K Means Clustering

Will Doyle

Introduction

K means clustering is an example of *unsupervised learning*, a set of techniques used to identify patterns of association within a dataset that are not driven by the analyst. This technique is employed when it is strongly suspected that there are latent classifications of individuals in a dataset, but those classifications are unknown.

There are many types of unsupervised learning—this is a very active area of development in data science. K-means is among the simplest, and is relatively easy to explain. It's also pretty good— it tends to get decent answers. K-means proceeds by finding some number (K) groups of observations that are quite similar to one another, but quite different from other groups of observations. Similarity in this case is defined as having minimum variation within the group. The way this is done in practice is to start by randomly assigning each observation to a cluster, then to calculate the cluster centroid, which is the means of the variables used for the algorithm. Next, assign each observation to the cluster centroid which is closest to its means. This continues until no more changes are possible. If the data have clear underlying partitions, then the cluster assignment will be pretty stable. If not, then each time you run this algorithm, you could get different answers. There are solutions to this problem we'll go over, but please remember this basic fact about K-means clustering, which is different than any of the algorithms we cover in this class:

K MEANS CLUSTERING IN ITS BASIC FORM CAN GIVE YOU DIFFERENT ANSWERS EACH TIME YOUR RUN IT.

In our example today, we're going to take survey data from a group of passengers who were going through San Francisco international airport over the course of a week. The goal is to identify groups of passengers. Once we've identified these groups, we'll use group membership to predict a couple of outcomes: whether or not the passenger had a problem, and whether or not the passenger said they had an outstanding experience.

```r
rm(list = ls())
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ------------------------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts --------------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stats)
library(flexclust)
```

```
## Warning: package 'flexclust' was built under R version 3.5.3
```

```
## Loading required package: grid
```

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

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
library(ggplot2)
library(LICORS)
```

```
## Warning: package 'LICORS' was built under R version 3.5.3
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.5.3
```

```
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 3.5.3
```

The SFO survey data is pretty standard survey data, the kind you might get back from surveymonkey or qualtrics or other similar survey software.

First, I load the dataset containing individual characteristics. We'll load the dependent variables later.

```
load("sf_cluster.RData")
sfo <- sf_cluster %>% select(-sf.runid)
```

Determining the Number of Clusters

The first step in running cluster analysis is to figure out how many clusters are needed. It's generally assumed that there are at least 3 clusters, but it's not easy to think about how many more might be needed.

The `stepFlexClust` command can be helpful here. What it will do is to run a cluster analysis a certain number of times for a certain number of clusters, choosing the best fit (minimum distance) from each set of runs for each number of clusters. We can then take a look at the distances generated and plot them.

```
# Test to see how many clusters are needed
c_test <- stepFlexclust(sfo, k = 2:7, nrep = 20)
```

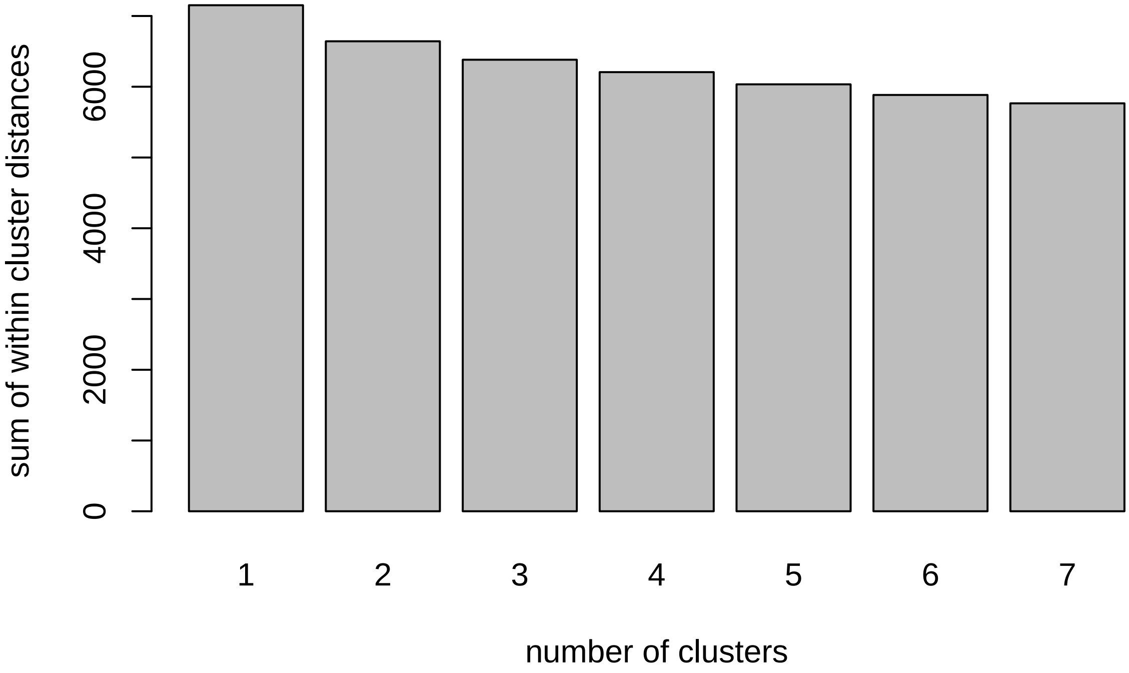
```
## 2 : * * * * * * * * * * * * * * * * * * * *
## 3 : * * * * * * * * * * * * * * * * * * * *
## 4 : * * * * * * * * * * * * * * * * * * * *
## 5 : * * * * * * * * * * * * * * * * * * * *
## 6 : * * * * * * * * * * * * * * * * * * * *
## 7 : * * * * * * * * * * * * * * * * * * * *
```

```
c_test
```

```
## stepFlexclust object of family 'kmeans'
##
## call:
## stepFlexclust(x = sfo, k = 2:7, nrep = 20)
```

```
##
##   iter converged  distsum
## 1   NA        NA 7152.029
## 2   10      TRUE 6642.461
## 3   11      TRUE 6381.596
## 4   11      TRUE 6206.289
## 5   12      TRUE 6034.030
## 6   15      TRUE 5883.865
## 7   14      TRUE 5765.531
```

```
plot(c_test)
```



It's not super clear where the cutoff should be for this data. The distances drop off by about 170 for a few steps, then decline to a drop of 150 or so. I'm going to choose 4 as the number of clusters, but it's fairly arbitrary. In general, a smaller number will result in a more tractable and stable solution.

Running K-Means Clustering

I'm going to run the `cclust` command twice, specifying that it should use 4 clusters, and the manhattan distance metric, which is scale-invariant.

```
# Cluster analysis
c1 <- cclust(sfo, k = 4, dist = "manhattan")
c1
```

```
## kcca object of family 'kmedians'
```

```
##
## call:
## cclust(x = sfo, k = 4, dist = "manhattan")
##
## cluster sizes:
##
##    1    2    3    4
##  893  870 1319  790
```

```r
## And again, see if results are stable
c2 <- cclust(sfo, k = 4, dist = "manhattan")
c2
```

```
## kcca object of family 'kmedians'
##
## call:
## cclust(x = sfo, k = 4, dist = "manhattan")
##
## cluster sizes:
##
##    1    2    3    4
## 1002  510 1248 1112
```

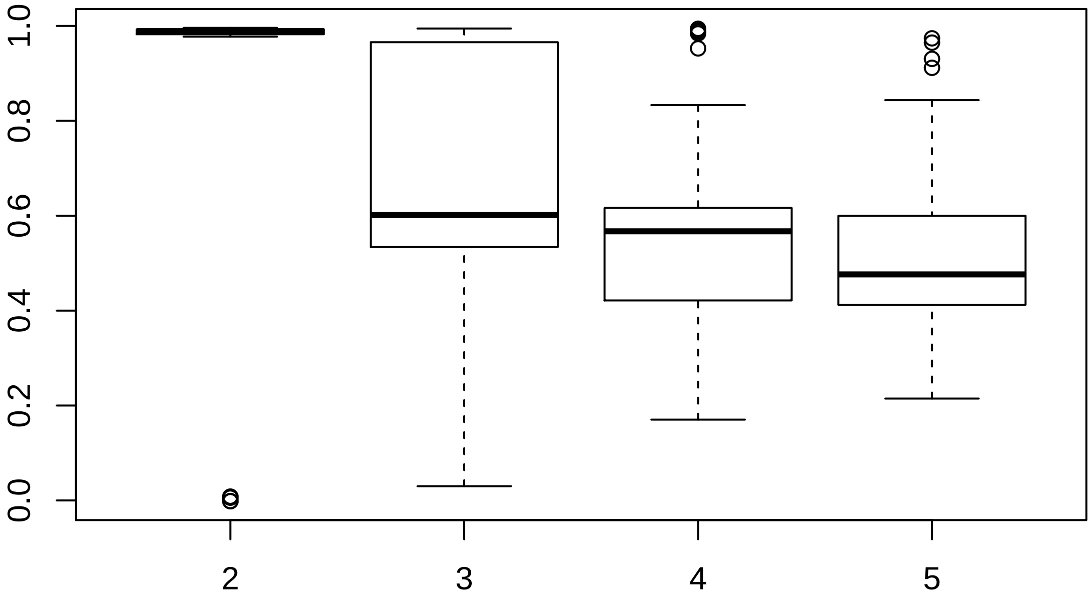
Hmm, this gave us WAY different answers. Here's why– the data aren't strongly separated by various groups, so it depends where the starting points were.

To visualize this, we'll use the `bootFlexclust` command, which will run the clustering algorithm on a subset of the data a certain number of times. This is called bootstrapping, and it shares many properties in common with cross-fold validation from last time. I'm going to do this for four different numbers of groups, from 2 through 5, then visualize the results.

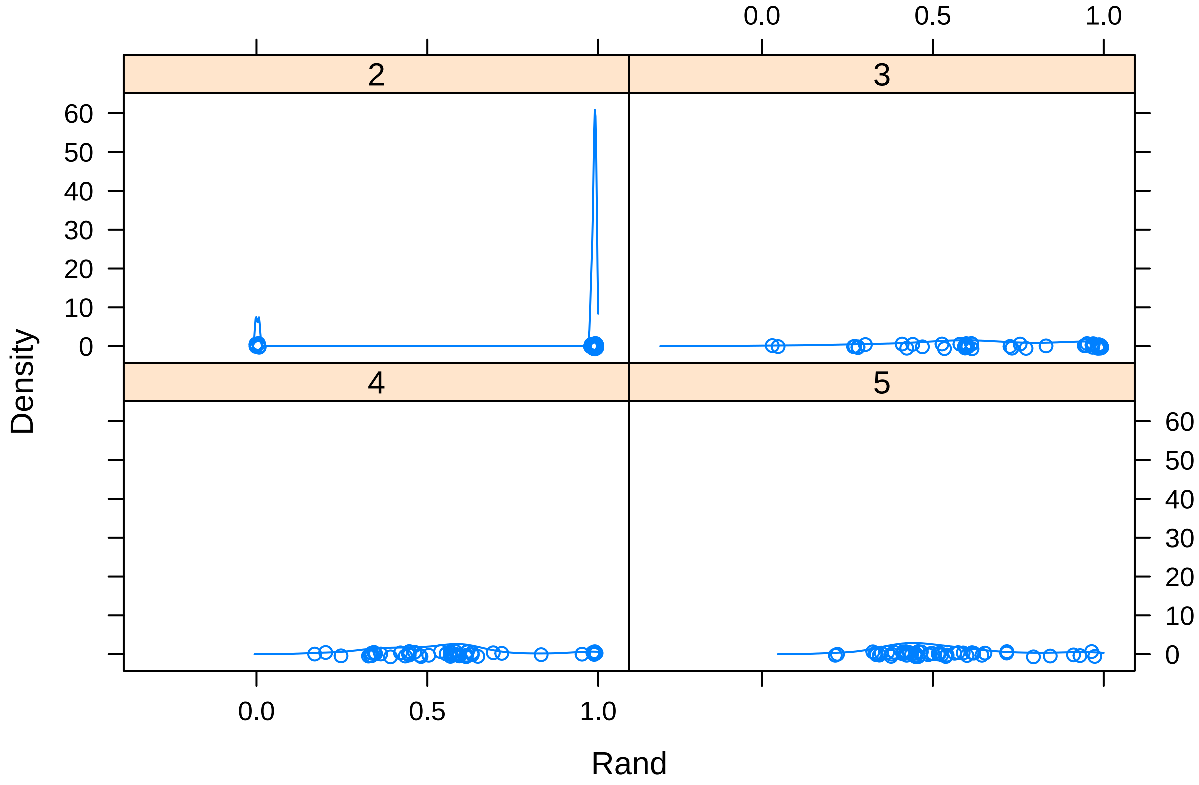
```r
c1a <- bootFlexclust(x = sfo, k = c(2, 3, 4, 5), nboot = 50, verbose = TRUE)
```

```
## 10 20 30 40 50
```

```r
plot(c1a)
```



```
densityplot(c1a)
```



With two groups, the cluster locations are pretty stable. At 3 they spread out. With 4 and five there appear to be some relatively steady points, but still it's pretty spread out.

Okay, so that's no good. What to do? Well, one thing to do is to apply this to a dataset with clearer groups. But, we're going to plow on regardless. If we really want to find stable groupings, we need to find a way to converge on a stable classification into groups.

The `kmeanspp` (stands for k-means ++) command will repeat the kmeans clustering algorithm with different starting points until it converges on a stable solution. It basically repeats the process we saw above, but with the intention of getting to a stable solution.

```
c2 <- kmeanspp(sfo, k = 4, start = "random", iter.max = 1000, nstart = 50)
table(c2$cluster)
```

```
##
##    1    2    3    4
## 1328  642  877 1025
```

Notice how the sample sizes in each group are identical, although the group numbers (which are arbitrary) are different after each run.

Understanding cluster assignments

So now what? We need to figure out what these clusters mean by inspecting them as a function of the constituent variables.

```
# Add predictions
sfo$cluster <- c2$cluster

# Examine relationship of cluster assignment to chars.
mycols <- c("Group 1", "Group 2", "Group 3", "Group 4")

# Frequency of travel
pt <- prop.table(table(sfo$cluster, sfo$sf.freq), margin = 1)
```

To create a nice table, we need to generate both row names and column names. We'll also convert to percentages.

```
rownames(pt) <- mycols
pt <- pt * 100
pt <- round(pt, 1)
```

The `kable` command will give us a nice looking table.

Group membership by frequent traveler status

```
kable(pt, row.names = TRUE, col.names = c("Not a frequent traveler", "Frequent traveler"))
```

	Not a frequent traveler	Frequent traveler
Group 1	99.8	0.2
Group 2	94.1	5.9
Group 3	87.2	12.8
Group 4	97.0	3.0

It looks like one group has more frequent travelers than the others.

```
pt <- prop.table(table(sfo$cluster, sfo$sf.mod_travel), margin = 1)

rownames(pt) <- mycols
pt <- pt * 100
pt <- round(pt, 1)
```

Group membership by moderate traveler status

```
kable(pt, row.names = TRUE, col.names = c("Not a moderate traveler", "Moderate traveler"))
```

	Not a moderate traveler	Moderate traveler
Group 1	0.7	99.3
Group 2	100.0	0.0
Group 3	13.8	86.2
Group 4	100.0	0.0

That samge group also has many more moderate travelers.

```r
# Age
prop.table(table(sfo$cluster, sfo$sf.age2544), margin = 1)
```

```
##
##             0         1
##   1 0.6596386 0.3403614
##   2 0.0000000 1.0000000
##   3 0.5131129 0.4868871
##   4 1.0000000 0.0000000
```

```r
prop.table(table(sfo$cluster, sfo$sf.age4564), margin = 1)
```

```
##
##             0         1
##   1 0.6686747 0.3313253
##   2 1.0000000 0.0000000
##   3 0.5746864 0.4253136
##   4 0.4419512 0.5580488
```

```r
# Reason for travel
pt <- prop.table(table(sfo$cluster, sfo$sf.business), margin = 1)

rownames(pt) <- mycols
pt <- pt * 100
pt <- round(pt, 1)
```

Group membership by reason for travel

```r
kable(pt, row.names = TRUE, col.names = c("Not a business traveler", "Business traveler"))
```

	Not a business traveler	Business traveler
Group 1	100.0	0.0
Group 2	69.9	30.1
Group 3	0.8	99.2
Group 4	82.0	18.0

Now we're getting somehwere: One group is overwhelmingly composed of business travlers, and is much more likely to include frequent or moderate travelers.

Let's look at income:

```r
# Income
prop.table(table(sfo$cluster, sfo$sf.inc150p), margin = 1) %>% kable()
```

0	1
0.8426205	0.1573795
0.8457944	0.1542056
0.6168757	0.3831243
0.8478049	0.1521951

And now at whether they checked their bags

```
# Checked bags?
prop.table(table(sfo$cluster, sfo$sf.bags), margin = 1) %>% kable()
```

0	1
0.4307229	0.5692771
0.4330218	0.5669782
0.5678449	0.4321551
0.3834146	0.6165854
The clearest	separation by far is the business travel difference, with some differences by frequency of travel as well.

```
# summarize groups

var.means <- colMeans(sfo)

## Drop the cluster means
var.means <- var.means[-(length(var.means))]
```

Summarizing group characteristics by cluster

It's really important to figure out what the clusters look like. The code below will summarize all of the contributing variables for each cluster, then create a plot that shows the mean of each variable within each cluster.

First, we do our normal `summarize` command, but this time over every variable in the analysis dataset, using `summarize_all` and `funs(mean)`.

```
# Summarize groups by cluster
sum1 <- sfo %>% group_by(cluster) %>% summarize_all(funs(mean), na.rm = TRUE)
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
##   # Before:
##   funs(name = f(.))
##
##   # After:
##   list(name = ~ f(.))
## This warning is displayed once per session.
```

```
sum1
```

```
## # A tibble: 4 x 19
##   cluster sf.business sf.pleasure sf.relatives sf.infreq sf.mod_travel
##     <int>       <dbl>       <dbl>        <dbl>     <dbl>         <dbl>
## 1       1       0           0.508        0.356     0             0.993
## 2       2       0.301       0.450        0.151     0.927         0
## 3       3       0.992       0            0         0             0.862
## 4       4       0.180       0.467        0.232     0.958         0
## # ... with 13 more variables: sf.freq <dbl>, sf.bags <dbl>, sf.male <dbl>,
## #   sf.inc50less <dbl>, sf.inc50100 <dbl>, sf.inc100150 <dbl>,
## #   sf.inc150p <dbl>, sf.age24less <dbl>, sf.age2544 <dbl>,
## #   sf.age4564 <dbl>, sf.age65p <dbl>, sf.purchase <dbl>, sf.resta <dbl>
```

Now we'll **gather** everything, so that we have just two variables: the proportion (as a number) and the variable (as a character).

```
sum2 <- gather(sum1, -cluster, key = variable, value = value)
sum2
```

```
## # A tibble: 72 x 3
##    cluster variable     value
##      <int> <chr>        <dbl>
##  1       1 sf.business  0
##  2       2 sf.business  0.301
##  3       3 sf.business  0.992
##  4       4 sf.business  0.180
##  5       1 sf.pleasure  0.508
##  6       2 sf.pleasure  0.450
##  7       3 sf.pleasure  0
##  8       4 sf.pleasure  0.467
##  9       1 sf.relatives 0.356
## 10       2 sf.relatives 0.151
## # ... with 62 more rows
```

```
sum_total <- sfo %>% summarize_all(funs(mean))

sum_total <- gather(sum_total, key = variable, value = overall_mean)
sum2 <- left_join(sum2, sum_total, by = "variable")
```

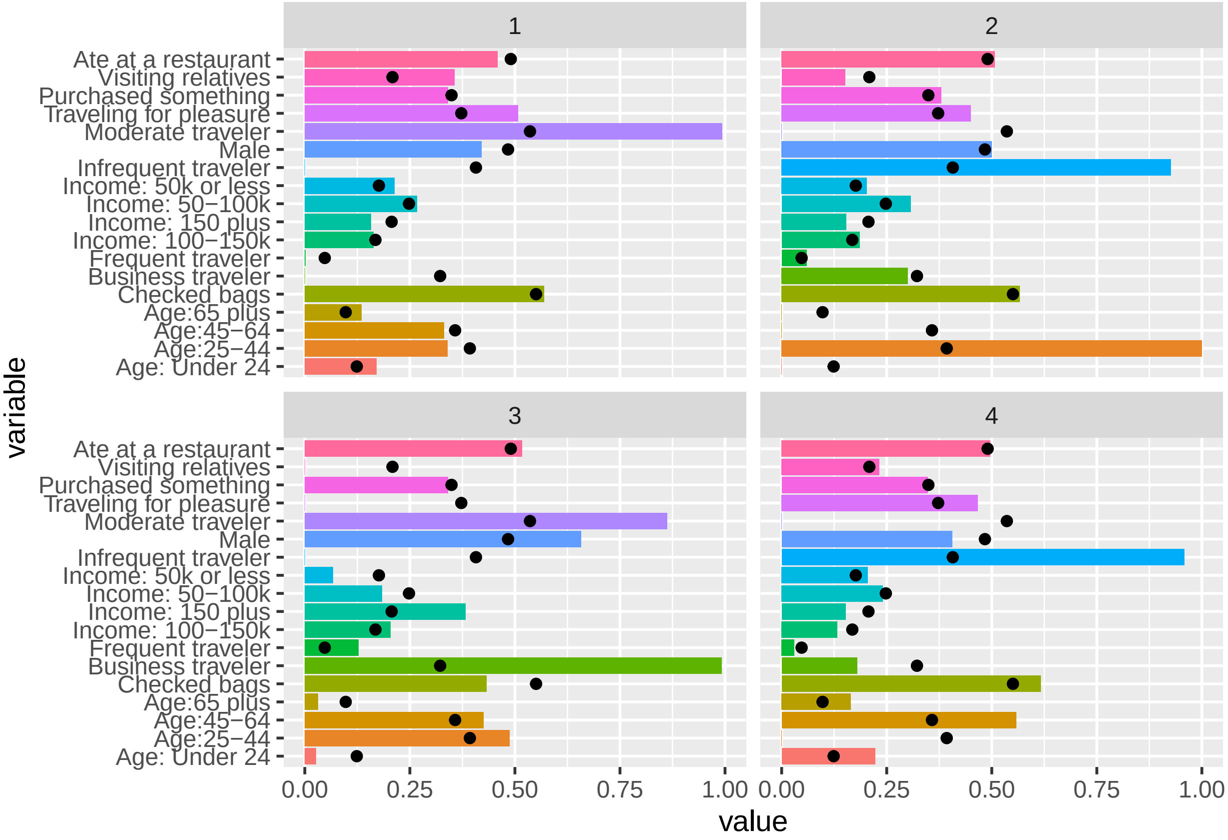
```
sum2 <- sum2 %>% group_by(variable) %>% mutate(varmeans = mean(value))
```

Survey Responses by Cluster

```
variable_labels <- rev(c(
  "Ate at a restaurant",
  "Visiting relatives",
  "Purchased something",
  "Traveling for pleasure",
  "Moderate traveler",
  "Male",
  "Infrequent traveler",
  "Income: 50k or less",
  "Income: 50-100k",
  "Income: 150 plus",
  "Income: 100-150k",
  "Frequent traveler",
  "Business traveler",
  "Checked bags",
  "Age:65 plus",
  "Age:45-64",
  "Age:25-44",
  "Age: Under 24"
))

# Plot characteristics by cluster
g1 <- ggplot(data = sum2, aes(y = value, x = variable, fill = variable))
g1 <- g1 + geom_bar(stat = "identity") + coord_flip() + theme(legend.position = "none")
```

```
g1 <- g1 + scale_x_discrete(labels = variable_labels)
g1 <- g1 + geom_point(data = sum2, aes(y = overall_mean, x = variable))
g1 <- g1 + facet_wrap(~cluster)
g1
```



Survey responses by classification

```
sfo$cluster_name <- NA
sfo$cluster_name[sfo$cluster == 1] <- "Business Traveler"
sfo$cluster_name[sfo$cluster == 2] <- "Vacationers"
sfo$cluster_name[sfo$cluster == 3] <- "Golden Years"
sfo$cluster_name[sfo$cluster == 4] <- "Aunt Sally Visiting the Kids"
```

Quick Exercise: Name these clusters for me

Modeling Using Clusters

Once you have clusters, then you can use these as independent variables to predict various outcomes. For instance, in our data, which clusters are likely to have problems in the airport? We'll use a logistic regression to see which groups are more likely to report having a problem.

```r
load("sf_complete.RDAta")
sf <- sf %>% select(hadproblem, outstanding)
sf <- bind_cols(sf, sfo)
```

```r
# Modeling: figure out which clusters are associated with different outcomes
# Did you have a problem?
sf$cluster <- c2$cluster

mod1 <- glm(hadproblem ~ as.factor(cluster), data = sf, family = binomial(link = "logit"))
summary(mod1)
```

```
##
## Call:
## glm(formula = hadproblem ~ as.factor(cluster), family = binomial(link = "logit"),
##     data = sf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5712  -0.4765  -0.4720  -0.4658   2.1330
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.13838    0.08935 -23.932  < 2e-16 ***
## as.factor(cluster)2 -0.02807    0.15770  -0.178  0.85873
## as.factor(cluster)3  0.40780    0.13001   3.137  0.00171 **
## as.factor(cluster)4  0.01994    0.13478   0.148  0.88239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2774.5  on 3871  degrees of freedom
## Residual deviance: 2761.7  on 3868  degrees of freedom
## AIC: 2769.7
##
## Number of Fisher Scoring iterations: 4
```

```r
# Create basic hypothetical dataset
hypo_data <- sf %>% data_grid(.model = mod1, cluster = 1:4)

pred1 <- predict(mod1, type = "response", newdata = hypo_data)

pred1 <- sort(pred1, decreasing = FALSE)

pred1 <- pred1 * 100
```

What this shows is that 10 percent of this group is predicted to have a problem– likely a group to focus on for improved customer service.

Or, on the other hand, which group might be more likely to rate the experience as outstanding?

```r
# Overall rating=outstanding
mod2 <- glm(outstanding ~ as.factor(cluster), data = sf, family = binomial(link = "logit"))
summary(mod2)
```

```
##
```

```
## Call:
## glm(formula = outstanding ~ as.factor(cluster), family = binomial(link = "logit"),
##     data = sf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7391  -0.7278  -0.6463  -0.5712   1.9461
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.19324    0.06494 -18.374  < 2e-16 ***
## as.factor(cluster)2 -0.26672    0.12001  -2.223   0.0262 *
## as.factor(cluster)3 -0.53735    0.11461  -4.688 2.75e-06 ***
## as.factor(cluster)4  0.03520    0.09788   0.360   0.7191
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3963.8  on 3871  degrees of freedom
## Residual deviance: 3932.7  on 3868  degrees of freedom
## AIC: 3940.7
##
## Number of Fisher Scoring iterations: 4
```

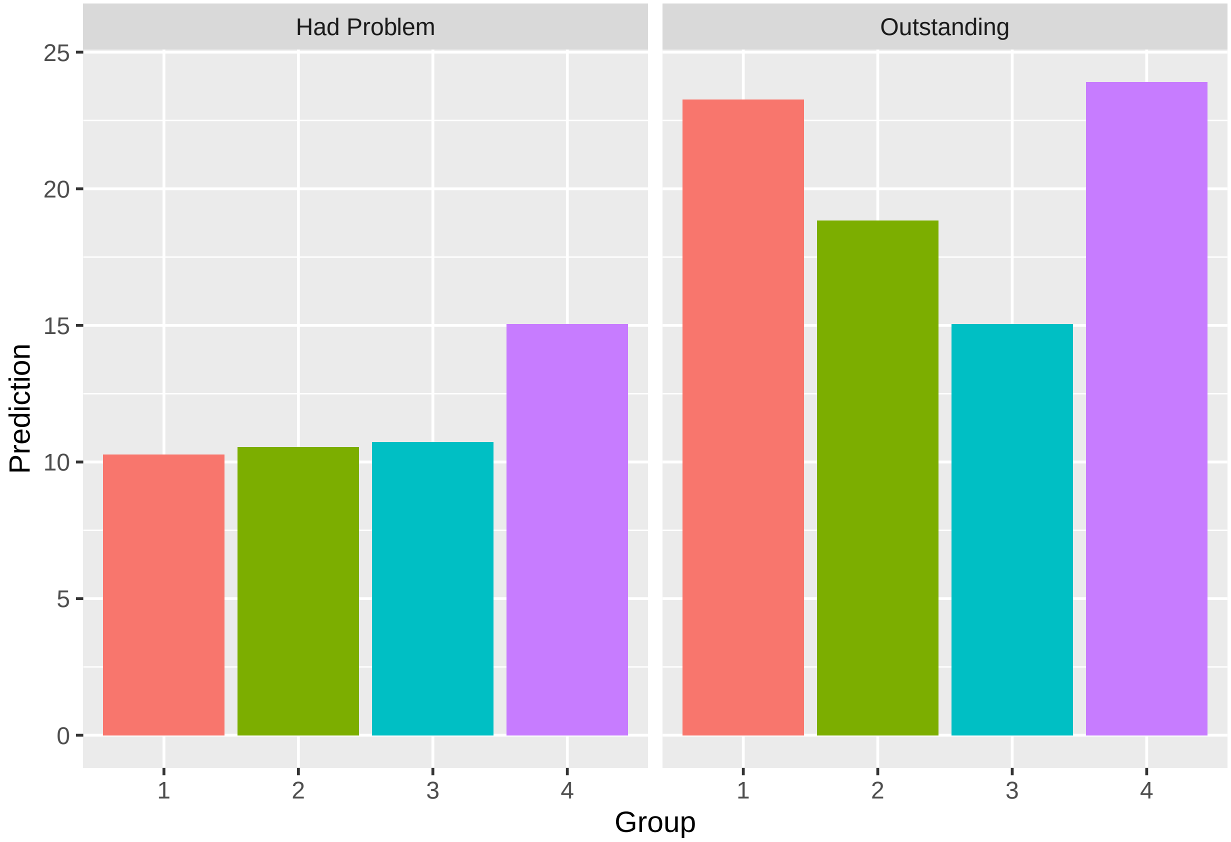
```
pred2 <- (predict(mod2, type = "response", newdata = hypo_data)) * 100

pred_data <- data.frame(
  c((1:4), (1:4)), c(rep("Had Problem", 4), rep("Outstanding", 4)),
  c(pred1, pred2)
)

names(pred_data) <- c("Cluster", "Outcome", "Prediction")
```

Predicted probability of Outcomes by Group Membership

```
# Plot predictions from two different outcome models
g1 <- ggplot(pred_data, aes(x = as.factor(Cluster), y = Prediction, fill = as.factor(Cluster)))
g1 <- g1 + geom_bar(stat = "identity")
g1 <- g1 + facet_wrap(~Outcome)
g1 <- g1 + theme(legend.position = "none") + xlab("Group")
g1
```



The plot shows which groups are predicted to report a problem, and which ones are more likely to say they had an outstanding experiences. Does this make sense?