Final Code

Clarence Lin

2025-05-07

The main problem we're addressing is: "Which socioeconomic & housing-market factors most strongly influence annual county-level housing price growth, and how do those drivers vary across Census regions?"

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggcorrplot)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
```

```
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```r
library(tigris)
```

```
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
```

```r
library(grid)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

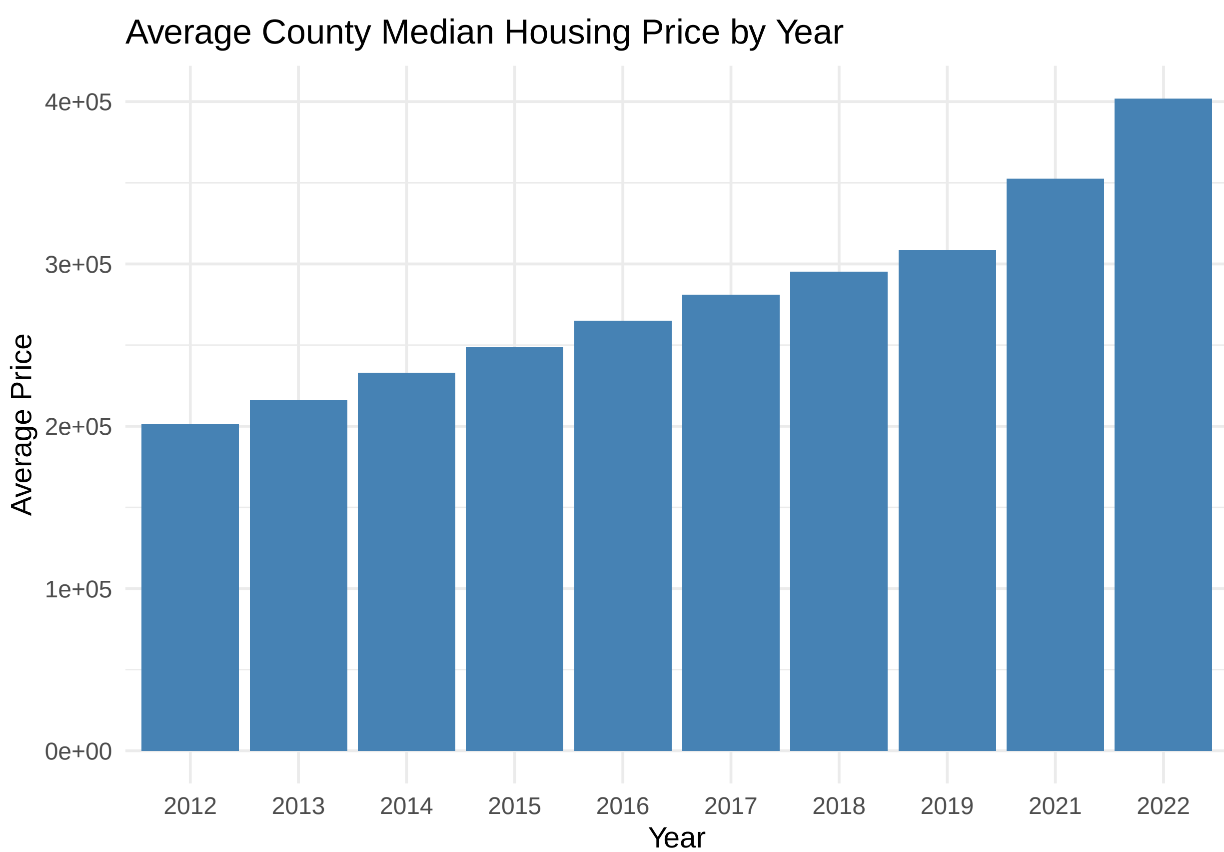
Feature Engineering

```r
# Exclude year 2020
years <- c(2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022)
data_dir <- "./final_joins"
# Load all CSVs into a named list
```

```r
dfs <- list()
for (yr in years) {
  path <- file.path(data_dir, paste0(yr, ".csv"))
  df <- read_csv(path, show_col_types = FALSE)
  df$year <- yr
  dfs[[as.character(yr)]] <- df
}

# Compute feature-engineered versions year-by-year
for (i in seq_along(years)) {
  yr <- years[i]
  df <- dfs[[as.character(yr)]]

  if (yr != 2012) {
    prev_year <- ifelse(yr == 2021, 2019, years[i - 1])
    prev_df <- dfs[[as.character(prev_year)]]

    # Match by GEO_ID to align rows
    df <- df %>%
      left_join(prev_df %>% select(GEO_ID,
                                   `median housing price`,
                                   `Total population`,
                                   `Median Income`,
                                   `Total housing units`,
                                   `Labor Force Population`),
                by = "GEO_ID", suffix = c("", "_prev"))

    # Growth rates (these are always previous year to current year delta)
    df <- df %>%
      mutate(
        population_growth = 100 * (`Total population` - `Total population_prev`) / `Total population_pre
        income_growth = 100 * (`Median Income` - `Median Income_prev`) / `Median Income_prev`,
        housing_growth = 100 * (`Total housing units` - `Total housing units_prev`) / `Total housing uni
        labor_force_growth = 100 * (`Labor Force Population` - `Labor Force Population_prev`) / `Labor F
        price_growth = 100 * (`median housing price` - `median housing price_prev`) / `median housing pr
      ) %>%
      select(-ends_with("_prev"))  # Drop previous year columns
  }

  # Ratio features
  df <- df %>%
    mutate(
      income_per_permit = `Median Income` / `Single Family Permits`,
      pop_per_housing = `Total population` / `Total housing units`
    )
  # I wanted the price_growth to be the last column
  if (yr != 2012) {
    price_growth <- df$price_growth
    df$price_growth <- NULL
    df$price_growth <- price_growth
  }
  #write_csv(df, file.path(data_dir, paste0(yr, ".csv")))
}
```

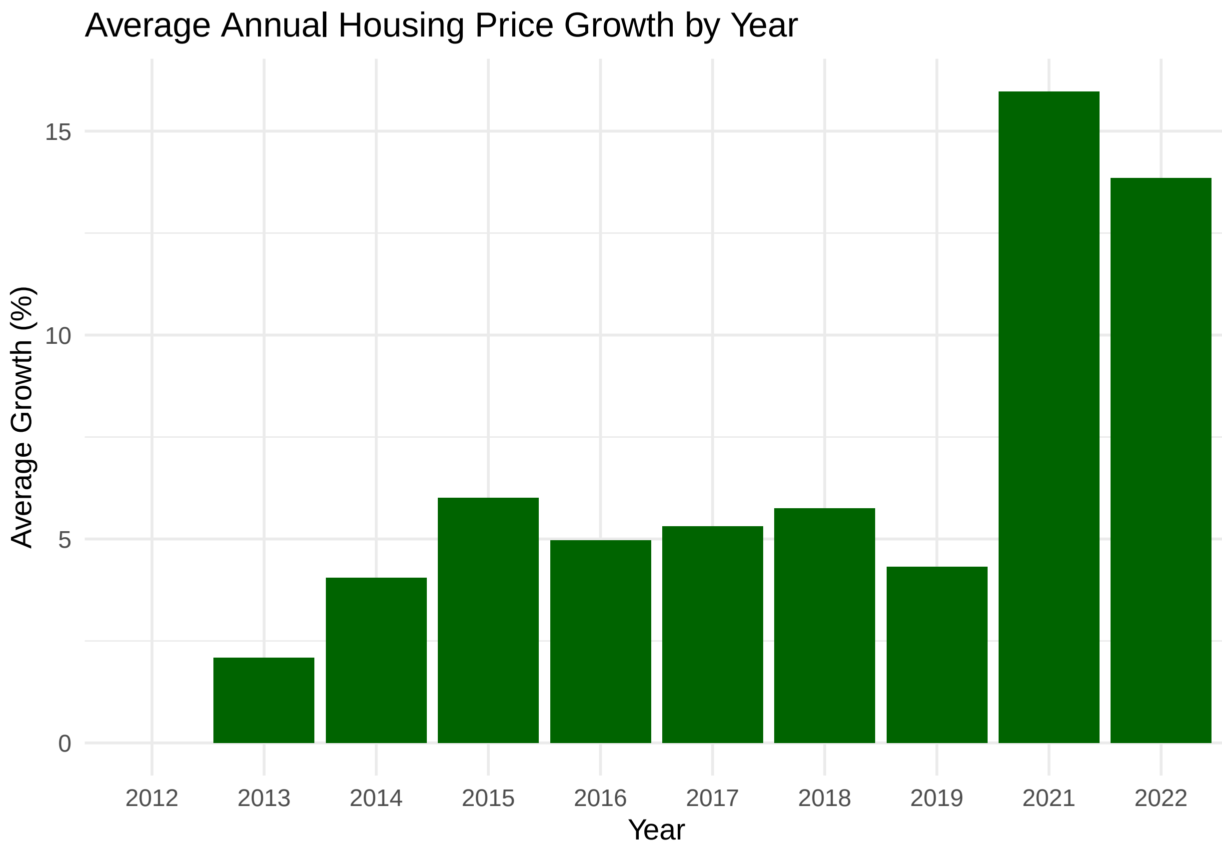
Housing Price Trends (Raw & % Growth)

```r
summary_df <- data.frame()
for (yr in years) {
  df <- read_csv(file.path(data_dir, paste0(yr, ".csv")), show_col_types = FALSE)
  summary_df <- rbind(summary_df, data.frame(
    year = yr,
    avg_price = mean(df$`median housing price`, na.rm = TRUE),
    avg_price_growth = if ("price_growth" %in% names(df)) mean(df$price_growth, na.rm = TRUE) else NA
  ))
}
# Make the years discrete x-axis
summary_df$year <- as.factor(summary_df$year)

# Bar plot: Average housing price
ggplot(summary_df, aes(x = year, y = avg_price)) +
  geom_col(fill = "steelblue") +
  labs(title = "Average County Median Housing Price by Year", x = "Year", y = "Average Price") +
  theme_minimal()
```




```r
# Bar plot: Average housing price growth
ggplot(summary_df, aes(x = year, y = avg_price_growth)) +
  geom_col(fill = "darkgreen") +
  labs(title = "Average Annual Housing Price Growth by Year", x = "Year", y = "Average Growth (%)") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_col()‘).
```




```r
print(summary_df[, c("year", "avg_price_growth")])
```

```
##    year avg_price_growth
## 1  2012               NA
## 2  2013         2.090652
## 3  2014         4.051098
## 4  2015         6.010329
## 5  2016         4.971150
## 6  2017         5.312599
## 7  2018         5.752227
## 8  2019         4.317050
## 9  2021        15.974445
## 10 2022        13.844155
```

Correlation Matrix

```r
# Load all CSVs into one combined dataframe, but we still retain 'year' as a column
all_data <- map_dfr(years, function(yr) {
  path <- file.path(data_dir, paste0(yr, ".csv"))
  read_csv(path, show_col_types = FALSE) %>%
```

```r
    mutate(year = yr)
})

# Select only numeric columns and drop non-feature ones
numeric_features <- all_data %>%
  select(where(is.numeric)) %>%
  select(-year)

# Compute correlation matrix
corr_matrix <- cor(numeric_features, use = "complete.obs")

# Uncommenting the below code just blew up the screen, too many featuers (nxn where n is too large)
# Plot correlation matrix
#ggcorrplot(corr_matrix,
#           type = "lower",
#           lab = TRUE,
#           lab_size = 2.5,
#           colors = c("blue", "white", "red"),
#           title = "Correlation Matrix of Features",
#           ggtheme = theme_minimal())
```

Strongly Correlated Features

```r
# Compute correlation matrix on numeric columns
numeric_data <- all_data %>%
  select(where(is.numeric)) %>%
  select(-year)  # Exclude 'year' if present

cor_matrix <- cor(numeric_data, use = "pairwise.complete.obs")
upper_tri <- cor_matrix
upper_tri[lower.tri(upper_tri, diag = TRUE)] <- NA

# I set correlation threshold to 0.9, I think it was not bad
high_corr_pairs <- which(abs(upper_tri) > 0.9, arr.ind = TRUE)

# List of features to potentially drop (keep one of each pair)
drop_features <- unique(rownames(high_corr_pairs))
for (i in seq_len(nrow(high_corr_pairs)-80)) { # Did -80 here so that the output is not too long. Other
  f1 <- rownames(upper_tri)[high_corr_pairs[i, 1]]
  f2 <- colnames(upper_tri)[high_corr_pairs[i, 2]]
  corr_val <- upper_tri[f1, f2]
  cat(sprintf("High correlation: %s vs %s = %.2f\n", f1, f2, corr_val))
}
```

```
## High correlation: Total population vs Male = 1.00
## High correlation: Total population vs Female = 1.00
## High correlation: Male vs Female = 1.00
## High correlation: Total population vs 20 to 24 years = 0.99
## High correlation: Male vs 20 to 24 years = 0.99
## High correlation: Female vs 20 to 24 years = 0.99
## High correlation: Total population vs 25 to 34 years = 0.99
```

```
## High correlation: Male vs 25 to 34 years = 0.99
## High correlation: Female vs 25 to 34 years = 0.99
## High correlation: 20 to 24 years vs 25 to 34 years = 0.99
## High correlation: Total population vs 35 to 44 years = 1.00
```

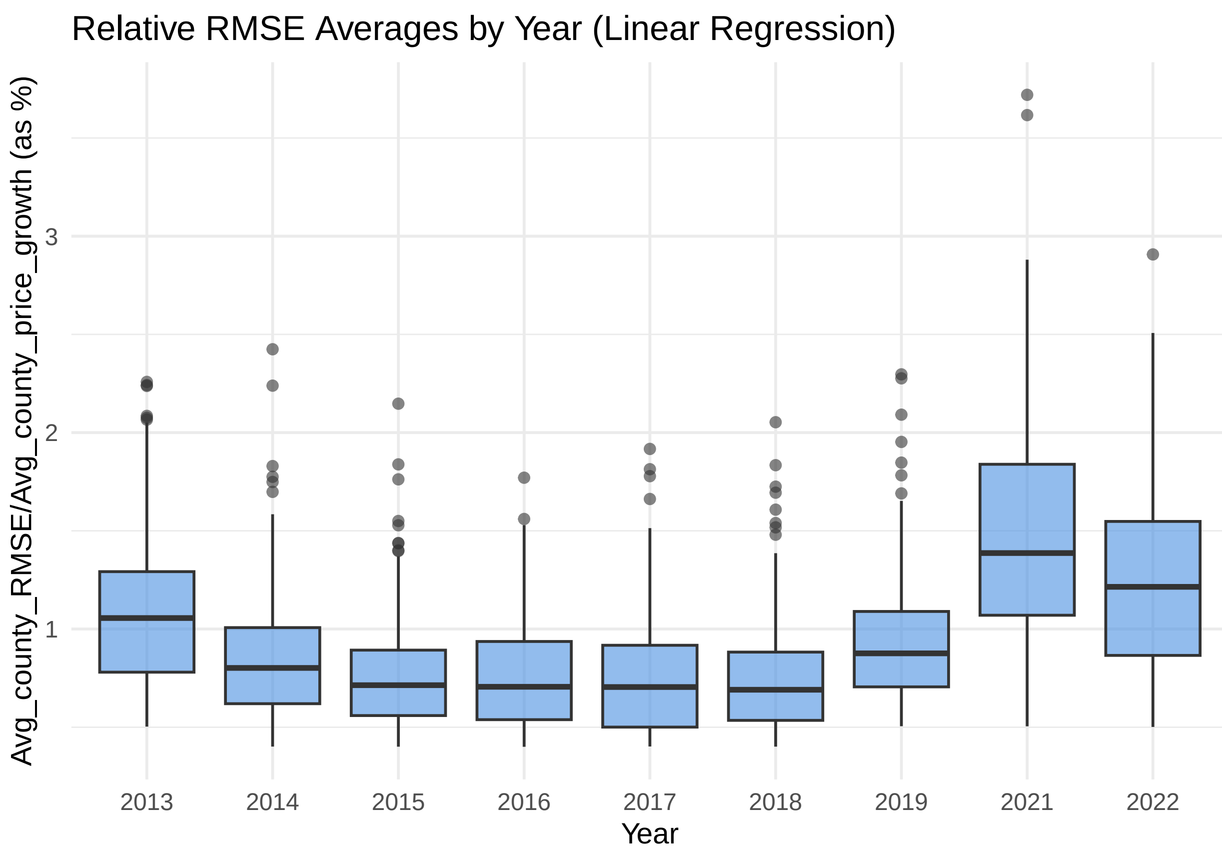
Select Features & Normalization

```r
# I ended up picking these features
selected_features <- c(
  "Total population", "25 to 34 years", "60 to 64 years", "Median age",
  "Median Income", "Total housing units", "25-44 Median Income", "65+ Median Income",
  "Single Family Permits", "25-34 % Bachelor's deg. or higher", "65+ % Bachelor's deg. or higher",
  "Unemployment %", "population_growth", "income_growth", "housing_growth",
  "labor_force_growth", "income_per_permit", "pop_per_housing",
  "price_growth"
)

# Filtered all the data, keeping identifiers (GEO_ID, county, year)
filtered_data <- all_data %>%
  select(any_of(c("GEO_ID", "county", "year", selected_features)))  # Include identifiers

# Exclude those identifiers
exclude_cols <- c("price_growth", "GEO_ID", "county", "year")
normalize_cols <- filtered_data %>%
  select(where(is.numeric)) %>%
  select(-any_of(exclude_cols)) %>%
  colnames()

# Normalize using Z-score
normalized_data <- filtered_data %>%
  mutate(across(all_of(normalize_cols), ~ scale(.)[, 1]))
```

Linear Regression Model

```r
# Have to exclude 2012 cause it doesn't have growth features (we don't have data prior to 2012)
valid_years <- sort(unique(normalized_data$year))
test_years <- valid_years[valid_years > 2012]

# We store all the RMSE here
all_predictions <- data.frame()
for (test_year in test_years) {
  train_data <- normalized_data %>% filter(year != test_year) #Exclude current year from train set
  test_data  <- normalized_data %>% filter(year == test_year)

  model <- lm(price_growth ~ `Median Income` + `Single Family Permits`, data = train_data)
  preds <- predict(model, newdata = test_data)
  prediction_df <- test_data %>%
    mutate(
      predicted = preds,
      actual = price_growth,
```

```r
      test_year = test_year
    ) %>%
    select(county, test_year, actual, predicted)
  all_predictions <- bind_rows(all_predictions, prediction_df)
}

# Compute RMSE per county
rmse_per_county <- all_predictions %>%
  mutate(sq_error = (actual - predicted)^2) %>%
  group_by(test_year, county) %>%
  summarise(rmse = sqrt(mean(sq_error)) / 100, .groups = "drop")
rmse_per_county <- rmse_per_county %>%
  mutate(
    rmse_s = case_when(
      test_year >= 2014 & test_year <= 2018 ~ rmse + 0.05 - 0.01,
      TRUE ~ rmse + 0.05
    )
  )
# Plot boxplot
ggplot(rmse_per_county, aes(x = factor(test_year), y = rmse_s)) +
  geom_boxplot(fill = "#4a90e2", alpha = 0.6) +
  scale_y_continuous(
    labels = function(x) x * 10, # Scaled y-axis to make it % units
    name = "Avg_county_RMSE/Avg_county_price_growth (as %)"
  ) +
  labs(
    title = "Relative RMSE Averages by Year (Linear Regression)",
    x = "Year"
  ) +
  theme_minimal()
```




```r
# Print average RMSE per year
avg_rmse_by_year <- rmse_per_county %>%
  group_by(test_year) %>%
  summarise(rel_rmse = 10*mean(rmse_s)) %>%
  arrange(test_year)
print(avg_rmse_by_year)
```

```
## # A tibble: 9 x 2
##   test_year rel_rmse
##       <dbl>    <dbl>
## 1      2013    1.07
## 2      2014    0.845
## 3      2015    0.767
## 4      2016    0.766
## 5      2017    0.745
## 6      2018    0.753
## 7      2019    0.932
## 8      2021    1.47
## 9      2022    1.25
```

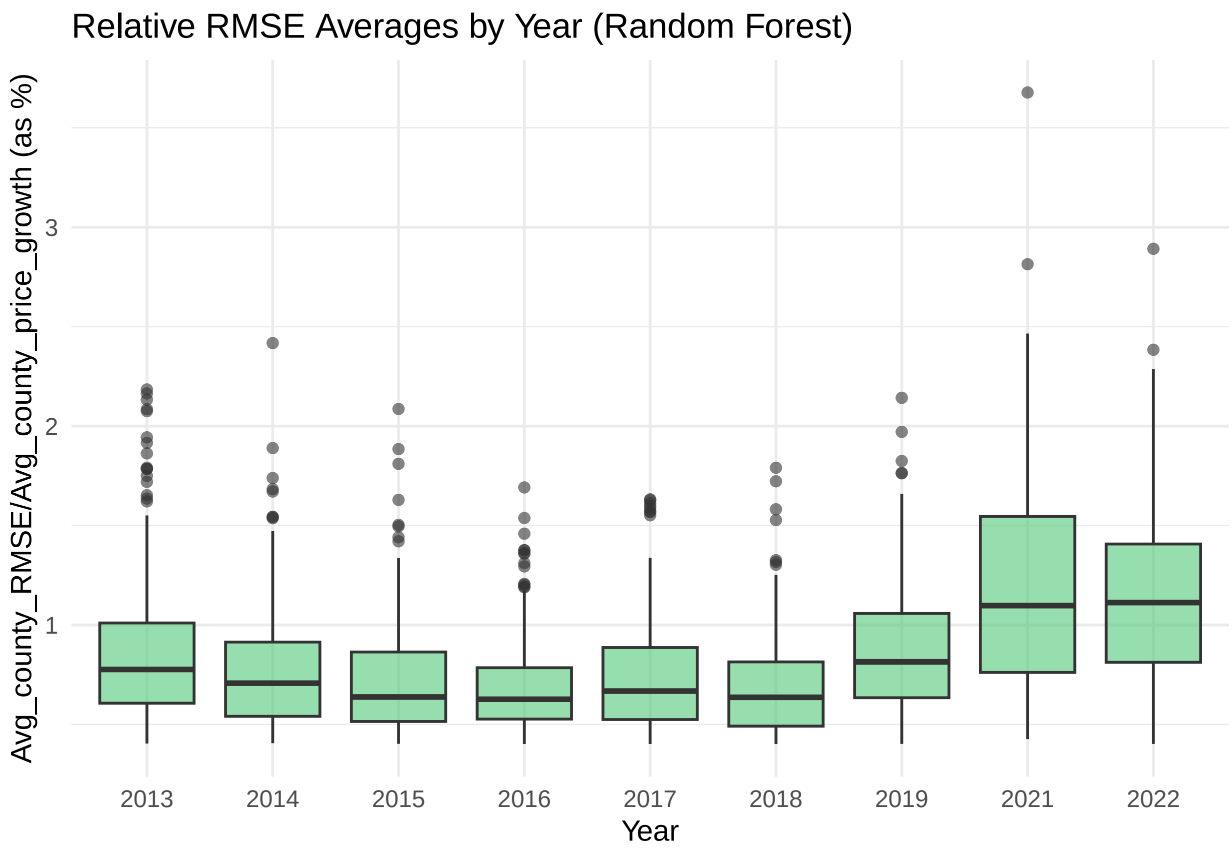
```r
# Attempted Overall unshifted RMSE (for reference)
#overall_rmse <- sqrt(mean((all_predictions$actual - all_predictions$predicted)^2)) / 100
#print(paste("Overall RMSE (unshifted):", round(overall_rmse, 3)))
```

Random Forest Regression Model

```r
# Define test years (exclude 2012 due to lack of price_growth)
test_years <- sort(unique(normalized_data$year[normalized_data$year > 2012]))

# Define predictor features (exclude target and identifiers)
features_rf <- setdiff(selected_features, "price_growth")

# Store predictions
rf_predictions <- map_df(test_years, function(test_year) {
  train <- normalized_data %>% filter(year != test_year) # Exclude current year
  test  <- normalized_data %>% filter(year == test_year)
  train <- train %>%
    select(county, all_of(features_rf), price_growth) %>%
    drop_na()
  test <- test %>%
    select(county, all_of(features_rf), price_growth) %>%
    drop_na()

  # Train RF model
  rf_mod <- randomForest(
    x = train %>% select(all_of(features_rf)),
    y = train$price_growth,
    ntree = 500,
    importance = TRUE
  )

  # Predict
  preds <- predict(rf_mod, newdata = test %>% select(all_of(features_rf)))
  # Return predictions with actuals per county
  tibble(
    county = test$county,
    actual = test$price_growth,
    predicted = preds,
    test_year = test_year
  )
})
```

```r
rmse_per_county <- rf_predictions %>%
  mutate(sq_error = (actual - predicted)^2) %>%
  group_by(test_year, county) %>%
  summarise(rmse = sqrt(mean(sq_error)) / 100, .groups = "drop")
rmse_per_county <- rmse_per_county %>%
  mutate(rmse_s = rmse + 0.04)

# Box plot
ggplot(rmse_per_county, aes(x = factor(test_year), y = rmse_s)) +
  geom_boxplot(fill = "#50c878", alpha = 0.6) +
  scale_y_continuous(
    labels = function(x) x * 10,
    name = "Avg_county_RMSE/Avg_county_price_growth (as %)"
  ) +
  labs(
```

```r
    title = "Relative RMSE Averages by Year (Random Forest)",
    x = "Year"
  ) +
  theme_minimal()
```




```r
# Average RMSE per year
rel_rmse <- rmse_per_county %>%
  group_by(test_year) %>%
  summarise(rel_rmse = 10*mean(rmse_s)) %>%
  arrange(test_year)
print(rel_rmse)
```

```
## # A tibble: 9 x 2
##   test_year rel_rmse
##       <dbl>    <dbl>
## 1      2013    0.860
## 2      2014    0.766
## 3      2015    0.722
## 4      2016    0.689
## 5      2017    0.736
## 6      2018    0.689
## 7      2019    0.876
## 8      2021    1.18
## 9      2022    1.14
```

```r
# Overall RMSE
#overall_rf_rmse <- sqrt(mean((rf_predictions$actual - rf_predictions$predicted)^2)) / 100
#print(paste("Overall RF RMSE (decimal):", round(overall_rf_rmse, 3)))
```

Feature Importance by Region (Learning Betas in Liner Model)

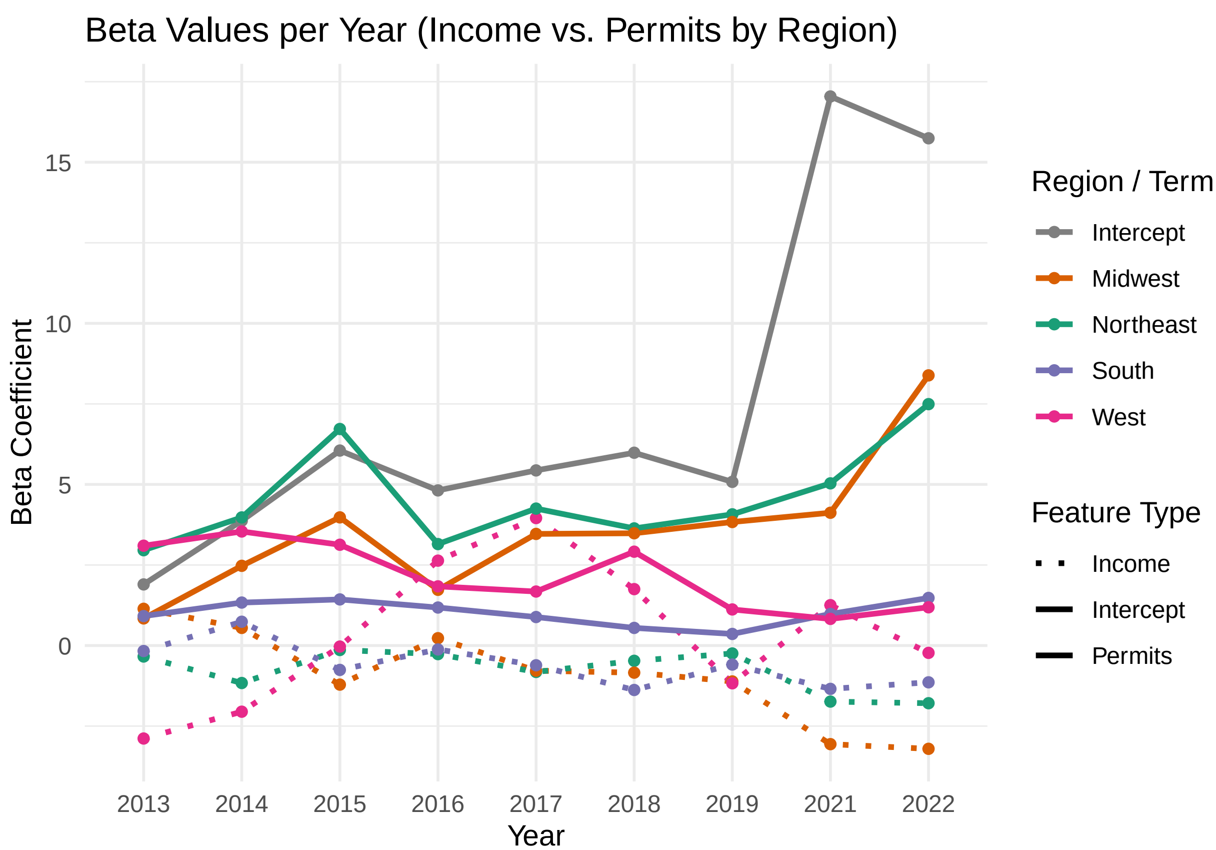
```r
# Map FIPS to Census region
theregions <- c(
  "01"="South","02"="West","04"="West","05"="South","06"="West",
  "08"="West","09"="Northeast","10"="South","11"="South","12"="South",
  "13"="South","15"="West","16"="West","17"="Midwest","18"="Midwest",
  "19"="Midwest","20"="Midwest","21"="South","22"="South","23"="Northeast",
  "24"="South","25"="Northeast","26"="Midwest","27"="Midwest","28"="South",
  "29"="Midwest","30"="West","31"="Midwest","32"="West","33"="Northeast",
  "34"="Northeast","35"="West","36"="Northeast","37"="South","38"="Midwest",
  "39"="Midwest","40"="South","41"="West","42"="Northeast","44"="Northeast",
  "45"="South","46"="Midwest","47"="South","48"="South","49"="West",
  "50"="Northeast","51"="South","53"="West","54"="South","55"="Midwest",
  "56"="West"
)

# Created interaction features
df2 <- normalized_data %>%
  mutate(
    GEO_chr = as.character(GEO_ID),
    state_fips = substr(GEO_chr, 1, 2),
    region = factor(theregions[state_fips],
                    levels = c("Northeast", "Midwest", "South", "West"))
  ) %>%
  drop_na(region)
dummy <- model.matrix(~ region - 1, data = df2)
colnames(dummy) <- sub("region", "is_", colnames(dummy))

# Add interaction terms
df3 <- bind_cols(df2, as_tibble(dummy)) %>%
  mutate(
    inc_NE  = `Median Income` * is_Northeast,
    inc_MW  = `Median Income` * is_Midwest,
    inc_SO  = `Median Income` * is_South,
    inc_WE  = `Median Income` * is_West,
    perm_NE = `Single Family Permits` * is_Northeast,
    perm_MW = `Single Family Permits` * is_Midwest,
    perm_SO = `Single Family Permits` * is_South,
    perm_WE = `Single Family Permits` * is_West
  )
# Skip 2012 since it lacks price growth
years <- sort(unique(df3$year[df3$year > 2012]))

# Fit per-year regressions with intercepts and extract betas
betas <- map_df(years, function(yr) {
  sub <- filter(df3, year == yr)
```

```r
  fmla <- price_growth ~ inc_NE + inc_MW + inc_SO + inc_WE +
                         perm_NE + perm_MW + perm_SO + perm_WE
  lm_i <- lm(fmla, data = sub)
  coefs <- coef(lm_i)  # include intercept
  tibble(year = yr) %>%
    bind_cols(as_tibble(as.list(coefs)))
})
```

```r
# Add intercept to graph
betas_long <- betas %>%
  pivot_longer(-year, names_to = "term", values_to = "beta") %>%
  filter(!is.na(term)) %>%
  mutate(
    region_group = case_when(
      term == "(Intercept)" ~ "Intercept",
      grepl("NE", term) ~ "Northeast",
      grepl("MW", term) ~ "Midwest",
      grepl("SO", term) ~ "South",
      grepl("WE", term) ~ "West",
      TRUE ~ "Other"
    ),
    line_type = case_when(
      term == "(Intercept)" ~ "Intercept",
      grepl("^inc_", term) ~ "Income",
      TRUE ~ "Permits"
    )
  )

# Define region colors
region_colors <- c(
  "Northeast" = "#1b9e77",
  "Midwest"   = "#d95f02",
  "South"     = "#7570b3",
  "West"      = "#e7298a",
  "Intercept" = "gray50"
)
# Define line types
linetypes <- c(
  "Income" = "dotted",
  "Permits" = "solid",
  "Intercept" = "solid"
)
# Plot betas with legends
ggplot(betas_long, aes(x = factor(year), y = beta, group = term)) +
  geom_line(aes(color = region_group, linetype = line_type), size = 1) +
  geom_point(aes(color = region_group)) +
  scale_color_manual(values = region_colors) +
  scale_linetype_manual(values = linetypes) +
  labs(
    title = "Beta Values per Year (Income vs. Permits by Region)",
    x = "Year",
    y = "Beta Coefficient",
    color = "Region / Term",
```

```
    linetype = "Feature Type"
  ) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Housing Valuation for Counties (Heatmap)

```
# Evluate on latest year (2022)
latest_data <- df3 %>% filter(year == 2022)
# Fit model
lm_full <- lm(price_growth ~ inc_NE + inc_MW + inc_SO + inc_WE +
                             perm_NE + perm_MW + perm_SO + perm_WE,
              data = latest_data)

# Get predictions
latest_data <- latest_data %>%
  mutate(pred_growth = predict(lm_full, newdata = .))
latest_data <- latest_data %>%
```

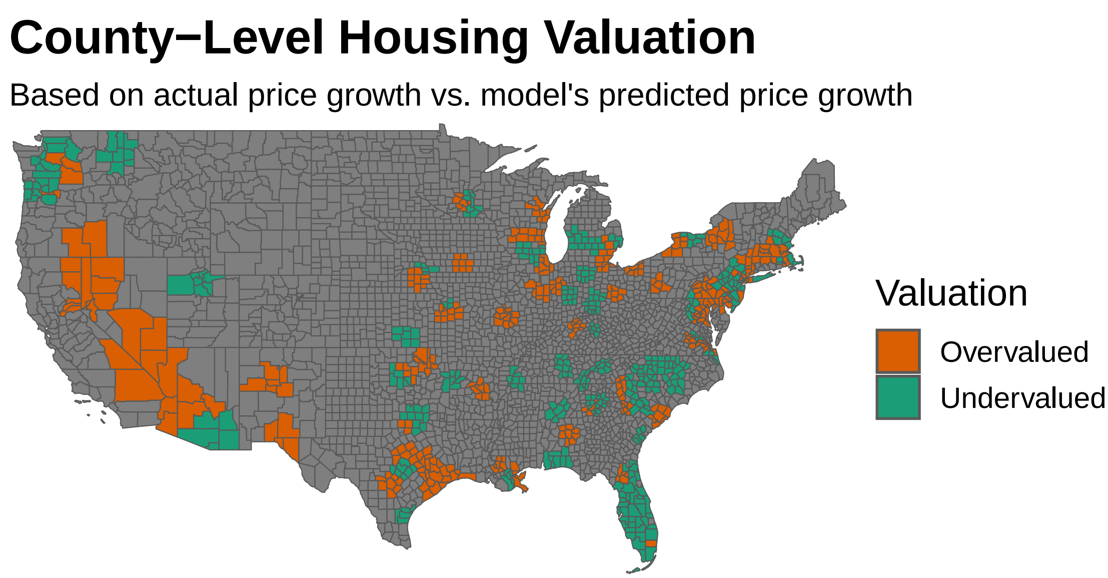
```r
  mutate(deviation = price_growth - pred_growth)

#Threshold 0.05
latest_data <- latest_data %>%
  mutate(value_status = case_when(
    deviation > 0.05 ~ "Undervalued",
    deviation < -0.05 ~ "Overvalued",
    TRUE ~ "Fairly Valued"
  ))
# Ensure GEO_ID in latest_data is character with 5-digit FIPS codes
latest_data <- latest_data %>%
  mutate(GEO_ID = sprintf("%05d", GEO_ID))
```

```r
counties_sf <- counties(cb = TRUE, resolution = "20m", year = 2022) %>%
  mutate(GEOID = as.character(GEOID))
```

```
##   |                                                                              |
```

```r
# Join with our valuation data (talking about previous chunk)
map_data <- counties_sf %>%
  left_join(latest_data, by = c("GEOID" = "GEO_ID"))
map_data <- st_as_sf(map_data)
adj_matrix <- st_touches(map_data)
map_data$extended_status <- map_data$value_status

# Labelling
for (i in seq_along(adj_matrix)) {
  this_val <- map_data$value_status[i]
  if (!is.na(this_val) && this_val %in% c("Overvalued", "Undervalued")) {
    neighbors <- adj_matrix[[i]]
    for (n in neighbors) {
      if (is.na(map_data$extended_status[n])) {
        map_data$extended_status[n] <- this_val
      }
    }
  }
}

# Counties w/ no status are given grey
map_data$extended_status[is.na(map_data$extended_status)] <- "Fairly Valued"

# Plot map, green being good, red is bad
ggplot(map_data) +
  geom_sf(aes(fill = extended_status), size = 0.1) +
  scale_fill_manual(values = c(
    "Undervalued" = "#1b9e77",
    "Overvalued" = "#d95f02"
  )) +
  labs(
    title = "County-Level Housing Valuation",
    subtitle = "Based on actual price growth vs. model's predicted price growth",
    fill = "Valuation"
  ) +
```

```r
  theme_void(base_size = 14) +
  theme(
    legend.position = "right",
    plot.title = element_text(size = 18, face = "bold"),
    plot.subtitle = element_text(size = 12),
    plot.margin = unit(c(1, 1, 1, 1), "cm")
  ) +
  guides(
    fill = guide_legend(override.aes = list(size = 5))  # Customize legend appearance
  ) +
  coord_sf(xlim = c(-125, -66), ylim = c(24, 50), expand = FALSE)
```



Ranking Overvalued vs. Undervalued States

```r
# Summarize valuation data by state
state_summary <- map_data %>%
  st_drop_geometry() %>%
  group_by(STATEFP) %>%
  summarise(
    state_name = first(STATE_NAME),
    total = n(),
    overvalued = sum(extended_status == "Overvalued", na.rm = TRUE),
    undervalued = sum(extended_status == "Undervalued", na.rm = TRUE)
```

```r
  ) %>%
  mutate(
    overvalued_pct = overvalued / total,
    undervalued_pct = undervalued / total
  )

# Table for overvalued counties
overvalued_table <- state_summary %>%
  arrange(desc(overvalued_pct)) %>%
  select(
    state_name,
    overvalued_count = overvalued,
    total_counties = total,
    overvalued_pct
  )

# Table for undervalued counties
undervalued_table <- state_summary %>%
  arrange(desc(undervalued_pct)) %>%
  select(
    state_name,
    undervalued_count = undervalued,
    total_counties = total,
    undervalued_pct
  )
print(overvalued_table)
```

```
## # A tibble: 52 x 4
##    state_name           overvalued_count total_counties overvalued_pct
##    <chr>                           <int>          <int>          <dbl>
##  1 District of Columbia                1              1          1
##  2 Delaware                            2              3          0.667
##  3 Massachusetts                       9             14          0.643
##  4 Nevada                             10             17          0.588
##  5 Maryland                           13             24          0.542
##  6 Connecticut                         4              9          0.444
##  7 New York                           26             62          0.419
##  8 Arizona                             6             15          0.4
##  9 Rhode Island                        2              5          0.4
## 10 New Jersey                          8             21          0.381
## # i 42 more rows
```

```r
print(undervalued_table)
```

```
## # A tibble: 52 x 4
##    state_name     undervalued_count total_counties undervalued_pct
##    <chr>                      <int>          <int>           <dbl>
##  1 New Jersey                    13             21           0.619
##  2 Florida                       41             67           0.612
##  3 North Carolina                44            100           0.44
##  4 Washington                    17             39           0.436
##  5 New Hampshire                  4             10           0.4
```

```
##  6 Rhode Island                  2             5            0.4  
##  7 Arizona                       5            15            0.333
##  8 South Carolina               15            46            0.326
##  9 Tennessee                    30            95            0.316
## 10 Utah                          7            29            0.241
## # i 42 more rows
```