the classification models.

```
##number of purchases per category
n.cat <- bf %>% group_by(Product_Category_1) %>%
              summarise(n = n()) %>% arrange(n)
barplot(n.cat$n, names = n.cat$Product_Category_1,
        main = "Total Purchases by Product Category")
```



I then check whether some categories just have more products, which would explain their popularity. Again, I find categories 1, 5, and 8 have the most products.

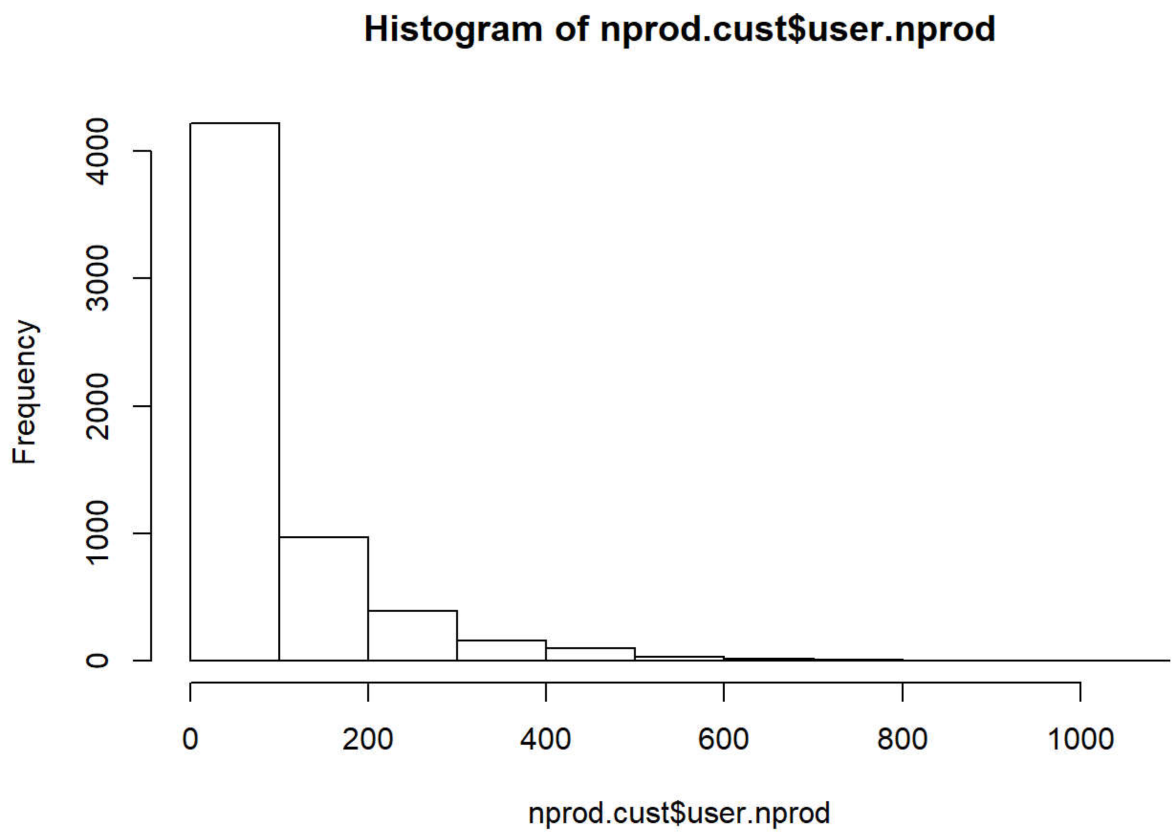
```
##number of products per category
n.prod <- bf %>% group_by(Product_Category_1) %>%
  summarise(nprod = n_distinct(Product_ID)) %>% arrange(nprod)
barplot(n.prod$nprod, names = n.prod$Product_Category_1,
        main = "Total Products by Product Category")
```



Data cleaning

I did some data cleaning to add and edit variables.

```
#Data cleaning: product and purchase by customer
nprod.cust <- bf %>% group_by(User_ID) %>%
        summarise(user.nprod = n_distinct(Product_ID))
hist(nprod.cust$user.nprod)
```




```
summary(nprod.cust$user.nprod)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   25.00   53.00   91.25  114.00 1025.00
```

```
purch.cust <- bf %>% group_by(User_ID) %>%
              summarise(totspend = sum(Purchase))
#add to dataset
bf <- left_join(bf, nprod.cust, by="User_ID")
bf <- left_join(bf, purch.cust, by="User_ID")


#Data cleaning: drop variables to simplify classifictaion
bf <- bf %>% select(-c(Product_Category_2,Product_Category_3))


#Data cleaning: edit variables
bf <- bf %>% mutate(Age = recode(Age, "0-17"=0,
                                      "18-25"=1,
                                      "26-35"=2,
                                      "36-45"=3,
                                      "46-50"=4,
                                      "51-55"=5,
                                      "55+"=6))
bf <- bf %>% mutate(Stay_In_Current_City_Years =
                      recode(Stay_In_Current_City_Years,
                              "0"=0, "1"=1, "2"=2, "3"=3, "4+"=4))

#Data cleaning: convert to factors
bf$Gender <- as.factor(bf$Gender)
bf$Occupation <- as.factor(bf$Occupation)
bf$City_Category <- as.factor(bf$City_Category )
bf$Product_Category_1 <- as.factor(bf$Product_Category_1)

#Data prep: make training and validation
set.seed(1)
train = sample(1:nrow(bf), nrow(bf)/2)
```

Regression: Predict Purchase Amount

I run several models to predict Purchase based on customer features. I first calculate the baseline RMSE with a simple OLS regression.

```
#LINEAR REG
lm.bf <- lm(Purchase ~ Gender + Age + Occupation +
              City_Category + Stay_In_Current_City_Years +
              Marital_Status,
              data=bf[train, ])
#predict
yhat = predict(lm.bf, newdata=bf[-train, ])
mse <- mean((bf$Purchase[-train] - yhat)^2)
sqrt(mse)
```

```
## [1] 4957.321
```

RMSE is equal to 4957.3210844. This means that an OLS model is expected to predict the target purchase amount by a margin of $4957.3210844, equivalent to about half a standard deviation.

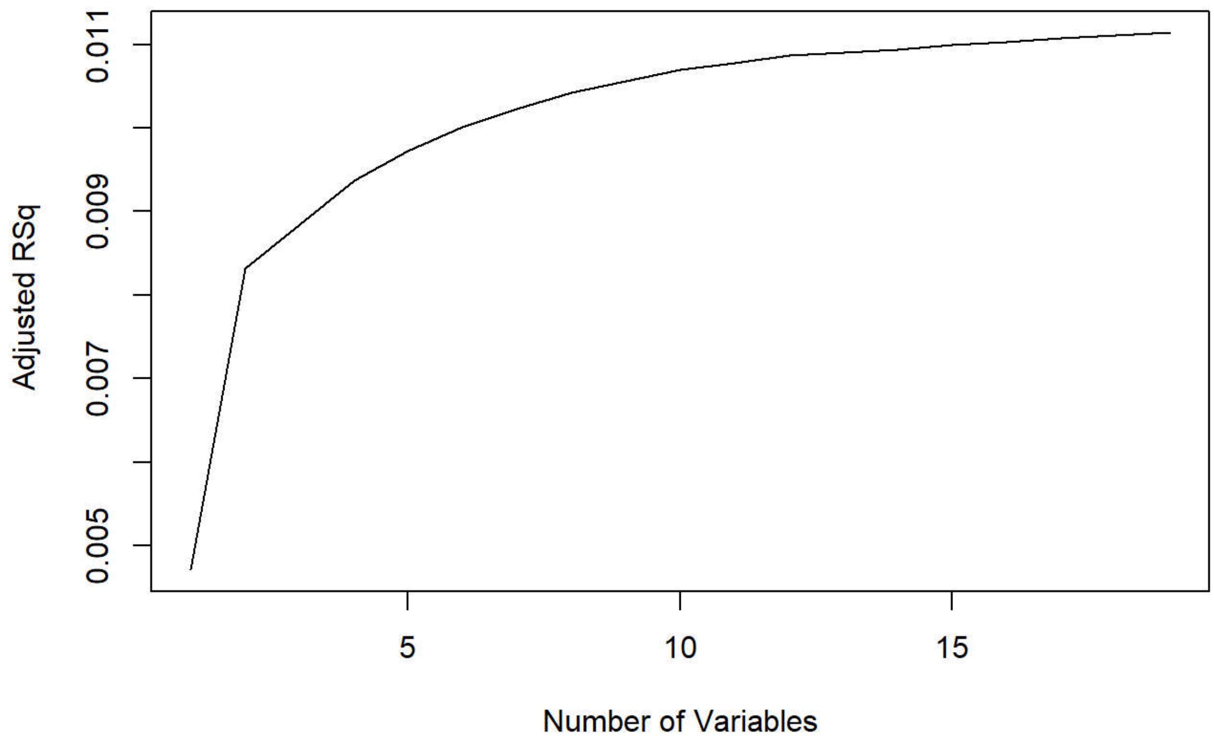
Feature Selection

To improve on the OLS model, I do some feature engineering to select the best variables to keep. This is especially useful since the dataset includes two categorical variables with many levels: occupation (21 levels) and product category (18 levels). Together they add almost 40 features to the model.

I first determine the optimal number of variables to include in the model. I compare full subset selection, forward, and backward selection. I use adjusted R-squared as my evaluation metric.

Below are the results of the forward selection approach, whichindicates about 10 variables before the improvement in adjusted R-squared from adding additional variables becomes neglible. (Similar results for full and backward selection ommitted.)

```
library(leaps)
#forward selection
regfit.fwd = regsubsets(Purchase ~ Gender + Age + Occupation +
                            City_Category +
                           Stay_In_Current_City_Years +
                           Marital_Status, bf, nvmax=19,
                        method = "forward")
reg.summary = summary(regfit.fwd)

plot(reg.summary$adjr2 ,xlab =" Number of Variables ",
     ylab=" Adjusted RSq",type="l")
```




```
which.max(reg.summary$adjr2)
```

```
## [1] 19
```

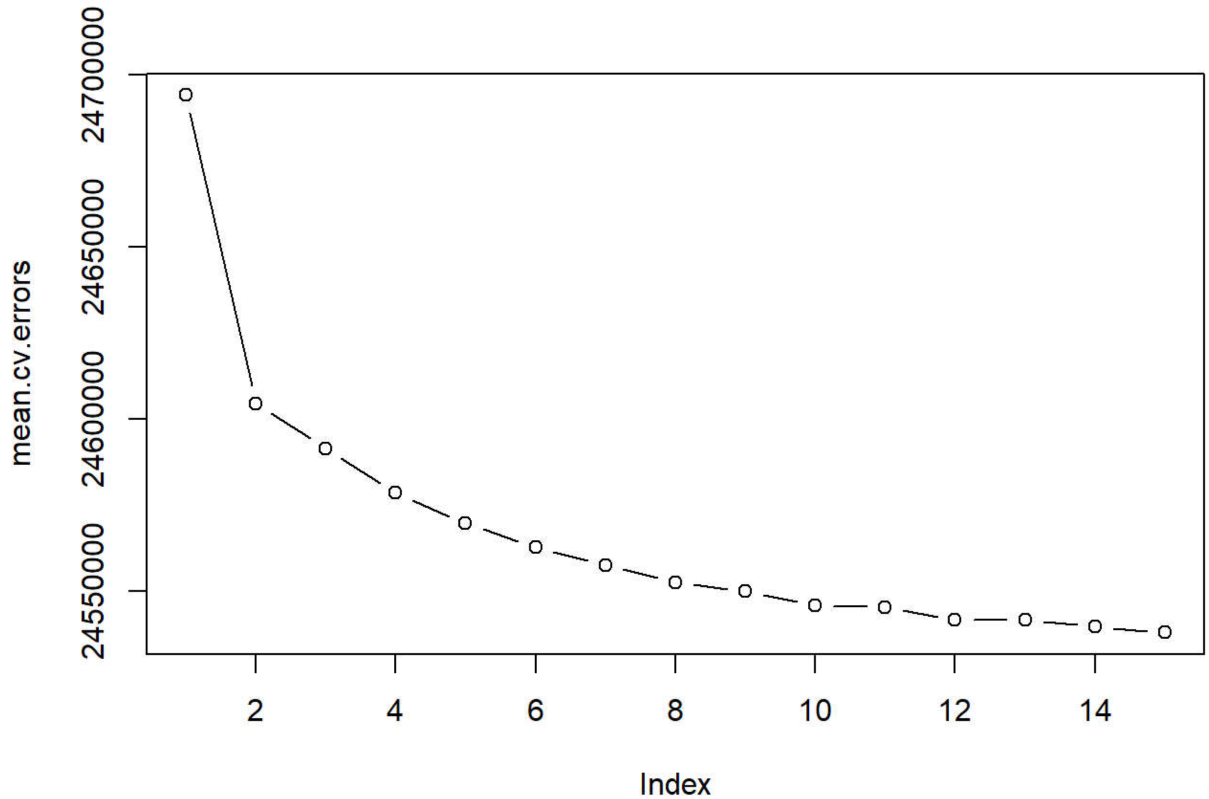
```
fwd.coefs <- coef(regfit.fwd, 10)
fwd.coefs
```

```
##    (Intercept)        GenderM    Occupation7   Occupation12   Occupation14
##      8402.4806       631.8746       198.5486       524.4904       339.5393
##   Occupation15   Occupation17   Occupation19   Occupation20 City_CategoryB
##       587.0321       506.2131      -505.8988      -247.4482       230.3504
## City_CategoryC
##       850.1810
```

```
#First, create a vector that allocates each observation to one of k = 10 folds
k=10
set.seed(1)
folds=sample(1:k,nrow(bf),replace =TRUE)
cv.errors = matrix(NA, k, 15, dimnames=list(NULL, paste(1:15) )) #matrix to store resu
lts

#make predict function
predict.regsubsets = function(object, newdata ,id ,...){
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id=id)
  xvars =names (coefi )
  mat[,xvars ]%*% coefi
}

#In the jth fold, the elements of folds that equal j
#are in the test set, and the remainder are in the training set
for(j in 1:k){
  best.fit = regsubsets(Purchase ~ Gender + Age + Occupation +
                           City_Category +
                          Stay_In_Current_City_Years +
                          Marital_Status,
                         data=bf[folds !=j,],
                         nvmax =15)
  for(i in 1:15) {
    pred=predict.regsubsets(best.fit, bf[folds==j,], id=i) #make predictions for each
model size
    #compute the test errors on the appropriate subset
    #store them in cv.errors
    cv.errors[j,i]=mean( (bf$Purchase[folds==j]-pred)^2)
  }
  #returns a kx15 matrix in cv.errors
}

mean.cv.errors = apply(cv.errors ,2, mean) #calculate errors for the j-variable model
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 24694029 24604631 24591561 24578726 24569726 24562970 24557629 24552690
##        9       10       11       12       13       14       15
## 24550119 24545840 24545483 24541803 24541852 24539767 24538128
```

```
par(mfrow =c(1,1))
plot(mean.cv.errors ,type='b')
```



As an alternative, I used cross-validation to determine the optimal number of variables in the model. I did this by performing best subset selection for up to 15 variables, within each of k=10 training sets. The results seem to confirm that 10-15 is the ideal range of variables. I select 12, and obtain the best 12 coefficients from the best subset method on the full sample.

```
reg.best = regsubsets(Purchase ~ Gender + Age + Occupation +
                            City_Category +
                           Stay_In_Current_City_Years +
                           Marital_Status, data=bf, nvmax=15)
coef(reg.best, 12)
```

```
##    (Intercept)        GenderM    Occupation1    Occupation7   Occupation10
##      8442.5744       623.7108      -175.2640       162.5461      -324.1805
##   Occupation12   Occupation14   Occupation15   Occupation17   Occupation19
##       488.8330       303.1415       550.9439       470.3583      -542.4365
##   Occupation20 City_CategoryB City_CategoryC
##      -283.1242       230.6716       857.8575
```

```
lm.fit <- lm(Purchase ~ Gender +
             (Occupation==1) + (Occupation==7) +
             (Occupation==10) + (Occupation==12) +
             (Occupation==14) + (Occupation==15) +
             (Occupation==17) + (Occupation==19) +
             (Occupation==20) + (City_Category=="B") +
             (City_Category=="C"), data=bf[train, ])
#predict
yhat = predict(lm.fit, newdata=bf[-train, ])
mse <- mean((bf$Purchase[-train] - yhat)^2) #MSE
sqrt(mse)
```

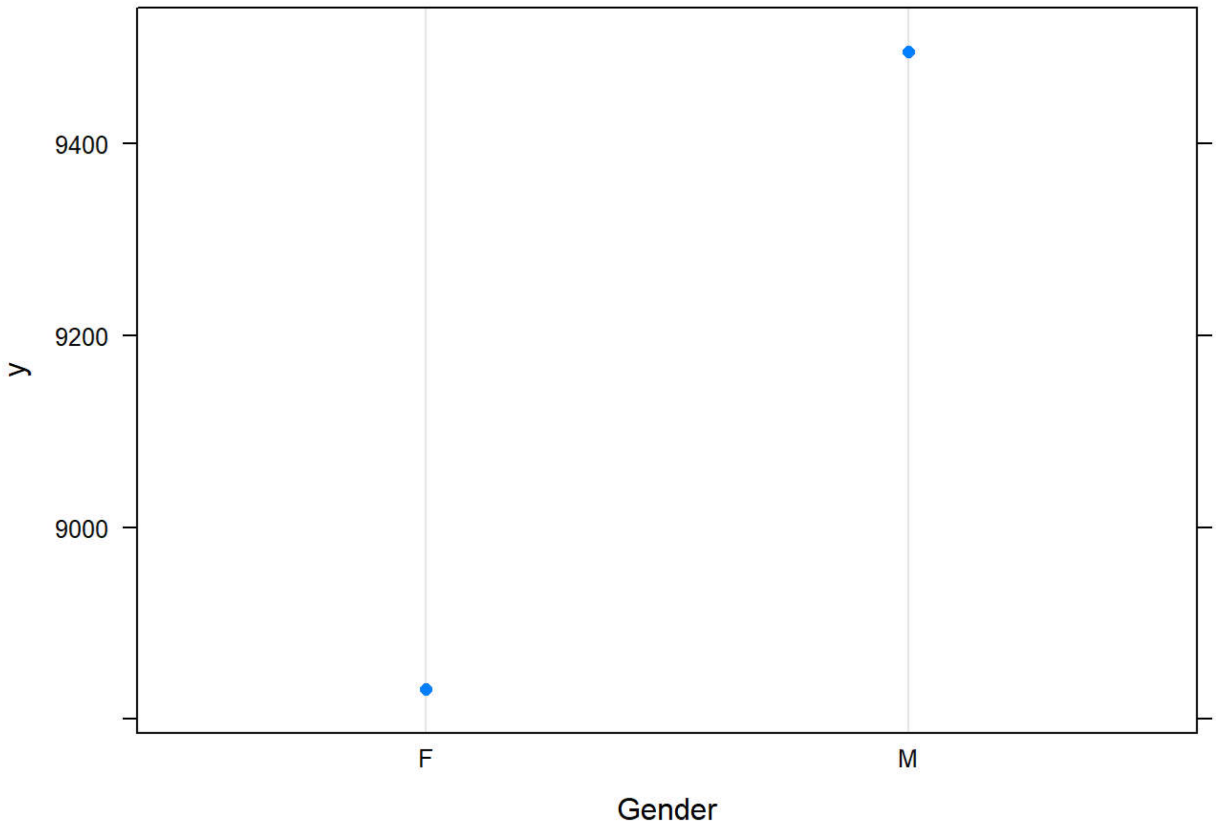
```
## [1] 4958.018
```

Selecting the best subset did not lead to a noticeable improvement in RMSE, which is now at 4958.0177481. This suggests that a linear model may not be the best model for the data. I turn to regression trees instead.

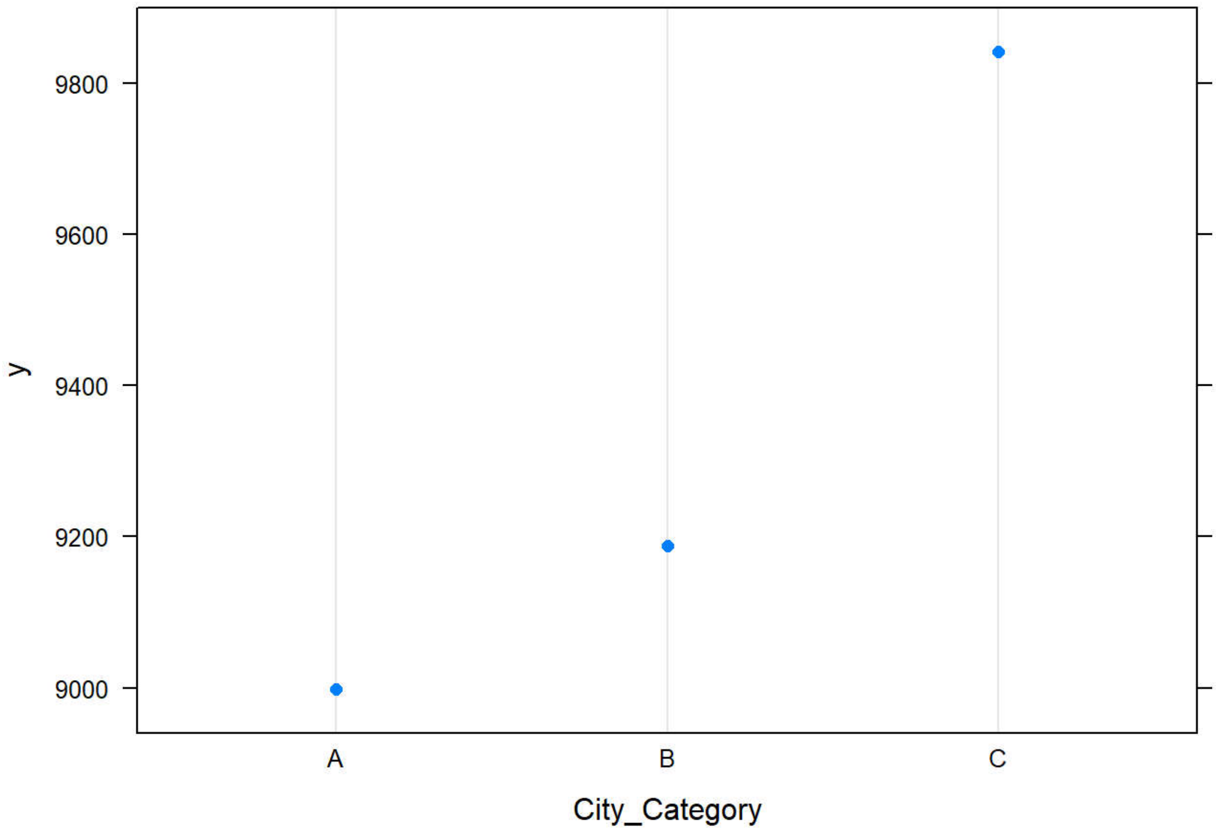
Regression Trees

In this section, I compare the performance of random forest with boosting to try to reduce the RMSE.

```
library(gbm)
boost.bf = gbm(Purchase ~ Gender + Age + Occupation +
                City_Category + Stay_In_Current_City_Years +
                Marital_Status,
                data=bf[train, ],
                distribution="gaussian",  #use ="bernoulli" for classification
                n.trees=500,
                interaction.depth = 4) #limit depth of trees


#partial dependence plots:marginal effect of selected vars
par(mfrow=c(1,2))
plot(boost.bf, i="Gender")
```




```
plot(boost.bf, i="City_Category")
```




```
#predict purchase on test set
yhat.boost=predict(boost.bf, newdata=bf[-train, ], n.trees=500)
mse <- mean((yhat.boost - bf$Purchase[-train])^2)
sqrt(mse) #RMSE = 4932
```

```
## [1] 4933.75
```

Again, the boosted trees reduced the RMSE, but not by much. My guess is that individual characteristics of consumers do not do as great a job of explaining the purchase amount as the products themselves. This is confirmed by comparing the R-squared of the original OLS model, and one including the product categories on the right hand side.

```
lm.bf <- lm(Purchase ~ Gender + Age + Occupation +
              City_Category + Stay_In_Current_City_Years +
              Marital_Status,
              data=bf[train, ])
sum.lm1 <- summary(lm.bf)
sum.lm1$r.squared
```

```
## [1] 0.01160177
```

```
lm.bf <- lm(Purchase ~ Gender + Age + Occupation +  Product_Category_1 +
              City_Category + Stay_In_Current_City_Years +
              Marital_Status,
              data=bf[train, ])
sum.lm2 <- summary(lm.bf)
sum.lm2$r.squared
```

```
## [1] 0.6318441
```

As suspected, including product categories increases the training R-squared from 0.01 to 0.63. But this is not particularly informative because it simply indicates that customers who by products from more expensive categories spend more money. It seems then that the appropriate model would be to predict *which categories* of goods customers will buy–a classification problem.

However, before continuing, I make a case for why the prediction models were still informative. Despite their low explanatory power, we now have a better idea of what types of customers tend to spend more: male customers, and those in a set of key occupations (masked). I would expect to see these same features be relevant for the classification models.

Classification

To build my classification model, I first order the product categories by price by calculating the average price of a product in each category. The target variable is now an ordered, multi-class response.

Attempt 1: Ordered logit

The setup lends itself well to a multinomial ordered logistic regression, where we estimate the odds that the customer will purchase a good in a category of increasing price. This first attempt has a response variable with 18 levels (one for each category), and as such I do not expect it to perform well.

```
#create ordered product category variable
order <- as.character(1:nrow(spend.cat))
labs <- ifelse(nchar(order)==1, paste0("0", order), order)
spend.cat$prod_cat_by_price <- labs
colnames(spend.cat)[2] <- "prod_cat_avg_price"
spend.cat$Product_Category_1 <- as.factor(spend.cat$Product_Category_1)
bf <- left_join(bf, spend.cat, by="Product_Category_1")
bf$prod_cat_by_price <- as.factor(bf$prod_cat_by_price)

library(MASS)
m <- polr(prod_cat_by_price ~ Gender + Age + Occupation +
            City_Category + Stay_In_Current_City_Years +
            Marital_Status,
          data = bf[train, ],
          Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = prod_cat_by_price ~ Gender + Age + Occupation +
##     City_Category + Stay_In_Current_City_Years + Marital_Status,
##     data = bf[train, ], Hess = TRUE)
##
## Coefficients:
##                                 Value Std. Error t value
## GenderM                     0.2586652   0.008202 31.5363
## Age                        -0.0105249   0.003108 -3.3868
## Occupation1                -0.0184715   0.015188 -1.2162
## Occupation2                -0.0569635   0.018269 -3.1181
## Occupation3                -0.0614250   0.021346 -2.8775
## Occupation4                 0.0018141   0.013882  0.1307
## Occupation5                 0.0187171   0.025219  0.7422
## Occupation6                -0.0155034   0.020307 -0.7635
## Occupation7                 0.0427129   0.014371  2.9721
## Occupation8                 0.1934373   0.064032  3.0210
## Occupation9                -0.1496883   0.033938 -4.4106
## Occupation10               -0.1417566   0.025508 -5.5574
## Occupation11               -0.0482814   0.025516 -1.8922
## Occupation12                0.1040111   0.017429  5.9679
## Occupation13               -0.0617400   0.031657 -1.9503
## Occupation14                0.0563024   0.018142  3.1034
## Occupation15                0.1071404   0.025032  4.2802
## Occupation16               -0.0122300   0.018846 -0.6489
## Occupation17                0.1311331   0.016129  8.1304
## Occupation18               -0.1624555   0.033405 -4.8632
## Occupation19               -0.1247913   0.029668 -4.2062
## Occupation20               -0.0617477   0.016878 -3.6584
## City_CategoryB              0.0249113   0.008500  2.9308
## City_CategoryC              0.1229365   0.009213 13.3437
## Stay_In_Current_City_Years  0.0007376   0.002665  0.2767
## Marital_Status              0.0023316   0.007374  0.3162
##
## Intercepts:
##       Value     Std. Error t value
## 01|02   -4.3842    0.0244  -179.9650
## 02|03   -3.8294    0.0209  -183.2579
## 03|04   -2.9967    0.0178  -168.2651
## 04|05   -2.8533    0.0175  -163.4055
## 05|06   -2.1119    0.0162  -130.1625
## 06|07   -0.3348    0.0153   -21.8258
## 07|08    0.5215    0.0154    33.9471
## 08|09    0.6765    0.0154    43.9646
## 09|10    0.6811    0.0154    44.2614
## 10|11    0.8708    0.0154    56.4474
## 11|12    0.8834    0.0154    57.2536
## 12|13    2.6231    0.0165   159.0411
```

```
## 13|14    2.8833    0.0169   171.0663
## 14|15    3.0904    0.0172   179.5640
## 15|16    3.1039    0.0172   180.0835
## 16|17    4.3504    0.0214   202.9346
## 17|18    4.9008    0.0250   195.7197
##
## Residual Deviance: 1060660.92
## AIC: 1060746.92
```

```
ctable <- coef(summary(m))

#validate ologit on test data
preds.p <- predict(m, newdat=bf[-train, ], type = "probs")
preds.class <- apply(preds.p, 1, which.max)
accuracy = mean(preds.class==bf$prod_cat_by_price[-train])
accuracy
```

```
## [1] 0.1076644
```

As suspected, the model only predicts the true product category at a rate of 0.1076644.

Attempt 2: Reduce number of categories

In this second attempt, I reduce the number of categories by breaking up the product price range into quintiles.

```
bf$prod_price_range <- cut(bf$prod_cat_avg_price,
                           breaks=c(0, quantile(bf$prod_cat_avg_price)), labels = 1:5)

m <- polr(prod_price_range ~ Gender + Age + Occupation +
            City_Category + Stay_In_Current_City_Years +
            Marital_Status,
          data = bf[train, ],
          Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = prod_price_range ~ Gender + Age + Occupation +
##     City_Category + Stay_In_Current_City_Years + Marital_Status,
##     data = bf[train, ], Hess = TRUE)
##
## Coefficients:
##                                 Value Std. Error  t value
## GenderM                     2.518e-01   0.008466 29.74839
## Age                        -7.174e-03   0.003184 -2.25285
## Occupation1                -3.458e-02   0.015584 -2.21878
## Occupation2                -6.735e-02   0.018789 -3.58439
## Occupation3                -7.762e-02   0.021977 -3.53181
## Occupation4                -1.360e-03   0.014251 -0.09541
## Occupation5                 2.604e-02   0.025787  1.00972
## Occupation6                -2.037e-02   0.020842 -0.97753
## Occupation7                 2.466e-02   0.014695  1.67840
## Occupation8                 1.985e-01   0.065776  3.01796
## Occupation9                -1.526e-01   0.035399 -4.31187
## Occupation10               -5.452e-02   0.026295 -2.07333
## Occupation11               -7.346e-02   0.026220 -2.80178
## Occupation12                9.358e-02   0.017854  5.24154
## Occupation13               -8.536e-02   0.032353 -2.63824
## Occupation14                2.411e-02   0.018605  1.29601
## Occupation15                9.683e-02   0.025685  3.76982
## Occupation16               -2.629e-02   0.019290 -1.36266
## Occupation17                1.284e-01   0.016496  7.78370
## Occupation18               -1.401e-01   0.033886 -4.13396
## Occupation19               -1.266e-01   0.030386 -4.16713
## Occupation20               -6.702e-02   0.017369 -3.85870
## City_CategoryB              3.507e-02   0.008731  4.01635
## City_CategoryC              1.251e-01   0.009453 13.22920
## Stay_In_Current_City_Years -7.306e-05   0.002732 -0.02674
## Marital_Status              3.468e-03   0.007565  0.45835
##
## Intercepts:
##     Value     Std. Error t value
## 1|2   -4.3815    0.0246  -178.1132
## 2|3   -0.3328    0.0158   -21.1151
## 3|4    0.5235    0.0158    33.1697
## 4|5    2.6225    0.0169   155.3861
##
## Residual Deviance: 705664.13
## AIC: 705724.13
```

```
ctable <- coef(summary(m))


#validate ologit on test data
preds.class <- predict(m, newdat=bf[-train, ])


#confusion matrix
table(preds.class, bf$prod_price_range[-train])
```

```
##
## preds.class     1     2     3     4     5
##           1     0     0     0     0     0
##           2  1526 52302 32228 42842 11712
##           3     0     0     0     0     0
##           4  1226 43229 23784 49071 10869
##           5     0     0     0     0     0
```

```
#accuracy
accuracy = mean(preds.class==bf$prod_price_range[-train])
accuracy
```

```
## [1] 0.3771471
```

As we can see from the confusion matrix, the model only assigns select classes while completely ignoring others. The predict function assigns the highest probability for each observation, which is sensitive to uneven sample probabilities. Even with class reduction, we still have a class imbalance problem since certain classes are extreme minorities:

```
prop.table(table(bf$prod_price_range))
```

```
##
##          1          2          3          4          5
## 0.01011948 0.35542629 0.20858779 0.34184498 0.08402145
```

Attempt 3: One-versus-All

Given imbalance, we could use an over- or under-sampling method to adjust the observed proportions of each class. I tried using SMOTE to balance the sample, but it did not improve predictions very much.

Instead, I opted to reduce the number of classes even further to just three: low-price, mid-price, and high-price items, each containing about a third of the data. I then used one-versus-all (calculating the probability of an observation being in class 1, 2, or 3 versus all other classes) to assign each observation to the class with the highest probability.

```r
#create new intervals
bf$prod_price_range <- cut(bf$prod_cat_avg_price,
                           breaks=3, labels = 1:3)
#create indicators
bf$prod_price_range1 <- ifelse(bf$prod_price_range==1, 1, 0)
bf$prod_price_range2 <- ifelse(bf$prod_price_range==2, 1, 0)
bf$prod_price_range3 <- ifelse(bf$prod_price_range==3, 1, 0)

#one-versus-all
glm.fit1 <- glm(prod_price_range1 ~ Gender + Age + Occupation +
                  City_Category + Stay_In_Current_City_Years +
                  Marital_Status,
                data = bf[train, ], family=binomial)
glm.fit2 <- glm(prod_price_range2 ~ Gender + Age + Occupation +
                  City_Category + Stay_In_Current_City_Years +
                  Marital_Status,
                data = bf[train, ], family=binomial)
glm.fit3 <- glm(prod_price_range3 ~ Gender + Age + Occupation +
                  City_Category + Stay_In_Current_City_Years +
                  Marital_Status,
                data = bf[train, ], family=binomial)

#get predicted probabilities of each class
pred1 <- predict(glm.fit1, bf[-train, ], type="response")
pred2 <- predict(glm.fit2, bf[-train, ], type="response")
pred3 <- predict(glm.fit3, bf[-train, ], type="response")
preds <- cbind(pred1, pred2, pred3)

#assign the class with the highest probability to each observation
pred.class <- apply(preds, 1, which.max)

#confusion matrix
table(pred.class, bf$prod_price_range[-train])
```

```
##
## pred.class     1     2     3
##          1 53891 42941 43434
##          2  3867  4120  2855
##          3 40525 31665 45491
```

```r
#accuracy rate
accuracy = mean(pred.class==bf$prod_price_range[-train])
accuracy
```

```
## [1] 0.3850678
```

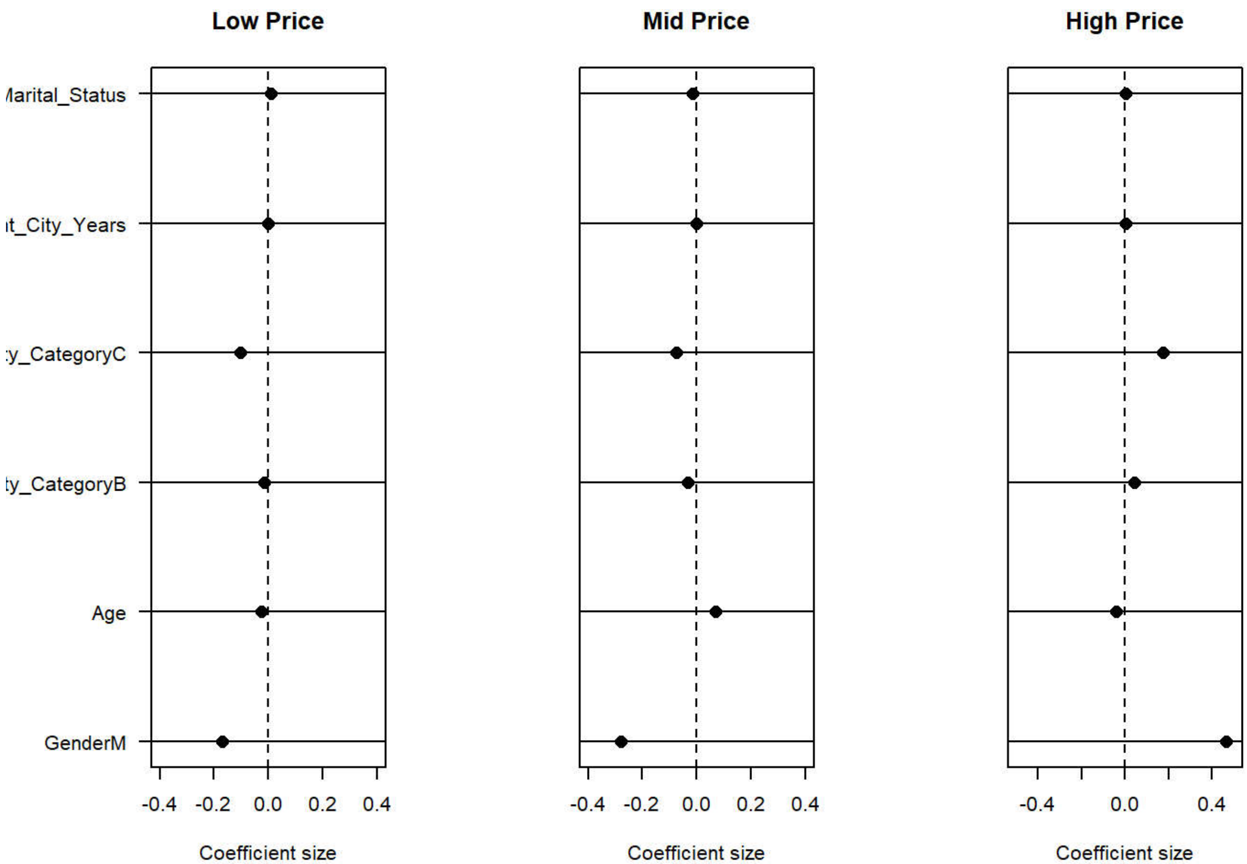
The OVA method performed the best, raising the accuracy rate to 0.3850678. While this is still not ideal, it is certainly a major improvement over the original ologit model.

Despite the significant limitations in accuracy rate, we can still answer our original question using this approach: what kinds of customers are likely to buy lower- versus higher-end products?

I answer this question by looking at the size and direction of the coefficients of each of the three OVA models. These are the original odds ratios of a logit regression. (With more time I would convert the coefficients to marginal effects for more ease of intepretation.)

I first look at the coefficients on gender, age, location, residency, and marital status. Gender and the City C location seem to have the largest association with a customer’s chosen price group. Male customers are less likely to buy low- and mid-priced products, and more likely to buy expensive products, as do those in City C. Age, residency in years, and marital status are especially not informative (as we also saw in the prediction model).

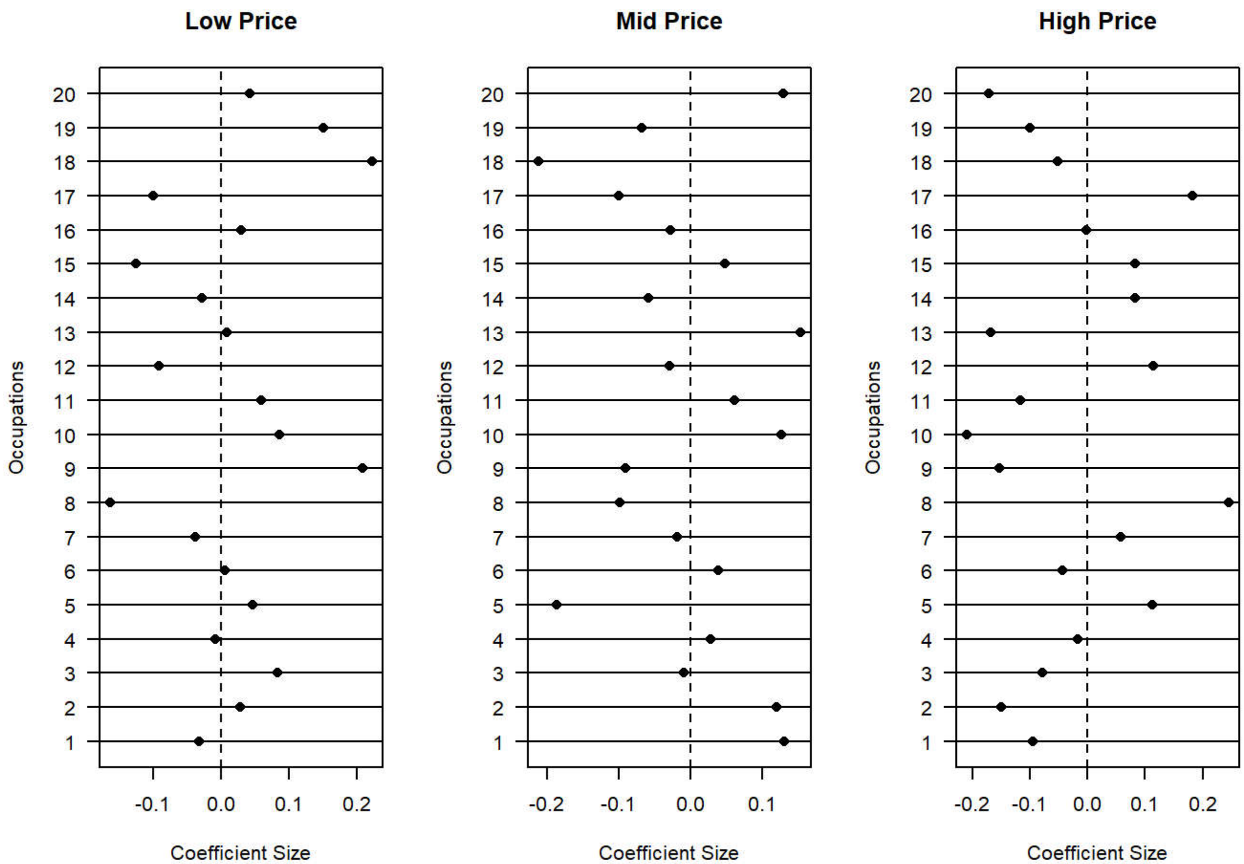
```
#plot coefficients by class
coefs <- cbind(coef(glm.fit1),coef(glm.fit2), coef(glm.fit3) )
ids <- which(row.names(coefs) %in% c("GenderM", "Age", "City_CategoryB", "City_Categor
yC", "Stay_In_Current_City_Years", "Marital_Status"))

par(mfrow=c(1, 3), mar=c(5,6,4,2))
plot(coefs[ids, 1], 1:length(ids), pch=16, cex=1.4,  xlim=c(-0.4, 0.4),
     xlab = "Coefficient size", ylab="", yaxt="n",
     main="Low Price")
abline(v=0, lty=2)
abline(h=c(1:length(ids)))
axis(side=2, at = 1:length(ids), labels=row.names(coefs)[ids], cex=.7, las=2)

plot(coefs[ids, 2], 1:length(ids), pch=16, cex=1.4,  xlim=c(-0.4, 0.4),
     xlab = "Coefficient size", ylab="", yaxt="n",
     main="Mid Price")
abline(v=0, lty=2)
abline(h=c(1:length(ids)))

plot(coefs[ids, 3], 1:length(ids), pch=16, cex=1.4,  xlim=c(-0.5, 0.5),
     xlab = "Coefficient size", ylab="", yaxt="n",
     main="High Price")
abline(v=0, lty=2)
abline(h=c(1:length(ids)))
```



Next, I looked at the coefficients of the occupation variable, which contains 20 levels. While this set of panels is a bit harder to read, certain occupations stand out as particularly associated with low-, mid-, and high-price products.

For instance, the store could target Occupations 18, 19, and 9 especially for low-price items, Occupations 20, 13, and 10, for mid-priced items, and Occupations 8 and 17 for high-priced items.

```
#plot occupation
occ.ids <- (1:nrow(coefs))[-c(1, ids)]

par(mfrow=c(1, 3), mar=c(5,4,4,2))

plot(coefs[occ.ids, 1], 1:length(occ.ids), pch=16, yaxt="n",
     ylab="Occupations", xlab="Coefficient Size", main="Low Price")
abline(h=c(1:length(occ.ids)))
abline(v=0, lty=2)
axis(side=2, at = 1:length(occ.ids), labels=1:20, las=2)

plot(coefs[occ.ids, 2], 1:length(occ.ids), pch=16, yaxt="n",
     ylab="Occupations", xlab="Coefficient Size", main="Mid Price")
abline(h=c(1:length(occ.ids)))
abline(v=0, lty=2)
axis(side=2, at = 1:length(occ.ids), labels=1:20, las=2)


plot(coefs[occ.ids, 3], 1:length(occ.ids), pch=16, yaxt="n",
     ylab="Occupations", xlab="Coefficient Size", main="High Price")
abline(h=c(1:length(occ.ids)))
abline(v=0, lty=2)
axis(side=2, at = 1:length(occ.ids), labels=1:20, las=2)
```



Recap

After some exploratory graphing, this analysis set out to predict Black Friday purchase amounts. Realizing that the problem was better suited for classification, I ran a series of logistic regression models to identify the customer features that would best predict purchases of higher-priced products. I found that male customers and customers in City C were the most reliable predictors of high-priced products. I also identified key occupations to target for low-, mid-, and high-priced products.

The accuracy of the model might be improved with unmasked data, which would allow for a more intuitive grouping of occupations to reduce this 20-level factor variable.