# Final Project Script

Christian Conroy & Lawrence Doppelt

May 6, 2018

## The Data

For this project, we use data provided by Yelp for the Yelp Dataset challenge, which can be found in JSON format from this website:

https://www.yelp.com/dataset/challenge

Documentation describing the variables and information contained in each json file comprising the yelp challenge dataset can be found here:

https://www.yelp.com/dataset/documentation/json

We then merge it with Zillow data on rental values across the United States by postal code. The Zillow rental value dataset can be downloaded from this website:

https://www.zillow.com/research/data/

## Setting the Working Directory and Loading in Required Packages

## Importing the data

First, we import the Yelp Check-In data and flatten it.

```
# Stream in Check-In Data
yelp_checkin <-
as.data.frame(jsonlite::stream_in(file("dataset/checkin.json")), flatten =
TRUE)
# Flatten Check-In Data
renquote <- function(l) if (is.list(l)) lapply(l, renquote) else enquote(l)
yelp_checkin_flat <- as.data.frame(lapply(unlist(renquote(yelp_checkin)),
eval))
```

## Reshaping the Data

We clean the time period variable names by collapsing the data to long form and using string functions to isolate the name of the weekday in string format.

```
# Convert from wide to long
yelp_checkin_flat_long <- reshape(yelp_checkin_flat, varying =
list(names(yelp_checkin_flat[1:168])), times =
names(yelp_checkin_flat[1:168]), idvar = 'business_id', v.names = 'checkin' ,
direction = 'long')
```

```
# Eliminate punctuation and digits
yelp_checkin_flat_long$time <- str_replace(yelp_checkin_flat_long$time,
"time.", "")
yelp_checkin_flat_long$time <- gsub('[[:digit:]]+', '',
yelp_checkin_flat_long$time)

# Isolate name of weekday
yelp_checkin_flat_long$time =
substr(yelp_checkin_flat_long$time,1,nchar(yelp_checkin_flat_long$time)-2)
```

Here, we collapse the data, reshape it from long to wide, then merge them together.

```
# Aggregate Check-In by business and time period to get Check-In average and
total by business for each day of the week.
yelp_checkin_collapse_mean <- as.data.frame(aggregate(checkin ~ business_id +
time, yelp_checkin_flat_long , mean))
yelp_checkin_collapse_sum <- as.data.frame(aggregate(checkin ~ business_id +
time, yelp_checkin_flat_long , sum))

# Convert from long to wide
yelp_checkin_wide_mean <- spread(yelp_checkin_collapse_mean, key = time,
value = checkin)
yelp_checkin_wide_sum <- spread(yelp_checkin_collapse_sum, key = time, value
= checkin)

# Merge averages and totals
yelp_checkin_wide <- inner_join(yelp_checkin_wide_mean,
yelp_checkin_wide_sum, by='business_id', match='all')
colnames(yelp_checkin_wide) <- c("business_id", "Friday_ave", "Monday_ave",
"Saturday_ave", "Sunday_ave", "Thursday_ave", "Tuesday_ave", "Wednesday_ave",
"Friday_total", "Monday_total", "Saturday_total", "Sunday_total",
"Thursday_total", "Tuesday_total", "Wednesday_total")
```

## Loading in Business Dataset to Merge with Check-In Data and Aggregate by Zip Code. Eliminate Useless Columns and Merge.

```
# Import business dataset
yelp_business <- fromJSON(sprintf("[%s]",
paste(readLines("dataset/business.json"), collapse=",")),
simplifyDataFrame=TRUE, flatten=TRUE)

# Merge check-in data with business data by business
checkinbiz <- inner_join(yelp_business, yelp_checkin_wide,
by=c('business_id'), match='all')

# Eliminate unnecessary columns
checkinbiz <- checkinbiz[-c(2:6, 8:101)]

# Collapse check-in data by zipcode to get total and average check-ins for
```

```
each weekday and zipcode
checkinzipmean <- as.data.frame(aggregate(. ~ postal_code, checkinbiz[2:9],
mean))
checkinzipsum <- as.data.frame(aggregate(. ~ postal_code, checkinbiz[c(2,
10:16)], sum))

checkinfull <- inner_join(checkinzipmean, checkinzipsum, by=c('postal_code'),
match='all')
```

## Importing the Yelp Review Data

Due to the large size of the Yelp Review JSON file, The Yelp Review dataset was collapsed to the zipcode level through merging, aggregation, and collapsing within the google cloud platform.

```
# Import data
yelp_review_long <- read.csv("yelplongPC_updated2.csv", header = T,
na.strings=c("NA"))

# Check missingness
sapply(yelp_review_long, function(x) sum(is.na(x)))

##                    X          postal_code            YearMonth
##                    0                    0                    0
## Number_of_businesses    Number_of_reviews             starsav
##                    0                    0                    0
##              starssd             usefulav             funnyav
##               168953                    0                    0
##               coolav             bizstars          bizstarssd
##                    0                    0               168953
##          bizrevcount             bizrevav          is_openave
##                    0                    0                    0

# Check how many unique zipcodes
length(unique(yelp_review_long$postal_code))

## [1] 15890

# Convert YearMonth from character to yearmon class
yelp_review_long$YearMonth <- as.yearmon(yelp_review_long$YearMonth)
```

Notice that there are no missing postal code values, with 15,980 unique postal codes.

## Zillow data

Here, we read in the Zillow data with information on all rental values across the US and Canada. We rename the “RegionName”" variable to “postal_code” then convert the data from wide to long in order to merge it with the Yelp dataset. We change the “time” variable to a date class and cut the Zillow data to match the dates of the Yelp data (while Zillow data goes back to the 1990s, Yelp business and review data only go back to 2010).

```
# Import data
zillow <- read_csv("zecon/Zip_Zri_AllHomesPlusMultifamily.csv", col_names =
TRUE)

# Reformat to prepare for merge with Yelp dataset.
names (zillow)[2] <- "postal_code"
zillow <- as.data.frame(zillow)
zillow_long <- reshape(zillow, varying = list(names(zillow[8:95])), times =
names(zillow[8:95]), idvar = 'postal_code', v.names = 'rentprice' , direction
= 'long')

sapply(zillow_long, function(x) sum(is.na(x)))

##    RegionID postal_code        City       State       Metro  CountyName
##           0           0           0           0      113960           0
##    SizeRank        time   rentprice
##           0           0       28354

zillow_long$time<- as.Date(strptime(paste(1, zillow_long$time),"%d %Y-%m"))

zillow_long <- zillow_long[zillow_long$time >= "2010-11-01" &
zillow_long$time <= "2017-12-31",]

zillow_long$month <- match(months(zillow_long$time), month.name)
zillow_long$year <- format(zillow_long$time,format="%Y")
zillow_long$YearMonth <- as.yearmon(paste(zillow_long$year,
zillow_long$month), "%Y %m")
names(zillow_long)

##  [1] "RegionID"    "postal_code" "City"        "State"       "Metro"
##  [6] "CountyName"  "SizeRank"    "time"        "rentprice"   "month"
## [11] "year"        "YearMonth"
```

Notice that there are 28,354 missing values for “rentprice”. We will attempt to recitfy this through imputation later.

## Merging all datasets

Here, we create the official merged dataset from which we conduct the analysis.

```
# Merge Check-In with Yelp
yelp_review_long <- left_join(yelp_review_long, checkinfull,
by=c('postal_code'), match='all')

# Merge Zillow with Yelp
Full_data_long <- inner_join(yelp_review_long, zillow_long,
by=c('postal_code', 'YearMonth'), match='all')
length(unique(Full_data_long$postal_code))

## [1] 564
```

```
sapply(Full_data_long, function(x) sum(is.na(x)))
```

```
##                    X          postal_code            YearMonth 
##                    0                    0                    0 
## Number_of_businesses    Number_of_reviews              starsav 
##                    0                    0                    0 
##              starssd             usefulav              funnyav 
##                 2977                    0                    0 
##               coolav             bizstars           bizstarssd 
##                    0                    0                 2977 
##          bizrevcount             bizrevav           is_openave 
##                    0                    0                    0 
##           Friday_ave           Monday_ave         Saturday_ave 
##                 1337                 1337                 1337 
##           Sunday_ave         Thursday_ave          Tuesday_ave 
##                 1337                 1337                 1337 
##        Wednesday_ave         Friday_total         Monday_total 
##                 1337                 1337                 1337 
##       Saturday_total         Sunday_total       Thursday_total 
##                 1337                 1337                 1337 
##        Tuesday_total      Wednesday_total             RegionID 
##                 1337                 1337                    0 
##                 City                State                Metro 
##                    0                    0                   64 
##           CountyName             SizeRank                 time 
##                    0                    0                    0 
##            rentprice                month                 year 
##                  580                    0                    0
```

The final frame consists of 37492 observations and 39 variables.

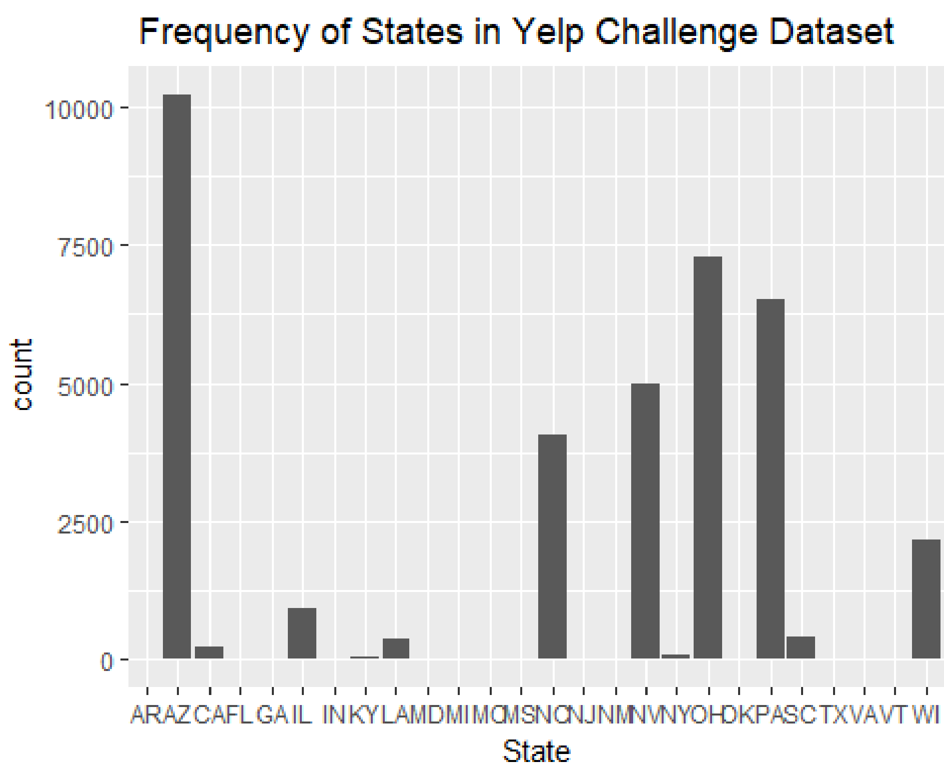
## Explortatory Data Analysis

Here is where we conduct the official analysis portion of the project

```
# First, subset business data based on eventual merging with Zillow
Full <- unique(Full_data_long$postal_code)
business_zillow <- dplyr::filter(yelp_business, postal_code %in% Full)

# Assess the count of states
Full_data_long$State <- Full_data_long$State %>% as.factor
Full_data_long$State %>% summary
```

```
##    AR    AZ    CA    FL    GA    IL    IN    KY    LA    MD    MI    MO 
##    22 10234   242     4    17   925    15    44   394     6     3     8 
##    MS    NC    NJ    NM    NV    NY    OH    OK    PA    SC    TX    VA 
##     4  4088     7     3  4981    75  7301     3  6510   405    17     6 
##    VT    WI 
##     7  2171
```

```
ggplot(Full_data_long, aes(State)) + geom_bar() + labs(title= " Frequency of
States in Yelp Challenge Dataset")
```

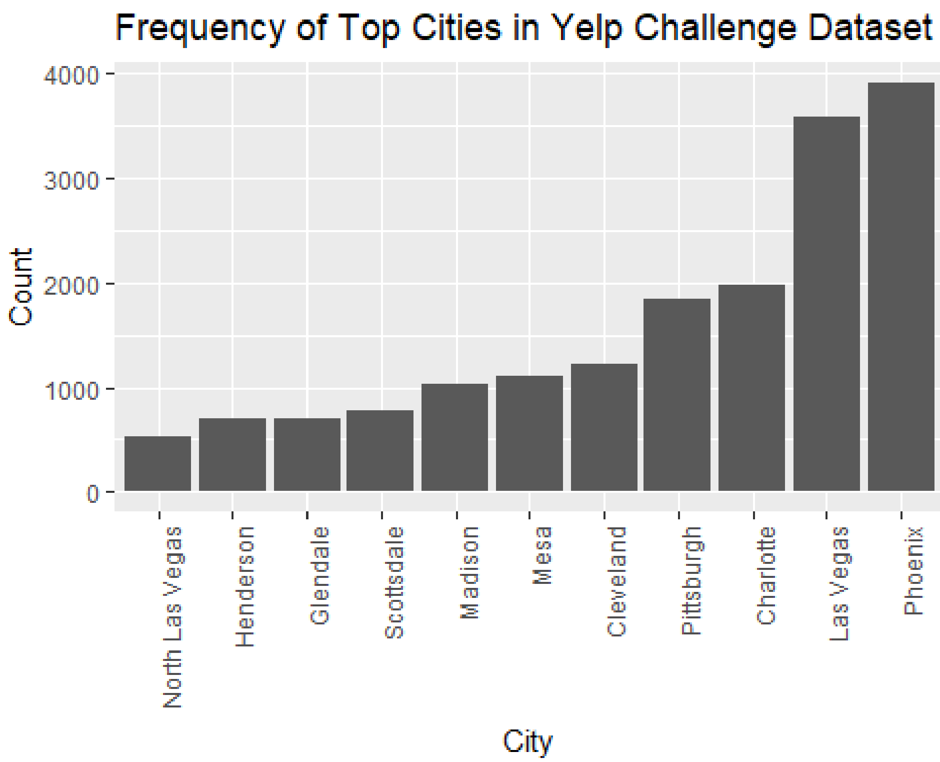



The states most represented in the dataset are Arizona, Ohio, Pennsylvania, Nevada, North Carolina, Wisconsin, and Illinois.

```
Full_data_long$City <- Full_data_long$City %>% as.factor
Full_data_long$City %>% summary

##          Phoenix        Las Vegas        Charlotte       Pittsburgh
##             3918             3580             1972             1847
##        Cleveland             Mesa          Madison       Scottsdale
##             1232             1115             1021              774
##        Glendale        Henderson  North Las Vegas         Chandler
##              695              691              522              516
##          Gilbert         Surprise           Peoria            Tempe
##              516              388              344              344
##        Champaign       Fort Mill         Gastonia           Shaler
##              258              256              251              237
##           Euclid           Lorain         Avondale          Concord
##              206              192              172              172
##   Cuyahoga Falls         Goodyear            Parma           Urbana
##              172              172              172              172
##           O'Hara             Ross    Chagrin Falls       Penn Hills
##              170              168              166              166
##         Sun City     Strongsville       Kannapolis      Queen Creek
##              159              158              156              119
```

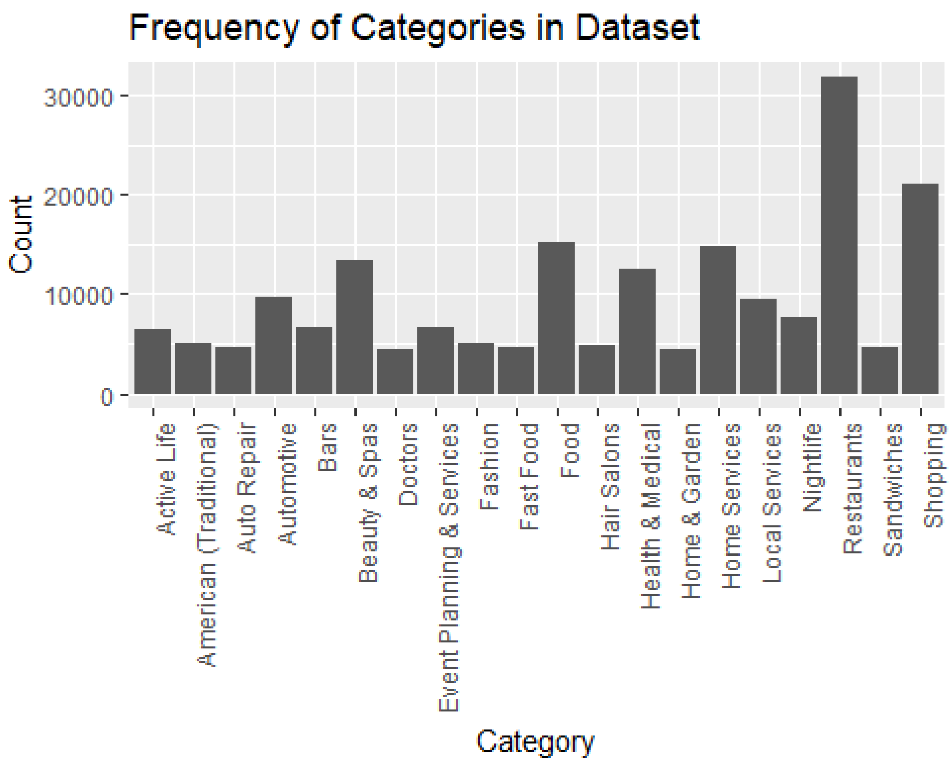
```
##            Grafton            Amherst             Anthem               Avon 
##                100                 86                 86                 86 
##            Bedford           Bellevue            Belmont              Berea 
##                 86                 86                 86                 86 
##        Bethel Park       Boulder City        Brecksville Broadview Heights 
##                 86                 86                 86                 86 
##         Brook Park           Carefree           Carnegie     Castle Shannon 
##                 86                 86                 86                 86 
## Cleveland Heights          Cornelius           Davidson             Elyria 
##                 86                 86                 86                 86 
##           Fairlawn      Fairview Park     Fountain Hills  Garfield Heights 
##                 86                 86                 86                 86 
##         Harrisburg             Hudson       Huntersville       Indian Trail 
##                 86                 86                 86                 86 
##               Kent           Lakewood    Litchfield Park          Lyndhurst 
##                 86                 86                 86                 86 
##            Malvern           Matthews             Medina             Mentor 
##                 86                 86                 86                 86 
##          Middleton          Mint Hill        Monroeville               Moon 
##                 86                 86                 86                 86 
##  Mount Charleston            Munhall         New Iberia  North Huntingdon 
##                 86                 86                 86                 86 
##      North Olmsted  North Ridgeville     North Strabane         Northfield 
##                 86                 86                 86                 86 
##    Paradise Valley      Parma Heights         Pineville         Richfield 
##                 86                 86                 86                 86 
##  Richmond Heights        Rocky River              Savoy        Seven Hills 
##                 86                 86                 86                 86 
##     Shaker Heights              Solon       South Euclid          Stallings 
##                 86                 86                 86                 86 
##               Stow        Streetsboro        Sun Prairie            (Other) 
##                 86                 86                 86               8891
```

```
temp <- row.names(as.data.frame(summary(Full_data_long$City, max=12)))
Full_data_long$City <- as.character(Full_data_long$City)
Full_data_long$top <- ifelse(
  Full_data_long$City %in% temp,
  Full_data_long$City,
  "Other"
)
Full_data_long$top <- as.factor(Full_data_long$top)

ggplot(Full_data_long[Full_data_long$top!="Other",],aes(x=factor(top,
levels=names(sort(table(top),increasing=TRUE))))) + geom_bar() +
labs(title="Frequency of Top Cities in Yelp Challenge Dataset") +
xlab("City") + ylab("Count") + theme(axis.text.x = element_text(angle = 90,
hjust = 1))
```




This assesses the count of cities in the dataset. Most frequent cities with Yelp information include Phoenix, Las Vegas, Charlotte, Pittsburgh, Cleveland.

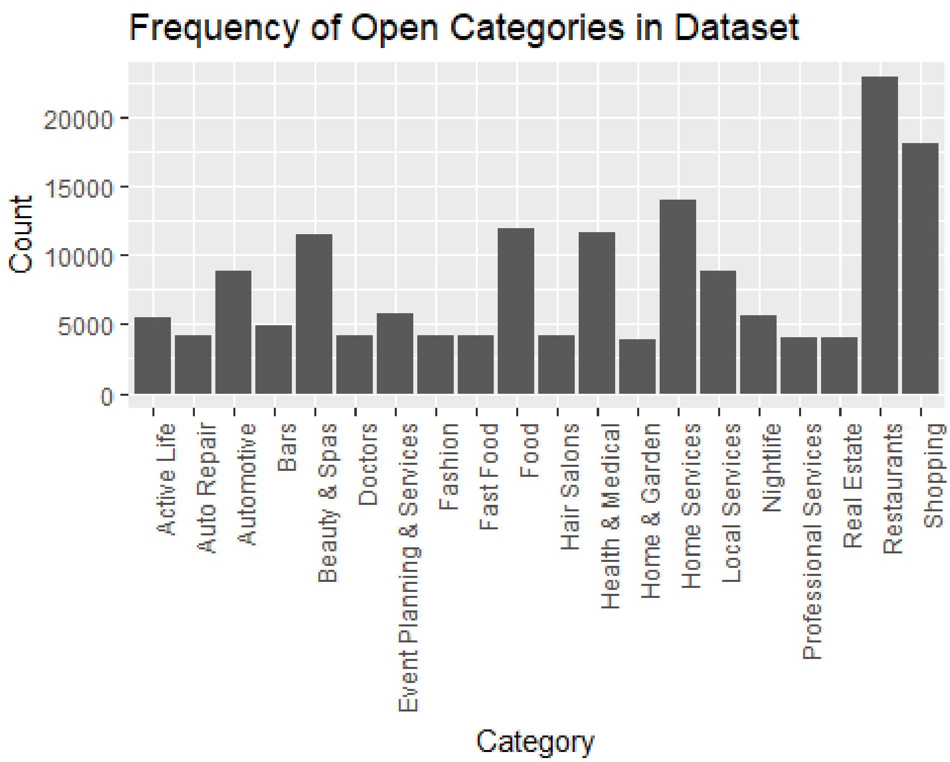
```
# Reformat data to suit plot
catplot <- business_zillow%>%select(-starts_with("hours"), -
starts_with("attribute")) %>% unnest(categories) %>%
  select(name,
categories)%>%group_by(categories)%>%summarise(n=n())%>%arrange(desc(n))%>%he
ad(20)
catplot <- as.data.frame(catplot)

ggplot(data=catplot, aes(x=categories, y=n)) +
  geom_bar(stat="identity") + theme(axis.text.x = element_text(angle = 90,
hjust = 1)) + labs(title="Frequency of Categories in Dataset") +
xlab("Category") + ylab("Count")
```




This counts the “categories” included in the dataset. These categories are tags that users might use to describe various businesses. We can see that the top category is “Restuarants” followed by “Shopping,” “Food,” “Home Services,” and “Beauty & Spa.”

The Yelp dataset also includes information on businesses that may have been open but are currently closed. The previous analyses included all of them, but here we assess the counts of categories only for businesses that are open.

```
catplot_open <- business_zillow %>%
  select(-starts_with("hours"), -starts_with("attribute")) %>%
  filter(is_open==1) %>%
  unnest(categories) %>%
  select(name, categories) %>%
  group_by(categories) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  head(20)

catplot_open  <- as.data.frame(catplot_open )

ggplot(data=catplot_open , aes(x=categories, y=n)) +
  geom_bar(stat="identity") + theme(axis.text.x = element_text(angle = 90,
hjust = 1)) + labs(title="Frequency of Open Categories in Dataset") +
xlab("Category") + ylab("Count")
```




Top categories include: “Restaurants,” “Shopping,” “Home Services,” “Food,” and “Health & Medical.” Interestingly, it seems like “Food” and “Beauty & Spas” might be closing at a higher rate than other categories.

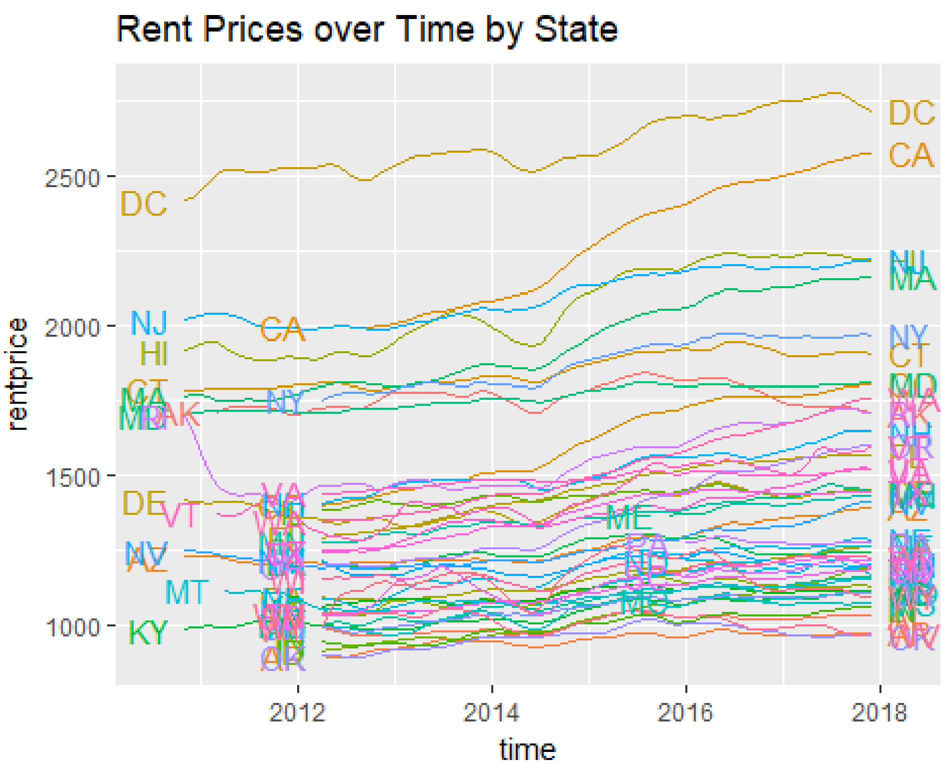
Next, we compare the rent prices across cities. We use cities as the categorizing variable because there are far too many zip codes in the dataset.

```
# Reimport data for plotting
zillow <- read_csv("zecon/Zip_Zri_AllHomesPlusMultifamily.csv", col_names =
TRUE)

# Eliminate unnecessary columns
zillow <- zillow[-c(94:95)]

# Convert from long to wide by city
zillow_collapse_wide <- zillow %>%
  group_by(City) %>%
  summarize_all(funs(mean))
names(zillow_long)

##  [1] "RegionID"    "postal_code" "City"        "State"       "Metro"
##  [6] "CountyName"  "SizeRank"    "time"        "rentprice"   "month"
## [11] "year"        "YearMonth"

# Collapse to long by State and time period.
zillow_collapse_long <- zillow_long %>%
```

```
  group_by(State, time) %>%
  summarize(rentprice = mean(rentprice))
```

Here, we compose a graph illustrating rents over time:

```
ggplot(zillow_collapse_long, aes(x = time, y=rentprice, group = State, colour
= State)) + geom_line()  + scale_colour_discrete(guide = 'none')  +
scale_x_date(expand=c(0.1, 0)) + geom_dl(aes(label = State), method =
list(dl.trans(x = x + .2), "last.points")) + geom_dl(aes(label = State),
method = list(dl.trans(x = x - .2), "first.points")) + labs(title = "Rent
Prices over Time by State", xlab = "Rent Price (USD)")
```

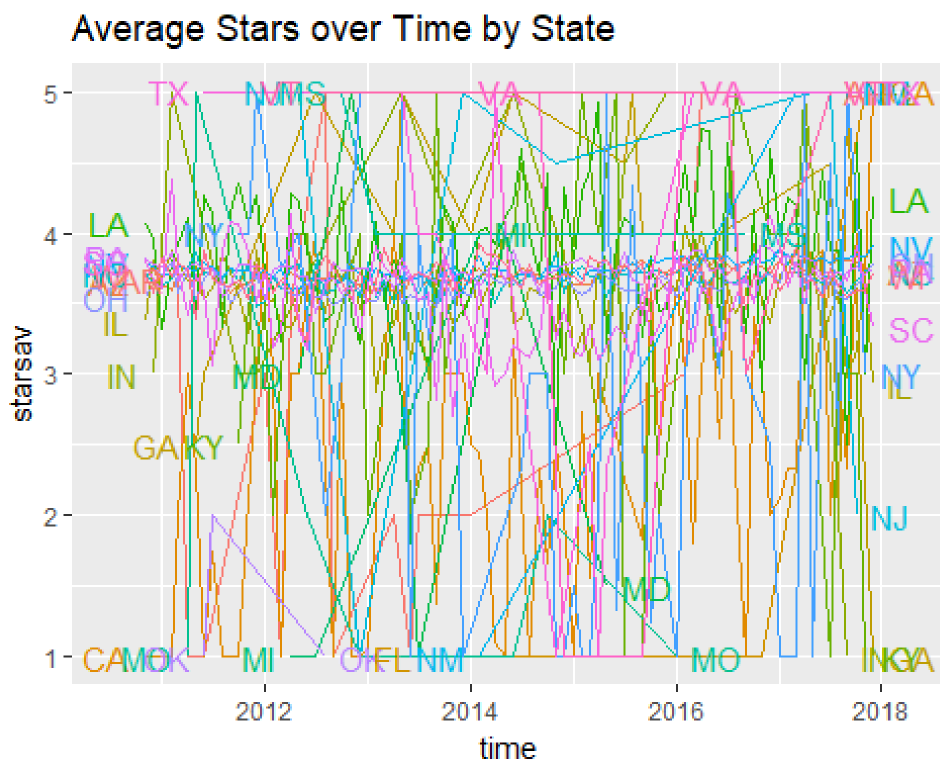



Few states in the dataset appear to have experienced overall declines in rent over the periods in question. California, Oregon, Colorado, Massachussets, and Washington appear to have experienced significant increases over the time period. DC, California, New Jersey, Hawaii, and Massachussets are consistently plagued by high rent prices.

## Comparing the Average Star Values Across Cities

```
# First, we collapse by state
FDL_collapse_long <- Full_data_long %>%
  group_by(State, time) %>%
  summarize(starsav = mean(starsav))

ggplot(FDL_collapse_long, aes(x = time, y=starsav, group = State, colour =
State)) + geom_line()  + scale_colour_discrete(guide = 'none')  +
scale_x_date(expand=c(0.1, 0)) + geom_dl(aes(label = State), method =
```

```
list(dl.trans(x = x + .2), "last.points")) +
  geom_dl(aes(label = State), method = list(dl.trans(x = x - .2),
"first.points")) + labs(title = "Average Stars over Time by State", xlab =
"Avg. Stars (1-5)")
```




As we can see, there don't appear to be any trends or patterns - it seems like a very random relationship.

## Modeling and Panel Data Regression Analysis

Now comes the fun part. We split the data into train and test sets where our test set comprises the last year (12 months) of our data. We run a Hausman test to determine whether we should run a fixed effects ("within") or a random effects model. Then we develop a Panel Linear Regression model to predict housing prices.

```
# Create test and train
Full_data_long_train <- Full_data_long[Full_data_long$time < "2017-01-01",]
Full_data_long_test <- Full_data_long[Full_data_long$time >= "2017-01-01",]

# Set parameters
my.formula <- rentprice ~ starsav + is_openave + funnyav + coolav + usefulav
my.index <- c('postal_code','time')

# Conduct test
my.hausman.test.train <- phtest(x = my.formula,
                                data = Full_data_long_train,
```

```r
                              model = c('within', 'random'),
                              index = my.index)
print(my.hausman.test.train)
```

```
##
##  Hausman Test
##
## data:  my.formula
## chisq = 1.2796, df = 5, p-value = 0.937
## alternative hypothesis: one model is inconsistent
```

The high p-value of 0.937 indicates that we should use a random effects model instead of a fixed effects model.

Now, we build our random effects model on the training dataset and predict on the test set. We calculate the Mean Averege Percent Error (MAPE) to see how accurately our model uses training values to predict rental prices in the test set.

```r
my.pdm.train <- plm(data = Full_data_long_train,
              formula = my.formula,
              model = 'random',
              index = my.index)
summary(my.pdm.train)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = my.formula, data = Full_data_long_train, model = "random",
##     index = my.index)
##
## Unbalanced Panel: n = 560, T = 1-74, N = 31397
##
## Effects:
##                     var   std.dev share
## idiosyncratic   6321.76     79.51 0.023
## individual    270893.25    520.47 0.977
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8490  0.9817  0.9822  0.9806  0.9822  0.9822
##
## Residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2126.91   -40.27    -6.88    -0.36    37.76  2068.60
##
## Coefficients:
##               Estimate Std. Error  t-value  Pr(>|t|)
## (Intercept) 1238.62701   22.65028  54.6848 < 2.2e-16 ***
## starsav       -2.85126    0.83058  -3.4329  0.000598 ***
## is_openave   131.20358    3.13631  41.8337 < 2.2e-16 ***
## funnyav      -10.17243    0.86013 -11.8266 < 2.2e-16 ***
```

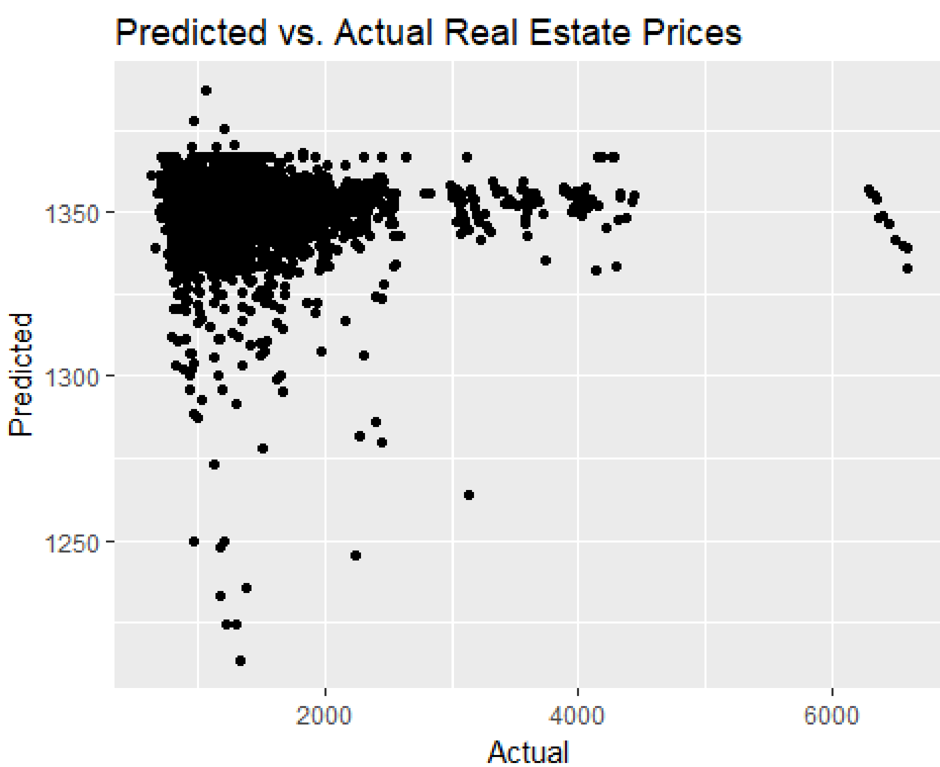
```
## coolav        11.10536    0.96571  11.4997 < 2.2e-16 ***
## usefulav     -11.10133    0.42243 -26.2800 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    226790000
## Residual Sum of Squares: 203110000
## R-Squared:      0.10443
## Adj. R-Squared: 0.10429
## F-statistic: 731.838 on 5 and 31391 DF, p-value: < 2.22e-16
```

```
Full_data_long_test$pred.plm.test <- predict(my.pdm.train,
Full_data_long_test, type='response')

plmmape <-
100*mean(abs(Full_data_long_test$pred.plm.test/Full_data_long_test$rentprice-
1), na.rm = T)
print(plmmape)
```

```
## [1] 21.39346
```

MAPE is only 21.39% right now. We will continue working on the model to get this error lower.

```
ggplot(Full_data_long_test, aes(x=rentprice, y=pred.plm.test)) +geom_point()
+ labs(title="Predicted vs. Actual Real Estate Prices") + xlab("Actual") +
ylab("Predicted")
```



The plot above shows a significant discrepency between actual and predicted rent prices.

## Generating Lag

We first have to ensure that both sets have the same postal codes in order to conduct the fixed effects analysis dictated by the hausman test.

```
Full_data_long_train <- Full_data_long[Full_data_long$time < "2017-01-01",]
Full_data_long_test <- Full_data_long[Full_data_long$time >= "2017-01-01",]
Full_data_long_test <-
Full_data_long_test[Full_data_long_test$postal_code!="05440",]

train <- unique(Full_data_long_train$postal_code)
test<- unique(Full_data_long_test$postal_code)

Full_data_long_test <- dplyr::filter(Full_data_long_test, postal_code %in%
train)
Full_data_long_train <- dplyr::filter(Full_data_long_train, postal_code %in%
test)

length(unique(Full_data_long_test$postal_code))

## [1] 522

length(unique(Full_data_long_train$postal_code))

## [1] 522
```

To fine-tune the model, we decided to lag the dependent variable to consider the possibility that last month's rent could be the best predictor of this month's rent. We follow a similar process to the one above for training, testing, and predicting.

```
my.lag.formula <- rentprice ~ lag(rentprice, 1) + starsav + is_openave +
funnyav + coolav + usefulav + Number_of_reviews

# Conduct Hausman Test
my.hausman.test.train.lag <- phtest(x = my.lag.formula,
                                    data = Full_data_long_train,
                                    model = c('within', 'random'),
                                    index = my.index)

print(my.hausman.test.train.lag)

##
##  Hausman Test
##
## data:  my.lag.formula
## chisq = 283.86, df = 7, p-value < 2.2e-16
## alternative hypothesis: one model is inconsistent
```

Then, we build the model on the training set and predict on the test set in order to calculate the MAPE.

```
# Regular LM using zip dummies
my.pdm.train.lag.lm <- lm (rentprice ~ lag(rentprice, 1) + starsav +
is_openave + funnyav + coolav + usefulav + Number_of_reviews + postal_code +
time, data = Full_data_long_train)

# Predict
Full_data_long_test$pred.plm.test.lag <- predict(my.pdm.train.lag.lm,
Full_data_long_test)

# MAPE
plmmape.lag <-
100*mean(abs(Full_data_long_test$pred.plm.test.lag/Full_data_long_test$rentpr
ice-1), na.rm = T)
print(plmmape.lag)

## [1] 5.181216e-13
```

Now we get a MAPE of 5.181216e-13, far lower than the non-lagged model. This supports the hypothesis that last month's rent could be the best predictor of this month's rent price.

```
ggplot(Full_data_long_test, aes(x=rentprice, y=pred.plm.test.lag))
+geom_point() + labs(title="Predicted vs. Actual Real Estate Prices") +
xlab("Actual") + ylab("Predicted")
```

## Multiple Imputation for Missing Values Using the Amelia Package

This process uses bootstrapping and an Expectation-Maximization algorithm to impute the missing values in a data set. In our model, we will be able to throw in almost all of our independent variables.

```
# Look at missingness to get a sense of what needs to be imputed.
sapply(Full_data_long, function(x) sum(is.na(x)))

##                    X          postal_code            YearMonth
##                    0                    0                    0
## Number_of_businesses    Number_of_reviews              starsav
##                    0                    0                    0
##              starssd             usefulav              funnyav
##                 2977                    0                    0
##               coolav             bizstars           bizstarssd
##                    0                    0                 2977
##          bizrevcount             bizrevav           is_openave
##                    0                    0                    0
##           Friday_ave           Monday_ave         Saturday_ave
##                 1337                 1337                 1337
##           Sunday_ave         Thursday_ave          Tuesday_ave
##                 1337                 1337                 1337
##        Wednesday_ave         Friday_total         Monday_total
##                 1337                 1337                 1337
##       Saturday_total         Sunday_total       Thursday_total
##                 1337                 1337                 1337
##        Tuesday_total      Wednesday_total             RegionID
##                 1337                 1337                    0
##                 City                State                Metro
##                    0                    0                   64
##           CountyName             SizeRank                 time
##                    0                    0                    0
##            rentprice                month                 year
##                  580                    0                    0
##                  top
##                    0

Full_data_long <- Full_data_long[-c(40)]
Imputed_Full_data_long <-amelia(Full_data_long,ts= 'time', cs= 'postal_code',
p2s=0, intercs = FALSE, idvars=c('City', 'State', 'Metro', 'CountyName',
'year', 'month', 'YearMonth'))

write.amelia(obj=Imputed_Full_data_long, file.stem="imputedfull")

data1 <- read.csv("imputedfull1.csv")
data2 <- read.csv("imputedfull2.csv")
data3 <- read.csv("imputedfull3.csv")
data4 <- read.csv("imputedfull4.csv")
data5 <- read.csv("imputedfull5.csv")
```

```r
data1 <- pdata.frame(data1, index = c("postal_code", "time"))
data2 <- pdata.frame(data2, index = c("postal_code", "time"))
data3 <- pdata.frame(data3, index = c("postal_code", "time"))
data4 <- pdata.frame(data4, index = c("postal_code", "time"))
data5 <- pdata.frame(data5, index = c("postal_code", "time"))

allimp <- imputationList(list(data1,data2,data3,data4,data5))
```

We create the train and tests set using the last 12 months (1 year) for the test set, but with imputed values from an Amelia imputation iteration.

```r
data5$time <- as.Date(data5$time, "%Y-%m-%d")
data5_train <- data5[data5$time < "2017-01-01",]
data5_test <- data5[data5$time >= "2017-01-01",]

# Adjust to ensure postal codes aere same for fixed effects analysis
# 5440 and 8054 were staying in as factor in test for some reason. Take out
here.
data5_test <- data5_test[data5_test$postal_code!="5440",]
data5_test <- data5_test[data5_test$postal_code!="8054",]

trainimp <- unique(data5_train$postal_code)
testimp <- unique(data5_test$postal_code)

data5_test <- dplyr::filter(data5_test, postal_code %in% trainimp)
data5_train <- dplyr::filter(data5_train, postal_code %in% testimp)

my.formula.impute.lag <- rentprice ~ lag(rentprice, 12) + starsav + starssd +
is_openave + funnyav + coolav + usefulav + Number_of_reviews +
Number_of_businesses + Friday_ave + Monday_ave + Saturday_ave + Sunday_ave +
Thursday_ave + Tuesday_ave + Wednesday_ave + Friday_total + Monday_total +
Saturday_total + Sunday_total + Thursday_total + Tuesday_total +
Wednesday_total

my.index <- c('postal_code','time')

# Conduct Hausman Test
my.hausman.test.train.impute.lag <- phtest(x = my.formula.impute.lag,
                                           data = data5_train,
                                           model = c('within', 'random'),
                                           index = my.index)

print(my.hausman.test.train.impute.lag)

##
##  Hausman Test
##
## data:  my.formula.impute.lag
```

```
## chisq = 29512, df = 23, p-value < 2.2e-16
## alternative hypothesis: one model is inconsistent
```

Build random effects model on train and predict on test.

```
# Regular LM using zip dummies
my.pdm.train.lag.lm.immpute <- lm (rentprice ~ lag(rentprice, 12) + starsav +
starssd + is_openave + funnyav + coolav + usefulav + Number_of_reviews +
Number_of_businesses + Friday_ave + Monday_ave + Saturday_ave + Sunday_ave +
Thursday_ave + Tuesday_ave + Wednesday_ave + Friday_total + Monday_total +
Saturday_total + Sunday_total + Thursday_total + Tuesday_total +
Wednesday_total + postal_code + time, data = data5_train)

data5_test$my.pdm.train.lag.lm.immpute <-
predict(my.pdm.train.lag.lm.immpute, data5_test)

plmmape_impute_lag <-
100*mean(abs(data5_test$pred.plm.test.impute.lag/data5_test$rentprice-1),
na.rm = T)
print(plmmape_impute_lag)

## [1] NaN
```

Imputation gives us a MAPE of 1.535932e-12. However, it's important to note that the MAPE could vary slightly depending on which imputed dataset we test on - for example, if we trained on Imputed datasets 2-5 and tested on 1.

```
ggplot(data5_test, aes(x=rentprice, y=my.pdm.train.lag.lm.immpute)) +
geom_point() + labs(title="Predicted vs. Actual Real Estate Prices") +
xlab("Actual") + ylab("Predicted")
```


Now, we conduct a reduced imputed model, which excludes checkin data

```
my.formula.impute.lag.Simple <- rentprice ~ lag(rentprice, 12) + starsav +
starssd + is_openave + funnyav + coolav + usefulav + Number_of_reviews +
Number_of_businesses

my.hausman.test.train.impute.lag.Simple <- phtest(x =
my.formula.impute.lag.Simple,
                                                  data = data5_train,
                                                  model = c('within',
'random'),
                                                  index = my.index)

print(my.hausman.test.train.impute.lag.Simple)

##
##  Hausman Test
##
## data:  my.formula.impute.lag.Simple
## chisq = 30100, df = 9, p-value < 2.2e-16
## alternative hypothesis: one model is inconsistent

# Build random effects model on train and predict on test
my.pdm.train.impute.lag.Simple  <- lm (rentprice ~ lag(rentprice, 12) +
starsav + starssd + is_openave + funnyav + coolav + usefulav +
Number_of_reviews + Number_of_businesses + postal_code + time, data =
data5_train)
```

```r
# Predict
data5_test$pred.plm.test.impute.lag.Simple <-
predict(my.pdm.train.impute.lag.Simple, data5_test)

plmmape_impute_lag.Simple <-
100*mean(abs(data5_test$pred.plm.test.impute.lag.Simple/data5_test$rentprice-
1), na.rm = T)
print(plmmape_impute_lag.Simple)

## [1] 1.308246e-12
```

Here, our imputation process gives us a MAPE of 4.75.

```r
ggplot(data5_test, aes(x=rentprice, y=pred.plm.test.impute.lag.Simple))
+geom_point() + labs(title="Predicted vs. Actual Real Estate Prices") +
xlab("Actual") + ylab("Predicted")
```


## Final Model

The last thing we do is subset the Business dataset to include only businesses categorized as “food” or “bars.” We do this because we expect these businesses will have a stronger relationship to rent prices than others, such as “Beauty & Spa” businesses.

First we set up the new subset data

```r
yelp_review_long_food <- read.csv("yelplongPC_food.csv", header = T,
na.strings=c("NA"))
sapply(yelp_review_long_food, function(x) sum(is.na(x)))
# No missing postal codes
length(unique(yelp_review_long_food$postal_code))
# 3274 unique postal codes
yelp_review_long_food$YearMonth <-
as.yearmon(yelp_review_long_food$YearMonth)

############### Merge Checkin with Yelp
# Nonexpanded
yelp_review_long_food <- left_join(yelp_review_long_food, checkinfull,
by=c('postal_code'), match='all')

############### Merge Zillow with Yelp

# Non-Expanded
Full_data_long_food <- inner_join(yelp_review_long_food, zillow_long,
by=c('postal_code', 'YearMonth'), match='all')
length(unique(Full_data_long_food$postal_code))
sapply(Full_data_long_food, function(x) sum(is.na(x)))

# Create train and test set using the last 12 months (1 year) for the test
set
Full_data_long_food_train <- Full_data_long_food[Full_data_long_food$time <
"2017-01-01",]
Full_data_long_food_test <- Full_data_long_food[Full_data_long_food$time >=
"2017-01-01",]

# Build the mixed effects model
# Hausman Test
# set options for Hausman test
names(Full_data_long_food)

##  [1] "X"                   "postal_code"        "YearMonth"
##  [4] "Number_of_businesses" "Number_of_reviews"  "Avnumreviews"
##  [7] "starsav"             "starssd"            "usefulav"
## [10] "funnyav"             "coolav"             "bizstars"
## [13] "bizrevcount"         "bizrevav"           "is_openave"
## [16] "Friday_ave"          "Monday_ave"         "Saturday_ave"
## [19] "Sunday_ave"          "Thursday_ave"       "Tuesday_ave"
## [22] "Wednesday_ave"       "Friday_total"       "Monday_total"
## [25] "Saturday_total"      "Sunday_total"       "Thursday_total"
## [28] "Tuesday_total"       "Wednesday_total"    "RegionID"
## [31] "City"                "State"              "Metro"
## [34] "CountyName"          "SizeRank"           "time"
## [37] "rentprice"           "month"              "year"

my.formula <- rentprice ~ starsav + is_openave + funnyav + coolav + usefulav
```

```r
my.index <- c('postal_code','time')
# Conduct Hausman Test
my.hausman.test.train.food <- phtest(x = my.formula,
                                     data = Full_data_long_food_train,
                                     model = c('within', 'random'),
                                     index = my.index)

# print result
print(my.hausman.test.train.food)

##
##  Hausman Test
##
## data:  my.formula
## chisq = 1.7284, df = 5, p-value = 0.8853
## alternative hypothesis: one model is inconsistent
```

The high p-value indicates that we should use a random effects model instead of a fixed effects model.

Now, we build our random effects model on the training dataset and predict on the test set. We calculate the Mean Averege Percent Error (MAPE) to see how accurately our model uses training values to predict rental prices in the test set.

```r
# Built random effects model on train
my.pdm.train.food <- plm(data = Full_data_long_food_train,
                    formula = my.formula,
                    model = 'random',
                    index = my.index)
summary(my.pdm.train.food)

## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = my.formula, data = Full_data_long_food_train, model =
"random",
##     index = my.index)
##
## Unbalanced Panel: n = 429, T = 1-74, N = 20281
##
## Effects:
##                     var   std.dev share
## idiosyncratic   5949.20     77.13 0.024
## individual    238632.32    488.50 0.976
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8440  0.9779  0.9810  0.9783  0.9816  0.9816
##
## Residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

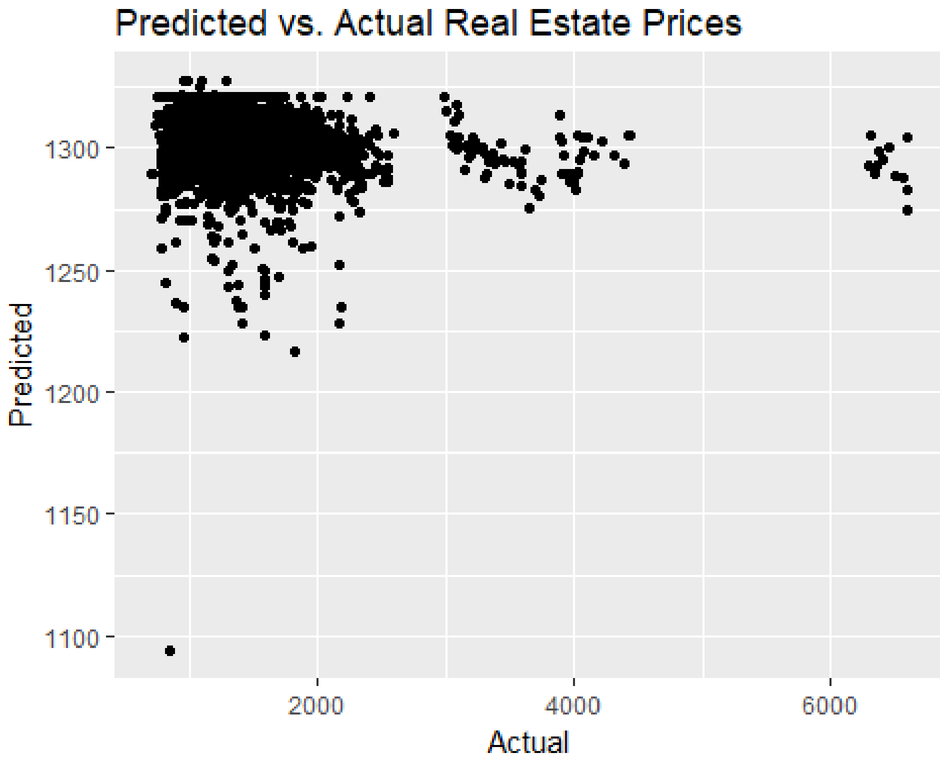
```
## -519.40  -42.58   -7.20    0.32   41.55  708.15
##
## Coefficients:
##               Estimate Std. Error  t-value  Pr(>|t|)
## (Intercept) 1274.70988   23.87348  53.3944 < 2.2e-16 ***
## starsav       -7.98247    0.63154 -12.6397 < 2.2e-16 ***
## is_openave    54.49289    3.13854  17.3625 < 2.2e-16 ***
## funnyav       -3.39684    0.80032  -4.2444 2.202e-05 ***
## coolav         6.71086    0.88533   7.5801 3.602e-14 ***
## usefulav      -6.19587    0.51782 -11.9652 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    126710000
## Residual Sum of Squares: 120430000
## R-Squared:      0.049614
## Adj. R-Squared: 0.04938
## F-statistic: 211.439 on 5 and 20275 DF, p-value: < 2.22e-16
```

```
# Predict
Full_data_long_food_test$pred.plm.test <- predict(my.pdm.train.food,
Full_data_long_food_test, type='response')

# MAPE
plmmape.food <-
100*mean(abs(Full_data_long_food_test$pred.plm.test/Full_data_long_food_test$
rentprice-1), na.rm = T)
print(plmmape.food)
```

```
## [1] 18.69031
```

MAPE is only 18.69% right now. We will continue working on the model to get this error lower.

```
ggplot(Full_data_long_food_test, aes(x=rentprice, y=pred.plm.test))
+geom_point() + labs(title="Predicted vs. Actual Real Estate Prices") +
xlab("Actual") + ylab("Predicted")
```




## Generating the Lag Model

We first have to ensure that both sets have the same postal codes in order to conduct the fixed effects analysis dictated by the hausman test.

```
# Other zips were staying in as factors in test for some reason. Take out
here.
Full_data_long_food_test <-
Full_data_long_food_test[Full_data_long_food_test$postal_code!="28803",]
Full_data_long_food_test <-
Full_data_long_food_test[Full_data_long_food_test$postal_code!="85266",]

trainimpfood <- unique(Full_data_long_food_train$postal_code)
testimpfood <- unique(data5_test$postal_code)

Full_data_long_food_test <- dplyr::filter(Full_data_long_food_test,
postal_code %in% trainimpfood)
Full_data_long_food_train <- dplyr::filter(Full_data_long_food_train,
postal_code %in% testimpfood)
```

To fine-tune the model, we decided to lag the dependent variable to consider the possibility that last month's rent could be the best predictor of this month's rent. We follow a similar process to the one above for training, testing, and predicting.

```
my.lag.formula <- rentprice ~ lag(rentprice, 1) + starsav + is_openave +
funnyav + coolav + usefulav + Number_of_reviews
```

```r
# Conduct Hausman Test
my.hausman.test.food.train.lag <- phtest(x = my.lag.formula,
                                         data = Full_data_long_food_train,
                                         model = c('within', 'random'),
                                         index = my.index)

# print result
print(my.hausman.test.food.train.lag)
```

```
##
##  Hausman Test
##
## data:  my.lag.formula
## chisq = 113.62, df = 7, p-value < 2.2e-16
## alternative hypothesis: one model is inconsistent
```

Then, we build the model on the training set and predict on the test set in order to calculate the MAPE.

```r
# Built fixed effects model on train
my.pdm.train.impute.lag.Simple  <- lm (rentprice ~ lag(rentprice, 1) +
starsav + is_openave + funnyav + coolav + usefulav + Number_of_reviews +
postal_code + time, data = Full_data_long_food_train)

# Predict
Full_data_long_food_test$pred.plm.test.lag <-
predict(my.pdm.train.impute.lag.Simple, Full_data_long_food_test)

# MAPE
plmmape.lag.food <-
100*mean(abs(Full_data_long_food_test$pred.plm.test.lag/Full_data_long_food_t
est$rentprice-1), na.rm = T)
print(plmmape.lag.food)
```

```
## [1] 6.675503e-13
```

MAPE is now 3.37%, which is the lowest MAPE that we've arrived to.

```r
ggplot(Full_data_long_food_test, aes(x=rentprice, y=pred.plm.test.lag))
+geom_point() + labs(title="Predicted vs. Actual Real Estate Prices") +
xlab("Actual") + ylab("Predicted")
```

Predicted vs. Actual Real Estate Prices