Analysis of the Dynamics of the U.S. Housing Price Index

Refactored Code and Workflow of STAT 1261 Final Project

Rohan Krishnan

2023-12-11

Introduction

As of November 2023, the U.S. housing market is valued at 47 trillion USD (Rosen, 2023). With an average year-over-year growth rate of 5.5%, the U.S. housing market is set to remain a key factor in calculating the overall welfare of the nation's economy (Rosen, 2023). On a consumer level, home ownership remains tied to the "American Dream", as working class citizens across the country strive towards purchasing, selling, or renting homes. Investors and economists rely on the Housing Price Index (referred to as the HPI from this point onwards) as a key metric for assessing investment feasibility and for determining national economic forecasts.

The HPI is a comprehensive macroeconomic measure that monitors the price fluctuations of **single-family** homes nationwide. It also serves as an analytical tool for approximating the changes in the rates of mortgage defaults, prepayments, and housing affordability (Liberto, 2023). The Federal Housing Finance Agency compiles this data by reviewing single family housing mortgages purchased or securitized by Fannie Mac or Freddie Mac. My primary goal for this statistical analysis project was to find an economic data set that I could explore to better understand the U.S. economy.

Research Question

The primary goal of this project is to examine how the HPI is affected by other macroeconomic factors. By examining how other commonly tracked measures affect the HPI, I aim to better understand the dynamics between macroeconomic measures and will create a useful basis for more focused research in the future. An improved understanding of the relationships may also prove useful in better understanding real estate and broader housing market dynamics.

In summary, this project aims to establish quantitative measures of the relationships between common macroeconomic measures and the HPI to better inform the direction of future research. Below is the main research question I aimed to answer for as part of the final project requirements:

- What variables are most useful in predicting the HPI?

    1. Which machine learning model best predicts the HPI given new data?

Statement of Purpose

In terms of a research-based objective, I created this project with the goal to broaden my knowledge about the economy, consumer behavior, and investment sectors associated within the U.S. housing market. Most importantly, however, this project aims to apply a wide range of machine learning techniques and provide statistical justifications and explanations for their use.

Methodology

This section will discuss my data sourcing, ingestion, and cleaning process.

Data Collection

To recreate the data set I used for my original STAT 1261 final project, I used two data sets of U.S. HPI influences from Kaggle. By comparing the values of variables between the two data sets and looking at the documentation of each, I chose only the variables whose values I could verify as accurate.

Data Set(s)

To create a unified data frame in R, I left-joined each individual csv file on `house_data.csv`. I chose `house_data.csv` as my base file because it only went back until 2003, unlike the other files that had data extending back to the 1970s. After left-joining the data into one data frame, I converted the `DATE` column to a “Date” type and used `dplyr` to filter the data into a data set that only contained measures taken after January 1st, 2003.

```r
#Load raw data and preliminary (post-2003) data
raw_data <- load_raw_data()
basic_describe_data(data = raw_data, data_name = "raw")
```

```
## The raw data set has 22 columns, 921 rows, spans 1947-01-01 to 2023-09-01,
## and has 12250 NA values.
```

```r
preliminary_data <- load_preliminary_data()
basic_describe_data(data = preliminary_data, data_name = "preliminary")
```

```
## The preliminary data set has 22 columns, 249 rows, spans 2003-01-01 to 2023-09-01,
## and has 519 NA values.
```

As is seen above, the original raw data had 921 observations but over 12,000 missing values due to the mismatch in reporting time frames. After filtering for measures after 2003, the data set is cut down to only 249 observations, a 72% reduction in usable observations, and only 519 missing values. Unfortunately, because the measures in housing_data.csv abruptly stopped in 2003, it was not feasible to perform imputation to maintain data volume.

Variables

As the main focus of this project is to explore the U.S. HPI, we have the following features and target variable(s):

Variable Name	Definition	Target or Feature?
CPIAUSCL	U.S. Consumer Price Index	Feature
FEDFUNDS	U.S. Federal Funds rate	Feature
GDP.x	U.S. GDP reported quarterly	Feature
CSUSHPISA	**U.S. Housing Price Index**	**Target**

Variable Name	Definition	Target or Feature?
building_permits	Number of new building permits in the U.S.	Feature
const_price_index	U.S. Construction Price Index	Feature
delinquency_rate	U.S. percentage of loans overdue by more than 30 days	Feature
GDP.y	(Unreliable) U.S. GDP reported monthly	NA
house_for_sale_or_sold	(Unreliable) Number of houses for sale or sold in the U.S. (?)	NA.
housing_subsidies	Value of U.S. housing subsidies	Feature
income	Median U.S. household income	Feature
interest_rate	(Unreliable) Value of U.S. interest rates	NA
mortgage_rate	Value of U.S. home mortgage rates	Feature
construction_unit	Number of new construction units in the U.S.	Feature
total_houses	Total number of houses in the U.S.	Feature
total_const_spending	Total U.S. construction spending	Feature
unemployment_rate	(Unreliable) U.S. unemployment rate	NA
urban_population	U.S. urban population (millions)	Feature
home_price_index	(Unreliable) Measure of U.S. Housing Price Index	NA
MORTGAGE30US	(Unreliable) The average interest rates on mortgage loans in the United States	NA
UNRATE	U.S. unemployment rate	Feature

From the above table, it's clear that there are several duplicate variables whose values do not match. The variables marked "unreliable" are variables whose values I could verify using FRED data. As I move through the data cleaning process, I will remove columns which have unreliable information or have too large a number of missing values.

Below is a summary table of the preliminary data. This data is what is output after joining all csv files and filtering for values after 2003.

```r
#Prelim data summary + NA by columns
stargazer(preliminary_data, summary = TRUE, type = "latex",
          title = "Preliminary Data Summary",
          header = FALSE, no.space = TRUE)
```

Table 2: Preliminary Data Summary

Statistic	N	Mean	St. Dev.	Min	Max
CPIAUCSL	249	232.681	30.280	182.600	307.481
FEDFUNDS	249	1.432	1.692	0.050	5.330
GDP.x	83	17,617.740	4,190.519	11,174.130	27,623.540
CSUSHPISA	247	184.069	45.327	128.461	306.720
building_permits	240	1,309.350	479.881	513	2,263
const_price_index	240	212.851	44.567	144.400	353.015
delinquency_rate	240	4.877	3.305	1.410	11.480
GDP.y	240	18,095.160	2,002.294	14,614.140	21,989.980
house_for_sale_or_sold	240	55.550	25.384	20	127
housing_subsidies	240	34.677	6.006	25.930	48.021
income	240	13,493.480	1,837.485	10,674.000	20,422.600
interest_rate	240	1.302	1.579	0.050	5.260
mortgage_rate	240	4.683	1.111	2.684	6.900
construction_unit	240	1,201.717	423.858	520	2,245
total_houses	240	121,344.400	6,113.869	111,278	131,202
total_const_spending	240	0.325	1.950	−5.900	5.000
unemployment_rate	240	6.012	2.034	3.500	14.700
urban_population	240	81.261	1.055	79.583	83.084
home_price_index	240	180.658	41.256	128.461	304.755
MORTGAGE30US	33	4.869	1.194	2.880	6.790
UNRATE	249	5.924	2.048	3.400	14.700

Some important points to note include the vastly different scaling between variables. For example, `GDP.x` seems to be measured in millions while `FEDFUNDS` is a measure of interest rate with a maximum value of 5.330. Also, `GDP.x` has noticeable fewer observations than the rest of the data, at only 83. This is because it is pulled directly from the FRED database, which only reports quarterly values. Since the data is monthly, there are 8 missing values for `GDP.x` per year. `MORTGAGE30US` seems to be even worse, with only 33 known values.

Below is a table showing the number of missing values per column.

```r
#Extract NA for prelim data and display
prelim_na_vals <- extract_na_per_column(preliminary_data)

prelim_na_vals %>% rename(Feature = variable,
                          "NA Values" = na_values) %>%
  stargazer(type = "latex", summary = FALSE, flip = FALSE,
          title = "Preliminary Data NA Values by Column",
          header = FALSE, no.space = TRUE)
```

Table 3: Preliminary Data NA Values by Column

	Feature	NA Values
1	DATE	0
2	CPIAUCSL	0
3	FEDFUNDS	0
4	GDP.x	166
5	CSUSHPISA	2
6	building_permits	9
7	const_price_index	9
8	delinquency_rate	9
9	GDP.y	9
10	house_for_sale_or_sold	9
11	housing_subsidies	9
12	income	9
13	interest_rate	9
14	mortgage_rate	9
15	construction_unit	9
16	total_houses	9
17	total_const_spending	9
18	unemployment_rate	9
19	urban_population	9
20	home_price_index	9
21	MORTGAGE30US	216
22	UNRATE	0

The two clear outliers in terms of missing values are `GDP.x` and `MORTGAGE30US`. I will explore methods of dealing with these two variables in the code below.

First, I will handle the missing data in the `GDP.x` column. As noted above, the St. Louis Federal Reserve (FRED) only reports real GDP on a quarterly basis, meaning the GDP column had missing values for 8 out of the 12 months for each year. Because there were accurate measures of GDP every 4 months, I decided to use seasonal decomposition to impute its intermediate values. In this method, the time series is decomposed into its trend, and seasonal components; then, the intermediate values are imputed using only the trend; finally, the seasonality is added back into the data. This allows for imputed values that smoothly follow the trend of the time series while also adhering to any seasonality present. I also made an indicator column to keep track of which values came from the data and which were imputed.

```r
#Create imputed GDP column
im_preliminary_data <- preliminary_data
im_preliminary_data$imputed_GDP <- impute_GDP(preliminary_data = preliminary_data)

im_preliminary_data <- create_imputed_column(
  imputed_preliminary_data = im_preliminary_data,
  "GDP.x", "imputed_GDP")
```

After imputing the GDP values, I worked on removing low-quality and duplicate columns. Above is the code I used, highlighting which columns I chose to remove. Because I could not verify the values in `MORTGAGE30US` and because it had so many missing values, I decided to remove it completely. I also removed duplicate columns for the federal funds rate, HPI, unemployment rate, and the `house_for_sale_or_sold` column, whose validity I could not properly confirm. After imputation and trimming, I was left with a data set of 18 columns compared to the original's 24. This data set was then saved for further use in the data visualization section.

```r
## This is the data saved for visualization
#Remove low quality columns
#Duplicate post_2003 to keep as intermediate data
trimmed_preliminary_data <- im_preliminary_data

#List of columns to delete
columns_to_delete <- c("MORTGAGE30US", "GDP.y", "interest_rate",
                       "home_price_index", "unemployment_rate",
                       "house_for_sale_or_sold")

#Loop through and set to NULL
for (i in columns_to_delete){
  trimmed_preliminary_data[,i] <- NULL
}

write(paste("Imputed data had", ncol(im_preliminary_data), "columns.",
      "\nTrimmed data has", ncol(trimmed_preliminary_data),
      "columns"), stdout())
```

```
## Imputed data had 24 columns.
## Trimmed data has 18 columns
```

At this point, I worked on creating different data sets for different parts of the modeling process. First, I removed GDP.x and the imputed indicator column to create a data set with only the variables that would be used for modeling.

```r
## This is the data saved for modeling.
# Remove GDP.x and imputed columns to create modeling data set.
modeling_preliminary_data <- trimmed_preliminary_data

#Remove GDP.x from post2003 and only leave imputeGDP, remove imputeYN
modeling_preliminary_data$GDP.x <- NULL
modeling_preliminary_data$imputed <- NULL

write(paste("The visualization data had", ncol(trimmed_preliminary_data),
            "columns.", "\nThe modeling data has",
            ncol(modeling_preliminary_data), "columns"), stdout())
```

```
## The visualization data had 18 columns.
## The modeling data has 16 columns
```

I then renamed the data for ease of reading.

```r
#Rename variables
renamed_preliminary_data <- modeling_preliminary_data

new_names = c("date", "urban_cpi", "fed_funds_rt", "hpi", "build_permits",
              "const_price_idx", "delinq_rt", "house_subsidies", "income",
              "mortgage_rt", "const_unit", "tot_house", "tot_const_spend",
              "urban_pop", "unem_rt", "imputed_gdp")

colnames(renamed_preliminary_data) <- new_names
```

After all of the above preprocessing, I was left with the following missing values:

```r
# Extract NA values & display in stargazer table
final_prelim_na_vals <- extract_na_per_column(renamed_preliminary_data)

final_prelim_na_vals %>% rename(Feature = variable,
                                "NA Values" = na_values) %>%
  stargazer(type = "latex", summary = FALSE, flip = FALSE,
          title = "NA Values by Column After Preprocessing",
          header = FALSE, no.space = TRUE)
```

Table 4: NA Values by Column After Preprocessing

	Feature	NA Values
1	date	0
2	urban_cpi	0
3	fed_funds_rt	0
4	hpi	2
5	build_permits	9
6	const_price_idx	9
7	delinq_rt	9
8	house_subsidies	9
9	income	9
10	mortgage_rt	9
11	const_unit	9
12	tot_house	9
13	tot_const_spend	9
14	urban_pop	9
15	unem_rt	0
16	imputed_gdp	0

After renaming the remaining variables for ease of use and examining the remaining missing values, I decided that the were few enough missing values that it was reasonable to simply drop the problem rows. I then extracted the month and year from the `DATE` column to use in my analyses before saving the data frame to `./data/modelling/` as `modelling.csv`.

```r
# Remove NAs, extract month and year
cleaned_data <- final_preprocessing(intermediate_data = renamed_preliminary_data)
```

Modeling and Analysis Plan

In order to better understand what factors affect the U.S. HPI, I first explored the features and distributions of the data. Then, I employed several supervised and unsupervised learning techniques to uncover the relationships between each explanatory variable and hpi. This section will outline the specific machine learning methods I chose to use as well as the overall roadmap I took when analyzing the data.

Description of Analysis

For this project, I used a multiple linear regression, bootstrapped regression, LASSO regression, ridge regression, decision tree (unpruned and pruned), and a random forest (untuned and tuned) to understand which variables are most important when predicting the U.S. HPI and to evaluate whether or not we can effectively predict hpi given out-of-sample values for our explanatory variables.

Analysis Plan

This project will follow a portion of the standard data science lifecycle. By this point, I have already completed the data mining and cleaning steps. The following sections will cover exploring the data, creating a few visualizations, and then generating models. I began by using individual density plots to assess each variable's distribution and skew. I then used boxplots to better understand any imbalances and to visually identify any outliers. Finally, I made a correlation matrix to quantify the linear correlation of each of the features with the U.S. HPI. After finishing the exploratory analysis, I generated several models (see above). For each model, I examined what variables got flagged as important and (when applicable) calculated the test mean squared error (MSE), root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) to evaluate each model's predictive prowess.

Results

This section will contain the results of my exploratory data analysis and the evaluation metrics for the models I built.

Exploratory Data Analysis

Before building any models, I worked on thouroughly analyzing and exploring the cleaned data set. This section will discuss our exploratory analysis of the data set.

Descriptive Statistics

Below is a summary table for all of the variables remaining in the data set fit for model generation. As mentioned above, many of these measures have very different scales.

```r
#Cleaned data summary stats
stargazer(cleaned_data, type = "latex", summary = TRUE,
          flip = FALSE,  title = "Final Cleaned Data Summary",
          header = FALSE, no.space = TRUE)
```

Table 5: Final Cleaned Data Summary

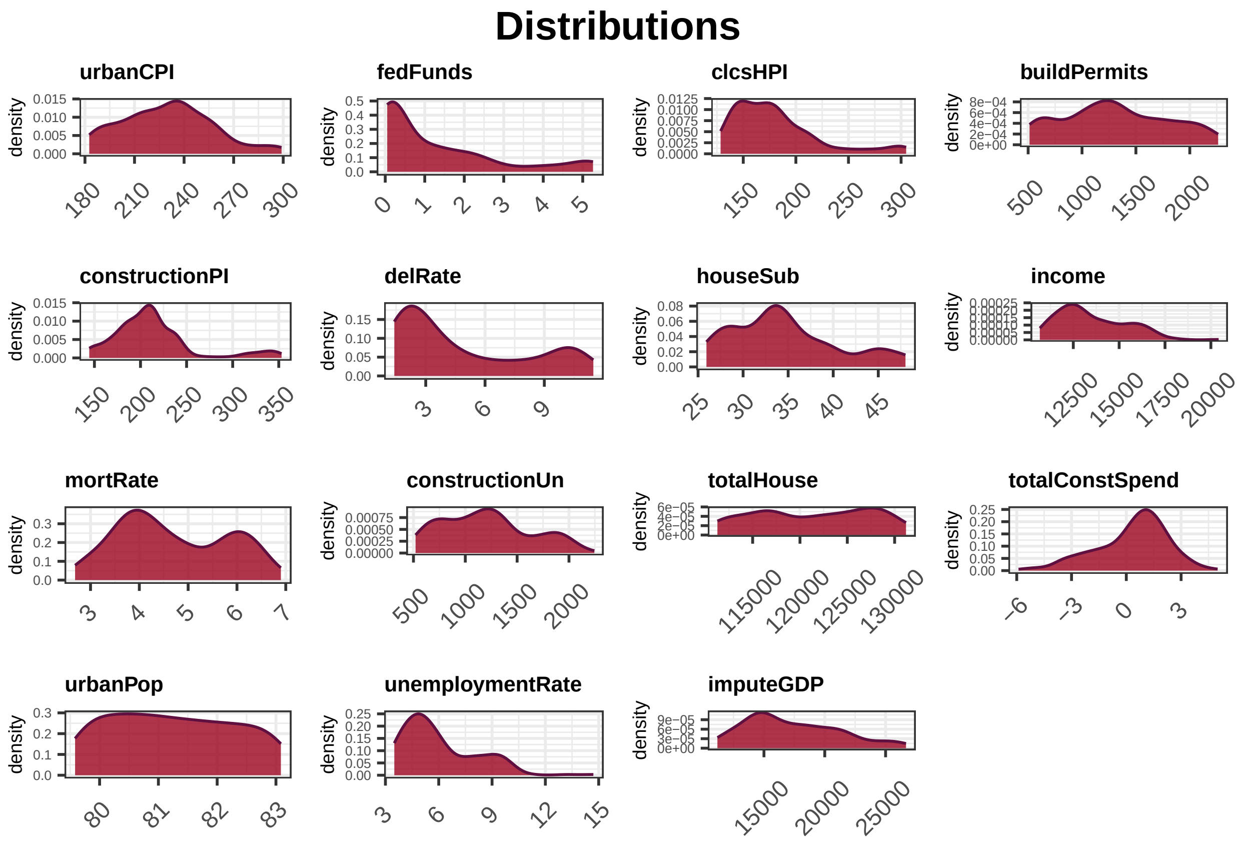
Statistic	N	Mean	St. Dev.	Min	Max
urban_cpi	240	230.023	27.475	182.600	298.990
fed_funds_rt	240	1.302	1.579	0.050	5.260
hpi	240	180.659	41.258	128.461	304.832
build_permits	240	1,309.350	479.881	513	2,263
const_price_idx	240	212.851	44.567	144.400	353.015
delinq_rt	240	4.877	3.305	1.410	11.480
house_subsidies	240	34.677	6.006	25.930	48.021
income	240	13,493.480	1,837.485	10,674.000	20,422.600
mortgage_rt	240	4.683	1.111	2.684	6.900
const_unit	240	1,201.717	423.858	520	2,245
tot_house	240	121,344.400	6,113.869	111,278	131,202
tot_const_spend	240	0.325	1.950	−5.900	5.000
urban_pop	240	81.261	1.055	79.583	83.084
unem_rt	240	6.012	2.034	3.500	14.700
imputed_gdp	240	17,324.820	3,835.256	11,174.130	26,678.540
year	240	2,012.500	5.778	2,003	2,022
month	240	6.500	3.459	1	12

Because of the differences in scales and the fact that there are only 16 features, I opted to create density and boxplot matrix visualizations. These visualizations allowed me to assess each of the 16 variables' spread and identify any potential outliers in the 240 observations.

Data Visualization

After examining the descriptive statistics for the final set of preprocessed variables, I used density plots, box plots, and a correlation matrix to visualize our data. I started by plotting a density plot matrix to examine the distribution of each variable individually. The density plot matrix of all 16 variables is shown below:

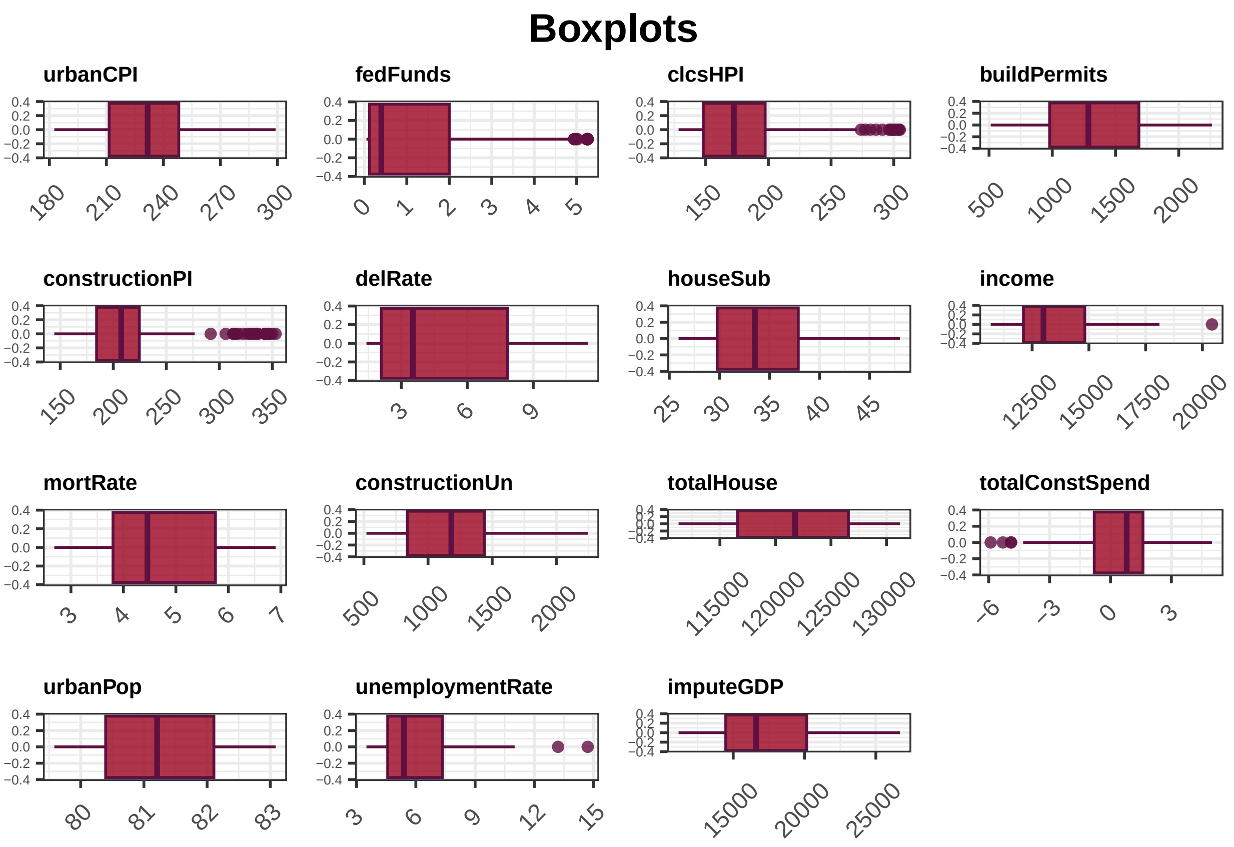
```r
#Load `modelling` data and plot density matrix
m_data <- read.csv("../data/modelling/modelling.csv")
density_plots <- plot_matrix(modeling_data = m_data, type = "density")
```



As is made clear by the matrix, none of the variables follow a perfect normal distribution. Urban CPI (urbanCPI) appeared heavily right skewed, with a gradual increase followed by a steep drop off. CPI values only tend to take on extremely high values in extreme economic conditions so it makes sense that those range of values would appear far less frequently than more "standard" values. The federal funds rate (fedFunds), for similar reasons, was also heavily right skewed. The Fed only pursues aggressive rate hikes to combat serious inflationary pressure. The U.S. HPI (clcsHPI) appeared to be heavily right skewed. This makes sense as the U.S. HPI likely tends to be around a certain range of values except during unusual economic circumstances. Building permits (buildPermits) seemed to be relatively uniform, with a slight bulge around the central value of the distribution. Building permits, on average, are likely relatively consistent over time but may decrease or increase slightly depending on other macroeconomic features like interest rates. The construction price index has a right skewed distribution. The delinquency rate has a somewhat right skewed distribution with a small increase at the tail. This increase may be due to the 2008 financial crisis. Housing subsidies are also clustered towards the left side of the chart. Median U.S. household income is clearly left-skewed, reflecting the well-known income inequality present in our economy. Mortgage rates appear to be bimodal, perhaps representing the two most common values depending on interest rates. Construction units, unemployment rate, and imputed GDP are both left skewed while housing supply and urban population appear to be relatively uniform. Total construction spend appears to be slightly left skewed, but is the closest to a normal distribution.

After looking at the density plots, it is clear that many of our variables are skewed. In order to get a better understanding of if their skewness is an inherent characteristic of the data or if it is caused by data that may not fit within the broader pattern, I looked at boxplots and identified outliers. Below is a matrix of box plots for all 16 variables in the data:

```r
#Plot box plot matrix
boxplots <- plot_matrix(modeling_data = m_data, type = "boxplot")
```



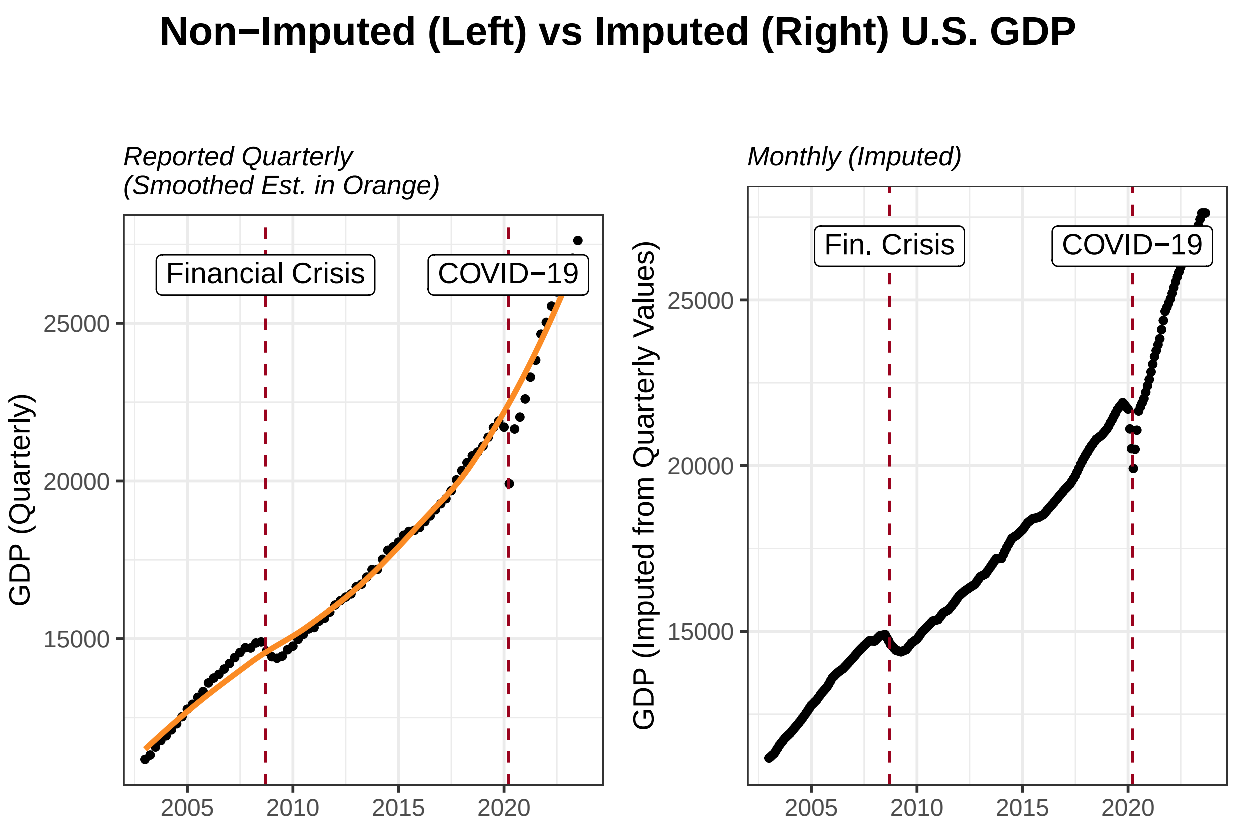
To get a better understanding of the range and outliers of each variable, I examined each of their box plots. Each box plots illustrates the minimum, maximum, 25th to 75th quartiles, and labels any values that fall outside 1.5 times the interquartile range separately. The U.S. HPI and construction price index are the two columns with a high amount of upper outliers. Total construction spending is the only variable with lower outliers.

As discussed above, a potential solution to non-normal distributions would be to remove outliers by either filtering out observations that fall outside of three standard deviations from the mean or that fall outside of 1.5 times the IQR away from the first and third quartiles respectively. Before modeling, I could have tried both methods, re-plotted the box plots and density functions, and decided which method best removes outliers without sacrificing too many observations. After better understanding the distributions and outliers of our variables, we looked at each variable's correlation with our response to understand what variables could be potentially significant. For the purposes of this analysis, I decided to leave the values in because I had already significantly trimmed down the data set.

After looking at the plot matrices, I created some other visualizations to highlight other interesting trends in the data.

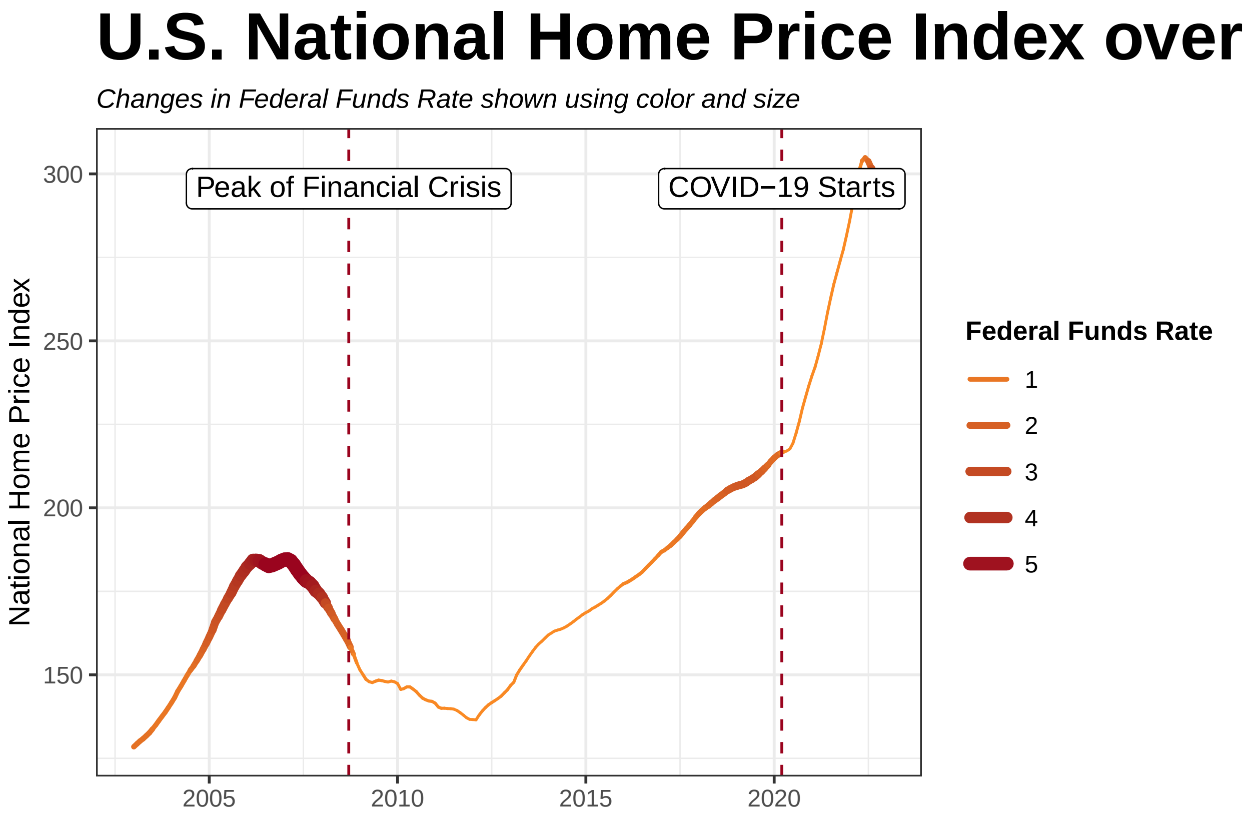
```r
#Pre vs post imputation GDP
v_data <- load_viz_data()
```

```
display <- plot_GDP_side_by_side(viz_data = v_data)
```



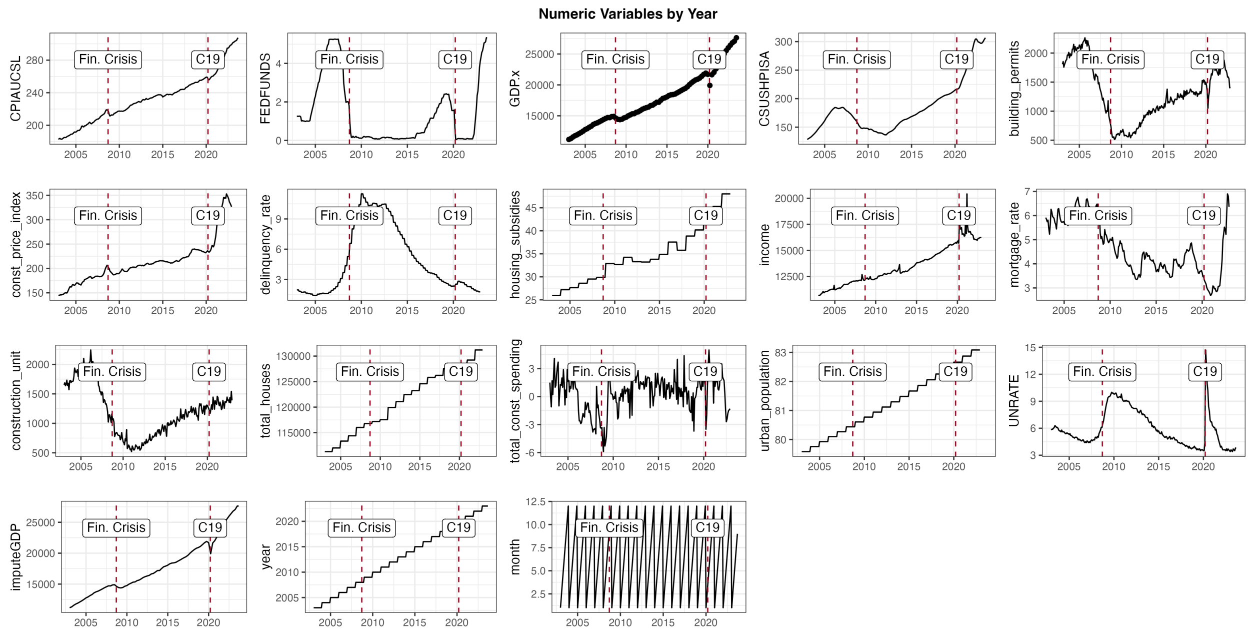
The above chart illustrates the seasonal decomposition process used for imputing U.S. GDP. It's clear that by taking into trend and seasonality, the imputer did a fairly good job of matching the movement of the quarterly reported GDP.

```
plot_hpi_values(viz_data = cleaned_data, date_col = "date", hpi_col = "hpi",
                ff_rt_col = "fed_funds_rt")
```



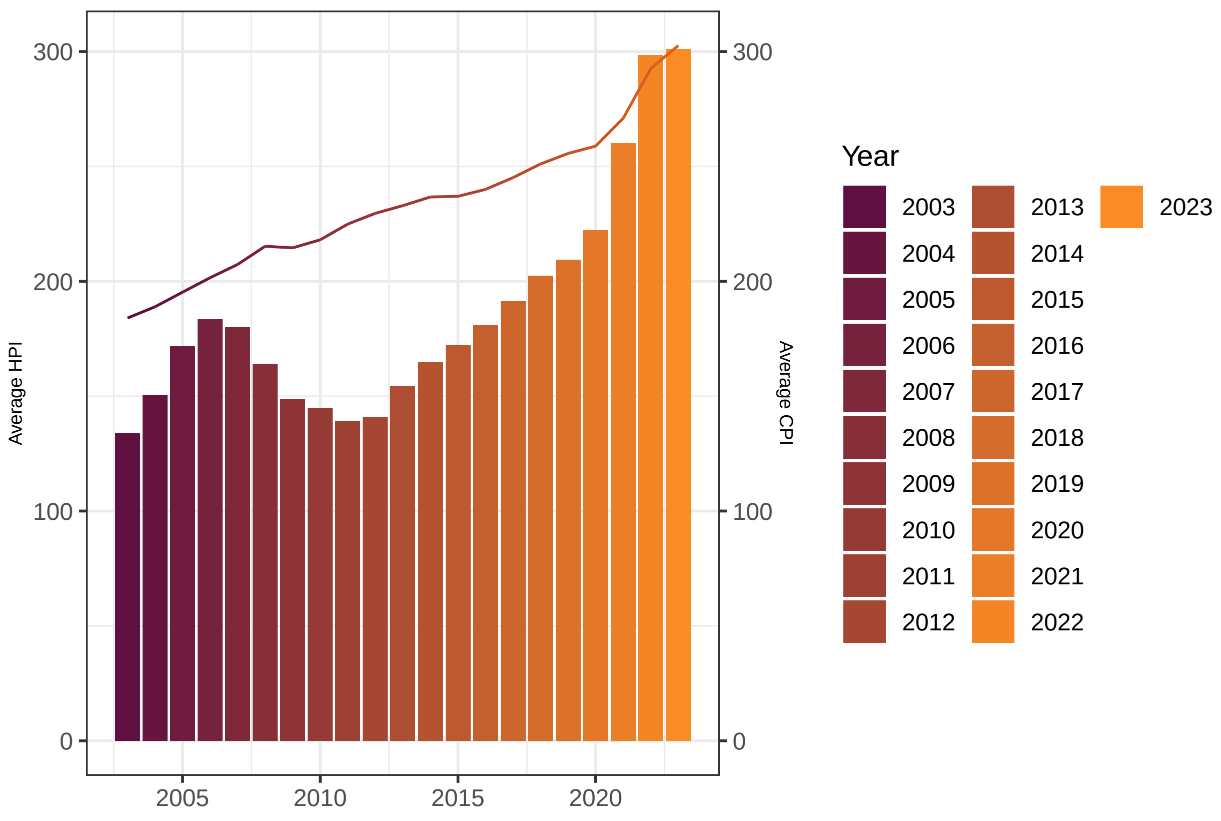
The above chart here illustrates the change in U.S. HPI and federal funds rate over time. The chart shows that the federal funds rate peaked alongside HPI right before the 2008 crash but otherwise did not show any clear paired movement.

```r
## Line plot matrix
plot_line_matrix(v_data)
```



The above plot matrix shows a line plot of all numeric variables over time. There are several somewhat linear relation ships but several that appear highly non-linear.

```r
## Plot hpi and cpi over time
plot_hpi_cpi(v_data)
```

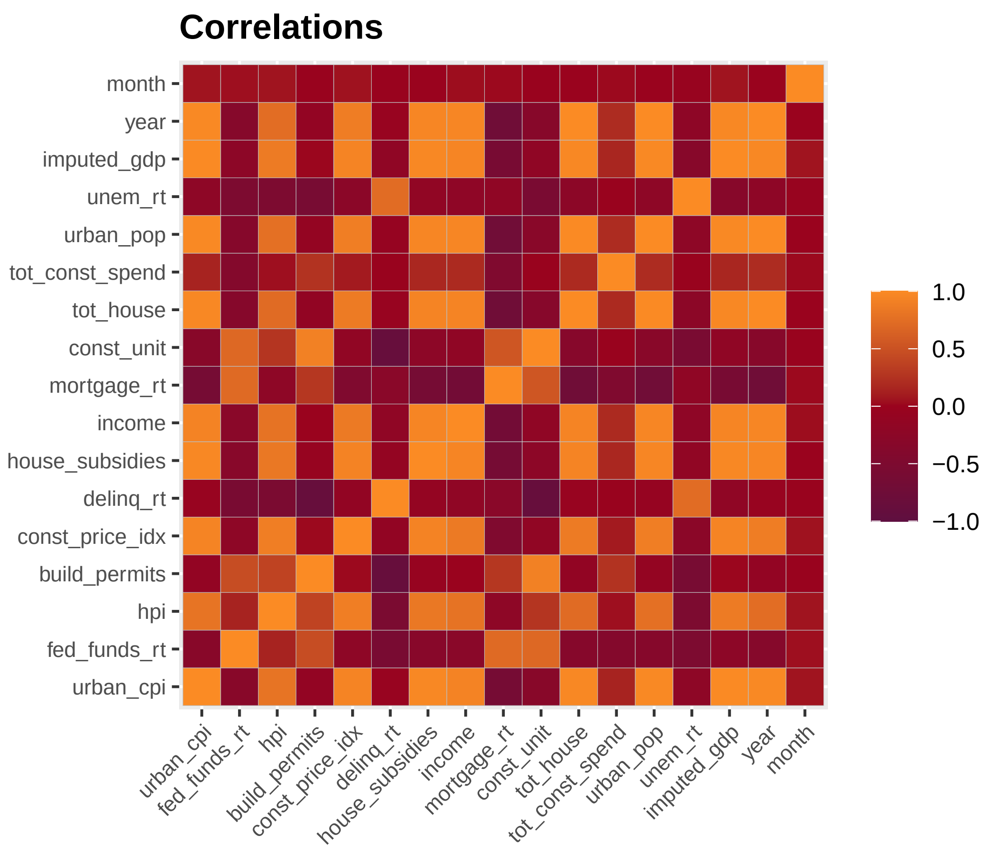


The above plot tracks average HPI and avereage CPI over time. Both metrics seem to follow the same general trend over the same scale.

Relationships

The next part of my variable analysis was to create a correlation matrix to better understand the relationship between the features and the U.S. HPI. Below is the full correlation matrix:

```r
plot_correlations(cleaned_data)
```



Based on the above plot, the U.S. HPI has a strong positive linear relationship with the U.S. consumer price index, U.S GDP, the construction price index, housing subsidies, U.S. median income, housing supply, urban population, and year. The U.S. HPI has a negative relationship with delinquency rates. As time series analysis is outside the scope of this project, it is important to note that these correlations are not completely representative of the relationships in the data as they do not account for time dependence.

That being said, the correlation matrix highlighted several variables of initial interest to keep an eye for when evaluating future models. After visually inspecting the matrix, I filtered out the variables with the strongest correlations with HPI. I considered an absolute correlation of 0.50 to be the threshold for strong correlation. Below is the code for extracting the most correlated variables:

```r
imp_vars <- filter_imp_vars(viz_data = v_data)

imp_vars %>% rename(Name = name,
                    Correlation = hpi) %>%
  stargazer(type = "latex", summary = FALSE, flip = FALSE,
          title = "Variables with an Absolute Correlation >0.50 with HPI",
          header = FALSE, no.space = TRUE, rownames = FALSE)
```

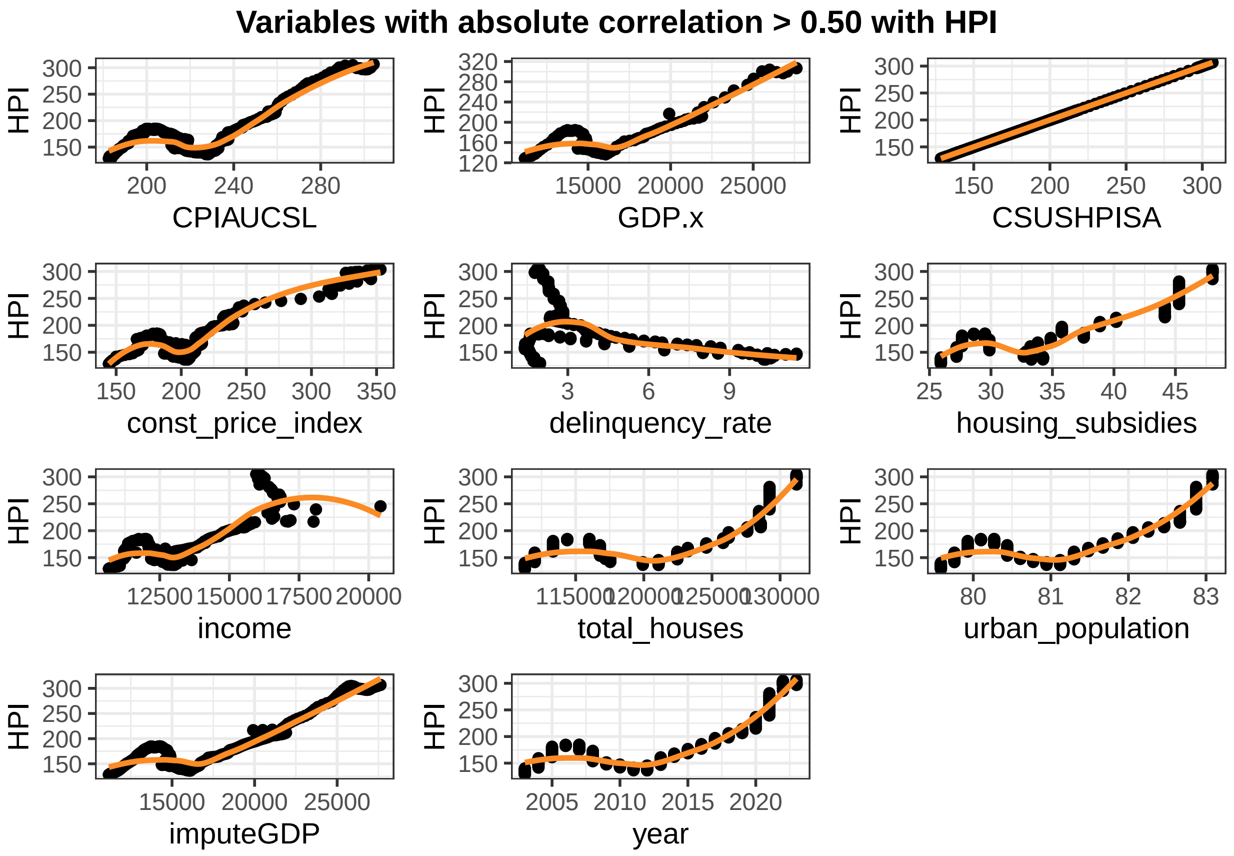
The variables highlighted in the introduction all appear with high correlations. Year having such a high correlation may indicate some temporal effects that could warrant a separate analysis and exploration.

In the final part of my analysis, I created scatter plots to examine the specific relationships between this subset of the variables and HPI. Below is the matrix of scatter plots:

```r
plot_imp_line_matrix(viz_data = v_data,
                     imp_vars = imp_vars)
```

Table 6: Variables with an Absolute Correlation >0.50 with HPI

Name	Correlation
CPIAUCSL	0.799
GDP.x	0.857
CSUSHPISA	1
const_price_index	0.887
delinquency_rate	-0.531
housing_subsidies	0.828
income	0.788
total_houses	0.720
urban_population	0.763
imputeGDP	0.857
year	0.740



Based on the above matrix, it is evident that, although these variables had linear correlations above 0.50 (or below -0.50), their relationships are not necessarily linear. For example, urban population and HPI appear to have a polynomial relationship with each other; though the relationship is clearly a positive one. Similarly, delinquency rates and hpi seem to have a strong negative relationship, though it may not be linear as there is a clear "tail" towards the smaller delinquency rate values. Overall, the most linear relationship appears to be between US GDP and hpi and the consumer price index and hpi.

Modeling

In order to model the behavior of hpi, we chose to explore a variety of models. We first started by splitting the data into a training and testing set.

```r
set.seed(100)
model_data <- read.csv("../data/modelling/modelling.csv")
train_test_split(model_data, 0.70, 0.30)
```

Multiple Linear Regression

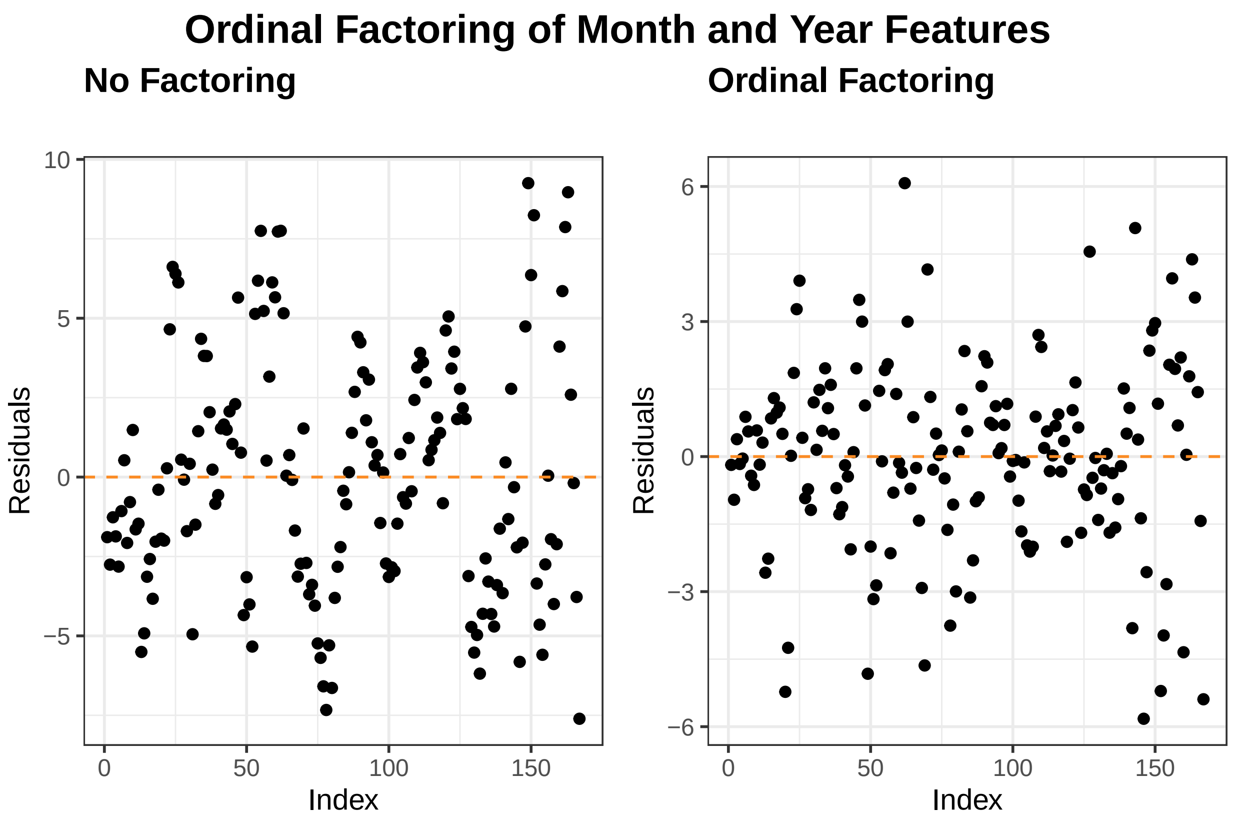
The first model I created was a multiple linear regression. By running a multiple linear regression, I got the predictions of HPI on a test set and examined the coefficients and p-value of each of the explanatory variables to determine variable importance.

I tested two regressions, one with an ordinally encoded year and month feature and one without. The residual plots of the two models are shown below:

```r
model_data_lr <- model_data %>% select(-date)

model_data_lr$factor_year <- factor(model_data_lr$year, order = TRUE,
                               levels = c(unique(model_data_lr$year)))
model_data_lr$factor_month <- factor(model_data_lr$month, order = TRUE,
                               levels = c(unique(model_data_lr$month)))
```

```r
train_test_split(model_data_lr, propTrain = 0.70, propTest = 0.30)
train_lr <- train
test_lr <- test
```

```r
model_no_factor <- lm(clcsHPI ~ . -factor_month - factor_year , train_lr)
model_factor <- lm(clcsHPI ~ . -year -month, train_lr)
preds_nf <- predict(model_no_factor, test_lr)
preds_f <- predict(model_factor, test_lr)

plot_residuals_comparison(model_nf = model_no_factor, model_f = model_factor)
```



Factoring (Right) seems to look more random than no factoring (Left)

From the above plots, it appears that ordinally encoding the year and month features resulted in slightly more random residuals, so I considered that the main model. I then output the ordinal model's coefficient output:

```r
stargazer(model_factor, type = "latex", header = FALSE,
          omit = c("year", "month"), title = "Linear Regression Coefficients",
          no.space = TRUE)
```

Table 7: Linear Regression Coefficients

	Dependent variable:
	clcsHPI
urbanCPI	−0.352
	(0.217)
fedFunds	2.786***
	(0.748)
buildPermits	0.012***
	(0.003)
constructionPI	0.308***
	(0.063)
delRate	−1.795***
	(0.622)
houseSub	0.541
	(0.892)
income	−0.001
	(0.001)
mortRate	1.690**
	(0.806)
constructionUn	0.007**
	(0.003)
totalHouse	0.002
	(0.003)
totalConstSpend	−0.249
	(0.187)
urbanPop	−141.239
	(952.818)
unemploymentRate	2.254***
	(0.436)
imputeGDP	0.012***
	(0.002)
Constant	11,183.710
	(77,201.090)
Observations	167
R^2	0.997
Adjusted R^2	0.997
Residual Std. Error	2.405 (df = 125)
F Statistic	1,214.754*** (df = 41; 125)
Note:	*p<0.1; **p<0.05; ***p<0.01

The model indicated that the federal funds rate, number of building permits, construction price index, delinquency rate, unemployment rate, and imputed GDP as the most significant predictors of the U.S. HPI.

Finally, I recorded the mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) for both models.

```r
errors <- initialize_errors(nrows = 4, row_names = c("MSE", "RMSE", "MAE", "MAPE"))
errors <- append_errors(errors_df = errors, preds = preds_nf,
                        model_name = "lr_nf", test = test_lr,
                        target = "clcsHPI")
errors <- append_errors(errors_df = errors, preds = preds_f,
                        model_name = "lr_f", test = test_lr,
                        target = "clcsHPI")
```

Bootstrap Regression

Bootstrapping takes advantage of the concept of resampling to generate a distribution-free interval for each variable's coefficient. Because bootstrapping is rather computationally heavy, I decided to only run one bootstrap model on the full formula over 100 samples. After running the bootstrap regression, I calculated the 95% confidence interval for each explanatory variable's coefficient to get a better idea of the specific effect each variable may have on hpi. The confidence intervals are displayed below:

```r
mods <- initialize_bootstrap(df = model_data %>% select(-date),
                             n_samples = 100, target = "clcsHPI")

bootstrap_models <- generate_bootstrap_models(mods)

bootstrap_results <- calc_95_perc_int(df = model_data,
                                      exclude_cols = c("date", "clcsHPI"),
                                      mods_boot = bootstrap_models)
```

```r
bootstrap_results %>% rename(Feature = name,
                             "Lower Bound" = lower_bound,
                             "Upper Bound" = upper_bound) %>%
  mutate(`Lower Bound` = round(`Lower Bound`, 5),
         `Upper Bound` = round(`Upper Bound`, 5)) %>%
  stargazer(type = "latex", header = FALSE,
            title = "Bootstrap Regression 95 Percent Confidence Intervals",
            summary = FALSE, flip = FALSE, digits = 5)
```

Table 8: Bootstrap Regression 95 Percent Confidence Intervals

	Feature	Lower Bound	Upper Bound
1	urbanCPI	-0.21437	0.63868
2	fedFunds	2.29996	4.72484
3	buildPermits	-0.00488	0.00659
4	constructionPI	0.32249	0.58713
5	delRate	-6.29629	-3.49725
6	houseSub	1.63695	3.07629
7	income	-0.01239	-0.00054
8	mortRate	-1.27837	3.01867
9	constructionUn	0.00645	0.02046
10	totalHouse	-0.00932	-0.00564
11	totalConstSpend	-0.41983	0.49334
12	urbanPop	-262.66302	-130.90204
13	unemploymentRate	1.00138	3.70763
14	imputeGDP	0.00387	0.01169
15	year	28.38889	51.27943
16	month	-0.57833	0.12285

The above table gives us insight into the specific range of effects each variable may have on hpi. Notably, several variables' 95% confidence range of coefficients contain 0 (specifically urban CPI, building permits, median income, mortgage rates, and total construction spending). Overall, the variables that have 0 in their interval overlap with the less significant variables from the linear regression.

Linear Model Selection

After running a full linear regression model, I ran a cross-validated LASSO regression model and a cross-validated ridge regression model. After training and recording the errors, I examined the selected variables by each algorithm (shown in appendix). The LASSO specifically dropped median income and urban population, adding additional insight into variables that may not be worth further examining in relation to the U.S. HPI. The ridge regression did not significantly reduce any variable's coefficient.

```r
preproc_model_data <- model_df_preprocess(model_data)

train_test_split(preproc_model_data, 0.70, 0.30)
train_lms <- train
test_lms <- test

#Get x_train, y_train, x_test, y_test
x_train <- create_x_matrix(target = "clcsHPI", train_or_test = train_lms)
y_train <- create_y_vector(target = "clcsHPI", train_or_test_df = train_lms)

x_test <- create_x_matrix(target = "clcsHPI", test_lms)
y_test <- create_y_vector(target = "clcsHPI", train_or_test_df = test_lms)


#Get best lasso lambda
lasso_optimal_lambda <- get_best_lambda(x_train = x_train,
                                        y_train = y_train, alpha = 1)
```

```r
preds_lasso <- predict(glmnet(x_train, y_train, alpha = 1),
                      s = lasso_optimal_lambda, newx = x_test)

errors <- append_errors(errors_df = errors, preds = preds_lasso,
                        model_name = "lasso", test = test_lms,
                        target = "clcsHPI")
```

```r
#Get best ridge lambda
ridge_optimal_lambda <- get_best_lambda(x_train = x_train,
                                        y_train = y_train, alpha = 0)

preds_ridge <- predict(glmnet(x_train, y_train, alpha = 0),
                      s = ridge_optimal_lambda, newx = x_test)

errors <- append_errors(errors_df = errors, preds = preds_ridge,
                        model_name = "ridge", test = test_lms,
                        target = "clcsHPI")
```

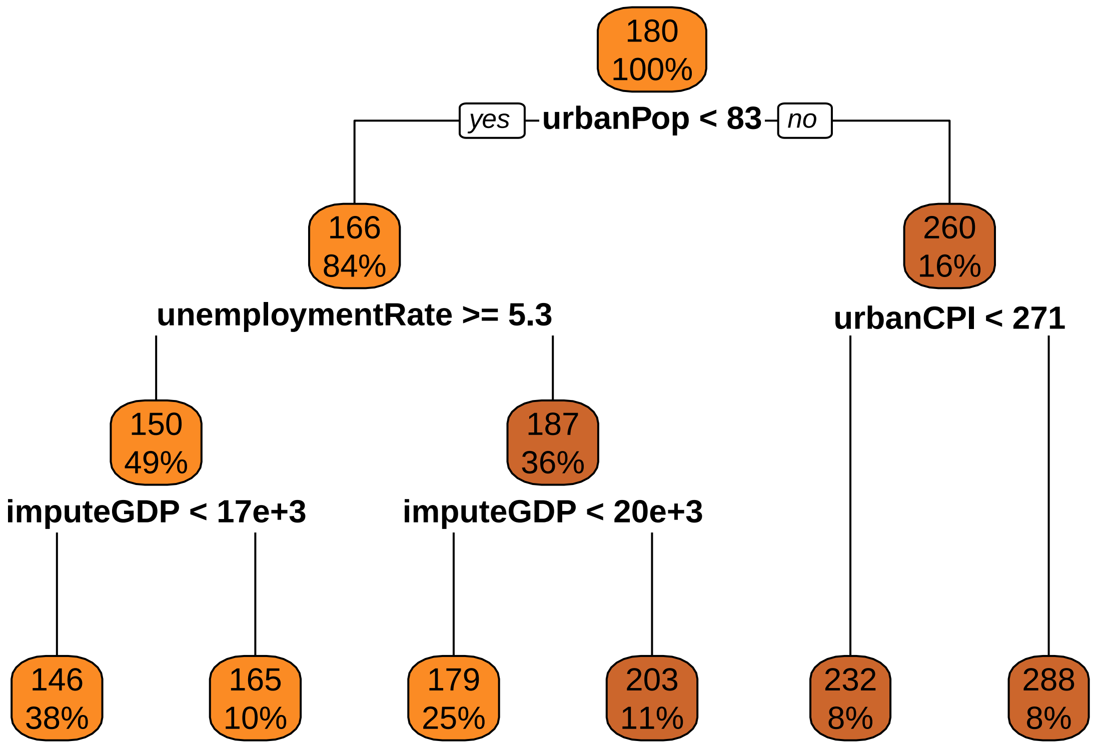
Regression Tree

After examining the linear regression models, I decided to explore non-parametric methods to understand variable importance and develop a predictive model for HPI. The first non-parametric model I tried was a regression tree.

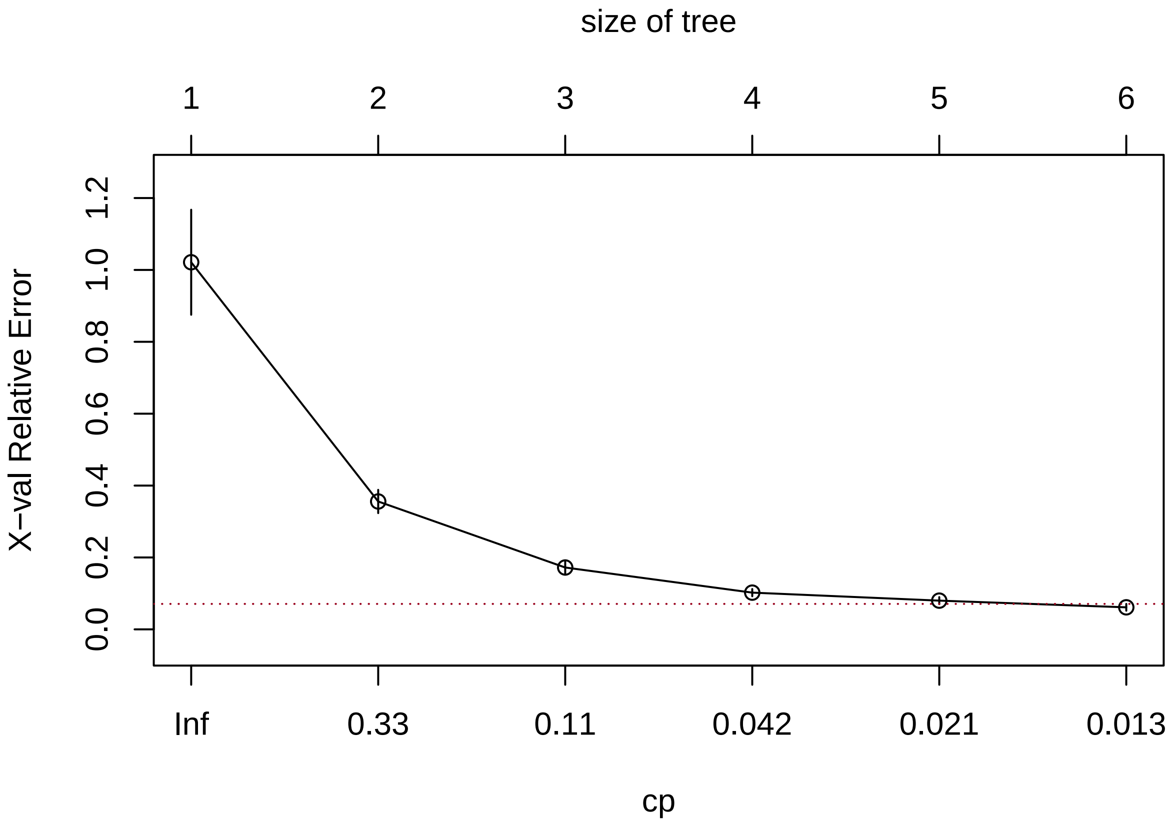
It is possible to prune a regression tree by finding the optimal complexity parameter, minimum observations in each split, and maximum number of terminal nodes. To optimize the model using these values, I first visualized the tree itself and the error as the size of the tree increased, as shown below:

```r
train_dt <- train
test_dt <- test

#full tree
set.seed(100)
model_tree_full <- rpart(clcsHPI~., data = train, method = "anova",
                   control = list(cp = 0, xval = 16))
```

```r
#Create tree
set.seed(100)
model_tree <- rpart(clcsHPI~.,data=train, method = "anova")

#Visualize regression tree
rpart.plot(model_tree, box.palette = year_palette[c(21,15)])
```



```r
#Check cp-size trade off -- maxdepth = 6
plotcp(model_tree, col = palette[2])
```



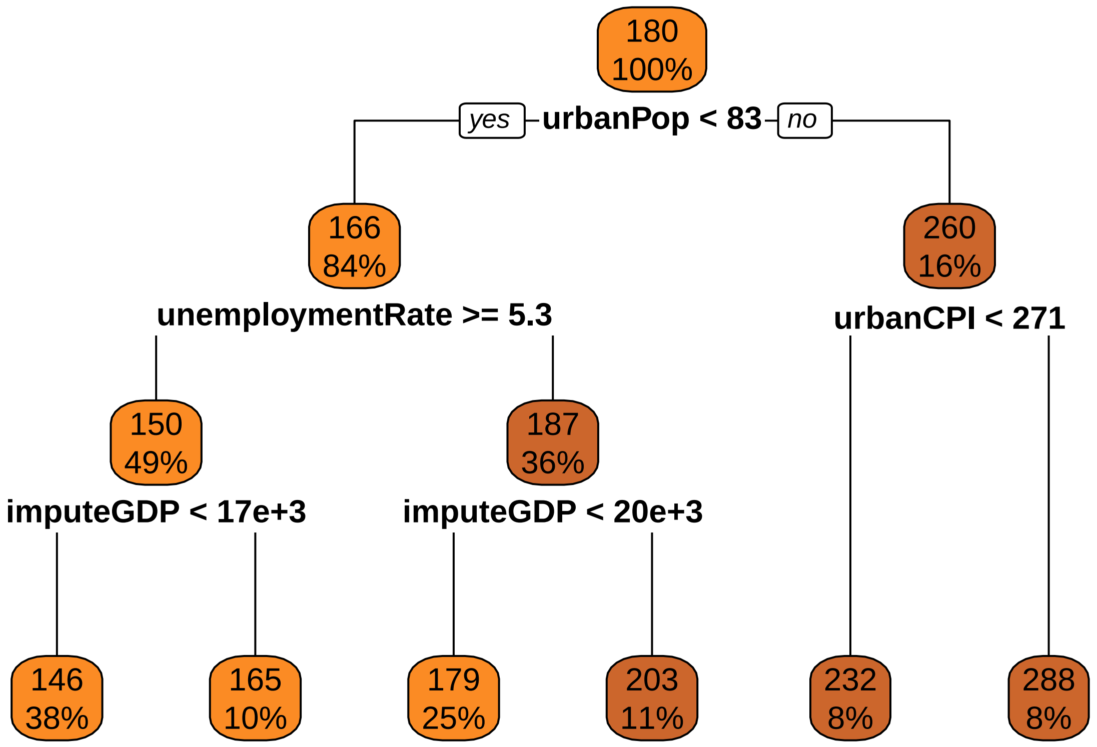
From the above figure, a maxdepth of 6 seemed to be optimal. I confirmed this by creating a regression tree with no cp and graphing its decrease in error as tree size increased.

After confirming the optimal maxdepth value, I performed a grid search to find the minsplit and cp value that minimized the error in the regression tree.

```r
#Grid search for optimal hyperparams

#Define search area
grid_tree <- expand.grid(
  minsplit = seq(5,20,1),
  maxdepth = seq(6,12,1)
)

#Search are for hyperparams
model_tree_list <- get_dt_hyperparameters(grid_tree = grid_tree,
                                          train_df = train_dt)

#Extract best minsplit and cp values
minsplit_optimal <- find_optimum_minsplit(grid_tree = grid_tree,
                                          model_list_tree = model_tree_list)

cp_optimal <- find_optimum_cp(grid_tree = grid_tree,
                              model_list_tree = model_tree_list)
```

After running the grid search, I found that the optimal parameters for our regression tree were a minsplit of 5, cp of 0.01, and maxdepth of 6. I then fit our optimized tree and visualized it, as shown below:

```r
#Plot optimal tree
set.seed(100)
model_tree_optimal <- rpart(clcsHPI ~ ., data = train, method = "anova",
                          control = list(minsplit = minsplit_optimal,
                                         maxdepth = 6, cp = cp_optimal))

#Visualize optimal regression tree
rpart.plot(model_tree_optimal, box.palette = year_palette[c(21,15)])
```



Comparing the two tree visualizations, it is clear that the two trees are identical. This means that the rpart algorithm was able to find the optimal values on its own, likely because the training data was quite small. After pruning our tree, I calculated its evaluation metrics on the test set, as shown in the code block below:

```r
preds_dt_optimal <- predict(model_tree_optimal, test_dt)


errors <- append_errors(errors_df = errors, preds = preds_dt_optimal,
                        model_name = "dt", test = test_dt,
                        target = "clcsHPI")
```

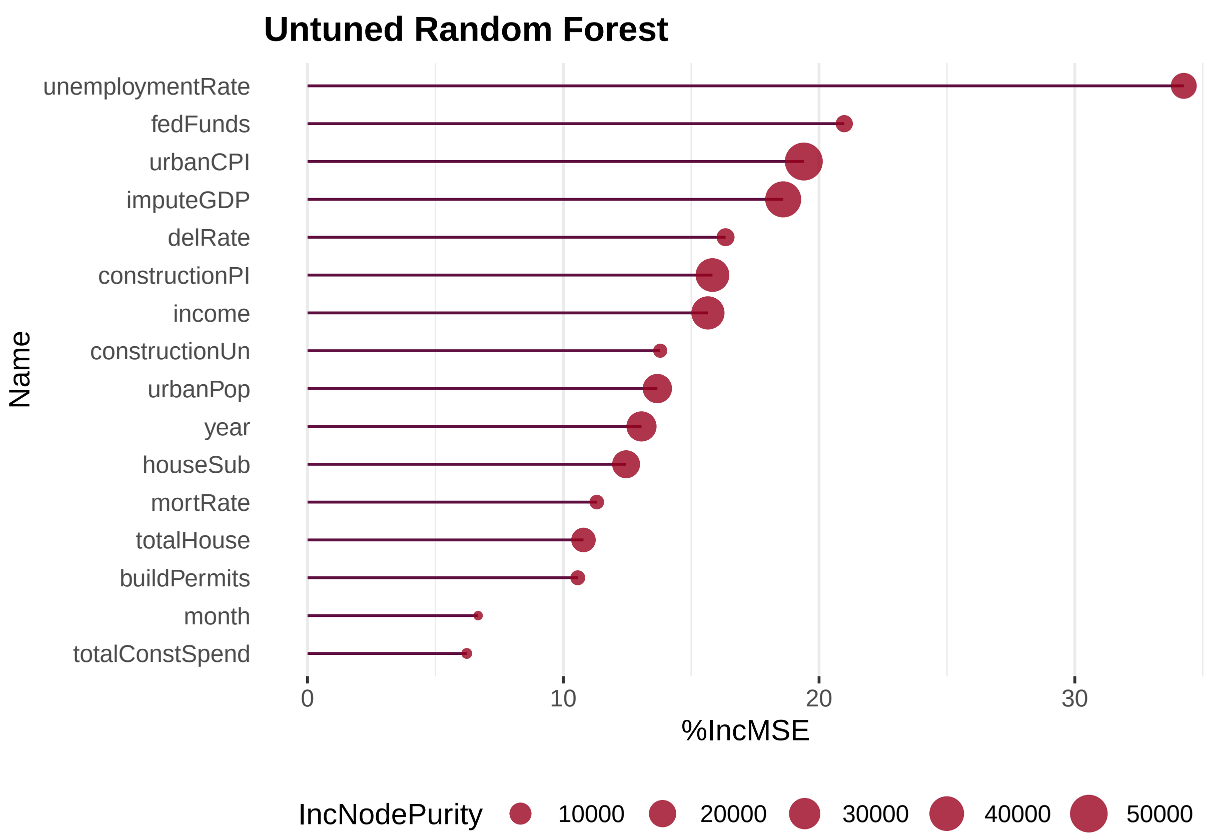
Random Forest

After the regression tree analysis, I wanted to add an ensemble-learning method to my analysis. So, I decided to run a random forest and then tune it. The below code block shows my initial random forest model. After fitting the initial random forest, I extracted the variable importance and visualized it. The figure below illustrates the variable importance of the random forest sorted by percent increase in MSE:

I ranked by percentage increase in MSE because it is the more informative metric. This metric indicates how much the prediction MSE increases when a particular variable is permuted. Based on the above plot, unemployment rate, federal funds, and the urban consumer price index are the three most important variables in predicting HPI.

```r
train_rf <- train
test_rf <- test

# Initial model
set.seed(100)
model_forest <- randomForest(clcsHPI ~ ., data = train, ntree = 1000, importance = TRUE)
```

```r
preds_forest <- predict(model_forest, test_rf)
errors <- append_errors(errors_df = errors, preds = preds_forest,
                        model_name = "untuned_rf", test = test_rf,
                        target = "clcsHPI")

plot_rf_var_imp(model_rf = model_forest, title = "Untuned Random Forest")
```



After constructing the initial random forest, I conducted a tuneRF search and grid search to find the optimal mtry value. The mtry parameter sets how many variables the model will randomly sample as candidates for each split. Optimizing mtry allows the model to manage the trade off between performance and over fitting. The code block below shows the process of constructing the two optimization processes:

```r
#TuneRF
best_mtry_tune_rf <- tune_rf_get_best_mtry(train_df = train_rf, target_col = "clcsHPI",
                     step_factor = 1.5, improve = 1e-5, ntree = 501)
```

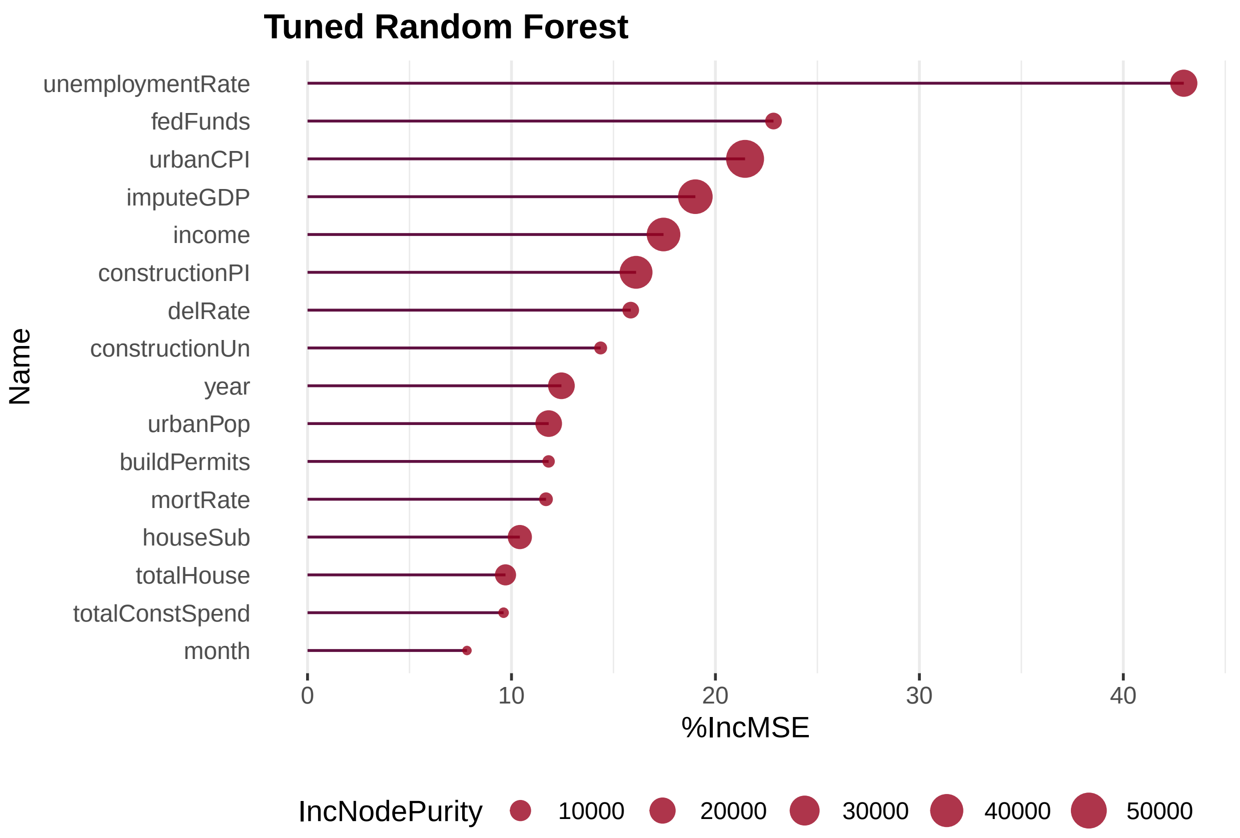
```r
#Grid search
best_mtry_grid <- grid_search_get_best_mtry(train_df = train_rf, num = 10, rpts = 3,
                         mtry_range = c(1:8))
```

As shown above, the optimal mtry to minimize RMSE is 5. After confirming the optimal mtry value, we fit our tuned random forest model, as shown in the code block below:

```r
set.seed(100)
model_forest_tuned <- randomForest(clcsHPI~., data=train_rf,
                           ntree=1000, mtry=best_mtry_grid, importance = TRUE)
```

After fitting the tuned model, I extracted the variable importance values (percent decrease in MSE and increase in node purity) and visualized them, as shown in below:

```r
plot_rf_var_imp(model_rf = model_forest_tuned, title = "Tuned Random Forest")
```



The figure above shows us that unemployment rate, federal funds, and the urban consumer price index were the three most important variables in the tuned random forest (same as the un-tuned model). It is not too surprising that the top variables are still on top, as tuning primarily helps with our model's predictive ability.

```r
preds_forest_tuned <- predict(model_forest_tuned, test_rf)
errors <- append_errors(errors_df = errors, preds = preds_forest_tuned,
                        model_name = "tuned_rf", test = test_rf,
                        target = "clcsHPI")
```

After examining the variable importance plot, I calculated the evaluation metrics for the tuned random forest. There was a noticeable improvement in MSE and RMSE after tuning, which was encouraging for my goal of predicting hpi. The evaluation metrics for the tuned random forest are listed in the code bock below:

Interpretation of Results

Across all of the models, both unemployment rate and the federal funds rate showed up consistently as an influential variable. This could indicate some sustained effect these have on HPI regardless of method of analysis. These results indicate that those variables may be an important metric to track for economists interested in housing markets or in the growth of the housing sector.

In addition to looking at influential variables, I wanted to assess which of the models was best at predicting hpi using out-of-sample observations. The figure below summarizes the model evaluation statistics for each of our predictive models:

```r
errors %>%
  rename("Linear Regression (No Ordinal Encoding)" = lr_nf,
         "Linear Regression (Ordinal Encoding)" = lr_f,
         "Lasso Regression" = lasso,
         "Ridge Regression" = ridge,
         "Decision Tree (Pruned)" = dt,
         "Random Forest (Untuned)" = untuned_rf,
         "Random Forest (Tuned)" = tuned_rf) %>%
  stargazer(type = "latex", flip = TRUE, summary = FALSE,
            header = FALSE,
            title = "Summary of Evaluation Metrics for all Tested Models")
```

Table 9: Summary of Evaluation Metrics for all Tested Models

	MSE	RMSE	MAE	MAPE
Linear Regression (No Ordinal Encoding)	29.311	5.414	4.159	0.023
Linear Regression (Ordinal Encoding)	12.834	3.582	2.403	0.013
Lasso Regression	6.811	2.610	1.898	0.010
Ridge Regression	9.822	3.134	2.128	0.012
Decision Tree (Pruned)	59.473	7.712	6.289	0.034
Random Forest (Untuned)	2.274	1.508	1.107	0.006
Random Forest (Tuned)	2.229	1.493	1.100	0.006

As seen above, the tuned random forest model was by far the best performing model. It outperforms the other models on all three metrics. This makes sense as the economy is a highly irregular system with many moving parts, not lending itself to just linear relationships between variables. Since the data was quite small, I did not run into any computational limitations when running the model.

These results indicate that it may be worth examining more non-linear or ensemble learning models in future examinations. When generalizing to the population, there are clearly complex non-normal variables that require more complex models to analyze effectively.

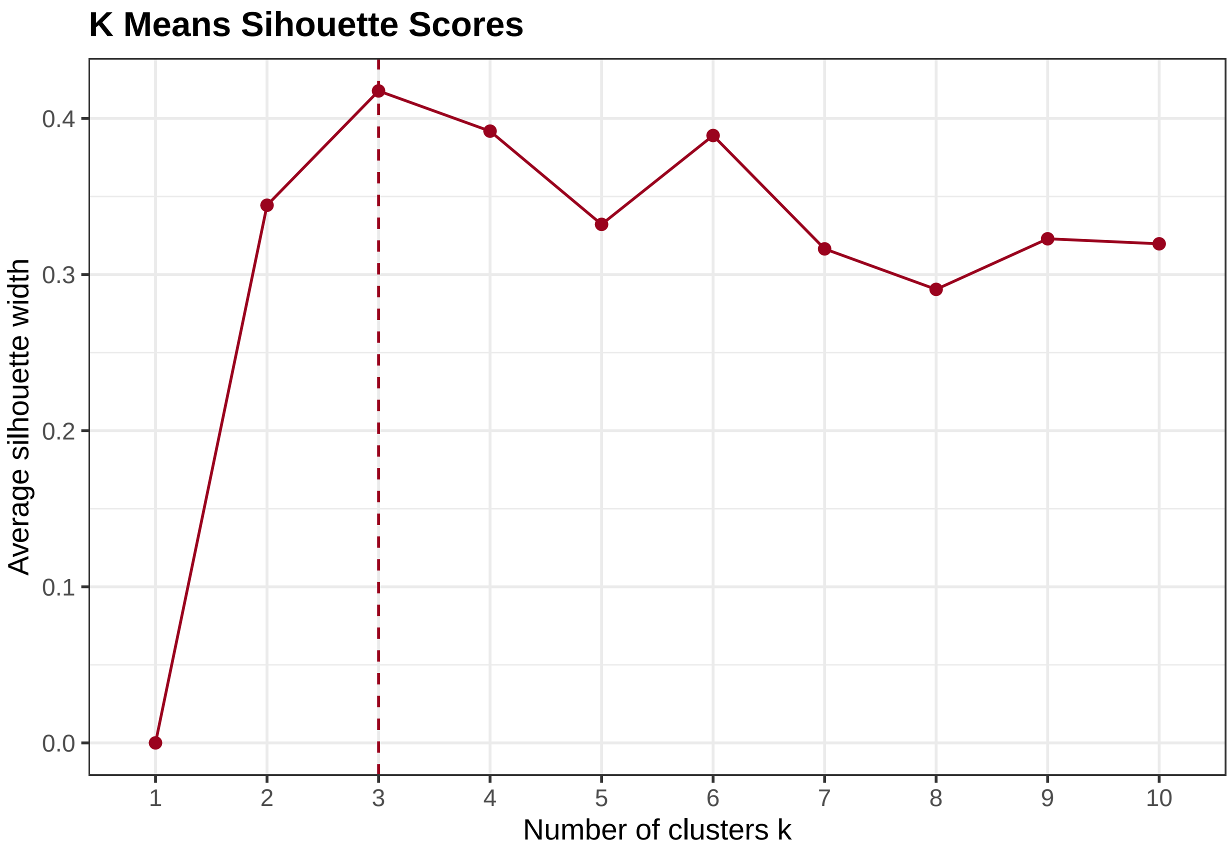
Post-Hoc Analysis

While analyzing economic factors and the U.S. HPI, I wanted to find out HPI trends in similar economic situations. To achieve this, I conducted unsupervised learning based on all variables except the response. In this section, I use K Means and PAM (partitioning around medoids) as unsupervised learning methods.

After creating new data excluding HPI from the data used above, I scaled the data to remove the varying units across variables.

Then, I tried to find the appropriate number of clusters for the K Means algorithm.A k of 3 was obtained using the silhouette score. It is important to note that there is no standardized way to select the number of clusters and my diagnostic plots did not all agree. In this case, I chose to use the silhouette score plot for both of my models since it highlighted the optimal number of clusters directly. The figure below illustrates the silhouette score.

```
df_scaled <- load_preprocess_scale_data()

plot_clustering_diagnostics(df_scaled = df_scaled,
                            method = "silhouette", algorithm = "kmeans")
```

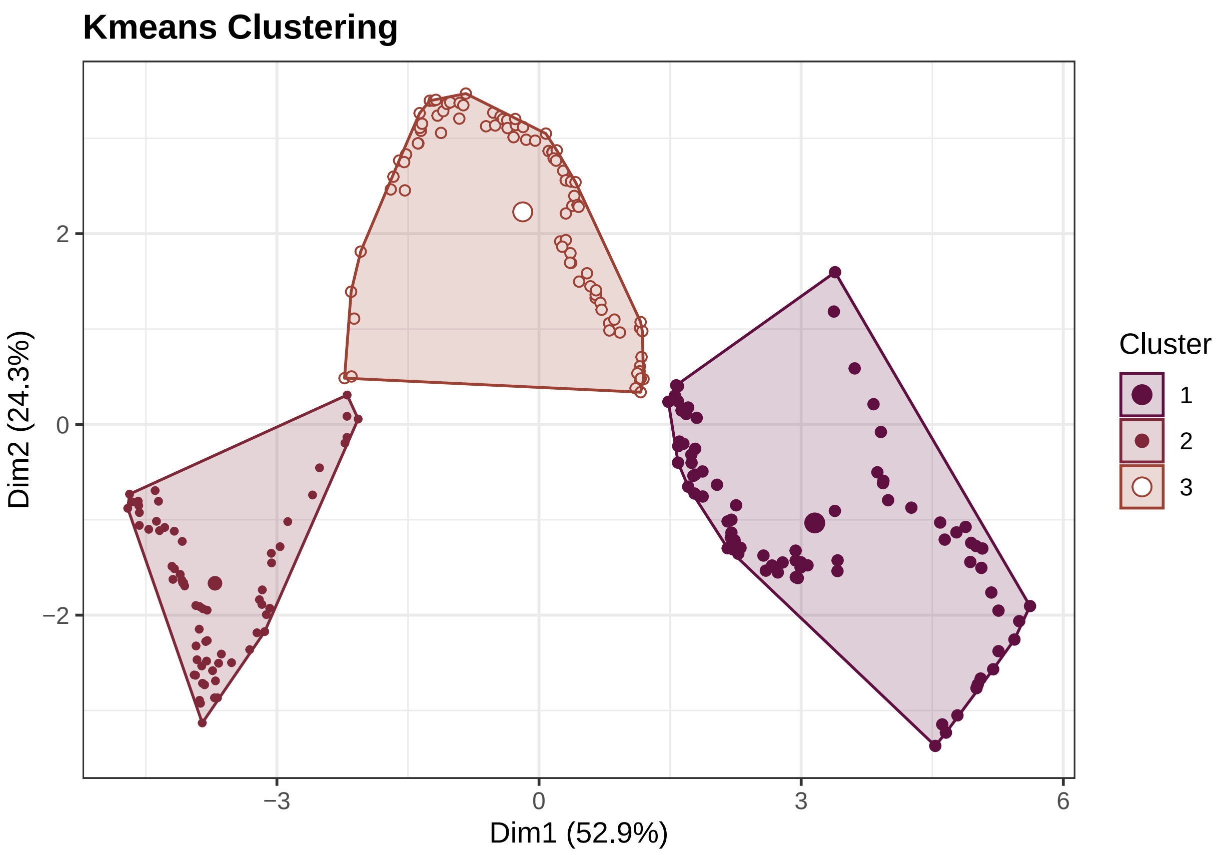


After deciding on the number of clusters, I ran my K Means algorithm.

```
#Fit K Means -- use silhouette score since metrics don't agree
set.seed(100)
model_kmeans <- kmeans(df_scaled, centers = 3)
```

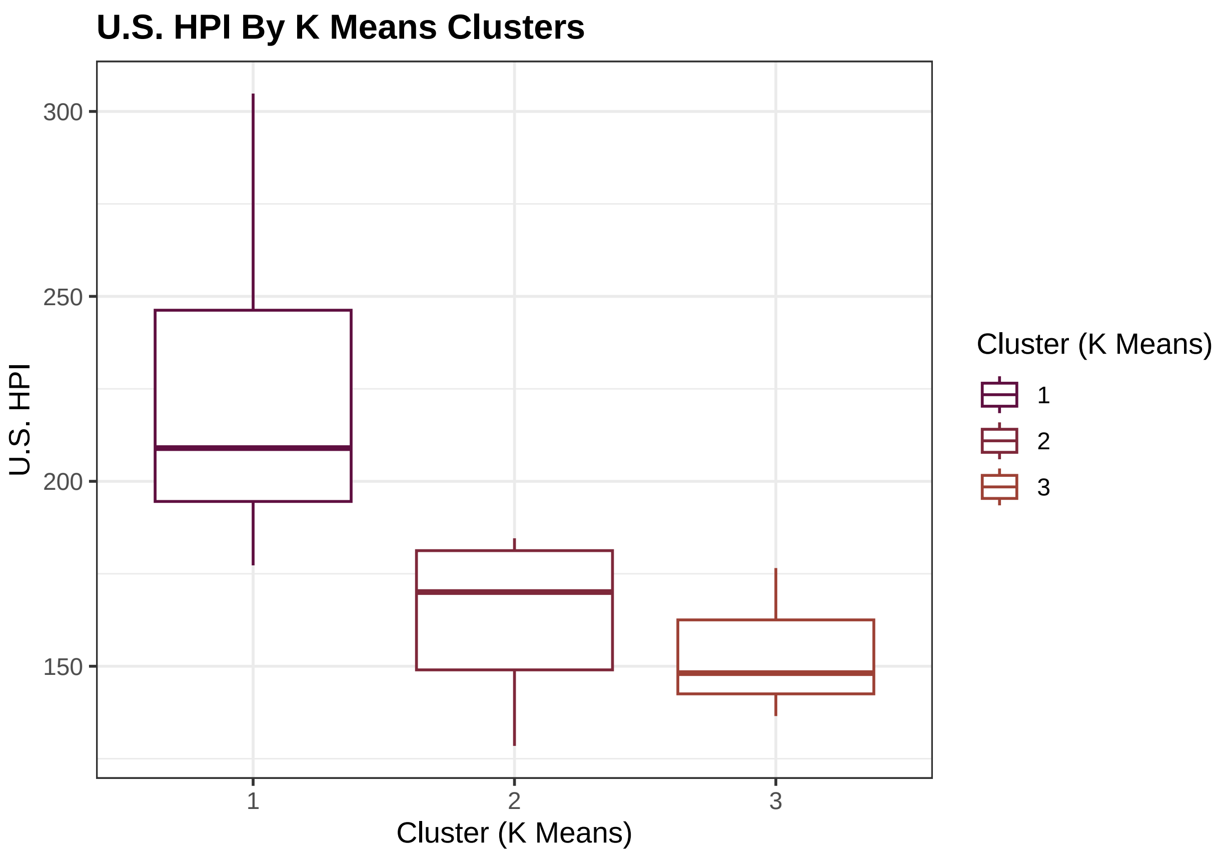
Finally, I visualized the K Means clustering, as seen below:

```
#Plot clusters
plot_clusters(model = model_kmeans, df_scaled = df_scaled, algorithm = "kmeans")
```



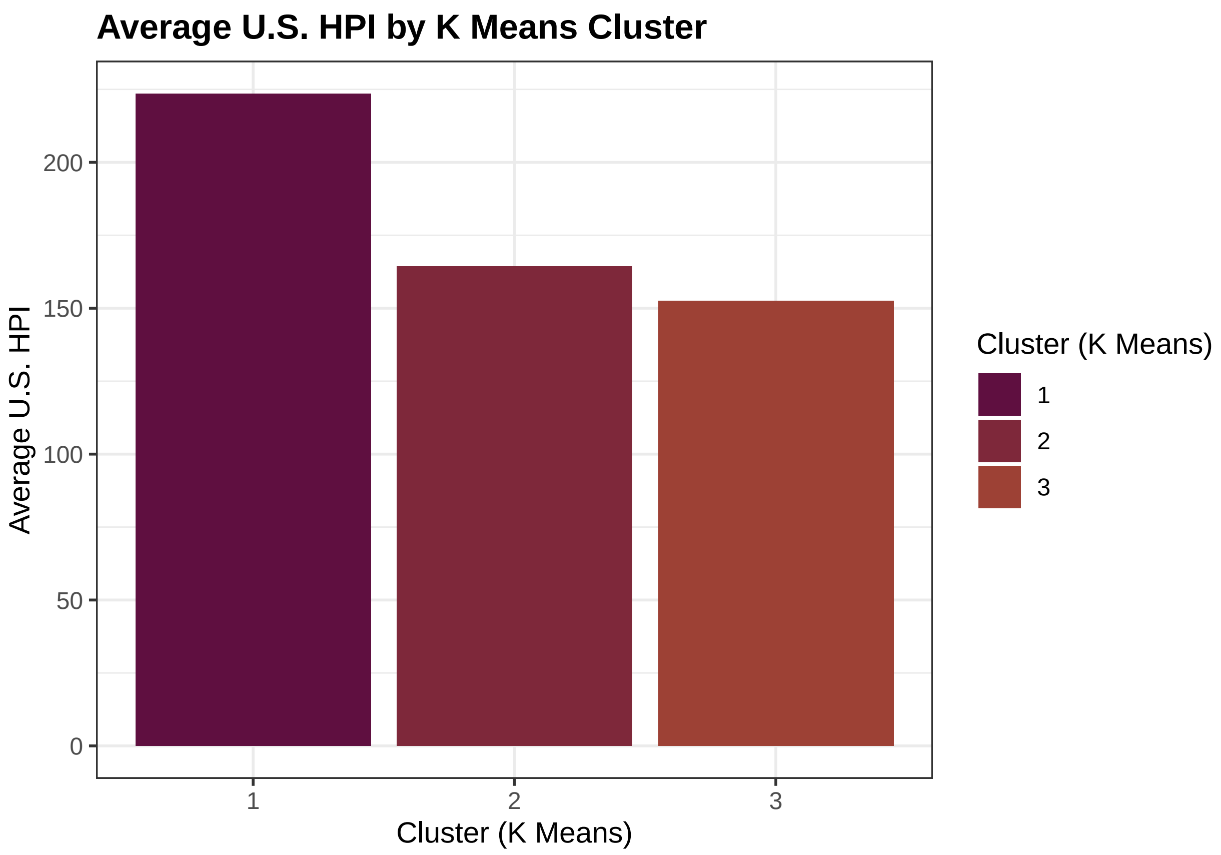
I then plotted box plots of the HPI distribution of the different clusters, as shown below:

```
#Boxplots of median hpi across clusters
plot_median_hpi_across_clusters(model = model_kmeans, algorithm = "kmeans")
```



I also plotted bar plots of the average HPI over the clusters.

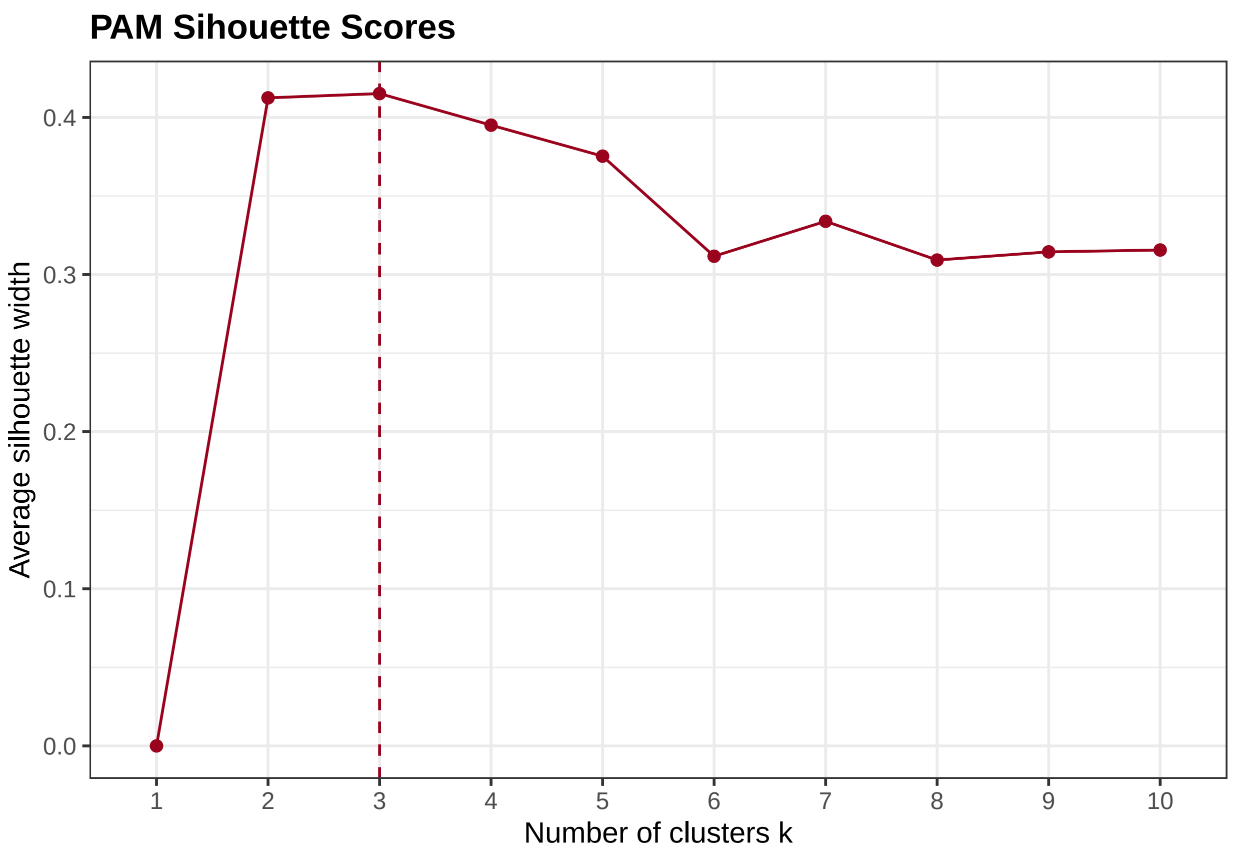
```
plot_bar_median_hpi_across_clusters(model = model_kmeans, algorithm = "kmeans")
```



Although there is some overlap, there is a significant jump in mean HPI between the two clusters. This clustering may be most useful when looking at the upper ends of HPI values in each group as that is where there is the least overlap.

After conducting the K Means clustering, I visualized the silhouette score for the PAM model, as shown below:

```
plot_clustering_diagnostics(df_scaled = df_scaled, method = "silhouette",
                            algorithm = "pam")
```



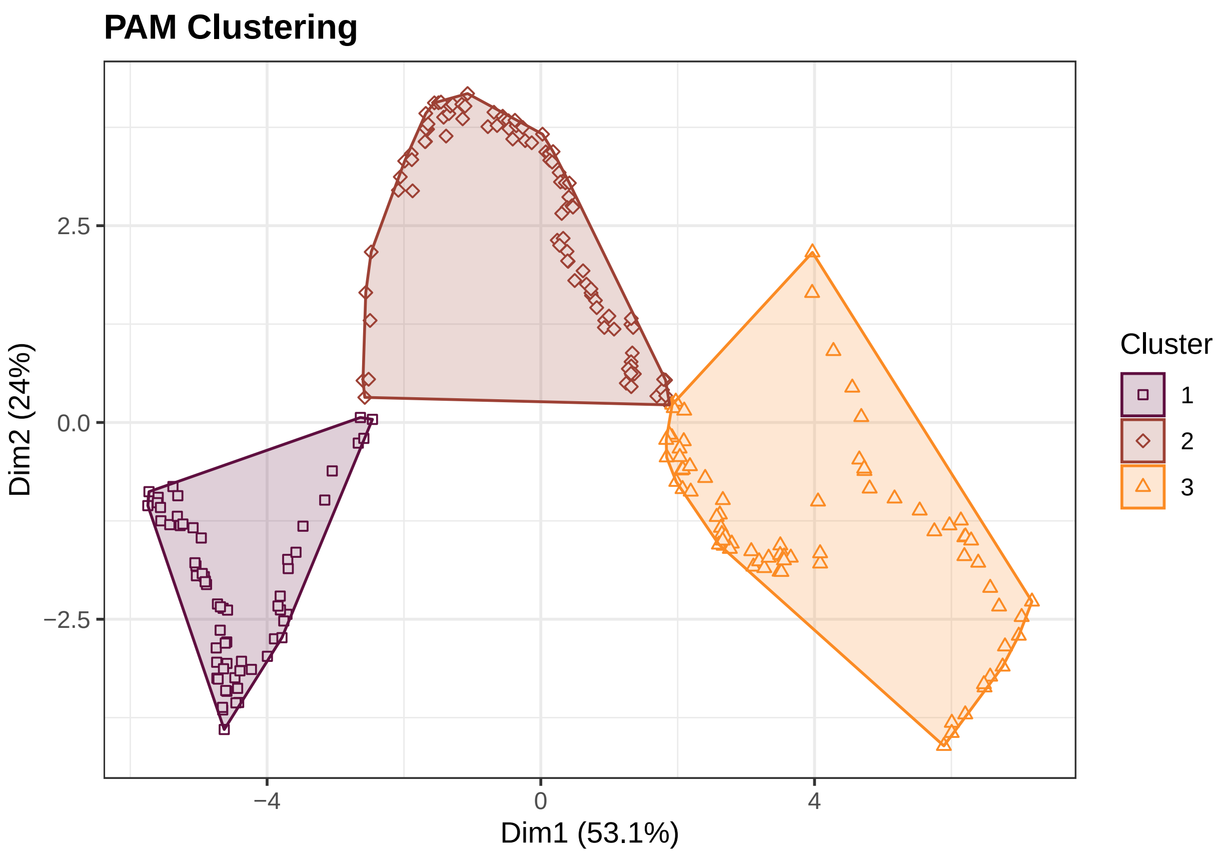
The silhouette plot indicated an optimal k of 3.

After deciding on the number of clusters, I fitted the PAM model.

```
set.seed(100)
model_pam <- pam(df_scaled, stand = T, metric = "manhattan", k = 3)
```

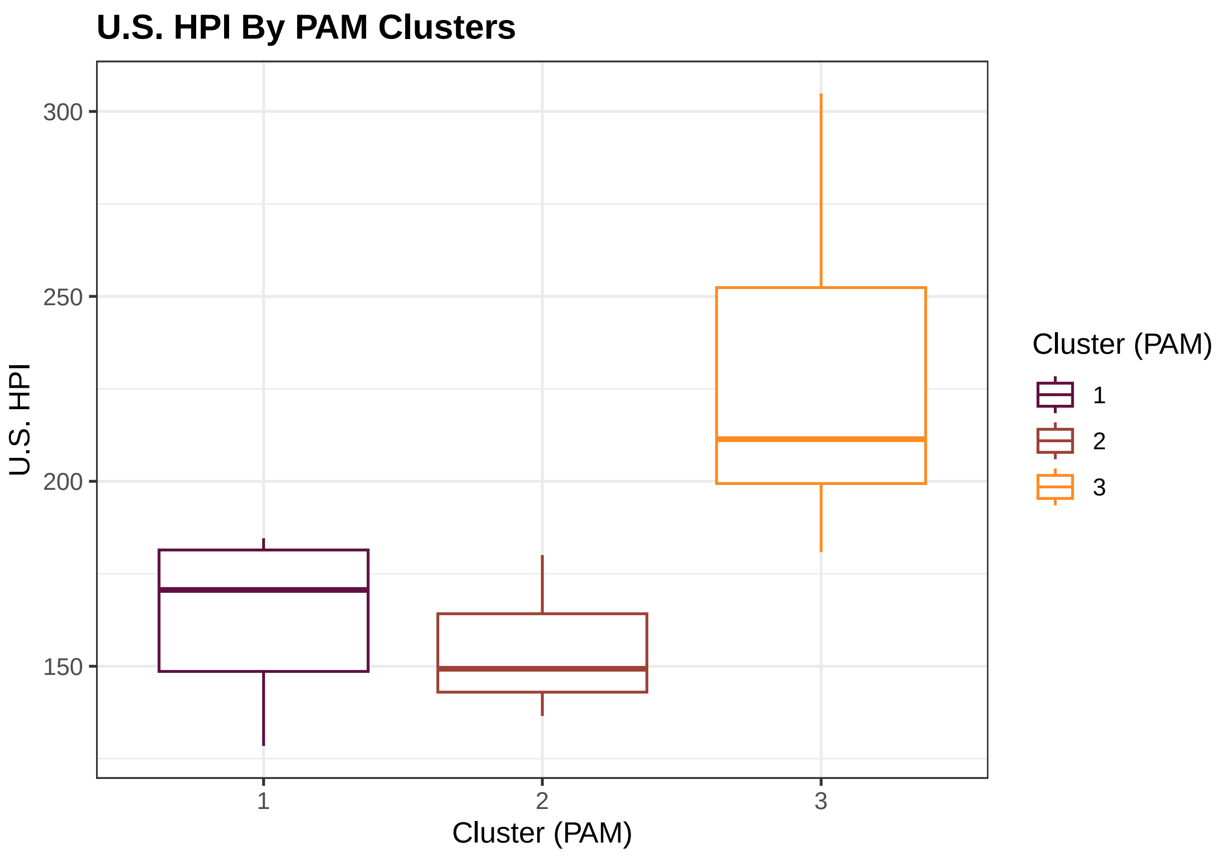
I then visualized the model, as shown below:

```
plot_clusters(model = model_pam, df_scaled = df_scaled, algorithm = "pam")
```

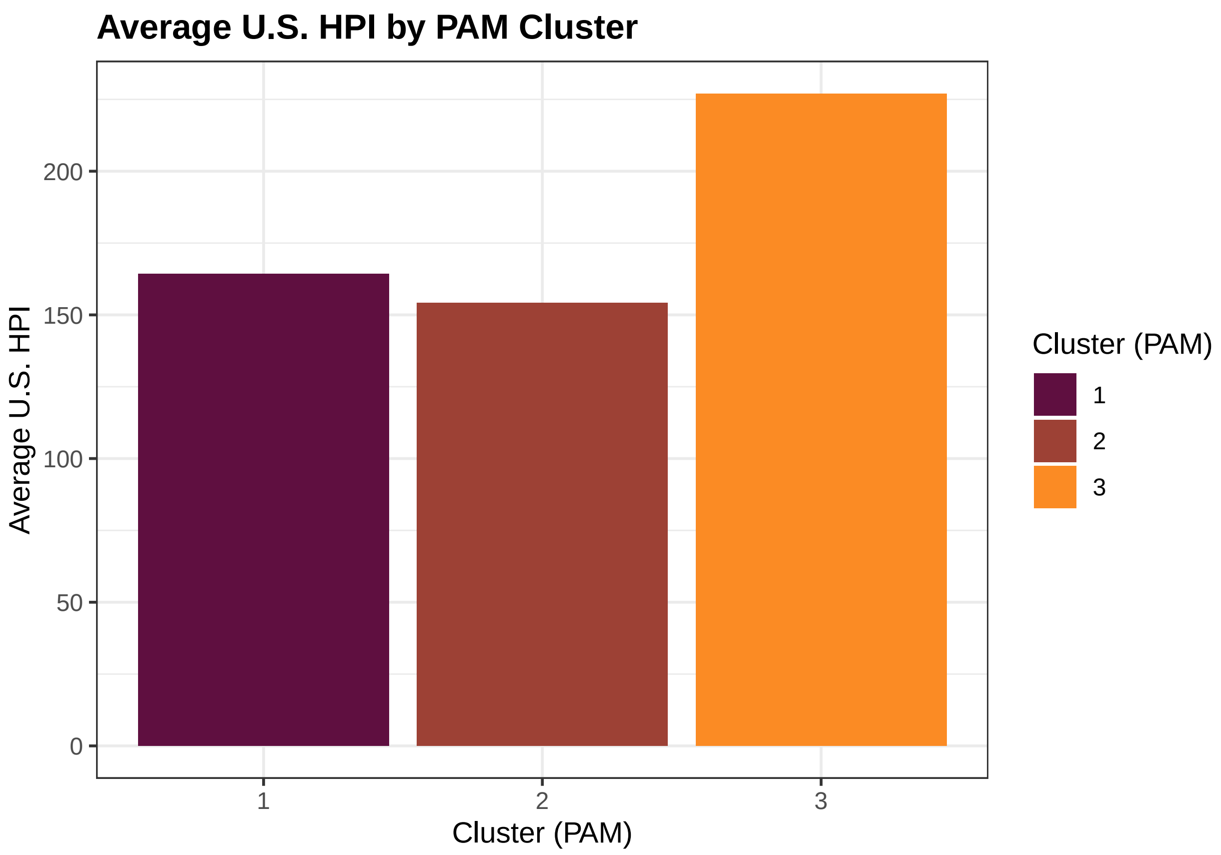


After visualizing the model,I plotted the HPI distribution across clusters with a box plot and the average HPI across clusters using a bar chart.

```
plot_median_hpi_across_clusters(model = model_pam, algorithm = "pam")
```



```r
plot_bar_median_hpi_across_clusters(model = model_pam, algorithm = "pam")
```



Similar to K Means, there is some overlap with a noticeable jump in median HPI between the 3 clusters. This clustering may be most useful when looking at the upper ends of HPI values in each group as that is where there is the least overlap.

When comparing the clustering results with K Means and PAM, only 1 observation was found to be different. As can be seen from the graphs, the shape and order of the 3 clusters were quite similar between the two methods. In the future, it may be useful to look to rejoin the clustered data with their associated dates to allow further economic analysis.

Conclusion

With the goal of better understanding the U.S. economy, this project was focused on modeling the relationship between various economic indicators and the U.S. HPI. Using housing and monthly macroeconomic factors, I tried to determine which variables among different, commonly tracked economic features such as GDP, urban cpi (consumer price index), and unemployment rate would impact HPI the most. Moreover, by attempting various modeling techniques, I tried to find a model that could explain HPI using various economic factors.

U.S. consumer price index, U.S GDP, the construction price index, housing subsidies, U.S. median income, housing supply, urban population, and year. The U.S. HPI has a negative relationship with delinquency rates To get a better understanding about the data, I conducted various exploratory analyses. I summarized the data and looked at the distributions of each variable by visualizing density functions and box plots. I also created a correlation matrix and scatter plots to learn about the overall relationship and correlations between HPI and other variables. Specifically, I found that the U.S. consumer price index, U.S GDP, the construction price index, housing subsidies, U.S. median income, housing supply, urban population, year, and delinquency rates had strong linear relationships with HPI.

After exploring the data, I created several models to understand variable importance and to predict HPI. I created a multiple linear regression, bootstrapped regression,a LASSO and ridge regression, regression tree, and a random forest. For each predictive model, I examined variable importance and calculated its mean absolute error, mean absolute percent error, mean squared error, and root mean squared error. I found various combinations of features affecting HPI. Among them, unemployment rate and the federal funds rate were included in all models. In terms of predictive ability, I found that the tuned random forest algorithm had the smallest RMSE among the modeling methods implemented, making it the most appropriate modeling technique for predicting HPI given other macroeconomic factors. Finally, as a post-hoc analysis, I ran a K Means and PAM clustering model. Through these unsupervised learning results, I clustered economic situations and confirmed that HPI was distributed noticeably differently depending on these clusters. This finding supports the hypothesis that economic factors have a significant relationship with HPI.

Contrary to the existing expectation that important variables in predicting HPI could be clearly identified through modeling, the results that were obtained in the models ended up being different depending on the model. The reason why this outcome occurred could be based on the characteristics of each model. While multiple linear regression analysis detects linear relationships between variables, regression trees or random forests can capture non-linear relationships. Additionally, in the case of bootstrap regression, the importance of predictor variables may vary depending on the bootstrapped sample.

This introductory exploration of analyzing the dynamics of the U.S. HPI can serve as a stepping stone to better understand the US real estate market and, by extension, the economy. It may help people develop their investment strategy and make financial decisions, as anticipated fluctuations in housing prices can impact decisions related to home purchases, sales, and rentals. HPI predictions can also contribute to evaluating the health of the real estate market, enabling governments and central banks to formulate policies to maintain stability in the overall economy.

Limitations

The real estate-related data I used in the analysis was time series data representing monthly indicators. Since the dependent variable, HPI, is likely affected by the passage of time, it would have been useful to apply time series modelling to the data. This was, however, outside of the scope.

Furthermore, in the original data sets, there were several excluded variables due to their large number of missing values. In general, there was a very low volume of data to work with. Having more observations, whether that was via daily observations or a larger time window of consideration would have likely made the models much more robust.

Suggestions for Future Research

For this research project, I considered all of the variables as numeric variables. With numeric variables, I could apply several analysis methods such as regression trees, random forest, and multiple linear regression. In future research, changing the class of some variables to a categorical class can open up new possibilities for analysis and modeling. This could allow for completely new results compared to the original method. Integrating the role of technology and its impact on the real estate market and how that affects the HPI variable may also prove particularly insightful.

Works Cited

Liberto, D. (2023, August 29). Understanding the House price index (HPI) and how it is used. Understanding the House Price Index (HPI) and How It Is Used. https://www.investopedia.com/terms/h/house-price-index-hpi.asp

Rosen, P. (2023, August 11). The US housing market hits a record value of $47 trillion as the inventory shortage fuels a price boom. Business Insider. https://markets.businessinsider.com/news/commodities/housing-market-inventory-shortage-home-prices-value-real-estate-property-2023-8.

Code Appendix

The full code and analysis process can be found on the project's public GitHub repository.

Original Column Names

```r
write(colnames(modeling_preliminary_data), stdout())
```

```
## DATE
## CPIAUCSL
## FEDFUNDS
## CSUSHPISA
## building_permits
## const_price_index
## delinquency_rate
## housing_subsidies
## income
## mortgage_rate
## construction_unit
## total_houses
## total_const_spending
## urban_population
## UNRATE
## imputed_GDP
```

Renamed Column Names

```r
write(colnames(renamed_preliminary_data), stdout())
```

```
## date
## urban_cpi
## fed_funds_rt
## hpi
## build_permits
## const_price_idx
## delinq_rt
## house_subsidies
## income
## mortgage_rt
## const_unit
## tot_house
## tot_const_spend
## urban_pop
## unem_rt
## imputed_gdp
```

Lasso Coefficients

```r
coef(glmnet(x_train, y_train, alpha = 1), s = lasso_optimal_lambda)
```

```
## 45 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)      -20.523994183
## urbanCPI           0.002599805
## fedFunds           2.748487470
## buildPermits       0.011278398
## constructionPI     0.372821451
## delRate           -2.546486339
## houseSub           0.552599952
## income             .
## mortRate           1.148669440
## constructionUn     0.006226760
## totalHouse         .
## totalConstSpend   -0.127509104
## urbanPop           .
## unemploymentRate   0.790487586
## imputeGDP          0.004534826
## year.L             .
## year.Q             1.453638527
## year.C            25.699977726
## year^4            -0.146399974
## year^5             .
## year^6             4.651611648
## year^7            -2.675087679
## year^8            -6.169210714
## year^9             .
## year^10            7.777334160
## year^11            0.322591132
## year^12            1.804398522
## year^13            6.018569346
## year^14            0.146362439
## year^15           -3.836262493
## year^16            1.955781782
## year^17            .
## year^18            .
## year^19           -0.628543017
## month.L            1.687724078
## month.Q            1.719489514
## month.C            0.780750868
## month^4           -0.912337736
## month^5           -0.698121724
## month^6           -0.249070075
## month^7           -0.217514397
## month^8           -0.023331159
## month^9            0.029186811
## month^10          -0.285266572
## month^11           0.041986505
```

Ridge Coefficients

```r
coef(glmnet(x_train, y_train, alpha = 0), s = ridge_optimal_lambda)
```

```
## 45 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)      -3.201699e+02
## urbanCPI          1.838138e-01
## fedFunds          1.807790e+00
## buildPermits      9.182184e-03
## constructionPI    1.760304e-01
## delRate          -1.322983e+00
## houseSub          8.710893e-01
## income            1.292202e-03
## mortRate          1.198345e+00
## constructionUn    8.111481e-03
## totalHouse        4.618046e-04
## totalConstSpend  -2.369552e-01
## urbanPop          3.376740e+00
## unemploymentRate -7.459935e-01
## imputeGDP         1.412067e-03
## year.L            1.433803e+01
## year.Q            2.472300e+01
## year.C            3.554295e+01
## year^4           -2.281180e+00
## year^5            6.417949e+00
## year^6            4.952960e+00
## year^7           -6.340336e+00
## year^8           -5.192504e+00
## year^9           -2.109013e+00
## year^10           3.949263e+00
## year^11          -8.927535e-01
## year^12           8.003859e-01
## year^13           5.570331e+00
## year^14           1.211916e+00
## year^15          -4.093414e+00
## year^16           1.309796e+00
## year^17           8.575940e-02
## year^18          -7.044687e-01
## year^19          -6.135018e-01
## month.L           3.880782e+00
## month.Q           1.171167e+00
## month.C           5.655972e-01
## month^4          -2.291435e-01
## month^5          -5.476888e-01
## month^6          -6.344924e-01
## month^7           1.718766e-01
## month^8          -4.986018e-01
## month^9           3.983831e-01
## month^10         -7.345075e-01
## month^11          8.897317e-02
```

K Means Output

```r
print(model_kmeans)
```

```
## K-means clustering with 3 clusters of sizes 84, 67, 89
##
## Cluster means:
##     urbanCPI   fedFunds buildPermits constructionPI    delRate   houseSub
## 1  1.0614853 -0.1434934    0.2764971      0.9654143 -0.5768878  1.1199768
## 2 -1.1857070  1.1220765    0.9811993     -0.9424235 -0.8444416 -1.1119357
## 3 -0.1092404 -0.7092773   -0.9996192     -0.2017126  1.1801816 -0.2199815
##       income   mortRate constructionUn totalHouse totalConstSpend   urbanPop
## 1  1.1621169 -0.6784573     0.08953478  1.1139509      0.22910761  1.1407144
## 2 -1.0600310  1.2343504     1.23246052 -1.2411350     -0.24294693 -1.2133768
## 3 -0.2988286 -0.2888884    -1.01231210 -0.1170319     -0.03334377 -0.1631884
##   unemploymentRate  imputeGDP       year         month
## 1       -0.5742222  1.1425168  1.1248921  7.930164e-18
## 2       -0.4102375 -1.0639329 -1.2437096 -7.550560e-02
## 3        0.8507930 -0.2773922 -0.1254201  5.684130e-02
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
##  [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 593.1491 343.2704 399.6618
##  (between_SS / total_SS =  65.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

PAM Output

```r
print(model_pam)
```

```
## Medoids:
##       ID    urbanCPI   fedFunds buildPermits constructionPI    delRate
## [1,]  27 -1.34387949  0.8415364    1.5684095     -0.9615128 -1.0458408
## [2,] 126  0.08814872 -0.7675743   -0.7655022     -0.0909079  1.3443067
## [3,] 200  0.94762042  0.5247823    0.4556337      0.4835118 -0.7372395
##         houseSub     income   mortRate constructionUn totalHouse totalConstSpend
## [1,] -1.1698110 -1.2107204  1.1206700      1.3808488 -1.3087376      0.08994007
## [2,] -0.2438116 -0.2926722 -0.5521980     -1.0987573  0.1822986      0.24375683
## [3,]  0.9169342  1.1758566 -0.9609612      0.2177225  1.1833015      0.55139033
##         urbanPop unemploymentRate  imputeGDP        year      month
## [1,] -1.26318322       -0.3992312 -1.1613184 -1.29795241 -1.0117751
## [2,]  0.03581483        0.7314800 -0.1162979  0.08653016 -0.1445393
## [3,]  1.13489414       -1.1366516  1.1573754  1.12489209  0.4336179
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [186] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## Objective function:
##    build     swap
## 9.597077 8.635189
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```