8.2 Exercise, The Housing Data

Decker, Reuben

2023-10-22

Pulling in the Housing Data

```
## 1 Pull in the housing data
## a. Work individually on this assignment. You are encouraged to collaborate on ideas and strat
egies pertinent to this assignment. Data for this assignment is focused on real estate transacti
ons recorded from 1964 to 2016 and can be found in Housing.xlsx. Using your skills in statistica
l correlation, multiple regression, and R programming, you are interested in the following varia
bles: Sale Price and several other possible predictors.
## Download the CSV file
housing_data <- read_excel("C:\\Users\\Reuben Decker\\Downloads\\housing data.xlsx")

## Seeing the CSV file
head(housing_data, 10)
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>	addr_full <chr>
2006-01-03	698000	1	3	*NA*	R1	17021 NE 113TH
2006-01-03	649990	1	3	*NA*	R1	11927 178TH PL
2006-01-03	572500	1	3	*NA*	R1	13315 174TH AV
2006-01-03	420000	1	3	*NA*	R1	3303 178TH AVE
2006-01-03	369900	1	3	15	R1	16126 NE 108TH
2006-01-03	184667	1	15	18 51	R1	8101 229TH DR
2006-01-04	1050000	1	3	*NA*	R1	21634 NE 87TH
2006-01-04	875000	1	3	*NA*	R1	21404 NE 67TH
2006-01-04	660000	1	3	*NA*	R1	7525 238TH AVE
2006-01-04	650000	1	3	*NA*	R1	17703 NE 26TH

1-10 of 10 rows | 1-8 of 24 columns

```
tail(housing_data, 10)
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>	addr_full <chr>
2016-12-14	180637	1	3	*NA*	R2	15793 NE 114TH

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>	addr_full <chr>
2016-12-15	865000	1	3	*NA*	R1	7710 246TH AVE
2016-12-15	840000	1	3	*NA*	R1	23004 NE 64TH
2016-12-15	833000	1	3	*NA*	R1	18522 NE 102NI
2016-12-15	824000	1	3	*NA*	R1	11314 177TH PL
2016-12-15	798930	1	3	*NA*	R1	22506 NE 102NI
2016-12-15	750000	1	3	*NA*	R1	13315 175TH AV
2016-12-15	629000	1	3	*NA*	R1	17716 NE 29TH
2016-12-16	835000	1	3	*NA*	R1	9917 182ND CT
2016-12-16	455500	1	3	*NA*	R1	8826 166TH AVE

1-10 of 10 rows | 1-8 of 24 columns

Viewing the Data

```
## 1.a.i
## If you worked with the Housing dataset in previous week - you are in luck, you likely have al
ready found any issues in the dataset and made the necessary transformations. If not, you will w
ant to take some time looking at the data with all your new skills and identifying if you have a
ny clean up that needs to happen.

## It is usually best to see what the data is like
## Viewing what type of class the columns are
sapply(housing_data, class)
```

```
$`Sale Date`
[1] "POSIXct" "POSIXt"

$`Sale Price`
[1] "numeric"

$sale_reason
[1] "numeric"

$sale_instrument
[1] "numeric"

$sale_warning
[1] "character"

$sitetype
[1] "character"

$addr_full
[1] "character"

$zip5
[1] "numeric"

$ctyname
[1] "character"

$postalctyn
[1] "character"

$lon
[1] "numeric"

$lat
[1] "numeric"

$building_grade
[1] "numeric"

$square_feet_total_living
[1] "numeric"

$bedrooms
[1] "numeric"

$bath_full_count
[1] "numeric"

$bath_half_count
[1] "numeric"

$bath_3qtr_count
```

```
[1] "numeric"

$year_built
[1] "numeric"

$year_renovated
[1] "numeric"

$current_zoning
[1] "character"

$sq_ft_lot
[1] "numeric"

$prop_type
[1] "character"

$present_use
[1] "numeric"
```

```
## Viewing the data at a glance
glimpse(housing_data)
```

```
Rows: 12,865
Columns: 24
$ `Sale Date`              <dttm> 2006-01-03, 2006-01-03, 2006-01-03, 2006-01-…
$ `Sale Price`             <dbl> 698000, 649990, 572500, 420000, 369900, 18466…
$ sale_reason              <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ sale_instrument          <dbl> 3, 3, 3, 3, 3, 15, 3, 3, 3, 3, 3, 3, 3, 3, 3,…
$ sale_warning             <chr> NA, NA, NA, NA, "15", "18 51", NA, NA, NA, NA…
$ sitetype                 <chr> "R1", "R1", "R1", "R1", "R1", "R1", "R1", "R1…
$ addr_full                <chr> "17021 NE 113TH CT", "11927 178TH PL NE", "13…
$ zip5                     <dbl> 98052, 98052, 98052, 98052, 98052, 98053, 980…
$ ctyname                  <chr> "REDMOND", "REDMOND", NA, "REDMOND", "REDMOND…
$ postalctyn               <chr> "REDMOND", "REDMOND", "REDMOND", "REDMOND", "…
$ lon                      <dbl> -122.1124, -122.1022, -122.1085, -122.1037, -…
$ lat                      <dbl> 47.70139, 47.70731, 47.71986, 47.63914, 47.69…
$ building_grade           <dbl> 9, 9, 8, 8, 7, 7, 10, 10, 9, 8, 9, 8, 8, 9, 1…
$ square_feet_total_living <dbl> 2810, 2880, 2770, 1620, 1440, 4160, 3960, 372…
$ bedrooms                 <dbl> 4, 4, 4, 3, 3, 4, 5, 4, 4, 4, 3, 3, 4, 3, 3, …
$ bath_full_count          <dbl> 2, 2, 1, 1, 1, 2, 3, 2, 2, 1, 2, 2, 1, 2, 2, …
$ bath_half_count          <dbl> 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, …
$ bath_3qtr_count          <dbl> 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, …
$ year_built               <dbl> 2003, 2006, 1987, 1968, 1980, 2005, 1993, 198…
$ year_renovated           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ current_zoning           <chr> "R4", "R4", "R6", "R4", "R6", "URPSO", "RA5",…
$ sq_ft_lot                <dbl> 6635, 5570, 8444, 9600, 7526, 7280, 97574, 30…
$ prop_type                <chr> "R", "R", "R", "R", "R", "R", "R", "R", "R", …
$ present_use              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
```

```
## NOTE: that the Sale Date classes are POSIXct and POSIXt, and as POSIXct is a subclass of POSI
Xt nothing needs to change.
```

Finding Errors

Duplicates

```
## It is also good to check the data for duplicates
## Checking the data for duplicates

## Counting how many rows are duplicates
duplicate_count <- sum(duplicated(housing_data))
duplicate_count
```

```
[1] 4
```

```
# Create a logical vector marking duplicate rows as TRUE
duplicate_logical <- duplicated(housing_data) | duplicated(housing_data, fromLast = TRUE)

# Subset the data frame to get the duplicate rows
duplicate_rows <- housing_data[duplicate_logical, ]
duplicate_rows
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>	addr_full <chr>
2006-05-10	561000	1	3	*NA*	R1	7518 146TH AVE
2006-05-10	561000	1	3	*NA*	R1	7518 146TH AVE
2006-09-14	650000	1	3	*NA*	R1	22853 NE 42ND
2006-09-14	650000	1	3	*NA*	R1	22853 NE 42ND
2016-08-29	685000	1	3	*NA*	R1	16713 NE 87TH
2016-08-29	685000	1	3	*NA*	R1	16713 NE 87TH
2016-09-08	265000	1	3	*NA*	R1	8508 171ST AVE
2016-09-08	265000	1	3	*NA*	R1	8508 171ST AVE

8 rows | 1-8 of 24 columns

```
View(duplicate_rows)


## Counting how many total duplicates
total_duplicate_count <- sum(duplicated(housing_data) | duplicated(housing_data, fromLast = TRU
E))
total_duplicate_count
```

```
[1] 8
```

```
## There are 4 sets of duplicate rows, so we need just unique instances.This gives us cleaned up
rows for duplicates.

housing_data <- unique(housing_data)
head(housing_data, 10)
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>	addr_full <chr>
2006-01-03	698000	1	3	*NA*	R1	17021 NE 113TH
2006-01-03	649990	1	3	*NA*	R1	11927 178TH PL
2006-01-03	572500	1	3	*NA*	R1	13315 174TH AV
2006-01-03	420000	1	3	*NA*	R1	3303 178TH AVE
2006-01-03	369900	1	3	15	R1	16126 NE 108TH
2006-01-03	184667	1	15	18 51	R1	8101 229TH DR
2006-01-04	1050000	1	3	*NA*	R1	21634 NE 87TH
2006-01-04	875000	1	3	*NA*	R1	21404 NE 67TH
2006-01-04	660000	1	3	*NA*	R1	7525 238TH AVE
2006-01-04	650000	1	3	*NA*	R1	17703 NE 26TH

1-10 of 10 rows | 1-8 of 24 columns

NAs

```
##Checking the data for NA rows (only showing the first and last 5 as place holders)
rows_with_na <- !complete.cases(housing_data)
head(rows_with_na, 5)
```

```
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

```
tail(rows_with_na, 5)
```

```
[1] TRUE TRUE TRUE TRUE TRUE
```

```
## Counting how many rows have NAs
count_true2 <- sum(rows_with_na)
count_true2
```

```
[1] 11753
```

```
## Viewing rows with NAs
housing_data[rows_with_na, ]
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sale_warning <chr>	sitetype <chr>
2006-01-03	698000	1	3	*NA*	R1
2006-01-03	649990	1	3	*NA*	R1
2006-01-03	572500	1	3	*NA*	R1
2006-01-03	420000	1	3	*NA*	R1
2006-01-03	184667	1	15	18 51	R1
2006-01-04	1050000	1	3	*NA*	R1
2006-01-04	875000	1	3	*NA*	R1
2006-01-04	660000	1	3	*NA*	R1
2006-01-04	650000	1	3	*NA*	R1
2006-01-04	599950	1	3	*NA*	R1

1-10 of 10,000 rows | 1-6 of 24 columns Previous **1** 2 3 4 5 6 … 1000 Next

```
## Counting how many total NAs per column
total_na_count <- colSums(is.na(housing_data))
total_na_count
```

```
             Sale Date               Sale Price              sale_reason
                     0                        0                        0
       sale_instrument             sale_warning                 sitetype
                     0                    10564                        0
             addr_full                     zip5                  ctyname
                     0                        0                     6077
            postalctyn                      lon                      lat
                     0                        0                        0
      building_grade square_feet_total_living                 bedrooms
                     0                        0                        0
       bath_full_count          bath_half_count          bath_3qtr_count
                     0                        0                        0
            year_built           year_renovated           current_zoning
                     0                        0                        0
             sq_ft_lot                prop_type              present_use
                     0                        0                        0
```

```
## Naming the columns with NAs
columns_with_na <- names(total_na_count[total_na_count > 0])

## Display the names of columns with NAs
cat("Columns with NAs:", paste(columns_with_na, collapse = ", "), "\n")
```

```
Columns with NAs: sale_warning, ctyname
```

```
housing_data <- housing_data %>%
  select(-sale_warning, -ctyname)

## There were two columns that gave us a lot of NAs. Therefore, as their data would not be used
in comparing it to the Sale Price, deleting them cleans up the date more and doesn't degrade the
integrity of our data.
```

Outliers

```
## Assumptions: I believe that the total number of bedrooms could be zero, and that if it is zer
o, then I would think that would mean that it is a studio house. I believe that the lot size cou
ld also be zero and that it is a studio house as well.

## Identifying outliers would be good at this point, as we have eliminated the larger types of e
rrors.

## Let's find the outliers that could be in our data. I think sale date, sale price, square feet
total living, bedrooms, bath full count, bath half count, bath 3qtr count, sq ft lot.

# Identifying outliers for Sale Date
sale_date_outliers <- housing_data$`Sale Date` < as.Date("1980-01-01") | housing_data$`Sale Date
` > as.Date("2050-01-01")

# Identifying outliers for Sale Price
sale_price_outliers <- housing_data$`Sale Price` > 10000000 | housing_data$`Sale Price` < 0

# Identifying outliers for Square Feet Total Living
square_feet_total_living_outliers <- housing_data$square_feet_total_living > 15000 | housing_dat
a$square_feet_total_living < 0

## Identifying outliers for Bedrooms
bedrooms_outliers <- housing_data$bedrooms > 10 | housing_data$bedrooms < 0

## Identifying outliers for Bath Full Count
bath_full_count_outliers <- housing_data$bath_full_count > 10 | housing_data$bath_full_count < 0

## Identifying outliers for Bath Half Count
bath_half_count_outliers <- housing_data$bath_half_count > 10 | housing_data$bath_half_count < 0

## Identifying outliers for Bath 3Qtr Count
bath_3qtr_count_outliers <- housing_data$bath_3qtr_count > 10 | housing_data$bath_3qtr_count < 0

## Identifying outliers for Square Feet Lot
sq_ft_lot_outliers <- housing_data$sq_ft_lot > 1000000 | housing_data$sq_ft_lot < 0

## This I believe covers all the values that would be outliers. I don't want to show them, but t
o count them, so I am counting them here.

count_sale_date_outliers <- sum(sale_date_outliers)
count_sale_price_outliers <- sum(sale_price_outliers)
count_square_feet_total_living_outliers <- sum(square_feet_total_living_outliers)
count_bedrooms_outliers <- sum(bedrooms_outliers)
count_bath_full_count_outliers <- sum(bath_full_count_outliers)
count_bath_half_count_outliers <- sum(bath_half_count_outliers)
count_bath_3qtr_count_outliers <- sum(bath_3qtr_count_outliers)
count_sq_ft_lot_outliers <- sum(sq_ft_lot_outliers)
```

```
## Displaying the counts
cat("Number of Sale Date outliers:", count_sale_date_outliers, "\n")
```

```
Number of Sale Date outliers: 0
```

```
cat("Number of Sale Price outliers:", count_sale_price_outliers, "\n")
```

```
Number of Sale Price outliers: 0
```

```
cat("Number of Square Feet Total Living outliers:", count_square_feet_total_living_outliers,
"\n")
```

```
Number of Square Feet Total Living outliers: 0
```

```
cat("Number of Bedrooms outliers:", count_bedrooms_outliers, "\n")
```

```
Number of Bedrooms outliers: 1
```

```
cat("Number of Bath Full Count outliers:", count_bath_full_count_outliers, "\n")
```

```
Number of Bath Full Count outliers: 1
```

```
cat("Number of Bath Half Count outliers:", count_bath_half_count_outliers, "\n")
```

```
Number of Bath Half Count outliers: 0
```

```
cat("Number of Bath 3Qtr Count outliers:", count_bath_3qtr_count_outliers, "\n")
```

```
Number of Bath 3Qtr Count outliers: 0
```

```
cat("Number of Square Feet Lot outliers:", count_sq_ft_lot_outliers, "\n")
```

```
Number of Square Feet Lot outliers: 8
```

```
## Okay I changed my original amount for some of my outliers, as I felt there were too many outl
iers, and now I feel that I have a closer amount that could be considered outliers.

## Let's see the rows of the outliers.

outliers_sale_date <- subset(housing_data, sale_date_outliers)
outliers_sale_date
```

0 rows | 1-10 of 22 columns

```
outliers_sale_price <- subset(housing_data, sale_price_outliers)
outliers_sale_price
```

0 rows | 1-10 of 22 columns

```
outliers_square_feet_total_living <- subset(housing_data, square_feet_total_living_outliers)
outliers_square_feet_total_living
```

0 rows | 1-10 of 22 columns

```
outliers_bedrooms <- subset(housing_data, bedrooms_outliers)
outliers_bedrooms
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sitetype <chr>	addr_full <chr>	zip5 <dbl>	po <c
2007-12-11	1825000	1	3	R1	9216 132ND AVE NE	98052	RI

1 row | 1-8 of 22 columns

```
outliers_bath_full_count <- subset(housing_data, bath_full_count_outliers)
outliers_bath_full_count
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sitetype <chr>	addr_full <chr>	zip5 <dbl>	pos <ch
2006-03-28	270000	1	3	R1	5806 249TH CT NE	98053	REI

1 row | 1-8 of 22 columns

```
outliers_bath_half_count <- subset(housing_data, bath_half_count_outliers)
outliers_bath_half_count
```

0 rows | 1-10 of 22 columns

```
outliers_bath_3qtr_count <- subset(housing_data, bath_3qtr_count_outliers)
outliers_bath_3qtr_count
```

0 rows | 1-10 of 22 columns

```
outliers_sq_ft_lot <- subset(housing_data, sq_ft_lot_outliers)
outliers_sq_ft_lot
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sitetype <chr>	addr_full <chr>
2007-12-13	379950	1	3	R1	6415 196TH AVE NE
2010-03-02	4400000	1	3	R1	12053 154TH PL NE
2010-07-06	698	1	26	R1	19805 NE NOVELTY HILL RD
2010-07-06	698	1	26	R1	19805 NE NOVELTY HILL RD
2013-06-13	14000	1	26	R1	20210 NE 85TH ST
2016-03-29	2165000	1	3	R1	11207 248TH AVE NE
2016-03-29	2165000	1	3	R1	11207 248TH AVE NE
2016-11-28	2050000	1	3	R1	6415 196TH AVE NE

8 rows | 1-7 of 22 columns

```
## Out of the 8 things that I identified that could have outliers only 3 said there could be out
liers. After looking at the 3 groups, only one appears to be an outlier and that is outliers_bat
h_full_count, I got a value of 23 bathrooms for a 4 bedroom house.

## Now I want to replace the 23 with the average whole number of bathrooms for 4 bedroom houses.

## I want to filter the data for 4-bedroom houses
filtered_data <- housing_data %>%
  filter(bedrooms == 4)

## Then calculate the average number of bathrooms using bath_full_count
average_bathrooms <- round(mean(filtered_data$bath_full_count))

## The replace the value of 23 in the "bath_full_count" column with the rounded whole number val
ue
housing_data <- housing_data %>%
  mutate(bath_full_count = ifelse(bath_full_count == 23, average_bathrooms, bath_full_count))

## Now, when I check for outliers, the values in the entire dataset are updated
outliers_bath_full_count <- subset(housing_data, bath_full_count_outliers)
outliers_bath_full_count
```

Sale Date <dttm>	Sale Price <dbl>	sale_reason <dbl>	sale_instrument <dbl>	sitetype <chr>	addr_full <chr>	zip5 <dbl>	pos <ch
2006-03-28	270000	1	3	R1	5806 249TH CT NE	98053	REI

1 row | 1-8 of 22 columns

```
## Now that the data has been cleaned up, it is now time to rename the columns.

housing_data <- housing_data %>%
  rename(
  'Sale Reason Code' = sale_reason,
'Sale Instrument Code' = sale_instrument,
'Site Type' = sitetype,
Address = addr_full,
Zip = zip5,
'City Name' = postalctyn,
Longitude = lon,
Latitude = lat,
Grade = building_grade,
'Sq ft' = square_feet_total_living,
Bedrooms = bedrooms,
'Full Baths' = bath_full_count,
'Half Baths' = bath_half_count,
'3 Quarters Bath' = bath_3qtr_count,
'Year Built' = year_built,
'Year Renovated' = year_renovated,
'Zoning' = current_zoning,
'Lot Size (sq ft)' = sq_ft_lot,
'Property Type' = prop_type,
'Present Use Code' = present_use
)
head(housing_data)
```

Sale Date <dttm>	Sale Price <dbl>	Sale Reason Code <dbl>	Sale Instrument Code <dbl>	Site Type <chr>	Address <chr>
2006-01-03	698000	1	3	R1	17021 NE 113TH
2006-01-03	649990	1	3	R1	11927 178TH PL I
2006-01-03	572500	1	3	R1	13315 174TH AVE
2006-01-03	420000	1	3	R1	3303 178TH AVE
2006-01-03	369900	1	3	R1	16126 NE 108TH
2006-01-03	184667	1	15	R1	8101 229TH DR N

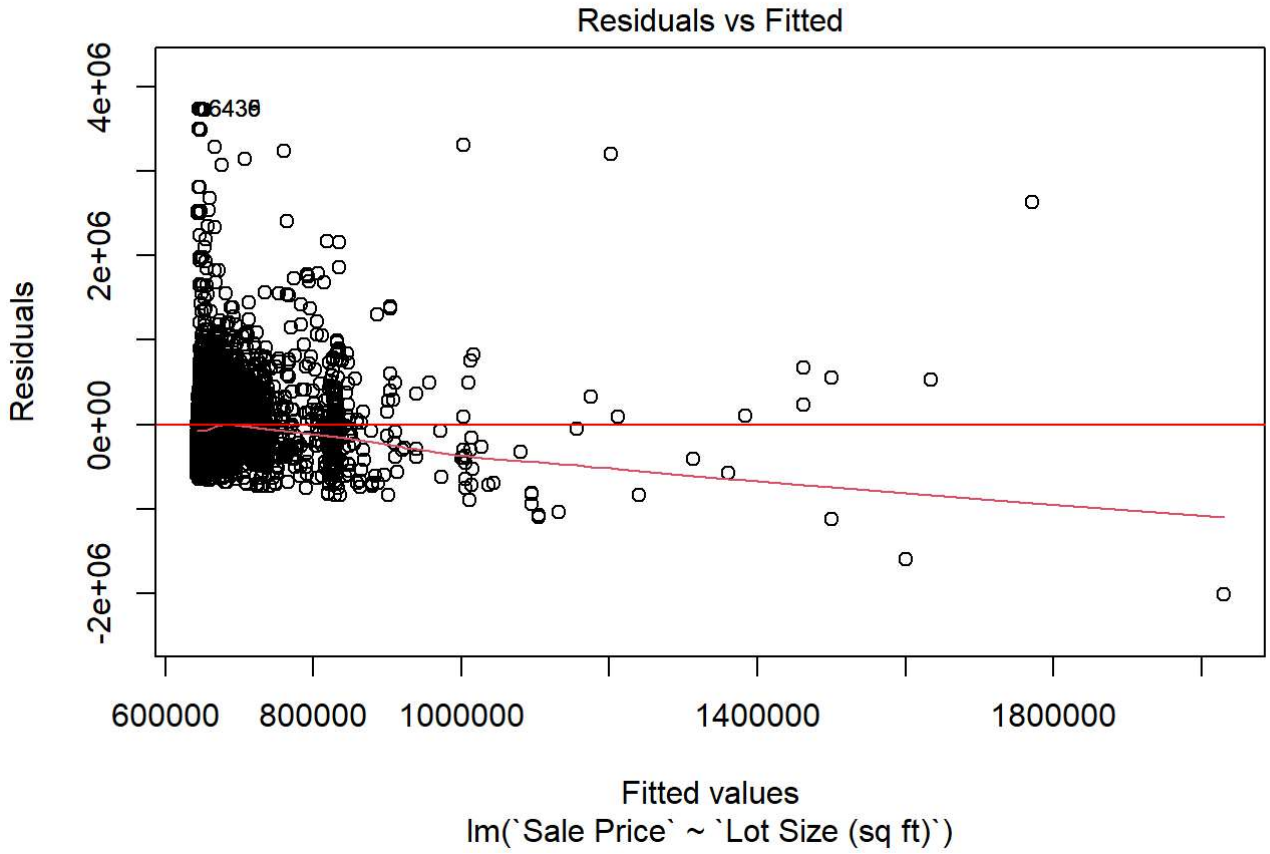
6 rows | 1-7 of 22 columns

```
## b. Complete the following:
## 1. Explain any transformations or modifications you made to the dataset.

## What I did was:
## A. Found duplicate rows, and made them unique.
## B. Found columns with NAs, and deleted them.
## C. Found outliers, and replaced them.
## D. Renamed columns.
```

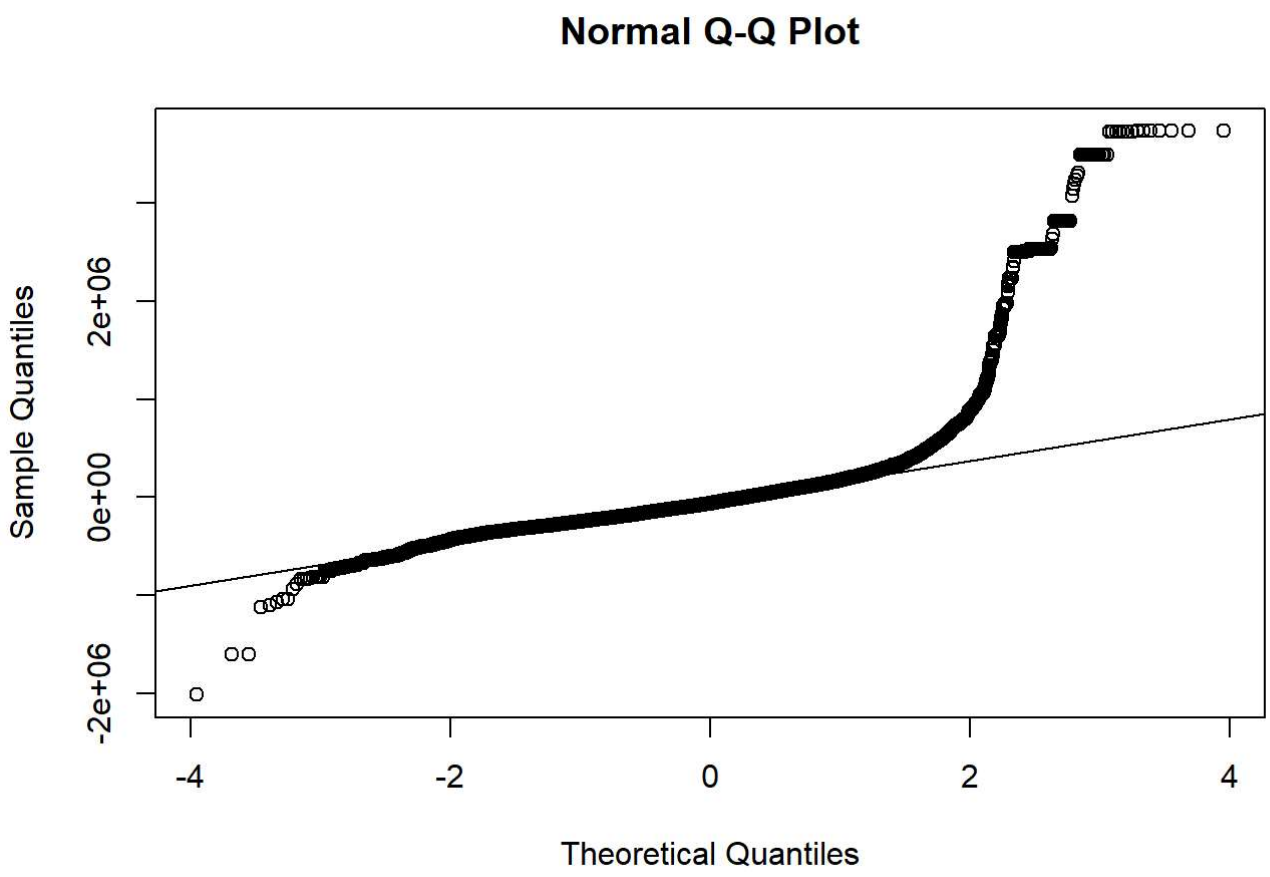
```
## 2. Create a linear regression model where "sq_ft_lot" predicts Sale Price.

## Fit a linear regression model
lm_model <- lm(`Sale Price` ~ `Lot Size (sq ft)`, data = housing_data)

## Summarize the model
summary(lm_model)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ `Lot Size (sq ft)`, data = housing_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2015913  -194880   -63332    91528  3735071
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        6.419e+05  3.801e+03  168.87   <2e-16 ***
## `Lot Size (sq ft)` 8.509e-01  6.218e-02   13.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 401500 on 12859 degrees of freedom
## Multiple R-squared:  0.01435,    Adjusted R-squared:  0.01428
## F-statistic: 187.2 on 1 and 12859 DF,  p-value: < 2.2e-16
```

```
## 3. Get a summary of your first model and explain your results (i.e., R2, adj. R2, etc.)

## The residuals min and max show how the sale price can deviate from the actual values. The qua
rtile in 1Q that 25% of the residuals fall below that value, or an underestimate of the sale pri
ce by $194,880 is happening. The median is predicting an underestimation of the sale price by $6
3,332. The quartile in 3Q is suggesting that most predicitions underestimate the sale price by
$91,528.

## The coefficient intercept means that with the low p-value the intercept is highly statistical
ly significant meaning it's not due to random chance. The coefficient for the Lot Size (sq ft) m
eans that for every additional sqaure foot in lot size, the predicted 'Sale Price' increases app
roximately $0.851. The low p-value indicates that the coefficient is highly statistically signif
icant.

## The R-Squared value suggests that this model only explains a small portion of the variablilit
y in 'Sale Price' and the adjusted R-Squared value accounts the number of predictors indicating
that the model may not be a good fit for the data.


## 4.Get the residuals of your model (you can use ‘resid’ or ‘residuals’ functions) and plot the
m. What the does the plot tell you about your predictions?

## Get the residuals from the linear regression model
residuals <- resid(lm_model)

## Create a residual plot
plot(lm_model, which = 1)
abline(h = 0, col = "red")  # Add a reference line at y = 0
```




```
## Since most are clustered around the bottom left corner, this indicates a lack of fit in my li
near regression model. Because there are two lines it futher supports the idea of non-linearity
in the regression model.


## 5 Use a qq plot to observe your residuals. Do your residuals meet the normality assumption?

qqnorm(residuals(lm_model))
qqline(residuals(lm_model))
```



Because the it deviates from the stright line, it indicates a departure of nomality in the residuals. Also, we see that there are outliers because the values are away from the line. The points are curving suggesting skewness. Because of these things I can say that they do not meet the nomality assuptions of linear regression.

6. Now, create a linear regression model that uses multiple predictor variables to predict Sale Price (feel free to derive new predictors from existing ones). Explain why you think each of these variables may add explanatory value to the model.

I think that people look at how many square fee they are getting whenever they are purchasing a house as well as how many bedrooms and full baths. I feel that anything other than a full bath is considered either a perk or a shurk. Perk because it's great to have or a shurk because they did not want to build another full bath.

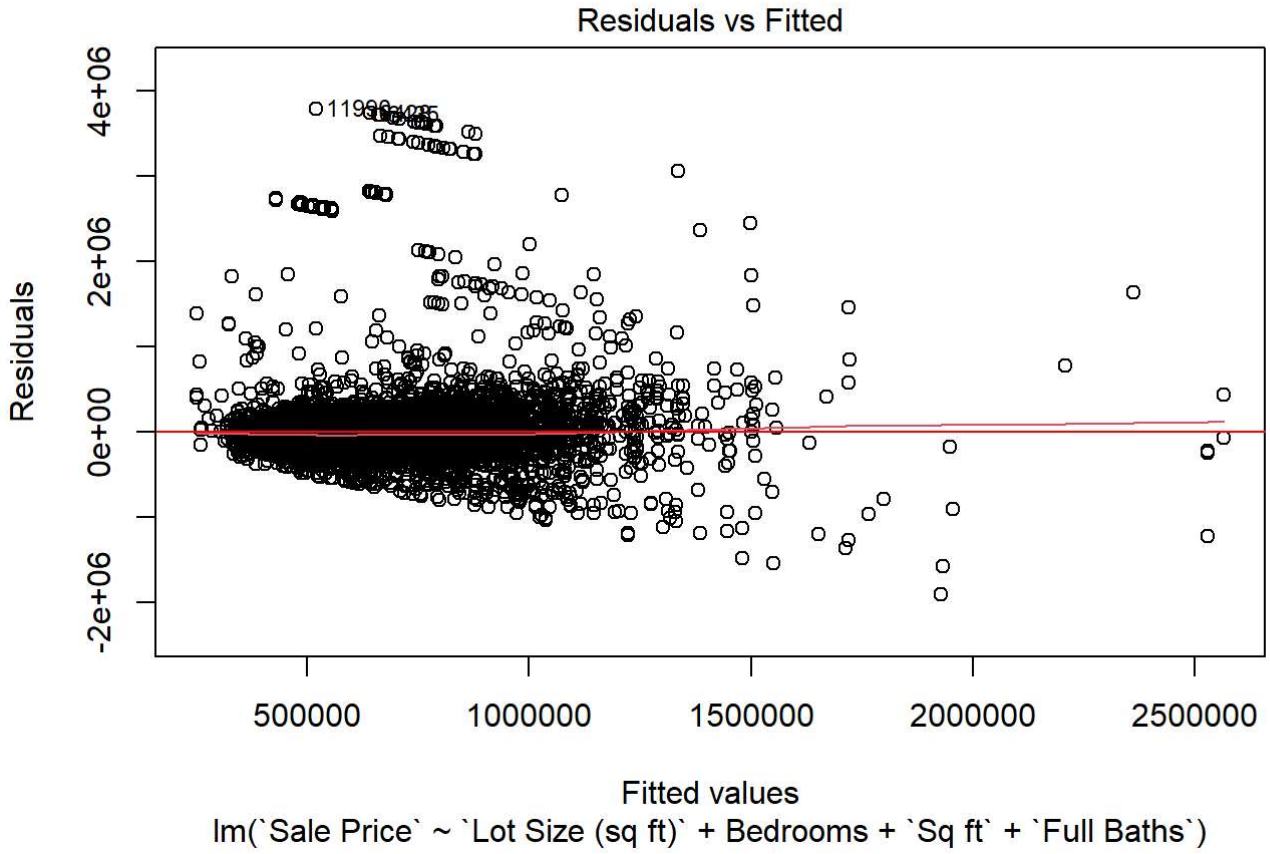
```
new_lm_model <- lm(`Sale Price` ~ `Lot Size (sq ft)` + Bedrooms + `Sq ft` + `Full Baths`, data = housing_data)
```

7. Get a summary of your next model and explain your results.

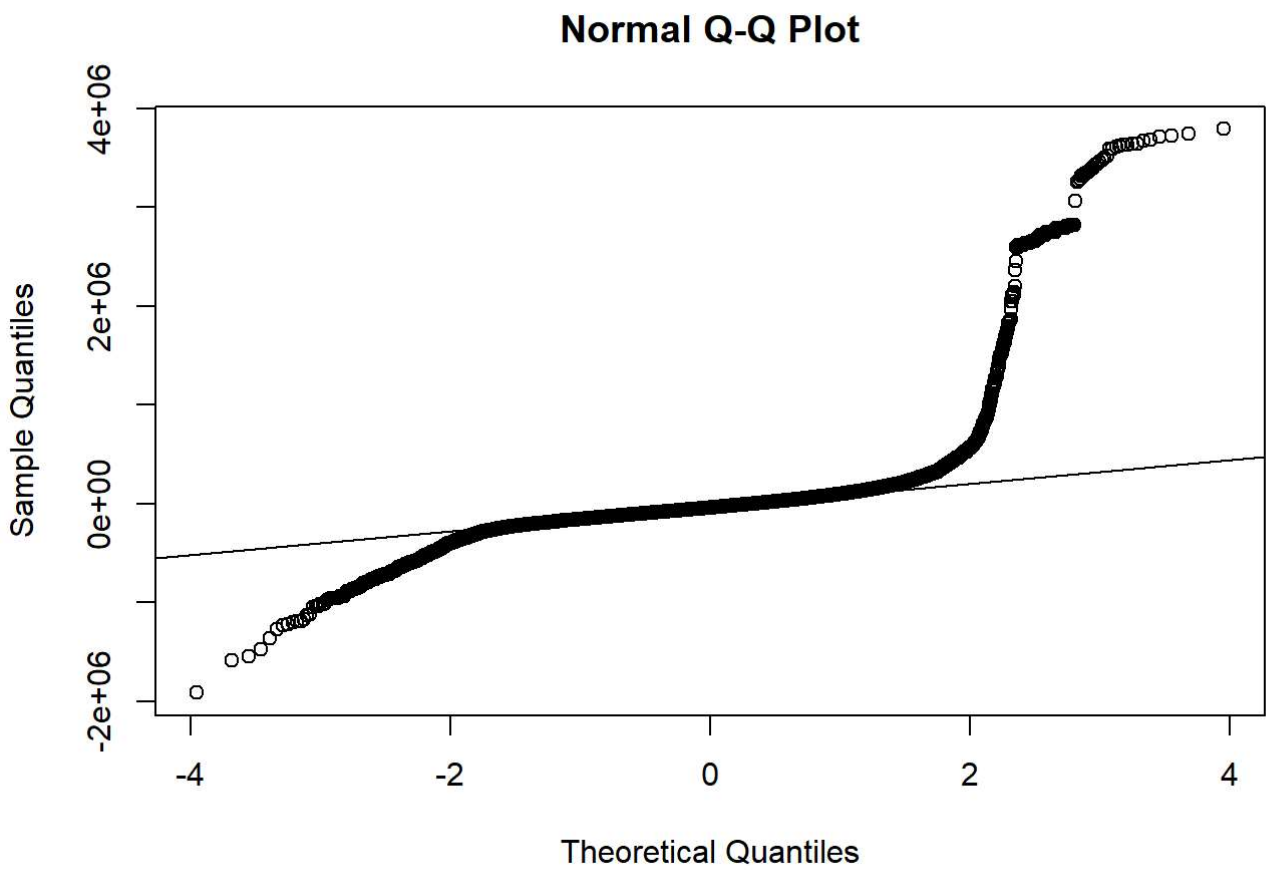
```
summary(new_lm_model)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ `Lot Size (sq ft)` + Bedrooms + `Sq ft` +
##     `Full Baths`, data = housing_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1914396  -117866   -40718    44469  3790302
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.935e+05  1.416e+04  13.671  < 2e-16 ***
## `Lot Size (sq ft)`  1.148e-01  5.775e-02   1.987    0.047 *
## Bedrooms           -2.449e+04  4.443e+03  -5.513 3.59e-08 ***
## `Sq ft`             1.786e+02  4.550e+00  39.262  < 2e-16 ***
## `Full Baths`        5.355e+04  6.030e+03   8.880  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 358700 on 12856 degrees of freedom
## Multiple R-squared:  0.2135, Adjusted R-squared:  0.2132
## F-statistic: 872.3 on 4 and 12856 DF,  p-value: < 2.2e-16
```

```
## The model suggest that the number of bedrooms, square footage, and the number of bathrooms ha
ve a significant impact on the sale price. The 'Lot Size (sq ft) variable has a p-value close to
0.05 suggesting that it might be marginally significant.

## The adjusted R-squared indicates that the model explains about 21.32% of the variability in s
ale prices.

## 8. Get the residuals of your second model (you can use ‘resid’ or ‘residuals’ functions) and
plot them. What the does the plot tell you about your predictions?

## Get the residuals from the linear regression model
residuals <- resid(new_lm_model)

## Create a residual plot
plot(new_lm_model, which = 1)
abline(h = 0, col = "red")  # Add a reference line at y = 0
```




```
## Because most of the residual are on the left side, this could mean that the model will better predict sale prices between the $500,000 to $1,500,000 range but not so well for prices above $1,500,000. Because there are some outliers above the rest, the model could overestimate certian types of properties.

## 9. Use a qq plot to observe your residuals.

qqnorm(residuals(new_lm_model))
qqline(residuals(new_lm_model))
```




```
## Do your residuals meet the normality assumption?
## Because the it deviates from the stright line, it indicates a departure of nomality in the re
siduals. Also, we see that there are outliers because the values are away from the line. The poi
nts are curving suggesting skewness. Because of these things I can say that they do not meet the
nomality assuptions of linear regression.
```

```
## Compare the results (i.e., R2, adj R2, etc) between your first and second model. Does your ne
w model show an improvement over the first?

## Yes the second model is an improvement from the first one, as it explains a larger portion of
the variance in 'Sale Price', making it a better predictor of property sale prices based on the
selected predictor variables.

## To confirm a ‘significant’ improvement between the second and first model, use ANOVA to compa
re them. What are the results?

## Perform ANOVA to compare the two models
anova_results <- anova(lm_model, new_lm_model)
anova_results
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	12859	2.073218e+15	*NA*	*NA*	*NA*	*NA*

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2	12856	1.654408e+15	3	4.188102e+14	1084.824	0

2 rows

```
## There is a significant improvement from the first model to the second. The p-value is less th
an the chosen significance level of 0.05, which is denoted as "< 2.2e-16 ***." This suggests tha
t the second model provides a significant improvement in explaining the variation in the Sale Pr
ice compared to the first model.

## 11. After observing both models (specifically, residual normality), provide your thoughts con
cerning whether the model is biased or not.

## The plots of the residuals from both models appear to show bias. The residuals do not seem to
follow a normal distribution and show some patterns in their distribution. Because the plots wer
e showing non-random patterns suggests the models are missing important predictor variables and
this can lead to bias in the predictions. The full picture may not be seen in the data used. May
need more data and other checks maybe needed to see if the model needs to be refined to reduce b
ias.
```

```
## 12. Another important aspect of regression tasks is determining the accuracy of your predicti
ons. For this section, we will look at root mean square error (RMSE), a common accuracy metric f
or regression models.

## 1. Install the 'Metrics' package in R Studio
library(Metrics)

## 2. Using the first model, we will make predictions on the dataset using the predict function.
An example would look like this (will vary for you based on variable names):

## 1. 'preds <- predict(object = modelName, newdata = dataset)'

preds <- predict(lm_model, newdata = housing_data)

## 2. Use the 'rmse' function to get RMSE for the model
rmse_value <- rmse(housing_data $'Sale Price', preds)
rmse_value
```

```
## [1] 401499.6
```

```
## 3. What is the RMSE for the first model?

## The value was $401449.6

## 4. Perform the same task for the second model. Provide the RMSE for the second model.

## 1. 'preds <- predict(object = modelName, newdata = dataset)'

new_preds <- predict(new_lm_model, newdata = housing_data)

## 2. Use the 'rmse' function to get RMSE for the model
new_rmse_value <- rmse(housing_data $'Sale Price', new_preds)
new_rmse_value
```

```
## [1] 358660.8
```

```
## 5. Did the second model's RMSE improve upon the first model? By how much?

improvement <- rmse_value - new_rmse_value
improvement
```

```
## [1] 42838.8
```

```
## It did improve by a reduction of $42,838.8, which means the second model is better at making predictions for housing sale prices. The lower RMSE values suggest that the model's predictions are closer to the actual sale prices.
```