FinalRMDScript\_050618\_ForPDF

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.

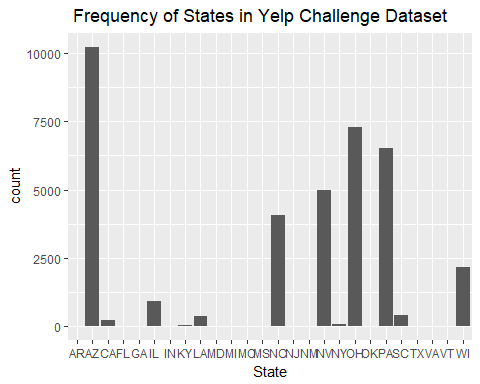
## Explortatory Data Analysis

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")

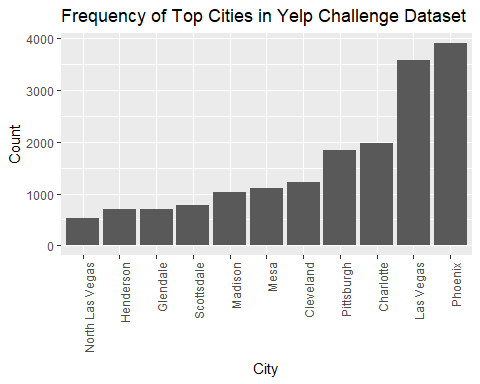


The states most represented in the dataset are Arizona, Ohio, Pennsylvania, Nevada, North Carolina, Wisconsin, and Illinois.

Full\_data\_long$City <- Full\_data\_long$City %>% as.factor  
Full\_data\_long$City %>% summary

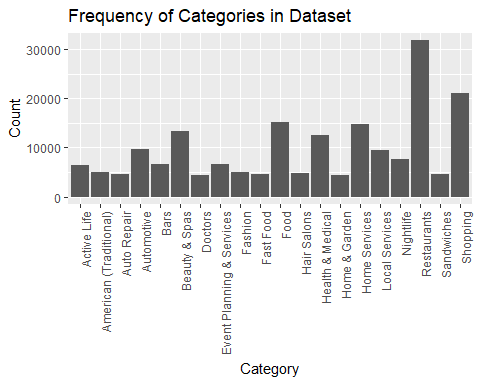
## Phoenix Las Vegas Charlotte Pittsburgh   
## 3918 3580 1972 1847   
## Cleveland Mesa Madison Scottsdale   
## 1232 1115 1021 774   
## Glendale Henderson North Las Vegas Chandler   
## 695 691 522 516   
## Gilbert Surprise Peoria Tempe   
## 516 388 344 344   
## Champaign Fort Mill Gastonia Shaler   
## 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.

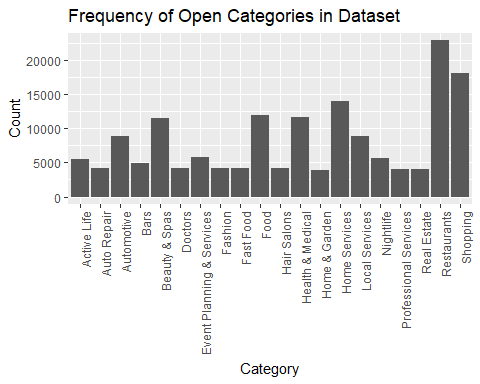
# Reformat data to suit plot  
catplot <- business\_zillow%>%select(-starts\_with("hours"), -starts\_with("attribute")) %>% unnest(categories) %>%  
 select(name, categories)%>%group\_by(categories)%>%summarise(n=n())%>%arrange(desc(n))%>%head(20)  
catplot <- as.data.frame(catplot)  
  
ggplot(data=catplot, 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.

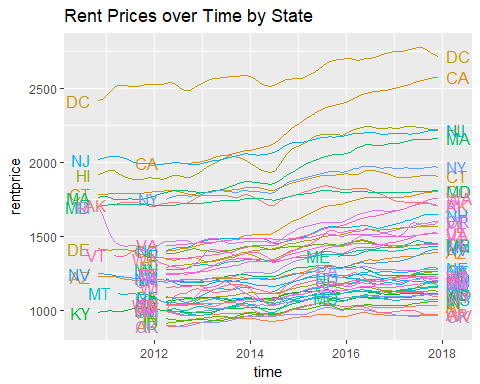
# Reimport data for plotting  
zillow <- read\_csv("zecon/Zip\_Zri\_AllHomesPlusMultifamily.csv", col\_names = TRUE)  
  
# Eliminate unnecessary columns  
zillow <- zillow[-c(94:95)]  
  
# Convert from long to wide by city  
zillow\_collapse\_wide <- zillow %>%   
 group\_by(City) %>%   
 summarize\_all(funs(mean))  
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:

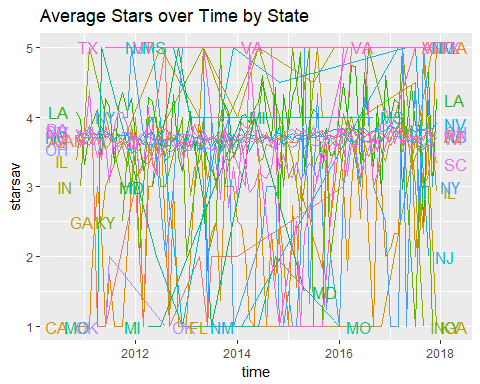
ggplot(zillow\_collapse\_long, aes(x = time, y=rentprice, group = State, colour = State)) + geom\_line() + scale\_colour\_discrete(guide = 'none') + scale\_x\_date(expand=c(0.1, 0)) + geom\_dl(aes(label = State), method = list(dl.trans(x = x + .2), "last.points")) + geom\_dl(aes(label = State), method = 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,  
 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)

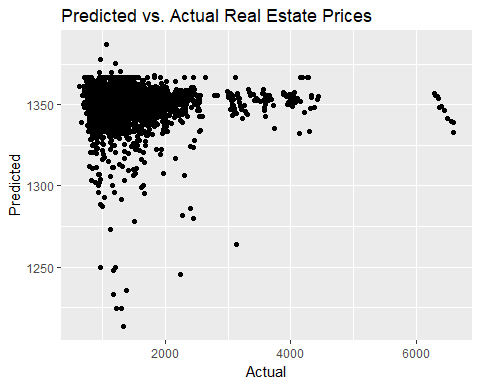
## Oneway (individual) effect Random Effect Model   
## (Swamy-Arora's transformation)  
##   
## Call:  
## plm(formula = my.formula, data = Full\_data\_long\_train, model = "random",   
## index = my.index)  
##   
## Unbalanced Panel: n = 560, T = 1-74, N = 31397  
##   
## Effects:  
## var std.dev 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 between actual and predicted rent prices.

## Generating the Lag Model

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 = 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)

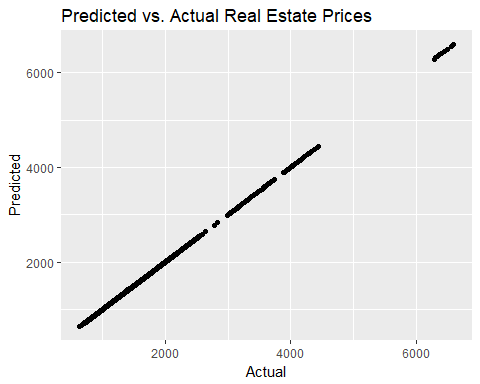
## Oneway (individual) effect Random Effect Model   
## (Swamy-Arora's transformation)  
##   
## Call:  
## plm(formula = my.lag.formula, data = Full\_data\_long\_train, model = "random",   
## index = my.index)  
##   
## Unbalanced Panel: n = 519, T = 1-73, N = 29252  
##   
## Effects:  
## var std.dev 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$pred.plm.test.lag/Full\_data\_long\_test$rentprice-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 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")  
  
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.

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  
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 = 34477, df = 23, 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 <- plm(data = data5\_train,   
 formula = my.formula.impute.lag,   
 model = 'random',  
 index = my.index)  
summary(my.pdm.train.impute.lag)

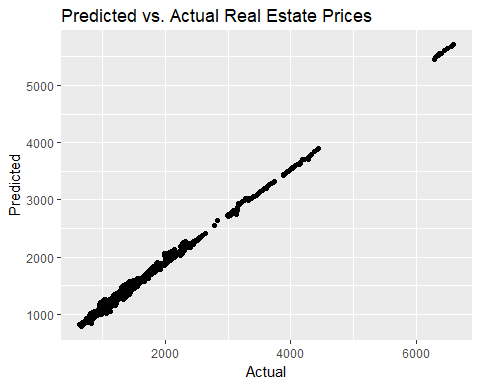
## Oneway (individual) effect Random Effect Model   
## (Swamy-Arora's transformation)  
##   
## Call:  
## plm(formula = my.formula.impute.lag, data = data5\_train, model = "random",   
## index = my.index)  
##   
## Unbalanced Panel: n = 425, T = 1-62, N = 22754  
##   
## Effects:  
## var std.dev share  
## idiosyncratic 5174.23 71.93 0.897  
## individual 592.48 24.34 0.103  
## theta:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.05276 0.64862 0.64862 0.63460 0.64862 0.64862   
##   
## Residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -1987.19 -33.71 -7.91 0.14 22.87 2462.89   
##   
## Coefficients:  
## Estimate Std. Error t-value Pr(>|t|)   
## (Intercept) 176.7334463 14.1504702 12.4896 < 2.2e-16 \*\*\*  
## lag(rentprice, 12) 0.8085751 0.0033394 242.1354 < 2.2e-16 \*\*\*  
## starsav 4.1171429 2.1924447 1.8779 0.0604107 .   
## starssd 8.9066786 3.0438146 2.9262 0.0034352 \*\*   
## is\_openave 60.3221976 6.6621539 9.0545 < 2.2e-16 \*\*\*  
## funnyav -6.8740518 2.1167569 -3.2474 0.0011661 \*\*   
## coolav 5.3818763 2.0193323 2.6652 0.0077002 \*\*   
## usefulav -5.6289367 0.9400151 -5.9881 2.154e-09 \*\*\*  
## Number\_of\_reviews -0.0135192 0.0108799 -1.2426 0.2140344   
## Number\_of\_businesses 0.5116833 0.0458448 11.1612 < 2.2e-16 \*\*\*  
## Friday\_ave -35.4058837 10.2668779 -3.4486 0.0005646 \*\*\*  
## Monday\_ave -10.0332019 8.3720370 -1.1984 0.2307667   
## Saturday\_ave 11.8762701 7.0463463 1.6855 0.0919154 .   
## Sunday\_ave 11.0252148 6.5274624 1.6891 0.0912234 .   
## Thursday\_ave 17.3572172 10.4323032 1.6638 0.0961671 .   
## Tuesday\_ave -26.8033796 11.7393873 -2.2832 0.0224277 \*   
## Wednesday\_ave 38.1144650 11.9401704 3.1921 0.0014142 \*\*   
## Friday\_total 0.0246564 0.0052733 4.6757 2.946e-06 \*\*\*  
## Monday\_total -0.0257308 0.0046082 -5.5837 2.381e-08 \*\*\*  
## Saturday\_total -0.0279379 0.0042969 -6.5018 8.099e-11 \*\*\*  
## Sunday\_total 0.0258313 0.0039383 6.5589 5.536e-11 \*\*\*  
## Thursday\_total 0.0037377 0.0082665 0.4521 0.6511654   
## Tuesday\_total 0.0286557 0.0110294 2.5981 0.0093794 \*\*   
## Wednesday\_total -0.0334005 0.0098277 -3.3986 0.0006785 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Total Sum of Squares: 980720000  
## Residual Sum of Squares: 235520000  
## R-Squared: 0.75986  
## Adj. R-Squared: 0.75961  
## F-statistic: 3126.95 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] 5.600033

Imputation gives us a MAPE of 5.127882. 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=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 = 35011, 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,   
 model = 'random',  
 index = my.index)  
summary(my.pdm.train.impute.lag.Simple)

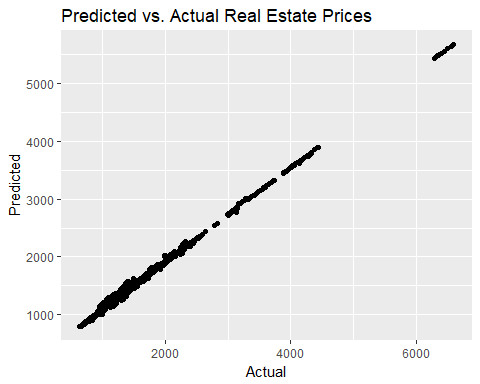
## Oneway (individual) effect Random Effect Model   
## (Swamy-Arora's transformation)  
##   
## Call:  
## plm(formula = my.formula.impute.lag.Simple, data = data5\_train,   
## model = "random", index = my.index)  
##   
## Unbalanced Panel: n = 425, T = 1-62, N = 22754  
##   
## Effects:  
## var std.dev share  
## idiosyncratic 5172.95 71.92 0.898  
## individual 586.81 24.22 0.102  
## theta:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.05231 0.64718 0.64718 0.63313 0.64718 0.64718   
##   
## Residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -1990.65 -33.75 -8.69 0.08 22.84 2479.88   
##   
## Coefficients:  
## Estimate Std. Error t-value Pr(>|t|)   
## (Intercept) 154.5764002 12.6938755 12.1772 < 2.2e-16 \*\*\*  
## lag(rentprice, 12) 0.8186285 0.0032840 249.2780 < 2.2e-16 \*\*\*  
## starsav 5.9524489 2.1987954 2.7071 0.0067916 \*\*   
## starssd 12.3853244 3.0418449 4.0716 4.684e-05 \*\*\*  
## is\_openave 74.8114630 6.5427659 11.4342 < 2.2e-16 \*\*\*  
## funnyav -7.6290408 2.1274875 -3.5859 0.0003366 \*\*\*  
## coolav 5.9415588 2.0294636 2.9276 0.0034187 \*\*   
## usefulav -6.4008440 0.9423133 -6.7927 1.128e-11 \*\*\*  
## Number\_of\_reviews -0.0240123 0.0088403 -2.7162 0.0066079 \*\*   
## Number\_of\_businesses 0.3254252 0.0378893 8.5888 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Total Sum of Squares: 986820000  
## Residual Sum of Squares: 238590000  
## R-Squared: 0.75823  
## Adj. R-Squared: 0.75813  
## F-statistic: 7925.16 on 9 and 22744 DF, p-value: < 2.22e-16

# 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] 5.445586

Here, our imputation process gives us a MAPE of 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 will have a stronger relationship to rent prices than others, such as “Beauty & Spa” businesses.

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