Age Old Battle: Fruits and Vegetables

Alister Salmon

February 10, 2024

```
# clear the R environment
rm(list=ls())

# import required libraries
library(dplyr)
library(plotly)
library(ggplot2)
library(caret)
```

Data Processing

```
# import the data
data.veg = data.frame(read.csv('Vegetable Prices 2020.csv'))
data.fruit = data.frame(read.csv('Fruit Prices 2020.csv'))
```

```
# rename columns
colnames(data.veg)[1] = 'Food'
colnames(data.fruit)[1] = 'Food'

# create type column
data.veg['isFruit'] = 0
data.fruit['isFruit'] = 1

# combine them
data = data.frame(rbind(data.veg, data.fruit))

head(data)
```

```
##           Food   Form RetailPrice RetailPriceUnit  Yield CupEquivalentSize
## 1 Acorn squash  Fresh      1.1804       per pound 0.4586            0.4519
## 2    Artichoke  Fresh      2.1913       per pound 0.3750            0.3858
## 3    Artichoke Canned      3.4119       per pound 0.6500            0.3858
## 4    Asparagus  Fresh      2.7576       per pound 0.4938            0.3968
## 5    Asparagus Canned      3.1269       per pound 0.6500            0.3968
## 6    Asparagus Frozen      6.7045       per pound 1.0335            0.3968
##   CupEquivalentUnit CupEquivalentPrice isFruit
## 1            pounds             1.1633       0
## 2            pounds             2.2545       0
```

```
## 3            pounds             2.0251       0
## 4            pounds             2.2159       0
## 5            pounds             1.9090       0
## 6            pounds             2.5742       0
```

```
  tail(data)
```

```
##                            Food   Form RetailPrice RetailPriceUnit Yield
## 150 Pomegranate, ready-to-drink  Juice      3.1220        per pint  1.00
## 151                  Raspberries  Fresh      6.6391       per pound  0.96
## 152                  Raspberries Frozen      4.1877       per pound  1.00
## 153                Strawberries  Fresh      2.5800       per pound  0.94
## 154                Strawberries Frozen      2.8189       per pound  1.00
## 155                   Watermelon  Fresh      0.3604       per pound  0.52
##     CupEquivalentSize CupEquivalentUnit CupEquivalentPrice isFruit
## 150            8.0000      fluid ounces             1.5610       1
## 151            0.3197            pounds             2.2107       1
## 152            0.3307            pounds             1.3849       1
## 153            0.3197            pounds             0.8774       1
## 154            0.3307            pounds             0.9322       1
## 155            0.3307            pounds             0.2292       1
```

```
  # export the cleaned data
  write.csv(data, 'fruit_veg.csv', row.names = FALSE)
```

Spilt Milk

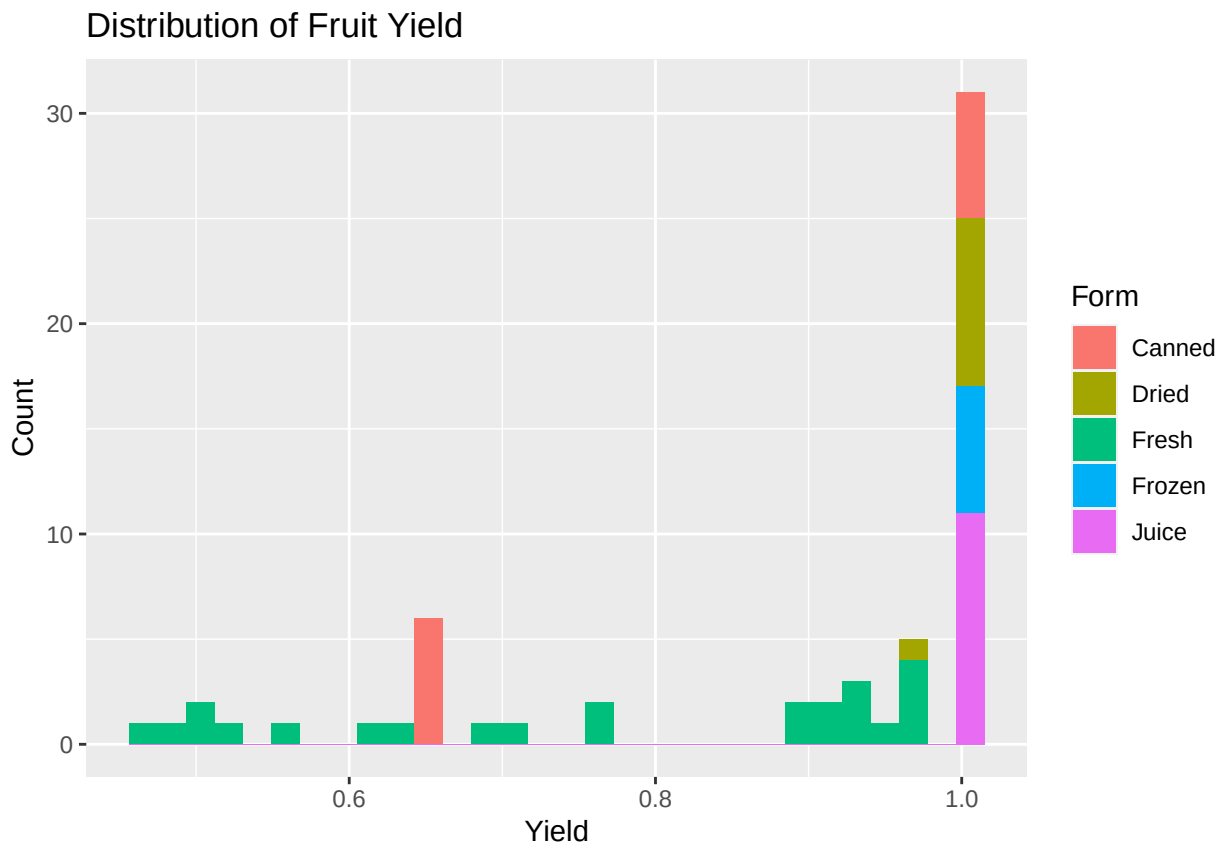
Visualize the distribution of yield and price of fruits and vegetable by form

Methodology

Distribution Based on Yield

```
  # histogram
  ggplot(data.fruit, aes(x=Yield, fill=Form)) +
    geom_histogram() + ggtitle('Distribution of Fruit Yield') + ylab('Count')
```

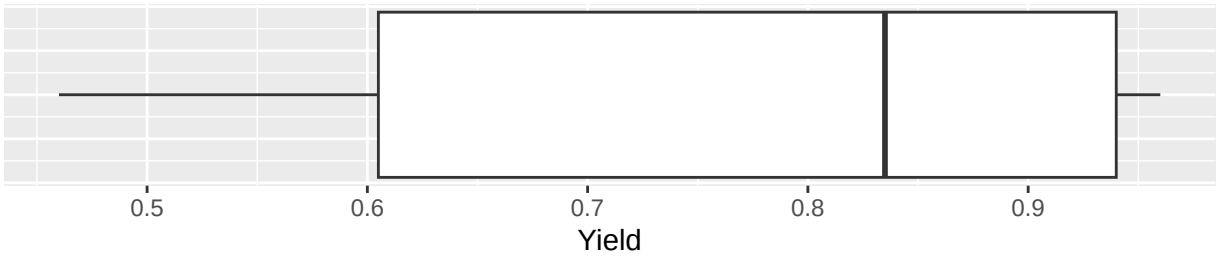
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggsave('visuals/fruit_hist_yield.jpg')
```

```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
# box plot of fresh
ggplot(filter(data.fruit, Form=='Fresh'), aes(x=Yield)) + geom_boxplot() +
  theme(axis.text.y = element_blank(), axis.line.y = element_blank(),
        axis.ticks.y = element_blank()) +
  coord_fixed(ratio = 0.1)
```

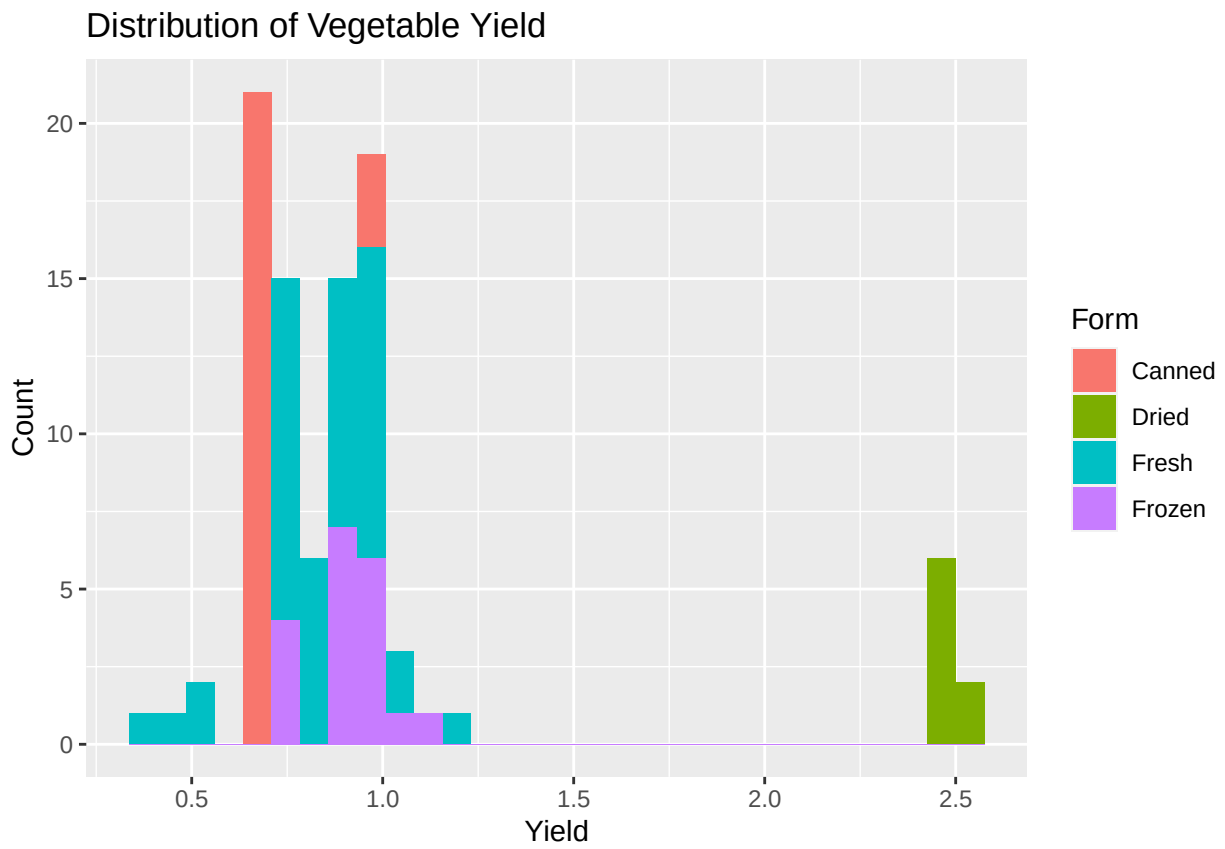



```
  ggsave('visuals/fruit_boxplot_fresh.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
  # histogram
  ggplot(data.veg, aes(x=Yield, fill=Form)) +
    geom_histogram() + ggtitle('Distribution of Vegetable Yield') + ylab('Count')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggsave('visuals/veg_hist_yield.jpg')
```

```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
# box plot of fresh
ggplot(filter(data.veg, Form=='Fresh'), aes(x=Yield)) + geom_boxplot() +
  theme(axis.text.y = element_blank(), axis.line.y = element_blank(),
        axis.ticks.y = element_blank()) +
  coord_fixed(ratio = 0.1)
```

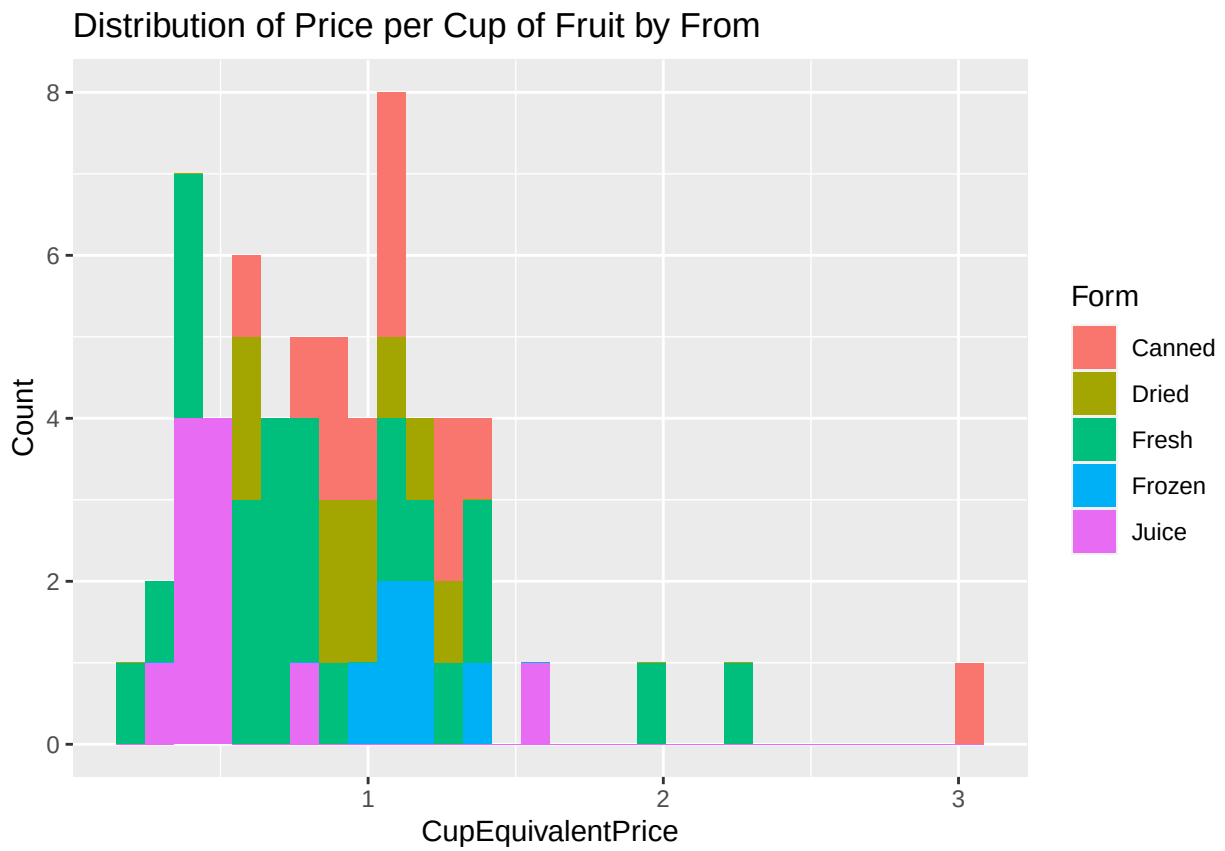

```
ggsave('visuals/veg_boxplot_fresh.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

Distribution Based on Price per Cup

```
# histogram
ggplot(data.fruit, aes(x=CupEquivalentPrice, fill=Form)) +
  geom_histogram() + ggtitle('Distribution of Price per Cup of Fruit by From') +
  ylab('Count')
```

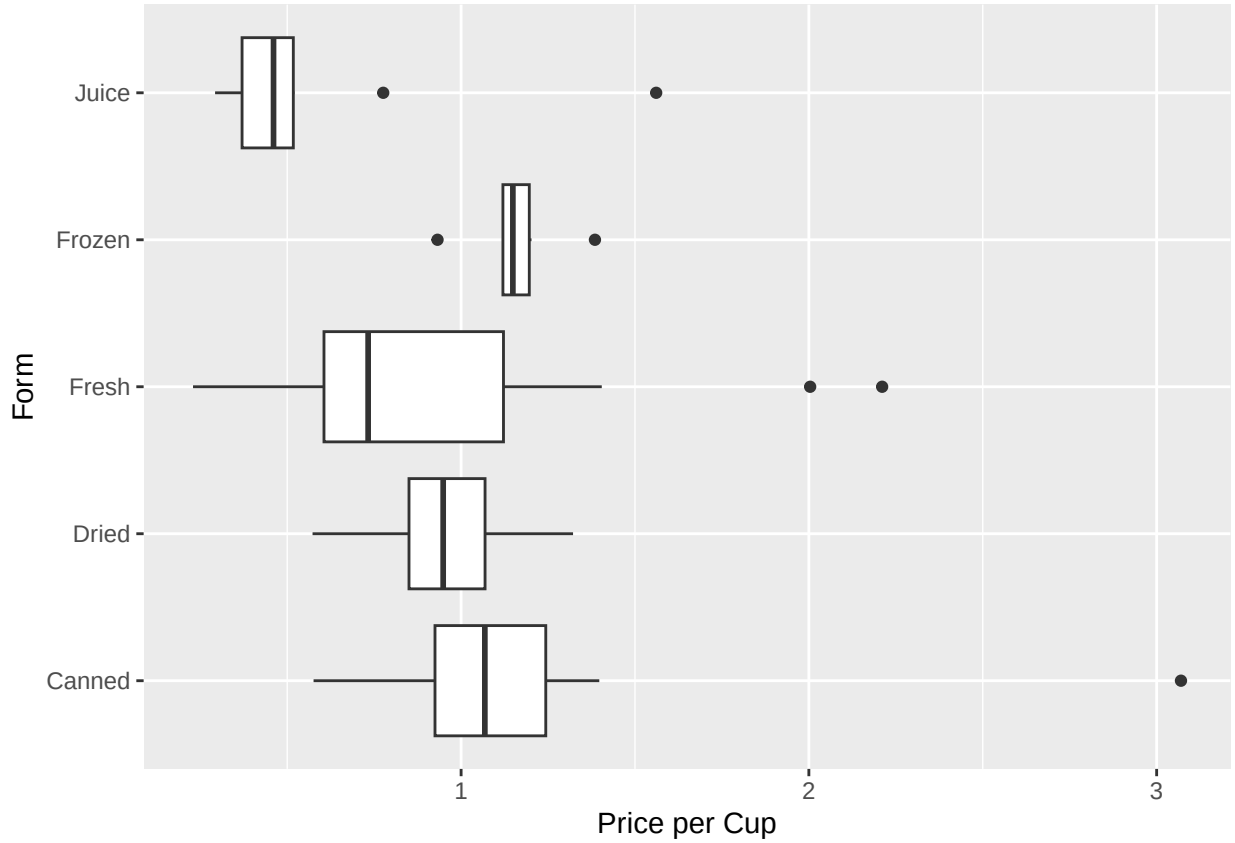
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggsave('visuals/fruit_hist_price.jpg')
```

```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
# box plot
ggplot(data.fruit, aes(x=CupEquivalentPrice, y=Form)) + geom_boxplot() +
  xlab('Price per Cup')
```

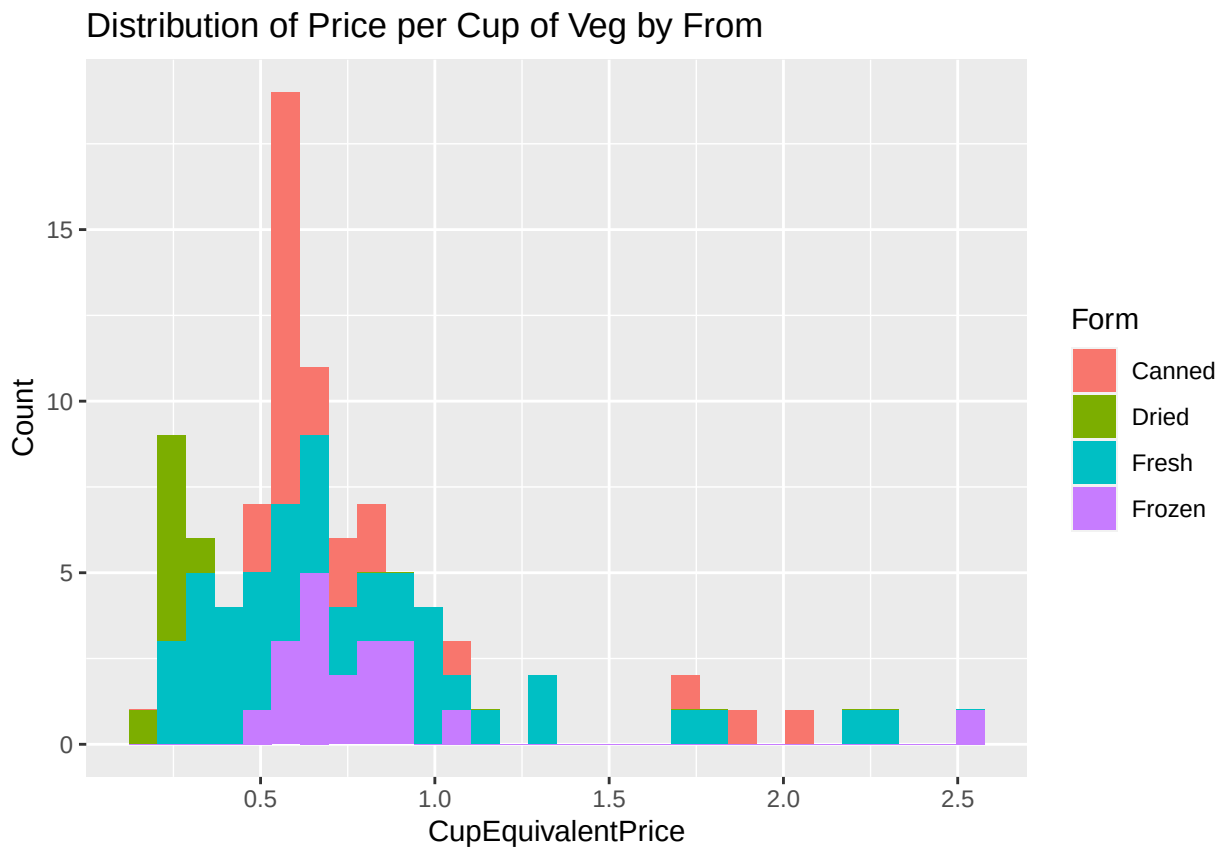



```
  ggsave('visuals/fruit_boxplot_price.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
  # histogram
  ggplot(data.veg, aes(x=CupEquivalentPrice, fill=Form)) +
    geom_histogram() + ggtitle('Distribution of Price per Cup of Veg by From') +
    ylab('Count')
```

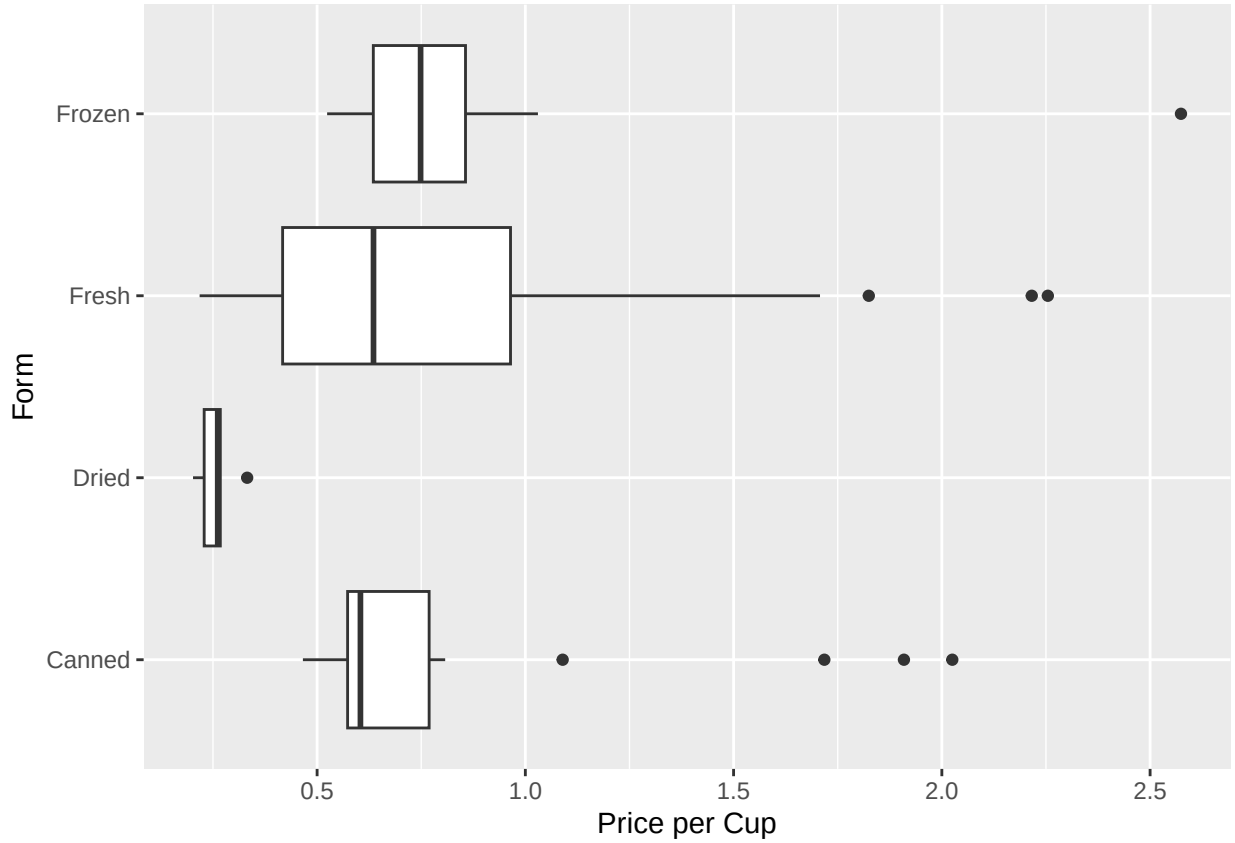
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggsave('visuals/veg_hist_price.jpg')
```

```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
# box plot
ggplot(data.veg, aes(x=CupEquivalentPrice, y=Form)) + geom_boxplot() +
  xlab('Price per Cup')
```




```
ggsave('visuals/veg_boxplot_price.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

Results

Price per cup of fruits and vegetables is roughly equivalent. Canned, dried, and frozen vegetables tend to be cheaper than that of fruit.

Apples to Oranges

Compare fruits (prices and yield)

Comparisons:

Cheapest and most expensive fruits

Avg. price/yields of fresh vs canned vs juice vs dried vs frozen

Methodology

```r
# order the data by CupEquivalentPrice
data.fruit.cup_price.ascending =
  data.fruit[order(data.fruit$CupEquivalentPrice),]

# get the least expensive
head(select(data.fruit.cup_price.ascending, Food, Form, CupEquivalentPrice), 5)
```

```
##                            Food  Form CupEquivalentPrice
## 62                    Watermelon Fresh             0.2292
## 9                        Bananas Fresh             0.2712
## 4     Apples, frozen concentrate Juice             0.2926
## 52 Pineapple, frozen concentrate Juice             0.3486
## 29    Grapes, frozen concentrate Juice             0.3559
```

```r
# get the most expensive
tail(select(data.fruit.cup_price.ascending, Food, Form, CupEquivalentPrice), 5)
```

```
##                                 Food   Form CupEquivalentPrice
## 13                       Blueberries  Fresh             1.4045
## 57       Pomegranate, ready-to-drink  Juice             1.5610
## 11                       Blackberries  Fresh             2.0037
## 58                        Raspberries  Fresh             2.2107
## 17 Cherries, packed in syrup or water Canned             3.0700
```

```r
data.fruit.yield.ascending = data.fruit[order(data.fruit$Yield),]

head(select(data.fruit.yield.ascending, Food, Form, Yield), 5)
```

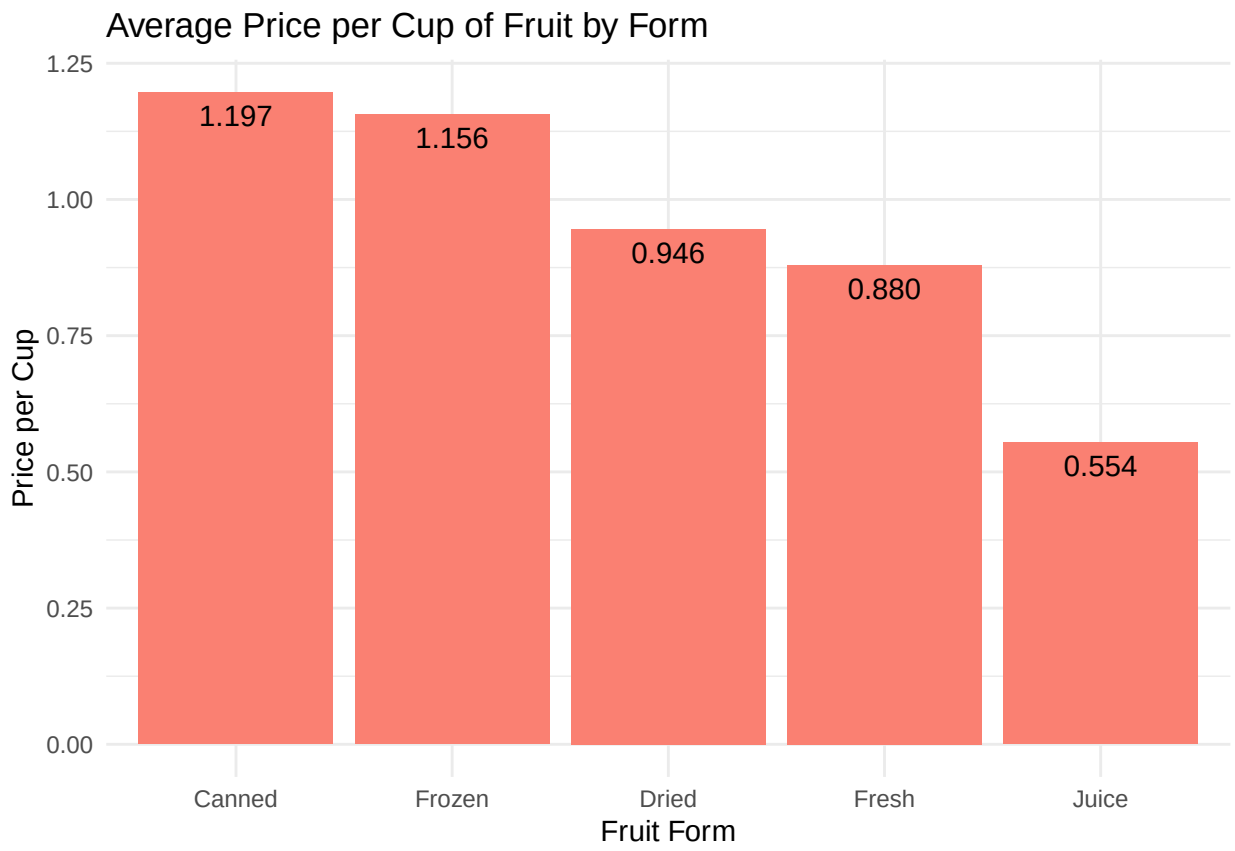
```
##          Food  Form Yield
## 30   Honeydew Fresh  0.46
## 24 Grapefruit Fresh  0.49
## 15 Cantaloupe Fresh  0.51
## 47  Pineapple Fresh  0.51
## 62 Watermelon Fresh  0.52
```

```r
tail(select(data.fruit.yield.ascending, Food, Form, Yield), 5)
```

```
##                            Food   Form Yield
## 54                Plum (prunes)  Dried     1
## 55 Plum (prune), ready-to-drink  Juice     1
## 57  Pomegranate, ready-to-drink  Juice     1
## 59                  Raspberries Frozen     1
## 61                Strawberries Frozen     1
```

```r
data.fruit.avg_form.price = aggregate(CupEquivalentPrice~Form, data=data.fruit, mean)

ggplot(data=data.fruit.avg_form.price, aes(x=reorder(Form,-CupEquivalentPrice),
                                y=CupEquivalentPrice)) +
  geom_bar(stat='identity', fill='salmon') + theme_minimal() +
  xlab('Fruit Form') + ylab('Price per Cup') +
  geom_text(aes(label=sprintf('%.3f', CupEquivalentPrice)), vjust=1.6) +
  ggtitle('Average Price per Cup of Fruit by Form')
```

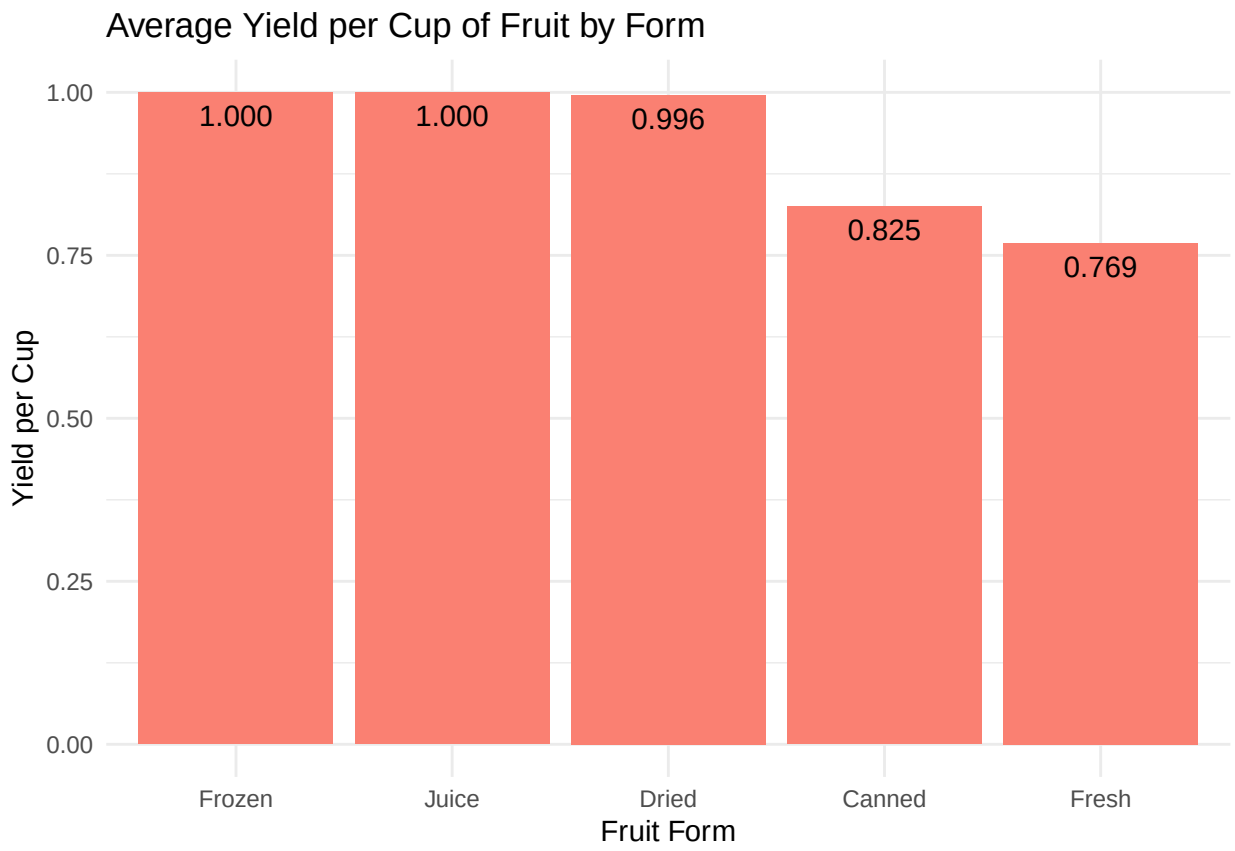



```
ggsave('visuals/fruit_price_by_form.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
data.fruit.avg_form.yield = aggregate(Yield~Form, data=data.fruit, mean)

ggplot(data=data.fruit.avg_form.yield, aes(x=reorder(Form,-Yield),
                                y=Yield)) +
  geom_bar(stat='identity', fill='salmon') + theme_minimal() +
  xlab('Fruit Form') + ylab('Yield per Cup') +
  geom_text(aes(label=sprintf('%.3f', Yield)), vjust=1.6) +
  ggtitle('Average Yield per Cup of Fruit by Form')
```

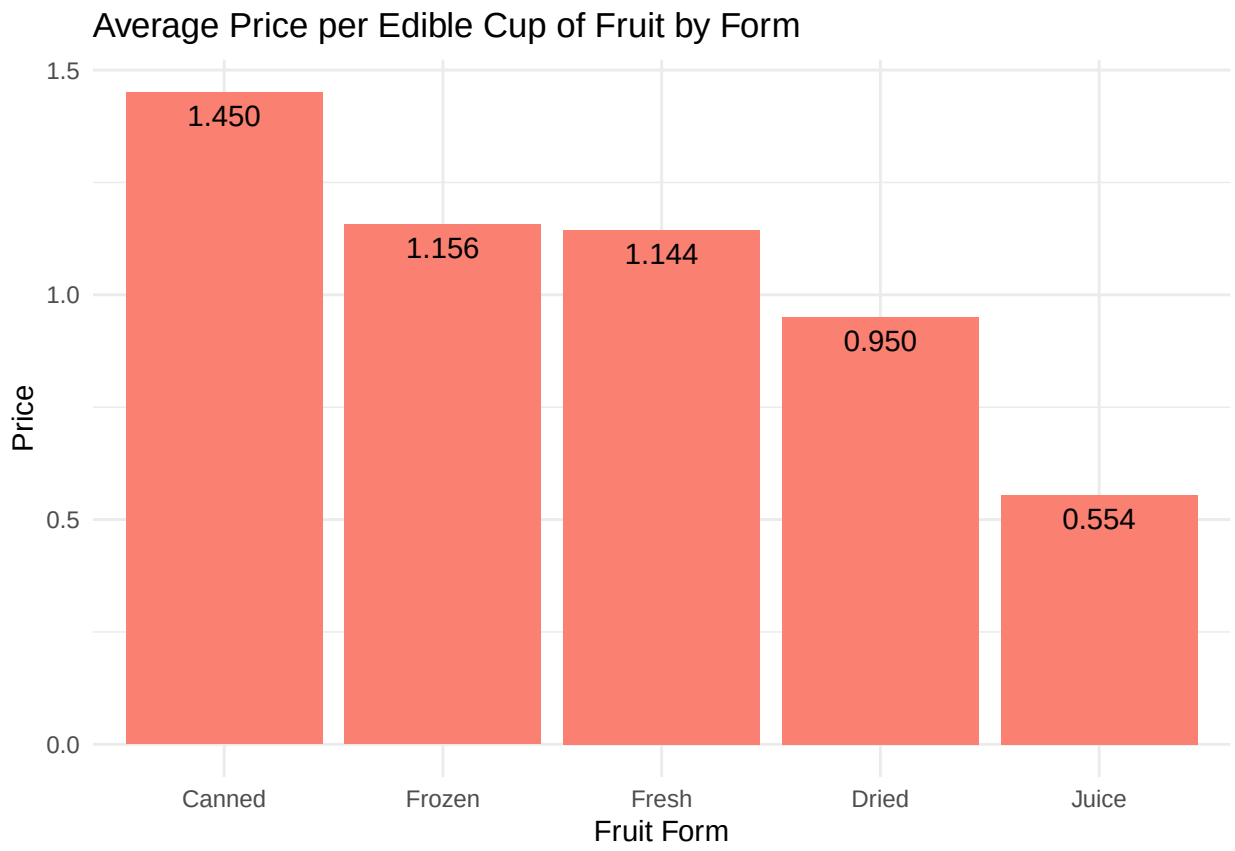



```
ggsave('visuals/fruit_yield_by_form.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
data.fruit.avg_form.avg_price_yield =
  data.frame(data.fruit.avg_form.price$Form,
             data.fruit.avg_form.price$CupEquivalentPrice /
               data.fruit.avg_form.yield$Yield)
colnames(data.fruit.avg_form.avg_price_yield) = c('Form','Price')

ggplot(data=data.fruit.avg_form.avg_price_yield,
       aes(x=reorder(Form,-Price), y=Price)) +
  geom_bar(stat='identity', fill='salmon') + theme_minimal() +
  xlab('Fruit Form') + ylab('Price') +
  geom_text(aes(label=sprintf('%.3f', Price)), vjust=1.6) +
  ggtitle('Average Price per Edible Cup of Fruit by Form')
```




```
ggsave('visuals/fruit_price_per_yield.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

Results

Cheapest form of fruit by edible cup is juice. Fresh fruit is suprisingly cheap; however, this could be due to the bananas and watermelon which are abnormally cheap. The most expensive form is frozen. I did not expect this; however, these results do not take into account calorie count. If that data was available, I am sure that the results would be much different.

Carrots to Cabbages

Compare vegetables (prices and yield)

Comparisons:

Cheapest and most expensive vegetables

Avg. price/yields of fresh vs canned vs frozen vs dried

Methodology

```
  # order the data by CupEquivalentPrice
  data.veg.cup_price.ascending =
    data.veg[order(data.veg$CupEquivalentPrice),]

  # get the least expensive
  head(select(data.veg.cup_price.ascending, Food, Form, CupEquivalentPrice), 5)
```

```
##                Food  Form CupEquivalentPrice
## 74      Pinto beans Dried             0.2021
## 10      Black beans Dried             0.2149
## 75         Potatoes Fresh             0.2179
## 68       Navy beans Dried             0.2335
## 53 Lettuce, iceberg Fresh             0.2540
```

```
  # get the most expensive
  tail(select(data.veg.cup_price.ascending, Food, Form, CupEquivalentPrice), 5)
```

```
##        Food   Form CupEquivalentPrice
## 5 Asparagus Canned             1.9090
## 3 Artichoke Canned             2.0251
## 4 Asparagus  Fresh             2.2159
## 2 Artichoke  Fresh             2.2545
## 6 Asparagus Frozen             2.5742
```

```
  data.veg.yield.ascending = data.veg[order(data.veg$Yield),]

  head(select(data.veg.yield.ascending, Food, Form, Yield), 5)
```

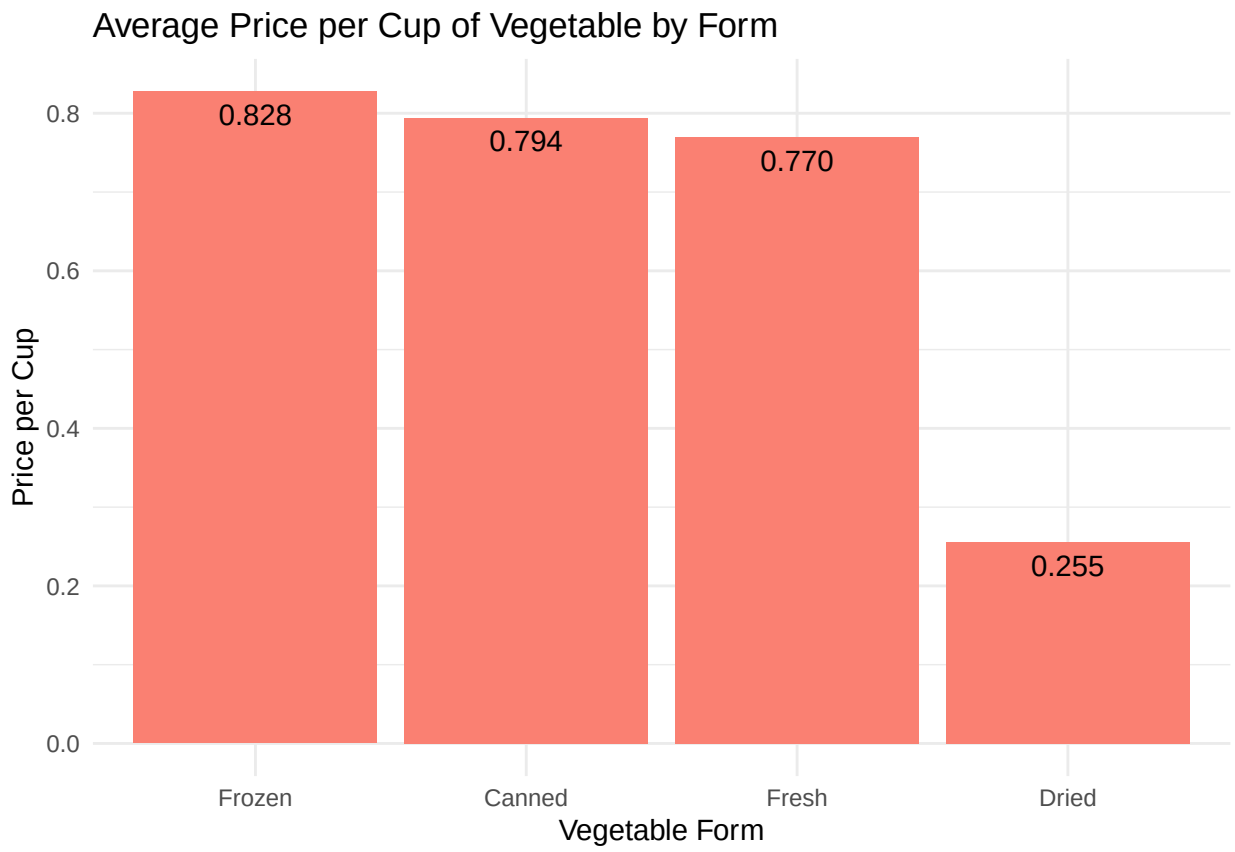
```
##            Food   Form  Yield
## 2     Artichoke  Fresh 0.3750
## 1  Acorn squash  Fresh 0.4586
## 4     Asparagus  Fresh 0.4938
## 35         Corn  Fresh 0.5400
## 3     Artichoke Canned 0.6500
```

```
  tail(select(data.veg.yield.ascending, Food, Form, Yield), 5)
```

```
##             Food  Form  Yield
## 52       Lentils Dried 2.4692
## 68    Navy beans Dried 2.4692
## 74   Pinto beans Dried 2.4692
## 12 Blackeye peas Dried 2.5397
## 58    Lima beans Dried 2.5397
```

```
  data.veg.avg_form.price = aggregate(CupEquivalentPrice~Form, data=data.veg, mean)
  #data.veg.avg_form.price =
    #data.veg.avg_form.price[order(data.veg.avg_form.price$CupEquivalentPrice),]
```

```
ggplot(data=data.veg.avg_form.price, aes(x=reorder(Form,-CupEquivalentPrice),
                                         y=CupEquivalentPrice)) +
  geom_bar(stat='identity', fill='salmon') + theme_minimal() +
  xlab('Vegetable Form') + ylab('Price per Cup') +
  geom_text(aes(label=sprintf('%.3f', CupEquivalentPrice)), vjust=1.6) +
  ggtitle('Average Price per Cup of Vegetable by Form')
```

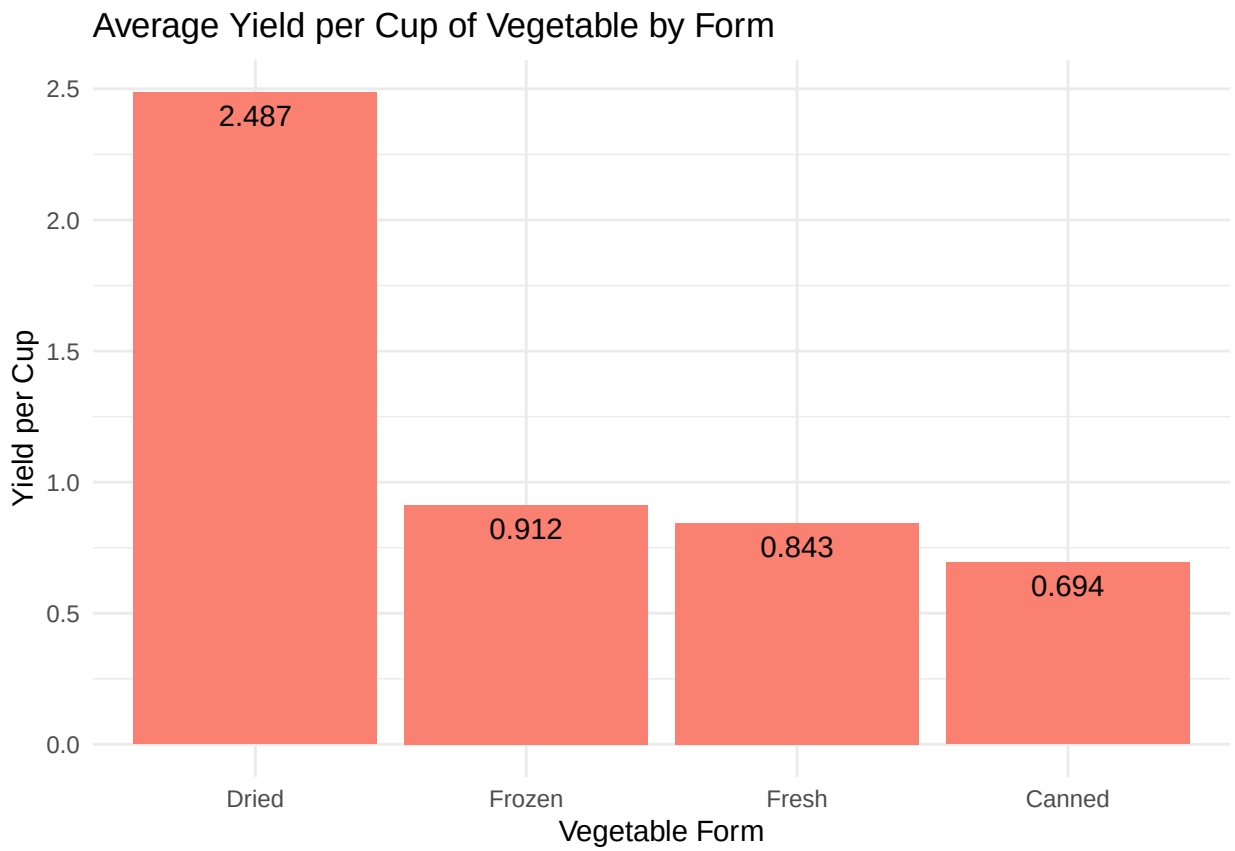



```
ggsave('visuals/veg_price_by_form.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
data.veg.avg_form.yield = aggregate(Yield~Form, data=data.veg, mean)
#data.veg.avg_form.yield =
  #data.veg.avg_form.yield[order(data.veg.avg_form.yield$Yield),]

ggplot(data=data.veg.avg_form.yield, aes(x=reorder(Form,-Yield),
                                         y=Yield)) +
  geom_bar(stat='identity', fill='salmon') + theme_minimal() +
  xlab('Vegetable Form') + ylab('Yield per Cup') +
  geom_text(aes(label=sprintf('%.3f', Yield)), vjust=1.6) +
  ggtitle('Average Yield per Cup of Vegetable by Form')
```

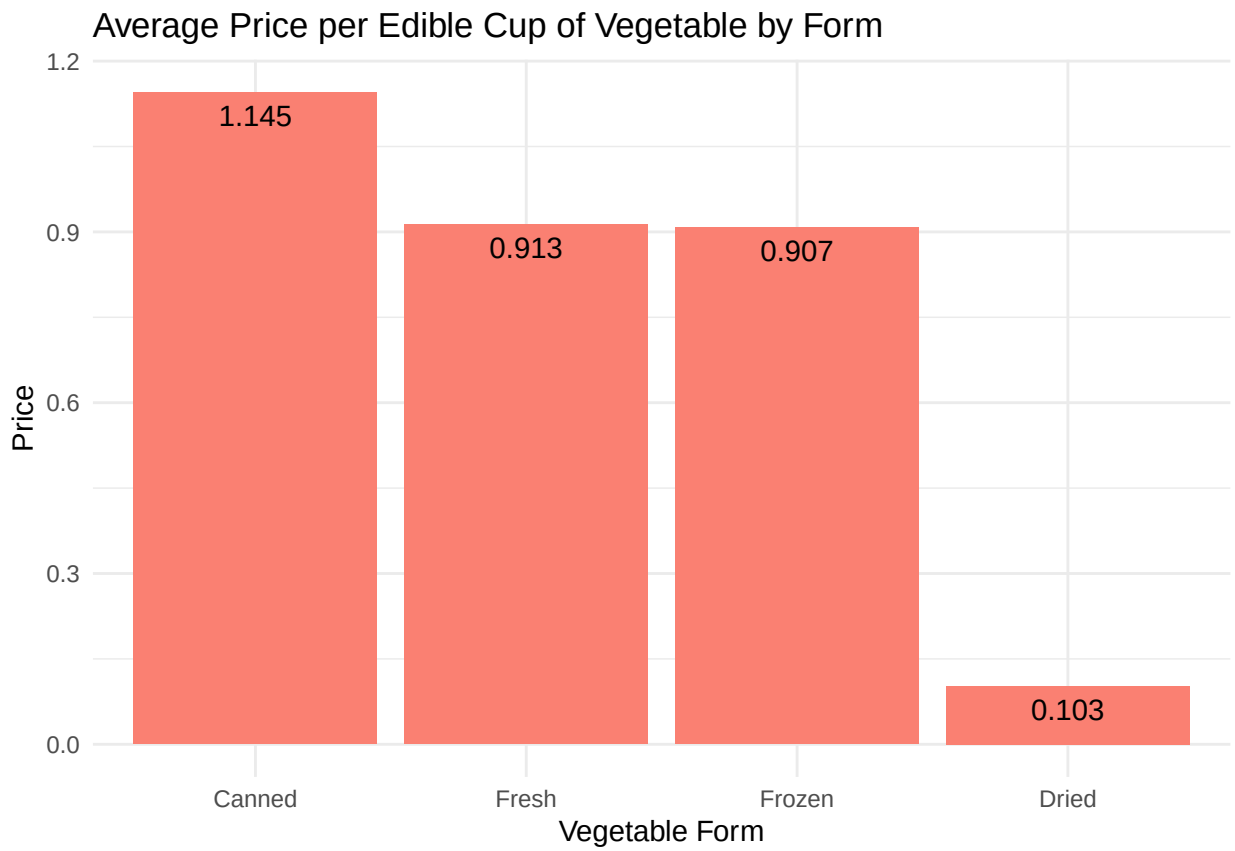



```
  ggsave('visuals/veg_yield_by_form.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

```
  data.veg.avg_form.avg_price_yield =
    data.frame(data.veg.avg_form.price$Form,
               data.veg.avg_form.price$CupEquivalentPrice /
                 data.veg.avg_form.yield$Yield)
  colnames(data.veg.avg_form.avg_price_yield) = c('Form','Price')

  ggplot(data=data.veg.avg_form.avg_price_yield,
         aes(x=reorder(Form,-Price), y=Price)) +
    geom_bar(stat='identity', fill='salmon') + theme_minimal() +
    xlab('Vegetable Form') + ylab('Price') +
    geom_text(aes(label=sprintf('%.3f', Price)), vjust=1.6) +
    ggtitle('Average Price per Edible Cup of Vegetable by Form')
```




```
ggsave('visuals/veg_price_per_yield.jpg')
```

```
## Saving 6.5 x 4.5 in image
```

Results

Cheapest form of vegetable by edible cup is canned. The most expensive form is frozen. The yield of dried is very high compard to the others but it becomes equalized when price per cup is involved. Once again, this does not consider calorie count!

Is a Tomato a fruit?

Determine if the natural clusters based on retail price, yield, and price per cup of fruits and vegetables is reflective of if they are a fruit or vegetable.

Methodology

```
set.seed(2024)

features = c('RetailPrice', 'Yield', 'CupEquivalentPrice')
training = select(data, all_of(features))
```

```
  set.seed(2024)

  # create the model
  model = kmeans(training, centers=2)

  # grab the model labels for the data
  data['cluster'] = model$cluster

  # kmeans labels = {1,2}. We want {0,1}
  data$cluster[data$cluster == 2] = 0

  confusion_matrix = table(data$isFruit, data$cluster)
  confusionMatrix(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 84  9
##   1 42 20
##
##                Accuracy : 0.671
##                  95% CI : (0.591, 0.7442)
##     No Information Rate : 0.8129
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2478
##
##  Mcnemar's Test P-Value : 7.433e-06
##
##             Sensitivity : 0.6667
##             Specificity : 0.6897
##          Pos Pred Value : 0.9032
##          Neg Pred Value : 0.3226
##              Prevalence : 0.8129
##          Detection Rate : 0.5419
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.6782
##
##        'Positive' Class : 0
##
```

```
  fig = plot_ly(data=data, x=~RetailPrice, y=~Yield, z=~CupEquivalentPrice,
                type='scatter3d', mode = 'markers', symbol=~I(isFruit), color=~cluster)
  fig = layout(fig, title='Clusters of Fruits and Vegetables Based on
         Price, Yield, and Price per Cup')
  fig
```

```
  data[grep('Tomato', data$Food),c('Food', 'isFruit', 'cluster')]
```

```
##                         Food isFruit cluster
## 86 Tomatoes, grape & cherry       0       1
```

```
## 87    Tomatoes, roma & plum       0       0
## 88    Tomatoes, large round       0       0
## 89                  Tomatoes      0       0
```

Results

We got an accuracy of `0.671`; however, the p-value was `1` so we cannot say we could accurately determine if a product was a fruit or vegetable. Interestingly enough, KMeans determined that grape and cherry tomatoes are fruits but any other type of tomato is a vegetable. I guess that settles the age old question: "is a tomato a fruit?".

Regression of Fruits and Vegetables

Create a logistic regression model to predict whether an item is a fruit or a vegetable

Input features: `Yield`, `CupEquivalentSize`, `CupEquivalentPrice`

Output: `Vegetable` or `Fruit`

Methodology

```
# create the train/test split

set.seed(2024)

# 10 fruit and 15 veg
num_fruit = 10
num_veg = 15

# Split the data into fruit and vegetable dataframes
data.fruit = filter(data, isFruit==1)
data.veg = filter(data, isFruit==0)

# generate the indicies for the test
test.fruit.ind = sample(1:length(data.fruit$Food), num_fruit)
test.veg.ind = sample(1:length(data.veg$Food), num_veg)

# grab the test items from each dataframe
test.fruit = data.fruit[test.fruit.ind,]
test.veg = data.veg[test.veg.ind,]

# gather the remaining items
train.fruit = data.fruit[-test.fruit.ind,]
train.veg = data.veg[-test.veg.ind,]

# combine the fruit and vegetable test/train dataframes
test = data.frame(rbind(test.fruit, test.veg))
train = data.frame(rbind(train.fruit, train.veg))

# shuffle the sets
test = test[sample(1:nrow(test)),]
```

```r
train = train[sample(1:nrow(train)),]

# check to make sure they are shuffled
head(train)
```

```
##                                        Food   Form RetailPrice RetailPriceUnit
## 7         Apricots, packed in syrup or water Canned      2.0600       per pound
## 23  Fruit cocktail, packed in syrup or water Canned      1.5932       per pound
## 10                            Berries, mixed Frozen      3.5585       per pound
## 361                                     Corn Canned      1.0287       per pound
## 311                            Celery sticks  Fresh      2.4041       per pound
## 37                                      Corn Frozen      1.6642       per pound
##     Yield CupEquivalentSize CupEquivalentUnit CupEquivalentPrice isFruit
## 7   0.650            0.4409            pounds              1.3974       1
## 23  0.650            0.4409            pounds              1.0808       1
## 10  1.000            0.3307            pounds              1.1768       1
## 361 0.650            0.3638            pounds              0.5757       0
## 311 1.000            0.2646            pounds              0.6360       0
## 37  0.963            0.3638            pounds              0.6286       0
##     cluster
## 7         0
## 23        0
## 10        1
## 361       0
## 311       0
## 37        0
```

Shuffling the subsets is redundant because there is no index feature of the dataset...

```r
# training the logistic model
log_model = glm(isFruit~Yield+CupEquivalentSize+CupEquivalentPrice,
                family = binomial, data = train)

summary(log_model)
```

```
##
## Call:
## glm(formula = isFruit ~ Yield + CupEquivalentSize + CupEquivalentPrice,
##     family = binomial, data = train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.9493     0.7459  -1.273   0.2031
## Yield               -0.6448     0.6012  -1.073   0.2835
## CupEquivalentSize    0.7598     0.5027   1.511   0.1307
## CupEquivalentPrice   0.8053     0.4323   1.863   0.0625 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 174.98  on 129  degrees of freedom
```

```
## Residual deviance: 151.35  on 126  degrees of freedom
## AIC: 159.35
##
## Number of Fisher Scoring iterations: 7
```

```
  # testing the model
  log_model.raw_prediction = predict(log_model, test, type="response")
  log_model.prediction = rep(0, nrow(test))
  log_model.prediction[log_model.raw_prediction > 0.5] = 1

  # confusion matrix
  log_model.classification_table = table(log_model.prediction, test$isFruit)
  log_model.cm = confusionMatrix(log_model.classification_table)

  log_model.cm
```

```
## Confusion Matrix and Statistics
##
##
## log_model.prediction  0  1
##                    0 13  6
##                    1  2  4
##
##                Accuracy : 0.68
##                  95% CI : (0.465, 0.8505)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.2735
##
##                   Kappa : 0.2857
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.8667
##             Specificity : 0.4000
##          Pos Pred Value : 0.6842
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6000
##          Detection Rate : 0.5200
##    Detection Prevalence : 0.7600
##       Balanced Accuracy : 0.6333
##
##        'Positive' Class : 0
##
```

Results

Using a 85/15 train/test split of the data, the logistic regression model had a prediction rate of `0.68`. However, the p-value is `0.2735`; therefore, we cannot confidently say that we can predict whether a product is a fruit of vegetable based on retail price, yield, and price per cup.