1. Using either the same dataset(s) you used in the previous weeks’ exercise or a brand-new dataset of your choosing, perform the following transformations (Remember, anything you learn about the Housing dataset in these two weeks can be used for a later exercise!)
   1. Using the dplyr package, use the 6 different operations to analyze/transform the data - GroupBy, Summarize, Mutate, Filter, Select, and Arrange – Remember this isn’t just modifying data, you are learning about your data also – so play around and start to understand your dataset in more detail.

We will be using the dplyr package to analyze/transform the [**Housing dataset**](http://content.bellevue.edu/cst/dsc/520/id/resources/10-week-housing-data/week-6-housing.xlsx) (real estate transactions recorded from 1964 to 2016). Our goal is to further understand the real estate trends by grouping, summarizing, and arranging the data in different ways.

**6 different operations**

Grey – Comments

Blue – Code

Purple – Output

# Loading the data set week-6-housing

> housing <- read.csv(file = 'R Data/week-6-housing.csv')

# Creating a data frame from housing

> dfhousing <- data.frame(housing)

# Loading the dplyr package

> library(dplyr)

**# 1** **GroupBy**

# I would like to look at the max, min, and mean of the Sale Price by date

# We will use the group\_by() function from the dylpr package to do so

> dfhousing %>%

+ group\_by(ï..Sale.Date) %>%

+ summarise(Pmax = max(Sale.Price), Pmin= min(Sale.Price), Pmean = mean(Sale.Price), Pstd = sd(Sale.Price)

+ )

# A tibble: 2,933 x 5

ï..Sale.Date Pmax Pmin Pmean Pstd

<chr> <int> <int> <dbl> <dbl>

1 1/1/2009 578000 578000 578000 NA

2 1/1/2011 357505 357505 357505 NA

3 1/10/2006 513262 482000 497631 22106.

4 1/10/2007 878500 649000 754090 106383.

5 1/10/2008 730000 590000 660000 98995.

6 1/10/2011 782500 360000 522500 227500

7 1/10/2012 740000 435000 570000 155483.

8 1/10/2013 661307 536963 593318. 52213.

9 1/10/2014 783000 400000 569000 166582.

10 1/11/2006 765000 265000 467500 263190.

# ... with 2,923 more rows

# The above did not really give me what I was looking for. The individual dates make this messy. I would really like to see the average sale price by year. In order to do that we will need to transform the data before we can group.

**# 2** **Mutate**

# I will do this by creating a new variable “soldyear”

# To extract the year from solddate, we will create a new column with only the last 4 characters from solddate.

> soldyear1 <- 4

> soldyear <- substr(dfhousing$ï..Sale.Dat, nchar(dfhousing$ï..Sale.Dat) - soldyear1 + 1, nchar(dfhousing$ï..Sale.Dat))

> head(soldyear)

[1] "2006" "2006" "2006" "2006" "2006" "2006"

# Now that we have a new variable to group by, we need to add it back to the data frame using the Mutate function.

> dfhousing <- mutate(dfhousing, soldyear)

> names(dfhousing)

[1] "ï..Sale.Date" "Sale.Price" "sale\_reason"

[4] "sale\_instrument" "sale\_warning" "sitetype"

[7] "addr\_full" "zip5" "ctyname"

[10] "postalctyn" "lon" "lat"

[13] "building\_grade" "square\_feet\_total\_living" "bedrooms"

[16] "bath\_full\_count" "bath\_half\_count" "bath\_3qtr\_count"

[19] "year\_built" "year\_renovated" "current\_zoning"

[22] "sq\_ft\_lot" "prop\_type" "present\_use"

[25] "soldyear"

# We can see here that the new column has been added to the data frame.

**# 3** **Summarize**

# Did adding a year column help us narrow down the dates we want to average? How many years are in our data set? We can use the summarize function to summarize the distinct dates and years to see how many different years are within our data set.

> dfhousing %>% summarize(distinct\_years = n\_distinct(soldyear),

+ distinct\_days = n\_distinct(ï..Sale.Date))

distinct\_years distinct\_days

1 11 2933

# We can see from above that we really narrowed down the dates when looking at yearly statistic rather than daily. This can help remove any seasonality and just look at how the average price has changed year over year.

# We can visualize this by grouping the new data and plotting it’s summary.

> dfhousingsummary <- dfhousing %>%

+ group\_by(soldyear) %>%

+ summarise(Pmax = max(Sale.Price), Pmin= min(Sale.Price), Pmean = mean(Sale.Price), Pstd = sd(Sale.Price)

+ )

> dfhousingsummary

# A tibble: 11 x 5

soldyear Pmax Pmin Pmean Pstd

<chr> <int> <int> <dbl> <dbl>

1 2006 3000000 31272 622632. 280782.

2 2007 2988000 1000 668989. 342609.

3 2008 3995000 1500 824286. 721501.

4 2009 2000000 873 536502. 209784.

5 2010 4400000 698 582346. 311420.

6 2011 4380542 4000 656493. 685580.

7 2012 3462000 2031 613781. 465927.

8 2013 3340000 2500 607419. 230446.

9 2014 2280000 5150 659054. 246931.

10 2015 2300000 18000 714098. 259720.

11 2016 4311000 37800 791393. 365204.

> plot(dfhousingsummary$soldyear, dfhousingsummary$Pmean)

Chart, scatter chart

Description automatically generated

# We can easily see from the plot above that something is greatly increasing the average sales price in 2008. Is this due to an outlier or a true trend in that year? 2008 also has the highest standard deviation, so maybe something else is going on here?

**# 4** **Filter**

# Let’s take a closer look at 2008 by filtering on the year and looking at the sale price by bedrooms.

> filter08 <- filter(dfhousing, soldyear == "2008")

> filter08$ï..Sale.Date <- as.factor(filter08$ï..Sale.Date)

> ggplot(filter08, aes(bedrooms, Sale.Price)) +

geom\_boxplot(aes(group = cut\_width(bedrooms, 1)))

Chart, box and whisker chart

Description automatically generated

# We can see that the number of bedrooms makes an impact on the price. But why the outliers in the 2 and 3 bedroom homes?

**# 5 Select**

# Let’s select Sales Price and Bedrooms from the 2008 data

# First, we filter out bedroom over 3 then select Sales Price and Bedroom using pipes.

> filter083 %>%

> filter(filter083, bedrooms < 4) %>%

> select(Sale.Price, bedrooms)

Sale.Price bedrooms

1 420000 3

2 369900 3

3 599950 3

4 526787 3

5 165000 3

……

# This narrows it down, but how can we quickly see those outliers from the box chart above?

**# 6 Arrange**

# Lets assign the selected view to a new data frame.

> filter083 %>%

> select(Sale.Price, bedrooms) %>%

> arrange(filter083, desc(Sale.Price)) Sale.Price bedrooms

1 4400000 3

2 4400000 3

3 4380542 3

4 4311000 3

5 3850000 3

6 3462000 3

7 3462000 3

8 3462000 3

9 3462000 3

10 3462000 3

11 3175000 2

12 3175000 2

13 3175000 3

14 3175000 3

…..

# We can now see our top selling 2-3 bedroom homes for 2008 at the vary top of our list.

# However, it still don’t find this very helpful. There are quite a few homes listed above mean which contradicts what we assumed looking at the box plot from above.

# What other variable are affecting the prices of these homes? How do the 2 and 3 bedroom homes compare to others over all years? We can continue re-arranging and modifying the data until we uncover more, or use other methods such as regression or Support Vector Machines.

* 1. Using the purrr package – perform 2 functions on your dataset.  You could use zip\_n, keep, discard, compact, etc.

# After becoming more familiar with the data set, I realized that I did not check for “homes” that were only lots of land (0 bedrooms and 0 bathrooms). Assuming that this would mean that the sold price is only referring to a lot of land (or inhabited home), we want to exclude these from the data set.

# Using the purr package, lets check if there are 0s for bedrooms or bathrooms.

> housing$bedrooms %>% has\_element(0)

[1] TRUE

> housing$bath\_full\_count %>% has\_element(0)

[1] TRUE

# Now that we know that the data set contains listings that are only lots of land. We can exclude or sperate them in our analysis since land is appraised differently than property with a habitable home.

# We can use the keep function to isolate every home more than 0 bedrooms.

> keep(housing$bedrooms, function(x) x > 0)…

[1] 4 4 4 3 3 4 5 4 4 4 3 3 4 3 3 3 4 4 3 3 2 3 4 2 4 4 3 3 3 2 2 3 4 5 4 3 2 3 4 3 5 4 3 2 3 4 4 3 3 4 3

[52] 4 3 4 2 2 3 3 3 2 4 3 5 4 3 5 4 2 4 3 3 4

* 1. Use the cbind and rbind function on your dataset

# Now that can view our housing dataset by the year sold. Lets see how the two cities in the data set differ.

#First, I will create two summaries of the towns based off the yearold variable that we created earlier.

# One for REDMOND

> REDMOND\_Summary <- dfhousing %>%

+ group\_by(soldyear) %>%

+ filter(!! rlang::parse\_expr(filter\_string)) %>%

+ summarise(REmax = ctyname, max(Sale.Price), REmin= min(Sale.Price), REmean = mean(Sale.Price), REstd = sd(Sale.Price)

+ )

> REDMOND\_Summary

# A tibble: 6,721 x 6

# Groups: soldyear [11]

soldyear REmax `max(Sale.Price)` REmin REmean REstd

<chr> <chr> <int> <int> <dbl> <dbl>

1 2006 REDMOND 3000000 31272 593233. 223931.

2 2006 REDMOND 3000000 31272 593233. 223931.

3 2006 REDMOND 3000000 31272 593233. 223931.

4 2006 REDMOND 3000000 31272 593233. 223931.

5 2006 REDMOND 3000000 31272 593233. 223931.

6 2006 REDMOND 3000000 31272 593233. 223931.

7 2006 REDMOND 3000000 31272 593233. 223931.

8 2006 REDMOND 3000000 31272 593233. 223931.

9 2006 REDMOND 3000000 31272 593233. 223931.

10 2006 REDMOND 3000000 31272 593233. 223931.

……

# One for SAMMAMISH

> filter\_string <- "ctyname == 'SAMMAMISH'"

> SAMMAMISH\_Summary <- dfhousing %>%

+ group\_by(soldyear) %>%

+ filter(!! rlang::parse\_expr(filter\_string)) %>%

+ summarise(REmax = ctyname, max(Sale.Price), REmin= min(Sale.Price), REmean = mean(Sale.Price), REstd = sd(Sale.Price)

+ )

> SAMMAMISH\_Summary

# A tibble: 66 x 6

# Groups: soldyear [11]

soldyear REmax `max(Sale.Price)` REmin REmean REstd

<chr> <chr> <int> <int> <dbl> <dbl>

1 2006 SAMMAMISH 1650000 875000 1233529. 260929.

2 2006 SAMMAMISH 1650000 875000 1233529. 260929.

3 2006 SAMMAMISH 1650000 875000 1233529. 260929.

4 2006 SAMMAMISH 1650000 875000 1233529. 260929.

5 2006 SAMMAMISH 1650000 875000 1233529. 260929.

6 2006 SAMMAMISH 1650000 875000 1233529. 260929.

7 2006 SAMMAMISH 1650000 875000 1233529. 260929.

8 2007 SAMMAMISH 1230000 552000 829136. 231616.

9 2007 SAMMAMISH 1230000 552000 829136. 231616.

10 2007 SAMMAMISH 1230000 552000 829136. 231616.

……

# In order to plot the two together, we want to “bind” the two summaries into a new data frame using cbind.

# First lets ensure the number columns and rows are the same.

> dim(REDMOND\_Summary)

[1] 6721 6

> dim(SAMMAMISH\_Summary)

[1] 66 6

# It looks like the two are not of the same length.. why would that be? Look again, we appear to have duplicate rows… we can remove these using the unique function

> unRE\_Summary <- unique(REDMOND\_Summary)

> unRE\_Summary

# A tibble: 11 x 6

# Groups: soldyear [11]

soldyear REmax `max(Sale.Price)` REmin REmean REstd

<chr> <chr> <int> <int> <dbl> <dbl>

1 2006 REDMOND 3000000 31272 593233. 223931.

2 2007 REDMOND 2625000 5000 681037. 362287.

3 2008 REDMOND 2080000 20146 618172. 233983.

4 2009 REDMOND 2000000 29537 542527. 190080.

5 2010 REDMOND 2300000 55450 563357. 263462.

6 2011 REDMOND 4380542 38201 756237. 907962.

7 2012 REDMOND 3200000 4059 560233. 336174.

8 2013 REDMOND 3340000 2500 590076. 233718.

9 2014 REDMOND 1750000 7000 649091. 210635.

10 2015 REDMOND 2300000 132858 705500. 245209.

11 2016 REDMOND 4311000 37800 783759. 347112.

> unSA\_Summary <- unique(SAMMAMISH\_Summary)

> unSA\_Summary

# A tibble: 11 x 6

# Groups: soldyear [11]

soldyear REmax `max(Sale.Price)` REmin REmean REstd

<chr> <chr> <int> <int> <dbl> <dbl>

1 2006 SAMMAMISH 1650000 875000 1233529. 260929.

2 2007 SAMMAMISH 1230000 552000 829136. 231616.

3 2008 SAMMAMISH 990000 542000 673750 213742.

4 2009 SAMMAMISH 755000 755000 755000 NA

5 2010 SAMMAMISH 1715000 505000 1042000 607347.

6 2011 SAMMAMISH 1960000 701400 1024280 525598.

7 2012 SAMMAMISH 1895000 475000 1295200 751545.

8 2013 SAMMAMISH 2160200 563500 1247640 837106.

9 2014 SAMMAMISH 1000000 560000 823400 187954.

10 2015 SAMMAMISH 1150000 434000 927167. 226780.

11 2016 SAMMAMISH 1142600 665000 832086. 178942.

# Checking the number of columns and rows again.

> dim(unRE\_Summary)

[1] 11 6

> dim(unSA\_Summary)

[1] 11 6

# They are the same!

# Lets add the two together using rbind

> townsummaries <- rbind(unRE\_Summary, unSA\_Summary)

> townsummaries

A screenshot of a computer

Description automatically generated with medium confidence

# Perfect! We can now see that the two summaries have been bond together

# But what if we wanted to compare these to add the postalctyn column do our data set?

# We can do this by using the cbind function.

> dim(townsummaries)

[1] 22 6

> state <-rep (c('Ohio'), each = 22)

> dim(state)

[1] 22 1

> state

[1] "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio"

[16] "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio" "Ohio"

> townsummaries <- cbind(townsummaries, state)

> townsummaries

# A tibble: 22 x 7

# Groups: soldyear [11]

soldyear REmax `max(Sale.Price)` REmin REmean REstd ...7

<chr> <chr> <int> <int> <dbl> <dbl> <chr>

1 2006 REDMOND 3000000 31272 593233. 223931. Ohio

2 2007 REDMOND 2625000 5000 681037. 362287. Ohio

3 2008 REDMOND 2080000 20146 618172. 233983. Ohio

4 2009 REDMOND 2000000 29537 542527. 190080. Ohio

5 2010 REDMOND 2300000 55450 563357. 263462. Ohio

6 2011 REDMOND 4380542 38201 756237. 907962. Ohio

7 2012 REDMOND 3200000 4059 560233. 336174. Ohio

8 2013 REDMOND 3340000 2500 590076. 233718. Ohio

9 2014 REDMOND 1750000 7000 649091. 210635. Ohio

10 2015 REDMOND 2300000 132858 705500. 245209. Ohio

# ... with 12 more rows

# With the state added, we can compare these to other states!

* 1. Split a string, then concatenate the results back together

#Split string

> string = "SAMMAMISH"

> SAMstrsplit <- strsplit(string, split="A")

> SAMstrsplit

[[1]]

[1] "S" "MM" "MISH"

# Concatante the string

> SAMconcate <- paste(SAMstrsplit, collapse="A")

> SAMconcate

[1] "SAMMAMISH"

|  |  |  |
| --- | --- | --- |
|  | **R Markdown (.Rmd)** | **Markdown (.md)** |
| *Run R code* | Yes | No |
| *Bibliography* | Yes | No |
| *Task list* | Maybe | Yes |
| *MathJax* | Yes | Maybe |
| *HTML widgets* | Yes | Yes |