5.2 Exercise

Felipe Rodriguez

2022-01-15

```
library(readxl)
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(purrr)
```

```
setwd('/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/')

housing <- read_excel('/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/data/week-7-
```

Using the dplyr package, use the 6 different operations to analyze/transform the data - GroupBy, Summarize, Mutate, Filter, Select, and Arrange

Group By

```
housing %>%
  group_by(zip5) %>%
  summarise(mean(`Sale Price`), mean(square_feet_total_living))
```

```
## # A tibble: 4 x 3
##    zip5 `mean(\`Sale Price\`)` `mean(square_feet_total_living)`
##   <dbl>                  <dbl>                            <dbl>
## 1 98052                649375.                            2499.
## 2 98053                672624.                            2580.
## 3 98059                645000                             4360
## 4 98074                951544.                            3682.
```

Summarize

```
housing %>% summarize(mean(`Sale Price`))
```

```
## # A tibble: 1 x 1
##   ‘mean(\‘Sale Price\‘)‘
##                    <dbl>
## 1                660738.
```

Mutate

```
housing_mutate <- housing %>% mutate(sq_ft_price =`Sale Price`/sq_ft_lot)
select(housing_mutate, 'sq_ft_price')
```

```
## # A tibble: 12,865 x 1
##    sq_ft_price
##          <dbl>
##  1      105.
##  2      117.
##  3       67.8
##  4       43.8
##  5       49.1
##  6       25.4
##  7       10.8
##  8       28.5
##  9       15.5
## 10        6.85
## # ... with 12,855 more rows
```

Filter

```
housing %>% filter(zip5 == 98053)
```

```
## # A tibble: 5,339 x 24
##    ‘Sale Date‘         Sale Pric~1 sale_~2 sale_~3 sale_~4 sitet~5 addr_~6  zip5
##    <dttm>                    <dbl>   <dbl>   <dbl> <chr>   <chr>   <chr>   <dbl>
##  1 2006-01-03 00:00:00      184667       1      15 18 51   R1      8101 2~ 98053
##  2 2006-01-04 00:00:00     1050000       1       3 <NA>    R1      21634 ~ 98053
##  3 2006-01-04 00:00:00      875000       1       3 <NA>    R1      21404 ~ 98053
##  4 2006-01-04 00:00:00      660000       1       3 <NA>    R1      7525 2~ 98053
##  5 2006-01-04 00:00:00      165000       1       3 <NA>    R1      2921 2~ 98053
##  6 2006-01-05 00:00:00      803000       1       3 <NA>    R1      3624 2~ 98053
##  7 2006-01-06 00:00:00      765000       1       3 <NA>    R1      8944 2~ 98053
##  8 2006-01-09 00:00:00      372500       1       3 <NA>    R1      26920 ~ 98053
##  9 2006-01-10 00:00:00      513262       1       3 <NA>    R1      11807 ~ 98053
## 10 2006-01-10 00:00:00      482000       1       3 <NA>    R1      9166 2~ 98053
## # ... with 5,329 more rows, 16 more variables: ctyname <chr>, postalctyn <chr>,
## #   lon <dbl>, lat <dbl>, building_grade <dbl>, square_feet_total_living <dbl>,
## #   bedrooms <dbl>, bath_full_count <dbl>, bath_half_count <dbl>,
## #   bath_3qtr_count <dbl>, year_built <dbl>, year_renovated <dbl>,
## #   current_zoning <chr>, sq_ft_lot <dbl>, prop_type <chr>, present_use <dbl>,
## #   and abbreviated variable names 1: `Sale Price`, 2: sale_reason,
## #   3: sale_instrument, 4: sale_warning, 5: sitetype, 6: addr_full
```

Select

```
select(housing, `Sale Date`, `Sale Price`)
```

```
## # A tibble: 12,865 x 2
##    `Sale Date`         `Sale Price`
##    <dttm>                     <dbl>
##  1 2006-01-03 00:00:00       698000
##  2 2006-01-03 00:00:00       649990
##  3 2006-01-03 00:00:00       572500
##  4 2006-01-03 00:00:00       420000
##  5 2006-01-03 00:00:00       369900
##  6 2006-01-03 00:00:00       184667
##  7 2006-01-04 00:00:00      1050000
##  8 2006-01-04 00:00:00       875000
##  9 2006-01-04 00:00:00       660000
## 10 2006-01-04 00:00:00       650000
## # ... with 12,855 more rows
```

Arrange

```
housing %>% arrange(desc(`Sale Price`))
```

```
## # A tibble: 12,865 x 24
##    `Sale Date`         Sale Pric~1 sale_~2 sale_~3 sale_~4 sitet~5 addr_~6  zip5
##    <dttm>                    <dbl>   <dbl>   <dbl> <chr>   <chr>   <chr>   <dbl>
##  1 2010-03-02 00:00:00     4400000       1       3 35 45   R1      12025 ~ 98052
##  2 2010-03-02 00:00:00     4400000       1       3 35 45   R1      12053 ~ 98052
##  3 2011-11-17 00:00:00     4380542       1      22 11 45   R1      17137 ~ 98052
##  4 2011-11-17 00:00:00     4380542       1      22 11 45   R1      11818 ~ 98052
##  5 2011-11-17 00:00:00     4380542       1      22 11 45   R1      17011 ~ 98052
##  6 2011-11-17 00:00:00     4380542       1      22 11 45   R1      16943 ~ 98052
##  7 2011-11-17 00:00:00     4380542       1      22 11 45   R1      16944 ~ 98052
##  8 2011-11-17 00:00:00     4380542       1      22 11 45   R1      16909 ~ 98052
##  9 2011-11-17 00:00:00     4380542       1      22 11 45   R1      17128 ~ 98052
## 10 2011-11-17 00:00:00     4380542       1      22 11 45   R1      17136 ~ 98052
## # ... with 12,855 more rows, 16 more variables: ctyname <chr>,
## #   postalctyn <chr>, lon <dbl>, lat <dbl>, building_grade <dbl>,
## #   square_feet_total_living <dbl>, bedrooms <dbl>, bath_full_count <dbl>,
## #   bath_half_count <dbl>, bath_3qtr_count <dbl>, year_built <dbl>,
## #   year_renovated <dbl>, current_zoning <chr>, sq_ft_lot <dbl>,
## #   prop_type <chr>, present_use <dbl>, and abbreviated variable names
## #   1: `Sale Price`, 2: sale_reason, 3: sale_instrument, 4: sale_warning, ...
```

Using the purrr package – perform 2 functions on your dataset.

discard

```
discard_purrr <- housing$ctyname %>% discard(is.na)
head(discard_purrr)
```

```
## [1] "REDMOND" "REDMOND" "REDMOND" "REDMOND" "REDMOND" "REDMOND"
```

has_element

```
housing$ctyname %>% has_element("OMAHA")
```

```
## [1] FALSE
```

Use the cbind and rbind function on your dataset

cbind

```
cbind_function <- cbind(Sale_Reason=housing$sale_reason, Sale_Price=housing$`Sale Price`)
head(cbind_function)
```

```
##      Sale_Reason Sale_Price
## [1,]           1     698000
## [2,]           1     649990
## [3,]           1     572500
## [4,]           1     420000
## [5,]           1     369900
## [6,]           1     184667
```

rbind

```
message("Number of rows before rbind")
```

```
## Number of rows before rbind
```

```
nrow(housing)
```

```
## [1] 12865
```

```
new_rows <- head(housing, 4)
housing_rbind <-rbind(housing, new_rows)
message("Number of rows after rbind")
```

```
## Number of rows after rbind
```

```
nrow(housing_rbind)
```

```
## [1] 12869
```

Split a string, then concatenate the results back together

```
library(stringr)
date <- str_split(string = housing$`Sale Date`, pattern="-")
date_matrix <- data.frame(Reduce(rbind, date))
names(date_matrix) <- c("Year_Sold", "Month_Sold", "Day_Sold")
head(date_matrix)
```

```
##      Year_Sold Month_Sold Day_Sold
## init      2006         01       03
## X         2006         01       03
## X.1       2006         01       03
## X.2       2006         01       03
## X.3       2006         01       03
## X.4       2006         01       03
```

```
new_housing <- cbind(housing, date_matrix)
head(select(new_housing, "Year_Sold", "Month_Sold", "Day_Sold"))
```

```
##      Year_Sold Month_Sold Day_Sold
## init      2006         01       03
## X         2006         01       03
## X.1       2006         01       03
## X.2       2006         01       03
## X.3       2006         01       03
## X.4       2006         01       03
```