Eric_Hirsch_622_Assignment_1

Predicting Sales Data

Eric Hirsch

10/7/2022

Contents

```
knitr::opts_chunk$set()
```

Predicting item type in sales data records

In this analysis, we review two sets of VBA-generated sales data records, one large (5,000) and one small (100). The data includes categorical and numerical data - the numerical data are largely collinear with each other as they represent all the components of profit - unit sales, unit price, total revenue, total cost and so on. Categorical predictors include Region, Country, Item Type, Sales channel, and Order Priority. There are, additionally, order and shipping dates. There are no missing values.

Because this is not "real" data, there are some unusual characteristics to the data shape. Perhaps the most unusual feature of the data is a one-to-one match between unit cost and item type. There are some other unusual findings as well - for example, we might expect the distribution of the lag time between order and shipment to be skewed right (most lags would be small and very large lags would rare), but instead it is uniform. Order ID, normally a throwaway variable, is moderately negatively correlated with revenue. This means we will need to rely heavily on the data and not make too many inferences based on common sense or business knowledge.

In order to address the multicollinearity in the dataset, we remove a number of predictors - we eliminate country and retain region, we eliminate shipping date but create a new variable for lag between order and shipping, and we eliminate most numerical variables except total profit and one other variable (because the same profit may be generated from high volume or low volume.) We retain units sold, because it has little relationship to unit cost (r=-.02) and moderate (relative to the other variables!) correlation with total profit (.58). We also remove order ID, despite its predictive power in this dataset, to make our analysis more meaningful.

Having wrangled the dataset, we explore opportunities to use machine learning to make predictions about the data. There are a number of labeled classes that would work here. Analysis suggests that item type would lend itself well to prediction, which makes sense since unit price is correlated with total profit and is a perfect match with item type. However, we choose first to examine region, which appears to be a more challenging and interesting exercise. It turns out that in practice, region shows little correlation with the other variables and prediction (not included below) does not perform better than random.

We choose, therefore, to predict item type. Machine learning practices do not exist in isolation from business use cases, so we construct one in order to guide our analysis. We imagine an intern has inadvertently eliminated item cost and item type from a dataset of sales records and we need to reconstruct them for a presentation to our parent company that afternoon. We have only this morning to reconstruct the data (which limits the amount of data we can process), but because it is only a general presentation we don't need to be fully accurate (80% accuracy will do). For the purpose of the presentation we don't need to be able to make inferences about item type, but we should know enough about how the predictions are being made in order to be comfortable that the process was done properly.

We now consider machine learning algorithms to correct the problem.

A number of factors weigh in on our decision of which models to choose. We know that:

- we have multiple classes to predict
- the classes are relatively balanced
- total profit, a key predictor, is not normally distributed (but could be transformed if need be)
- many of our predictors are categorical, but they may be overshadowed by total profit
- we likely have enough records in the larger dataset to avoid the "curse of dimensionality" but possibly not the smaller
- the larger dataset is probably near, but not at, the limit of what our resources can comfortably handle - if our predictions are inadequate we can increase the size of the dataset.

Given these conditions, we choose one parametric (Multinomial Regression) and one non-parametric method (Random Forest) in order to compare them. Both algorithms generally perform well under the circumstances above. Because the parametric method relies on statistical assumptions, it will be as good as those assumptions are accurate. Our analysis shows that the relationship between total profit and item type, insofar as total profit relies on item cost, is linear with a log transformation, so we take a log transformation of total profit for this analysis.

Our parametric method (Multinomial Regression) is likely be simpler, faster and require less data than our nonparametric one. However, it may not create as good a fit with the data. Our nonparametric method (Random Forest) requires more data and will be slower, but will likely create a better fit. This may lead to more accuracy, or conversely the method may overfit the data. MR additionally has the advantages that it may be more interpretable and, because we get probabilities instead of firm classes, it may be more flexible in terms of how we apply it.

We use 10-fold cross validation and apply both methods to the training data. Without doing any custom tuning, both perform well, but random forest significantly outperforms multinomial regression, with a mean 88% vs 48% accuracy respectively on the training set. A log transformation of total profit improves the multinomial regression to a mean of 53%. We then use RF to predict our classes and we achieve 87% accuracy, which exceeds our business case benchmark.

An analysis of influential factors and a look at an example of one tree, however, shows something important about our exercise. Total profit is almost exclusively the only factor random forest uses in determining item type, suggesting that the exercise is perhaps somewhat trivial. This was the risk we took when we retained total profit despite knowing about the relationship between profit, unit cost and item type. To create a more interesting analysis, it might have been better to leave profit out - a quick examination of one tree without total profit shows the influence of order lag, units sold, region and order date, a finding that is corroborated by earlier analysis that showed the correlation between item type and these predictors.

Following this exercise, we examine the smaller dataset and make some comparisons with the larger. Since the 5,000 database is generated by the same VBA algorithm s this one, we would expect similarities.

Multicollinearity and distributions look similar. Unit costs and item types continue to match one-to-one. The standard deviation of total profit is higher, as is the mean. Not surprisingly, this dataset behaves just as if it were a sample of the larger one, with similar characteristics except that sample means and standard deviations are further from the population compared to the larger one.

A distribution of classes shows how sparse the data has become. Some classes have 3 or less members and some are missing altogether. These “curse of dimensionalitity” issues are likely to hurt the analysis.

With less data to train on and more variation in the dataset, we expect a loss of predictive power. In fact, our accuracy on the training set degrades to .37% (was 87%) and .28% (was 53%) for RF and MR respectively. Because the data is so sparse, the classes in the training set don’t match those in the evaluation set so we can’t do predictions.

In the business use case we would be satisfied with our results on the 5,000 record database, understanding that in the vast majority of cases, item type is going to be determined almost entirely by the profit margin. We use our algorthm to predict the missing records and send it off to our data visualization department for the presentation.

1. Data Exploration

A. Summary Statistics For this exercise we will examine the 5,000 record and 100 record datasets from the assignment website.

The datasets contain fabricated sales orders generated by VBA for the purpose of practicing analysis.

We begin with the 5,000 record dataset. There are 14 columns, including 7 numeric columns, 5 character and two date. One of the predictors is an ID so we drop it. Here is a summary of the remaining 13 variables:

```
df2 <- read.csv("D:\\RStudio\\CUNY_622\\1\\Salesdata_5000.csv")
```

```
df2 <- df2 %>%
  dplyr::select(-Order.ID)

summary(df2)
```

```
##     Region            Country            Item.Type          Sales.Channel
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Order.Priority      Order.Date         Ship.Date           Units.Sold
##  Length:5000        Length:5000        Length:5000        Min.   :   2
##  Class :character   Class :character   Class :character   1st Qu.:2453
##  Mode  :character   Mode  :character   Mode  :character   Median :5123
##                                                           Mean   :5031
##                                                           3rd Qu.:7576
##                                                           Max.   :9999
##    Unit.Price       Unit.Cost      Total.Revenue       Total.Cost
##  Min.   :  9.33   Min.   :  6.92   Min.   :     65   Min.   :     48
##  1st Qu.: 81.73   1st Qu.: 35.84   1st Qu.: 257417   1st Qu.: 154748
##  Median :154.06   Median : 97.44   Median : 779409   Median : 468181
##  Mean   :265.75   Mean   :187.49   Mean   :1325738   Mean   : 933093
##  3rd Qu.:437.20   3rd Qu.:263.33   3rd Qu.:1839975   3rd Qu.:1189578
##  Max.   :668.27   Max.   :524.96   Max.   :6672676   Max.   :5248025
##   Total.Profit
##  Min.   :     16.9
##  1st Qu.:  85339.3
##  Median : 279095.2
##  Mean   : 392644.6
##  3rd Qu.: 565106.4
##  Max.   :1726007.5
```

```
str(df2)
```

```
## 'data.frame':    5000 obs. of  13 variables:
##  $ Region        : chr  "Central America and the Caribbean" "Central America and the Caribbean" "Eur
##  $ Country       : chr  "Antigua and Barbuda " "Panama" "Czech Republic" "North Korea" ...
##  $ Item.Type     : chr  "Baby Food" "Snacks" "Beverages" "Cereal" ...
##  $ Sales.Channel : chr  "Online" "Offline" "Offline" "Offline" ...
##  $ Order.Priority: chr  "M" "C" "C" "L" ...
##  $ Order.Date    : chr  "12/20/2013" "7/5/2010" "9/12/2011" "5/13/2010" ...
##  $ Ship.Date     : chr  "1/11/2014" "7/26/2010" "9/29/2011" "6/15/2010" ...
##  $ Units.Sold    : int  552 2167 4778 9016 7542 48 8258 927 8841 9817 ...
##  $ Unit.Price    : num  255.3 152.6 47.5 205.7 152.6 ...
##  $ Unit.Cost     : num  159.4 97.4 31.8 117.1 97.4 ...
##  $ Total.Revenue : num  140915 330641 226716 1854591 1150758 ...
##  $ Total.Cost    : num  88000 211152 151893 1055864 734892 ...
##  $ Total.Profit  : num  52915 119488 74823 798727 415866 ...
```

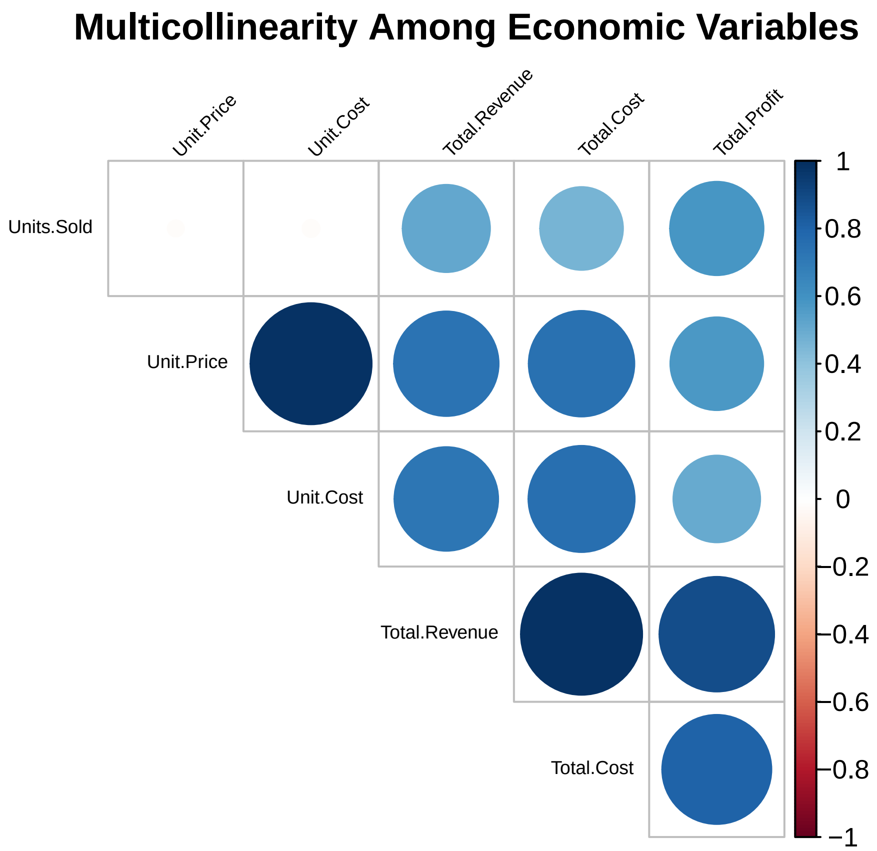
```
df2$Item.Type <- factor(df2$Item.Type)
df2$Region <- factor(df2$Region)
```

B. Multicollinearity We suspect a high degree of multicollinearity among the numeric variables, since they are components of each other - for example, total profits is made up of costs and revenues, while revenues are determined by prices and volume. We also may assume that order date and shipping date are related, and country and region will also be directly related.

The heatmap below shows the multicollinearity among the economic variables.

```
df2Num <- df2 %>%
  dplyr::select_if(is.numeric)

z <- EHExplore_Multicollinearity(df2Num, printCorrs = TRUE, title="Multicollinearity Among Economic Var
```

```
## corrplot 0.92 loaded
##                 Units.Sold  Unit.Price   Unit.Cost Total.Revenue Total.Cost
## Units.Sold      1.00000000 -0.01749167 -0.01971201     0.5118209  0.4610137
## Unit.Price     -0.01749167  1.00000000  0.98623095     0.7350631  0.7496609
## Unit.Cost      -0.01971201  0.98623095  1.00000000     0.7226761  0.7581004
## Total.Revenue   0.51182089  0.73506309  0.72267611     1.0000000  0.9878272
## Total.Cost      0.46101374  0.74966094  0.75810043     0.9878272  1.0000000
## Total.Profit    0.58641579  0.57902433  0.50593567     0.8839900  0.8005063
##                Total.Profit
## Units.Sold        0.5864158
## Unit.Price        0.5790243
## Unit.Cost         0.5059357
## Total.Revenue     0.8839900
## Total.Cost        0.8005063
## Total.Profit      1.0000000
```



There are many different strategies we can take with the issue of multicollinearity, but because certain columns completely duplicate the information of other columns, we can't ignore it. We choose, for now, to retain a minimum of variables - Total Profit (as it summarizes most of the others), and, because the same profit may come from high revenue and high costs or low revenue and low costs, we include Unit Cost as well. (Unit cost has the lowest correlation with Total Profit of all the predictors (r=.51)).

As for dates, we convert order date to an integer representing the number of days that have passed since 1/1/2000. We also create a new variable, Order.Lag, since the difference between order date and shipping date might be predictive.

Finally, we eliminate country and retain region. This leaves us a dataframe of 8 variables.

```r
df2$Order.Date <- as.Date(df2$Order.Date, format="%m/%d/%Y")
df2$Ship.Date <- as.Date(df2$Ship.Date, format="%m/%d/%Y")

df2$Order.Lag <- as.integer(df2$Ship.Date-df2$Order.Date)
df2$OrderDaysSince2000 <- as.integer(df2$Ship.Date-as.Date("2000-01-01"))

df3 <- df2 %>%
    dplyr::select(-Order.Date, -Ship.Date, -Country, -Unit.Price, -Total.Revenue, -Total.Cost, -Units.S
```

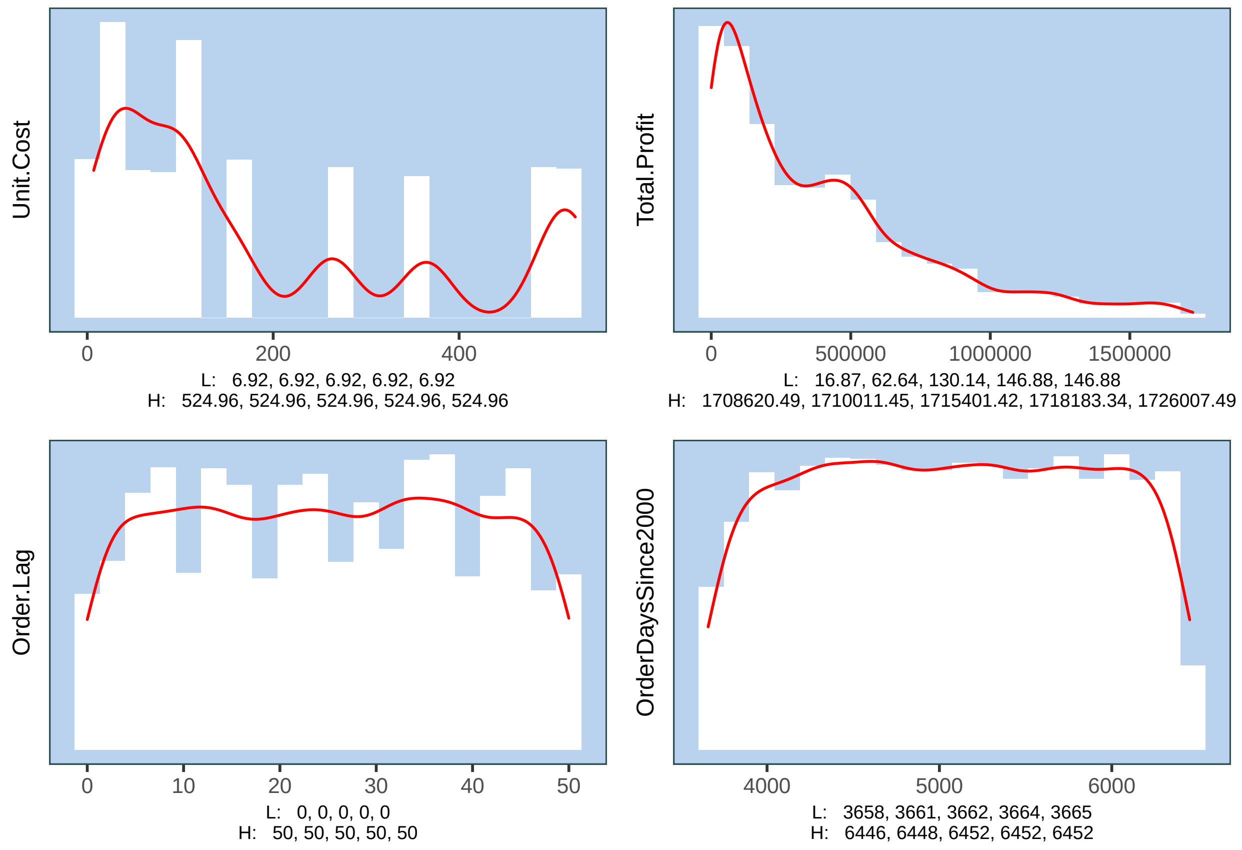
```r
summary(df3)
```

```
##                               Region                Item.Type
##  Asia                            : 719   Beverages      : 447
##  Australia and Oceania           : 416   Fruits         : 447
##  Central America and the Caribbean: 534   Baby Food      : 445
##  Europe                          :1330   Cosmetics      : 424
##  Middle East and North Africa    : 610   Household      : 424
##  North America                   : 106   Office Supplies: 420
##  Sub-Saharan Africa              :1285   (Other)        :2393
##  Sales.Channel      Order.Priority       Unit.Cost       Total.Profit
##  Length:5000        Length:5000        Min.   :  6.92   Min.   :     16.9
##  Class :character   Class :character   1st Qu.: 35.84   1st Qu.:  85339.3
##  Mode  :character   Mode  :character   Median : 97.44   Median : 279095.2
##                                        Mean   :187.49   Mean   : 392644.6
##                                        3rd Qu.:263.33   3rd Qu.: 565106.4
##                                        Max.   :524.96   Max.   :1726007.5
##
##    Order.Lag     OrderDaysSince2000
##  Min.   : 0.00   Min.   :3658
##  1st Qu.:12.00   1st Qu.:4388
##  Median :25.00   Median :5066
##  Mean   :25.05   Mean   :5066
##  3rd Qu.:38.00   3rd Qu.:5754
##  Max.   :50.00   Max.   :6452
##
```

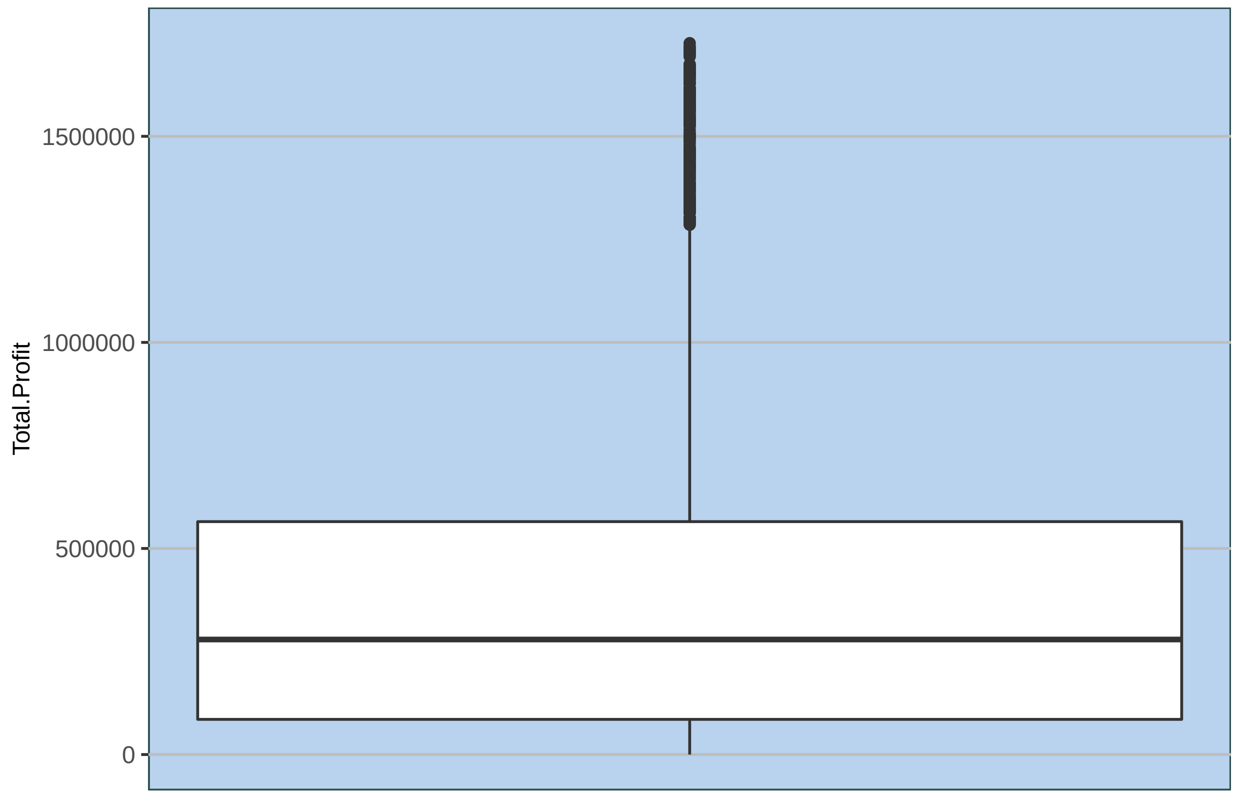
C. Distributions When we examine the distributions of the numeric variables, we find that Total profit is highly skewed, total cost is somewhat skewed, and the date variables are relatively uniform. There are many odd gaps in the cost distribution, which appears to be a series of discrete values. We may consider doing a log transformation of profit if need be. Since the data is fabricated, the uniformity of the date distributions suggests to me that these dates are just pulled randomly from a uniform distribution and won't be useful.

Not surprisingly, a boxplot shows a great number of outliers for total profits - this is consistent with the skew in the distribution.

```r
a <- EHSummarize_SingleColumn_Histograms(df3)
grid.arrange(grobs=a[c(1:4)])
```



```
a <- EHSummarize_SingleColumn_Boxplots(df3)
grid.arrange(grobs=a[2])
```



L: 16.87, 62.64, 130.14, 146.88, 146.88
H: 1708620.49, 1710011.45, 1715401.42, 1718183.34, 1726007.49

D. Relationships We can run a regression on total profit just to get an idea of some of the relationships between the numeric and categorical variables. We can see from this exploration that item types are strongly correlated with profits, as are medium priority items, but nothing else is. Unit cost could not be calculated because of singularities. We know unit cost is not fully correlated with total profit, so it must be fully correlated with another variable or in conjunction with other variables.

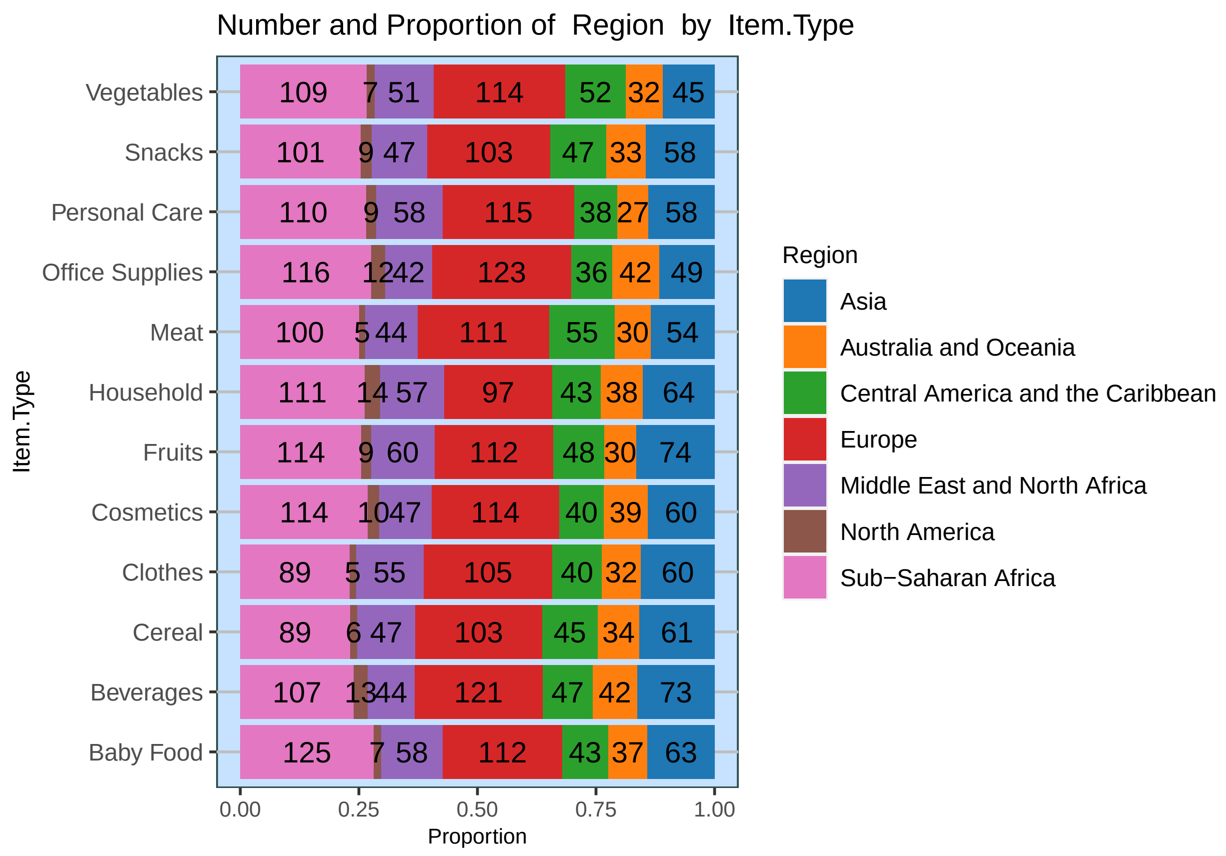
```
x <- lm(Total.Profit ~., data=df3)
summary(x)
```

```
##
## Call:
## lm(formula = Total.Profit ~ ., data = df3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -869368 -146888    1874  141660  847247
##
## Coefficients: (1 not defined because of singularities)
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          4.970e+05  3.153e+04  15.762  < 2e-16
## RegionAustralia and Oceania          3.597e+03  1.687e+04   0.213   0.8312
## RegionCentral America and the Caribbean -9.154e+02  1.565e+04  -0.058   0.9534
## RegionEurope                        -1.667e+04  1.268e+04  -1.314   0.1889
## RegionMiddle East and North Africa  -1.396e+04  1.508e+04  -0.926   0.3545
## RegionNorth America                 -3.254e+04  2.850e+04  -1.142   0.2536
## RegionSub-Saharan Africa               9.604e+03  1.277e+04   0.752   0.4519
## Item.TypeBeverages                    -4.117e+05  1.834e+04 -22.452  < 2e-16
## Item.TypeCereal                       -4.083e+04  1.907e+04  -2.141   0.0323
## Item.TypeClothes                      -1.123e+05  1.904e+04  -5.896 3.98e-09
## Item.TypeCosmetics                     3.893e+05  1.857e+04  20.958  < 2e-16
## Item.TypeFruits                       -4.763e+05  1.832e+04 -25.993  < 2e-16
## Item.TypeHousehold                     3.312e+05  1.858e+04  17.826  < 2e-16
## Item.TypeMeat                         -2.166e+05  1.887e+04 -11.475  < 2e-16
## Item.TypeOffice Supplies               1.450e+05  1.863e+04   7.784 8.49e-15
## Item.TypePersonal Care                -3.602e+05  1.868e+04 -19.283  < 2e-16
## Item.TypeSnacks                       -2.236e+05  1.888e+04 -11.846  < 2e-16
## Item.TypeVegetables                   -1.665e+05  1.874e+04  -8.888  < 2e-16
## Sales.ChannelOnline                   -4.564e+03  7.751e+03  -0.589   0.5560
## Order.PriorityH                        1.026e+04  1.107e+04   0.927   0.3542
## Order.PriorityL                       -3.084e+03  1.119e+04  -0.276   0.7828
## Order.PriorityM                        2.292e+04  1.099e+04   2.085   0.0371
## Unit.Cost                                     NA         NA      NA       NA
## Order.Lag                              1.230e+02  2.655e+02   0.463   0.6431
## OrderDaysSince2000                    -2.448e+00  4.889e+00  -0.501   0.6165
##
## (Intercept)                           ***
## RegionAustralia and Oceania
## RegionCentral America and the Caribbean
## RegionEurope
## RegionMiddle East and North Africa
## RegionNorth America
## RegionSub-Saharan Africa
## Item.TypeBeverages                    ***
## Item.TypeCereal                       *
## Item.TypeClothes                      ***
## Item.TypeCosmetics                    ***
## Item.TypeFruits                       ***
## Item.TypeHousehold                    ***
## Item.TypeMeat                         ***
## Item.TypeOffice Supplies              ***
## Item.TypePersonal Care                ***
## Item.TypeSnacks                       ***
## Item.TypeVegetables                   ***
## Sales.ChannelOnline
## Order.PriorityH
## Order.PriorityL
## Order.PriorityM                       *
## Unit.Cost
## Order.Lag
## OrderDaysSince2000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 273500 on 4976 degrees of freedom
## Multiple R-squared:  0.4922, Adjusted R-squared:  0.4899
## F-statistic: 209.7 on 23 and 4976 DF,  p-value: < 2.2e-16
```

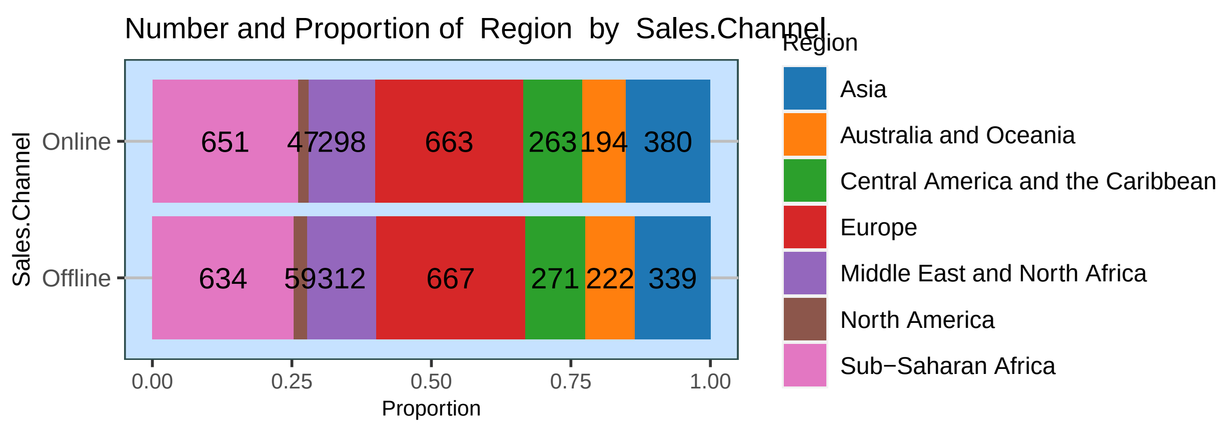
This analysis suggests that item type may be the most reasonable class to predict. However, region may also work, if it is correlated with some of the other variables besides profit. We can test this conjecture with some further analysis.

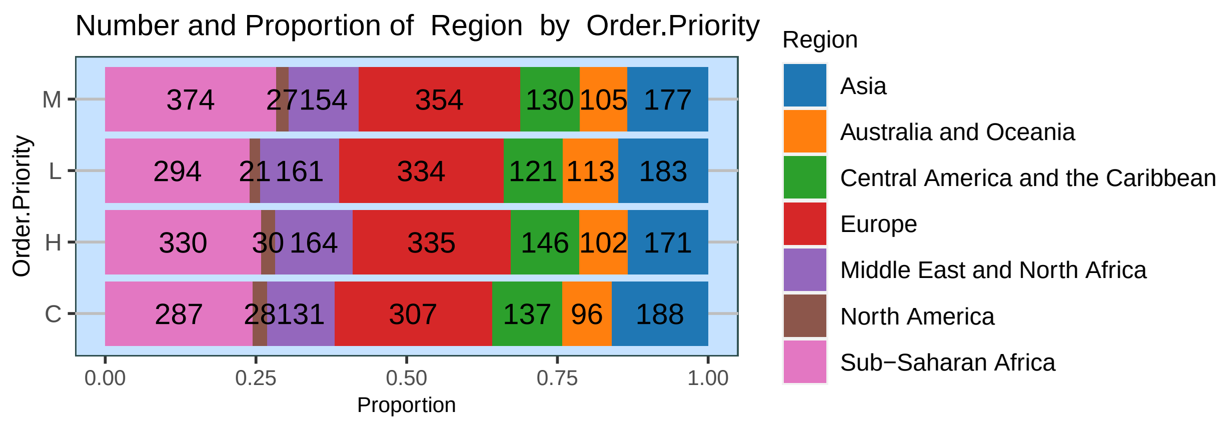
Bar charts and boxplots show relatively little relationship between region and item type, sales channel, and order priority.

```r
a <- EHExplore_TwoCategoricalColumns_Barcharts(df3, "Region")
grid.arrange(grobs=a[c(2)])
```

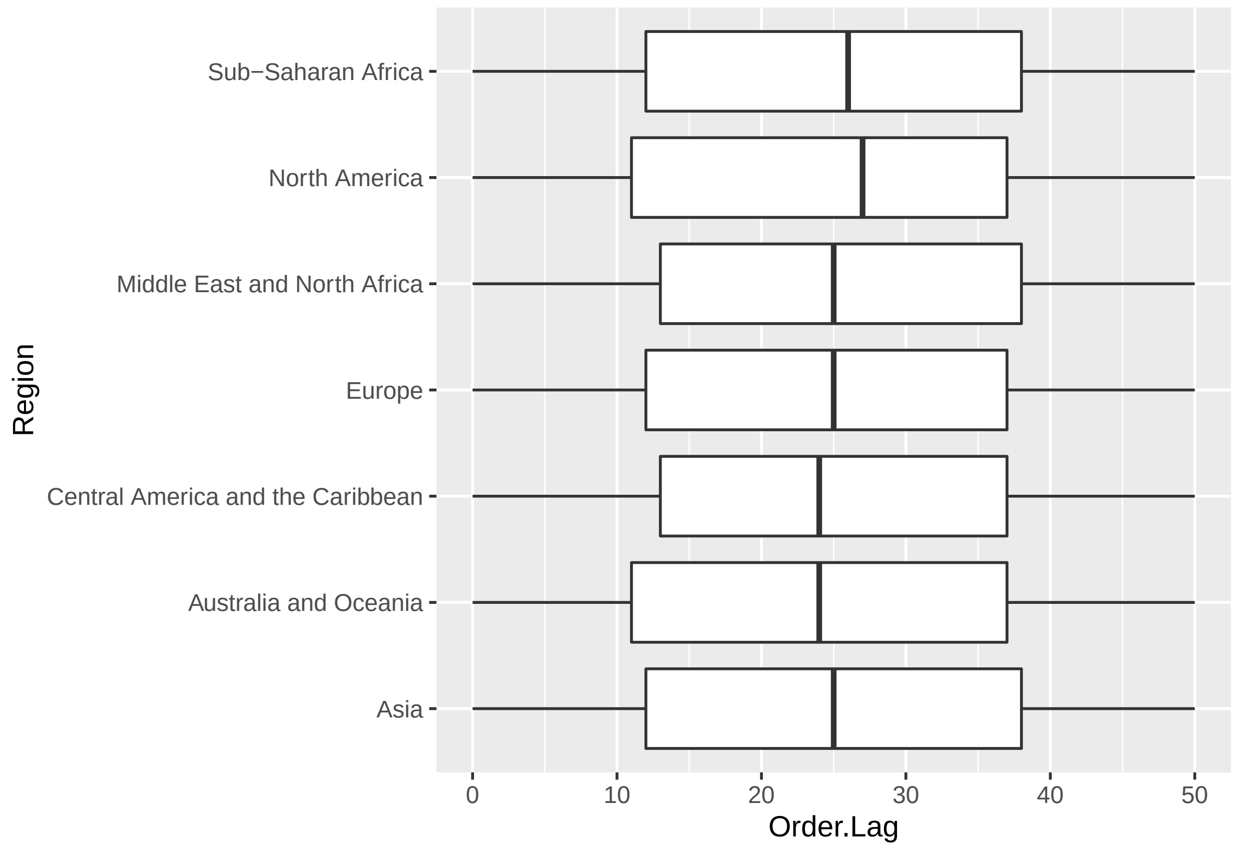


```r
grid.arrange(grobs=a[c(3:4)])
```

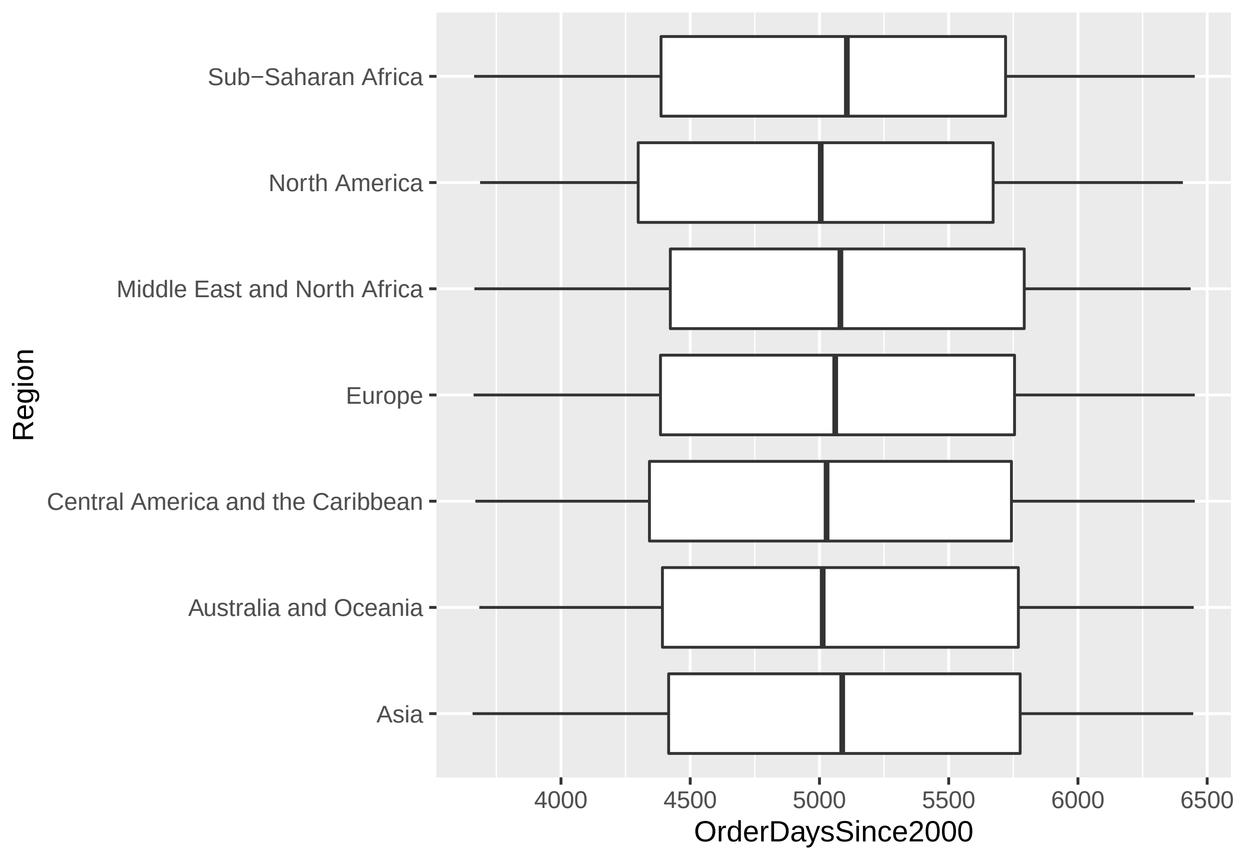




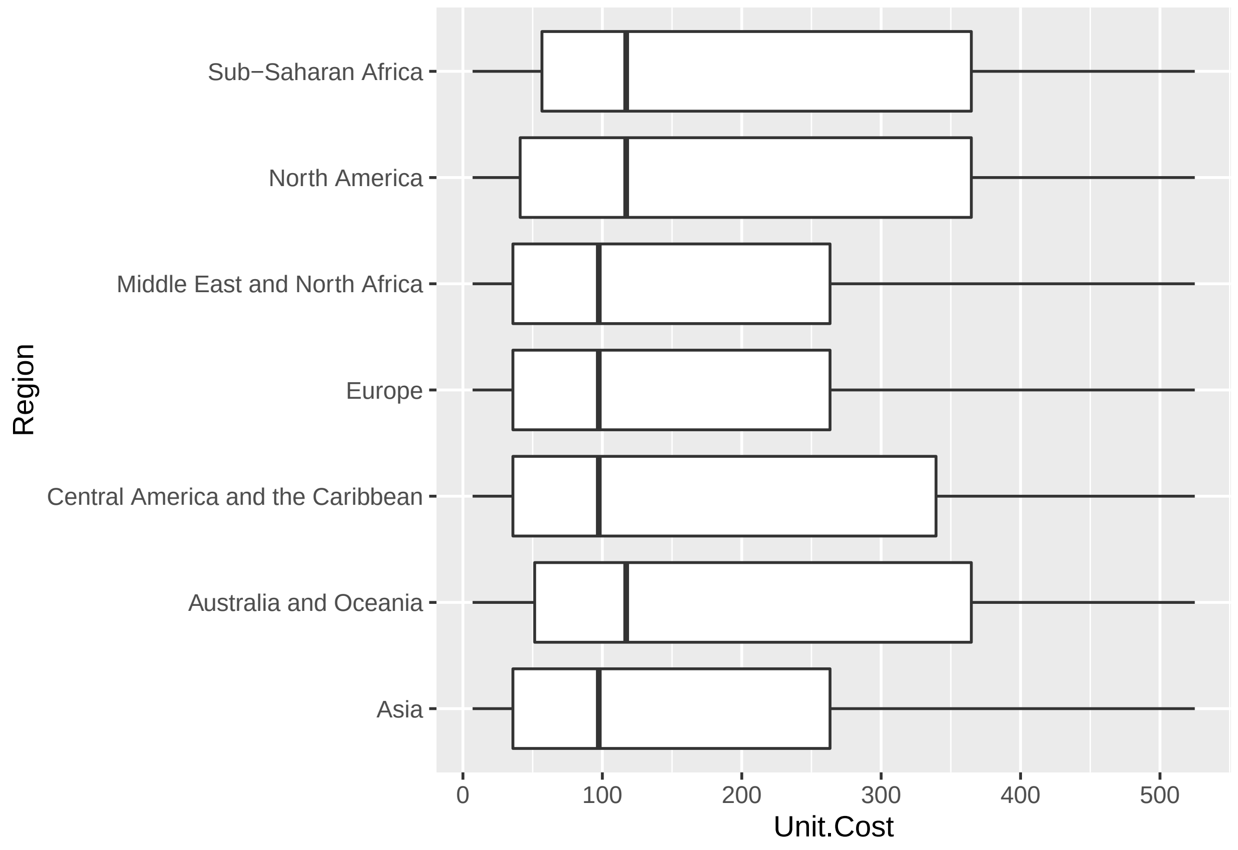
```
ggplot(df2, aes(Region, Order.Lag)) +
  geom_boxplot() +
  coord_flip()
```



```
ggplot(df2, aes(Region, OrderDaysSince2000)) +
  geom_boxplot() +
  coord_flip()
```



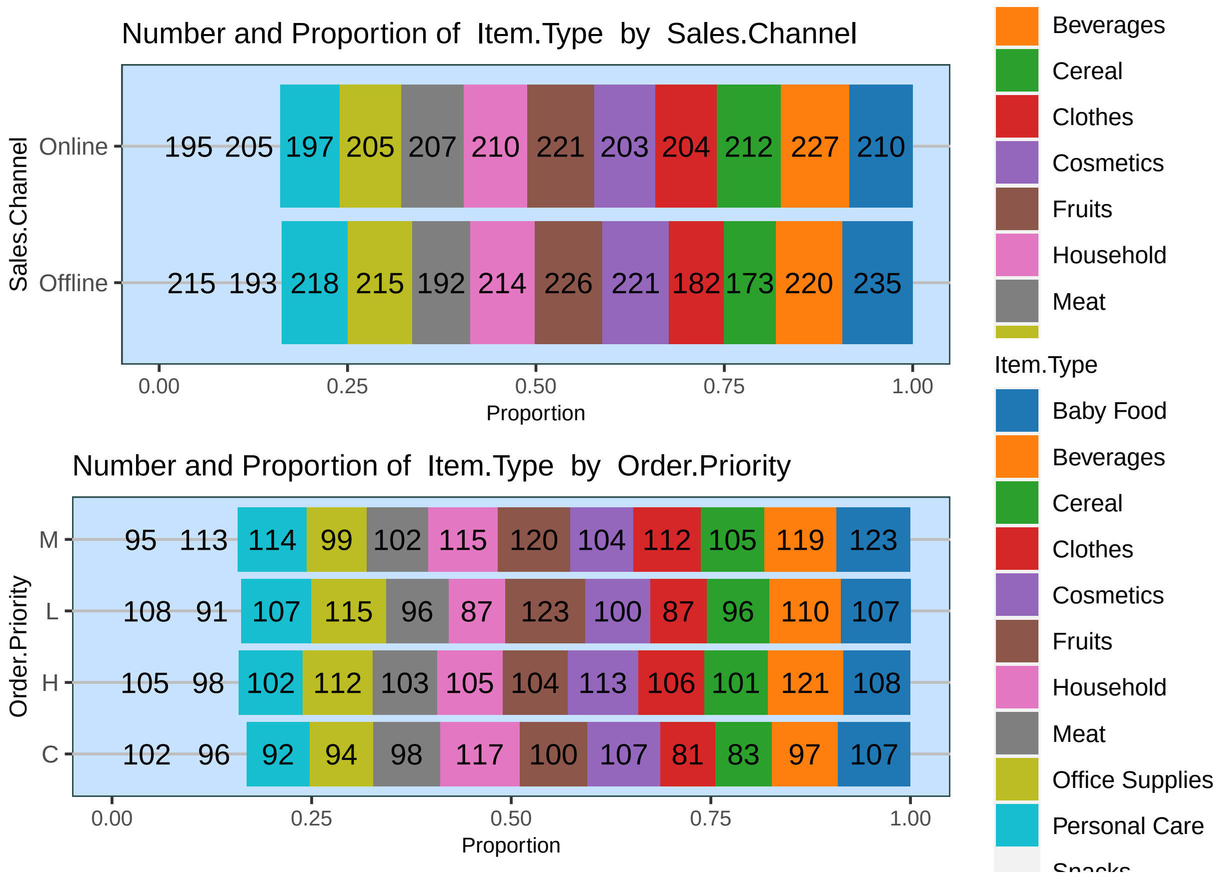
```
ggplot(df2, aes(Region, Unit.Cost)) +
  geom_boxplot() +
  coord_flip()
```



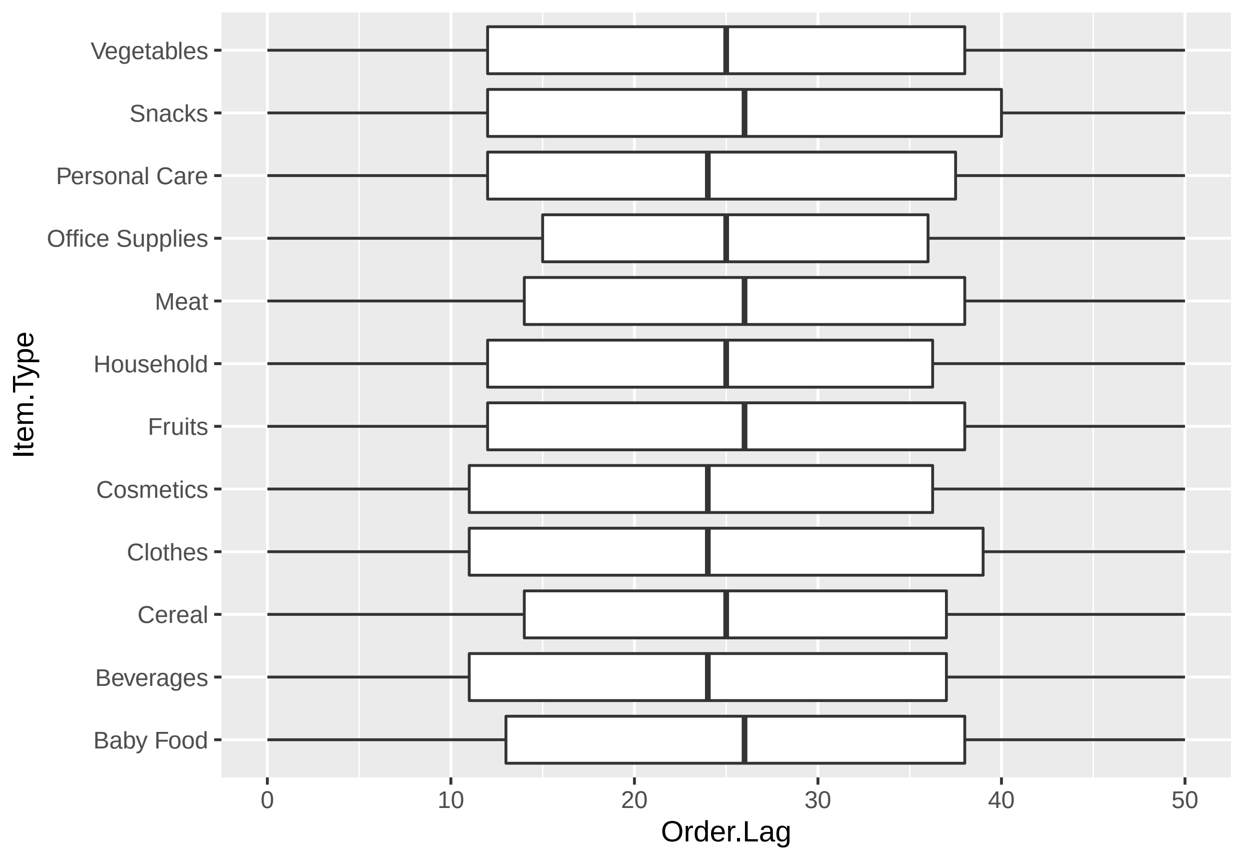
E. Choosing item type as the variable to predict We therefore choose Item Type to predict for this analysis. As with region, we can ask, "how does it correlate with the non-economic variables?" Item type shows some relationship with order lag and order date. However, the most striking relationship is with unit cost. Now we see the source of the singularity - each item type has one, and only one, unit cost and vice versa. The two are completely correlated. When we take the log of unit cost, we see a relatively linear relationship. Just to be sure, a regression shows an R2 of 1.

```
a <- EHExplore_TwoCategoricalColumns_Barcharts(df3, "Item.Type")
grid.arrange(grobs=a[c(3:4)])
```

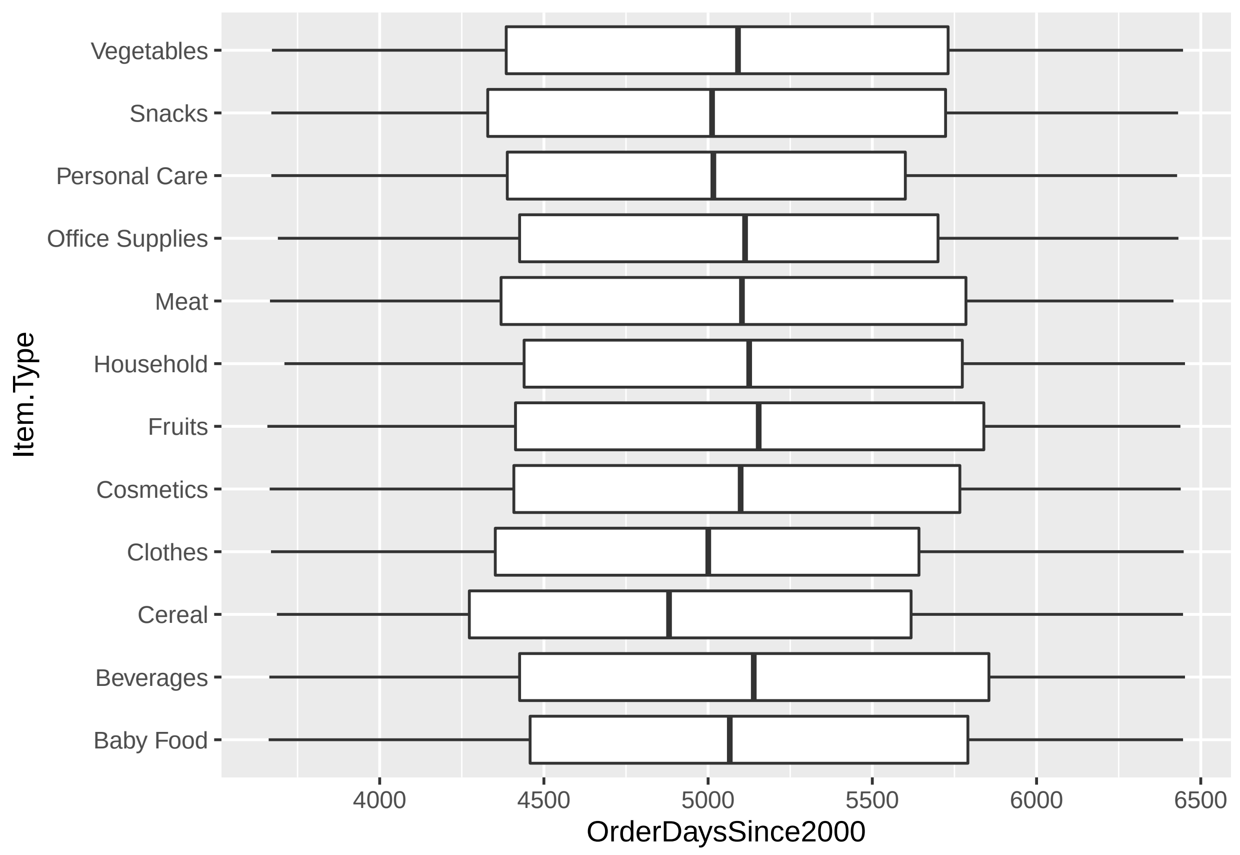
```
## Warning: This manual palette can handle a maximum of 10 values. You have
## supplied 12.

## Warning: This manual palette can handle a maximum of 10 values. You have
## supplied 12.
```



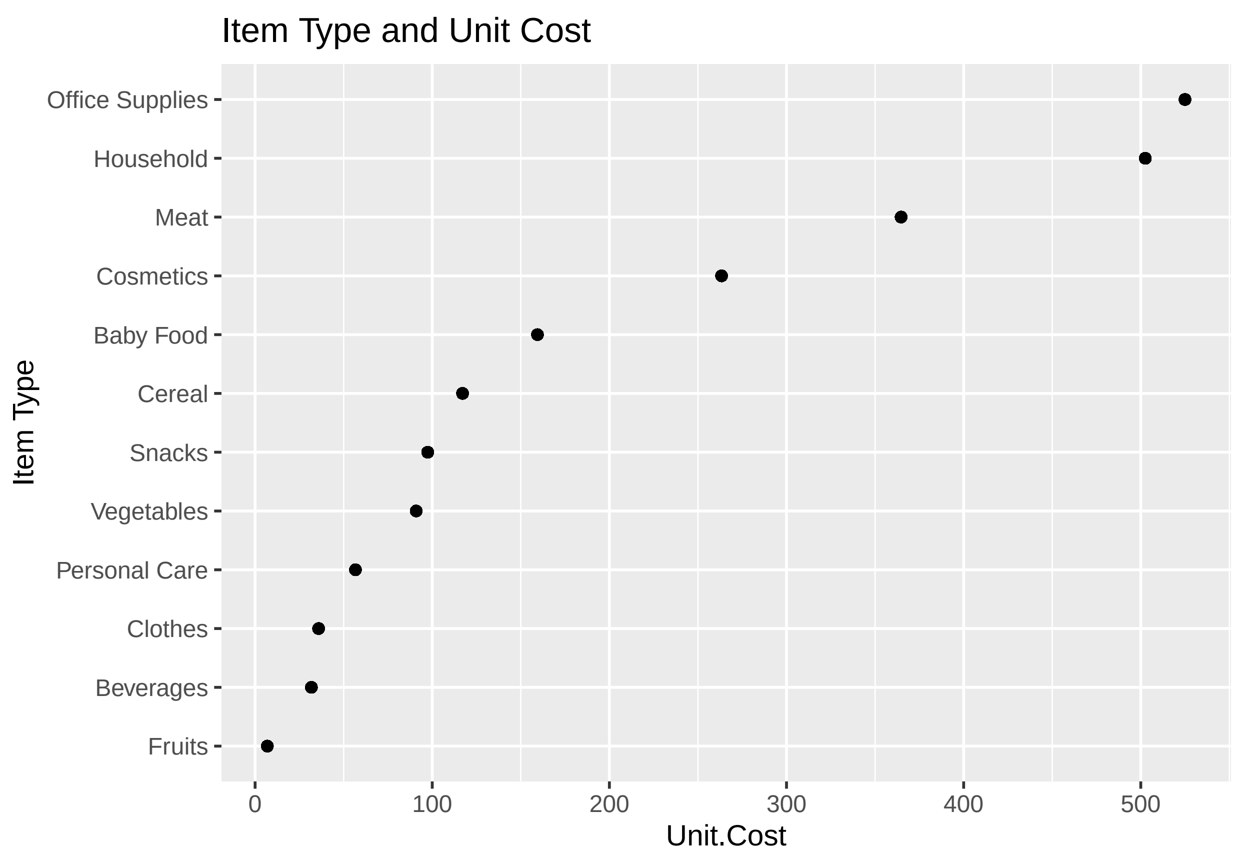
```r
df2a <- df2

ggplot(df2a, aes(Item.Type, Order.Lag)) +
  geom_boxplot() +
  coord_flip()
```



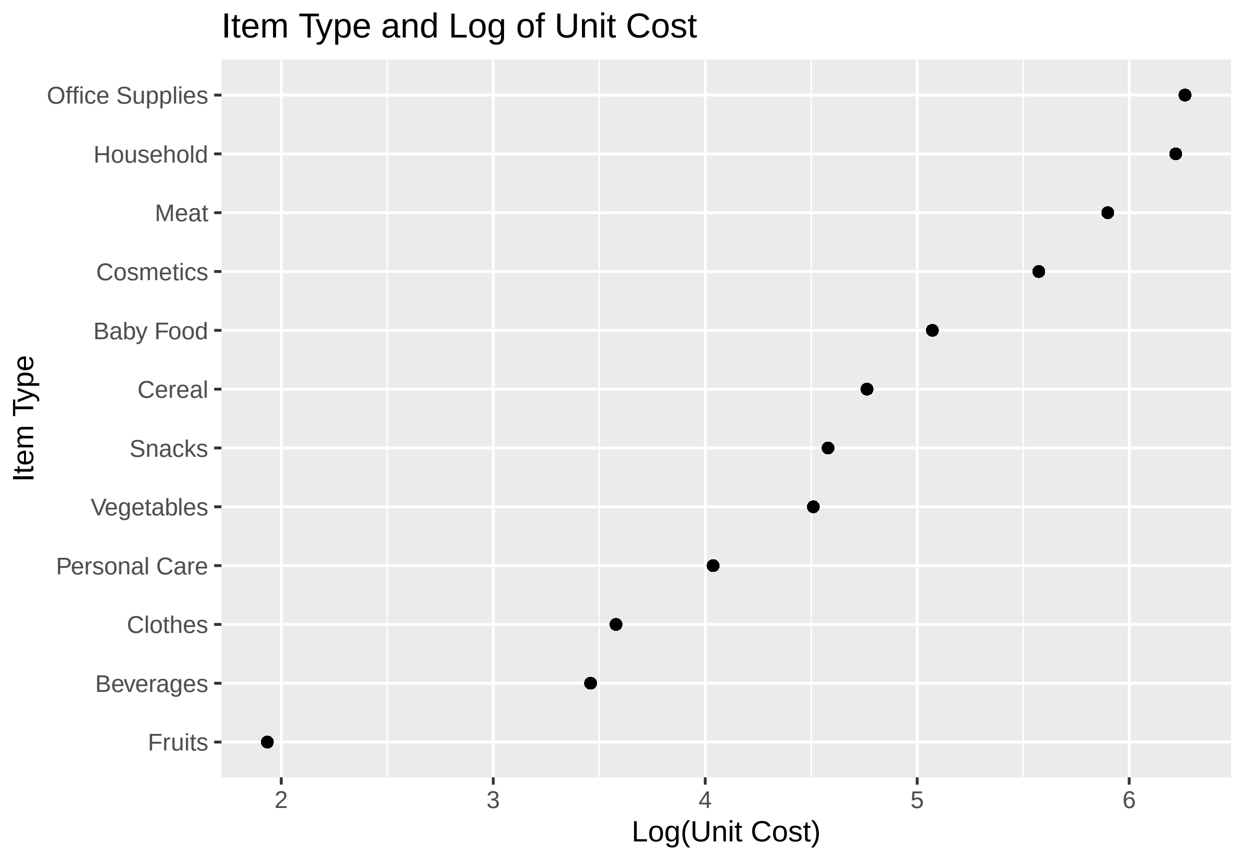
```
ggplot(df2a, aes(Item.Type, OrderDaysSince2000)) +
  geom_boxplot() +
  coord_flip()
```



```
df2a$Item.Type <- as.factor(df2a$Item.Type)
ggplot(df2a, aes(Unit.Cost, fct_reorder(Item.Type,
                         Unit.Cost))) +
  geom_point() +
  ggtitle("Item Type and Unit Cost") +
    ylab("Item Type")
```



```r
df2a$Unit.Cost = log(df2a$Unit.Cost)
df2a$Item.Type <- as.factor(df2a$Item.Type)

ggplot(df2a, aes(Unit.Cost, fct_reorder(Item.Type,
                         Unit.Cost))) +
  geom_point() +
  ylab("Item Type") +
  xlab("Log(Unit Cost)") +
  ggtitle("Item Type and Log of Unit Cost")
```



```r
dfx3 <- df3 %>%
  dplyr::select(Unit.Cost, Item.Type)

x <- lm(Unit.Cost ~Item.Type, data=dfx3)
summary(x)
```

```
##
## Call:
## lm(formula = Unit.Cost ~ Item.Type, data = dfx3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.263e-10 -1.170e-13  0.000e+00  5.100e-14  2.305e-10
##
## Coefficients:
##                           Estimate Std. Error    t value Pr(>|t|)
## (Intercept)              1.594e+02  2.919e-13  5.461e+14   <2e-16 ***
## Item.TypeBeverages      -1.276e+02  4.124e-13 -3.095e+14   <2e-16 ***
## Item.TypeCereal         -4.231e+01  4.286e-13 -9.872e+13   <2e-16 ***
## Item.TypeClothes        -1.236e+02  4.283e-13 -2.885e+14   <2e-16 ***
## Item.TypeCosmetics       1.039e+02  4.179e-13  2.486e+14   <2e-16 ***
## Item.TypeFruits         -1.525e+02  4.124e-13 -3.698e+14   <2e-16 ***
## Item.TypeHousehold       3.431e+02  4.179e-13  8.211e+14   <2e-16 ***
## Item.TypeMeat            2.053e+02  4.245e-13  4.835e+14   <2e-16 ***
## Item.TypeOffice Supplies  3.655e+02  4.189e-13  8.726e+14   <2e-16 ***
## Item.TypePersonal Care   -1.028e+02  4.202e-13 -2.445e+14   <2e-16 ***
## Item.TypeSnacks          -6.198e+01  4.248e-13 -1.459e+14   <2e-16 ***
## Item.TypeVegetables      -6.849e+01  4.215e-13 -1.625e+14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.158e-12 on 4988 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.73e+29 on 11 and 4988 DF,  p-value: < 2.2e-16
```

With unit cost in the analysis, a machine learning exploration is not justified, since a lookup table in Excel would perform just as well. We will retain total profit, and add Units.Sold, which has little correlation with Unit.Cost. We remove Unit.Cost, add Units.Sold, dummify the categorical variables and scale all the predictors.

```
z=list("Item.Type")

df3a <- df3 %>%
  dplyr::select(-Unit.Cost)

df3a$Units.Sold <- scale(df2$Units.Sold)

df3b <- EHPrepare_CreateDummies(df3a, exclude=z, dropFirst=TRUE)
```

```
## Warning: Predicate functions must be wrapped in `where()`.
##
##   # Bad
##   data %>% select(is.factor)
##
##   # Good
##   data %>% select(where(is.factor))
##
## i Please update your code.
## This message is displayed once per session.
```

```
## Warning: Predicate functions must be wrapped in `where()`.
##
##   # Bad
##   data %>% select(is.character)
##
##   # Good
##   data %>% select(where(is.character))
##
## i Please update your code.
## This message is displayed once per session.
```

```
## Warning in EHPrepare_CreateDummies(df3a, exclude = z, dropFirst = TRUE): NAs
## introduced by coercion
```

```
df4 <- EHPrepare_ScaleAllButTarget(df3b, "Item.Type")

df4$Total.Profit = log(df4$Total.Profit+10)
```

2. Models

A. Preparing the data The data needs to be partitioned into a training set and an evaluation set. We examine our classes in the training set and see that they are relatively uniform.

```
set.seed(042760)
i <- createDataPartition(df4$Item.Type, p=0.80, list=FALSE)
dfEval <- df4[-i,]
dfTrain <- df4[i,]

dfTrain %>% count(Item.Type)
```

```
##          Item.Type   n
## 1        Baby Food 356
## 2        Beverages 358
## 3           Cereal 308
## 4          Clothes 309
## 5        Cosmetics 340
## 6           Fruits 358
## 7        Household 340
## 8             Meat 320
## 9  Office Supplies 336
## 10   Personal Care 332
## 11          Snacks 319
## 12      Vegetables 328
```

B. Selecting Models A number of factors weigh in to our decision of which models to choose. We know that we have multiple classes to predict, that total profit, a key predictor, is not normally distributed, and that, given the strong match between item type and unit cost on the one hand and total profits and unit costs on the other, classes are likely to be relatively separate. Many of our predictors are categorical so we don't expect strong linear relationships. The number of categories is small compared to the number of records, so our data is not sparse.

Random Forest (RF) and multinomial regression (MR) will likely perform well under these conditions, so this is what we choose. MR has the advantages that it may be more interpretable and, because we get probabilities instead of firm classes, it is more flexible.

We have chosen one parametric (MR) and one non-parametric method (RF). The parametric method will likely be simpler, faster and require less data. However, it may not create as good a fit with the data. The nonparametric method (RF) requires more data and will be slower, but will likely create a better fit. This will lead to more accuracy, (unless the paradigm overfits the data which is more of a concern here than with MR.)

We will use 10-fold cross validation.

```
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8830.582243
## iter  20 value 7858.520702
## iter  30 value 7211.976311
## iter  40 value 6667.881240
## iter  50 value 6250.901441
## iter  60 value 5917.536165
## iter  70 value 5505.327323
## iter  80 value 5100.638803
## iter  90 value 4482.684968
## iter 100 value 3928.352336
## final  value 3928.352336
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8832.337200
## iter  20 value 8253.277613
## iter  30 value 8081.515752
## iter  40 value 8007.793851
## iter  50 value 7996.256757
## iter  60 value 7992.194899
## iter  70 value 7991.499087
## iter  80 value 7991.095994
## iter  90 value 7991.043181
## iter 100 value 7991.027791
## final  value 7991.027791
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8830.584038
## iter  20 value 7859.167383
## iter  30 value 7214.653622
## iter  40 value 6679.104757
## iter  50 value 6260.627781
## iter  60 value 5934.424664
## iter  70 value 5534.641911
## iter  80 value 5163.292381
## iter  90 value 4645.056327
## iter 100 value 4288.069875
## final  value 4288.069875
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8953.118659
## iter  10 value 8827.597622
## iter  20 value 7679.364300
## iter  30 value 7220.969436
## iter  40 value 6659.120157
## iter  50 value 6310.170006
## iter  60 value 6035.294006
## iter  70 value 5596.467944
## iter  80 value 5018.873746
## iter  90 value 4582.433246
## iter 100 value 4056.612776
## final  value 4056.612776
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8953.118659
## iter  10 value 8828.938298
## iter  20 value 8328.784466
## iter  30 value 8178.640083
## iter  40 value 8015.824948
## iter  50 value 7992.908488
## iter  60 value 7988.885779
## iter  70 value 7987.866808
## iter  80 value 7987.644340
## iter  90 value 7987.573540
## iter 100 value 7987.555420
## final  value 7987.555420
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8953.118659
## iter  10 value 8827.598988
## iter  20 value 7680.643350
## iter  30 value 7223.625225
## iter  40 value 6663.206883
## iter  50 value 6318.385678
## iter  60 value 6047.860583
## iter  70 value 5622.199845
## iter  80 value 5084.852613
## iter  90 value 4700.374249
## iter 100 value 4315.965383
## final  value 4315.965383
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8836.213994
## iter  20 value 7891.917951
## iter  30 value 7182.764425
## iter  40 value 6686.544069
## iter  50 value 6322.956066
## iter  60 value 5911.933997
## iter  70 value 5520.498071
## iter  80 value 4980.593493
## iter  90 value 4593.907559
## iter 100 value 4139.859157
## final  value 4139.859157
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8838.109668
## iter  20 value 8382.079541
## iter  30 value 8236.301268
## iter  40 value 8034.748074
## iter  50 value 7999.362456
## iter  60 value 7988.319877
## iter  70 value 7985.582196
## iter  80 value 7985.277004
## iter  90 value 7985.100779
## iter 100 value 7985.015391
## final  value 7985.015391
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8836.215947
## iter  20 value 7892.933825
## iter  30 value 7185.879464
## iter  40 value 6691.761215
## iter  50 value 6331.421337
## iter  60 value 5928.063676
## iter  70 value 5549.132990
## iter  80 value 5049.403056
## iter  90 value 4720.662464
## iter 100 value 4390.995547
## final  value 4390.995547
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8818.978222
## iter  20 value 7949.745814
## iter  30 value 7277.508739
## iter  40 value 6565.125606
## iter  50 value 6279.406804
## iter  60 value 5984.530117
## iter  70 value 5583.995106
## iter  80 value 4958.418949
## iter  90 value 4359.461076
## iter 100 value 3895.552956
## final  value 3895.552956
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8821.542957
## iter  20 value 8327.630362
## iter  30 value 8146.810414
## iter  40 value 8019.010881
## iter  50 value 8004.805967
## iter  60 value 7998.649708
## iter  70 value 7997.777697
## iter  80 value 7997.617362
## iter  90 value 7997.551339
## iter 100 value 7997.530980
## final  value 7997.530980
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8818.980855
## iter  20 value 7950.554226
## iter  30 value 7279.526960
## iter  40 value 6572.244766
## iter  50 value 6290.000038
## iter  60 value 5999.694627
## iter  70 value 5619.517043
## iter  80 value 5049.793512
## iter  90 value 4537.047184
## iter 100 value 4253.777078
## final  value 4253.777078
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8834.451457
## iter  20 value 8008.734789
## iter  30 value 7112.834806
## iter  40 value 6609.035701
## iter  50 value 6313.152042
## iter  60 value 6008.391687
## iter  70 value 5567.048715
## iter  80 value 4991.923059
## iter  90 value 4455.965401
## iter 100 value 4126.555633
## final  value 4126.555633
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8835.668444
## iter  20 value 8255.018059
## iter  30 value 8127.437648
## iter  40 value 8018.908535
## iter  50 value 7999.664286
## iter  60 value 7992.999457
## iter  70 value 7991.542863
## iter  80 value 7990.967835
## iter  90 value 7990.805669
## iter 100 value 7990.768773
## final  value 7990.768773
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8834.452696
## iter  20 value 8009.513456
## iter  30 value 7115.319480
## iter  40 value 6614.107975
## iter  50 value 6320.523092
## iter  60 value 6021.854227
## iter  70 value 5595.226882
## iter  80 value 5073.444598
## iter  90 value 4628.862266
## iter 100 value 4397.845448
## final  value 4397.845448
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8817.985895
## iter  20 value 8120.586388
## iter  30 value 7325.192959
## iter  40 value 6656.735712
## iter  50 value 6356.012236
## iter  60 value 5942.302725
## iter  70 value 5540.486072
## iter  80 value 5126.376024
## iter  90 value 4640.510004
## iter 100 value 4200.028930
## final  value 4200.028930
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8820.642029
## iter  20 value 8370.281508
## iter  30 value 8167.349948
## iter  40 value 8033.886351
## iter  50 value 8002.409043
## iter  60 value 7994.135109
## iter  70 value 7992.283073
## iter  80 value 7991.940892
## iter  90 value 7991.860606
## iter 100 value 7991.841603
## final  value 7991.841603
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8955.603566
## iter  10 value 8817.988624
## iter  20 value 8121.261373
## iter  30 value 7327.483454
## iter  40 value 6659.357084
## iter  50 value 6361.986251
## iter  60 value 5956.741729
## iter  70 value 5566.884494
## iter  80 value 5168.424011
## iter  90 value 4735.424648
## iter 100 value 4419.537983
## final  value 4419.537983
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8809.886201
## iter  20 value 8002.056559
## iter  30 value 7276.189550
## iter  40 value 6580.280862
## iter  50 value 6263.665229
## iter  60 value 5889.408671
## iter  70 value 5491.805583
## iter  80 value 5021.753691
## iter  90 value 4451.651909
## iter 100 value 3961.207739
## final  value 3961.207739
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8812.565733
## iter  20 value 8276.014517
## iter  30 value 8136.724833
## iter  40 value 7994.529458
## iter  50 value 7982.668674
## iter  60 value 7980.371435
## iter  70 value 7979.297279
## iter  80 value 7979.113634
## iter  90 value 7979.005741
## iter 100 value 7978.982626
## final  value 7978.982626
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8809.888959
## iter  20 value 8002.949994
## iter  30 value 7278.930081
## iter  40 value 6585.876672
## iter  50 value 6272.147071
## iter  60 value 5905.702993
## iter  70 value 5521.498482
## iter  80 value 5098.307915
## iter  90 value 4615.763155
## iter 100 value 4293.068447
## final  value 4293.068447
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8834.539149
## iter  20 value 7728.765207
## iter  30 value 7098.273636
## iter  40 value 6520.441298
## iter  50 value 6251.133152
## iter  60 value 5777.958672
## iter  70 value 5317.060671
## iter  80 value 4754.638704
## iter  90 value 4263.354568
## iter 100 value 3777.122796
## final  value 3777.122796
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8836.073198
## iter  20 value 8212.122996
## iter  30 value 8059.635864
## iter  40 value 7995.445916
## iter  50 value 7986.973167
## iter  60 value 7984.214624
## iter  70 value 7983.634723
## iter  80 value 7983.479835
## iter  90 value 7983.451561
## iter 100 value 7983.429684
## final  value 7983.429684
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8834.540726
## iter  20 value 7729.569736
## iter  30 value 7100.643529
## iter  40 value 6532.174677
## iter  50 value 6264.816171
## iter  60 value 5806.172042
## iter  70 value 5367.718503
## iter  80 value 4874.300072
## iter  90 value 4472.319429
## iter 100 value 4188.176852
## final  value 4188.176852
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8844.479705
## iter  20 value 7855.630038
## iter  30 value 7255.058031
## iter  40 value 6583.613398
## iter  50 value 6234.075649
## iter  60 value 5839.501517
## iter  70 value 5272.792863
## iter  80 value 4730.861843
## iter  90 value 4275.819279
## iter 100 value 3827.743816
## final  value 3827.743816
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8846.096014
## iter  20 value 8222.753368
## iter  30 value 8078.875807
## iter  40 value 8007.139604
## iter  50 value 8000.096120
## iter  60 value 7998.242082
## iter  70 value 7997.537047
## iter  80 value 7997.216851
## iter  90 value 7997.143185
## iter 100 value 7997.116595
## final  value 7997.116595
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8958.088472
## iter  10 value 8844.481358
## iter  20 value 7856.194622
## iter  30 value 7257.255283
## iter  40 value 6588.780406
## iter  50 value 6241.841179
## iter  60 value 5858.580032
## iter  70 value 5325.267215
## iter  80 value 4836.023043
## iter  90 value 4486.563393
## iter 100 value 4192.435779
## final  value 4192.435779
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8810.731291
## iter  20 value 7858.445397
## iter  30 value 7247.384143
## iter  40 value 6771.662934
## iter  50 value 6351.388169
## iter  60 value 5836.634645
## iter  70 value 5457.491481
## iter  80 value 4969.337440
## iter  90 value 4437.693447
## iter 100 value 4047.836490
## final  value 4047.836490
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8813.410660
## iter  20 value 8375.201825
## iter  30 value 8167.673552
## iter  40 value 8010.774916
## iter  50 value 7984.392572
## iter  60 value 7980.235628
## iter  70 value 7979.024669
## iter  80 value 7978.653843
## iter  90 value 7978.589301
## iter 100 value 7978.557892
## final  value 7978.557892
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 8950.633753
## iter  10 value 8810.734049
## iter  20 value 7859.376812
## iter  30 value 7248.459810
## iter  40 value 6775.800329
## iter  50 value 6354.158283
## iter  60 value 5854.239886
## iter  70 value 5486.017849
## iter  80 value 5013.563181
## iter  90 value 4592.459528
## iter 100 value 4346.558330
## final  value 4346.558330
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 9949.566226
## iter  10 value 9827.670015
## iter  20 value 8784.312830
## iter  30 value 8298.600061
## iter  40 value 7553.998166
## iter  50 value 7199.745567
## iter  60 value 6643.298322
## iter  70 value 6078.935115
## iter  80 value 5611.889588
## iter  90 value 5212.965236
## iter 100 value 4820.835826
## final  value 4820.835826
## stopped after 100 iterations
```

```
summary(multinom)
```

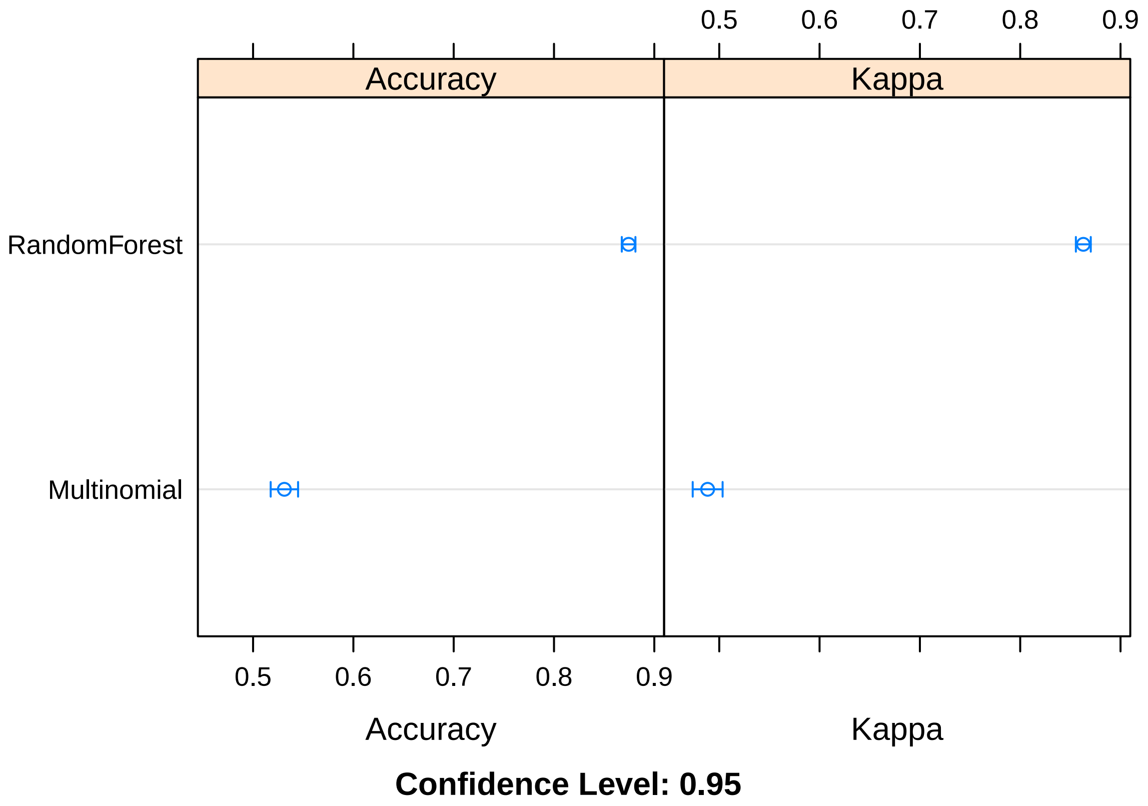
```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay)
##
## Coefficients:
```

```
##                 (Intercept) Total.Profit    Order.Lag OrderDaysSince2000
## Beverages         439.04278  -194.316569  0.045385776         0.05611772
## Cereal            -95.43075    41.288175  0.205698625        -0.07112285
## Clothes            10.62261    -4.687122  0.087571996         0.11123320
## Cosmetics        -559.03791   237.329623  0.142619644         0.05492555
## Fruits            563.95447  -250.792215  0.055894065         0.09162799
## Household        -509.43061   216.679454  0.220650923         0.16064108
## Meat              126.86088   -55.542189  0.052301461         0.21384992
## Office Supplies  -307.69117   132.059634  0.007645634         0.04141197
## Personal Care     373.70024  -164.952458 -0.027466403        -0.05911101
## Snacks            147.54252   -64.525533  0.313455023         0.17882436
## Vegetables         86.80710   -37.911326  0.143060686         0.20226919
##                  Units.Sold  Region_Asia Region_Australia.and.Oceania
## Beverages         7.2057549 -0.002083524                   0.16895616
## Cereal           -2.7765694 -0.134972794                  -0.01983091
## Clothes           0.3946595 -0.174260683                  -0.11575868
## Cosmetics       -21.7568793 -0.046184813                   0.04766841
## Fruits            7.7632701  0.107335532                   0.07934822
## Household       -19.3769464 -0.254992976                  -0.01119828
## Meat              3.0044055 -0.156226466                  -0.09345022
## Office Supplies -10.2122637  0.168579142                   0.28859038
## Personal Care     6.7893917 -0.164297199                  -0.02568257
## Snacks            3.3512057 -0.194154321                  -0.19203017
## Vegetables        2.1773113 -0.026636558                   0.03386872
##                 Region_Europe Region_Middle.East.and.North.Africa
## Beverages        -0.070346978                       -0.3209184495
## Cereal           -0.099729156                       -0.1646561727
## Clothes          -0.210377340                       -0.1492396395
## Cosmetics        -0.056108780                       -0.2344523215
## Fruits           -0.079683511                       -0.3162926430
## Household        -0.424087209                       -0.3367524829
## Meat             -0.237016598                       -0.2660716504
## Office Supplies   0.177489610                        0.0603001268
## Personal Care    -0.300673839                       -0.5311914200
## Snacks           -0.291638606                       -0.2596763793
## Vegetables        0.006913408                       -0.0007620266
##                 Region_North.America Region_Sub.Saharan.Africa
## Beverages                 0.34687141              -0.134205898
## Cereal                    0.32548768              -0.281733152
## Clothes                  -0.02691562              -0.262617593
## Cosmetics                 0.20736548              -0.400753444
## Fruits                    0.39995082               0.002205768
## Household                 0.31118186              -0.674801734
## Meat                      0.14476499              -0.261241396
## Office Supplies           0.31920579               0.005232878
## Personal Care             0.26094978              -0.283132823
## Snacks                    0.22952883              -0.271120901
## Vegetables                0.17949379               0.011355336
##                 Sales.Channel_Offline Order.Priority_C Order.Priority_H
## Beverages                -0.044926676      -0.44844563     -0.240770482
## Cereal                   -0.129032729      -0.14094378     -0.048298214
## Clothes                  -0.053488774      -0.13306744      0.003313253
## Cosmetics                -0.211911278       0.05472092      0.155458444
## Fruits                    0.022383035      -0.16410475     -0.099142784
## Household               -0.264027404      -0.12862744     -0.159459373
## Meat                    -0.031933696      -0.22154810     -0.114148316
## Office Supplies         -0.222580711      -0.09199075      0.142998382
## Personal Care            0.016115183      -0.43057584     -0.339344248
## Snacks                  -0.036454804      -0.13602167     -0.061565146
## Vegetables               0.009514435       0.11091290      0.075985614
##                 Order.Priority_L
## Beverages            -0.18144160
## Cereal                0.05124219
## Clothes              -0.10701756
## Cosmetics             0.12037053
## Fruits               -0.01817004
## Household            -0.07932789
## Meat                 -0.14848234
## Office Supplies       0.11100354
## Personal Care        -0.28853369
## Snacks               -0.07196923
## Vegetables            0.07299867
##
## Std. Errors:
##                 (Intercept) Total.Profit  Order.Lag OrderDaysSince2000
## Beverages         18.639295     8.277296 0.10971674         0.10856748
## Cereal            10.517494     4.551018 0.08014899         0.07982214
## Clothes            8.146863     3.542183 0.07851097         0.07815690
## Cosmetics         23.735147    10.049502 0.13759240         0.13502819
## Fruits            20.730469     9.242931 0.11609744         0.11424366
## Household         21.767122     9.250042 0.13482500         0.13229954
## Meat              11.168743     4.885819 0.08463918         0.08442803
## Office Supplies   16.849696     7.229808 0.11051790         0.10889279
## Personal Care     16.309547     7.209403 0.10786317         0.10710537
## Snacks            11.274267     4.934375 0.08127722         0.08041399
## Vegetables         9.247945     4.036301 0.07933551         0.07926737
##                 Units.Sold Region_Asia Region_Australia.and.Oceania
## Beverages        0.3242442   0.1554534                    0.1394574
## Cereal           0.3175534   0.1125849                    0.1000169
## Clothes          0.2288535   0.1101226                    0.1009469
## Cosmetics        0.9893434   0.1968793                    0.1708571
## Fruits           0.3277616   0.1638052                    0.1537977
## Household        0.8688215   0.1878236                    0.1613218
## Meat             0.2668403   0.1176386                    0.1072255
## Office Supplies  0.5834145   0.1700048                    0.1439215
## Personal Care    0.3194152   0.1478393                    0.1343929
## Snacks           0.2652555   0.1121699                    0.1064297
## Vegetables       0.2353545   0.1195535                    0.1061074
##                 Region_Europe Region_Middle.East.and.North.Africa
## Beverages           0.1780934                           0.1542737
## Cereal              0.1259299                           0.1069599
## Clothes             0.1227664                           0.1022516
## Cosmetics           0.2218263                           0.1868900
## Fruits              0.1908452                           0.1642713
## Household           0.2107702                           0.1768663
## Meat                0.1311320                           0.1138900
## Office Supplies     0.1941304                           0.1601021
## Personal Care       0.1680476                           0.1509593
## Snacks              0.1253832                           0.1067961
## Vegetables          0.1338975                           0.1106020
##                  Region_North.America Region_Sub.Saharan.Africa
## Beverages                   0.1749894                 0.1780004
## Cereal                      0.1585435                 0.1272136
## Clothes                     0.2081455                 0.1225604
## Cosmetics                   0.1982560                 0.2255708
## Fruits                      0.1769898                 0.1889921
## Household                   0.1878088                 0.2143650
## Meat                        0.1728882                 0.1315016
## Office Supplies             0.1832310                 0.1940548
## Personal Care               0.1734646                 0.1671751
## Snacks                      0.1625735                 0.1245821
## Vegetables                  0.1744887                 0.1330550
##                  Sales.Channel_Offline Order.Priority_C Order.Priority_H
## Beverages                   0.10834034       0.13423301       0.12957238
## Cereal                      0.07934422       0.09535701       0.09615128
## Clothes                     0.07827156       0.09312144       0.09234310
## Cosmetics                   0.13460229       0.16547453       0.16558292
## Fruits                      0.11464564       0.14107508       0.13953517
## Household                   0.13150732       0.15944908       0.16113495
## Meat                        0.08474714       0.10079311       0.10084248
## Office Supplies             0.10866008       0.13589873       0.13098760
## Personal Care               0.10678169       0.13006503       0.12777835
## Snacks                      0.08070344       0.09612928       0.09758593
## Vegetables                  0.07951624       0.09465618       0.09929849
##                  Order.Priority_L
## Beverages              0.12757721
## Cereal                 0.09379656
## Clothes                0.09618058
## Cosmetics              0.16213716
## Fruits                 0.13695759
## Household              0.15537317
## Meat                   0.10201658
## Office Supplies        0.13082162
## Personal Care          0.12574104
## Snacks                 0.09776340
## Vegetables             0.09928866
##
## Residual Deviance: 9641.672
## AIC: 9971.672
```

```
results <- resamples(list(Multinomial=multinom, RandomForest=rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Multinomial, RandomForest
## Number of resamples: 10
##
## Accuracy
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Multinomial  0.510 0.5221963 0.5255923 0.5312129 0.5295936 0.5664160    0
## RandomForest 0.865 0.8661287 0.8745293 0.8743731 0.8773321 0.8952618    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Multinomial  0.4652005 0.4787375 0.4824912 0.4884267 0.4866343 0.5268072    0
## RandomForest 0.8527005 0.8539404 0.8630886 0.8629269 0.8661483 0.8857139    0
```

```
dotplot(results)
```



Random Forest performs quite well. Mean accuracy is 88% at mtry = 14.

C. Making Predicions Now we test our random forest model on the evaluation set. We see that certain classes (beverages, fruits and personal care) are predicted very well, while others (meat, snacks) perform less well. An analysis of why is beyond the scope of this exercise.

```
print(rf)
```

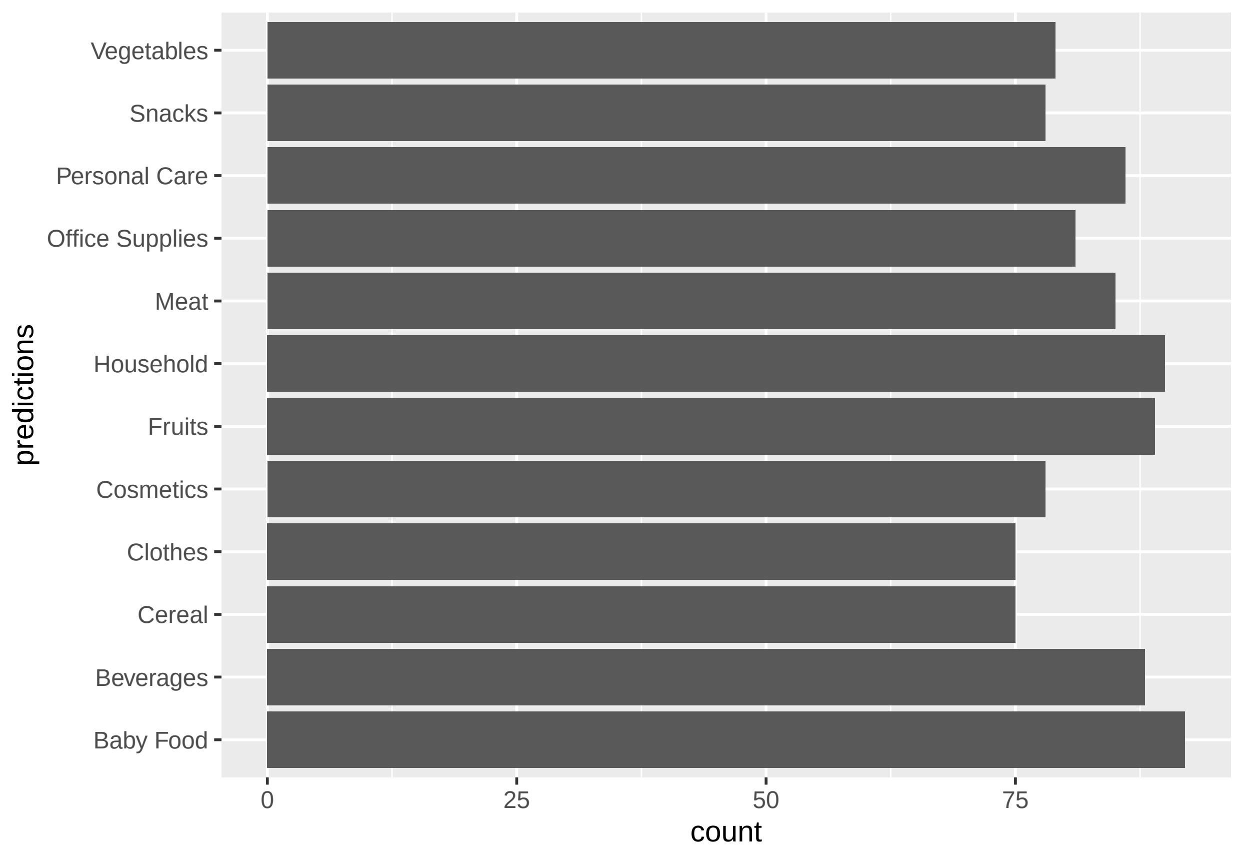
```
## Random Forest
##
## 4004 samples
##   14 predictor
##   12 classes: 'Baby Food', 'Beverages', 'Cereal', 'Clothes', 'Cosmetics', 'Fruits', 'Household', 'Me
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3604, 3603, 3604, 3605, 3602, 3603, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.3456735  0.2848244
##    8    0.8271602  0.8114007
##   14    0.8743731  0.8629269
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 14.
```

```
predictions <- predict(rf, dfEval)
x <- factor(dfEval$Item.Type)
confusionMatrix(predictions, x)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        Baby Food Beverages Cereal Clothes Cosmetics Fruits Household
##   Baby Food              77         0     11       1         0      0         0
##   Beverages               0        86      0       0         0      0         0
##   Cereal                  9         0     66       0         0      0         0
##   Clothes                 0         0      0      74         0      0         0
##   Cosmetics               0         0      0       0        64      0        14
##   Fruits                  0         0      0       0         0     89         0
##   Household               0         0      0       0        20      0        68
##   Meat                    1         0      0       0         0      0         0
##   Office Supplies         2         0      0       0         0      0         2
##   Personal Care           0         3      0       0         0      0         0
##   Snacks                  0         0      0       1         0      0         0
##   Vegetables              0         0      0       1         0      0         0
##                  Reference
## Prediction        Meat Office Supplies Personal Care Snacks Vegetables
##   Baby Food          0               3             0      0          0
##   Beverages          0               0             2      0          0
##   Cereal             0               0             0      0          0
##   Clothes            1               0             0      0          0
##   Cosmetics          0               0             0      0          0
##   Fruits             0               0             0      0          0
##   Household          0               2             0      0          0
##   Meat              52               1             0     28          3
##   Office Supplies    0              77             0      0          0
##   Personal Care      0               1            81      1          0
##   Snacks            24               0             0     49          4
##   Vegetables         2               0             0      1         75
##
## Overall Statistics
##
##                Accuracy : 0.8614
##                  95% CI : (0.8384, 0.8823)
##     No Information Rate : 0.0894
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.8488
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Baby Food Class: Beverages Class: Cereal
## Sensitivity                   0.86517          0.96629       0.85714
## Specificity                   0.98346          0.99779       0.99021
## Pos Pred Value                0.83696          0.97727       0.88000
## Neg Pred Value                0.98673          0.99670       0.98806
## Prevalence                    0.08936          0.08936       0.07731
## Detection Rate                0.07731          0.08635       0.06627
## Detection Prevalence          0.09237          0.08835       0.07530
## Balanced Accuracy             0.92432          0.98204       0.92367
##                      Class: Clothes Class: Cosmetics Class: Fruits
## Sensitivity                 0.96104          0.76190       1.00000
## Specificity                 0.99891          0.98465       1.00000
## Pos Pred Value              0.98667          0.82051       1.00000
## Neg Pred Value              0.99674          0.97821       1.00000
## Prevalence                  0.07731          0.08434       0.08936
## Detection Rate              0.07430          0.06426       0.08936
## Detection Prevalence        0.07530          0.07831       0.08936
## Balanced Accuracy           0.97998          0.87328       1.00000
##                      Class: Household Class: Meat Class: Office Supplies
## Sensitivity                   0.80952     0.65823                0.91667
## Specificity                   0.97588     0.96401                0.99561
## Pos Pred Value                0.75556     0.61176                0.95062
## Neg Pred Value                0.98234     0.97036                0.99235
## Prevalence                    0.08434     0.07932                0.08434
## Detection Rate                0.06827     0.05221                0.07731
## Detection Prevalence          0.09036     0.08534                0.08133
## Balanced Accuracy             0.89270     0.81112                0.95614
##                      Class: Personal Care Class: Snacks Class: Vegetables
## Sensitivity                       0.97590       0.62025           0.91463
## Specificity                       0.99452       0.96838           0.99562
## Pos Pred Value                    0.94186       0.62821           0.94937
## Neg Pred Value                    0.99780       0.96732           0.99237
## Prevalence                        0.08333       0.07932           0.08233
## Detection Rate                    0.08133       0.04920           0.07530
## Detection Prevalence              0.08635       0.07831           0.07932
## Balanced Accuracy                 0.98521       0.79431           0.95513
```

```
dfPred <- as.data.frame(predictions)
ggplot(dfPred, aes(predictions)) +
  geom_bar() +
  coord_flip()
```



Which factors are most important? We can see that total profit is driving the analysis. Decision trees leaving total profit in and taking it out make this more clear.

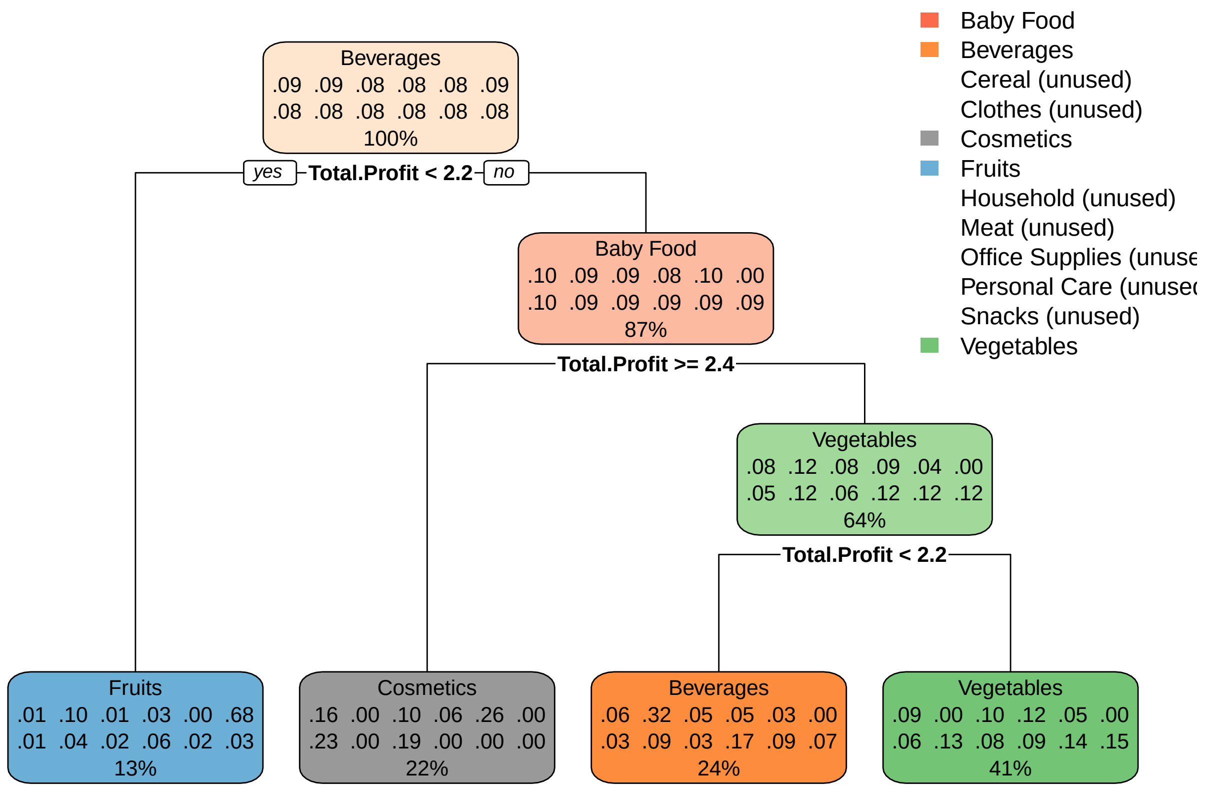
```
varImp(rf)
```

```
## rf variable importance
##
##                                      Overall
## Total.Profit                        100.0000
## Units.Sold                           88.9406
## OrderDaysSince2000                    6.9361
## Order.Lag                             6.1824
## Region_Sub.Saharan.Africa             0.9742
## Order.Priority_H                      0.9089
## Order.Priority_L                      0.8641
## Region_Europe                         0.7902
## Sales.Channel_Offline                 0.7727
## Order.Priority_C                      0.7260
## Region_Asia                           0.6152
## Region_Middle.East.and.North.Africa   0.5185
## Region_Australia.and.Oceania          0.4816
## Region_North.America                  0.0000
```

```
dt <- train(Item.Type~., data=dfTrain, method="rpart", metric=metric, trControl=tc)
library(rpart.plot)
```

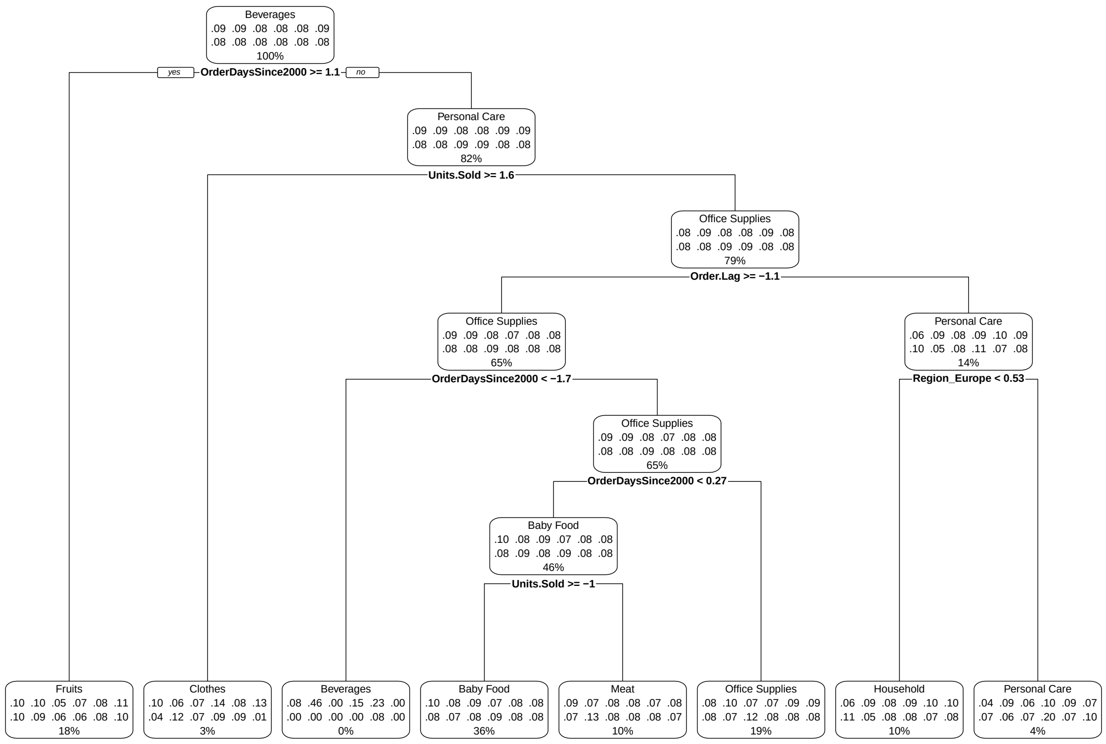
```
## Loading required package: rpart
```

```
rpart.plot(dt$finalModel)
```


Baby Food
Beverages
Cereal (unused)
Clothes (unused)
Cosmetics
Fruits
Household (unused)
Meat (unused)
Office Supplies (unuse
Personal Care (unused
Snacks (unused)
Vegetables
Beverages .09 .09 .08 .08 .08 .09 .08 .08 .08 .08 .08 .08 100%
yes
Total.Profit < 2.2
no
Baby Food .10 .09 .09 .08 .10 .00 .10 .09 .09 .09 .09 .09 87%
Total.Profit >= 2.4
Vegetables .08 .12 .08 .09 .04 .00 .05 .12 .06 .12 .12 .12 64%
Total.Profit < 2.2
Fruits .01 .10 .01 .03 .00 .68 .01 .04 .02 .06 .02 .03 13%
Cosmetics .16 .00 .10 .06 .26 .00 .23 .00 .19 .00 .00 .00 22%
Beverages .06 .32 .05 .05 .03 .00 .03 .09 .03 .17 .09 .07 24%
Vegetables .09 .00 .10 .12 .05 .00 .06 .13 .08 .09 .14 .15 41%

```
dfTrain2 <- dfTrain %>%
  dplyr::select(-Total.Profit)

dt <- train(Item.Type~., data=dfTrain2, method="rpart", metric=metric, trControl=tc)

library(rpart.plot)
rpart.plot(dt$finalModel)
```

```
## Warning: All boxes will be white (the box.palette argument will be ignored) because
## the number of classes in the response 12 is greater than length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```



D. Analyzing the Smaller Dataset Now we examine the smaller dataset and make some comparisons. Since the 5000 database is a superset of this one, we would expect similarities.

Multicollinearity and distributions look the similar. Unit Costs and Item Types continue to match one to one. The standard deviation of Total.Profit is higher, as is the mean. This dataset looks like a sample of the larger one, wth our sample means and standard deviations further from the population compared to the larger one.

A distribution of classes also shows how sparse the data has become. Some classes have 3 or less members and some are missing altogether. These "curse of dimensionality" issues are going to hurt the analysis.

With less data to train on and more variation in the dataset, We expect a loss of predictive power with the drop in n. In fact, our accuracy on the training set degrades to .37% (was 87%) and .28% (was 53%) for RF and MR respectively. Because the data is so sparse, the classes in the training set don't match those in the evaluation set so we can't do predicitons.

```
df2Small <- read.csv("D:\\RStudio\\CUNY_622\\1\\Salesdata_100.csv")

dfSmall <- df2Small %>%
  dplyr::select(-Order.ID)

summary(dfSmall)
```

```
##     Region            Country            Item.Type          Sales.Channel
##  Length:100         Length:100         Length:100         Length:100
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Order.Priority      Order.Date         Ship.Date           Units.Sold
##  Length:100         Length:100         Length:100         Min.   : 124
##  Class :character   Class :character   Class :character   1st Qu.:2836
##  Mode  :character   Mode  :character   Mode  :character   Median :5382
##                                                            Mean   :5129
##                                                            3rd Qu.:7369
##                                                            Max.   :9925
##    Unit.Price       Unit.Cost      Total.Revenue       Total.Cost
##  Min.   :  9.33   Min.   :  6.92   Min.   :   4870   Min.   :   3612
##  1st Qu.: 81.73   1st Qu.: 35.84   1st Qu.: 268721   1st Qu.: 168868
##  Median :179.88   Median :107.28   Median : 752314   Median : 363566
##  Mean   :276.76   Mean   :191.05   Mean   :1373488   Mean   : 931806
##  3rd Qu.:437.20   3rd Qu.:263.33   3rd Qu.:2212045   3rd Qu.:1613870
##  Max.   :668.27   Max.   :524.96   Max.   :5997055   Max.   :4509794
##   Total.Profit
##  Min.   :   1258
##  1st Qu.: 121444
##  Median : 290768
##  Mean   : 441682
##  3rd Qu.: 635829
##  Max.   :1719922
```

```
sd(df2$Total.Profit)
```

```
## [1] 382935.1
```

```
sd(dfSmall$Total.Profit)
```

```
## [1] 438537.9
```

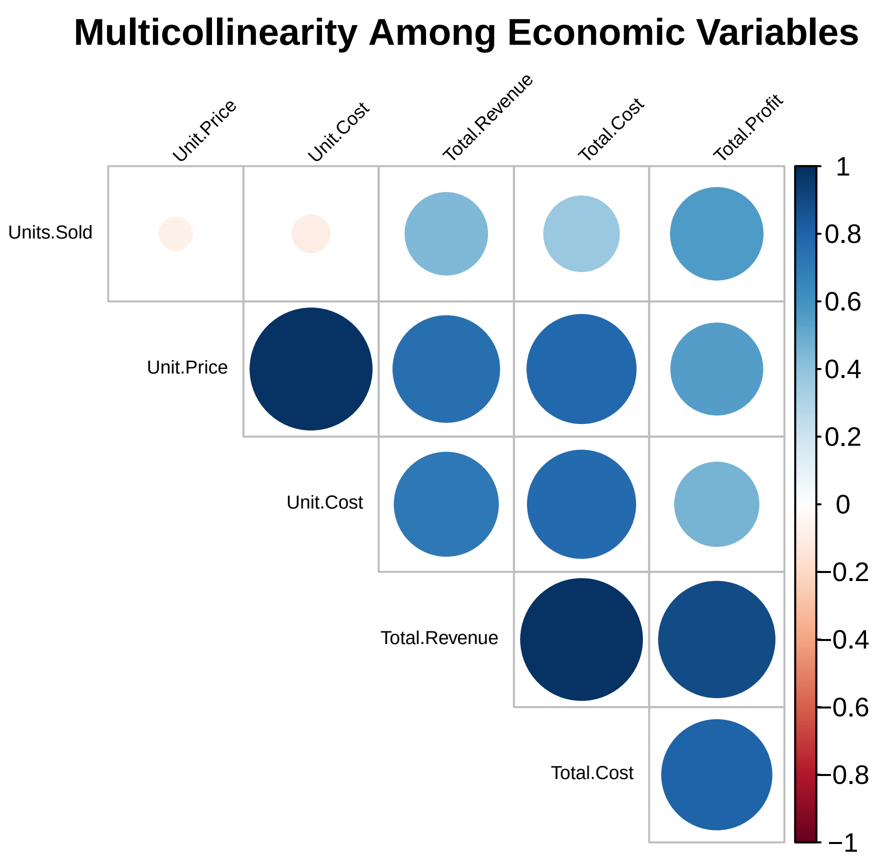
```
mean(df2$Total.Profit)
```

```
## [1] 392644.6
```

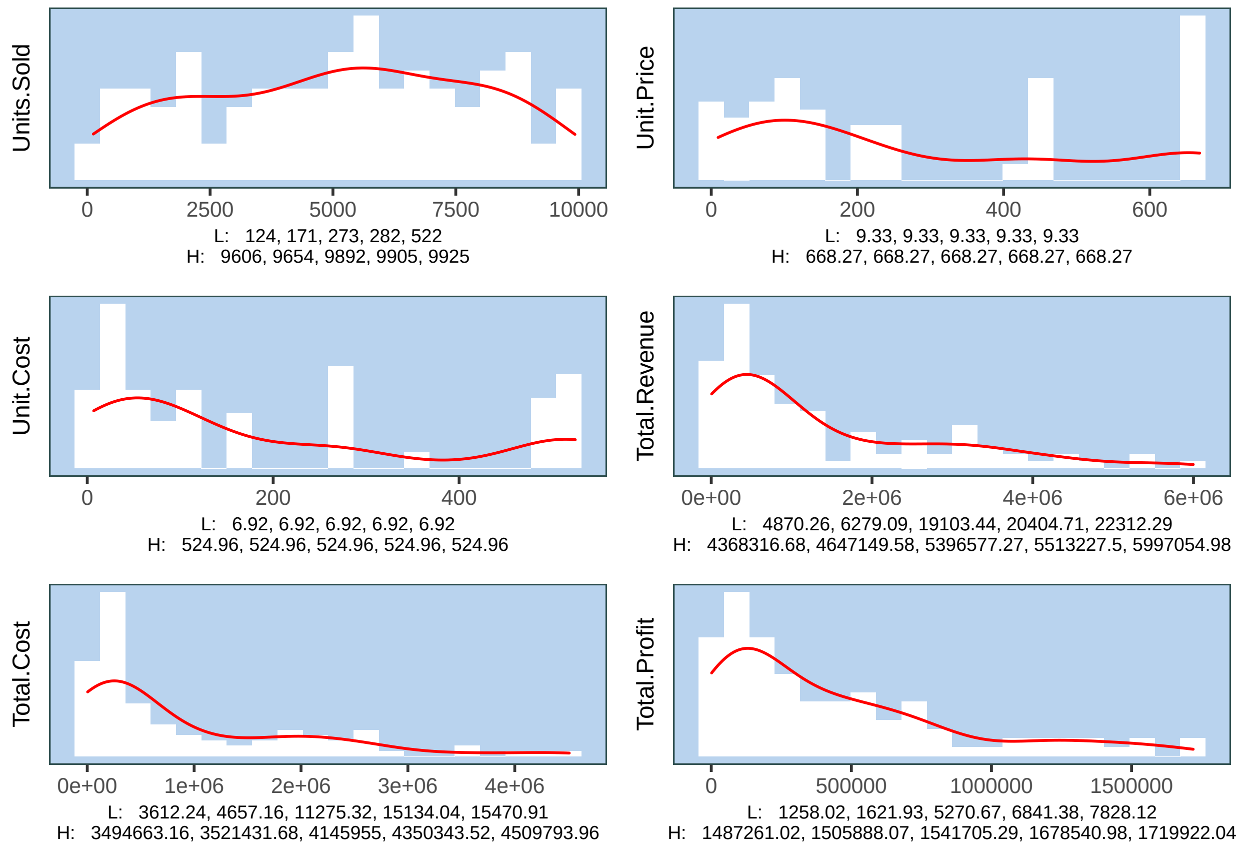
```
mean(dfSmall$Total.Profit)
```

```
## [1] 441682
```

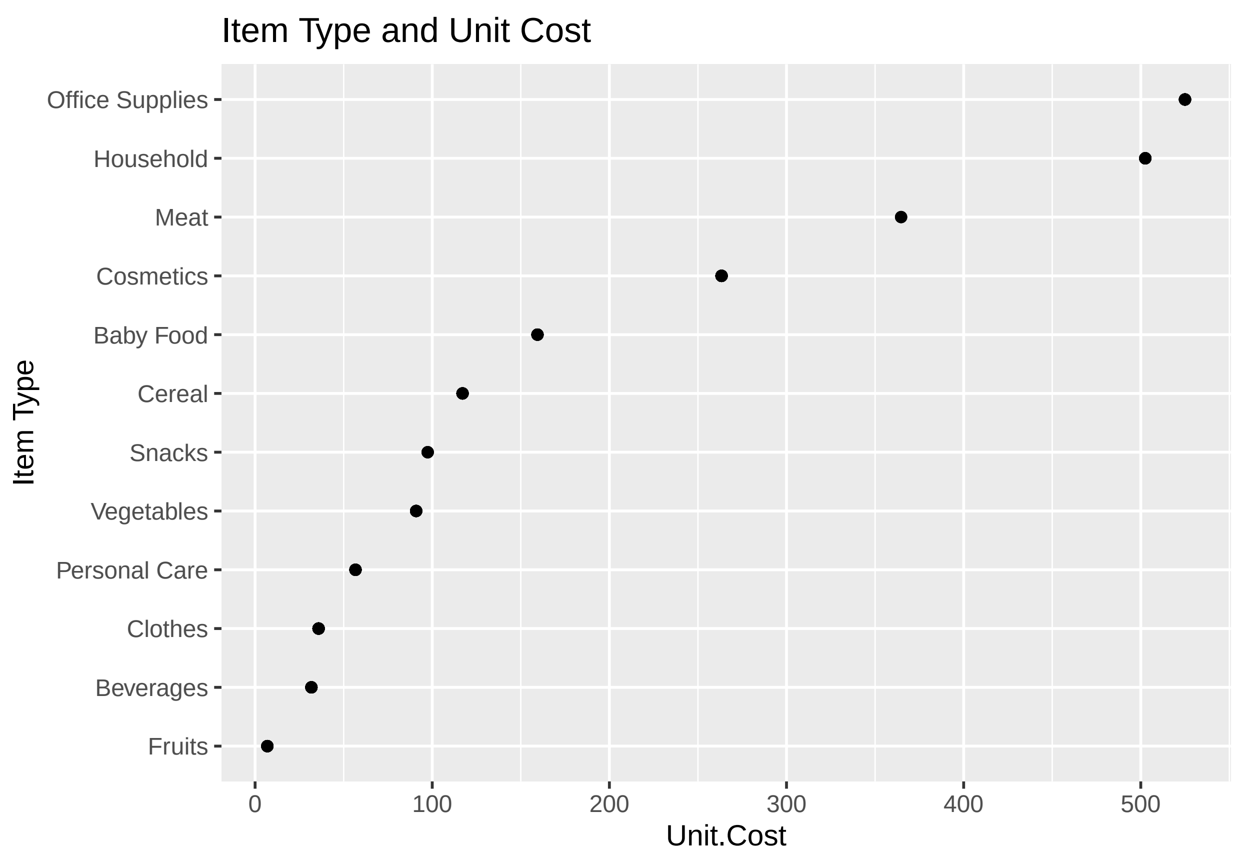
```
dfSmallNum <- dfSmall %>%
  dplyr::select_if(is.numeric)

z <- EHExplore_Multicollinearity(dfSmallNum, title="Multicollinearity Among Economic Variables")
```



```
a <- EHSummarize_SingleColumn_Histograms(dfSmall)
grid.arrange(grobs=a[c(1:6)])
```



```
dfSmall$Item.Type <- as.factor(dfSmall$Item.Type)
ggplot(dfSmall, aes(Unit.Cost, fct_reorder(Item.Type,
                         Unit.Cost))) +
  geom_point() +
  ggtitle("Item Type and Unit Cost") +
    ylab("Item Type")
```



```
df2 <-dfSmall

df2$Item.Type <- factor(df2$Item.Type)
df2$Region <- factor(df2$Region)

df2$Order.Date <- as.Date(df2$Order.Date, format="%m/%d/%Y")
df2$Ship.Date <- as.Date(df2$Ship.Date, format="%m/%d/%Y")

df2$Order.Lag <- as.integer(df2$Ship.Date-df2$Order.Date)
df2$OrderDaysSince2000 <- as.integer(df2$Ship.Date-as.Date("2000-01-01"))

df3 <- df2 %>%
    dplyr::select(-Order.Date, -Ship.Date, -Country, -Unit.Price, -Total.Revenue, -Total.Cost, -Units.So


z=list("Item.Type")

df3a <- df3 %>%
  dplyr::select(-Unit.Cost)

df3a$Units.Sold <- scale(df2$Units.Sold)

df3b <- EHPrepare_CreateDummies(df3a, exclude=z, dropFirst=TRUE)
```

```
## Warning in EHPrepare_CreateDummies(df3a, exclude = z, dropFirst = TRUE): NAs
## introduced by coercion
```

```r
df4 <- EHPrepare_ScaleAllButTarget(df3b, "Item.Type")

df4$Total.Profit = log(df4$Total.Profit+10)

set.seed(042760)
i <- createDataPartition(df4$Item.Type, p=0.80, list=FALSE)
dfEval <- df4[-i,]
dfTrain <- df4[i,]

dfTrain %>% count(Item.Type)
```

```
##          Item.Type  n
## 1        Baby Food  6
## 2        Beverages  7
## 3           Cereal  6
## 4          Clothes 11
## 5        Cosmetics 11
## 6           Fruits  8
## 7        Household  8
## 8             Meat  2
## 9  Office Supplies 10
## 10   Personal Care  8
## 11          Snacks  3
## 12      Vegetables  5
```

```r
tc <- trainControl(method="cv", number=10)
metric <- "Accuracy"

set.seed(042760)
multinom <- train(Item.Type~., data=dfTrain, method="multinom", metric=metric, trControl=tc)
```

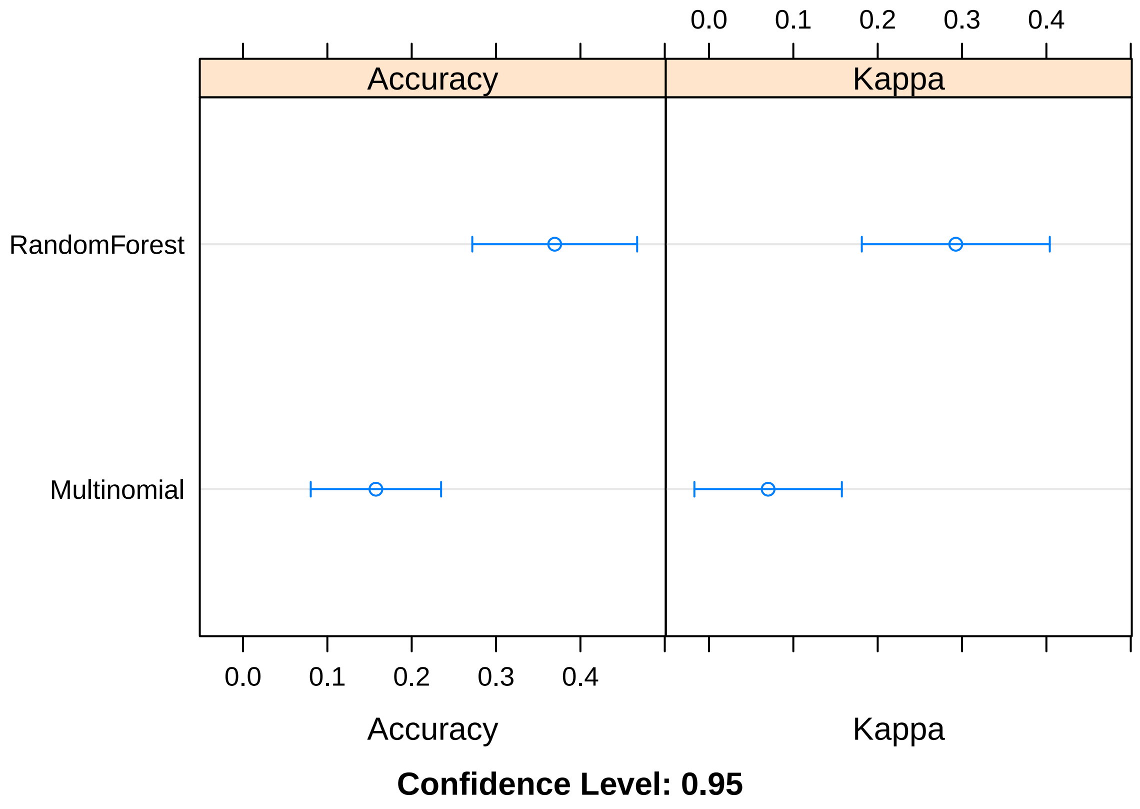
```
## # weights:  192 (165 variable)
## initial  value 188.852905
## iter  10 value 103.018728
## iter  20 value 86.847844
## iter  30 value 77.889433
## iter  40 value 61.551245
## iter  50 value 37.614462
## iter  60 value 9.535358
## iter  70 value 0.034810
## final  value 0.000067
## converged
## # weights:  192 (165 variable)
## initial  value 188.852905
## iter  10 value 108.495298
## iter  20 value 101.795998
## iter  30 value 100.430208
## iter  40 value 100.367548
## iter  50 value 100.366625
## final  value 100.366620
## converged
## # weights:  192 (165 variable)
## initial  value 188.852905
## iter  10 value 103.025184
## iter  20 value 86.878962
## iter  30 value 78.015729
## iter  40 value 63.137111
## iter  50 value 50.358753
## iter  60 value 46.170756
## iter  70 value 45.663236
## iter  80 value 44.748606
## iter  90 value 42.799146
## iter 100 value 41.573355
## final  value 41.573355
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 109.781010
## iter  20 value 92.711717
## iter  30 value 82.052656
## iter  40 value 69.833847
## iter  50 value 52.000654
## iter  60 value 36.160902
## iter  70 value 24.415579
## iter  80 value 5.256947
## iter  90 value 0.012838
## final  value 0.000088
## converged
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 112.965964
## iter  20 value 105.830006
## iter  30 value 105.142407
## iter  40 value 105.122699
## iter  50 value 105.122430
## final  value 105.122428
## converged
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 109.786552
## iter  20 value 92.778603
## iter  30 value 82.219332
## iter  40 value 71.369119
## iter  50 value 60.582670
## iter  60 value 55.746413
## iter  70 value 53.836926
## iter  80 value 52.261126
## iter  90 value 50.038377
## iter 100 value 49.233692
## final  value 49.233692
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 107.682412
## iter  20 value 90.103370
## iter  30 value 81.426142
## iter  40 value 64.376516
## iter  50 value 45.913681
## iter  60 value 29.148818
## iter  70 value 1.042812
## iter  80 value 0.003636
## final  value 0.000092
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 112.190126
## iter  20 value 104.632444
## iter  30 value 103.445546
## iter  40 value 103.395070
## iter  50 value 103.394090
## final  value 103.394080
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 107.687509
## iter  20 value 90.133428
## iter  30 value 81.529985
## iter  40 value 66.198126
## iter  50 value 57.085787
## iter  60 value 51.292198
## iter  70 value 50.131813
## iter  80 value 49.479003
## iter  90 value 46.473696
## iter 100 value 44.986370
## final  value 44.986370
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 106.183407
## iter  20 value 87.985763
## iter  30 value 77.955194
## iter  40 value 63.692390
## iter  50 value 47.400562
## iter  60 value 13.711074
## iter  70 value 0.036085
## final  value 0.000073
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 111.457333
## iter  20 value 103.177487
## iter  30 value 101.778529
## iter  40 value 101.700607
## iter  50 value 101.699676
## final  value 101.699669
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 106.189624
## iter  20 value 88.013050
## iter  30 value 78.082205
## iter  40 value 65.332619
## iter  50 value 55.820567
## iter  60 value 47.931371
## iter  70 value 43.938002
## iter  80 value 42.348125
## iter  90 value 40.040644
## iter 100 value 37.402045
## final  value 37.402045
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 104.822217
## iter  20 value 90.997559
## iter  30 value 81.836319
## iter  40 value 68.880383
## iter  50 value 54.221817
## iter  60 value 35.866718
## iter  70 value 0.705101
## iter  80 value 0.001563
## final  value 0.000059
## converged
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 109.685619
## iter  20 value 102.559755
## iter  30 value 101.530260
## iter  40 value 101.481221
## iter  50 value 101.480394
## final  value 101.480388
## converged
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 104.828498
## iter  20 value 91.018826
## iter  30 value 81.931492
## iter  40 value 69.941342
## iter  50 value 59.533375
## iter  60 value 51.948611
## iter  70 value 49.256069
## iter  80 value 48.705351
## iter  90 value 45.645165
## iter 100 value 43.785857
## final  value 43.785857
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 104.109247
## iter  20 value 85.021226
## iter  30 value 75.126198
## iter  40 value 64.820763
## iter  50 value 46.680819
## iter  60 value 29.185376
## iter  70 value 3.705876
## iter  80 value 0.010863
## final  value 0.000052
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 109.123323
## iter  20 value 100.811107
## iter  30 value 99.363505
## iter  40 value 99.309025
## iter  50 value 99.308033
## final  value 99.308026
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 104.114925
## iter  20 value 85.053050
## iter  30 value 75.273093
## iter  40 value 66.000802
## iter  50 value 54.990251
## iter  60 value 49.882478
## iter  70 value 48.410854
## iter  80 value 47.898191
## iter  90 value 45.914812
## iter 100 value 44.394154
## final  value 44.394154
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 108.938720
## iter  20 value 94.591588
## iter  30 value 84.113248
## iter  40 value 70.670081
## iter  50 value 63.846065
## iter  60 value 50.051782
## iter  70 value 24.754975
## iter  80 value 0.982297
## iter  90 value 0.002006
## final  value 0.000068
## converged
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 114.968680
## iter  20 value 109.180895
## iter  30 value 107.143357
## iter  40 value 107.045517
## iter  50 value 107.043457
## final  value 107.043433
## converged
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 108.945884
## iter  20 value 94.614992
## iter  30 value 84.225708
## iter  40 value 71.743362
## iter  50 value 65.716613
## iter  60 value 57.550100
## iter  70 value 54.762417
## iter  80 value 53.849011
## iter  90 value 52.398940
## iter 100 value 49.456141
## final  value 49.456141
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 104.290524
## iter  20 value 82.574340
## iter  30 value 73.118981
## iter  40 value 62.783897
## iter  50 value 52.384909
## iter  60 value 37.005819
## iter  70 value 20.382619
## iter  80 value 14.634728
## iter  90 value 0.715800
## iter 100 value 0.003996
## final  value 0.003996
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 106.074020
## iter  20 value 97.539203
## iter  30 value 96.523158
## iter  40 value 96.462725
## iter  50 value 96.461798
## final  value 96.461791
## converged
## # weights:  192 (165 variable)
## initial  value 186.367999
## iter  10 value 104.298248
## iter  20 value 82.604488
## iter  30 value 73.239524
## iter  40 value 63.973153
## iter  50 value 55.968099
## iter  60 value 50.208676
## iter  70 value 47.777025
## iter  80 value 46.393596
## iter  90 value 45.538903
## iter 100 value 43.745298
## final  value 43.745298
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 112.078321
## iter  20 value 98.051520
## iter  30 value 88.707983
## iter  40 value 71.707723
## iter  50 value 51.998267
## iter  60 value 34.989380
## iter  70 value 20.809800
## iter  80 value 7.857126
## iter  90 value 0.090202
## iter 100 value 0.000324
## final  value 0.000324
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 116.668174
## iter  20 value 109.653751
## iter  30 value 108.783803
## iter  40 value 108.737306
## iter  50 value 108.736346
## final  value 108.736338
## converged
## # weights:  192 (165 variable)
## initial  value 193.822719
## iter  10 value 112.083514
## iter  20 value 98.073261
## iter  30 value 88.813909
## iter  40 value 73.788206
## iter  50 value 61.612169
## iter  60 value 57.620107
## iter  70 value 55.728581
## iter  80 value 54.817893
## iter  90 value 52.580320
## iter 100 value 49.708638
## final  value 49.708638
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 108.749890
## iter  20 value 94.366067
## iter  30 value 84.629970
## iter  40 value 70.951442
## iter  50 value 59.682576
## iter  60 value 49.126230
## iter  70 value 29.303789
## iter  80 value 10.648187
## iter  90 value 0.321788
## iter 100 value 0.000807
## final  value 0.000807
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 111.846118
## iter  20 value 105.630068
## iter  30 value 104.504918
## iter  40 value 104.444910
## iter  50 value 104.443333
## final  value 104.443309
## converged
## # weights:  192 (165 variable)
## initial  value 191.337812
## iter  10 value 108.755903
## iter  20 value 94.389367
## iter  30 value 84.727622
## iter  40 value 72.240352
## iter  50 value 64.407086
## iter  60 value 59.702630
## iter  70 value 54.154437
## iter  80 value 53.652721
## iter  90 value 52.134297
## iter 100 value 50.537953
## final  value 50.537953
## stopped after 100 iterations
## # weights:  192 (165 variable)
## initial  value 211.217065
## iter  10 value 125.853319
## iter  20 value 110.695406
## iter  30 value 101.510521
## iter  40 value 81.801680
## iter  50 value 68.583394
## iter  60 value 63.222024
## iter  70 value 62.571319
## iter  80 value 62.108147
## iter  90 value 60.437964
## iter 100 value 58.679128
## final  value 58.679128
## stopped after 100 iterations
```

```
rf <- train(Item.Type~., data=dfTrain, method="rf", metric=metric, trControl=tc)
```

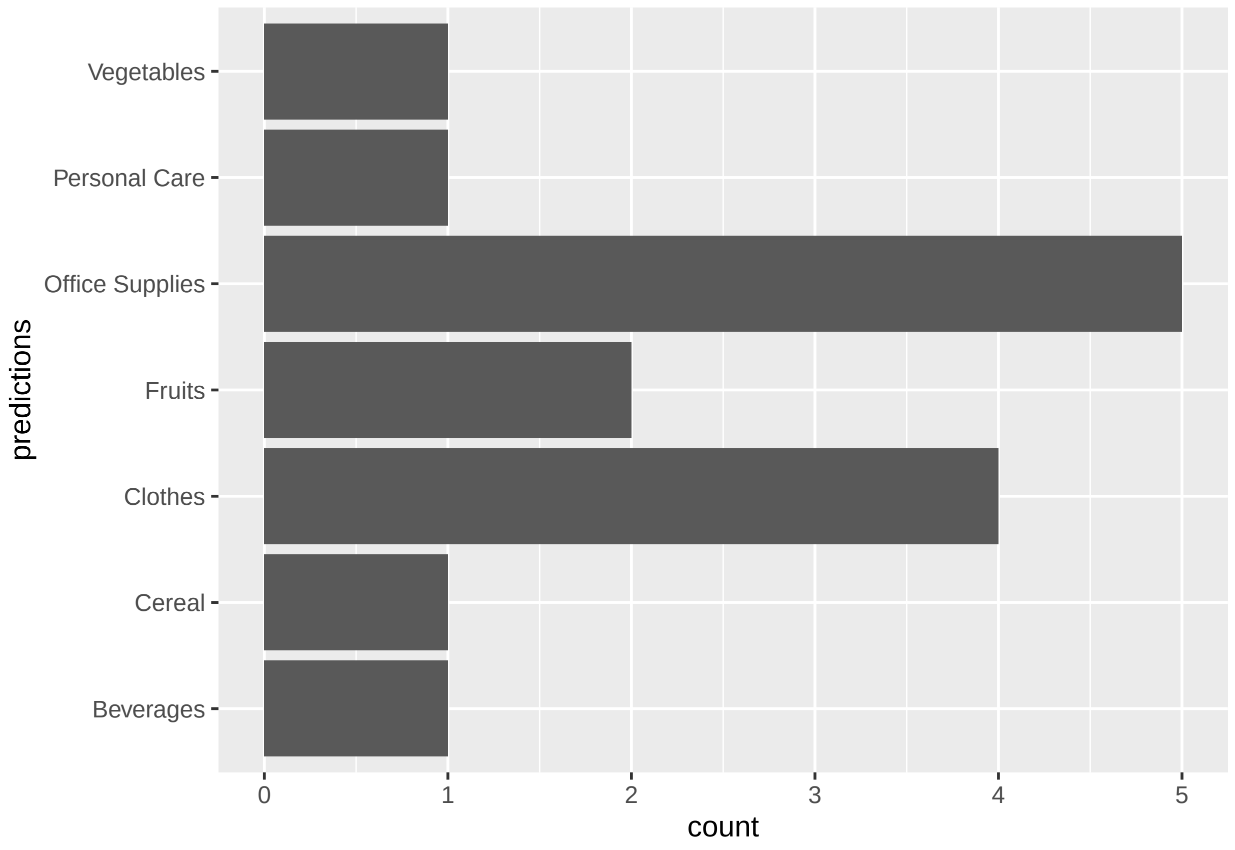
```
results <- resamples(list(Multinomial=multinom, RandomForest=rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Multinomial, RandomForest
## Number of resamples: 10
##
## Accuracy
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Multinomial   0.0 0.1000000 0.1250000 0.1575794 0.2023810 0.3750000    0
## RandomForest  0.2 0.2708333 0.3333333 0.3695455 0.4821429 0.5714286    0
##
## Kappa
##                      Min.    1st Qu.     Median       Mean   3rd Qu.      Max.
## Multinomial  -0.10344828 0.01262772 0.02249747 0.07011721 0.1064619 0.3220339
## RandomForest  0.08045977 0.16359447 0.26527212 0.29258033 0.4198565 0.5227273
##              NA's
## Multinomial     0
## RandomForest    0
```

```
dotplot(results)
```



```
predictions <- predict(rf, dfEval)
x <- factor(dfEval$Item.Type)
#confusionMatrix(predictions, x)

dfPred <- as.data.frame(predictions)
ggplot(dfPred, aes(predictions)) +
  geom_bar() +
  coord_flip()
```



E. Conclusion Item types were predicted from other aspects of country sales records, such as total profits, region, unit sales, priority, etc. Random forest outperformed multinomial regression for both datasets. The smaller dataset suffered from low n and sparseness - with a corresponding drop in accuracy.