MC-D607 Final Project HUD-API

Marco Castro

2024-11-26

Fair Market Rents (FMR) vs Market Rate Rents

This project aims to ascertain whether the market rate rental costs for 1 and 2 bedroom apartments are consistent with the federal government's Fair Market Rents (FMR) — the 40th percentile of gross rents for typical rental units paid by tenants that moved within the last 20 months. FMR is used to calculate benefits such as the Housing Choice Vouchers used to help unhoused individuals find permanent housing.

As an annually-released estimate, FMR data may not capture the real-time economic realities that drive local housing markets. In other words, market rates can sometimes outpace FMR estimates when market conditions quickly drive rental prices up in certain areas. This means that some individuals approved for housing assistance benefits may be unable to find an apartment for the approved voucher amount in certain areas.

This project uses FMR data pulled from HUD using their publicly available API. Market rate data was scraped from the rental listing site Trulia in three distinct metropolitan areas: Atlanta, GA, Buffalo, NY and San Diego, CA to obtain an up-to-date snapshot of real rental costs that can be compared against data from HUD's FMR Dataset. The comparison will help to determine if the FMR data released earlier this year for fiscal year Oct 2024-Sept 2025 is still relevant to current housing costs in these areas. For the purposes of this project, I will work with data only for apartments with less than 3 bedrooms.

```
# A1: declare the cities we will analyze
target_cities <- c('Atlanta-Sandy Springs-Roswell','Buffalo-Cheektowaga-Niagara Falls','San Diego-Carlsb
max_bedrooms <- 3
```

A: Getting FMR data using the HUD API

The section below declares the global variables and functions for to call the HUD FMR & IL API using the `httr2` package using a HUD API key (A2).

```
# A2: Custom function to call HUD FMR & IL API
call_hud <- function(endpoint) {
  # declare constants
  domain <- "https://www.huduser.gov"
  method   <- "fmr" # alt method is il or mtspil
  path <-paste("hudapi/public/", method, "/", endpoint,sep="")

  # init request
  req <- request(domain) |>
    req_headers("Accept" = "application/json") |>
    req_auth_bearer_token(token) |>
    req_url_path(path) |>
    req_user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr

  # call api
  resp <- req_perform(req)
```

```r
  # parse response
  json_resp <- resp |>
    resp_body_string() |>
    fromJSON()

  # return parsed response
  return(json_resp)
}
```

Retrieving the Metro Ids

The custom function `call_hud` is used in the section below (A3) to request the HUD metropolitan area id (`entity_id`) for each of our target cities. This id will then be used to retrieve the FMR for each of the zipcodes that make up the area as well as the estimate for the area overall. The request returns a JSON object for all metropolitan areas. After converting the response to a dataframe (A4), I separated the **area_name** column to cities, states, and area type. Finally, I filtered for using the target cities array declared earlier.

```r
  # A3: Fetch Metro Area Ids from API
  metros <- call_hud("listMetroAreas")

  # A4: cleanup and filter list of metro areas
  #     for target areas
  metro_areas <- as.data.frame(metros) |>
    # split area_name col into city and type
    separate_wider_regex(
      area_name, c(area_name= ".*", ",\\s", type=".*")
    ) |>
    # split type col into state and type
    mutate(
      state = substring(type,1,2),
    ) |>
    # filter by our target areas
    filter(area_name %in% target_cities) |>
    select(c('cbsa_code',  'area_name', 'state'))

glimpse(metro_areas)
```

```
## Rows: 3
## Columns: 3
## $ cbsa_code <chr> "METRO12060M12060", "METRO15380M15380", "METRO41740M41740"
## $ area_name <chr> "Atlanta-Sandy Springs-Roswell", "Buffalo-Cheektowaga-Niagar~
## $ state     <chr> "GA", "NY", "CA"
```

Retrieving FMRs for Target Cities

With the ids for the metropolitan areas, I was able to use the `data` endpoint from HUD's API to retrieve the FMR values for each metropolitan area and its corresponding zip codes (A5). I created a custom function `get_metro_data` that requests the FMR estimates and cleans up the response. This function is called from a loop that iterates through each of our target areas. Finally, the loop uses `rbind` to merge the dataframes for each target city into a single dataframe.

```r
# A5: Call, clean-up and merge metro area data
# Custom function to call and clean up metro area data
get_metro_data <- function(row) {
  # use "cbsa_code" col as entity_id for API call
```

```r
  entity_resp <- call_hud(paste("data", row$cbsa_code, sep="/"))

  # clean up data
  df <- as_tibble(entity_resp$data$basicdata) |>
    rename_all(~tolower(str_trim(str_replace_all(., "-", "_")))) |>
    rename(studio = efficiency) |>
    mutate(
      area_name = row$area_name
    ) |>
    relocate(area_name, .before=zip_code)
}

# init df with first row
metro_fmrs <- tibble()

# loop through all other metro areas
for (i in 1:nrow(metro_areas)) {
  metro_fmrs <- rbind(metro_fmrs, get_metro_data(metro_areas[i,]))
}

head(metro_fmrs)
```

```
## # A tibble: 6 x 7
##   area_name   zip_code studio one_bedroom two_bedroom three_bedroom four_bedroom
##   <chr>       <chr>     <int>       <int>       <int>         <int>        <int>
## 1 Atlanta-Sa~ MSA lev~   1591        1653        1830          2205         2653
## 2 Atlanta-Sa~ 30002      1100        1150        1270          1530         1840
## 3 Atlanta-Sa~ 30003      1730        1800        1990          2400         2890
## 4 Atlanta-Sa~ 30004      1830        1910        2110          2540         3060
## 5 Atlanta-Sa~ 30005      2030        2110        2340          2820         3390
## 6 Atlanta-Sa~ 30006      1660        1730        1910          2300         2770
```

```r
zip_codes <- metro_fmrs |>
  filter(zip_code != 'MSA level') |>
  select(c("area_name", "zip_code"))
```

Tidying the FMR data

I used `pivot_longer` to break up each FMR cost estimate into its own row broken up by apartment type in each apartment (A6). I then used `mutate` to convert the bedroom classifications to an integer representation based on the number of bedrooms. For example, a studio was given a value of 0, `one_bedroom` was given a value of 1, etc. This gave me a dataframe where every fmr for each area and apartment size (in # of bedrooms) is an observation.

```r
# A6:  tidy data
metro_fmr_by_zip_and_num_bds <- metro_fmrs |>
  pivot_longer(
    cols = studio:four_bedroom,
    names_to = c("bedrooms"),
    values_to = "fmr"
  ) |>
  mutate(
    bedrooms = case_when(
      bedrooms == 'studio' ~ as.integer(0),
      bedrooms == 'one_bedroom' ~ as.integer(1),
```

```
      bedrooms == 'two_bedroom' ~ as.integer(2),
      bedrooms == 'three_bedroom' ~ as.integer(3),
      bedrooms == 'four_bedroom' ~ as.integer(4)
    )
  ) |>
  arrange(zip_code) |>
  filter(bedrooms <= max_bedrooms)

metro_fmr_msa_by_num_bds <- metro_fmr_by_zip_and_num_bds |>
  filter(zip_code == 'MSA level')

head(metro_fmr_by_zip_and_num_bds)
```

```
## # A tibble: 6 x 4
##   area_name                          zip_code bedrooms   fmr
##   <chr>                              <chr>       <int> <int>
## 1 Buffalo-Cheektowaga-Niagara Falls 14001           0   860
## 2 Buffalo-Cheektowaga-Niagara Falls 14001           1   900
## 3 Buffalo-Cheektowaga-Niagara Falls 14001           2  1050
## 4 Buffalo-Cheektowaga-Niagara Falls 14001           3  1290
## 5 Buffalo-Cheektowaga-Niagara Falls 14004           0   860
## 6 Buffalo-Cheektowaga-Niagara Falls 14004           1   900
```

B: Scraping Market Rate Data

In this section, I use the `httr` package to scrape Trulia, a real estate listing portal, for rental listings in the zipcodes we retrieved for each of our target cities. In the block below (B1), I initialize an empty tibble and declare the variable types that I will be using to populate with the scraped rental listing data. I also set our user agent headers to mimic a web browser. Finally, I declare two custom functions:

1. **throttled_GET**: throttle the HTML requests to Trulia. This is used to send a delayedGET request using a timer using the `slowly` function in order to minimize possibly being blocked
2. **cleanup_price**: converts the string value from Trulia to an integer by removing any non-numerical characters. For example, the string "$1,000/mo" would return 1000.

```
# B1: Declare parser constants

# init market rate tibble
real_estate_df <- tibble(
  zip_code = character(),
  bedrooms = integer(),
  market_rate = integer()
)

# update user agents
user_agent <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/(
set_config(add_headers(`User-Agent` = user_agent))

delay <- 10       # number of seconds to delay each GET request

# custom function for delayed scraping
throttled_GET <- slowly(
  ~read_html(.),
  rate = rate_delay(delay)
)
```

```r
# custom function to remove misc chars
cleanup_price <- function(s) {
  t <- str_replace_all(s, "([$,])", "") #remove dollar signs and commas
  x <- gsub("^([0-9]+).*", "\\1", t) #only return first group of integers
  return(as.integer(x))
}
```

Looping through our Zip Codes

In the following section, I use a *for loop* to iterate through each of the zip codes retrieved from HUD for our target cities to retrieve current (B2). I set each zip code (`zip`), the number of bedrooms (`beds`), and a page counter (`current_page`) as parameters for our GET request, which is called via the custom `throttled_GET` function (B3). Upon retrieving the HTML from our response, I use `html_element` to find all listings on the price, then save the monthly rental cost in a list `listings` (B4). I use another *for loop* through this list and append the values as new rows in the `real_estate_df` tibble (B5). Finally, I save the results to a CSV called "trulia_data.csv" for future use (B6).

```r
# B2: Scrape Real Estate Data

# can be altered for multipage scraping
current_page <- 1 # page counter
start <- 1         # start index
end <- 1           # end index. Use #nrow(zip_codes) to use zip_code length

# loop through zip codes list
for (beds in 0:max_bedrooms) {

  for (i in start:end) {
    # get zip code at current index
    zip <- zip_codes[i,]$zip_code

    # B3: build path based on the current zip, beds and page vars
    # and send to our throttled GET request function
    page_reponse <- throttled_GET(paste("https://www.trulia.com/for_rent/", zip, "_zip/", beds, "_beds/

    # find price from current page html
    result_container <- page_reponse %>%
      html_element(xpath='//ul[@data-testid="search-result-list-container"]' )

    # B4: save our rental prices to Trulia
    listings <- result_container %>%
      html_elements(xpath = '//li//div[@data-testid="property-price"]' ) %>%
      html_text()

    # B5: loop through all listing prices
    # and append to market rate tibble
    for (s in listings)
      real_estate_df <- real_estate_df |>
        add_row(zip_code = zip, bedrooms = beds, market_rate = cleanup_price(s))
  }
}

head(real_estate_df, n=10)
```

```
## # A tibble: 10 x 3
##    zip_code bedrooms market_rate
##    <chr>       <int>       <int>
##  1 30002           0        1400
##  2 30002           0        1397
##  3 30002           0        1575
##  4 30002           0        1303
##  5 30002           1        1075
##  6 30002           1        1100
##  7 30002           1        1198
##  8 30002           1        1525
##  9 30002           1        1548
## 10 30002           1        1175
```

```r
# B6: write rental listings to csv
write_csv(real_estate_df, "trulia_data_temp.csv")
```

C: Working with Combined Data

In this section, I combine the FMR data I retrieved using HUD's API and the market rate data scraped from Trulia. First, I read in the data scraped from Trulia from section B from the CSV file "trulia_data.csv" and filtered out any major outliers (C1). Next I `group_by` using the zip codes and number of bedrooms and `summarise` to calculate the 40th percentile of rental costs for every type of apartment size within each zip code and assign the grouped results to the dataframe `market_rate_by_zip_and_num_bds`.

```r
# C1: Read previously parsed file from Trulia
# add mean for market rate data based on zipcode and # of bedrooms
# then add metro ids
market_rates <- read_csv(
    file = "trulia_data.csv",
    col_names = TRUE,
    col_types = cols(.default = col_character(), market_rate = "i", bedrooms = "i"),
    show_col_types = FALSE
  ) |>
  filter(
    bedrooms < 3 &
    market_rate < 10000 # remove outliers
  )

# C2: group by zip codes and # bedrooms
market_rate_by_zip_and_num_bds <- market_rates |>
  group_by(across(all_of(c("zip_code", "bedrooms")))) |>
  summarise(
    market_rate = round(quantile(market_rate,probs=0.4))
  )

head(market_rate_by_zip_and_num_bds, n=3)
```

```
## # A tibble: 3 x 3
## # Groups:   zip_code [2]
##   zip_code bedrooms market_rate
##   <chr>       <int>       <dbl>
## 1 14001           1        1040
## 2 14001           2        1145
## 3 14004           1        1748
```

Combining FMR and Market Rate Dataframes By Zip Code

Next, I used the `right_join` function to join the `metro_fmr_by_zip_and_num_bds` dataframe to the `market_rate_by_zip_and_num_bds` dataframe by zip codes (C3). Because some of the zip codes did not have any active listings on the date we scraped Trulia, using `right_join` will ensure that only the zip codes that had listings will be preserved in our `combined_df_by_zip` dataframe. Next, I used `pivot_longer` to break up each estimated rental cost as a unique observation that can be grouped or filtered by area_name, zip code, apartment size (# of bedrooms), and estimate type (fmr vs market rate).

```
### C3: Combine FMR and Market Rate Dataframes By Zip Code
combined_df_by_zip <- metro_fmr_by_zip_and_num_bds |>
  right_join(market_rate_by_zip_and_num_bds, by= c('zip_code'='zip_code', 'bedrooms'='bedrooms')) |>
  mutate(bedrooms = as.factor(bedrooms))


# C4: pivot longer by estimate type
combined_df_by_zip_and_type <- combined_df_by_zip |>
  pivot_longer(
    cols = c(fmr, market_rate),
    names_to = c("type"),
    values_to = "rent"
  ) |>
  mutate(
    type = str_replace_all(type, "_", " ")
  )

head(combined_df_by_zip_and_type, n=8)
```

```
## # A tibble: 8 x 5
##   area_name                         zip_code bedrooms type         rent
##   <chr>                             <chr>    <fct>    <chr>       <dbl>
## 1 Buffalo-Cheektowaga-Niagara Falls 14001    1        fmr           900
## 2 Buffalo-Cheektowaga-Niagara Falls 14001    1        market rate  1040
## 3 Buffalo-Cheektowaga-Niagara Falls 14001    2        fmr          1050
## 4 Buffalo-Cheektowaga-Niagara Falls 14001    2        market rate  1145
## 5 Buffalo-Cheektowaga-Niagara Falls 14004    1        fmr           900
## 6 Buffalo-Cheektowaga-Niagara Falls 14004    1        market rate  1748
## 7 Buffalo-Cheektowaga-Niagara Falls 14004    2        fmr          1050
## 8 Buffalo-Cheektowaga-Niagara Falls 14004    2        market rate  1750
```

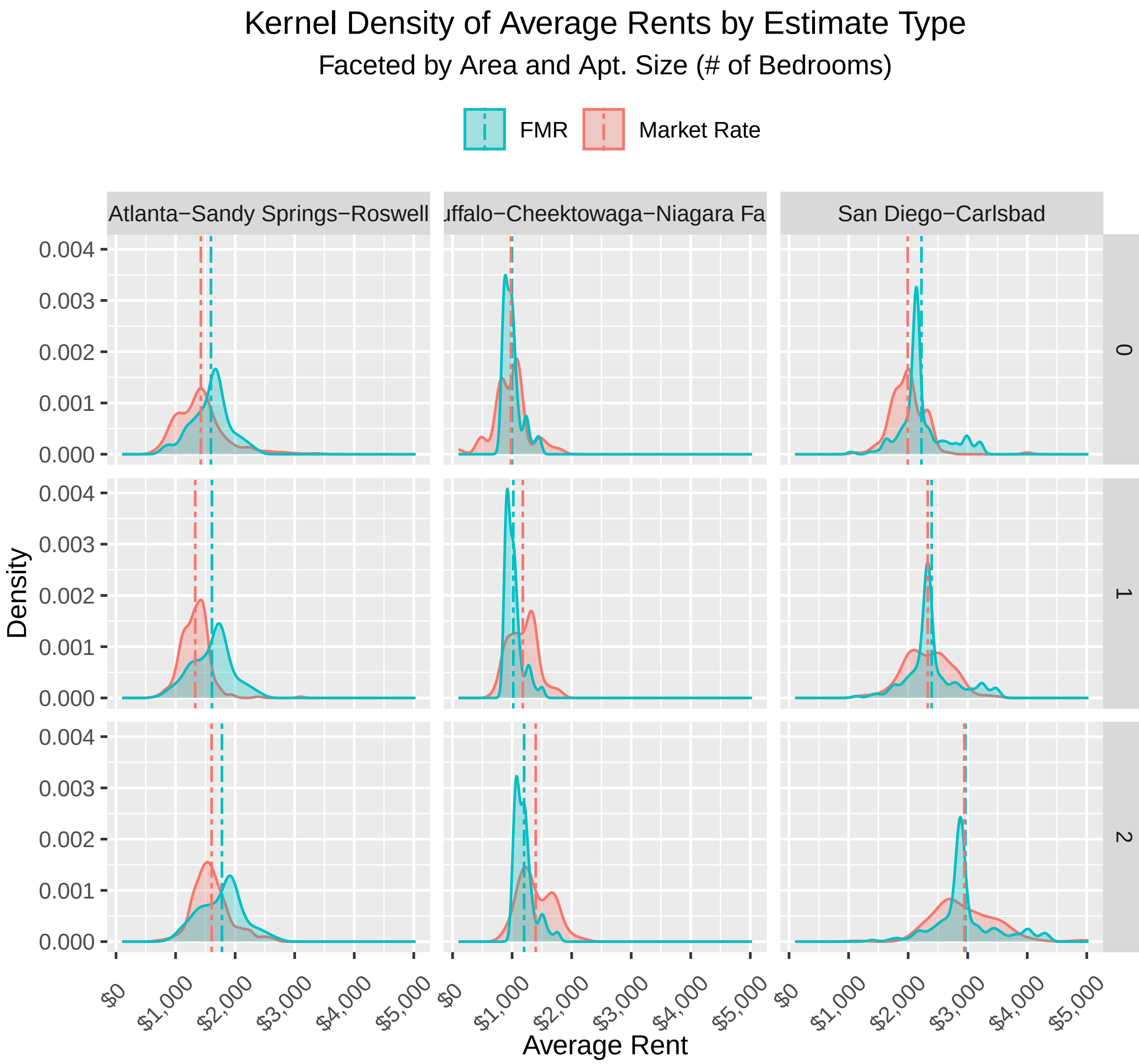
Kernel Density of Estimate Types

With the dataframe tidied, we can start analysing our data. The chart below (C5) shows a kernel density of the distribution of average rents by estimate types (FMR vs Market Rent) using facets to group the data by city and apartment size (number of bedrooms).

```
# C5: Calculate Distribution of raw estimates
mean_rents <- combined_df_by_zip_and_type |>
  group_by(type, area_name, bedrooms) |>
  summarise(mean_rent= mean(rent)
)
```

```
## `summarise()` has grouped output by 'type', 'area_name'. You can override using
## the `.groups` argument.
```

```
# distribution of fmr vs market reate
ggplot(combined_df_by_zip_and_type, aes(rent, fill=fct_rev(type), color=fct_rev(type))) +
  geom_density(alpha = .3) +
  geom_vline(data=mean_rents, aes(xintercept=mean_rent, col=type), linetype = "twodash") +
  theme(
    legend.position = "top",
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    axis.text.x = element_text(angle = 45, vjust = .6, hjust=.75)
  ) +
  labs(
    title = "Kernel Density of Average Rents by Estimate Type",
    subtitle = "Faceted by Area and Apt. Size (# of Bedrooms)",
    x = "Average Rent",
    y = "Density"
  ) +
  scale_x_continuous(labels = scales::dollar_format()) +
  scale_fill_discrete(
    name = element_blank(),
    labels=c('Market Rate', 'FMR'),
    guide = guide_legend(reverse = TRUE)
  ) +
  scale_color_discrete(
    name = element_blank(),
    labels=c('Market Rate', 'FMR'),
    guide = guide_legend(reverse = TRUE)
  ) +
  facet_grid(vars(bedrooms), vars(area_name))
```



The chart suggests that the market rates estimates generally trail FMR costs for all apartment sizes in the Atlanta Metropolitan area. In the Buffalo metro, market rates appear to be somewhat similar for studios, but have a wider spread for 1 and 2 bedrooms and rental costs are higher than FMR. In San Diego, the median FMR and market rate rents are somewhat similar, but there is a wider spread of rental costs across zip codes for market rate estimates, while FMR rents are more concentrated represented through a higher peak.

Count of Zipcodes by Estimate Type

Next, I estimated the difference between the market rate and FMR estimated rents to calculate the margin of error and to classify which observations (zip codes) have market rate estimates at or below FMR and which are above (C6). This classification can be used to calculate and visualize the counts of each estimate type grouped by area, apartment size (number of bedrooms), and classification (at-or-below vs above FMR).

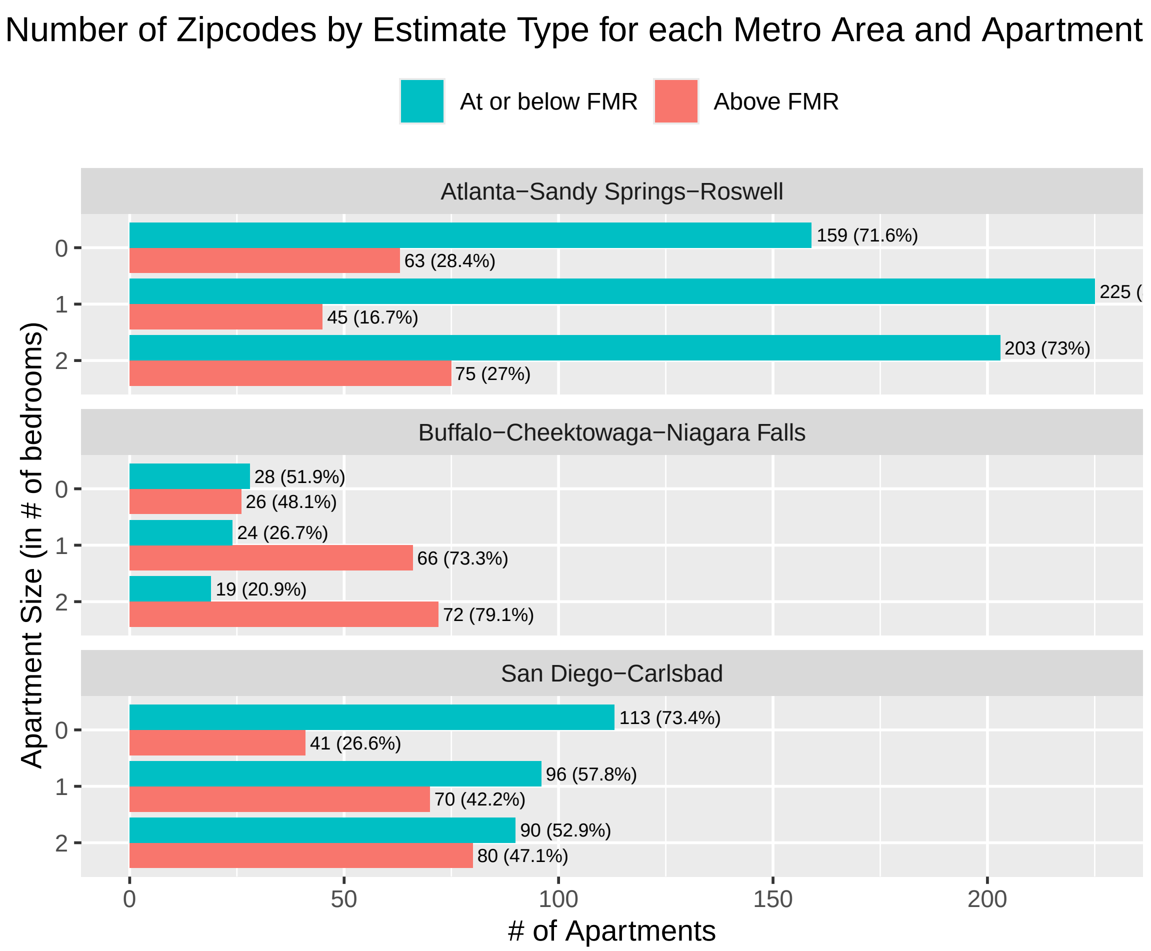
```r
# C6: Calculate Rent Differences by Zip
# create df with the margin of error
combined_df_by_zip_diffs <- combined_df_by_zip |>
  mutate(
    margin_of_error = market_rate - fmr,
    at_or_below_fmr = as.factor(fmr >= market_rate)
  ) |>
  select(-c(market_rate))

# C7: Counts by Estimate Type
counts_by_estimate_type <- combined_df_by_zip_diffs |>
  group_by(area_name, bedrooms, at_or_below_fmr) |>
  summarise(
    count_type =  n(),
  ) |>
  ungroup() |>
  arrange(area_name, bedrooms, desc(at_or_below_fmr))

# calc total counts
combined_df_by_zip_diffs_total_counts <- combined_df_by_zip_diffs |>
  group_by(area_name, bedrooms) |>
  summarise(
    count_total = n(),
  ) |>
  ungroup()

# calculate percent of type per group
counts_by_estimate_type <- counts_by_estimate_type |>
  left_join(combined_df_by_zip_diffs_total_counts, by=c("area_name", "bedrooms")) |>
  mutate(
    type_ratio = count_type / count_total
  )

# plot
ggplot(combined_df_by_zip_diffs, aes(y=fct_rev(bedrooms), fill=at_or_below_fmr)) +
  geom_bar(position = position_dodge()) +
  #geom_text(stat='count', aes(label = after_stat(count),), hjust=-.3, position = position_dodge(width=
  geom_text(data=counts_by_estimate_type, mapping = aes(label = paste(count_type, " (", round(type_rati
  theme(
    legend.position = "top",
    plot.title = element_text(hjust = 0.5)
  ) +
  labs(
    title = "Number of Zipcodes by Estimate Type for each Metro Area and Apartment Size",
    x = "# of Apartments",
    y = "Apartment Size (in # of bedrooms)"
  ) +
  scale_fill_discrete(
    name = element_blank(),
    labels=c('Above FMR', 'At or below FMR'),
    guide = guide_legend(reverse = TRUE)
  ) +
  facet_wrap(~area_name, ncol=1)
```



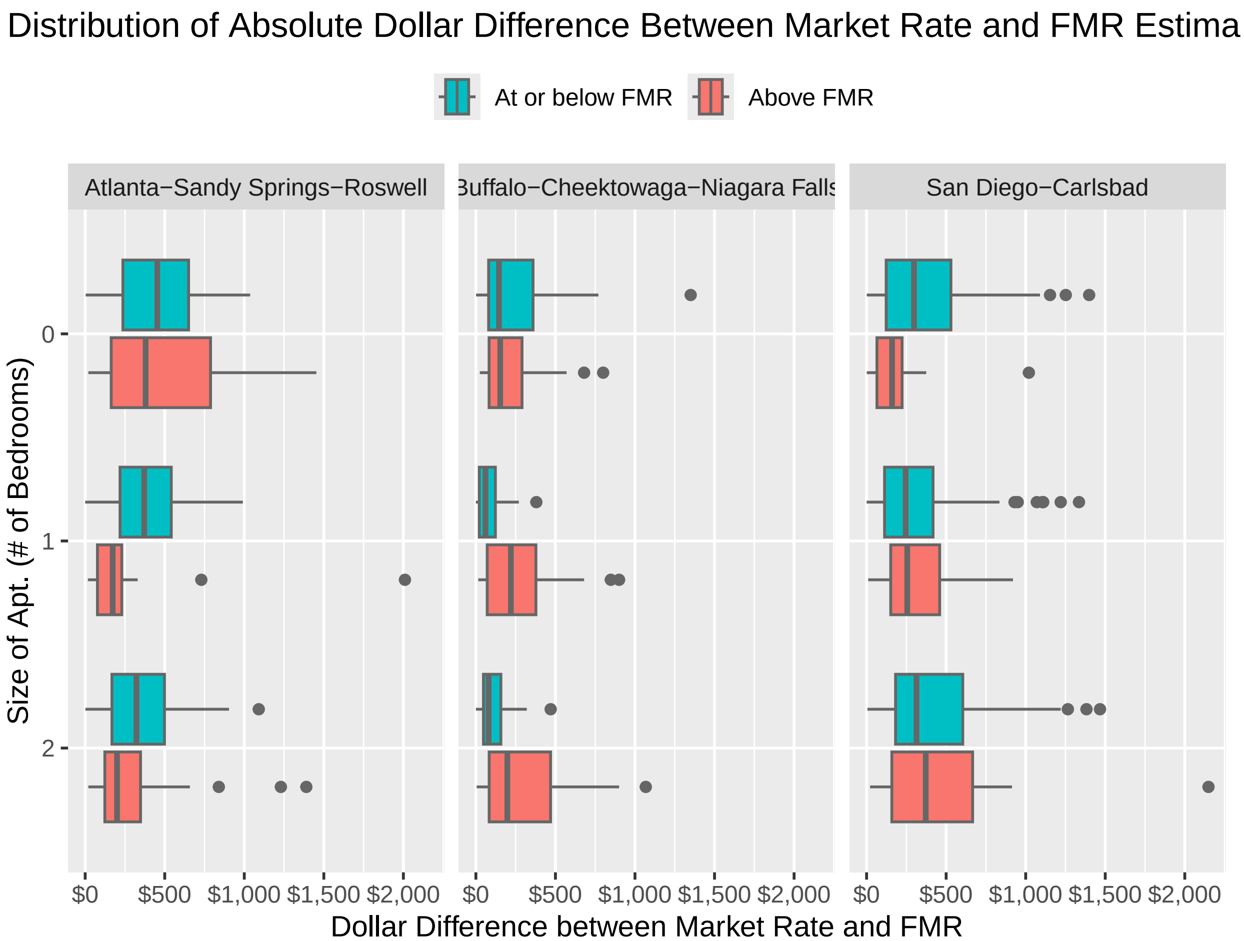
This visualization shows of our 9 groups, only 1- and 2-bedroom apartments in Buffalo had a higher number of zip codes of apartments where market rate rents exceeded FMR.

Dollar Differences

This next visualization shows a boxplot of the absolute dollar difference between market rate and FMR. This chart supports the rental mismatch we observed earlier for 1- and 2-bedroom apartments in Buffalo.

```
# C8: Dollar differences
ggplot(combined_df_by_zip_diffs, aes(abs(margin_of_error), fct_rev(bedrooms), fill=at_or_below_fmr)) +
  geom_boxplot(color="#666666") +
  theme(
    legend.position="top",
    plot.title = element_text(hjust = 0.5)
  ) +
  labs(
    title = "Distribution of Absolute Dollar Difference Between Market Rate and FMR Estimates",
    x = "Dollar Difference between Market Rate and FMR",
    y = "Size of Apt. (# of Bedrooms)"
  ) +
  scale_x_continuous(labels = scales::dollar_format()) +
  scale_fill_discrete(
    name = element_blank(),
    labels=c('Above FMR', 'At or below FMR'),
    guide = guide_legend(reverse = TRUE)
```

```r
) +
facet_wrap(~area_name, ncol=3, shrink=FALSE)
```




```r
# facet_grid(vars(bedrooms), vars(area_name))
```

Here we see that apartments exceeding FMR have a higher dollar difference than those at or below FMR. The range of rents for these two groups is also much wider for apartments which may speak to type of aparments available. Interestingly we see wider ranges in market rents for studios in Atlanta and two-bedrooms in San Diego.

Median Rent Difference Above FMR

The next question to answer is what is median rent difference above FMR for the identified the groups of interest. First, I will use `combined_df_by_zip_diffs` created in step C6. This dataframe takes our original joined dataframe of rent estimates (FMR and Market Rate) by zip code and added a `margin_of_error` column to store the difference between the Market Rate estimate and FMR. I can group my dataframe by area, apartment size (number of bedrooms), and a boolean categorizing whether the estimate is at-or-below or above FMR for the given zipcode (C8). I then used **pivot_wider** to split the values for the median at-or-below FMR and for the median at FMR. In this example, the metropolitan areas by apartment size become my observations. These values are stored in a dataframe called `median_diffs_df`.

```r
# C8: Calculate median margin of error for each area and bedroom size
median_diffs_df <- combined_df_by_zip_diffs |>
  group_by(area_name, bedrooms, at_or_below_fmr) |>
  summarise(
```

```
    median_difference = median(margin_of_error)
  ) |>
  ungroup() |>
  mutate(at_or_below_fmr = ifelse(at_or_below_fmr == TRUE, "Median $Diff Below FMR", "Median $Diff Above
  pivot_wider(
    names_from = at_or_below_fmr,
    values_from = median_difference
  )

head(median_diffs_df, n=3)
```

```
## # A tibble: 3 x 4
##   area_name                bedrooms Median $Diff Above F~1 Median $Diff Below F~2
##   <chr>                    <fct>                     <dbl>                  <dbl>
## 1 Atlanta-Sandy Springs--~ 0                           380                   -453
## 2 Atlanta-Sandy Springs--~ 1                           173                   -371
## 3 Atlanta-Sandy Springs--~ 2                           200                   -322
## # i abbreviated names: 1: `Median $Diff Above FMR`, 2: `Median $Diff Below FMR`
```

Next I calculated the median FMR for each of metrolpolitan area by apartment size (C9). I will then join this dataframe to the `median_diffs_df` created in the step above and do some clean up to prepare it for output (10). Finally, I will print out the table for my areas of interest. This table lets us see the FMR for the area of interest and the median rental cost we should expect to see for apartments listed in Dec 2024 in those areas.

```
# C9: Calculate median FMR for each area and bedroom size
median_fmrs <- combined_df_by_zip_diffs |>
  group_by(area_name, bedrooms) |>
  summarise(
    median_fmr = median(fmr)
  ) |>
  ungroup()

# C10: Join dfs and filter by groups of interest
median_diffs_df_clean <- median_diffs_df |>
  left_join(median_fmrs, by=c("area_name", "bedrooms")) |>
  rename(FMR = median_fmr, Area = area_name, Bedrooms = bedrooms) |>
  relocate(FMR, .before=`Median $Diff Above FMR`) |>
  mutate(
    `FMR` = paste("$", abs(round(`FMR`,0)), sep=""),
    `Median $Diff Below FMR` = paste("$", abs(round(`Median $Diff Below FMR`,0)), sep=""),
    `Median $Diff Above FMR` = paste("$", abs(round(`Median $Diff Above FMR`,0)), sep="")
  )

head(median_diffs_df, n=3)
```


```
## # A tibble: 3 x 4
##   area_name                bedrooms Median $Diff Above F~1 Median $Diff Below F~2
##   <chr>                    <fct>                     <dbl>                  <dbl>
## 1 Atlanta-Sandy Springs--~ 0                           380                   -453
## 2 Atlanta-Sandy Springs--~ 1                           173                   -371
## 3 Atlanta-Sandy Springs--~ 2                           200                   -322
## # i abbreviated names: 1: `Median $Diff Above FMR`, 2: `Median $Diff Below FMR`
```


Groups of Interest where Market Rates Exceeded FMR One and Two bedroom apartments in the Buffalo metropolitan area stood out as groups of interest because over 70% of zip codes for each group had an estimated rental cost at the 40th percentile above the FMR. These groups also showed a higher median dollar ($) difference for the rental cost above FMR as shown below:

```
groups_of_interest_above_fmr <- median_diffs_df_clean |>
  filter((Area == 'Buffalo-Cheektowaga-Niagara Falls' & Bedrooms == 1) | (Area == 'Buffalo-Cheektowaga-N
  select(-c(`Median $Diff Below FMR`))


formattable(groups_of_interest_above_fmr,
            align=c("l","c","c","c","c"),
            list(
              Area = formatter("span", style = x ~ style(color = "gray")),
              `Median $Diff Above FMR`= formatter("span", style = ~ style(color = "red"))
            )
          )
```

Area

Bedrooms

FMR

Median $Diff Above FMR

Buffalo-Cheektowaga-Niagara Falls

1

$990

$220

Buffalo-Cheektowaga-Niagara Falls

2

$1180

$198

Groups of Interest where FMR Exceeded Market Rates Two bedroom apartments for San Diego were also identified as a group of interest because the percentage of apartments for this group at-or-below FMR and those above FMR is somewhat similar, with 52.9% and 47.1% respectively. Additionally, our boxplot showed that apartments above FRM had a wider distribution of the dollar difference from FRM for each zip code vs. those at-or-below FMR. The table below also shows that median dollar difference for apartments above FMR is higher than median dollar difference below FMR.

```
groups_of_interest_below_fmr <- median_diffs_df_clean |>
  filter((Area == 'San Diego-Carlsbad' & Bedrooms == 2))

formattable(groups_of_interest_below_fmr,
            align=c("l","c","c","c","c"),
            list(
              Area = formatter("span", style = x ~ style(color = "gray")),
              `Median $Diff Below FMR`= formatter("span", style = ~ style(color = "green")),
              `Median $Diff Above FMR`= formatter("span", style = ~ style(color = "red"))
            )
          )
```

Area

Bedrooms

FMR

Median $Diff Above FMR

Median $Diff Below FMR

San Diego-Carlsbad

2

$2880

$372

$314

Conclusion

- Analysis is based on a "snapshot" of parsed data on December 12th showing only a sample of current listings.
- Some zip codes had no rental listings available on that particular day.
- AJAX pagination issues
- Should collect real estate data over time to better represent real market conditions. Alt. use time-series data like the Zillow Observed Rent Index (ZORI).
- Future work: Expand to include 3/4 bedroom apartments and other metropolitan areas