EDA

Tong Su

2025-04-17

data structure and dictionary

```
arb_df <- read.csv("./data/Zip_Zhvi_2bedroom.csv")

str(arb_df)
```

```
## 'data.frame':    8946 obs. of  262 variables:
##  $ RegionID  : int  61627 61628 61625 97515 97519 72418 61617 61637 97130 97691 ...
##  $ RegionName: int  10013 10014 10011 94024 94028 33109 10003 10023 93108 94301 ...
##  $ City      : chr  "New York" "New York" "New York" "Los Altos" ...
##  $ State     : chr  "NY" "NY" "NY" "CA" ...
##  $ Metro     : chr  "New York" "New York" "New York" "San Jose" ...
##  $ CountyName: chr  "New York" "New York" "New York" "Santa Clara" ...
##  $ SizeRank  : int  1744 379 15 4335 7158 8800 21 3 5941 4583 ...
##  $ X1996.04  : int  NA NA NA 419500 NA NA NA NA NA 376300 ...
##  $ X1996.05  : int  NA NA NA 422300 NA NA NA NA NA 375900 ...
##  $ X1996.06  : int  NA NA NA 430400 NA NA NA NA NA 375000 ...
##  $ X1996.07  : int  NA NA NA 440400 NA NA NA NA NA 373000 ...
##  $ X1996.08  : int  NA NA NA 447100 NA NA NA NA NA 373300 ...
##  $ X1996.09  : int  NA NA NA 447900 NA NA NA NA NA 374500 ...
##  $ X1996.10  : int  NA NA NA 448000 NA NA NA NA NA 375600 ...
##  $ X1996.11  : int  NA NA NA 450100 NA NA NA NA NA 377900 ...
##  $ X1996.12  : int  NA NA NA 452200 NA NA NA NA NA 381000 ...
##  $ X1997.01  : int  NA NA NA 456100 NA NA NA NA NA 384600 ...
##  $ X1997.02  : int  NA NA NA 464600 NA NA NA NA NA 390200 ...
##  $ X1997.03  : int  NA NA NA 470400 NA NA NA NA NA 395800 ...
##  $ X1997.04  : int  NA NA NA 472300 NA NA NA NA NA 399600 ...
##  $ X1997.05  : int  NA NA NA 475400 NA NA NA NA NA 402600 ...
##  $ X1997.06  : int  NA NA NA 480000 NA NA NA NA NA 406600 ...
##  $ X1997.07  : int  NA NA NA 483200 NA NA NA NA NA 412900 ...
##  $ X1997.08  : int  NA NA NA 488900 NA NA NA NA NA 420900 ...
##  $ X1997.09  : int  NA NA NA 496900 NA NA NA NA NA 428600 ...
##  $ X1997.10  : int  NA NA NA 504700 NA NA NA NA NA 435800 ...
##  $ X1997.11  : int  NA NA NA 511300 NA NA NA NA NA 442200 ...
##  $ X1997.12  : int  NA NA NA 519500 NA NA NA NA NA 446900 ...
##  $ X1998.01  : int  NA NA NA 525900 NA NA NA NA NA 448900 ...
##  $ X1998.02  : int  NA NA NA 529600 656200 NA NA NA NA 450200 ...
##  $ X1998.03  : int  NA NA NA 535900 666600 NA NA NA NA 453300 ...
##  $ X1998.04  : int  NA NA NA 542300 684300 NA NA NA NA 455200 ...
##  $ X1998.05  : int  NA NA NA 543300 690200 NA NA NA NA 454800 ...
##  $ X1998.06  : int  NA NA NA 543300 690000 NA NA NA NA 457500 ...
##  $ X1998.07  : int  NA NA NA 550000 679000 NA NA NA NA 463000 ...
```

```
##  $ X1998.08  : int  NA NA NA 560000 677500 NA NA NA NA 468300 ...
##  $ X1998.09  : int  NA NA NA 571200 688800 NA NA NA NA 472600 ...
##  $ X1998.10  : int  NA NA NA 582300 695600 NA NA NA 702500 477200 ...
##  $ X1998.11  : int  NA NA NA 589800 685000 NA NA NA 709100 479100 ...
##  $ X1998.12  : int  NA NA NA 589300 682000 NA NA NA 716100 482400 ...
##  $ X1999.01  : int  NA NA NA 582800 689100 NA NA NA 723100 487500 ...
##  $ X1999.02  : int  NA NA NA 577600 696500 NA NA NA 729200 489300 ...
##  $ X1999.03  : int  NA NA NA 578800 701800 NA NA NA 741700 488700 ...
##  $ X1999.04  : int  NA NA NA 580000 712800 NA NA NA 757700 491100 ...
##  $ X1999.05  : int  NA NA NA 576600 715500 NA NA NA 770500 497200 ...
##  $ X1999.06  : int  NA NA NA 576800 717600 NA NA NA 778700 503300 ...
##  $ X1999.07  : int  NA NA NA 585700 737400 NA NA NA 789100 510100 ...
##  $ X1999.08  : int  NA NA NA 601700 774600 662700 NA NA 797200 519400 ...
##  $ X1999.09  : int  NA NA NA 624300 803200 620300 NA NA 799000 535200 ...
##  $ X1999.10  : int  NA NA NA 648100 818900 608600 NA NA 796000 550700 ...
##  $ X1999.11  : int  NA NA NA 665900 830100 600800 NA NA 797600 566800 ...
##  $ X1999.12  : int  NA NA NA 681700 845900 603300 NA NA 801000 583600 ...
##  $ X2000.01  : int  NA NA NA 699800 865000 619000 NA NA 807100 600600 ...
##  $ X2000.02  : int  NA NA NA 716400 879600 630700 NA NA 818700 614500 ...
##  $ X2000.03  : int  NA NA NA 734300 892900 626500 NA NA 839000 622700 ...
##  $ X2000.04  : int  NA NA NA 758500 907400 625200 NA NA 855500 631000 ...
##  $ X2000.05  : int  NA NA NA 784600 943900 620200 NA NA 868500 645300 ...
##  $ X2000.06  : int  NA NA NA 808700 984200 618700 NA NA 882200 662700 ...
##  $ X2000.07  : int  NA NA NA 831900 1006400 625600 NA NA 896100 680500 ...
##  $ X2000.08  : int  NA NA NA 851100 1020200 646700 NA NA 908400 695200 ...
##  $ X2000.09  : int  NA NA NA 865700 1069100 668400 NA NA 924400 703900 ...
##  $ X2000.10  : int  NA NA NA 876800 1124300 690500 NA NA 935900 712400 ...
##  $ X2000.11  : int  NA NA NA 886200 1175800 715100 NA NA 940500 723700 ...
##  $ X2000.12  : int  NA NA NA 900200 1213300 741300 NA NA 952600 734300 ...
##  $ X2001.01  : int  NA NA NA 920500 1239600 754500 NA NA 973700 744600 ...
##  $ X2001.02  : int  NA NA NA 931300 1259700 757800 NA NA 992400 750600 ...
##  $ X2001.03  : int  NA NA NA 929500 1283300 749400 NA NA 1007900 754200 ...
##  $ X2001.04  : int  NA NA NA 922500 1283900 748400 NA NA 1024700 751000 ...
##  $ X2001.05  : int  NA NA NA 916300 1276400 769100 NA NA 1038200 740400 ...
##  $ X2001.06  : int  NA NA NA 897300 1294200 798700 NA NA 1046700 725300 ...
##  $ X2001.07  : int  NA NA NA 865600 1301300 832100 NA NA 1057300 706000 ...
##  $ X2001.08  : int  NA NA NA 833500 1268400 839000 NA NA 1074500 683200 ...
##  $ X2001.09  : int  NA NA NA 811200 1230500 818200 NA NA 1097300 664900 ...
##  $ X2001.10  : int  NA NA NA 796100 1211900 813700 NA NA 1120100 652100 ...
##  $ X2001.11  : int  NA NA NA 794200 1205600 830400 NA NA 1134700 641600 ...
##  $ X2001.12  : int  NA NA NA 799500 1192000 832800 NA NA 1150600 632300 ...
##  $ X2002.01  : int  NA NA NA 798900 1167200 837600 NA NA 1174000 625600 ...
##  $ X2002.02  : int  NA NA NA 799900 1160500 848200 NA NA 1201400 625100 ...
##  $ X2002.03  : int  NA NA NA 813000 1179600 851600 NA NA 1217400 631100 ...
##  $ X2002.04  : int  NA NA NA 831300 1197500 866100 NA NA 1217600 641400 ...
##  $ X2002.05  : int  NA NA NA 850200 1200100 892300 NA NA 1216200 651500 ...
##  $ X2002.06  : int  NA NA NA 874000 1198800 901200 NA NA 1223700 661100 ...
##  $ X2002.07  : int  NA NA NA 891100 1212100 889500 NA NA 1234800 665300 ...
##  $ X2002.08  : int  NA NA NA 890900 1228200 890000 NA NA 1239500 661500 ...
##  $ X2002.09  : int  NA NA NA 883000 1227500 904600 NA NA 1243600 651900 ...
##  $ X2002.10  : int  NA NA NA 879700 1225700 927800 NA NA 1256200 644900 ...
##  $ X2002.11  : int  NA NA NA 880100 1225800 943200 NA NA 1284400 644200 ...
##  $ X2002.12  : int  NA NA NA 880100 1227200 947700 NA NA 1318500 649300 ...
##  $ X2003.01  : int  NA NA NA 877300 1244100 955300 NA NA 1340400 657800 ...
```

```
##  $ X2003.02  : int  NA NA NA 874100 1263600 954500 NA NA 1358500 659500 ...
##  $ X2003.03  : int  NA NA NA 873600 1256000 969000 NA NA 1384200 654200 ...
##  $ X2003.04  : int  NA NA NA 878500 1244500 1013800 NA NA 1424000 650900 ...
##  $ X2003.05  : int  NA NA NA 884500 1235400 1073700 NA NA 1448100 652900 ...
##  $ X2003.06  : int  NA NA NA 887100 1229000 1114600 NA NA 1460900 649900 ...
##  $ X2003.07  : int  NA NA NA 886000 1238400 1147300 NA NA 1483400 644900 ...
##  $ X2003.08  : int  NA NA NA 890700 1259900 1191200 NA NA 1515400 648400 ...
##  $ X2003.09  : int  NA NA NA 901000 1263400 1203800 NA NA 1532800 662500 ...
##  $ X2003.10  : int  NA NA NA 910100 1279800 1122500 NA NA 1541400 677400 ...
##  $ X2003.11  : int  NA NA NA 913600 1316500 1077900 NA NA 1557300 686200 ...
##   [list output truncated]
```

```
data_dict <- tibble(
  Variable = names(arb_df),
  DataType = sapply(arb_df, class),
  Category = case_when(
    sapply(arb_df, is.numeric) ~ "Numerical",
    sapply(arb_df, is.character) | sapply(arb_df, is.factor) ~ "Categorical",
    TRUE ~ "Other"
  )
)

print(data_dict)
```

```
## # A tibble: 262 x 3
##    Variable   DataType  Category
##    <chr>      <chr>     <chr>
##  1 RegionID   integer   Numerical
##  2 RegionName integer   Numerical
##  3 City       character Categorical
##  4 State      character Categorical
##  5 Metro      character Categorical
##  6 CountyName character Categorical
##  7 SizeRank   integer   Numerical
##  8 X1996.04   integer   Numerical
##  9 X1996.05   integer   Numerical
## 10 X1996.06   integer   Numerical
## # i 252 more rows
```

```
head(names(arb_df), 20)
```

```
##  [1] "RegionID"   "RegionName" "City"       "State"      "Metro"
##  [6] "CountyName" "SizeRank"   "X1996.04"   "X1996.05"   "X1996.06"
## [11] "X1996.07"   "X1996.08"   "X1996.09"   "X1996.10"   "X1996.11"
## [16] "X1996.12"   "X1997.01"   "X1997.02"   "X1997.03"   "X1997.04"
```

```
tail(names(arb_df), 20)
```

```
##  [1] "X2015.11" "X2015.12" "X2016.01" "X2016.02" "X2016.03" "X2016.04"
##  [7] "X2016.05" "X2016.06" "X2016.07" "X2016.08" "X2016.09" "X2016.10"
## [13] "X2016.11" "X2016.12" "X2017.01" "X2017.02" "X2017.03" "X2017.04"
## [19] "X2017.05" "X2017.06"
```

```
# Rename those columns to clean date format
names(arb_df) <- gsub("^X(\\d{4})\\.(\\d{2})$", "\\1-\\2", names(arb_df))
```

```r
ts_cols <- names(arb_df)[grepl("^\\d{4}-\\d{2}$", names(arb_df))]
length(ts_cols)
```

```
## [1] 255
```

```r
df_long <- arb_df %>%
  pivot_longer(
    cols = all_of(ts_cols),
    names_to = "Date",
    values_to = "Price"
  ) %>%
  mutate(Date = as.Date(paste0(Date, "-01"), format = "%Y-%m-%d"))
```

```r
glimpse(df_long)
```

```
## Rows: 2,281,230
## Columns: 9
## $ RegionID   <int> 61627, 61627, 61627, 61627, 61627, 61627, 61627, 61627, 616~
## $ RegionName <int> 10013, 10013, 10013, 10013, 10013, 10013, 10013, 10013, 100~
## $ City       <chr> "New York", "New York", "New York", "New York", "New York",~
## $ State      <chr> "NY", "NY", "NY", "NY", "NY", "NY", "NY", "NY", "NY", "NY",~
## $ Metro      <chr> "New York", "New York", "New York", "New York", "New York",~
## $ CountyName <chr> "New York", "New York", "New York", "New York", "New York",~
## $ SizeRank   <int> 1744, 1744, 1744, 1744, 1744, 1744, 1744, 1744, 1744, 1744,~
## $ Date       <date> 1996-04-01, 1996-05-01, 1996-06-01, 1996-07-01, 1996-08-01~
## $ Price      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

```r
summary(df_long$Date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "1996-04-01" "2001-07-01" "2006-11-01" "2006-10-31" "2012-03-01" "2017-06-01"
```

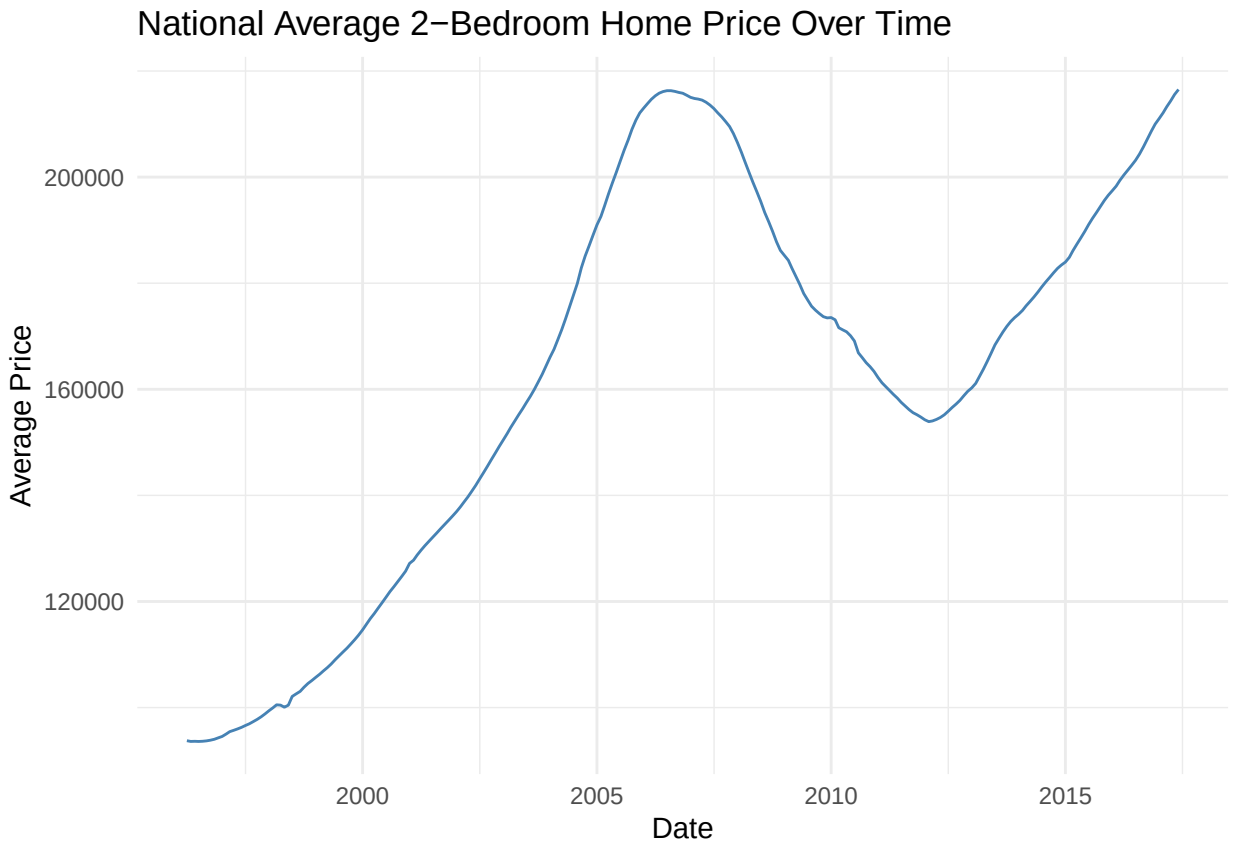
```r
# Build data dictionary for df_long
data_dict_long <- tibble(
  Variable = names(df_long),
  DataType = sapply(df_long, function(x) class(x)[1]),
  Category = case_when(
    sapply(df_long, is.numeric) ~ "Numerical",
    sapply(df_long, is.character) | sapply(df_long, is.factor) ~ "Categorical",
    inherits(df_long, "Date") ~ "Date",
    TRUE ~ "Other"
  ),
  Description = c(
    "Unique identifier for region",
    "ZIP code or area name",
    "City name",
    "State abbreviation",
    "Metro area name",
    "County name",
    "National size rank of the region",
    "Month of the observation (converted to Date)",
    "Zillow Home Value Index (ZHVI) for 2-bedroom homes"
  )
)

# View the data dictionary
```

```r
print(data_dict_long)
```

```
## # A tibble: 9 x 4
##   Variable   DataType  Category    Description
##   <chr>      <chr>     <chr>       <chr>
## 1 RegionID   integer   Numerical   Unique identifier for region
## 2 RegionName integer   Numerical   ZIP code or area name
## 3 City       character Categorical City name
## 4 State      character Categorical State abbreviation
## 5 Metro      character Categorical Metro area name
## 6 CountyName character Categorical County name
## 7 SizeRank   integer   Numerical   National size rank of the region
## 8 Date       Date      Other       Month of the observation (converted to Date)
## 9 Price      integer   Numerical   Zillow Home Value Index (ZHVI) for 2-bedroom~
```

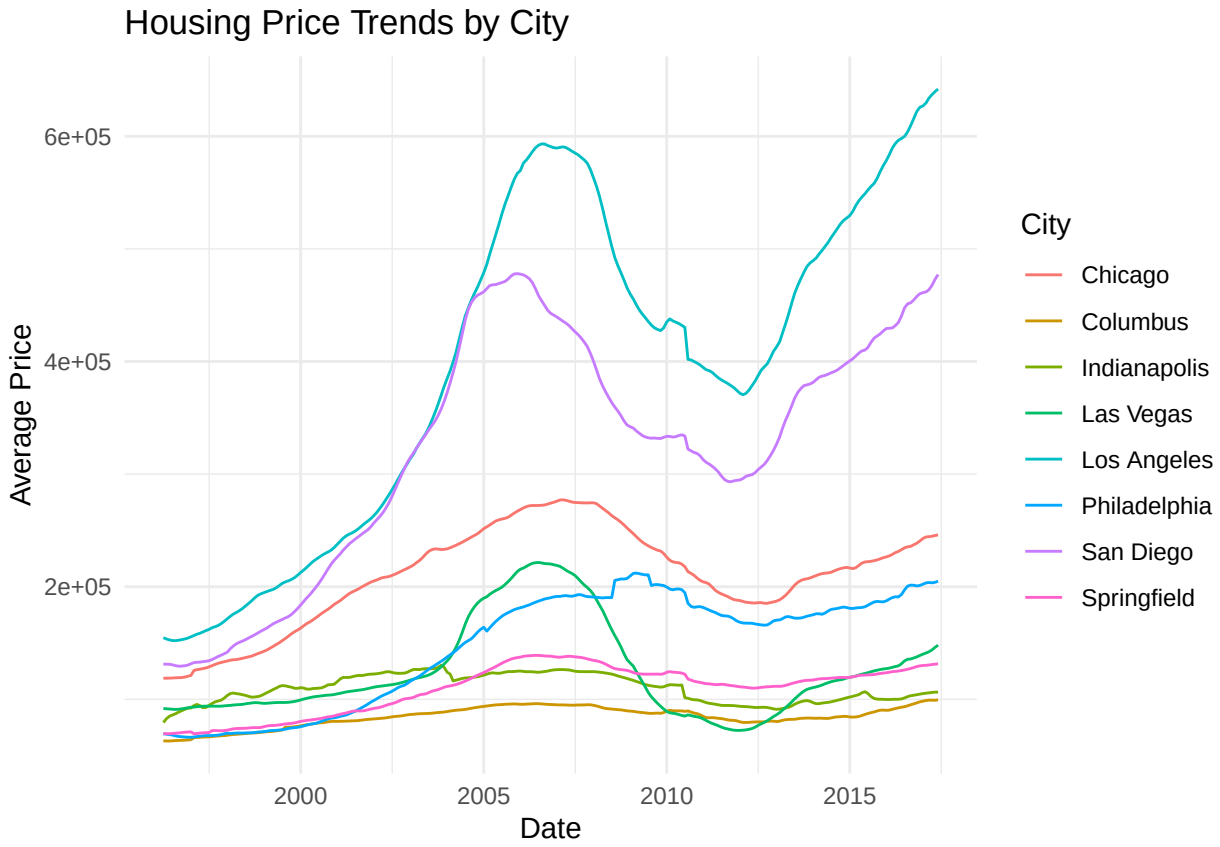
```r
df_long %>%
  group_by(Date) %>%
  summarise(AvgPrice = mean(Price, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = AvgPrice)) +
  geom_line(color = "steelblue") +
  labs(title = "National Average 2-Bedroom Home Price Over Time",
       x = "Date", y = "Average Price") +
  theme_minimal()
```



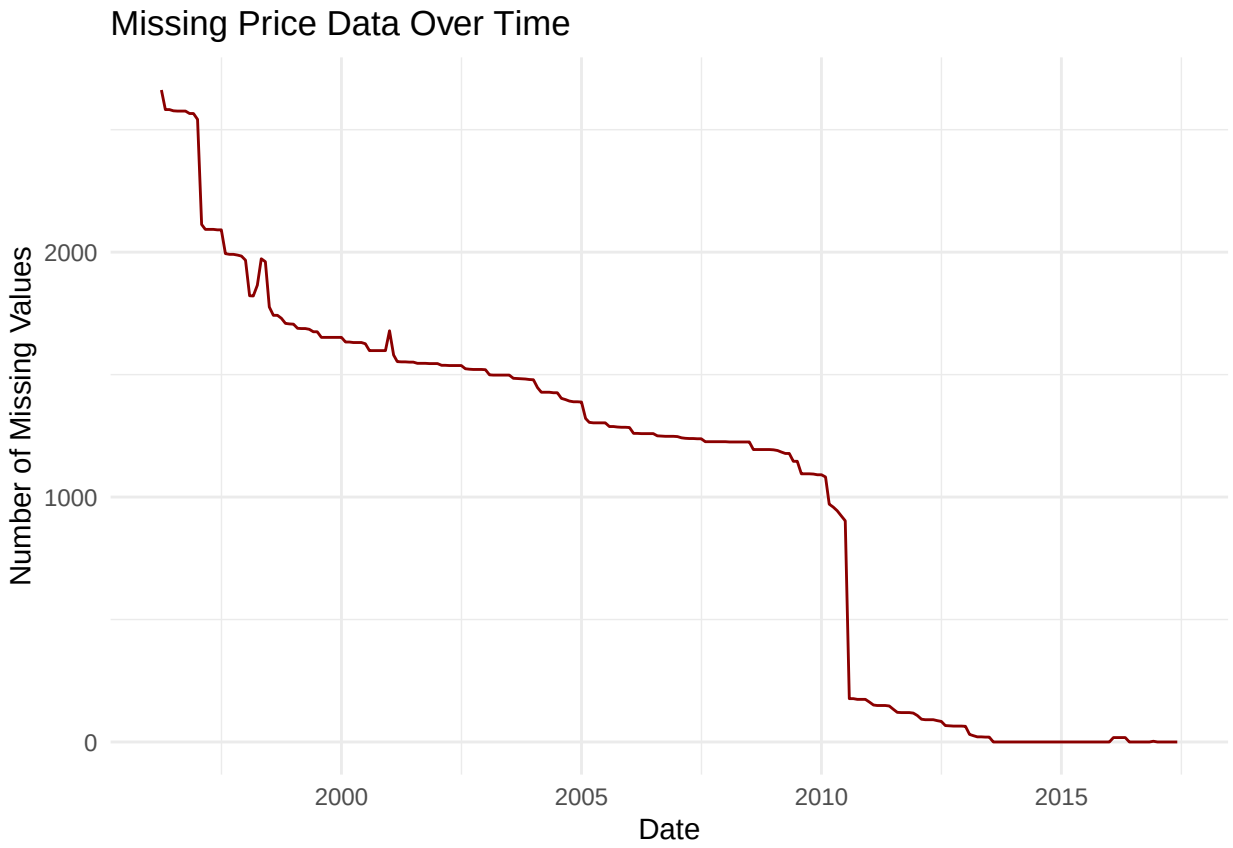

```r
top_cities <- df_long %>%
  count(City, sort = TRUE) %>%
  slice_max(n, n = 6) %>%
  pull(City)
```

```r
df_long %>%
  filter(City %in% top_cities) %>%
  group_by(City, Date) %>%
  summarise(AvgPrice = mean(Price, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = AvgPrice, color = City)) +
  geom_line() +
  labs(title = "Housing Price Trends by City", x = "Date", y = "Average Price") +
  theme_minimal()
```

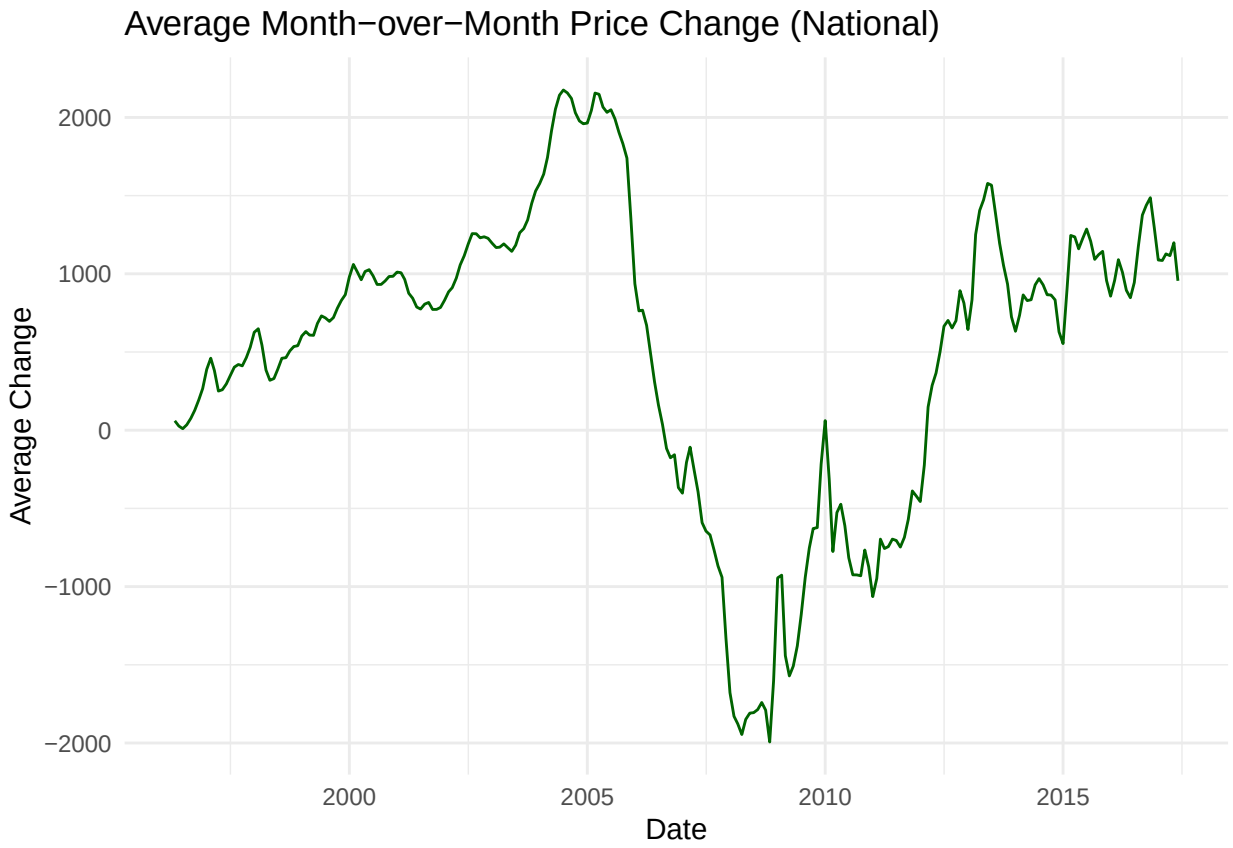
```
## `summarise()` has grouped output by 'City'. You can override using the
## `.groups` argument.
```




```r
df_long %>%
  group_by(Date) %>%
  summarise(MissingCount = sum(is.na(Price))) %>%
  ggplot(aes(x = Date, y = MissingCount)) +
  geom_line(color = "darkred") +
  labs(title = "Missing Price Data Over Time", x = "Date", y = "Number of Missing Values") +
  theme_minimal()
```

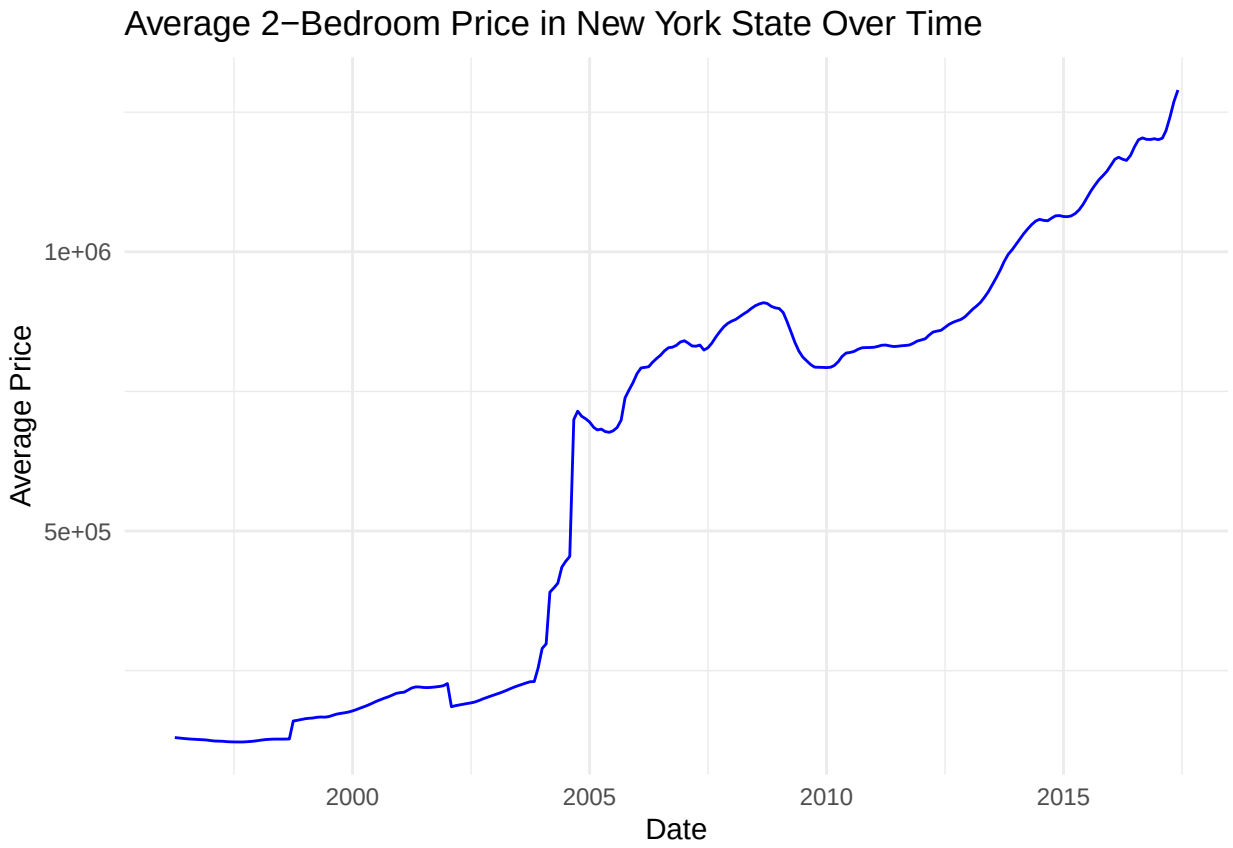



```
df_long %>%
  group_by(RegionID) %>%
  arrange(Date) %>%
  mutate(MoM_Change = Price - lag(Price)) %>%
  filter(!is.na(MoM_Change)) %>%
  group_by(Date) %>%
  summarise(AvgMoMChange = mean(MoM_Change, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = AvgMoMChange)) +
  geom_line(color = "darkgreen") +
  labs(title = "Average Month-over-Month Price Change (National)", x = "Date", y = "Average Change") +
  theme_minimal()
```




```
df_nyc <- df_long %>%
  filter(State == "NY", City == "New York")
```

```
df_nyc %>%
  group_by(Date) %>%
  summarise(AvgPrice = mean(Price, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = AvgPrice)) +
  geom_line(color = "blue") +
  labs(title = "Average 2-Bedroom Price in New York State Over Time",
       x = "Date", y = "Average Price") +
  theme_minimal()
```

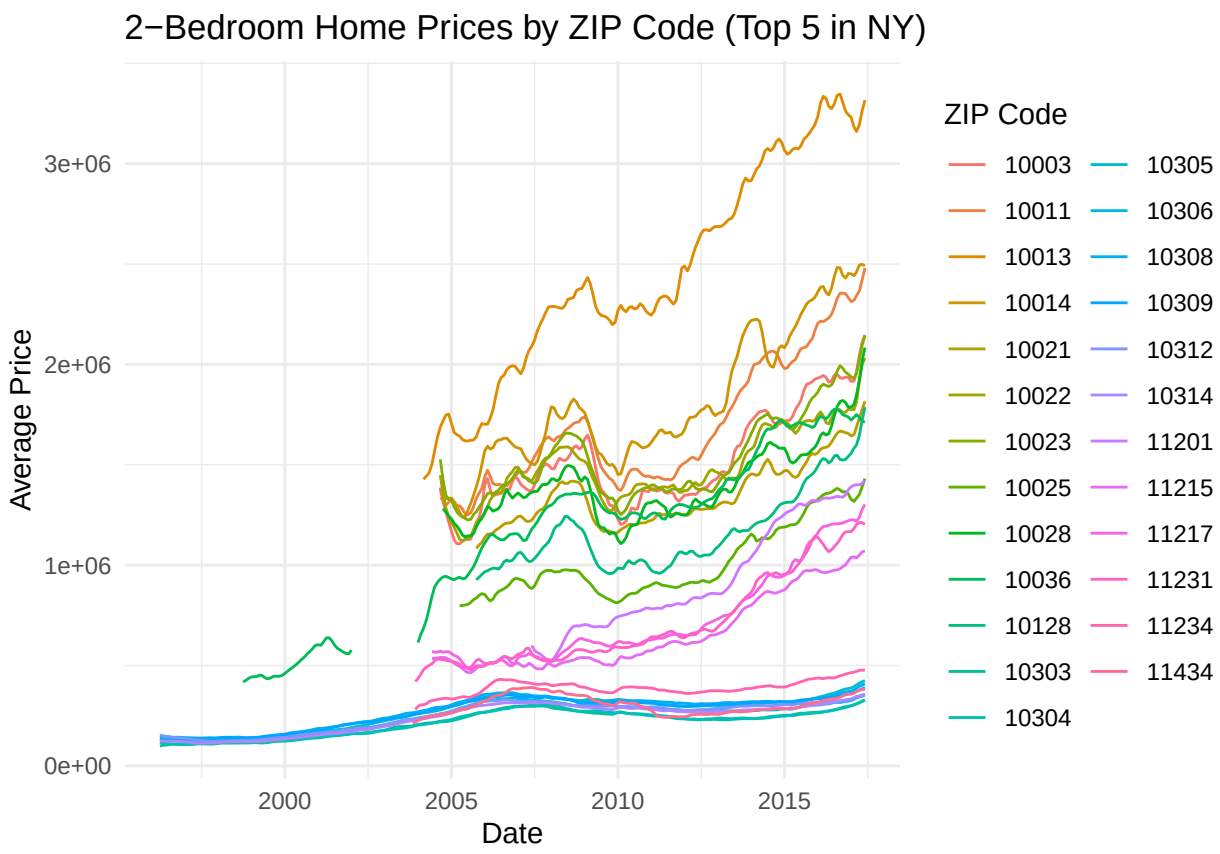



```
top_zips <- df_nyc %>%
  count(RegionName, sort = TRUE) %>%
  slice_max(n, n = 5) %>%
  pull(RegionName)

df_nyc %>%
  filter(RegionName %in% top_zips) %>%
  group_by(RegionName, Date) %>%
  summarise(AvgPrice = mean(Price, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = AvgPrice, color = as.factor(RegionName))) +
  geom_line() +
  labs(title = "2-Bedroom Home Prices by ZIP Code (Top 5 in NY)",
       x = "Date", y = "Average Price", color = "ZIP Code") +
  theme_minimal()
```

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

```
## Warning: Removed 1673 rows containing missing values or values outside the scale range
## (`geom_line()`).
```


2−Bedroom Home Prices by ZIP Code (Top 5 in NY)
Average Price
3e+06
2e+06
1e+06
0e+00
2000
2005
2010
2015
Date
ZIP Code
10003
10011
10013
10014
10021
10022
10023
10025
10028
10036
10128
10303
10304
10305
10306
10308
10309
10312
10314
11201
11215
11217
11231
11234
11434