Logistic Regression

Contents

eBay Data

The original data consisting of eBay auctions for iPads was simplified and new variables derived. The goal of the analysis is to predict which iPad listed on eBay will be sold, a binary outcome. We have a number of pieces of information on each listing. Variables are described below.

1. UniqueID: Id
2. sold: Whether listed iPad sold (sold = 1) or not sold (sold = 0)
3. biddable = Whether this is an auction (biddable=1) or a sale with a fixed price (biddable=0).
4. startprice = The start price (in US Dollars) for the auction (if biddable=1) or the sale price (if biddable=0).
5. condition = The condition of the product (new, used, etc.)
6. cellular = Cellular connectivity for ipad (Cellular, No cellular or Unknown)
7. carrier = The cellular carrier for which the iPad is equipped (if cellular=1); listed as "None" if cellular=0
8. color = The color of the iPad - Black, Gold, Space Gray, Unknown or White
9. storage = The iPad's storage capacity (in gigabytes) - Less than 128 GB, 128 GB or Unknown
10. productline = Model of iPad being sold
11. noDescription: Whether the posting item has a description or not
12. charCountDescription: Length of description in terms of number of characters
13. upperCaseDescription: Number of upper case letters in the Description
14. startprice_99end: Whether the startprice ends in 99 (startprice_99end = 1) or not (startprice_99end = 0)

Read Data

Before running code below, be sure to set your working directory to point to the folder where the file is saved.

```
data = read.csv("eBayClean.csv", stringsAsFactors = T)
```

Split Data

```
library(caTools)
set.seed(617)
split = sample.split(data$sold,SplitRatio = 0.7)
train = data[split,]
test = data[!split,]
```

Explore

```
head(train)
```

```
##    UniqueID sold     biddable startprice condition    cellular carrier
## 1     10001    0 not biddable     159.99      used No cellular    None
## 3     10003    1 not biddable     199.99      used No cellular    None
## 6     10006    1     biddable     175.00      used    Cellular    AT&T
## 8     10008    0 not biddable     329.99       new No cellular    None
## 9     10009    1     biddable       0.99      used    Cellular Unknown
## 10    10010    1     biddable     150.00      used No cellular    None
##         color          storage productline        noDescription
## 1       Black Less than 128 GB      iPad 2 contains description
## 3       White Less than 128 GB      iPad 4       no description
## 6  Space Gray Less than 128 GB iPad mini 2       no description
## 8       White Less than 128 GB  iPad mini3       no description
## 9       White Less than 128 GB      iPad 1       no description
## 10      White Less than 128 GB      iPad 4       no description
##    charCountDescription upperCaseDescription startprice_99end
## 1                    45                    1        99 ending
```

```
## 3                         0                    0        99 ending
## 6                         0                    0  not a 99 ending
## 8                         0                    0        99 ending
## 9                         0                    0        99 ending
## 10                        0                    0  not a 99 ending
```

```
names(train)
```

```
##  [1] "UniqueID"              "sold"                  "biddable"
##  [4] "startprice"            "condition"             "cellular"
##  [7] "carrier"               "color"                 "storage"
## [10] "productline"           "noDescription"         "charCountDescription"
## [13] "upperCaseDescription" "startprice_99end"
```

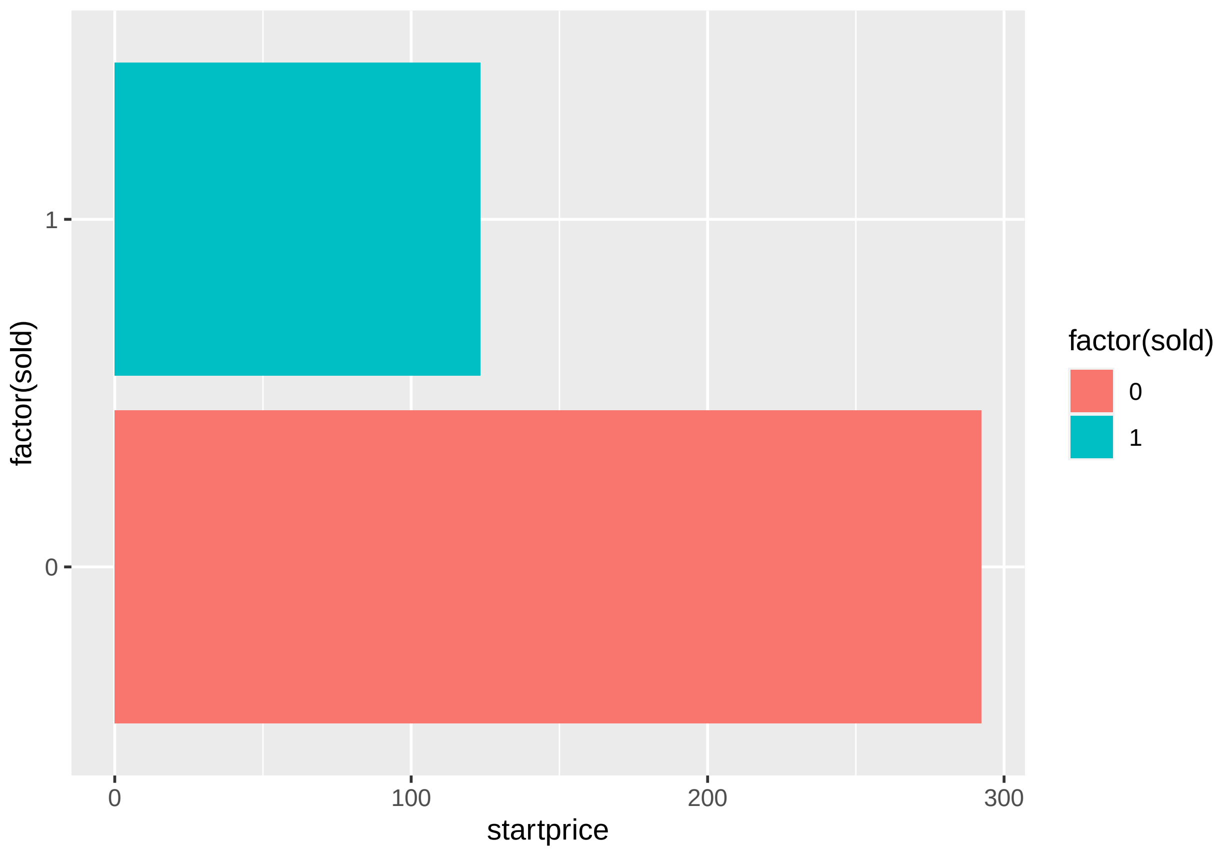
Examine relationships between each variable and `sold`

`startprice`

```
tapply(train$startprice,train$sold,mean)
```

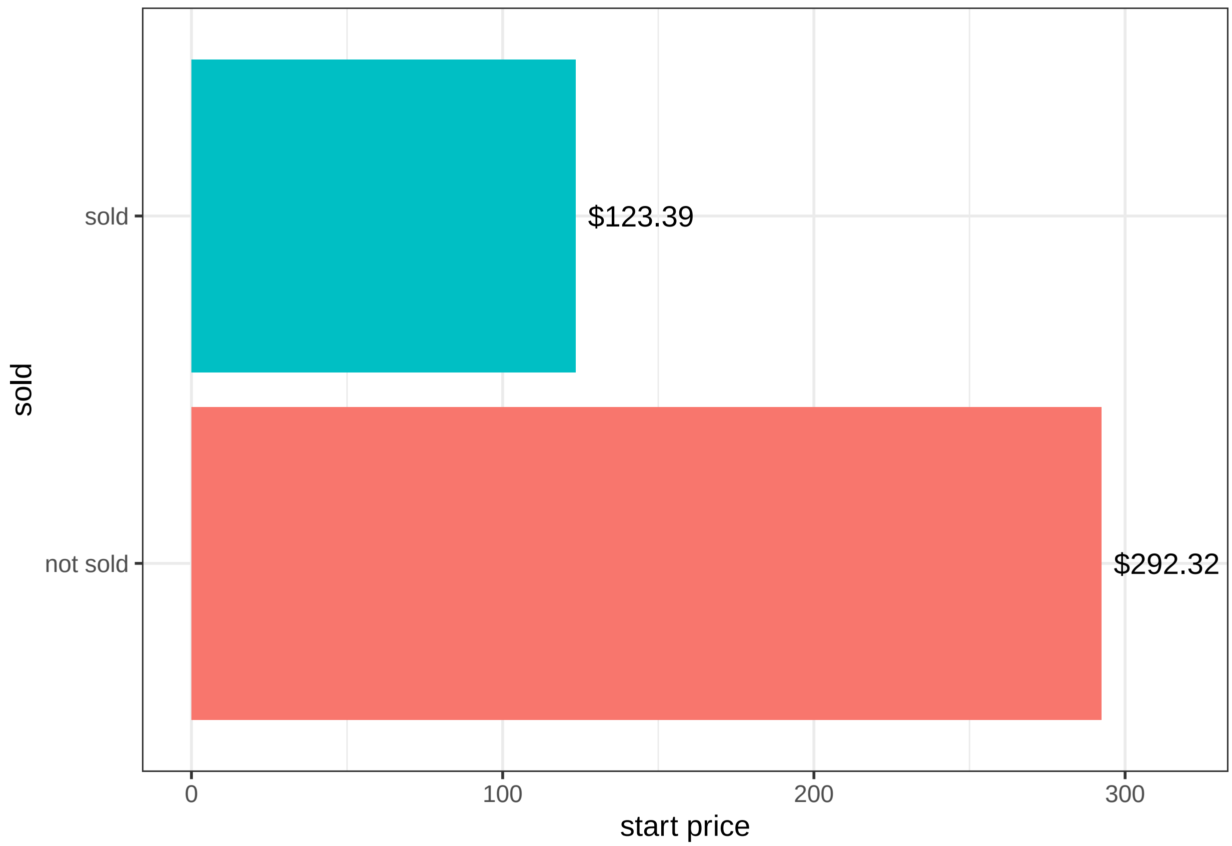
```
##        0        1
## 292.3205 123.3935
```

```
library(ggplot2)
ggplot(data=train,aes(x=factor(sold),y=startprice,fill=factor(sold)))+
  geom_bar(stat='summary',fun='mean')+
  coord_flip()
```



While the chart below is cleaner and more informative, it requires much more code.

```r
library(dplyr)
train%>%
  mutate(sold = factor(sold,labels = c('not sold','sold')))%>%
  group_by(sold)%>%
  summarize(avg_startprice = mean(startprice))%>%
  ungroup()%>%
  ggplot(aes(x=sold,y=avg_startprice,fill=sold))+
  geom_col()+
  geom_text(aes(x=sold,y=avg_startprice,
                label=paste0('$',round(avg_startprice,2),'    ')),
            nudge_y=25)+
  ylab('start price')+
  guides(fill=F)+
  coord_flip()+
  theme_bw()
```



At this stage, we only interested in exploring the data, so we will stick with simpler charts.

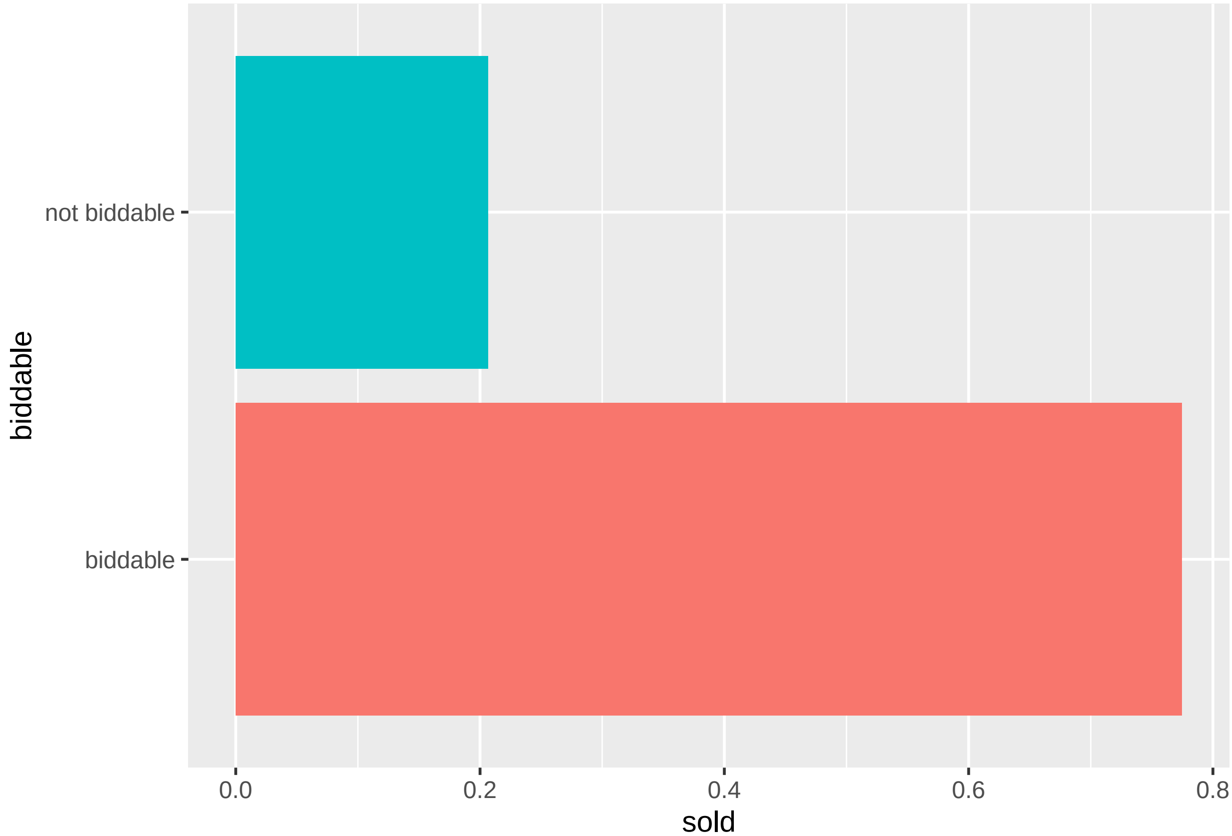
`biddable`

Unlike the `startprice`, `biddable` is a binary variable, so we have plotted biddable against the proportion of ipads sold.

```r
tapply(train$sold,train$biddable,mean)
```

```
##      biddable not biddable
##     0.7747440    0.2064156
```

```r
ggplot(data=train,aes(x=biddable,y=sold,fill=biddable))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

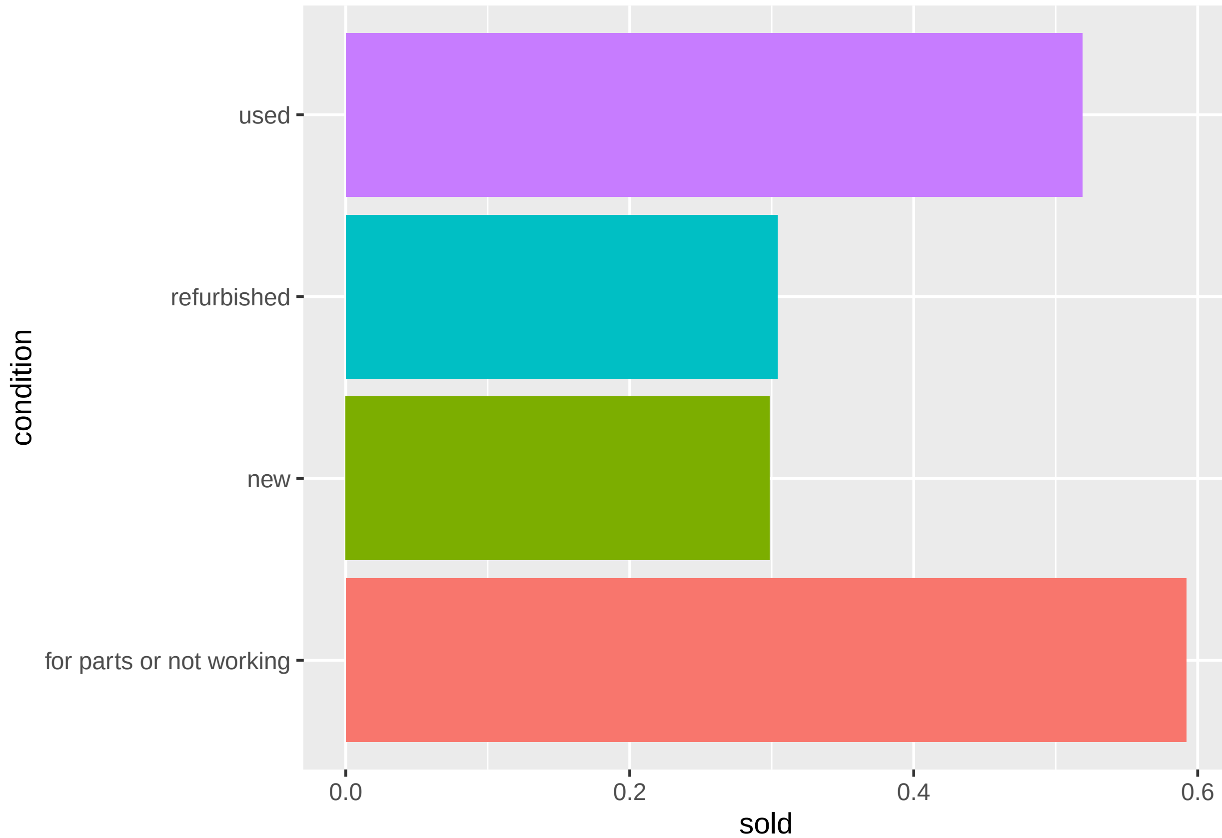


condition

```r
tapply(train$sold,train$condition,mean)
```

```
## for parts or not working                      new              refurbished
##                0.5920000                0.2985612                0.3039216
##                     used
##                0.5187970
```

```r
ggplot(data=train,aes(x=condition,y=sold,fill=condition))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

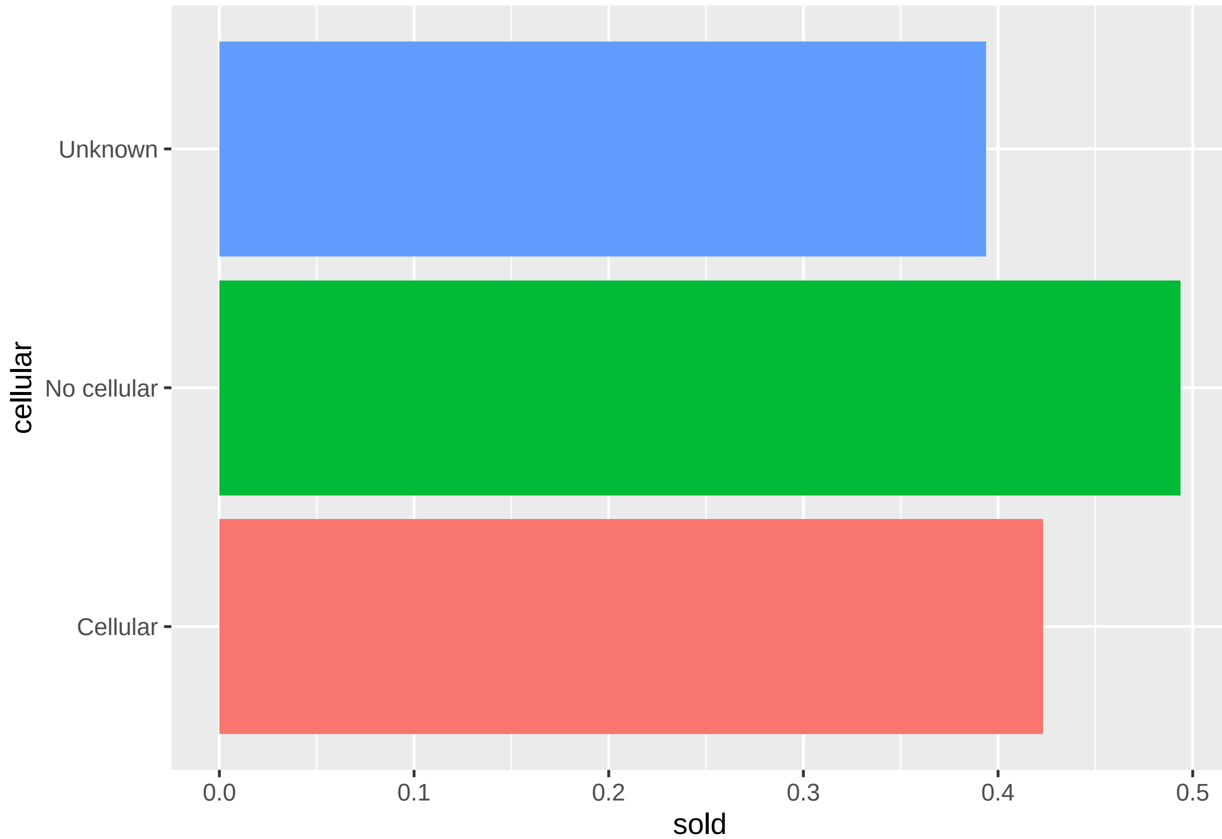


cellular

```
tapply(train$sold,train$cellular,mean)
```

```
##    Cellular No cellular     Unknown
##   0.4229692   0.4936387   0.3937500
```

```
ggplot(data=train,aes(x=cellular,y=sold,fill=cellular))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

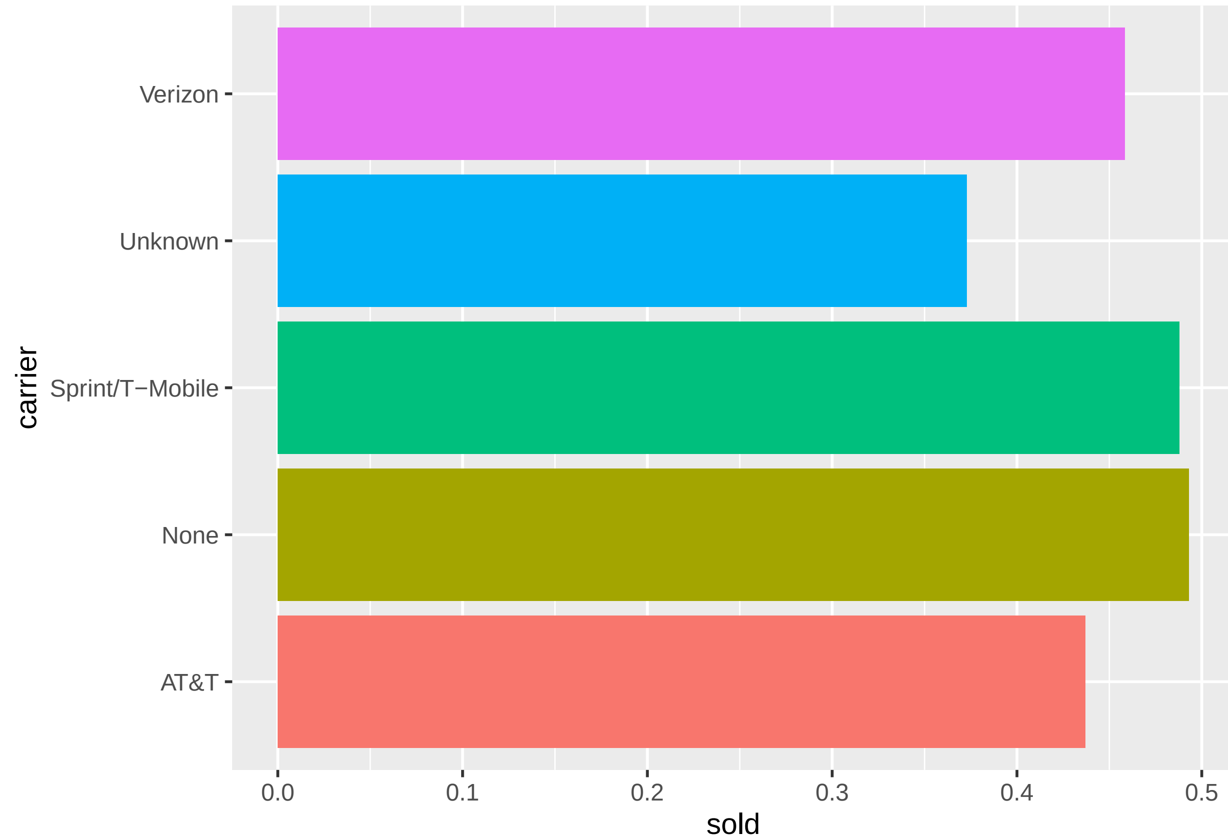


carrier

```
tapply(train$sold,train$carrier,mean)
```

```
##            AT&T            None Sprint/T-Mobile         Unknown         Verizon
##       0.4370370       0.4930114       0.4878049       0.3729508       0.4583333
```

```
ggplot(data=train,aes(x=carrier,y=sold,fill=carrier))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

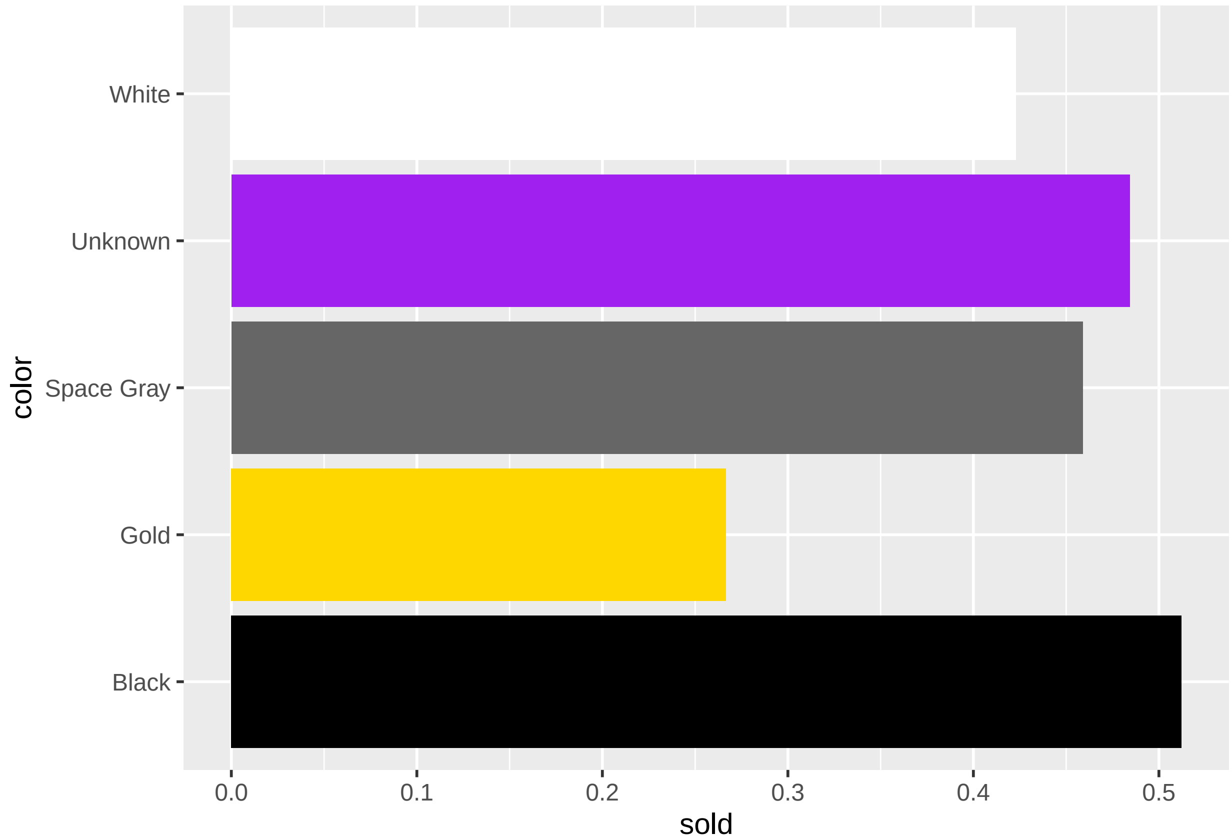


color

```
tapply(train$sold,train$color,mean)
```

```
##      Black       Gold Space Gray    Unknown      White
##  0.5121107  0.2666667  0.4589041  0.4842767  0.4229607
```

```
ggplot(data=train,aes(x=color,y=sold))+
  geom_bar(stat='summary',
           fun='mean',
           fill=c('black','gold','grey40','purple','white'))+
  coord_flip()
```

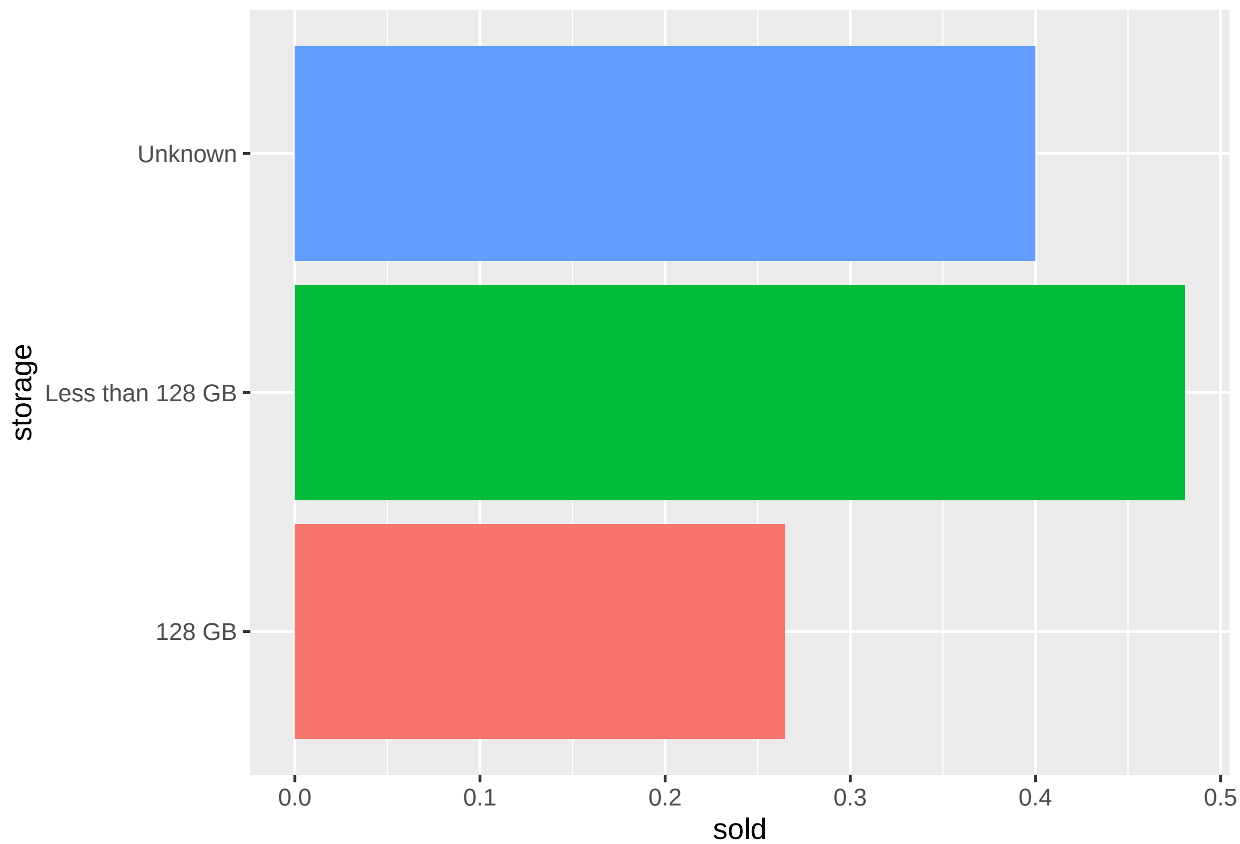


storage

```
tapply(train$sold,train$storage,mean)
```

```
##           128 GB Less than 128 GB          Unknown
##        0.2647059        0.4807175        0.4000000
```

```
ggplot(data=train,aes(x=storage,y=sold,fill=storage))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

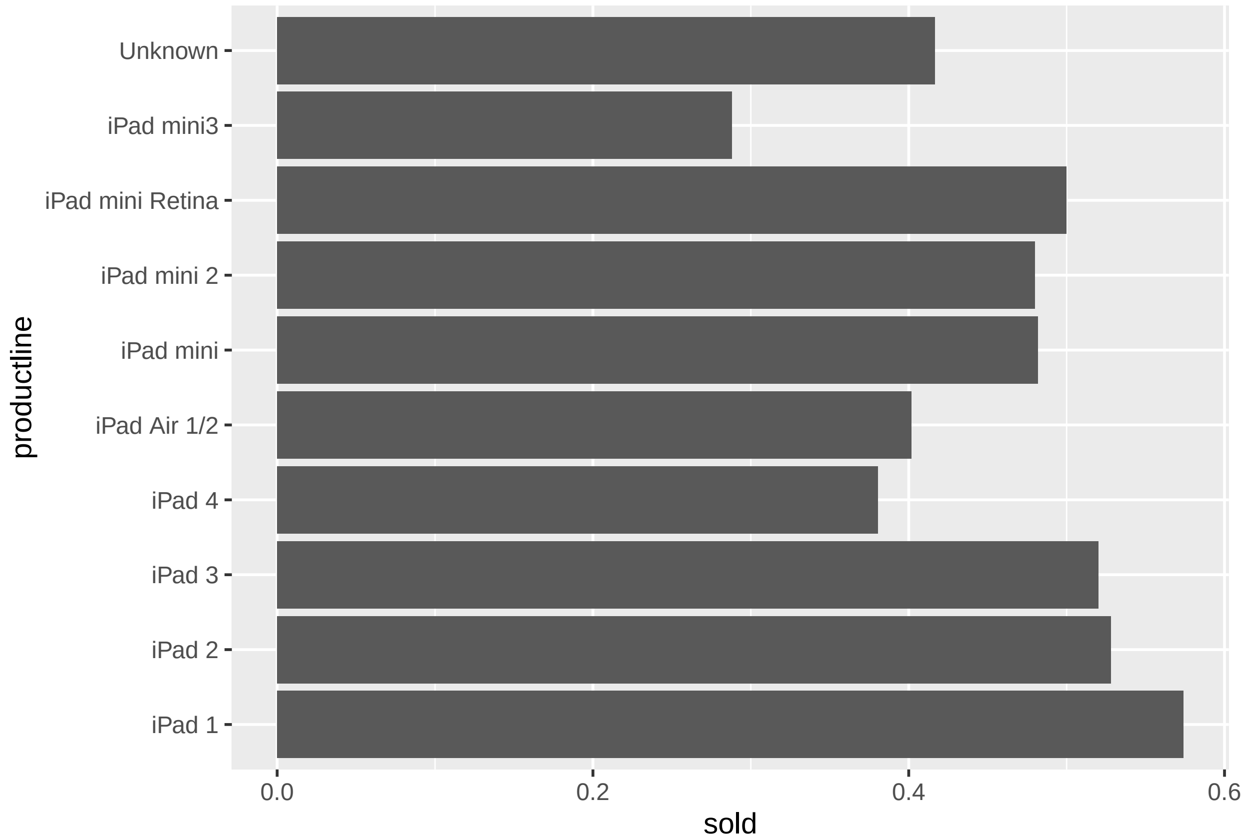


productline

```
tapply(train$sold,train$productline,mean)
```

```
##          iPad 1           iPad 2           iPad 3           iPad 4
##       0.5740741        0.5282051        0.5200000        0.3805310
##    iPad Air 1/2        iPad mini      iPad mini 2 iPad mini Retina
##       0.4016064        0.4818653        0.4800000        0.5000000
##       iPad mini3          Unknown
##       0.2878788        0.4166667
```

```
ggplot(data=train,aes(x=productline,y=sold))+
  geom_bar(stat='summary',fun='mean')+
  coord_flip()
```

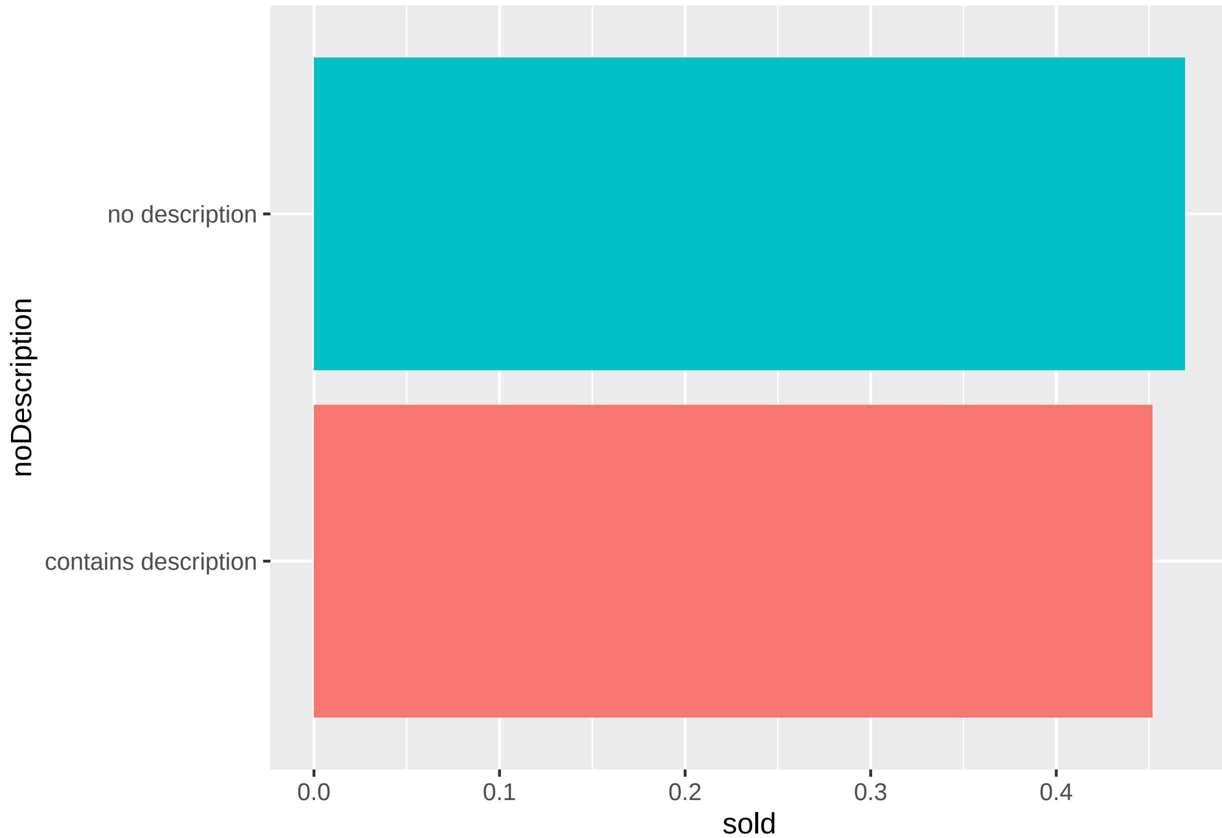


no description

```
tapply(train$sold,train$noDescription,mean)
```

```
## contains description       no description
##            0.4516729            0.4692810
```

```
ggplot(data=train,aes(x=noDescription,y=sold,fill=noDescription))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

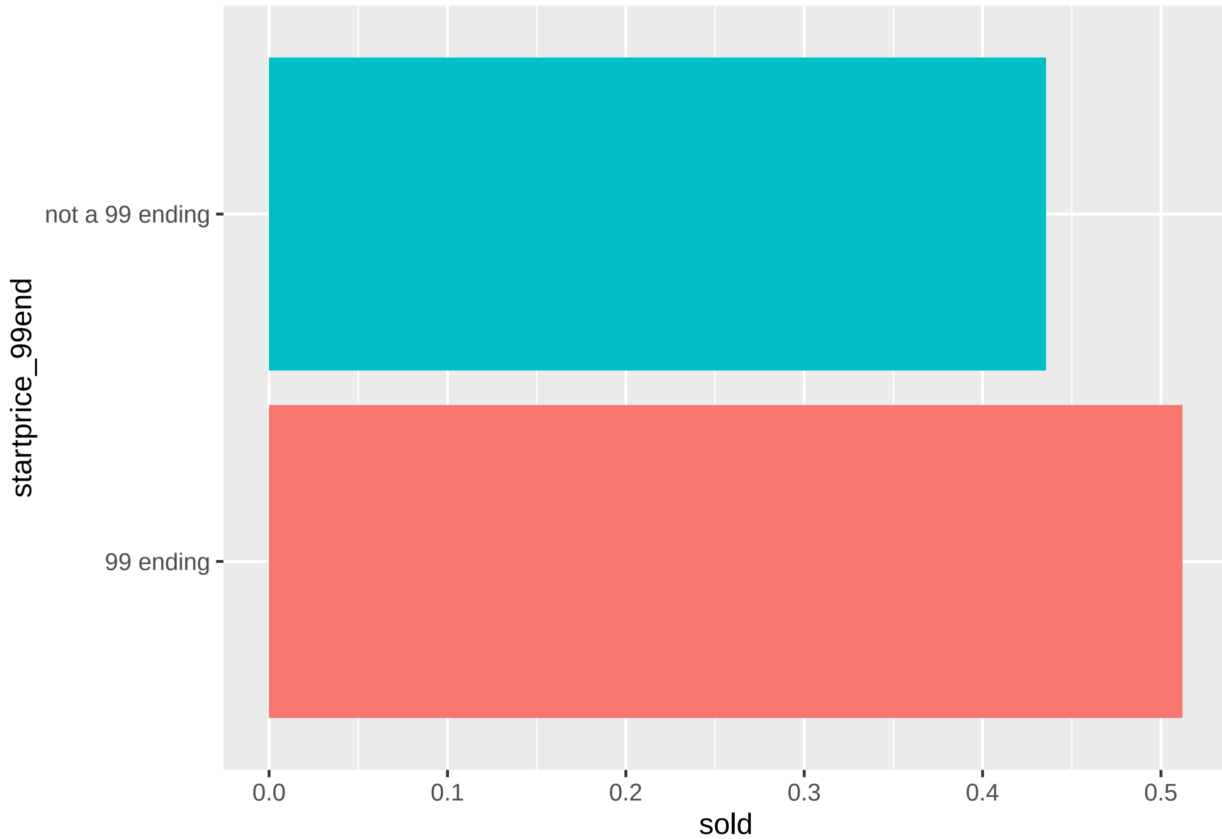


99 ending

```
tapply(train$sold,train$startprice_99end,mean)
```

```
##       99 ending not a 99 ending
##       0.5121951       0.4354460
```

```
ggplot(data=train,aes(x=startprice_99end,y=sold,fill=startprice_99end))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

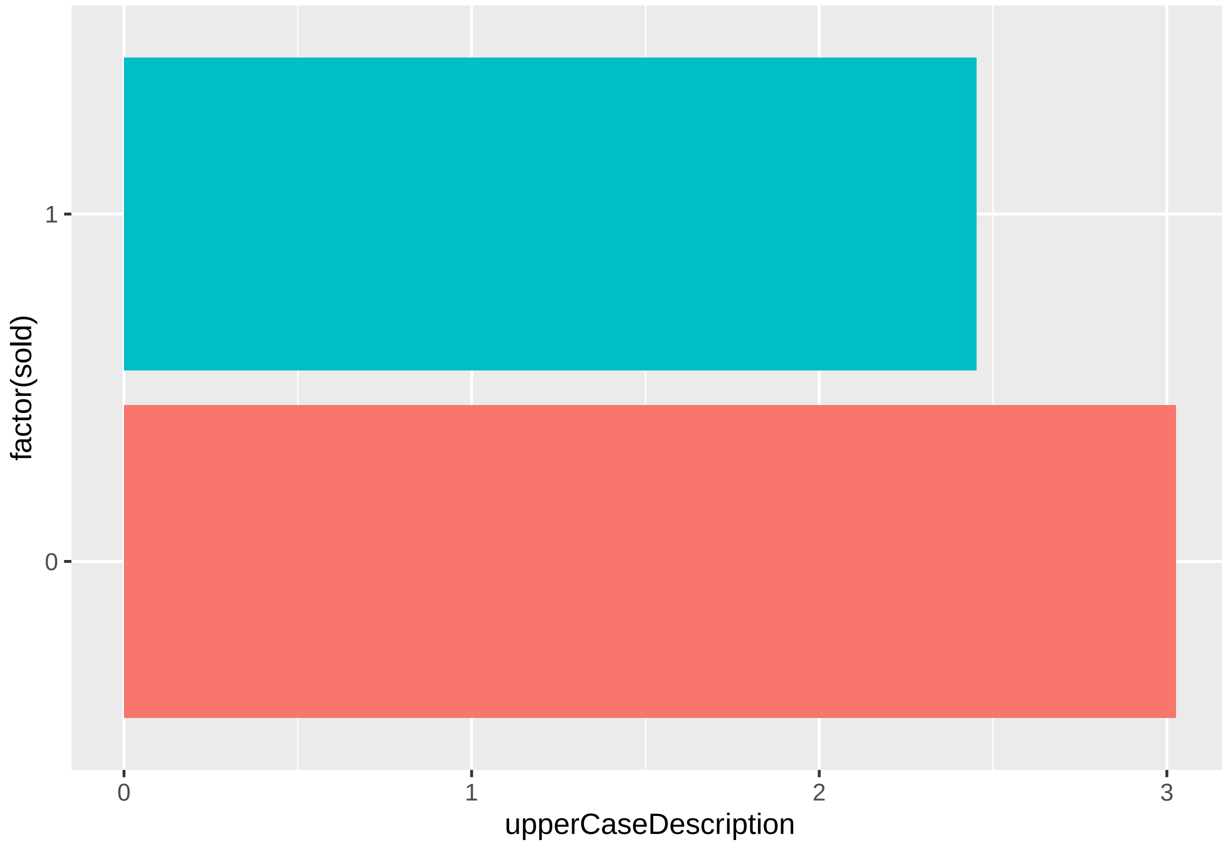


upperCaseDescription

```
tapply(train$upperCaseDescription,train$sold,mean)
```

```
##        0        1
## 3.025678 2.451827
```

```
ggplot(data=train,aes(x=factor(sold),y=upperCaseDescription,fill=factor(sold)))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```

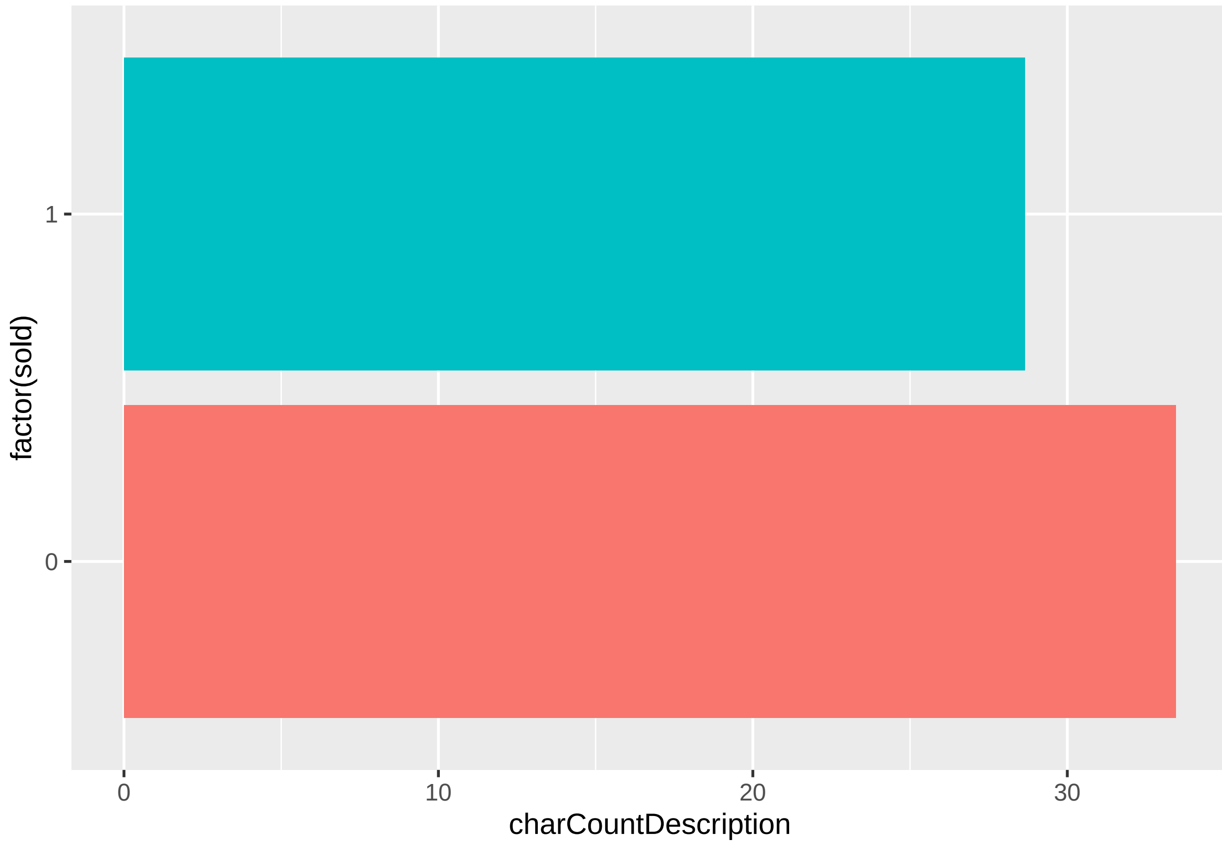


charCountDescription

```
tapply(train$charCountDescription,train$sold,mean)
```

```
##        0        1
## 33.45078 28.65116
```

```
ggplot(data=train,aes(x=factor(sold),y=charCountDescription,fill=factor(sold)))+
  geom_bar(stat='summary',fun='mean')+
  guides(fill=F)+
  coord_flip()
```



Model 1

Estimate Model

The bar chart above indicates differences in startprice between iPads that sold vs. those that did not sell. So, let us see if `startprice` has an effect on whether an iPad sells.

```
model1 = glm(sold~startprice,data=train,family='binomial')
model1
```

```
##
## Call:  glm(formula = sold ~ startprice, family = "binomial", data = train)
##
## Coefficients:
## (Intercept)   startprice
##    1.441392    -0.008202
##
## Degrees of Freedom: 1302 Total (i.e. Null);  1301 Residual
## Null Deviance:       1799
## Residual Deviance: 1426  AIC: 1430
```

Prediction

What is the probability of an iPad priced at $200 selling?

There are two ways of doing this. One could enter coefficients into the model.

```
exp(model1$coef[1] + model1$coef[2]*200)/(1+exp(model1$coef[1] + model1$coef[2]*200))
```

```
## (Intercept)
##   0.4503876
```

Or use predict function. Note we need to specify type='response' to get a probability.

```
predict(model1,newdata=data.frame(startprice=200),type='response')
```

```
##         1
## 0.4503876
```

Inference

Is the coeff of `startprice` significant?

```
summary(model1)
```

```
##
## Call:
## glm(formula = sold ~ startprice, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8077  -0.9319  -0.2894   0.8521   2.8677
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.4413920  0.1127191   12.79   <2e-16 ***
## startprice  -0.0082025  0.0005261  -15.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1798.8  on 1302  degrees of freedom
## Residual deviance: 1426.1  on 1301  degrees of freedom
## AIC: 1430.1
##
## Number of Fisher Scoring iterations: 5
```

How does one interpret the coefficient of `startprice`?

```
summary(model1)$coef[2]
```

```
## [1] -0.008202484
```

Specifically, what is the percent increase in likelihood of an iPad being sold with a $1 increase in `startprice`?

```
100*(exp(summary(model1)$coef[2])-1)
```

```
## [1] -0.8168935
```

Model 2

The bar chart of `storage` indicated differences in percentage of iPads sold for different levels of `storage`. So, let us use storage to predict whether an iPad is sold. As you examine the result below, bear in mind that `storage` is a nominal scaled variable with three levels.

```
levels(train$storage)
```

```
## [1] "128 GB"           "Less than 128 GB" "Unknown"
```

Estimate Model

```r
model2 = glm(sold~storage,data=train,family='binomial')
summary(model2)
```

```
##
## Call:
## glm(formula = sold ~ storage, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1448  -1.1448  -0.7842   1.2104   1.6304
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.0217     0.2749  -3.717 0.000202 ***
## storageLess than 128 GB   0.9445     0.2813   3.357 0.000787 ***
## storageUnknown            0.6162     0.3321   1.856 0.063520 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1798.8  on 1302  degrees of freedom
## Residual deviance: 1784.2  on 1300  degrees of freedom
## AIC: 1790.2
##
## Number of Fisher Scoring iterations: 4
```

Predict

Let us predict the probability of sale (`sold`=1) for an ipad with storage Less than 128 GB.

Method 1: Entering in coefficients

```r
exp(model2$coef[1]+model2$coef[2]*1+model2$coef[3]*0)/
  (1+exp(model2$coef[1]+model2$coef[2]*1+model2$coef[3]*0))
```

```
## (Intercept)
##   0.4807175
```

Method 2: Using predict function

```r
predict(model2,newdata=data.frame(storage='Less than 128 GB'),type='response')
```

```
##         1
## 0.4807175
```

Inference

Is the impact of storage on probability of selling an iPad statistically significant?

```r
summary(model2)
```

```
##
## Call:
## glm(formula = sold ~ storage, family = "binomial", data = train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.1448  -1.1448  -0.7842   1.2104   1.6304
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.0217     0.2749  -3.717 0.000202 ***
## storageLess than 128 GB   0.9445     0.2813   3.357 0.000787 ***
## storageUnknown            0.6162     0.3321   1.856 0.063520 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1798.8  on 1302  degrees of freedom
## Residual deviance: 1784.2  on 1300  degrees of freedom
## AIC: 1790.2
##
## Number of Fisher Scoring iterations: 4
```

What does the coefficient of storage mean? Specifically, what is the chance of selling an iPad with storage Less than 128 GB (relative to 128GB storage)?

```
summary(model2)$coef[2] # coefficient of storage "Less than 128 GB"
```

```
## [1] 0.9444829
```

How many times better is the likelihood of selling an iPad with less than 128 GB vs an iPad with 128 GB of storage?

```
exp(summary(model2)$coef[2])
```

```
## [1] 2.571483
```

Phrased differently, how much more is the likelihood of selling an iPad with less than 128 GB of storage relative to one with 128GB

```
100*(exp(summary(model2)$coef[2])-1)
```

```
## [1] 157.1483
```

Model 3

Let us estimate a model using all the predictor variables.

Estimate

```
names(train)
```

```
##  [1] "UniqueID"             "sold"                 "biddable"
##  [4] "startprice"           "condition"            "cellular"
##  [7] "carrier"              "color"                "storage"
## [10] "productline"          "noDescription"        "charCountDescription"
## [13] "upperCaseDescription" "startprice_99end"
```

```
model3 = glm(sold~biddable+startprice+condition+cellular+carrier+color+
               storage+productline+noDescription+upperCaseDescription+startprice_99end,
             data=train,
             family='binomial')
summary(model3)
```

```
##
## Call:
## glm(formula = sold ~ biddable + startprice + condition + cellular +
##     carrier + color + storage + productline + noDescription +
##     upperCaseDescription + startprice_99end, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5904  -0.6937  -0.2412   0.5880   3.5455
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      3.308473   0.625191   5.292 1.21e-07 ***
## biddablenot biddable            -1.714183   0.164660 -10.410  < 2e-16 ***
## startprice                      -0.010928   0.001067 -10.244  < 2e-16 ***
## conditionnew                     0.740711   0.347126   2.134 0.032856 *
## conditionrefurbished            -0.218706   0.374096  -0.585 0.558799
## conditionused                    0.437807   0.259256   1.689 0.091276 .
## cellularNo cellular             11.295423 535.411489   0.021 0.983169
## cellularUnknown                 -0.506228   0.506912  -0.999 0.317963
## carrierNone                    -11.459179 535.411508  -0.021 0.982925
## carrierSprint/T-Mobile           0.723014   0.510843   1.415 0.156971
## carrierUnknown                  -0.172301   0.414798  -0.415 0.677860
## carrierVerizon                   0.367958   0.370403   0.993 0.320516
## colorGold                       -0.368708   0.527783  -0.699 0.484804
## colorSpace Gray                 -0.037119   0.306962  -0.121 0.903750
## colorUnknown                    -0.239797   0.210803  -1.138 0.255312
## colorWhite                      -0.328564   0.222736  -1.475 0.140178
## storageLess than 128 GB         -1.152463   0.460386  -2.503 0.012306 *
## storageUnknown                  -0.755094   0.631170  -1.196 0.231564
## productlineiPad 2                0.202039   0.280591   0.720 0.471496
## productlineiPad 3                0.741904   0.345099   2.150 0.031569 *
## productlineiPad 4                0.757366   0.364372   2.079 0.037659 *
## productlineiPad Air 1/2          2.058510   0.395930   5.199 2.00e-07 ***
## productlineiPad mini             0.396212   0.296231   1.338 0.181057
## productlineiPad mini 2           1.425504   0.418475   3.406 0.000658 ***
## productlineiPad mini Retina      2.348397   0.953592   2.463 0.013790 *
## productlineiPad mini3            1.630888   0.530966   3.072 0.002130 **
## productlineUnknown               0.034357   0.379439   0.091 0.927853
## noDescriptionno description     -0.008146   0.170710  -0.048 0.961942
## upperCaseDescription            -0.013995   0.008532  -1.640 0.100930
## startprice_99endnot a 99 ending -0.037325   0.164092  -0.227 0.820063
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1798.8  on 1302  degrees of freedom
## Residual deviance: 1136.8  on 1273  degrees of freedom
## AIC: 1196.8
##
## Number of Fisher Scoring iterations: 12
```

Is model3 better than the models 1 and 2? Compare AIC. Lower AIC, better the model.

```r
summary(model3)$aic
```

```
## [1] 1196.761
```

```r
summary(model2)$aic
```

```
## [1] 1790.18
```

```r
summary(model1)$aic
```

```
## [1] 1430.122
```

Inference

Which coefficients are statistically significant?

```r
summary(model3)
```

```
##
## Call:
## glm(formula = sold ~ biddable + startprice + condition + cellular +
##     carrier + color + storage + productline + noDescription +
##     upperCaseDescription + startprice_99end, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5904  -0.6937  -0.2412   0.5880   3.5455
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       3.308473   0.625191   5.292 1.21e-07 ***
## biddablenot biddable             -1.714183   0.164660 -10.410  < 2e-16 ***
## startprice                       -0.010928   0.001067 -10.244  < 2e-16 ***
## conditionnew                      0.740711   0.347126   2.134 0.032856 *
## conditionrefurbished             -0.218706   0.374096  -0.585 0.558799
## conditionused                     0.437807   0.259256   1.689 0.091276 .
## cellularNo cellular              11.295423 535.411489   0.021 0.983169
## cellularUnknown                  -0.506228   0.506912  -0.999 0.317963
## carrierNone                     -11.459179 535.411508  -0.021 0.982925
## carrierSprint/T-Mobile            0.723014   0.510843   1.415 0.156971
## carrierUnknown                   -0.172301   0.414798  -0.415 0.677860
## carrierVerizon                    0.367958   0.370403   0.993 0.320516
## colorGold                        -0.368708   0.527783  -0.699 0.484804
## colorSpace Gray                  -0.037119   0.306962  -0.121 0.903750
## colorUnknown                     -0.239797   0.210803  -1.138 0.255312
## colorWhite                       -0.328564   0.222736  -1.475 0.140178
## storageLess than 128 GB          -1.152463   0.460386  -2.503 0.012306 *
## storageUnknown                   -0.755094   0.631170  -1.196 0.231564
## productlineiPad 2                 0.202039   0.280591   0.720 0.471496
## productlineiPad 3                 0.741904   0.345099   2.150 0.031569 *
## productlineiPad 4                 0.757366   0.364372   2.079 0.037659 *
## productlineiPad Air 1/2           2.058510   0.395930   5.199 2.00e-07 ***
## productlineiPad mini              0.396212   0.296231   1.338 0.181057
## productlineiPad mini 2            1.425504   0.418475   3.406 0.000658 ***
## productlineiPad mini Retina       2.348397   0.953592   2.463 0.013790 *
```

```
## productlineiPad mini3            1.630888   0.530966   3.072 0.002130 **
## productlineUnknown               0.034357   0.379439   0.091 0.927853
## noDescriptionno description     -0.008146   0.170710  -0.048 0.961942
## upperCaseDescription            -0.013995   0.008532  -1.640 0.100930
## startprice_99endnot a 99 ending -0.037325   0.164092  -0.227 0.820063
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1798.8  on 1302  degrees of freedom
## Residual deviance: 1136.8  on 1273  degrees of freedom
## AIC: 1196.8
##
## Number of Fisher Scoring iterations: 12
```

Predict

```
pred = predict(model3,type='response')
```

Now, let us examine the quality of predictions by comparing them to the true values. Here are the first ten observations.

```
data.frame(sold = train$sold[1:10],
           predicted_probability = pred[1:10])
```

```
##    sold predicted_probability
## 1     0             0.3005919
## 3     1             0.2594178
## 6     1             0.8833245
## 8     0             0.2153054
## 9     1             0.8883491
## 10    1             0.7638966
## 11    1             0.3290937
## 13    1             0.8537269
## 15    1             0.1493956
## 16    0             0.3154026
```

To convert prediction probabilities to a binary outcome, one can use a cut off or threshold value. Lets use a cutoff of 0.5.

```
data.frame(sold = train$sold[1:10],
           predicted_probability = pred[1:10],
           prediction_binary = as.integer(pred[1:10]>0.5))
```

```
##    sold predicted_probability prediction_binary
## 1     0             0.3005919                 0
## 3     1             0.2594178                 0
## 6     1             0.8833245                 1
## 8     0             0.2153054                 0
## 9     1             0.8883491                 1
## 10    1             0.7638966                 1
## 11    1             0.3290937                 0
## 13    1             0.8537269                 1
## 15    1             0.1493956                 0
## 16    0             0.3154026                 0
```

Strength of Model

Log-likelihood based measures that are commonly used to summarize strength of a model. These include

- -2 Log Likelihood (or Residual Deviance): Measure of error in the model. Lower is better.
- Model Chi-square: Difference of error between baseline and new model; higher and statistically significant is better.
- Pseudo R2: Designed to mimic R2 from linear regression. There are a few types including: McFadden R2, Cox and Snell R2, and Nagleklerke R2.
- AIC: Measure of relative quality of model. Lower is better. Can only be used to compare two models. Not meaningful in an absolute sense.

-2 Log Likelihood (or Residual Deviance): Measure of error in the model. Lower is better.

```
model3$deviance  # or -2*logLik(model3)
```

```
## [1] 1136.761
```

Model Chi-square: Difference of error between baseline (null model) and new model; higher and statistically significant is better.

```
model_chi_square  = model3$null.deviance - model3$deviance
df = model3$df.null - model3$df.residual
p_val = pchisq(model3$null.deviance - model3$deviance,
       df=model3$df.null - model3$df.residual,
       lower.tail=FALSE)
paste('Chi-square value of',model_chi_square , 'with',df , 'df', 'corresponding to a p-value of', p_val
```

```
## [1] "Chi-square value of 662.051788294497 with 29 df corresponding to a p-value of 8.12783513635733e
```

Pseudo R2: Designed to mimic R2 from linear regression. There are a few types including: McFadden R2, Cox and Snell R2, and Nagleklerke R2.

```
dev = model3$deviance
nulldev = model3$null.deviance
n = nrow(train)
mcfadden_r2 = 1 - dev/nulldev
cox_and_snell_r2 = 1 - exp(-(nulldev - dev)/n)
nagelklerke_r2 = cox_and_snell_r2/(1 - exp(-nulldev/n))
mcfadden_r2; cox_and_snell_r2; nagelklerke_r2
```

```
## [1] 0.3680494
```

```
## [1] 0.3983612
```

```
## [1] 0.5321763
```

AIC: Measure of relative quality of model. Lower is better. Can only be used to compare two models. Not meaningful in an absolute sense.

```
model3$aic
```

```
## [1] 1196.761
```

Another alternative is to use the `LogRegR2` from `library(descr)`

```
library(descr)
```

```
## Warning: package 'descr' was built under R version 4.0.4
```

```
LogRegR2(model3)
```

```
## Chi2                  662.0518
## Df                    29
## Sig.                  0
## Cox and Snell Index   0.3983612
## Nagelkerke Index      0.5321763
## McFadden's R2         0.3680494
```

Accuracy

Once the probabilities are converted to a binary outcome, they can easily be compared to the true values. Let us summarize the predictions in a classification table.

```
ct = table(sold = train$sold,
           predictions = as.numeric(pred>0.5))
ct
```

```
##     predictions
## sold   0   1
##    0 595 106
##    1 148 454
```

Overall quality of predictions can be computed as the proportion of correct predictions, known as accuracy. Cost of false negatives can be accounted for by the Specificity and cost of false negatives by the Sensitivity.

```
accuracy = sum(ct[1,1],ct[2,2])/nrow(train); accuracy
```

```
## [1] 0.8050652
```

```
specificity = ct[1,1]/sum(ct[1,1],ct[1,2]); specificity
```

```
## [1] 0.8487874
```

```
sensitivity = ct[2,2]/sum(ct[2,1],ct[2,2]); sensitivity
```

```
## [1] 0.7541528
```

While higher accuracy is desirable, what is a good value really depends on the data. It may be tempting to use a coin flip as a threshold for predicting a binary outcome but in most cases this may be setting a very low bar. A better baseline is to compare it to majority class. This is the proportion of correct predictions if one predicts the outcome to be the same as the more common category. In our specific example, the majority class threshold would be the proportion of ipads that are not sold.

```
prop.table(table(train$sold))
```

```
##
##         0         1
## 0.5379893 0.4620107
```

```
# or max(sum(train$sold==0), sum(train$sold==1))/nrow(train)
```

Predict on test

Model Performance on train sample is bound to be be inflated, so we are going to evaluate prediction quality or model performance on the test sample. But, before that, let us establish a baseline for accuracy. Since in our train sample, most ipads were not sold, the majority class is sold==0.

```
sum(test$sold==0)/nrow(test)
```

```
## [1] 0.5376344
```

Accuracy

Now, let us compute prediction quality on the test sample using model3

```
pred = predict(model3,newdata=test,type='response')
ct = table(sold = test$sold,
           predictions = as.integer(pred>0.5)); ct
```

```
##     predictions
## sold   0   1
##    0 240  60
##    1  77 181
```

```
accuracy = sum(ct[1,1],ct[2,2])/nrow(test); accuracy
```

```
## [1] 0.7544803
```

```
specificity = ct[1,1]/sum(ct[1,1],ct[1,2]); specificity
```

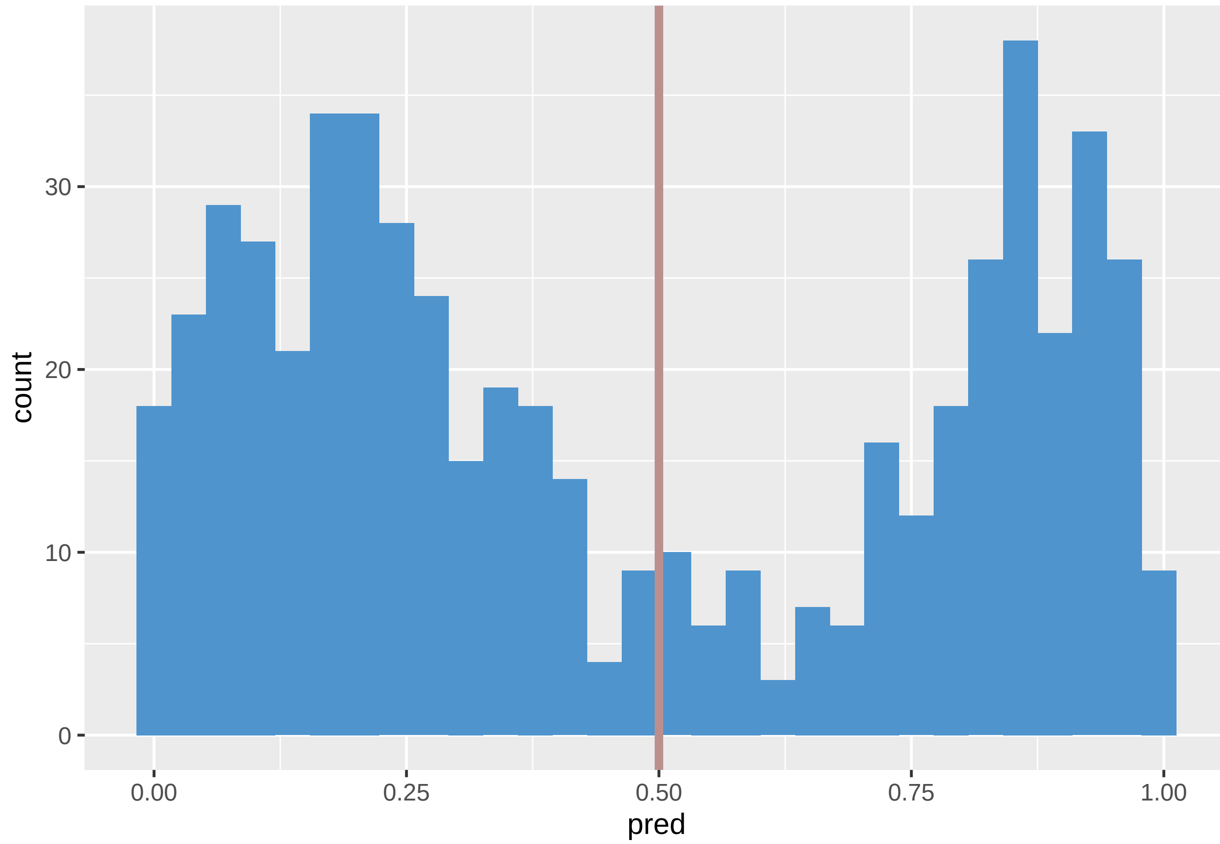
```
## [1] 0.8
```

```
sensitivity = ct[2,2]/sum(ct[2,1],ct[2,2]); sensitivity
```

```
## [1] 0.7015504
```

Cutoff Value The accuracy (also known as hit-ratio) generated above is dependent on the cutoff applied to the prediction probabilities. The histogram below shows a bimodal distribution of predicted probabilities and the cufoff used above. So far, we have used a cutoff of 0.5, however this threshold is is arbitrary. Using a different cutoff is likely to generate a different value of accuracy.

```
ggplot(data=data.frame(pred),aes(x=pred))+
  geom_histogram(fill='steelblue3')+
  geom_vline(xintercept =0.5, size=1.5,color='rosybrown')
```



Let us use a different cutoff (and also a slighly different albeit equivalent way of computing accuracy). You can experiment with other cutoffs as well.

```
accuracy_0.5 = sum(as.integer(pred>0.5)==test$sold)/nrow(test); accuracy_0.5
```

```
## [1] 0.7544803
```

```
accuracy_0.4 = sum(as.integer(pred>0.4)==test$sold)/nrow(test); accuracy_0.4
```

```
## [1] 0.734767
```

```
accuracy_0.3 = sum(as.integer(pred>0.3)==test$sold)/nrow(test); accuracy_0.3
```

```
## [1] 0.6953405
```

```
accuracy_0.6 = sum(as.integer(pred>0.6)==test$sold)/nrow(test); accuracy_0.6
```

```
## [1] 0.7724014
```

```
accuracy_0.7 = sum(as.integer(pred>0.7)==test$sold)/nrow(test); accuracy_0.7
```

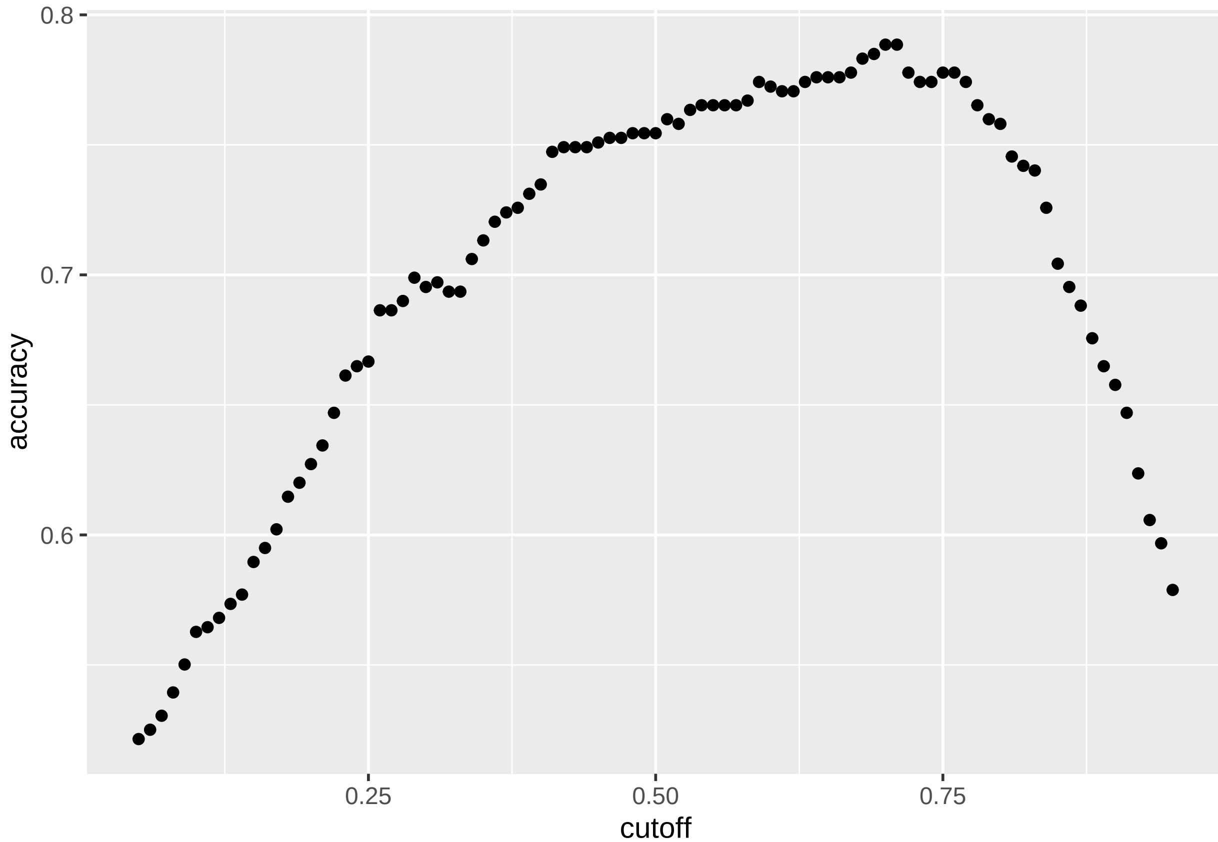
```
## [1] 0.7885305
```

Here is a plot of accuracy values varying by cutoff. While accuracy and its sister metrics specificity and sensitivity are simple and intuitive, their dependence on cutoff value calls for a metric that is independent of the cufoff value.

```
acc =
  sapply(X = seq(0.050,0.95,0.01),
       FUN = function(x) sum((as.integer(pred>x) == test$sold))/nrow(test))

ggplot(data=data.frame(cutoff=seq(0.05,0.95,0.01),
```

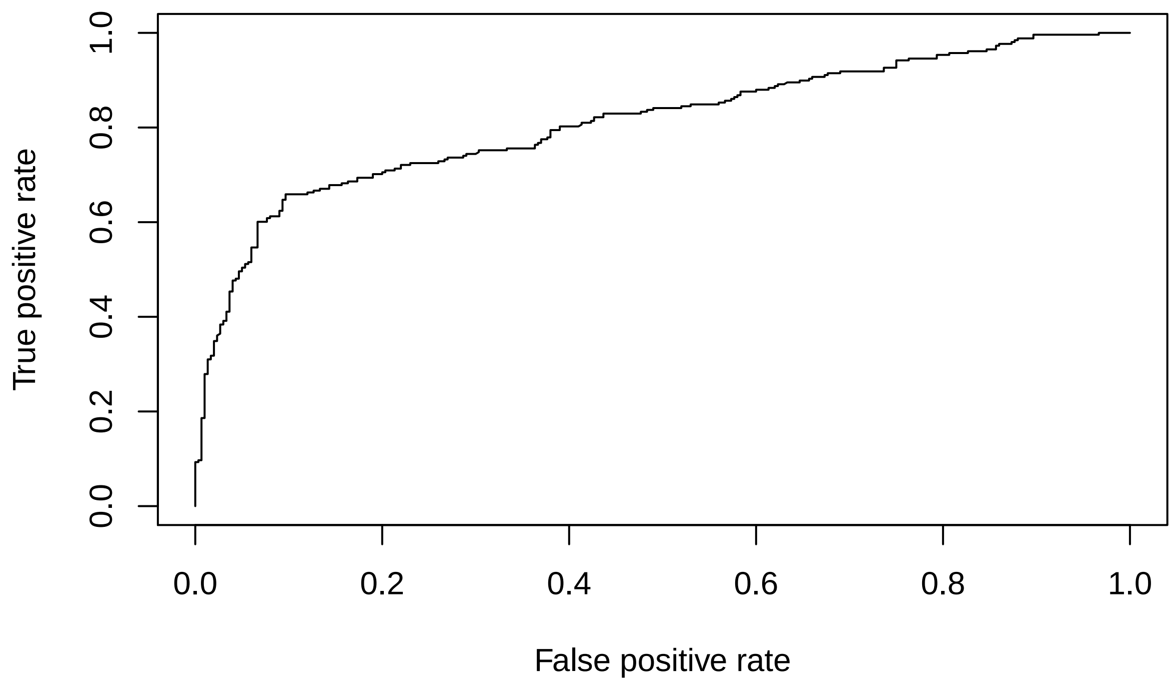
```
                       accuracy=acc),
      aes(x=cutoff,y=accuracy))+
  geom_point()
```



ROC and Area Under the Curve

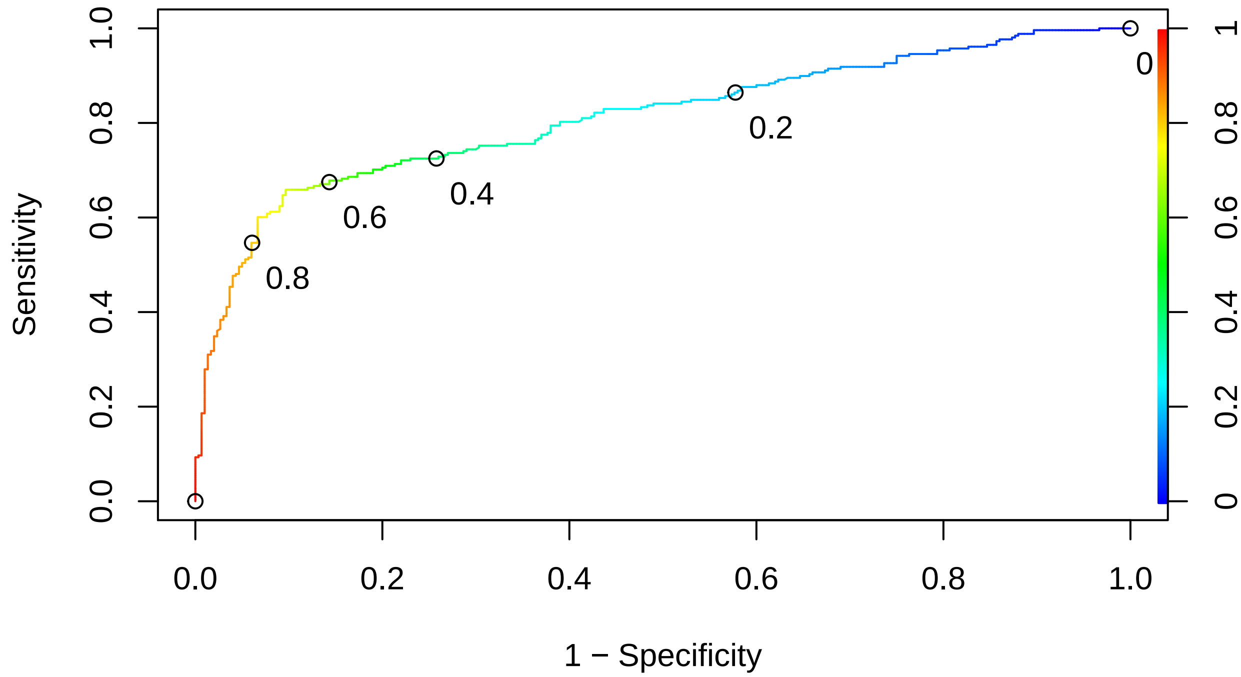
ROC curves allow us to visualize the impact of different thresholds on Specificity and Sensitivity. AUC is a model performance measure that is independent of any particular cutoff or threshold.

```
library(ROCR)
ROCRpred = prediction(pred,test$sold)
ROCRperf = performance(ROCRpred,"tpr","fpr")
plot(ROCRperf)
```



Color coded and annotated ROC curve

```
plot(ROCRperf,colorize=TRUE,print.cutoffs.at=seq(0,1,0.2),text.adj=c(-0.3,2),
     xlab="1 - Specificity",ylab="Sensitivity")
```



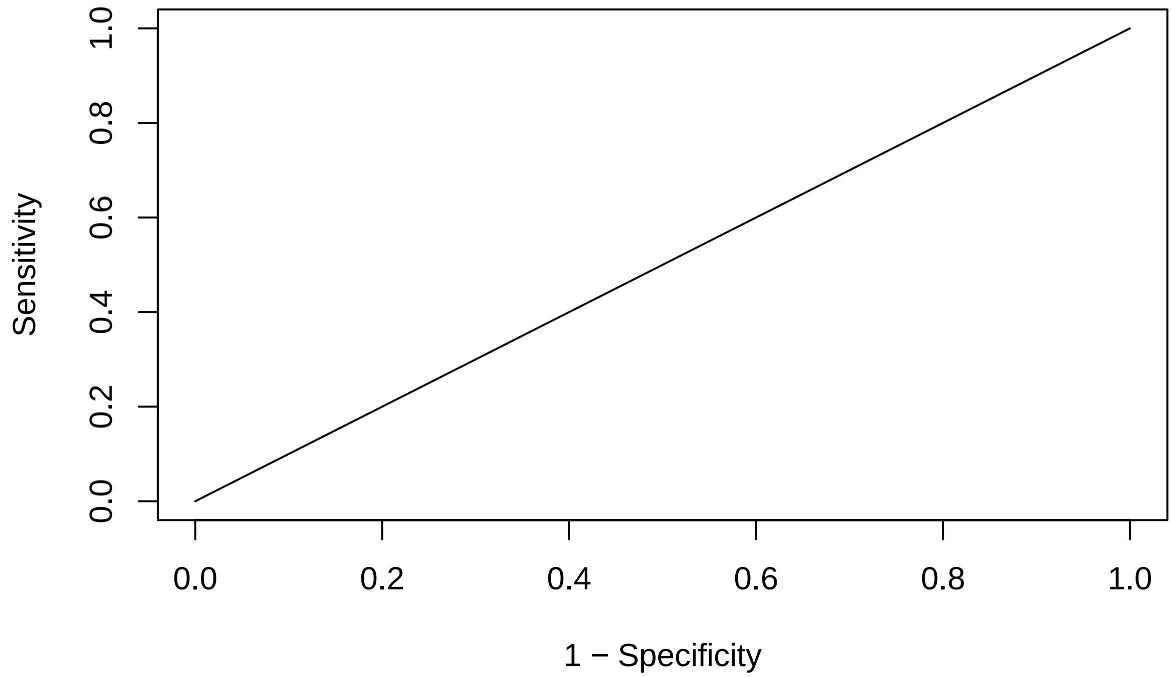
Area Under the Curve

```
as.numeric(performance(ROCRpred,"auc")@y.values) # auc measure
```

```
## [1] 0.8111111
```

As a reference, here is what the ROC for a baseline model will look like

```
baselinePred = pred*0
ROCRpred = prediction(baselinePred,test$sold)
ROCRperf = performance(ROCRpred,"tpr","fpr")
plot(ROCRperf,xlab="1 - Specificity",ylab="Sensitivity") # relabeled axes
```



Baseline AUC

```
as.numeric(performance(ROCRpred,"auc")@y.values)
```

```
## [1] 0.5
```