FinalRMDScript_050618_ForPDF

Christian Conroy

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 checkin data and flatten it.

```
# Stream in Checkin Data
yelp_checkin <-
as.data.frame(jsonlite::stream_in(file("dataset/checkin.json")), flatten =
TRUE)
# Flatten Checkin 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 to long and using string functions to isolate day of the week name in string.

```
# 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')
```

```
# Elimminate 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 will collapse the data, reshape it from long to wide, then merge them together.

```
# Aggregate checkin by business and time period to get checkin average and
totalby 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. Then we eliminate useless columns and merge.

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

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

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

# Collapse checkin data by zipcode to get total and average checkins each
```

```r
weekday for each 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

Because of 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.

```r
# 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 ym 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 that has information on all rental values across the US and Canada. We will 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 will change the “time” variable to a date class, and finally we 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. We will later attempt to recitfy this through imputation.

Merging all datasets

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

```
# Merge Checkin 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

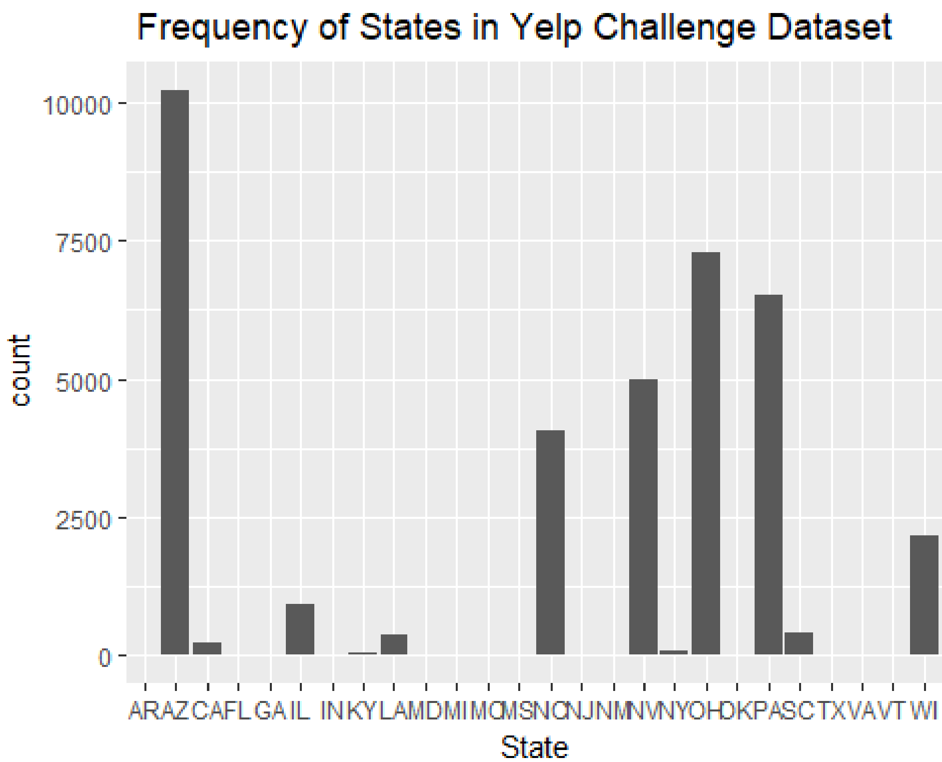
Here is where we conduct the official analysis portion of the project

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

# Assessing 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.

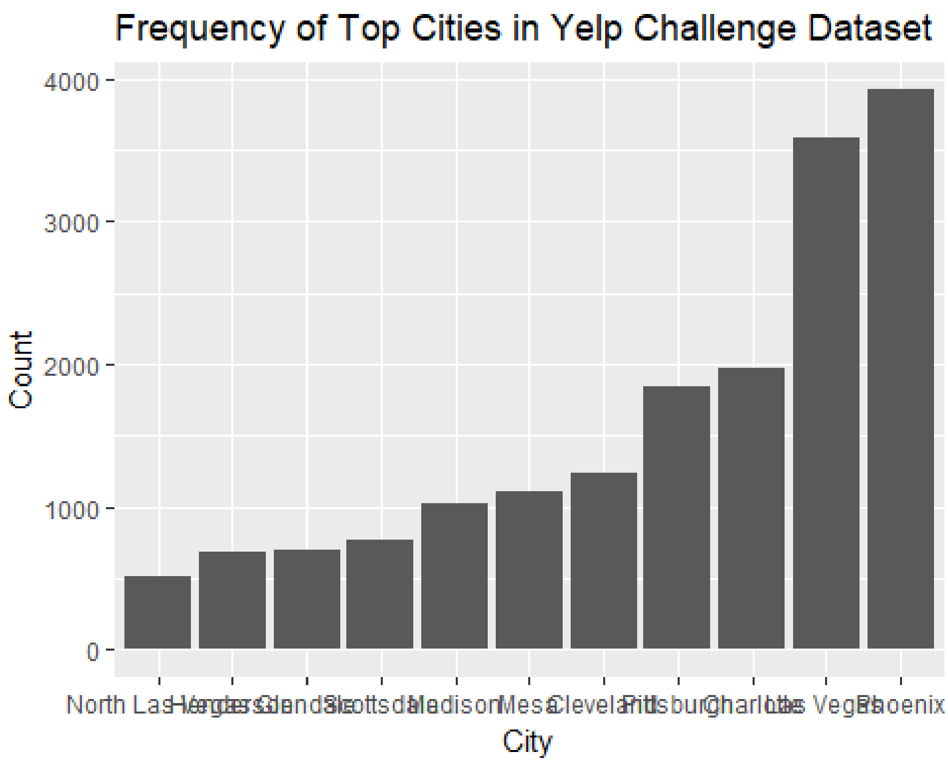
```
Full_data_long$City <- Full_data_long$City %>% as.factor
Full_data_long$City %>% summary
```

```
##            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
```

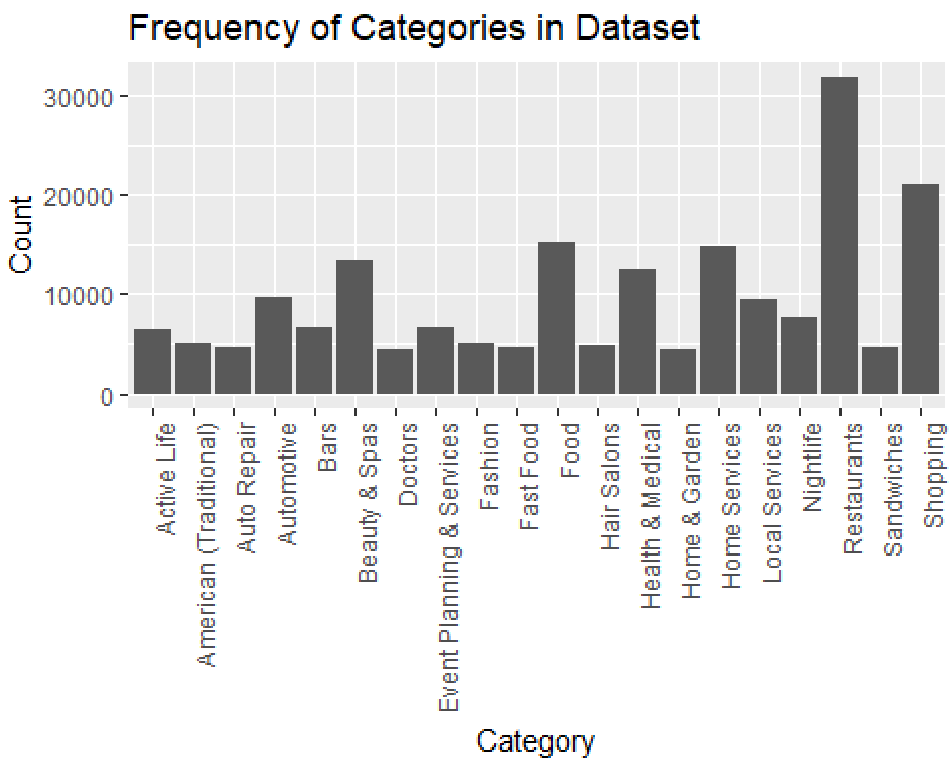
```
temp <- row.names(as.data.frame(summary(Full_data_long$City, max=12))) # create a df or something else with the summary output.
Full_data_long$City <- as.character(Full_data_long$City) # IMPORTANT! Here was the problem: turn into character values
Full_data_long$top <- ifelse(
  Full_data_long$City %in% temp, ## condition: match aDDs$answer with row.names in summary df
  Full_data_long$City, ## then it should be named as aDDs$answer
  "Other" ## else it should be named "Other"
)
Full_data_long$top <- as.factor(Full_data_long$top) # factorize the output again
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")
```



This assesses the count of cities in the dataset. Top cities include Phoenix, Las Vegas, Charlotte, Pittsburgh, Cleveland.

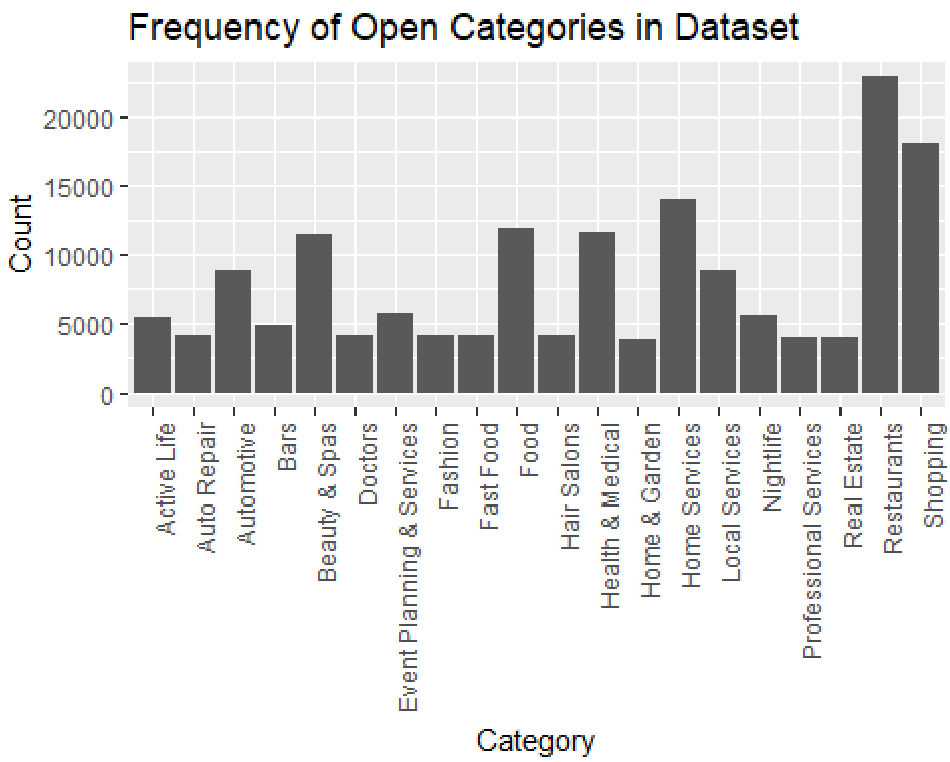
```
# Reformat data to suit plot
catplot <- business_zillow%>%select(-starts_with("hours"), -
starts_with("attribute")) %>% unnest(categories) %>%
  select(name,
categories)%>%group_by(categories)%>%summarise(n=n())%>%arrange(desc(n))%>%he
ad(20)
catplot <- as.data.frame(catplot)

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

The Yelp dataset 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 buinesses 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")
```



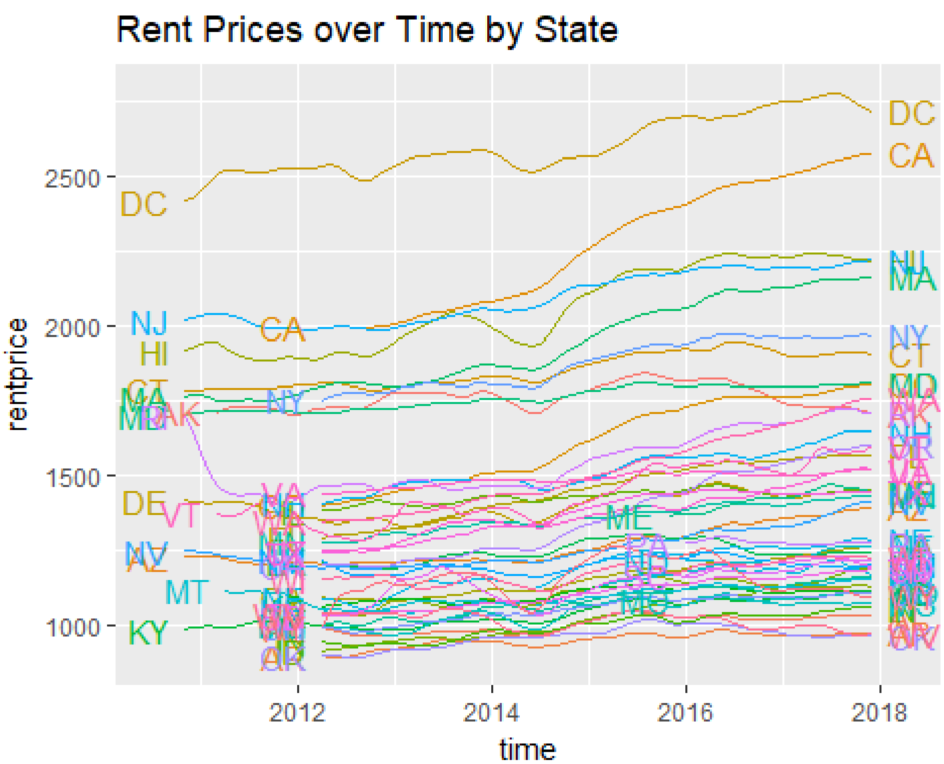
Top categories include: Restaurants, Shopping, Home Services, Food, Health & Medical. Interestingly, it seems like "Food" and "Beauty & Spas" might be closing at higher rate than other categories.

Now, we compare the rent prices across cities. We use cities as 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 unneeded 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))
```

Finally, 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)")
```

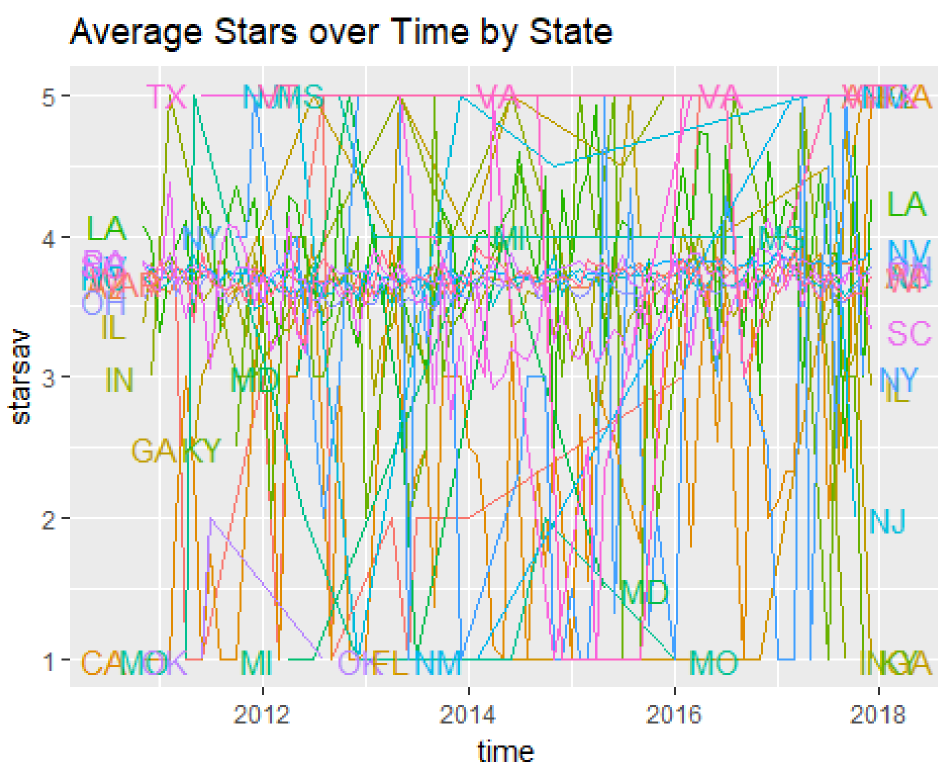


Few states in the dataset appear to have experienced overall declines in rent over the periods in question. California, Oregon, Colorado, Massachussets, and Washington appear to have experienced significant increases over the time period. DC, California, New Jersey, Hawaii, and Massachussets are consistently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

Comparing the average star values across cities. First, we must collapse by state.

```
# 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. 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 set. 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,
                                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.

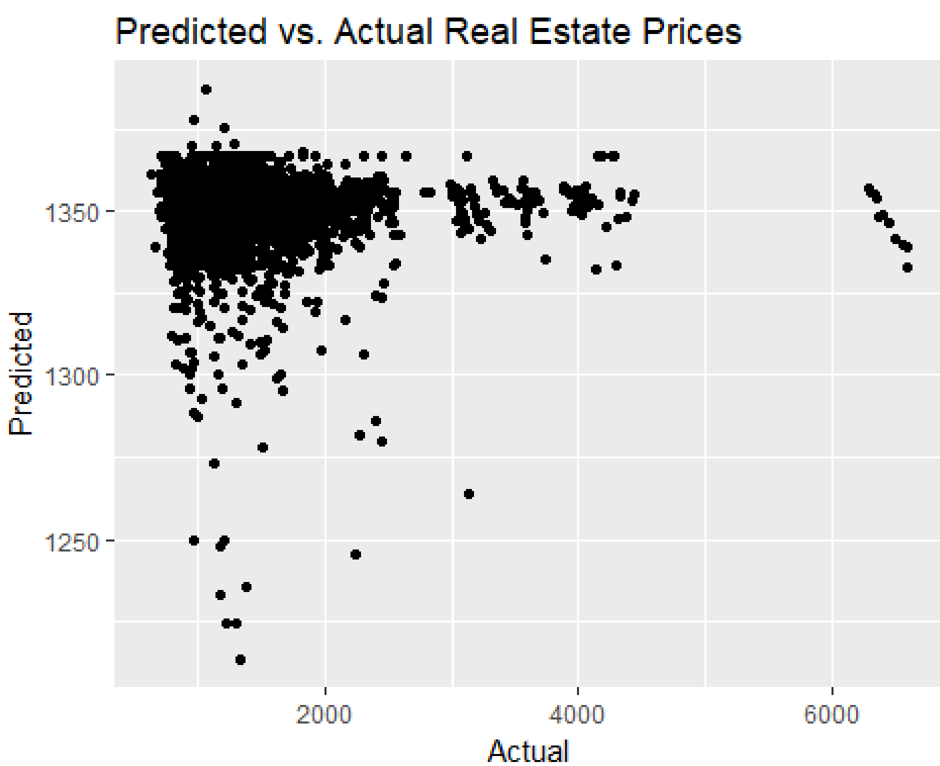
```
my.pdm.train <- plm(data = Full_data_long_train,
              formula = my.formula,
              model = 'random',
              index = my.index)
summary(my.pdm.train)

## 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 existing between actual and predicted rent prices.

Generating the Lag Model

To fine-tune the model, we decide to lag the dependent variable to consider the possibility that last month's rent could be the best predictor of this month's rent price. 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 result
print(my.hausman.test.train.lag)

##
##  Hausman Test
##
## data:  my.lag.formula
## chisq = 289.55, 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.

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

## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = my.lag.formula, data = Full_data_long_train, model =
"random",
##     index = my.index)
##
## Unbalanced Panel: n = 519, T = 1-73, N = 29252
##
## Effects:
##                    var std.dev share
## idiosyncratic 161.483  12.708 0.993
## individual      1.107   1.052 0.007
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
## 0.00341 0.17795 0.18361 0.17177 0.18361 0.18361
##
## Residuals:
##     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -142.410    -4.848     0.001     0.009     5.213   132.569
##
## Coefficients:
##                      Estimate  Std. Error   t-value  Pr(>|t|)
## (Intercept)       -0.81953342  0.73031005   -1.1222  0.261799
## lag(rentprice, 1)  1.00093673  0.00017581 5693.3592 < 2.2e-16 ***
## starsav           -0.25238363  0.14094489   -1.7907  0.073359 .
## is_openave         3.34976459  0.48784817    6.8664 6.715e-12 ***
## funnyav           -0.49065915  0.15640230   -3.1372  0.001708 **
## coolav            -0.01465219  0.16578108   -0.0884  0.929573
## usefulav          -0.19242788  0.07260164   -2.6505  0.008043 **
## Number_of_reviews  0.00124545  0.00029642    4.2016 2.659e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    5351700000
## Residual Sum of Squares: 4771300
## R-Squared:      0.99911
## Adj. R-Squared: 0.99911
## F-statistic: 4681730 on 7 and 29244 DF, p-value: < 2.22e-16
```

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

# 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] 0.2112457
```

Now we get a MAPE of 0.211, 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 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")

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))
```

Now, we will 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.

```
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",]

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
```

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

```
##
##  Hausman Test
##
## data:  my.formula.impute.lag
## chisq = 29325, df = 23, p-value < 2.2e-16
## alternative hypothesis: one model is inconsistent
```

Build random effects model on train and predict on test.

```r
my.pdm.train.impute.lag <- plm(data = data5_train,
                    formula = my.formula.impute.lag,
                    model = 'random',
                    index = my.index)
summary(my.pdm.train.impute.lag)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = my.formula.impute.lag, data = data5_train, model = "random",
##     index = my.index)
##
## Unbalanced Panel: n = 425, T = 1-62, N = 22754
##
## Effects:
##                   var std.dev share
## idiosyncratic 4866.88   69.76 0.901
## individual     533.53   23.10 0.099
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05068 0.64187 0.64187 0.62772 0.64187 0.64187
##
## Residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2176.60   -31.74    -6.65     0.07    23.44  1316.70
##
## Coefficients:
##                         Estimate  Std. Error  t-value  Pr(>|t|)
## (Intercept)          135.3726454  13.4967249  10.0300 < 2.2e-16 ***
## lag(rentprice, 12)     0.8362233   0.0032052 260.8992 < 2.2e-16 ***
## starsav                4.3925615   2.0918703   2.0998 0.0357552 *
## starssd               12.4481104   2.9101549   4.2775 1.898e-05 ***
## is_openave            56.1246937   6.3694282   8.8116 < 2.2e-16 ***
## funnyav               -6.9686115   2.0246885  -3.4418 0.0005789 ***
```

```
## coolav                  4.8148438   1.9313426   2.4930 0.0126738 *
## usefulav               -4.2287045   0.8987007  -4.7054 2.549e-06 ***
## Number_of_reviews      -0.0168226   0.0103938  -1.6185 0.1055644
## Number_of_businesses    0.5307349   0.0437210  12.1391 < 2.2e-16 ***
## Friday_ave            -25.4383904   9.6657602  -2.6318 0.0084990 **
## Monday_ave            -13.9461762   7.8711624  -1.7718 0.0764401 .
## Saturday_ave            6.0077063   6.6304076   0.9061 0.3649010
## Sunday_ave             12.7093866   6.1359542   2.0713 0.0383423 *
## Thursday_ave           17.2457619   9.8452913   1.7517 0.0798430 .
## Tuesday_ave           -21.8732883  11.0744473  -1.9751 0.0482674 *
## Wednesday_ave          32.0943737  11.2596608   2.8504 0.0043706 **
## Friday_total            0.0231701   0.0049521   4.6789 2.901e-06 ***
## Monday_total           -0.0218585   0.0043300  -5.0482 4.495e-07 ***
## Saturday_total         -0.0242709   0.0040466  -5.9979 2.029e-09 ***
## Sunday_total            0.0221750   0.0037099   5.9772 2.303e-09 ***
## Thursday_total          0.0023054   0.0077629   0.2970 0.7664894
## Tuesday_total           0.0247671   0.0103663   2.3892 0.0168932 *
## Wednesday_total        -0.0307317   0.0092509  -3.3220 0.0008950 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    1000700000
## Residual Sum of Squares: 215590000
## R-Squared:      0.78455
## Adj. R-Squared: 0.78434
## F-statistic: 3598.79 on 23 and 22730 DF, p-value: < 2.22e-16

data5_test$pred.plm.test.impute.lag <- predict(my.pdm.train.impute.lag,
data5_test, type='response')

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] 5.102004
```

Imputation gives us 5.127882 (Might be different if we tried the other 4 imputed data sets)

```
ggplot(data5_test, aes(x=rentprice, y=pred.plm.test.impute.lag)) +
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.index <- c('postal_code','time')

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 = 30105, 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 <- plm(data = data5_train,
                              formula = my.formula.impute.lag.Simple,
```

```r
                                model = 'random',
                                index = my.index)
summary(my.pdm.train.impute.lag.Simple)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = my.formula.impute.lag.Simple, data = data5_train,
##     model = "random", index = my.index)
##
## Unbalanced Panel: n = 425, T = 1-62, N = 22754
##
## Effects:
##                   var std.dev share
## idiosyncratic 4865.16   69.75 0.902
## individual     531.40   23.05 0.098
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05051 0.64130 0.64130 0.62714 0.64130 0.64130
##
## Residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2192.86   -32.02    -7.49     0.01    23.50  1328.23
##
## Coefficients:
##                        Estimate  Std. Error  t-value  Pr(>|t|)
## (Intercept)         114.1485383  12.1258561   9.4136 < 2.2e-16 ***
## lag(rentprice, 12)    0.8450445   0.0031533 267.9877 < 2.2e-16 ***
## starsav               6.2409575   2.0976999   2.9751 0.0029316 **
## starssd              16.2780616   2.9084258   5.5969 2.208e-08 ***
## is_openave           71.2143502   6.2535988  11.3877 < 2.2e-16 ***
## funnyav              -7.8383700   2.0344937  -3.8527 0.0001171 ***
## coolav                5.3923406   1.9406631   2.7786 0.0054637 **
## usefulav             -4.9796951   0.9009426  -5.5272 3.289e-08 ***
## Number_of_reviews    -0.0266971   0.0083995  -3.1784 0.0014827 **
## Number_of_businesses  0.3370970   0.0359868   9.3672 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    1003100000
## Residual Sum of Squares: 218300000
## R-Squared:      0.78237
## Adj. R-Squared: 0.78228
## F-statistic: 9084.77 on 9 and 22744 DF, p-value: < 2.22e-16
```

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

```
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] 4.943544
```

Imputation gives us 5.037

```
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 to have a stronger relationship to rent prices than others, such as Beauty & Spas.

There is not a significant change in the MAPE for the subset, with the non-imputed subset without a lagged dependent producing a MAPE of 18.69, the non-imputed subset with a lagged dependent producing a MAPE of .2142, and the imputed subset with a lag producing a MAPE of 3.46.