DSC520_Week5_Guruprasad_Velikadu_Krishnamoorthy

Guruprasad Velikadu Krishnamoorthy

2023-01-15

Assignment Week 5

Loading the required Packages

```
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(stringr)
library(purrr)
```

Set the working directory to the root of your DSC 520 directory and initial settings

```
knitr::opts_knit$set(root.dir ="C:/Users/Gurup/GURU/Learning/Masters/Term_2/DSC520_T302_Statistics_for_
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 120), tidy = TRUE)
```

A.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

```
# Reading from the Excel
excel_path = "data/week-6-housing.xlsx"
housing_df <- read_excel(excel_path)
```

```r
# Renaming a few columns with desired format
housing_df <- housing_df %>%
    rename(Sale_Date = "Sale Date")
housing_df <- housing_df %>%
    rename(Sale_Price = "Sale Price")

# group_by and summarize Example 1-This returns the Average sales price per year built
housing_df %>%
    group_by(year_built) %>%
    summarize(Avg_SalePrice = mean(Sale_Price))
```

```
## # A tibble: 109 x 2
##    year_built Avg_SalePrice
##         <dbl>         <dbl>
##  1       1900       394500.
##  2       1903       430000
##  3       1905       620000
##  4       1906       550000
##  5       1909         1070
##  6       1910       150000
##  7       1912       619667.
##  8       1913       457500
##  9       1914       835000
## 10       1915       228150
## # ... with 99 more rows
```

```r
# Example 2- This returns the Average Lot size based on Zipcode
housing_df %>%
    group_by(zip5) %>%
    summarize(Avg_lotSize = mean(sq_ft_lot))
```

```
## # A tibble: 4 x 2
##    zip5 Avg_lotSize
##   <dbl>       <dbl>
## 1 98052      11960.
## 2 98053      36247.
## 3 98059      39109
## 4 98074      45004.
```

```r
# mutate and select Create a new field `Dimensions`
housing_df %>%
    select(square_feet_total_living, bedrooms, bath_full_count, bath_half_count) %>%
    mutate(Dimensions = sprintf("%s Sq Foot, %s Bed %s Full-Bath %s half-bath home ", square_feet_total_
        bath_full_count, bath_half_count))
```

```
## # A tibble: 12,865 x 5
##    square_feet_total_living bedrooms bath_full_count bath_half_count Dimensions
##                       <dbl>    <dbl>           <dbl>           <dbl> <chr>
##  1                     2810        4               2               1 "2810 Sq F~
##  2                     2880        4               2               0 "2880 Sq F~
##  3                     2770        4               1               1 "2770 Sq F~
```

```
##  4                    1620        3               1               0 "1620 Sq F~
##  5                    1440        3               1               0 "1440 Sq F~
##  6                    4160        4               2               1 "4160 Sq F~
##  7                    3960        5               3               0 "3960 Sq F~
##  8                    3720        4               2               1 "3720 Sq F~
##  9                    4160        4               2               1 "4160 Sq F~
## 10                    2760        4               1               0 "2760 Sq F~
## # ... with 12,855 more rows
```

```
housing_df1 <- housing_df

# create a new dataframe housing_df1 with a few columns selected and add a new column `Year_of_Sale`
housing_df1 %<>%
    mutate(Year_of_Sale = as.numeric(format(Sale_Date, "%Y"))) %>%
    select(Sale_Date, Year_of_Sale, Sale_Price, year_built)
housing_df1
```

```
## # A tibble: 12,865 x 4
##    Sale_Date           Year_of_Sale Sale_Price year_built
##    <dttm>                     <dbl>      <dbl>      <dbl>
##  1 2006-01-03 00:00:00         2006     698000       2003
##  2 2006-01-03 00:00:00         2006     649990       2006
##  3 2006-01-03 00:00:00         2006     572500       1987
##  4 2006-01-03 00:00:00         2006     420000       1968
##  5 2006-01-03 00:00:00         2006     369900       1980
##  6 2006-01-03 00:00:00         2006     184667       2005
##  7 2006-01-04 00:00:00         2006    1050000       1993
##  8 2006-01-04 00:00:00         2006     875000       1988
##  9 2006-01-04 00:00:00         2006     660000       1978
## 10 2006-01-04 00:00:00         2006     650000       1976
## # ... with 12,855 more rows
```

```
# Using Filter to restrict the records
filtered_df <- housing_df1 %>%
    filter(Sale_Price > 1e+06 & Year_of_Sale > 2010)

# Using Arrange
filtered_df %>%
    arrange(desc(Sale_Price), Year_of_Sale)
```

```
## # A tibble: 566 x 4
##    Sale_Date           Year_of_Sale Sale_Price year_built
##    <dttm>                     <dbl>      <dbl>      <dbl>
##  1 2011-11-17 00:00:00         2011    4380542       2012
##  2 2011-11-17 00:00:00         2011    4380542       2010
##  3 2011-11-17 00:00:00         2011    4380542       2012
##  4 2011-11-17 00:00:00         2011    4380542       2012
##  5 2011-11-17 00:00:00         2011    4380542       2010
##  6 2011-11-17 00:00:00         2011    4380542       2012
##  7 2011-11-17 00:00:00         2011    4380542       2012
##  8 2011-11-17 00:00:00         2011    4380542       2012
##  9 2011-11-17 00:00:00         2011    4380542       2010
## 10 2011-11-17 00:00:00         2011    4380542       2011
## # ... with 556 more rows
```

B.Using the purrr package – perform 2 functions on your dataset. You could use zip_n, keep, discard, compact, etc.

```
# Creating Average sales list
AvgSales_df <- housing_df1 %>%
    group_by(Year_of_Sale) %>%
    summarise(AvgSales = mean(Sale_Price)) %>%
    select(Year_of_Sale, AvgSales)
Avgsales_list <- list(AvgSales_df)
# Function 1: Use keep function to check if all Sale price in the list is >500K
keep(Avgsales_list, ~all(.x$AvgSales > 5e+05))
```

```
## [[1]]
## # A tibble: 11 x 2
##    Year_of_Sale AvgSales
##           <dbl>    <dbl>
##  1         2006  622632.
##  2         2007  668989.
##  3         2008  824286.
##  4         2009  536502.
##  5         2010  582346.
##  6         2011  656493.
##  7         2012  613781.
##  8         2013  607419.
##  9         2014  659054.
## 10         2015  714098.
## 11         2016  791393.
```

```
# Function 2: Discard function to discard the years after 2010.
discard(AvgSales_df$Year_of_Sale, ~.x > 2010)
```

```
## [1] 2006 2007 2008 2009 2010
```

```
# Function 3: Using map_dbl function to add a column to housing_df1
housing_df1 <- housing_df1 %>%
    mutate(age_of_prop_when_bought = map2_dbl(Year_of_Sale, -1 * (year_built), sum))

sale_price_fn <- function(x) {
    ifelse(x > 20000, x, 0)
}

# Function 4: Using compose function from Purrr to chain 2 functions
compose_fn <- compose(round, sale_price_fn)
# Function 5: Removing the nulls using compact and restricting resultset to 20 for display purpose
compose_fn(housing_df1$Sale_Price)[1:20] %>%
    compact()
```

```
##  [1]  698000  649990  572500  420000  369900  184667 1050000  875000  660000
## [10]  650000  599950  526787  470000  165000  803000  507950  765000  589950
## [19]  501000  372500
```

C. Use the cbind and rbind function on your dataset

```
# Create a single column dataframes of length of 20
address <- housing_df[1:20, "addr_full"]
postal_city_name <- housing_df[1:20, "postalctyn"]
zip <- housing_df[1:20, "zip5"]
# Use Cbind to combine the columns
Redmond_20_df <- cbind(address, postal_city_name, zip)
dim(Redmond_20_df)
```

```
## [1] 20  3
```

```
# create a new Dataframe
Sammamish_df <- housing_df %>%
    filter(ctyname %in% "SAMMAMISH") %>%
    select(addr_full, postalctyn, zip5)
dim(Sammamish_df)
```

```
## [1] 66  3
```

```
# Use rbind to combine both dataframes
rbind_df <- rbind(Sammamish_df, Redmond_20_df)
# Results shows that rows from both Dataframes are combined
dim(rbind_df)
```

```
## [1] 86  3
```

```
head(rbind_df)
```

```
## # A tibble: 6 x 3
##   addr_full        postalctyn  zip5
##   <chr>            <chr>      <dbl>
## 1 24620 NE 27TH PL REDMOND    98074
## 2 24628 NE 27TH PL REDMOND    98074
## 3 2005 250TH PL NE REDMOND    98074
## 4 2250 246TH PL NE REDMOND    98074
## 5 2208 247TH CT NE REDMOND    98074
## 6 2219 246TH PL NE REDMOND    98074
```

D. Split a string, then concatenate the results back together

```
# Split the addr_full column
split_string <- str_split(string = rbind_df$addr_full, pattern = " ")
# Include only the list with 4 elements for the purpose of this demonstration
split_string <- ifelse(sapply(split_string, length) == 4, split_string, NA)
# Check the number of elements in split string
length(split_string)
```

```
## [1] 86
```

```r
# Function to extract the House number from the split string
extract_house_nbr_func <- function(x) {
    x[[1]]
}

# Other way to Extract the house number from the street address
str_extract(string = rbind_df$addr_full, pattern = "^\\d{1,5}")
```

```
##  [1] "24620" "24628" "2005"  "2250"  "2208"  "2219"  "24424" "2530"  "24649"
## [10] "2004"  "24633" "2041"  "2533"  "2522"  "2503"  "2412"  "24406" "2413"
## [19] "2516"  "2515"  "2210"  "2554"  "2036"  "2030"  "2005"  "2402"  "24617"
## [28] "2527"  "2203"  "2234"  "2006"  "2205"  "2205"  "2030"  "2030"  "2503"
## [37] "2503"  "2030"  "24424" "2521"  "2548"  "2006"  "2006"  "24633" "2512"
## [46] "2411"  "2413"  "24424" "2218"  "24526" "3100"  "24439" "2200"  "2250"
## [55] "2211"  "24425" "2028"  "24531" "2028"  "2533"  "24406" "2403"  "3108"
## [64] "2219"  "2515"  "2415"  "17021" "11927" "13315" "3303"  "16126" "8101"
## [73] "21634" "21404" "7525"  "17703" "14924" "7858"  "17905" "2921"  "3624"
## [82] "7850"  "8944"  "11922" "3201"  "26920"
```

```r
# Function to extract street details by combining string element 2,3 and 4
extract_street_addr_func <- function(x) {
    paste(x[[2]], x[[3]], x[[4]])
}

# Add new columns house_nbr, street_name by splitting the string
rbind_df <- rbind_df %>%
    mutate(house_nbr = map_chr(split_string, extract_house_nbr_func), street_name = map_chr(split_string
rbind_df
```

```
## # A tibble: 86 x 5
##    addr_full          postalctyn  zip5 house_nbr street_name
##    <chr>              <chr>      <dbl> <chr>     <chr>
##  1 24620 NE 27TH PL   REDMOND    98074 24620     NE 27TH PL
##  2 24628 NE 27TH PL   REDMOND    98074 24628     NE 27TH PL
##  3 2005 250TH PL NE   REDMOND    98074 2005      250TH PL NE
##  4 2250 246TH PL NE   REDMOND    98074 2250      246TH PL NE
##  5 2208 247TH CT NE   REDMOND    98074 2208      247TH CT NE
##  6 2219 246TH PL NE   REDMOND    98074 2219      246TH PL NE
##  7 24424 NE 27TH PL   REDMOND    98074 24424     NE 27TH PL
##  8 2530 248TH TER NE  REDMOND    98074 2530      248TH TER NE
##  9 24649 NE 22ND ST   REDMOND    98074 24649     NE 22ND ST
## 10 2004 247TH PL NE   REDMOND    98074 2004      247TH PL NE
## # ... with 76 more rows
```

```r
# Combining the split string
addr_matrix <- data.frame(Reduce(rbind, split_string))
rbind_df$street_address <- with(addr_matrix, paste(X2, X3, X4))
# Adding a new column Complete address by combining strings
rbind_df$complete_address <- with(rbind_df, paste(house_nbr, street_address, postalctyn, zip5))
# Final output that shows the address that was split and combined to get the complete address
rbind_df %>%
    select(house_nbr, street_address, complete_address)
```

```
## # A tibble: 86 x 3
##    house_nbr street_address complete_address
##    <chr>     <chr>          <chr>
##  1 24620     NE 27TH PL     24620 NE 27TH PL REDMOND 98074
##  2 24628     NE 27TH PL     24628 NE 27TH PL REDMOND 98074
##  3 2005      250TH PL NE    2005 250TH PL NE REDMOND 98074
##  4 2250      246TH PL NE    2250 246TH PL NE REDMOND 98074
##  5 2208      247TH CT NE    2208 247TH CT NE REDMOND 98074
##  6 2219      246TH PL NE    2219 246TH PL NE REDMOND 98074
##  7 24424     NE 27TH PL     24424 NE 27TH PL REDMOND 98074
##  8 2530      248TH TER NE   2530 248TH TER NE REDMOND 98074
##  9 24649     NE 22ND ST     24649 NE 22ND ST REDMOND 98074
## 10 2004      247TH PL NE    2004 247TH PL NE REDMOND 98074
## # ... with 76 more rows
```

Session info

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22621)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] purrr_0.3.5   stringr_1.4.1 dplyr_1.0.10  readxl_1.4.1
##
## loaded via a namespace (and not attached):
##  [1] rstudioapi_0.14  knitr_1.41       magrittr_2.0.3   tidyselect_1.2.0
##  [5] R6_2.5.1         rlang_1.0.6      fastmap_1.1.0    fansi_1.0.3
##  [9] tools_4.2.2      xfun_0.34        utf8_1.2.2       DBI_1.1.3
## [13] cli_3.4.1        withr_2.5.0      htmltools_0.5.3  assertthat_0.2.1
## [17] yaml_2.3.6       digest_0.6.30    tibble_3.1.8     lifecycle_1.0.3
## [21] formatR_1.12     vctrs_0.5.0      glue_1.6.2       evaluate_0.18
## [25] rmarkdown_2.18   stringi_1.7.8    compiler_4.2.2   pillar_1.8.1
## [29] cellranger_1.1.0 generics_0.1.3   pkgconfig_2.0.3
```