```
library(leaps)
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(ggplot2)
library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Warning: package 'survival' was built under R version 4.0.5

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(ggcorrplot)


# Importing data
Housing <- read.csv("Desktop/Uni/Statistical Learning & Regression
Models/kc_house_data.csv")
```

```r
# Inspecting data
str(Housing)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000"
"20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890
...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470
6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890
...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003
...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198
98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390
...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570
...
```

```r
# Convert to factor
catPred           <-
c('bedrooms','bathrooms','floors','waterfront','view','condition','grade')
contPred          <-
c('sqft_living','sqft_lot','sqft_above','sqft_basement','sqft_living15','sqft
_lot15','yr_built','yr_renovated','lat','long')
Housing[,catPred] <- lapply(Housing[,catPred], as.factor)

str(Housing)
```

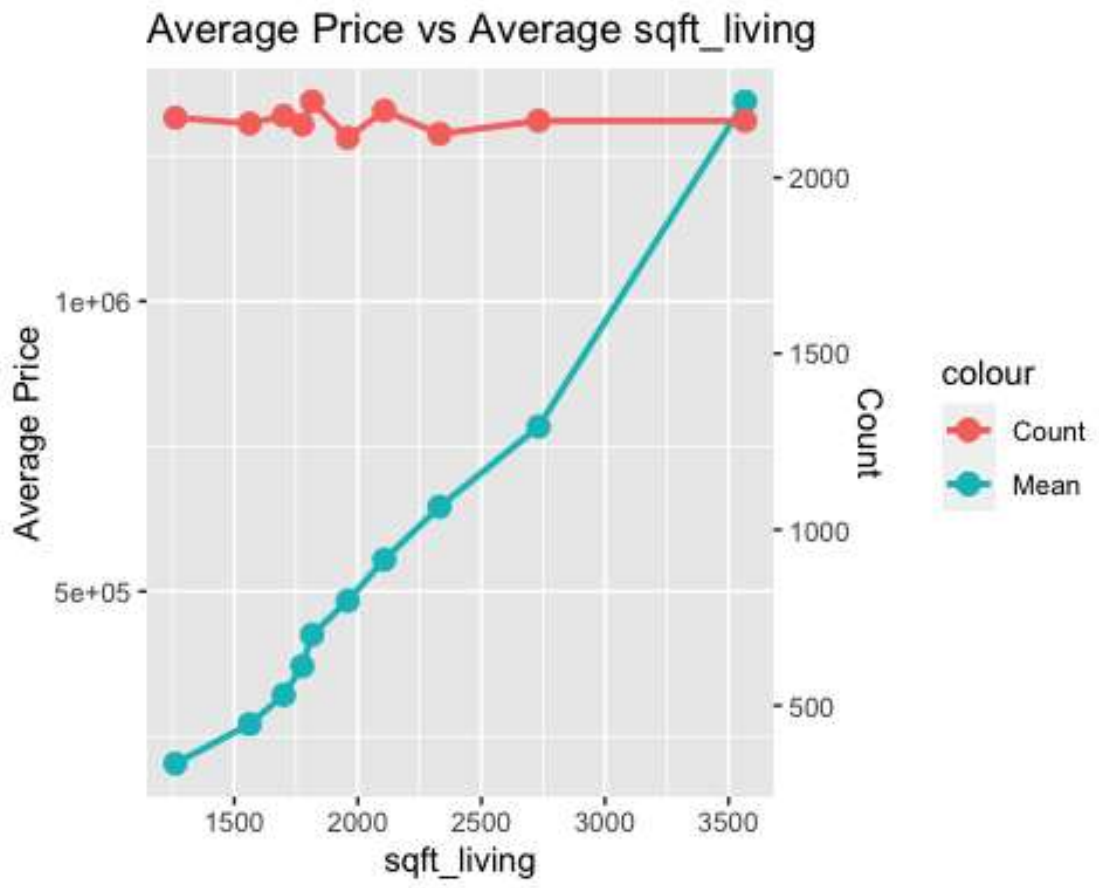
```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000"
"20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : Factor w/ 13 levels "0","1","2","3",..: 4 4 3 5 4 5 4 4
```

```
4 4 ...
##  $ bathrooms    : Factor w/ 30 levels "0","0.5","0.75",..: 4 9 4 12 8 18 9
6 4 10 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890
...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470
6560 ...
##  $ floors       : Factor w/ 6 levels "1","1.5","2",..: 1 3 1 1 1 1 3 1 1 3
...
##  $ waterfront   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ view         : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1
1 ...
##  $ condition    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 5 3 3 3 3 3
3 ...
##  $ grade        : Factor w/ 12 levels "1","3","4","5",..: 6 6 5 6 7 10 6 6
6 6 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890
...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003
...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198
98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390
...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570
...
```

```
# Note: It's likely that lat, long, yr_built, and yr_renovated wouldn't be
modeled as numeric, but treating as numeric for plotting analysis


###----- Plot data to assess one-way trends and possible factor groupings  --
---###

#--- Continuous Predictors ---#
for (i in 1:length(contPred)) {
  cont              <- contPred[i]
  contReduced       <- Housing[,c('price',cont)]
  contReduced$group <- as.numeric(cut2(contReduced$price, g=10))

  print(cont)

  grouped <- contReduced %>%
             group_by(group) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              MeanX = mean(.data[[cont]], na.rm=TRUE),
```

```
                                Count = n())

  scl <- max(grouped$MeanY)/max(grouped$Count)

  print(ggplot(grouped, aes(x = MeanX)) +
                geom_line(aes(y = MeanY, colour = "Mean"), lwd = 1) +
                geom_point(aes(y = MeanY, colour = "Mean"), lwd = 3) +
                geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1) +
                geom_point(aes(y = Count*scl, colour = "Count"), lwd = 3) +
                scale_y_continuous(name = "Average Price", sec.axis =
sec_axis(~./scl, name = "Count")) +
                labs(x = cont) +
                ggtitle(paste('Average Price vs Average', cont)))

}
```

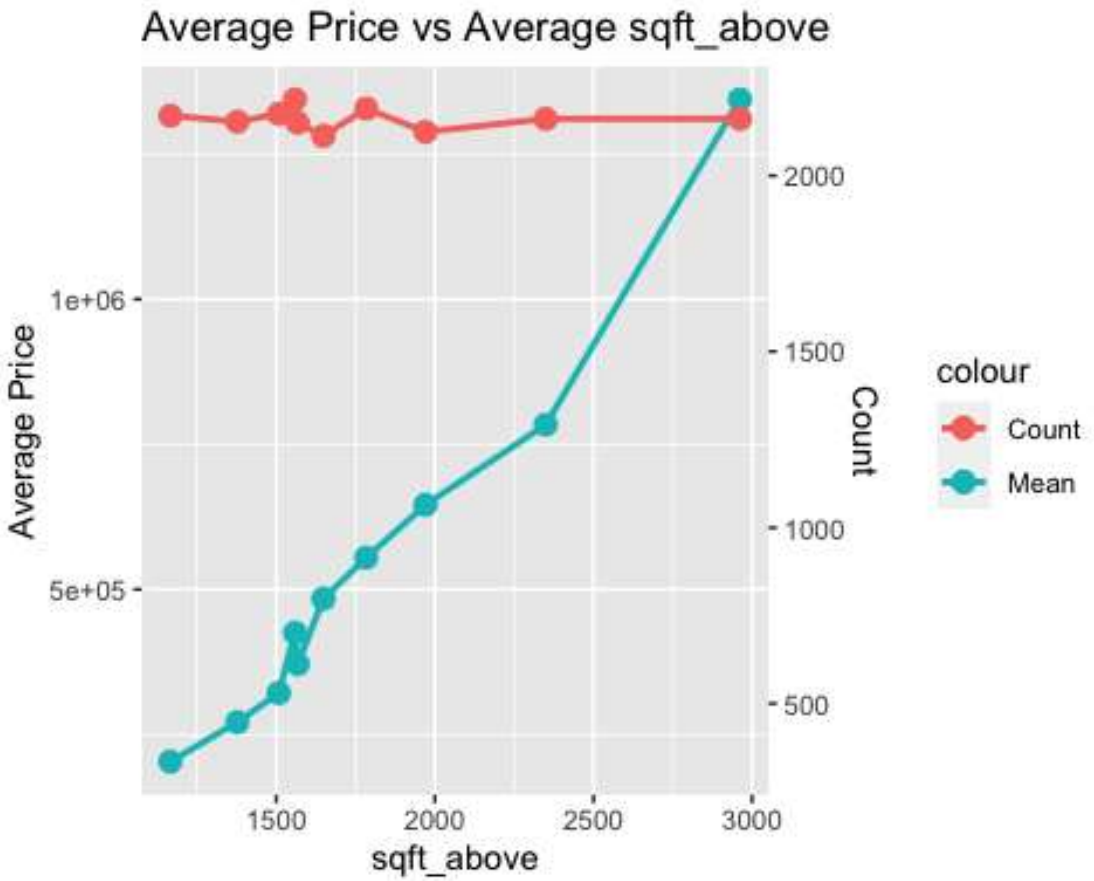
```
## [1] "sqft_living"
```



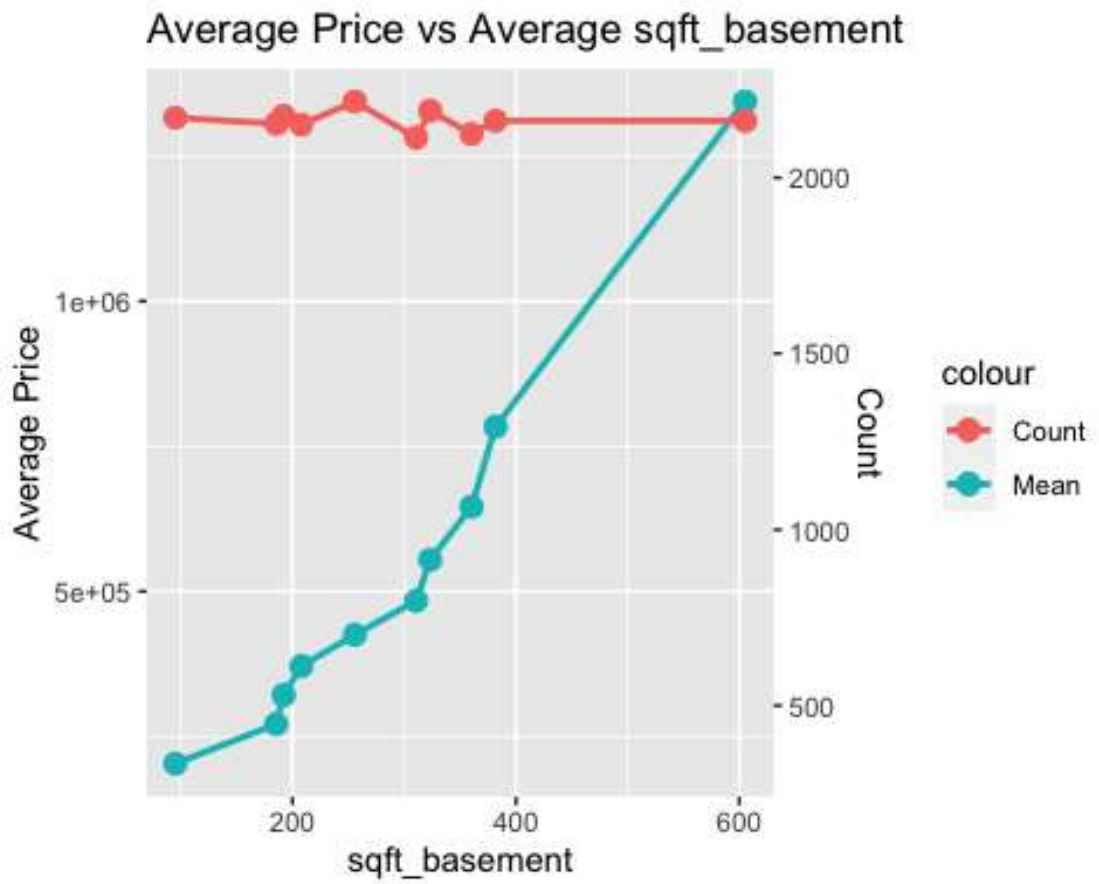

```
## [1] "sqft_lot"
```


Average Price vs Average sqft_lot
Average Price
1e+06
5e+05
10000
15000
20000
sqft_lot
2000
1500
1000
500
Count
colour
Count
Mean

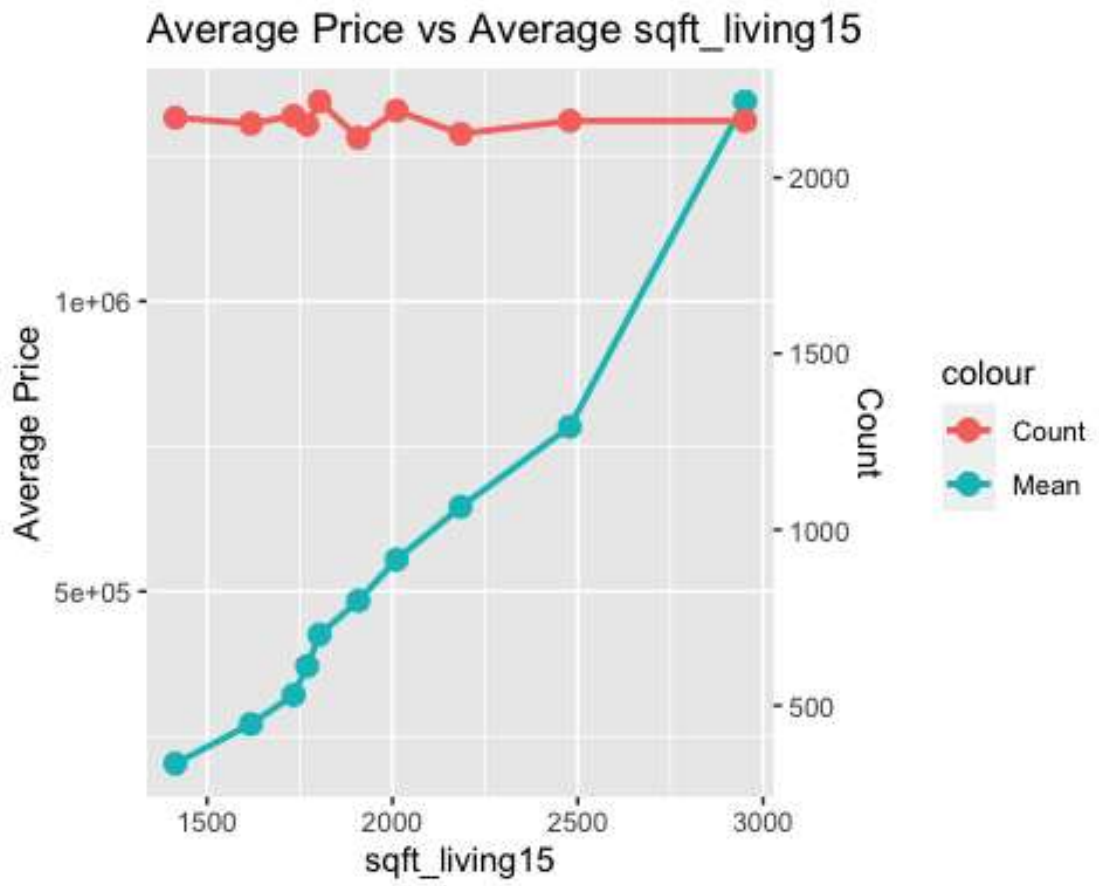
```
## [1] "sqft_above"
```




```
## [1] "sqft_basement"
```



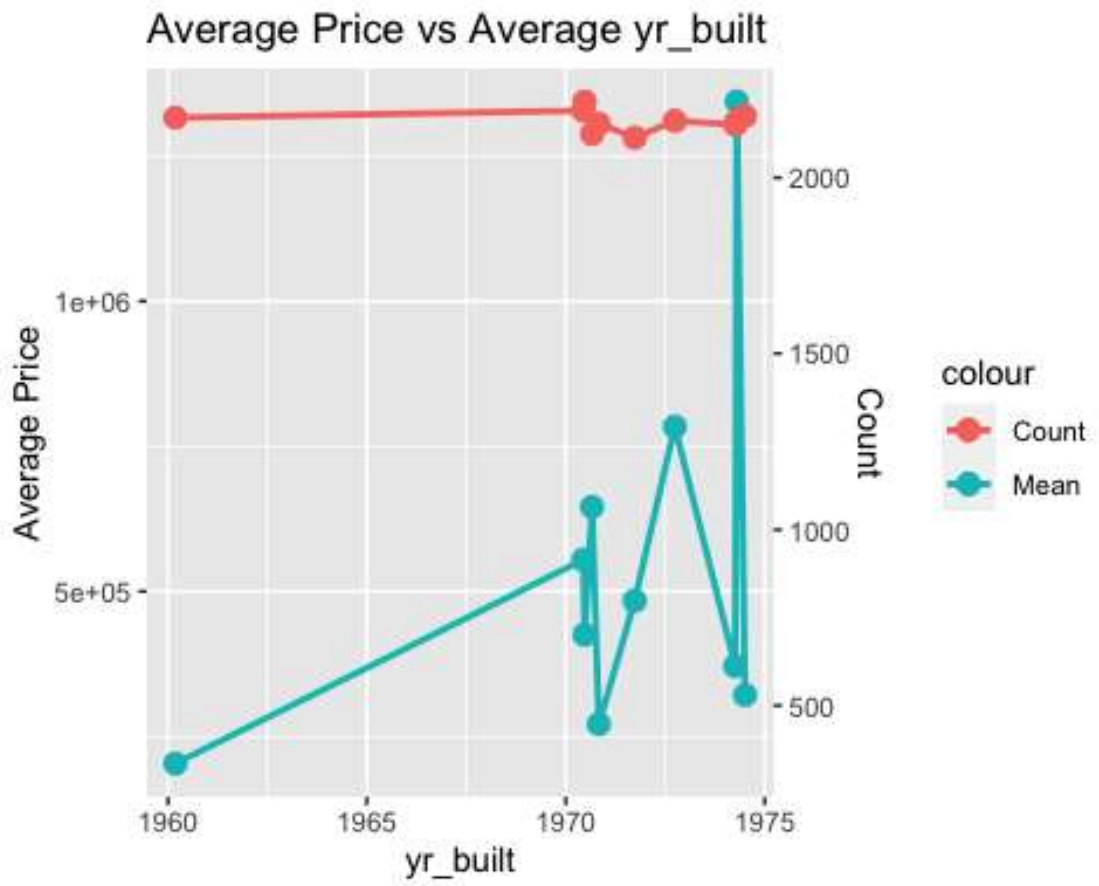

```
## [1] "sqft_living15"
```


Average Price vs Average sqft_living15
Average Price
1e+06
5e+05
2000
1500
1000
500
Count
1500
2000
2500
3000
sqft_living15
colour
Count
Mean

```
## [1] "sqft_lot15"
```


Average Price vs Average sqft_lot15
Average Price
1e+06
5e+05
2000
1500
1000
500
Count
10000
12000
14000
16000
18000
sqft_lot15
colour
Count
Mean

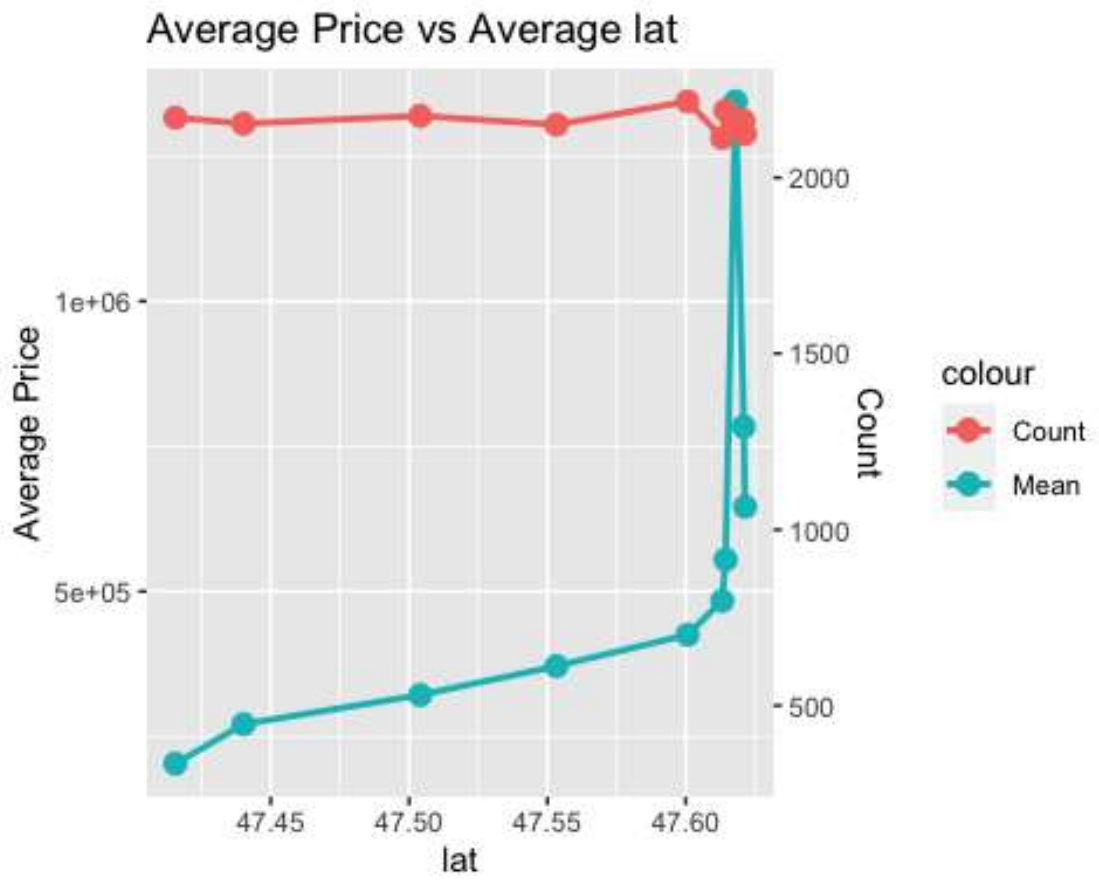
```
## [1] "yr_built"
```



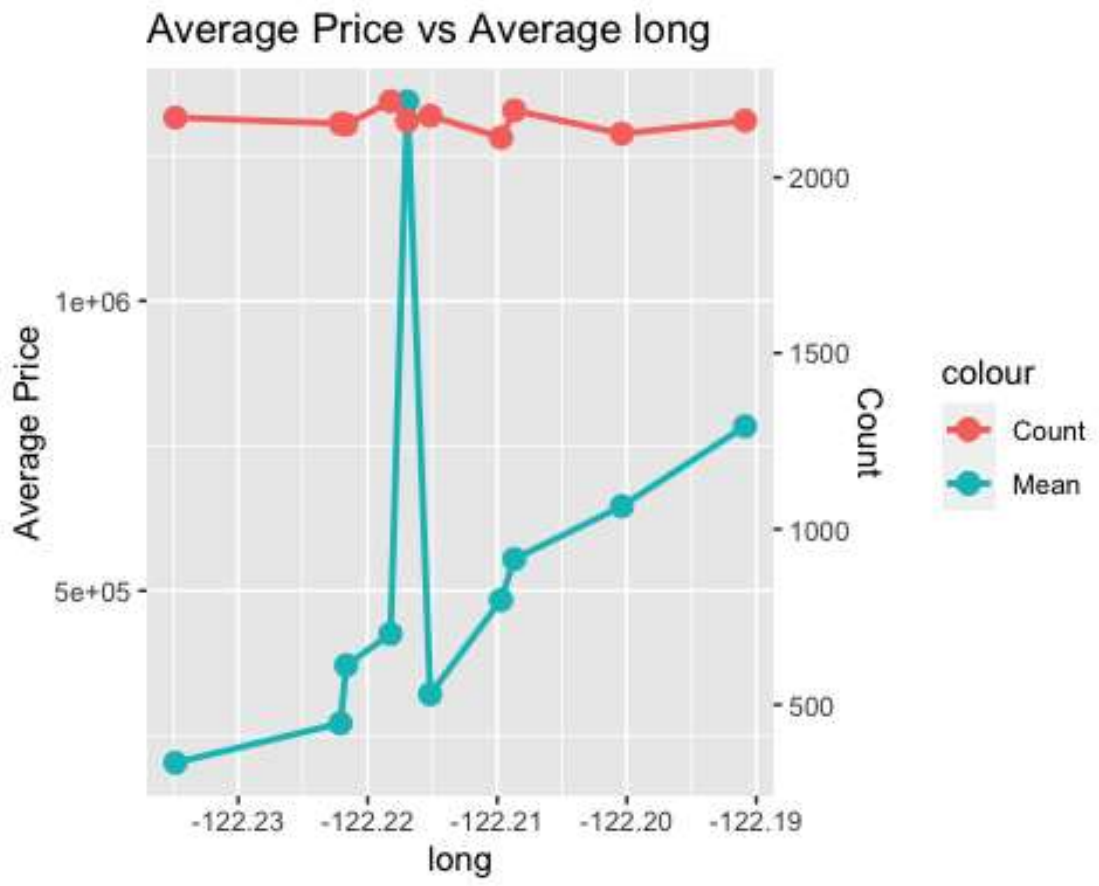

```
## [1] "yr_renovated"
```


Average Price vs Average yr_renovated
Average Price
1e+06
5e+05
Count
2000
1500
1000
500
50
100
150
200
yr_renovated
colour
Count
Mean

```
## [1] "lat"
```

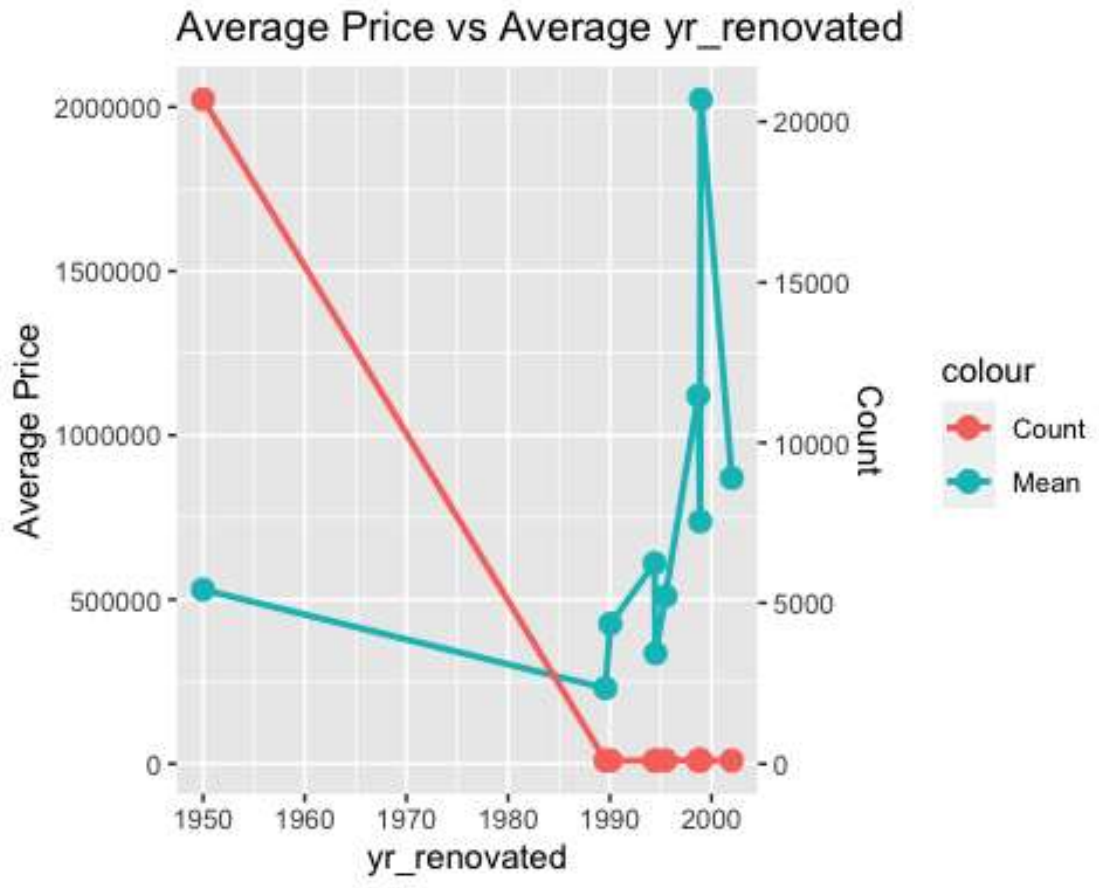

Average Price vs Average lat
Average Price
1e+06
5e+05
47.45
47.50
47.55
47.60
lat
2000
1500
1000
500
Count
colour
Count
Mean

```
## [1] "long"
```

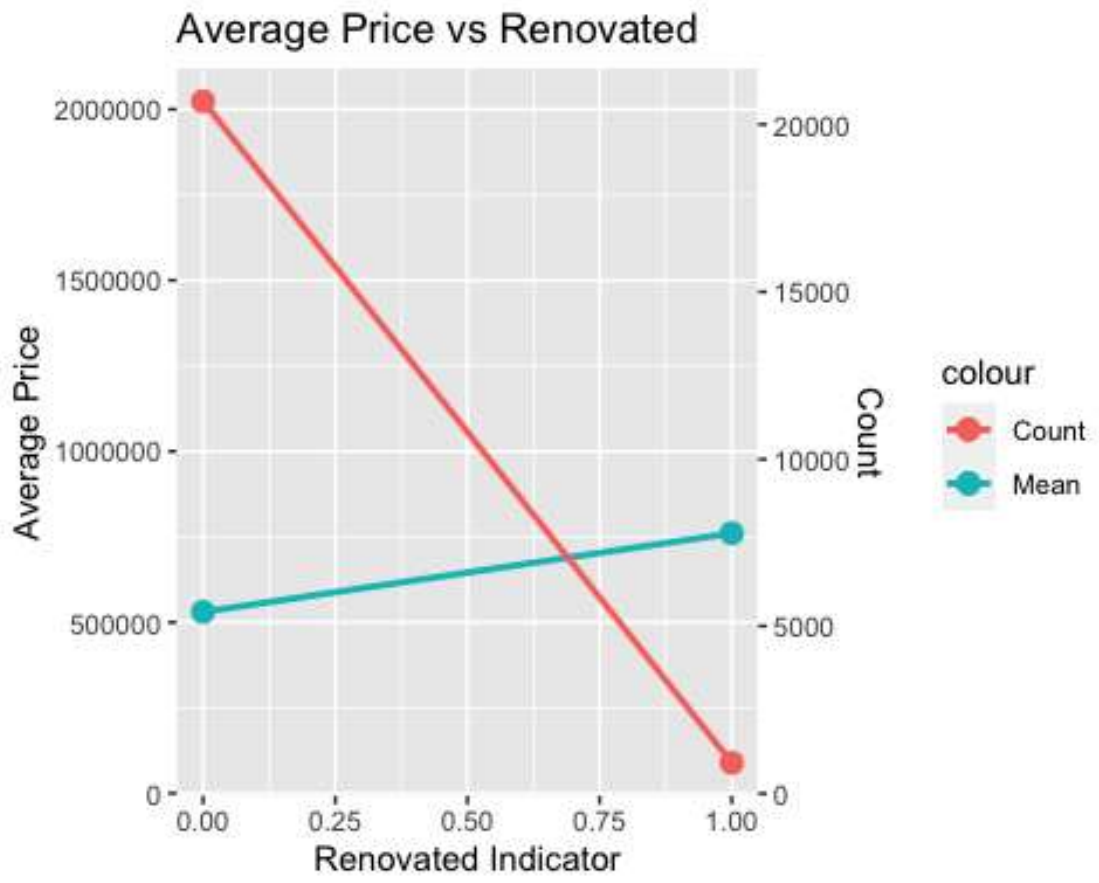



```
### Observations ###

#-- Decent trends on a one-way basis for sqft_living, sqft_log, sqft_above,
sqft_basement, sqft_living, and sqft_lot15
#-- No obvious trend for yr_built, could potentially create indicator
variable at peak
#-- For yr_renovated and sqft_basement, need to do again but exclude 0's into
own group
#-- Potential indicator variable for lat and long variables




##- Looking into yr_built -##
yrBuilt       <- Housing[,c('price','yr_built')]
yrBuilt$group <- as.numeric(cut2(yrBuilt$price, g=10))

grouped <- yrBuilt %>%
             group_by(group) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              MeanX = mean(yr_built, na.rm=TRUE),
```

```
                                Count = n())

# Note: Year Built ranges from 1900 to 2015, when grouped evenly and ordered
by price, the average yr_built for each group ranges from 1960 to 1974, this
indicates that there is
#       no obvious trend between the year built and price




##- Looking into yr_renovated -##
yrReno          <- Housing[,c('price','yr_renovated')]
yrReno_0        <- yrReno[yrReno$yr_renovated == 0,]
yrReno_1        <- yrReno[yrReno$yr_renovated != 0,]
yrReno_1$group <- as.numeric(cut2(yrReno_1$price, g=9))+1
yrReno_0$group <- 1

yrReno <- rbind(yrReno_0, yrReno_1)

grouped <- yrReno %>%
           group_by(group) %>%
           dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                            MeanX = mean(yr_renovated, na.rm=TRUE),
                            Count = n())

# Adjusting 0 as 1950, just so x-axis looks better
grouped[1,"MeanX"] <- 1950

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = MeanX)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 3) +
       geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1) +
       geom_point(aes(y = Count*scl, colour = "Count"), lwd = 3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'yr_renovated') +
       ggtitle(paste('Average Price vs Average yr_renovated'))
```

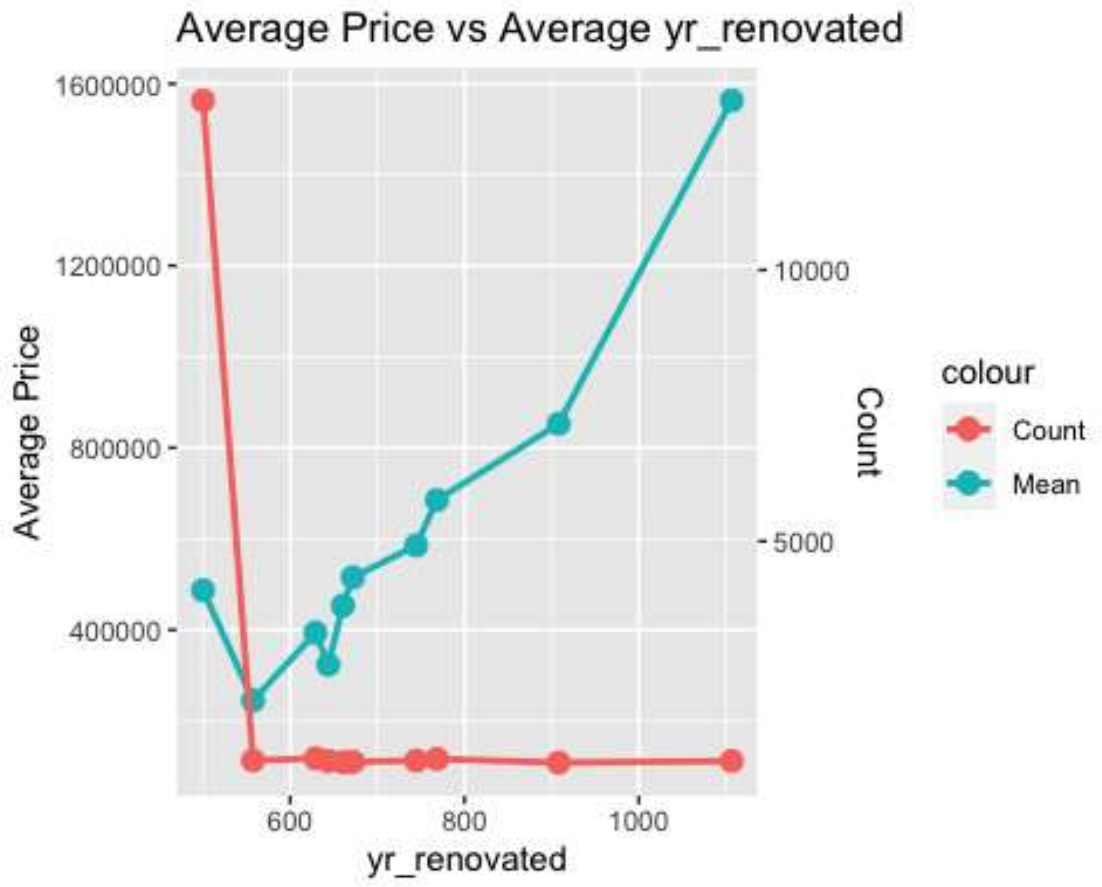



```
## Very small count for homes that have been renovated, trying binary for
either renovated or not
yrReno         <- Housing[,c('price','yr_renovated')]
yrReno$group   <- ifelse(yrReno$yr_renovated == 0, 0, 1)

grouped <- yrReno %>%
           group_by(group) %>%
           dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                            MeanX = mean(yr_renovated, na.rm=TRUE),
                            Count = n())

ggplot(grouped, aes(x = group)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 3) +
       geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1) +
       geom_point(aes(y = Count*scl, colour = "Count"), lwd = 3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Renovated Indicator') +
       ggtitle(paste('Average Price vs Renovated'))
```



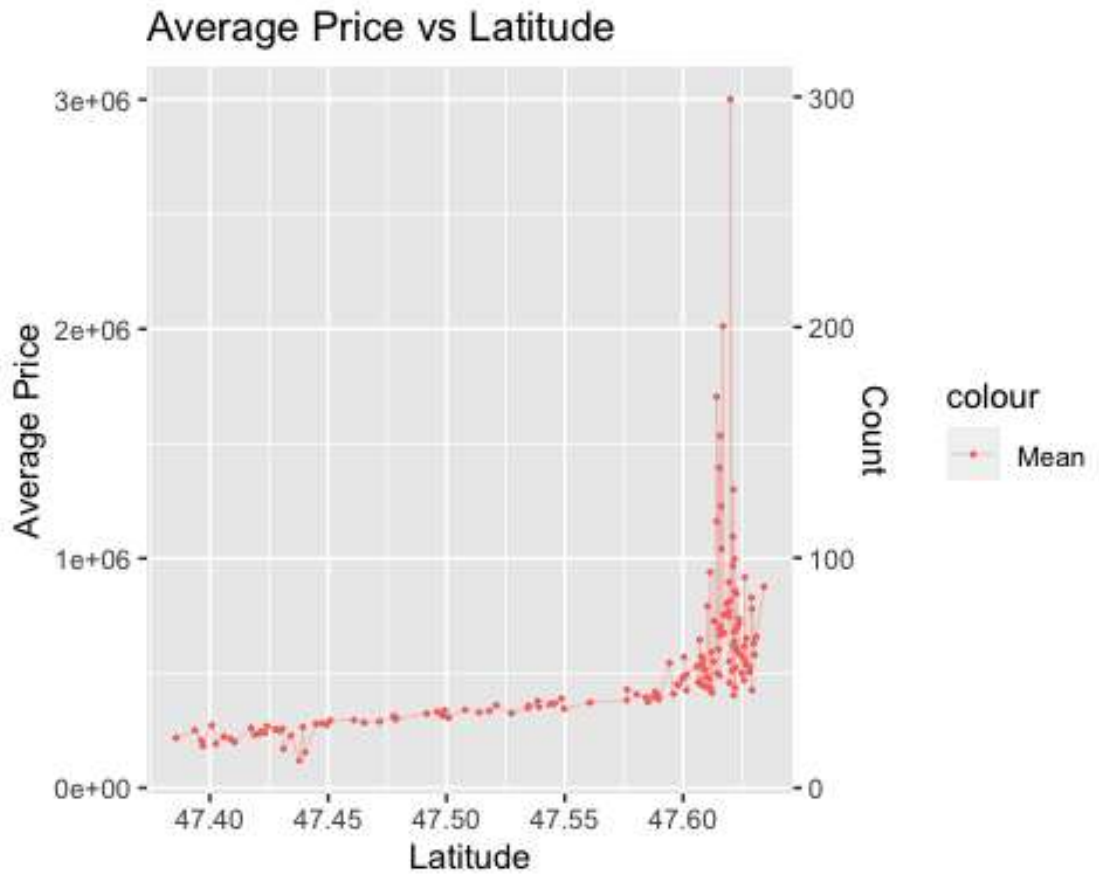

```
## Big difference in count, but a binary predictor could be beneficial
Housing$renovated <- ifelse(Housing$yr_renovated == 0, 0, 1)




##- Looking into sqft_basement -##
sqftBase         <- Housing[,c('price','sqft_basement')]
sqftBase_0       <- sqftBase[sqftBase$sqft_basement == 0,]
sqftBase_1       <- sqftBase[sqftBase$sqft_basement != 0,]
sqftBase_1$group <- as.numeric(cut2(sqftBase_1$price, g=9))+1
sqftBase_0$group <- 1

sqftBase <- rbind(sqftBase_0, sqftBase_1)

grouped <- sqftBase %>%
           group_by(group) %>%
           dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                            MeanX = mean(sqft_basement, na.rm=TRUE),
                            Count = n())

# Adjusting 0 as 500, just so x-axis looks better
```

```r
grouped[1,"MeanX"] <- 500

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = MeanX)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 3) +
       geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1) +
       geom_point(aes(y = Count*scl, colour = "Count"), lwd = 3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'yr_renovated') +
       ggtitle(paste('Average Price vs Average yr_renovated'))
```




```r
## Very similar price for no basement or small basement, so creating an
indicator variable for if basement is greater than 750 sqft
Housing$basementCat <- as.factor(ifelse(Housing$sqft_basement > 750, 1, 0))
```

```
##- Looking into lat -##
lat       <- Housing[,c('price','lat')]
lat$group <- as.numeric(cut2(lat$price, g=150))

grouped <- lat %>%
             group_by(group) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              MeanX = round(mean(lat, na.rm=TRUE),4),
                              Count = n())

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = MeanX)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 0.1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 0.3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Latitude') +
       ggtitle(paste('Average Price vs Latitude'))
```



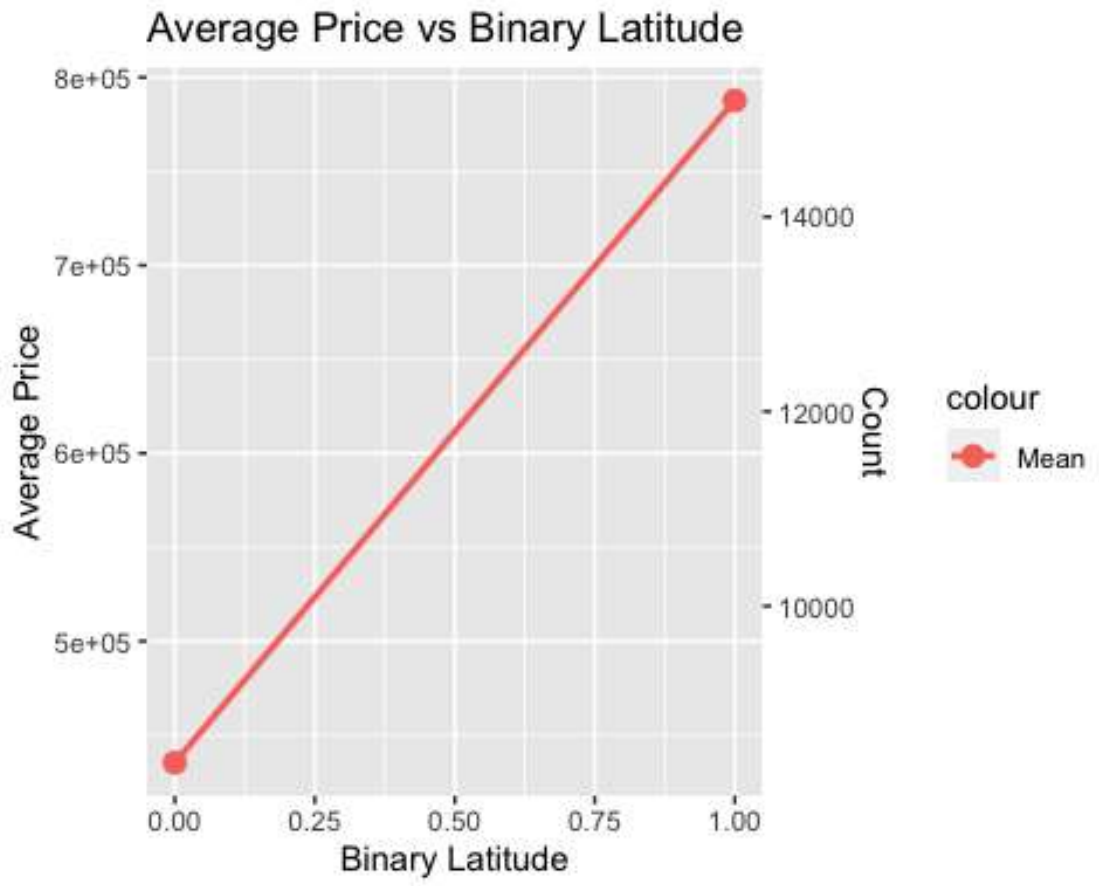

```
#- It's hard to see, but there's obviously something going on at certain
latitudes (possibly waters edge or something),
#- will create a binary predictor for certain latitudes (large risk of over-
```

```
fitting, so will have to be careful).
data.frame(grouped[order(grouped$MeanY, decreasing = T),])

##     group     MeanY   MeanX Count
## 1     149 3000791.0 47.6200   144
## 2     148 2011651.9 47.6168   143
## 3     147 1704452.0 47.6142   139
## 4     146 1534701.9 47.6157   142
## 5     145 1394378.4 47.6154   151
## 6     144 1300323.6 47.6213   140
## 7     143 1227201.5 47.6162   149
## 8     142 1161605.9 47.6141   141
## 9     141 1095390.5 47.6210   139
## 10    140 1041981.2 47.6163   152
## 11    139  996473.1 47.6217   138
## 12    138  967759.8 47.6211   147
## 13    137  941126.2 47.6115   148
## 14    136  917801.5 47.6261   140
## 15    135  896996.8 47.6195   148
## 16    134  876671.7 47.6342   144
## 17    133  859493.7 47.6218   129
## 18    132  845595.3 47.6225   159
## 19    131  829595.6 47.6290   144
## 20    130  815565.3 47.6198   144
## 21    129  802385.3 47.6185   135
## 22    128  791513.9 47.6104   152
## 23    127  780044.9 47.6292    96
## 24    126  769567.3 47.6194   194
## 25    125  756982.4 47.6173   123
## 26    124  748385.7 47.6192   165
## 27    123  736811.0 47.6235   144
## 28    122  726980.0 47.6132   140
## 29    121  718020.9 47.6237   148
## 30    120  707951.0 47.6156   144
## 31    119  699415.4 47.6228   144
## 32    118  690935.4 47.6155   142
## 33    117  681648.3 47.6217   146
## 34    116  674481.9 47.6173   124
## 35    115  666266.3 47.6154   150
## 36    114  657367.2 47.6309   156
## 37    113  650248.6 47.6267   141
## 38    112  645625.9 47.6070   118
## 39    111  637242.8 47.6217   172
## 40    110  629329.7 47.6298   115
## 41    109  624332.5 47.6214   132
## 42    108  617086.4 47.6257   184
## 43    107  609155.0 47.6217   118
## 44    106  603927.7 47.6150    92
## 45    105  599423.1 47.6227   190
## 46    104  591837.3 47.6120   170
```

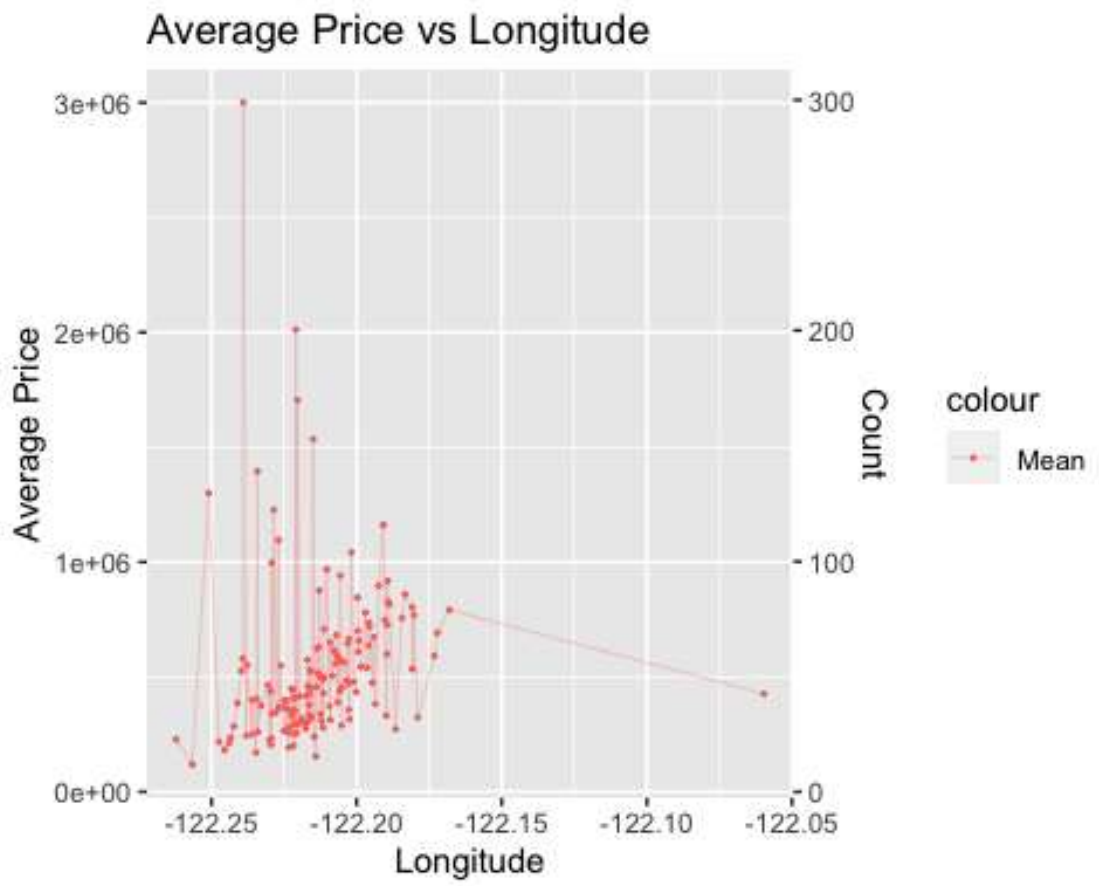
```
## 47    103  584795.3 47.6239   145
## 48    102  579016.1 47.6302   138
## 49    101  574158.2 47.6078   136
## 50    100  569088.1 47.6004   129
## 51     99  563877.5 47.6257   132
## 52     98  557839.7 47.6084   192
## 53     97  552351.3 47.6196    64
## 54     96  549667.7 47.6129   228
## 55     95  544197.6 47.5942   139
## 56     94  539289.5 47.6264   149
## 57     93  534510.6 47.6077   144
## 58     92  529671.8 47.6059   142
## 59     91  526508.5 47.6283    31
## 60     90  523248.5 47.6221   251
## 61     89  516068.0 47.6093   150
## 62     88  510471.9 47.6208   146
## 63     87  504386.6 47.6280   121
## 64     86  499999.7 47.6248   156
## 65     85  498474.8 47.6144   115
## 66     84  493949.5 47.6015   121
## 67     83  489183.1 47.6155   157
## 68     82  483965.7 47.6098   142
## 69     81  478922.9 47.5998   162
## 70     80  474381.2 47.6112   175
## 71     79  469290.4 47.6258   149
## 72     78  464962.4 47.6073   135
## 73     77  460662.6 47.6065   144
## 74     76  456336.6 47.6196   153
## 75     75  452474.5 47.5976    87
## 76     74  449994.0 47.5985   195
## 77     73  447619.3 47.6079    91
## 78     72  443685.7 47.6097   153
## 79     71  439570.8 47.6098   185
## 80     70  436658.2 47.6221    54
## 81     69  433817.1 47.6117   188
## 82     68  428952.5 47.5763   200
## 83     67  425530.0 47.6292     3
## 84     66  424356.2 47.6015   219
## 85     65  418989.7 47.5883   208
## 86     64  414777.6 47.6122   144
## 87     63  410508.5 47.5959   145
## 88     62  407254.2 47.5804    69
## 89     61  403632.2 47.6215   171
## 90     60  399985.2 47.5876   191
## 91     59  397767.3 47.5898   118
## 92     58  393892.3 47.5844   144
## 93     57  389678.0 47.5487   166
## 94     56  385381.2 47.5896   151
## 95     55  382075.3 47.5762    64
## 96     54  379131.2 47.5386   199
```

```
## 97     53  374913.0 47.5852   168
## 98     52  371975.6 47.5607    58
## 99     51  368880.5 47.5458   196
## 100    50  363853.0 47.5437   158
## 101    49  359789.3 47.5210   157
## 102    48  356989.3 47.5348    75
## 103    47  353788.4 47.5390   197
## 104    46  349242.8 47.5344   299
## 105    45  344481.6 47.5497   142
## 106    44  340407.2 47.5078   147
## 107    43  338006.7 47.4993   131
## 108    42  334454.1 47.5181   168
## 109    41  331505.8 47.4961    29
## 110    40  329081.3 47.5138   246
## 111    39  325000.0 47.5275   148
## 112    38  323162.4 47.4917   100
## 113    37  319314.4 47.4980   199
## 114    36  315129.6 47.4984   144
## 115    35  312438.7 47.4777    92
## 116    34  309191.5 47.5008   186
## 117    33  304998.9 47.5006   143
## 118    32  302087.1 47.4790    53
## 119    31  299733.9 47.4782   247
## 120    30  295562.6 47.4610   133
## 121    29  292366.7 47.4507    91
## 122    28  289023.6 47.4717   202
## 123    27  284669.3 47.4652   150
## 124    26  281261.8 47.4473    38
## 125    25  278921.5 47.4449   247
## 126    24  274833.7 47.4493   146
## 127    23  272086.9 47.4008    68
## 128    22  268985.0 47.4240   222
## 129    21  264855.9 47.4393   136
## 130    20  260741.0 47.4174   152
## 131    19  257899.9 47.4307   118
## 132    18  254011.5 47.4277   170
## 133    17  250792.6 47.3937    17
## 134    16  249031.9 47.4295   263
## 135    15  244173.6 47.4215   152
## 136    14  239926.8 47.4231   134
## 137    13  235714.7 47.4208   144
## 138    12  231104.8 47.4188   149
## 139    11  228100.0 47.4343    90
## 140    10  222903.3 47.4059   199
## 141     9  217994.0 47.3858    95
## 142     8  212247.0 47.4087   198
## 143     7  205875.2 47.3964   144
## 144     6  199701.1 47.4104   144
## 145     5  192083.8 47.4026   142
## 146     4  181920.2 47.3971   141
```

```
## 147     3  170050.5 47.4311   147
## 148     2  153915.0 47.4402   145
## 149     1  118263.9 47.4378   146

groups         <- c(140,141,142,143,144,145,146,147,148,149)
Housing$latBin <- ifelse(Housing$lat %in% unique(lat[lat$group %in% groups,
'lat']), 1, 0)

grouped <- Housing %>%
             group_by(latBin) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              Count = n())

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = latBin)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Binary Latitude') +
       ggtitle(paste('Average Price vs Binary Latitude'))
```



```r
##- Looking into Long -##    Note: Longitude could be highly correlated with
latitude, so need to watch out for that
long       <- Housing[,c('price','long')]
long$group <- as.numeric(cut2(lat$price, g=150))

grouped <- long %>%
             group_by(group) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              MeanX = round(mean(long, na.rm=TRUE),4),
                              Count = n())

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = MeanX)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 0.1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 0.3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Longitude') +
       ggtitle(paste('Average Price vs Longitude'))
```



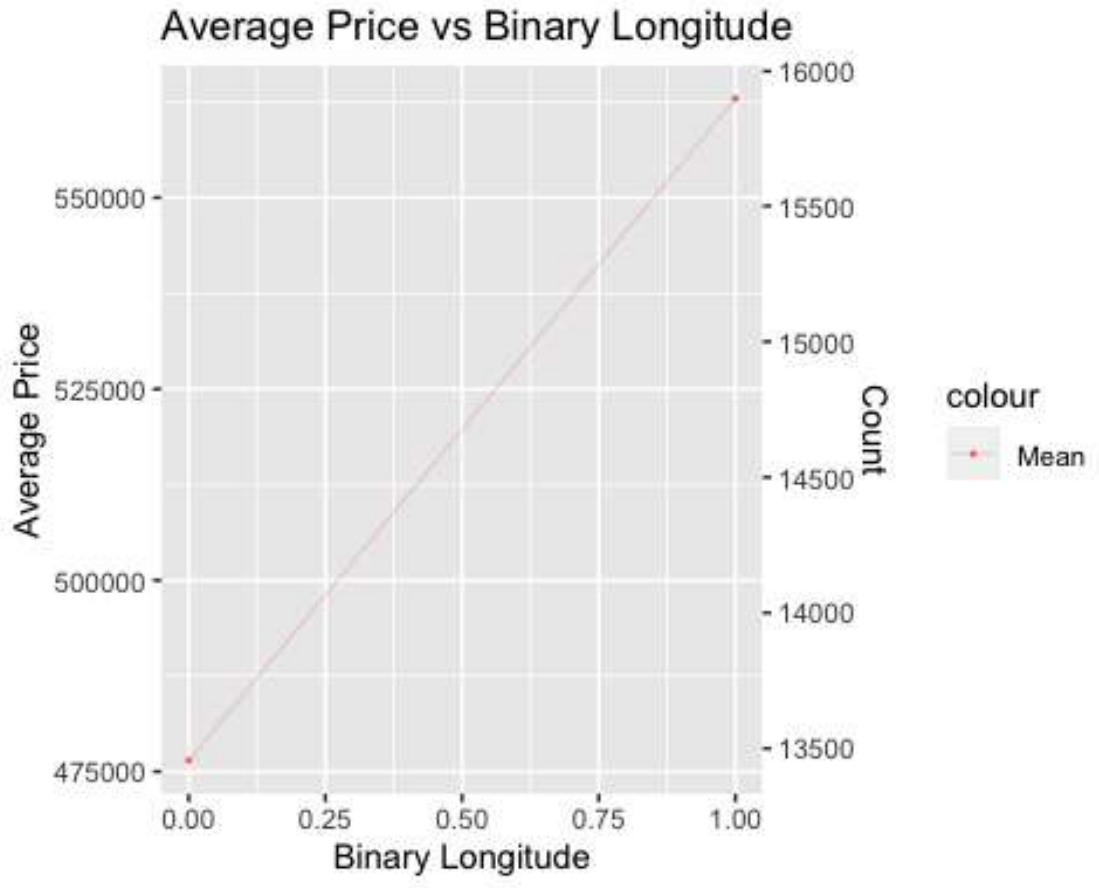

```r
# Peak isn't as obvious as latitude, but could be something there
data.frame(grouped[order(grouped$MeanY, decreasing = T),])
```

```
##     group     MeanY     MeanX Count
## 1     149 3000791.0 -122.2391   144
## 2     148 2011651.9 -122.2209   143
## 3     147 1704452.0 -122.2204   139
## 4     146 1534701.9 -122.2150   142
## 5     145 1394378.4 -122.2342   151
## 6     144 1300323.6 -122.2509   140
## 7     143 1227201.5 -122.2286   149
## 8     142 1161605.9 -122.1908   141
## 9     141 1095390.5 -122.2269   139
## 10    140 1041981.2 -122.2018   152
## 11    139  996473.1 -122.2293   138
## 12    138  967759.8 -122.2103   147
## 13    137  941126.2 -122.2056   148
## 14    136  917801.5 -122.1893   140
## 15    135  896996.8 -122.1924   148
## 16    134  876671.7 -122.2128   144
## 17    133  859493.7 -122.1833   129
## 18    132  845595.3 -122.1997   159
## 19    131  829595.6 -122.1892   144
## 20    130  815565.3 -122.1887   144
## 21    129  802385.3 -122.1809   135
## 22    128  791513.9 -122.1681   152
## 23    127  780044.9 -122.1970    96
## 24    126  769567.3 -122.1803   194
## 25    125  756982.4 -122.1844   123
## 26    124  748385.7 -122.1902   165
## 27    123  736811.0 -122.1958   144
## 28    122  726980.0 -122.1894   140
## 29    121  718020.9 -122.1956   148
## 30    120  707951.0 -122.2111   144
## 31    119  699415.4 -122.1996   144
## 32    118  690935.4 -122.1722   142
## 33    117  681648.3 -122.2070   146
## 34    116  674481.9 -122.1941   124
## 35    115  666266.3 -122.2028   150
## 36    114  657367.2 -122.1992   156
## 37    113  650248.6 -122.2091   141
## 38    112  645625.9 -122.2030   118
## 39    111  637242.8 -122.1958   172
## 40    110  629329.7 -122.2130   115
## 41    109  624332.5 -122.2136   132
## 42    108  617086.4 -122.2079   184
## 43    107  609155.0 -122.1995   118
## 44    106  603927.7 -122.2071    92
## 45    105  599423.1 -122.1895   190
## 46    104  591837.3 -122.1732   170
## 47    103  584795.3 -122.2063   145
## 48    102  579016.1 -122.2392   138
## 49    101  574158.2 -122.2169   136
```

```
## 50    100  569088.1 -122.2053   129
## 51     99  563877.5 -122.2044   132
## 52     98  557839.7 -122.2071   192
## 53     97  552351.3 -122.2377    64
## 54     96  549667.7 -122.2261   228
## 55     95  544197.6 -122.1985   139
## 56     94  539289.5 -122.1964   149
## 57     93  534510.6 -122.1808   144
## 58     92  529671.8 -122.2160   142
## 59     91  526508.5 -122.2397    31
## 60     90  523248.5 -122.2158   251
## 61     89  516068.0 -122.2131   150
## 62     88  510471.9 -122.2132   146
## 63     87  504386.6 -122.2084   121
## 64     86  499999.7 -122.2113   156
## 65     85  498474.8 -122.2117   115
## 66     84  493949.5 -122.2123   121
## 67     83  489183.1 -122.2114   157
## 68     82  483965.7 -122.2033   142
## 69     81  478922.9 -122.2010   162
## 70     80  474381.2 -122.1946   175
## 71     79  469290.4 -122.2029   149
## 72     78  464962.4 -122.2305   135
## 73     77  460662.6 -122.2168   144
## 74     76  456336.6 -122.2138   153
## 75     75  452474.5 -122.2052    87
## 76     74  449994.0 -122.2159   195
## 77     73  447619.3 -122.2224    91
## 78     72  443685.7 -122.2217   153
## 79     71  439570.8 -122.2057   185
## 80     70  436658.2 -122.2296    54
## 81     69  433817.1 -122.2001   188
## 82     68  428952.5 -122.2114   200
## 83     67  425530.0 -122.0597     3
## 84     66  424356.2 -122.2164   219
## 85     65  418989.7 -122.2173   208
## 86     64  414777.6 -122.2196   144
## 87     63  410508.5 -122.2217   145
## 88     62  407254.2 -122.2213    69
## 89     61  403632.2 -122.2344   171
## 90     60  399985.2 -122.2360   191
## 91     59  397767.3 -122.2249   118
## 92     58  393892.3 -122.2241   144
## 93     57  389678.0 -122.2065   166
## 94     56  385381.2 -122.2410   151
## 95     55  382075.3 -122.1935    64
## 96     54  379131.2 -122.2165   199
## 97     53  374913.0 -122.2329   168
## 98     52  371975.6 -122.2094    58
## 99     51  368880.5 -122.2268   196
```

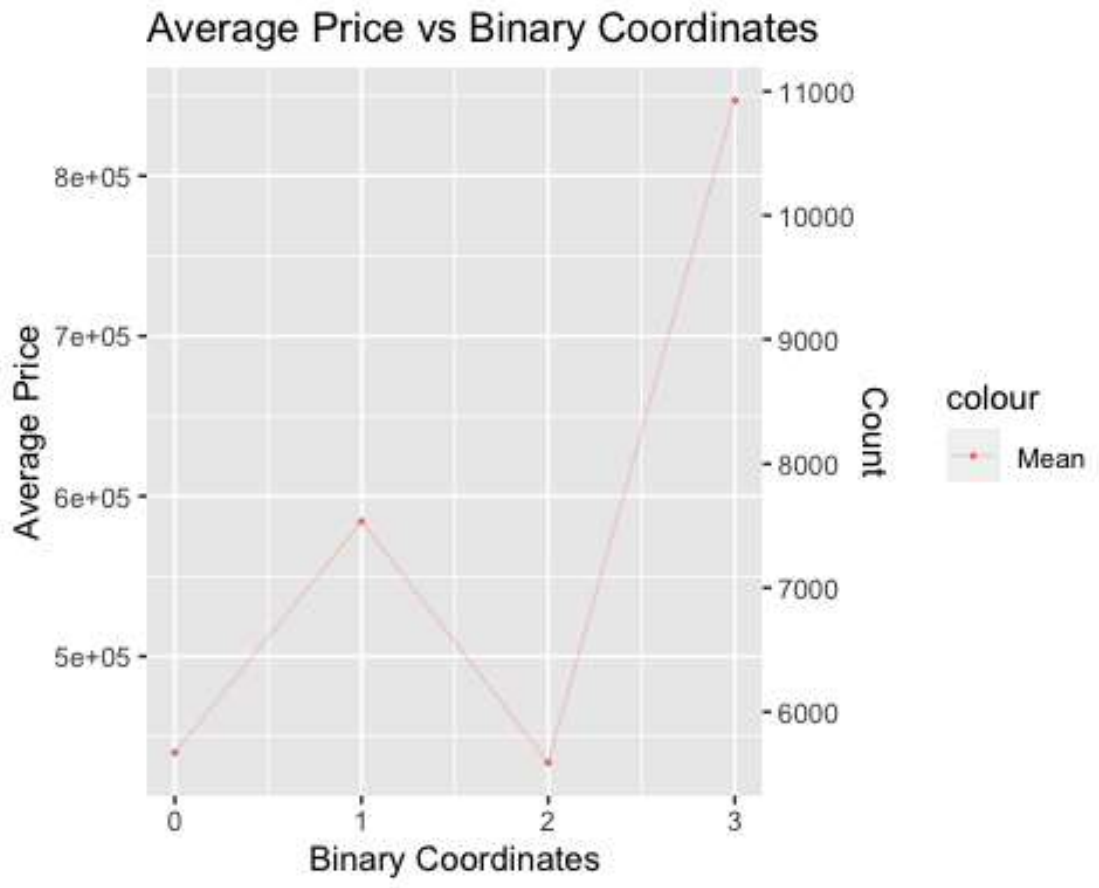
```
## 100    50  363853.0 -122.2250   158
## 101    49  359789.3 -122.2235   157
## 102    48  356989.3 -122.2027    75
## 103    47  353788.4 -122.2212   197
## 104    46  349242.8 -122.2222   299
## 105    45  344481.6 -122.2279   142
## 106    44  340407.2 -122.2290   147
## 107    43  338006.7 -122.2124   131
## 108    42  334454.1 -122.2164   168
## 109    41  331505.8 -122.1898    29
## 110    40  329081.3 -122.2222   246
## 111    39  325000.0 -122.2154   148
## 112    38  323162.4 -122.1789   100
## 113    37  319314.4 -122.2190   199
## 114    36  315129.6 -122.2024   144
## 115    35  312438.7 -122.2091    92
## 116    34  309191.5 -122.2120   186
## 117    33  304998.9 -122.2166   143
## 118    32  302087.1 -122.2187    53
## 119    31  299733.9 -122.2182   247
## 120    30  295562.6 -122.2214   133
## 121    29  292366.7 -122.2204    91
## 122    28  289023.6 -122.2052   202
## 123    27  284669.3 -122.2423   150
## 124    26  281261.8 -122.2230    38
## 125    25  278921.5 -122.2116   247
## 126    24  274833.7 -122.2175   146
## 127    23  272086.9 -122.1866    68
## 128    22  268985.0 -122.2241   222
## 129    21  264855.9 -122.2251   136
## 130    20  260741.0 -122.2341   152
## 131    19  257899.9 -122.2237   118
## 132    18  254011.5 -122.2207   170
## 133    17  250792.6 -122.2215    17
## 134    16  249031.9 -122.2360   263
## 135    15  244173.6 -122.2380   152
## 136    14  239926.8 -122.2146   134
## 137    13  235714.7 -122.2434   144
## 138    12  231104.8 -122.2294   149
## 139    11  228100.0 -122.2621    90
## 140    10  222903.3 -122.2300   199
## 141     9  217994.0 -122.2474    95
## 142     8  212247.0 -122.2439   198
## 143     7  205875.2 -122.2295   144
## 144     6  199701.1 -122.2216   144
## 145     5  192083.8 -122.2233   142
## 146     4  181920.2 -122.2455   141
## 147     3  170050.5 -122.2346   147
## 148     2  153915.0 -122.2141   145
## 149     1  118263.9 -122.2566   146
```

```
groups          <- c(142,143,144,145,146,147,148,149)
Housing$longBin <- ifelse(Housing$long %in% unique(long[long$group %in%
groups, 'long']), 1, 0)

grouped <- Housing %>%
             group_by(longBin) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              Count = n())

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = longBin)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 0.1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 0.3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Binary Longitude') +
       ggtitle(paste('Average Price vs Binary Longitude'))
```




```
## Plotting both Longitude and Latitude together
latLong       <- Housing[,c('price','lat','long','latBin','longBin')]
latLong$group <- ifelse(latLong$latBin == 0 & latLong$longBin == 0, 0,
                        ifelse(latLong$latBin == 1 & latLong$longBin == 0, 1,
```

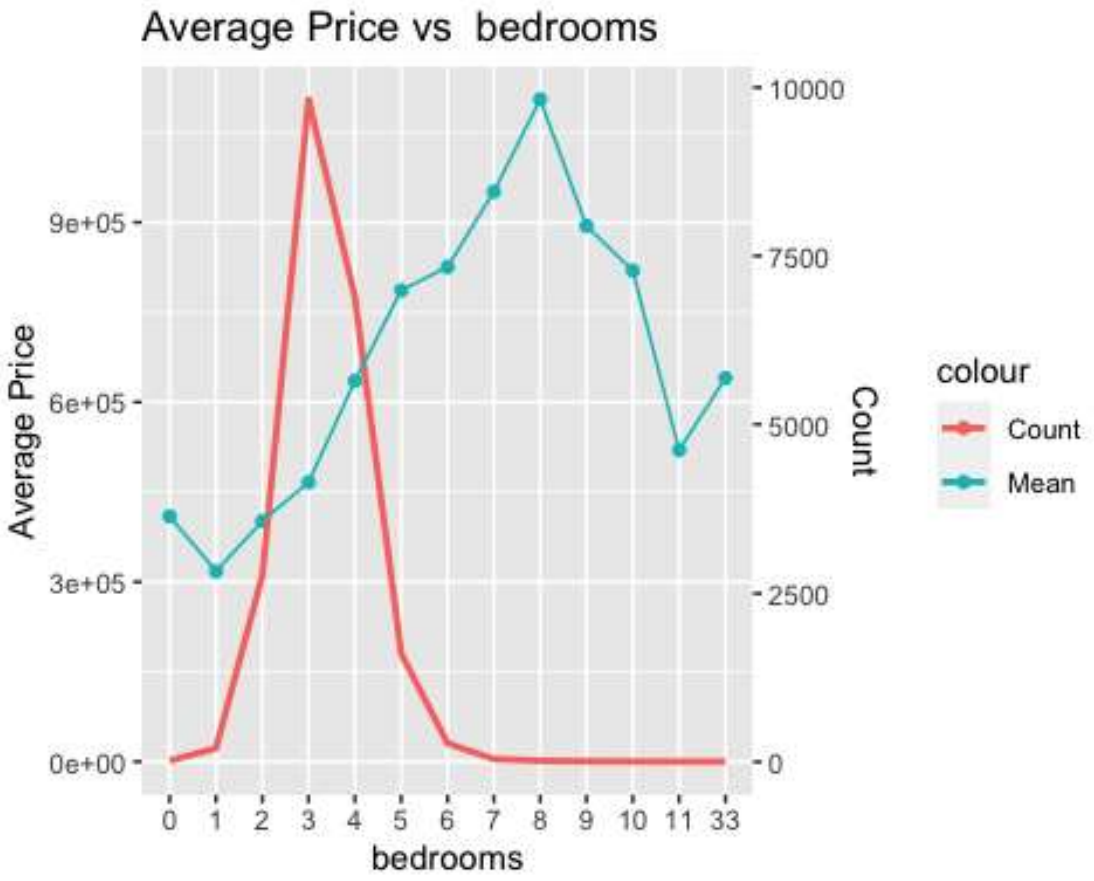
```
                                ifelse(latLong$latBin == 0 & latLong$longBin
== 1, 2, 3)))

## Definitions:
#   - 0: Not in either group
#   - 1: Only in Latitude
#   - 2: Only in Longitude
#   - 3: In both groups

grouped <- latLong %>%
             group_by(group) %>%
             dplyr::summarize(MeanY = mean(price, na.rm=TRUE),
                              Count = n())

scl <- max(grouped$MeanY)/max(grouped$Count)

ggplot(grouped, aes(x = group)) +
       geom_line(aes(y = MeanY, colour = "Mean"), lwd = 0.1) +
       geom_point(aes(y = MeanY, colour = "Mean"), lwd = 0.3) +
       scale_y_continuous(name = "Average Price", sec.axis = sec_axis(~./scl,
name = "Count")) +
       labs(x = 'Binary Coordinates') +
       ggtitle(paste('Average Price vs Binary Coordinates'))
```



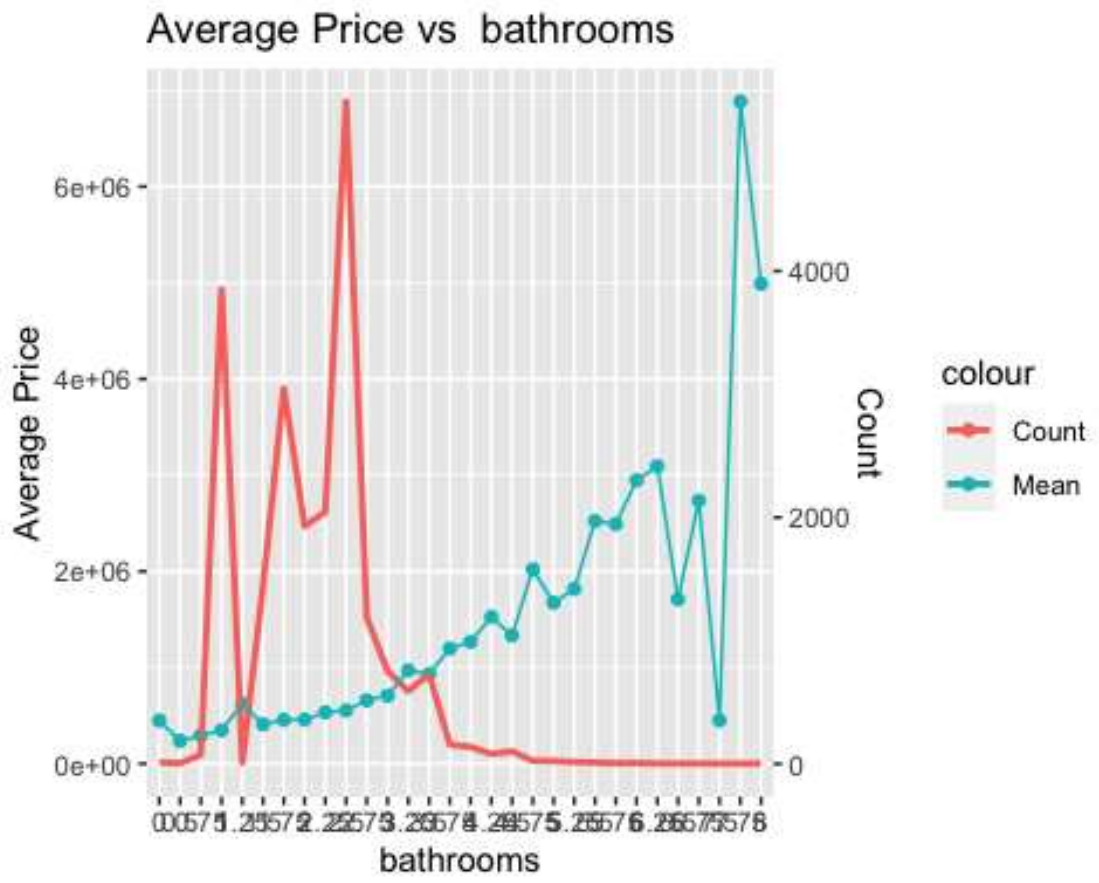
```
## Looks like being only in the longitude group isn't any different to being
in no groups, need to keep an eye on this when modeling
```

```
#--- Categorical Predictors ---#
for (i in 1:length(catPred)) {
  catPr <- catPred[i]

  print(catPr)

  grouped <- Housing %>%
             group_by(.data[[catPr]]) %>%
             dplyr::summarize(Mean = mean(price, na.rm=TRUE),
                              Count = n())

  scl     <- max(grouped$Mean)/max(grouped$Count)

  print(ggplot(grouped, aes(x=.data[[catPr]], y=Mean, group=1, color =
"Mean")) +
               geom_point(stat='summary', fun=sum) +
               geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1)  +
               scale_y_continuous(name = "Average Price", sec.axis =
sec_axis(~./scl, name = "Count")) +
               stat_summary(fun=mean, geom="line") +
               ggtitle(paste('Average Price vs ', catPr)))
}
```

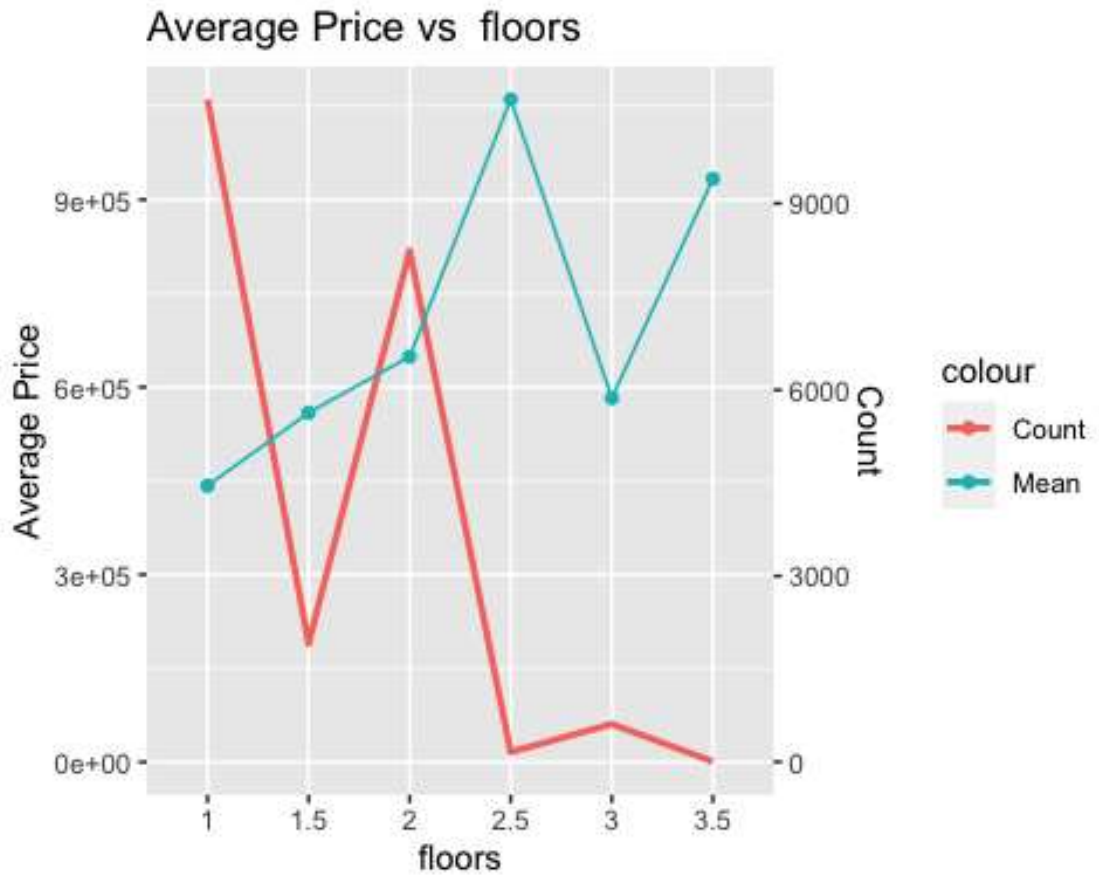
```
## [1] "bedrooms"
```


Average Price vs bedrooms
Average Price
9e+05
6e+05
3e+05
0e+00
10000
7500
5000
2500
0
Count
0 1 2 3 4 5 6 7 8 9 10 11 33
bedrooms
colour
Count
Mean

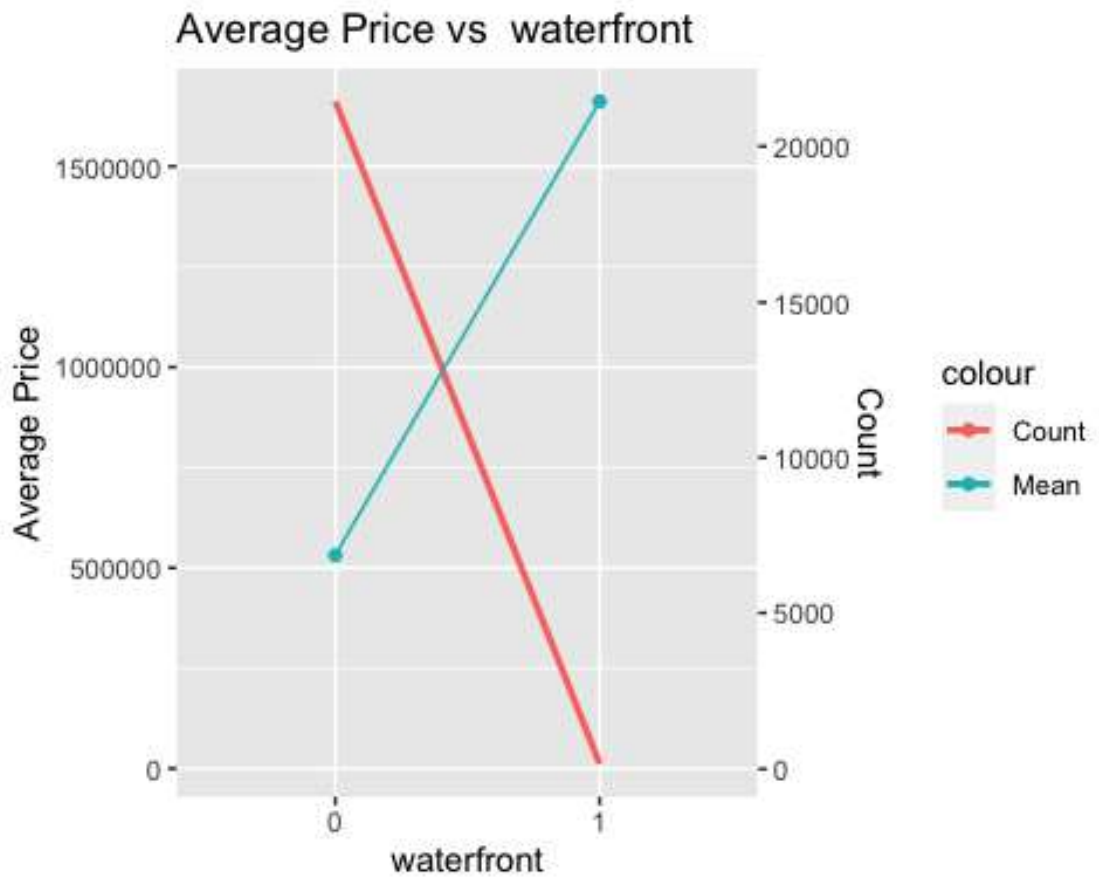
```
## [1] "bathrooms"
```


Average Price vs bathrooms
6e+06
4e+06
2e+06
0e+00
Average Price
4000
2000
0
Count
bathrooms
colour
Count
Mean

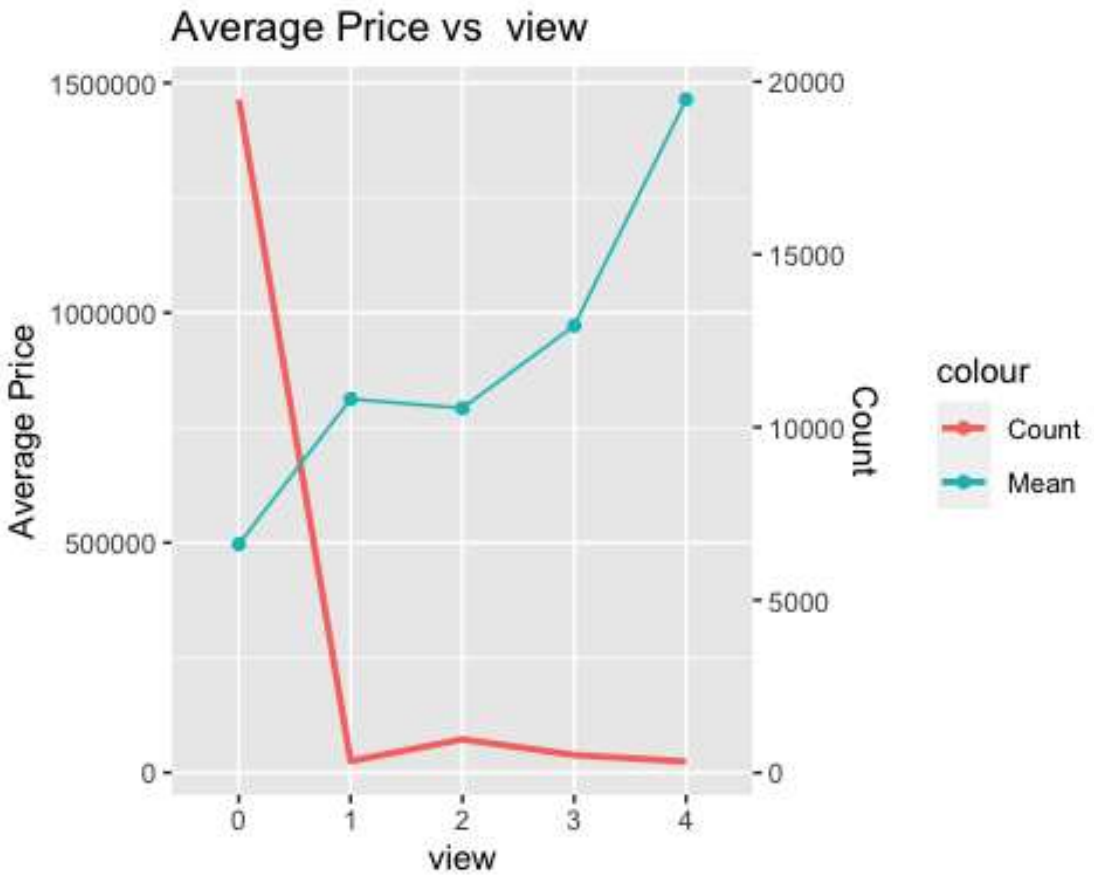
```
## [1] "floors"
```



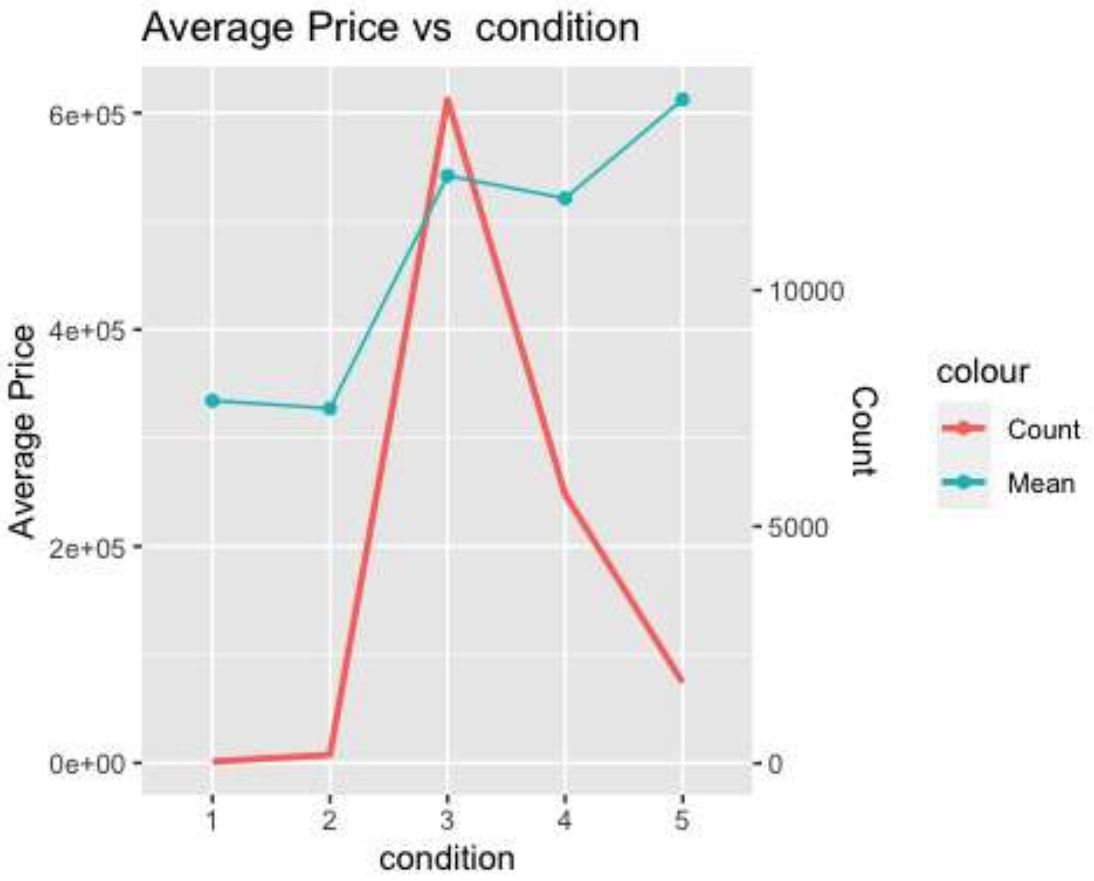

```
## [1] "waterfront"
```


Average Price vs waterfront
1500000
1000000
500000
0
Average Price
20000
15000
10000
5000
0
Count
0
1
waterfront
colour
Count
Mean

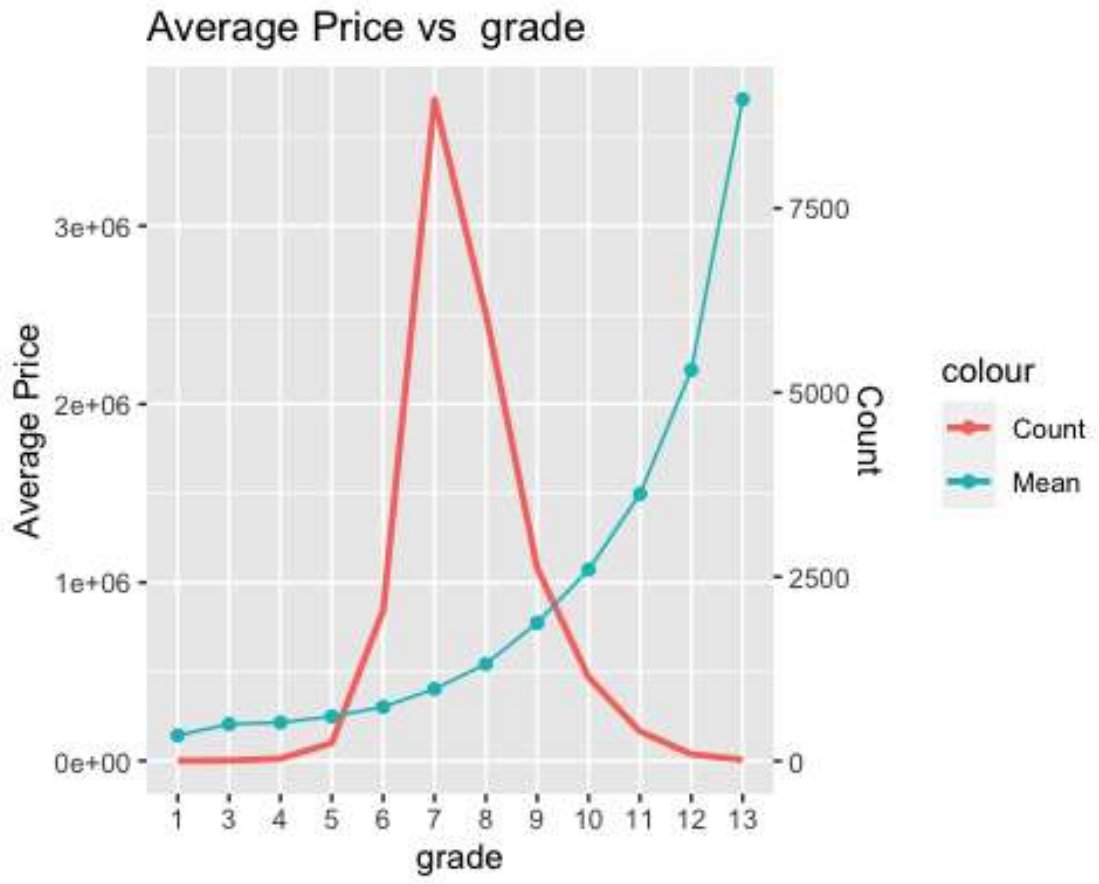
```
## [1] "view"
```


Average Price vs view
1500000
1000000
500000
0
Average Price
20000
15000
10000
5000
0
Count
0
1
2
3
4
view
colour
Count
Mean

```
## [1] "condition"
```

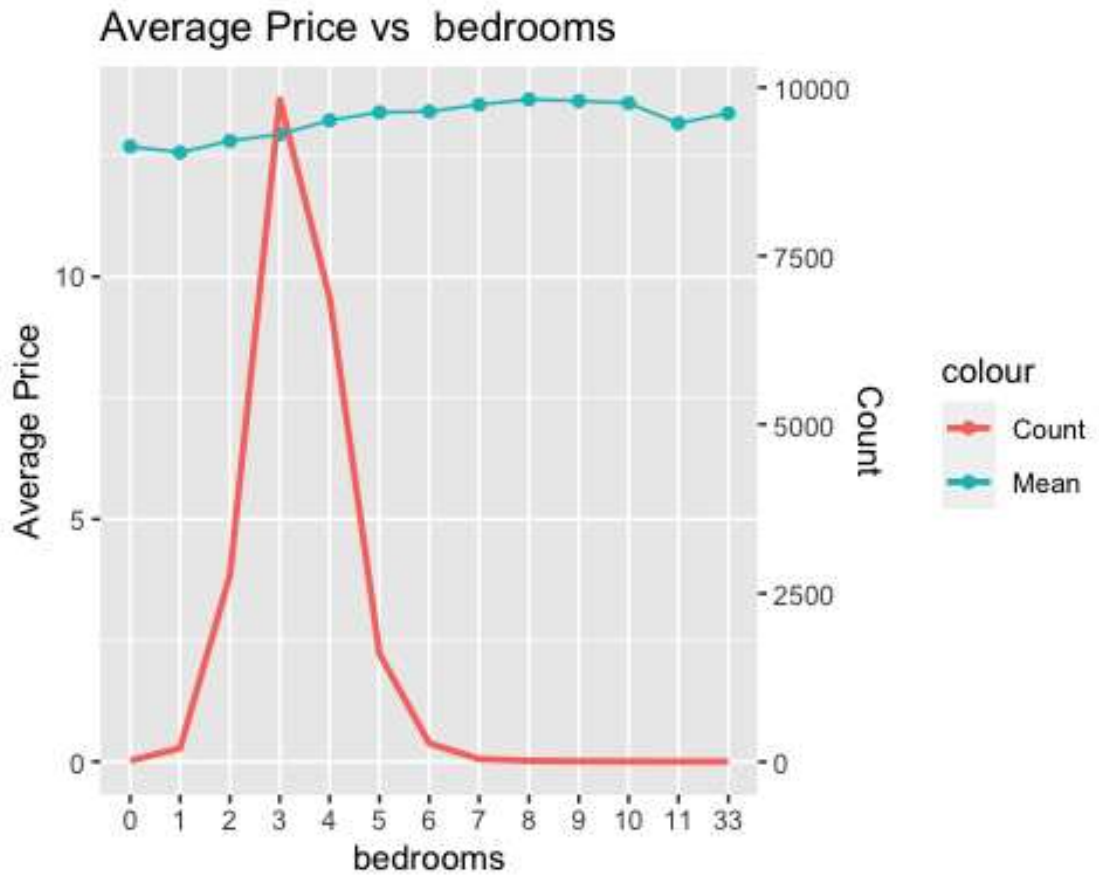

Average Price vs condition
6e+05
4e+05
2e+05
0e+00
Average Price
10000
5000
0
Count
1
2
3
4
5
condition
colour
Count
Mean

```
## [1] "grade"
```

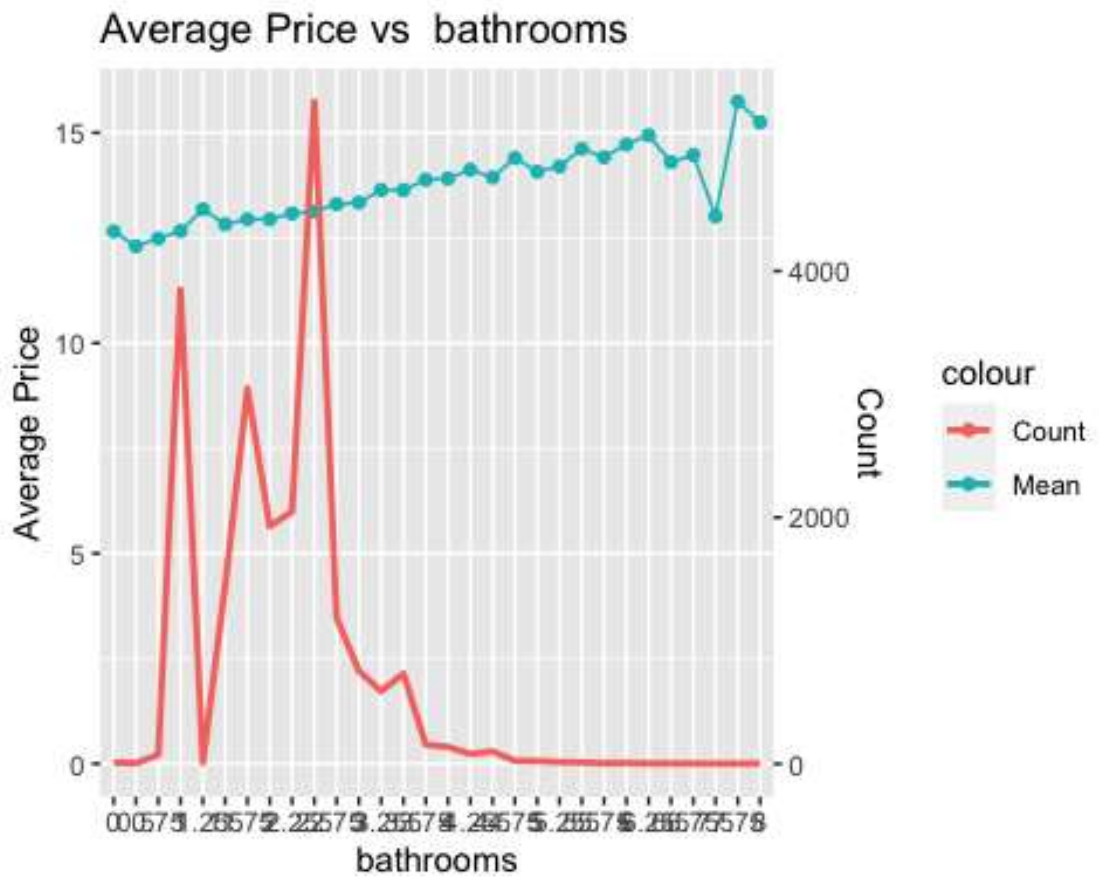



```
## Looking at Ln(Price)
for (i in 1:length(catPred)) {
  catPr <- catPred[i]

  print(catPr)

  grouped <- Housing %>%
             group_by(.data[[catPr]]) %>%
             dplyr::summarize(Mean = mean(log(price), na.rm=TRUE),
                              Count = n())

  scl     <- max(grouped$Mean)/max(grouped$Count)

  print(ggplot(grouped, aes(x=.data[[catPr]], y=Mean, group=1, color =
"Mean")) +
              geom_point(stat='summary', fun=sum) +
              geom_line(aes(y = Count*scl, colour = "Count"), lwd = 1)  +
              scale_y_continuous(name = "Average Price", sec.axis =
sec_axis(~./scl, name = "Count")) +
              stat_summary(fun=mean, geom="line") +
              ggtitle(paste('Average Price vs ', catPr)))
}
```

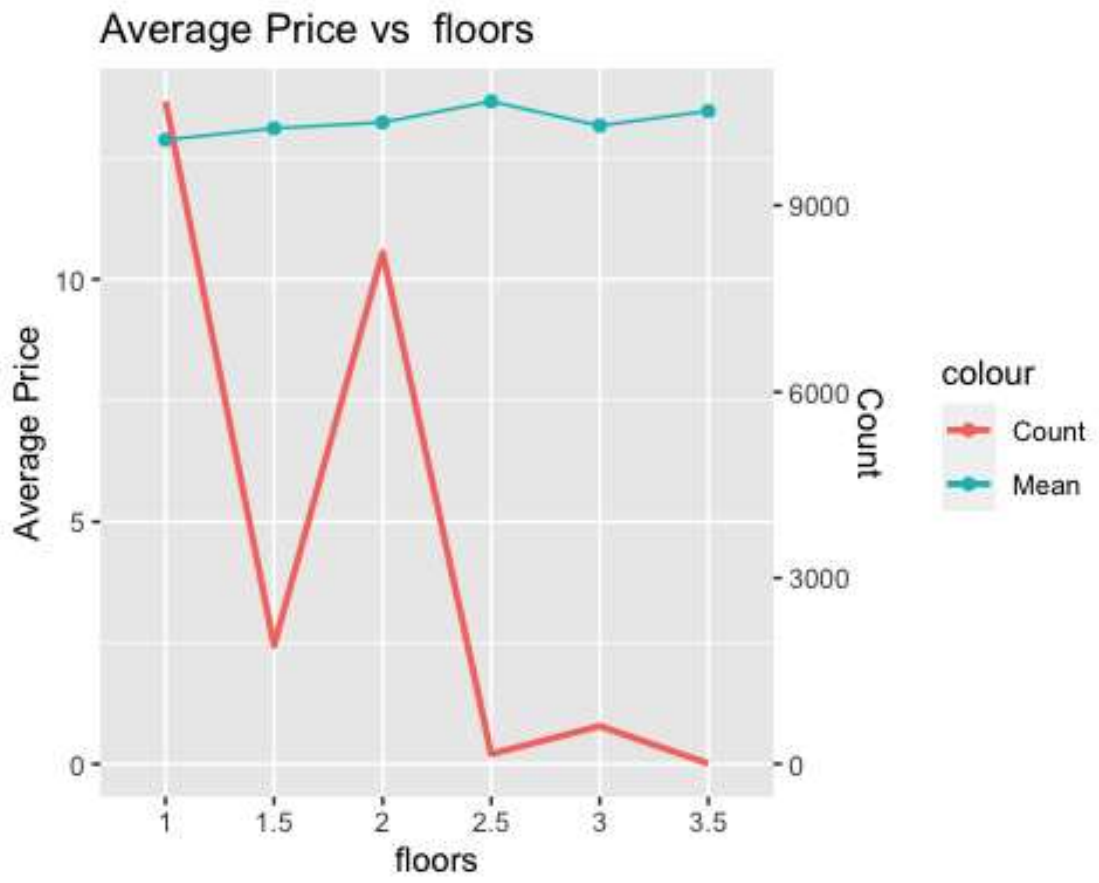
```
## [1] "bedrooms"
```



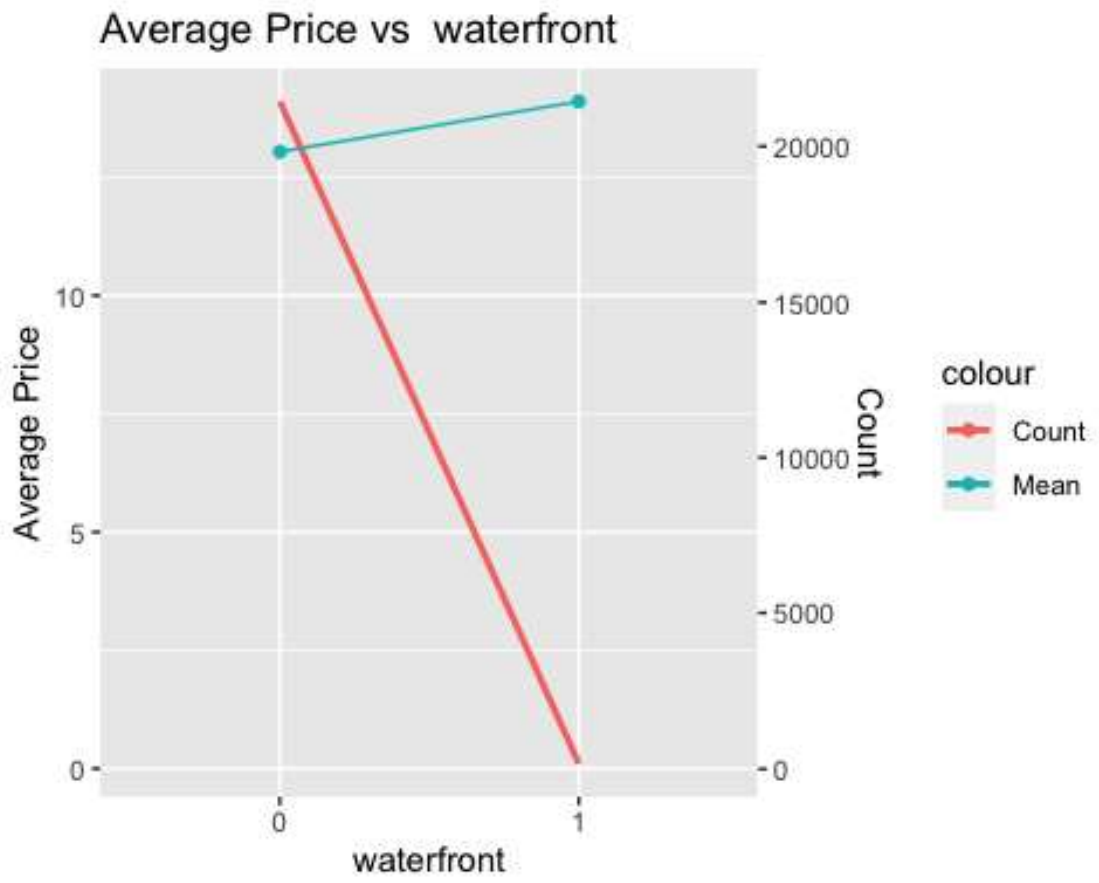

```
## [1] "bathrooms"
```


Average Price vs bathrooms
15
10
5
0
Average Price
4000
2000
0
Count
colour
Count
Mean
bathrooms

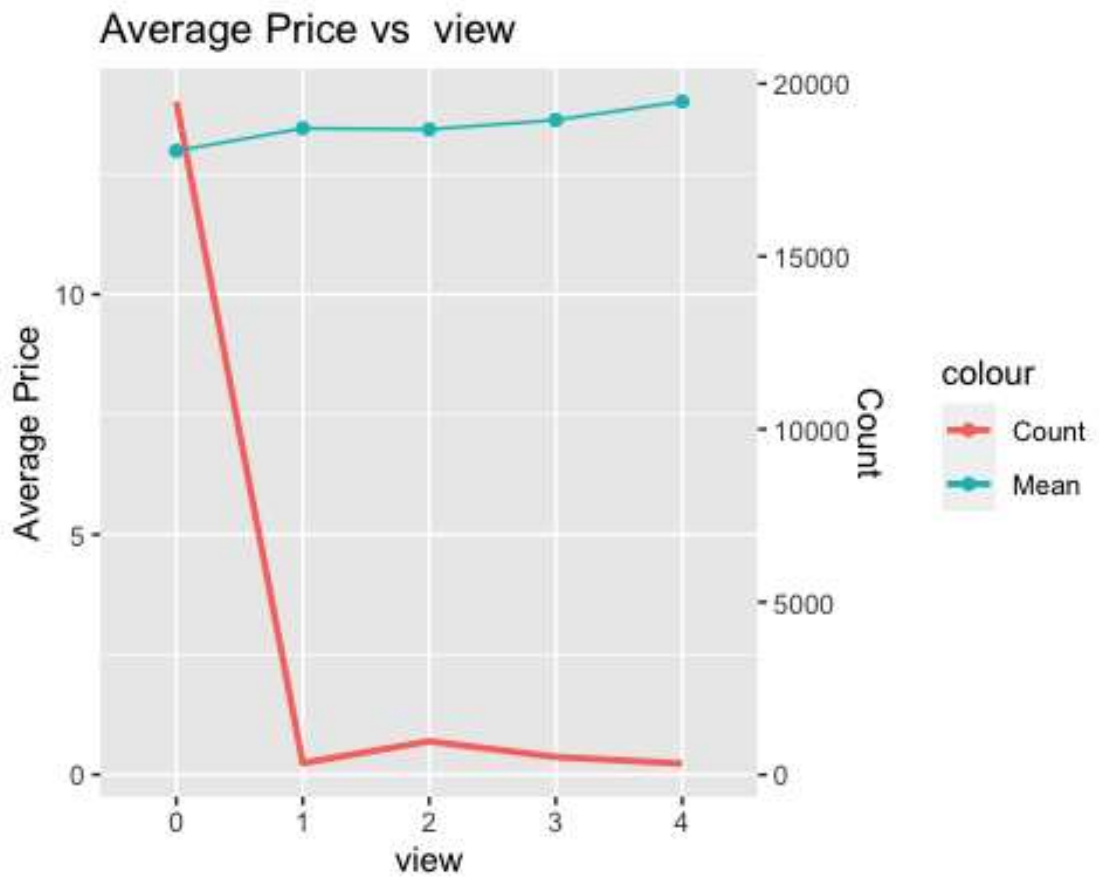
```
## [1] "floors"
```



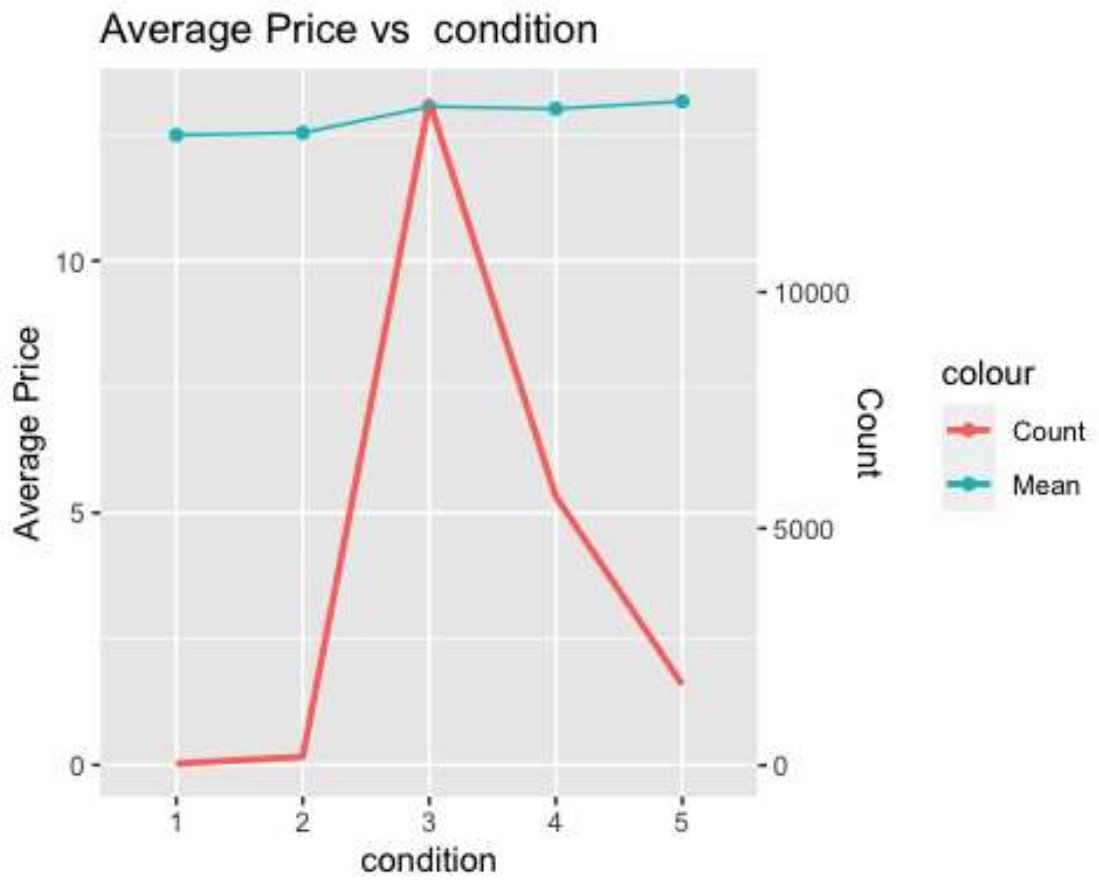

```
## [1] "waterfront"
```


Average Price vs waterfront
Average Price
10
5
0
20000
15000
10000
5000
0
Count
0
1
waterfront
colour
Count
Mean

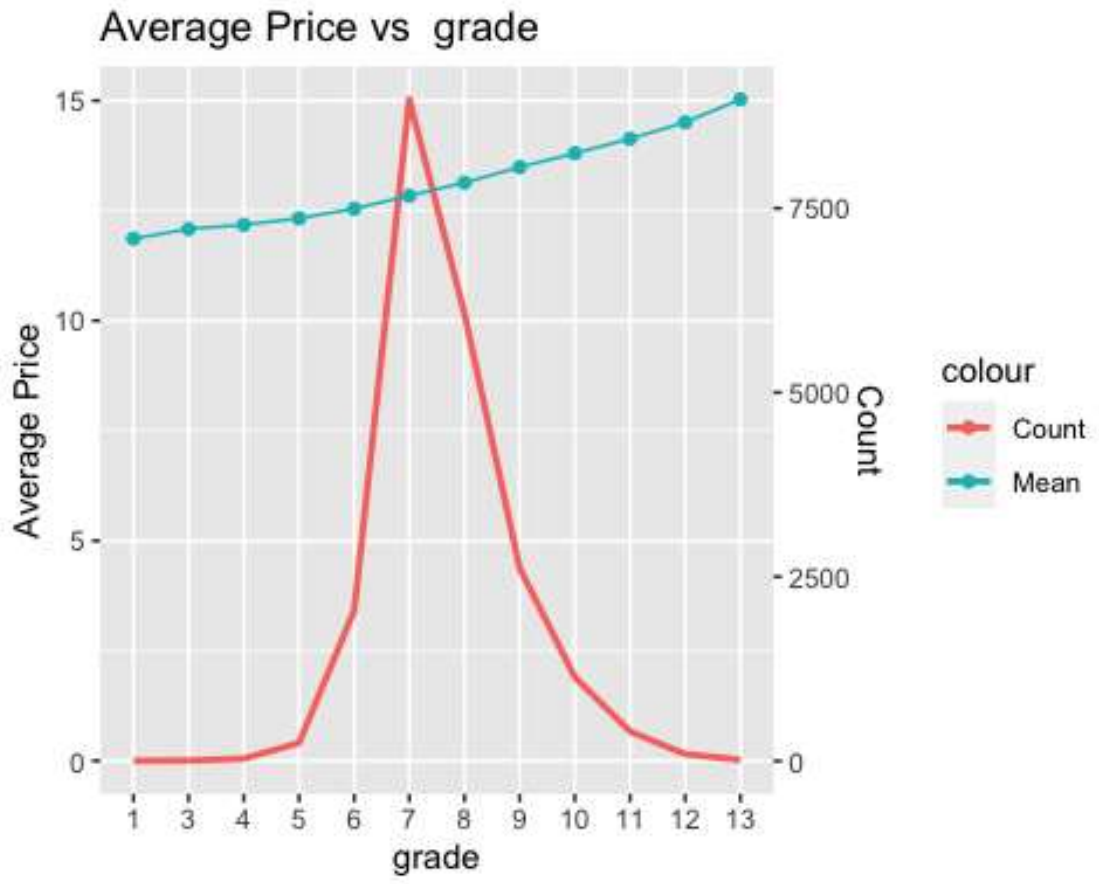
```
## [1] "view"
```


Average Price vs view
Average Price
Count
view
colour
Count
Mean
0
5
10
0
5000
10000
15000
20000
0
1
2
3
4

```
## [1] "condition"
```


Average Price vs condition
Average Price
10
5
0
10000
5000
0
Count
1
2
3
4
5
condition
colour
Count
Mean

```
## [1] "grade"
```



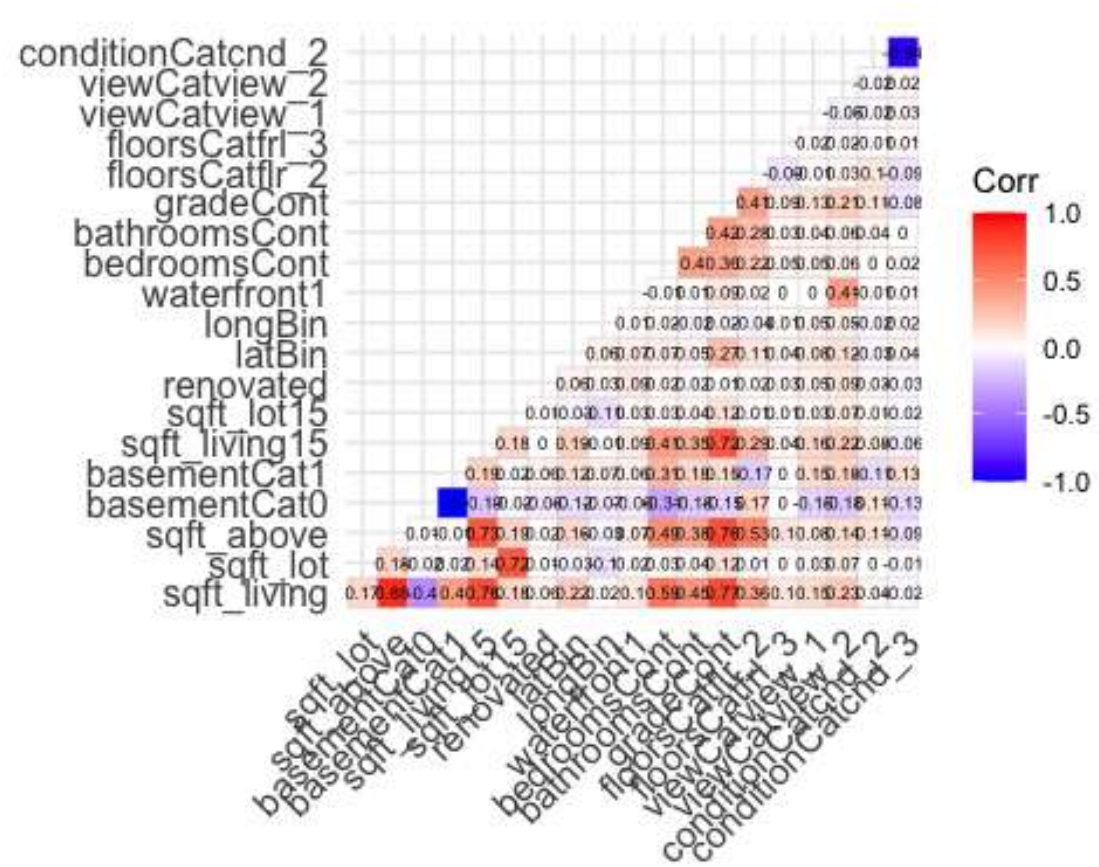

```
### Observations ###

#-- Could treat bedrooms as continuous given the observed trends, low count
above 7, so maybe group them
#-- Same for bathrooms, except group below 0.75 and above 4.75
#-- Group view, low count from 1 and above
#-- Grade looks really predictive, need to group low counts


## Note: Given the trends of bedrooms, bathrooms and grade, it is worth
modeling them as both a factor and continuous to
##       see if either will work better


#### -- Grouping Factors -- ####

# We need to keep in mind that regsubsets is very computationally expensive
if too many variables are used, so we need to included a limited number
predictors, these predictors need
# to be the most likely variables to add value to the model. Also, if a
factor has too many levels, this will dramatically increase the computational
time to fit.
```

```
Housing$bedroomsCont  <- as.numeric(ifelse(as.numeric(Housing$bedrooms) > 6,
7, Housing$bedrooms))
Housing$bathroomsCont <- as.numeric(ifelse(as.numeric(Housing$bathrooms) <
0.75, 0.75,
                                    ifelse(as.numeric(Housing$bathrooms) >
4.75, 4.75, Housing$bathrooms)))
Housing$gradeCont     <- as.numeric(ifelse(as.numeric(Housing$grade) < 5, 5,
Housing$grade))

Housing$floorsCat     <- ifelse(Housing$floors %in% c('1.5','2','3'),
'flr_2',
                                ifelse(Housing$floors %in% c('2.5','3.5'),
'frl_3', 'flr_1'))

Housing$viewCat       <- ifelse(Housing$view %in% c('1','2'), 'view_1',
                                ifelse(Housing$view %in% c('3','4'),
'view_2', 'view_0'))

Housing$conditionCat  <- ifelse(Housing$condition %in% c('1','2'), 'cnd_1',
                                ifelse(Housing$condition %in% c('3','4'),
'cnd_2', 'cnd_3'))

str(Housing)

## 'data.frame':    21613 obs. of  31 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000"
"20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : Factor w/ 13 levels "0","1","2","3",..: 4 4 3 5 4 5 4 4
4 4 ...
##  $ bathrooms    : Factor w/ 30 levels "0","0.5","0.75",..: 4 9 4 12 8 18 9
6 4 10 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890
...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470
6560 ...
##  $ floors       : Factor w/ 6 levels "1","1.5","2",..: 1 3 1 1 1 1 3 1 1 3
...
##  $ waterfront   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ view         : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1
1 ...
##  $ condition    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 5 3 3 3 3 3
3 ...
##  $ grade        : Factor w/ 12 levels "1","3","4","5",..: 6 6 5 6 7 10 6 6
6 6 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890
...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
```

```
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003
...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198
98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390
...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570
...
##  $ renovated    : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ basementCat  : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
##  $ latBin       : num  0 0 0 1 1 1 0 0 1 0 ...
##  $ longBin      : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ bedroomsCont : num  4 4 3 5 4 5 4 4 4 4 ...
##  $ bathroomsCont: num  4 4.75 4 4.75 4.75 4.75 4.75 4.75 4 4.75 ...
##  $ gradeCont    : num  6 6 5 6 7 10 6 6 6 6 ...
##  $ floorsCat    : chr  "flr_1" "flr_2" "flr_1" "flr_1" ...
##  $ viewCat      : chr  "view_0" "view_0" "view_0" "view_0" ...
##  $ conditionCat : chr  "cnd_2" "cnd_2" "cnd_2" "cnd_3" ...
```

```
## Scale Data
catPred    <-
c('renovated','latBin','longBin','waterfront','floorsCat','viewCat','conditio
nCat','basementCat')
contPred   <-
c('sqft_living','sqft_lot','sqft_above','sqft_living15','sqft_lot15','bedroom
sCont','bathroomsCont','gradeCont')
HousingRed <- cbind(scale(Housing[,c(contPred)]),
Housing[,c(catPred,'price')])


# Split into train and test
train <- sample(1:nrow(HousingRed),round(nrow(HousingRed)*0.7),rep=F)
test  <- -train


### Regsubsets is very computationally expensive if too many variables are
used, so we need to included a limited number predictors, these predictors
need
### to be the most likely variables to add value to the model

# Predictors
predictors <-
c('sqft_living','sqft_lot','sqft_above','basementCat','sqft_living15','sqft_l
ot15','renovated','latBin','longBin',

'waterfront','bedroomsCont','bathroomsCont','gradeCont','floorsCat','viewCat'
,'conditionCat')
```

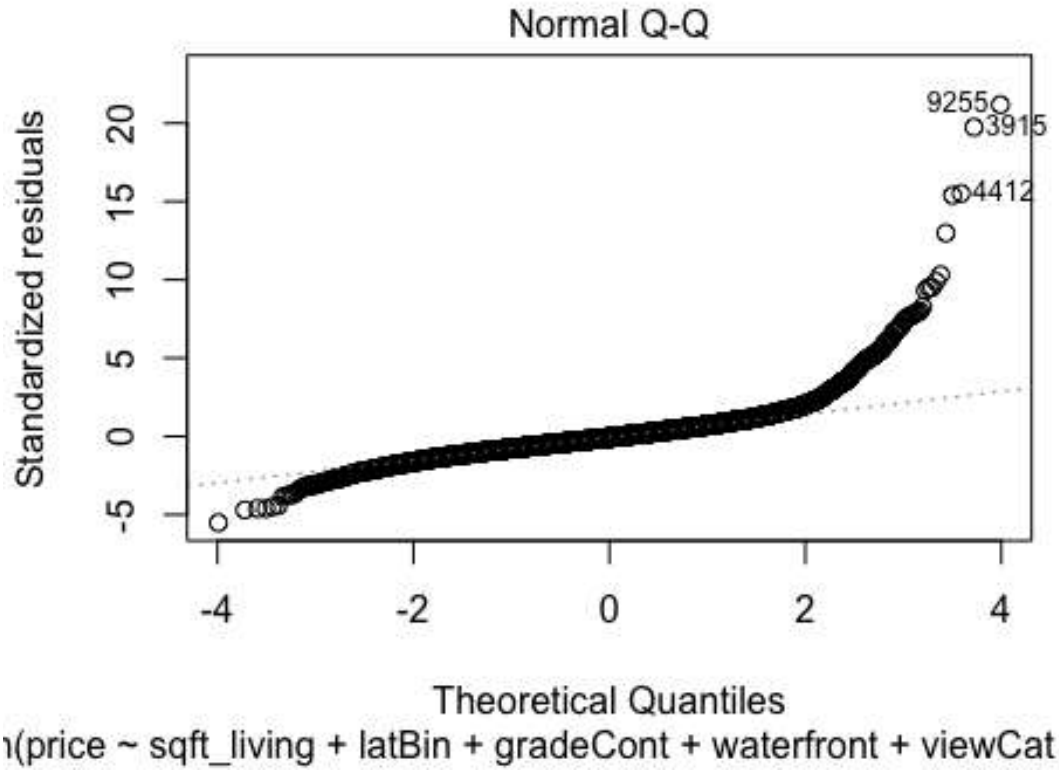
```r
## Looking at correlation Matrix
model.matrix(~0+., data=HousingRed[,predictors]) %>%
  cor(use="pairwise.complete.obs") %>%
  ggcorrplot(show.diag = F, type="lower", lab=TRUE, lab_size=2)
```




```r
#-- Some high correlations, but mostly fine


# Ranked Prediction Function
rankedPrediction <- function (pred, actual) {
  reduc           <- data.frame(cbind(pred, actual))
  colnames(reduc) <- c("Pred", "Actual")
  reduc$group     <- as.numeric(cut2(reduc$Actual, g=20))

  grouped <- reduc %>%
              group_by(group) %>%
              dplyr::summarize(Actual = mean(Actual, na.rm=TRUE),
                               Pred = mean(Pred, na.rm=TRUE),
                               Count = n())
```

```
  print(ggplot(grouped, aes(x = group)) +
              geom_line(aes(y = Actual, colour = "Actual"), lwd = 1) +
              geom_point(aes(y = Actual, colour = "Actual"), lwd = 3) +
              geom_line(aes(y = Pred, colour = "Pred"), lwd = 1) +
              geom_point(aes(y = Pred, colour = "Pred"), lwd = 3) +
              labs(x = "Group") +
              ggtitle(paste('Average Actual vs Predicted')))
}


### Forward Stepwise ###
lmp <- lm(price ~ ., data = HousingRed[train,c('price',predictors)])
lm0 <- lm(price ~ 1, data = HousingRed[train,c('price',predictors)])
fwd <- step(lm0, scope = list(lower = lm0, upper = lmp), direction =
"forward", k = 2, trace = 0)
summary(fwd)

##
## Call:
## lm(formula = price ~ sqft_living + latBin + gradeCont + waterfront +
##     viewCat + bathroomsCont + conditionCat + longBin + renovated +
##     floorsCat + bedroomsCont + sqft_lot15 + sqft_living15 + sqft_above +
##     basementCat + sqft_lot, data = HousingRed[train, c("price",
##     predictors)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1130252  -112318   -15645    90371  4336770
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         418146      18098  23.104  < 2e-16 ***
## sqft_living         199147       7098  28.059  < 2e-16 ***
## latBin              186726       3875  48.189  < 2e-16 ***
## gradeCont           105239       3025  34.792  < 2e-16 ***
## waterfront1         522723      20884  25.030  < 2e-16 ***
## viewCatview_1        86085       7356  11.702  < 2e-16 ***
## viewCatview_2       153218      10078  15.203  < 2e-16 ***
## bathroomsCont       -29538       1977 -14.942  < 2e-16 ***
## conditionCatcnd_2    12350      17768   0.695   0.4870
## conditionCatcnd_3   117552      18688   6.290 3.25e-10 ***
## longBin              49841       3849  12.950  < 2e-16 ***
## renovated           112930       8452  13.362  < 2e-16 ***
## floorsCatflr_2      -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3      144356      19378   7.449 9.88e-14 ***
## bedroomsCont        -18306       2194  -8.345  < 2e-16 ***
## sqft_lot15          -14755       2475  -5.961 2.56e-09 ***
## sqft_living15        14212       2829   5.023 5.14e-07 ***
## sqft_above          -43326       6420  -6.749 1.55e-11 ***
## basementCat1        -45880       8144  -5.633 1.80e-08 ***
```

```
## sqft_lot              5779       2419   2.389   0.0169 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205800 on 15109 degrees of freedom
## Multiple R-squared:  0.6729, Adjusted R-squared:  0.6725
## F-statistic:  1636 on 19 and 15109 DF,  p-value: < 2.2e-16

# Residuals not great
plot(fwd, which = 2)
```



lm(price ~ sqft_living + latBin + gradeCont + waterfront + viewCat + ba

```
### Backward Stempwise ###
bck <-
step(lmp,scope=list(lower=lm0,upper=lmp),direction="backward",k=2,trace=0)
summary(bck)

##
## Call:
## lm(formula = price ~ sqft_living + sqft_lot + sqft_above + basementCat +
##     sqft_living15 + sqft_lot15 + renovated + latBin + longBin +
##     waterfront + bedroomsCont + bathroomsCont + gradeCont + floorsCat +
##     viewCat + conditionCat, data = HousingRed[train, c("price",
##     predictors)])
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1130252  -112318   -15645    90371  4336770
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         418146      18098  23.104  < 2e-16 ***
## sqft_living         199147       7098  28.059  < 2e-16 ***
## sqft_lot              5779       2419   2.389   0.0169 *
## sqft_above          -43326       6420  -6.749 1.55e-11 ***
## basementCat1        -45880       8144  -5.633 1.80e-08 ***
## sqft_living15        14212       2829   5.023 5.14e-07 ***
## sqft_lot15          -14755       2475  -5.961 2.56e-09 ***
## renovated           112930       8452  13.362  < 2e-16 ***
## latBin              186726       3875  48.189  < 2e-16 ***
## longBin              49841       3849  12.950  < 2e-16 ***
## waterfront1         522723      20884  25.030  < 2e-16 ***
## bedroomsCont        -18306       2194  -8.345  < 2e-16 ***
## bathroomsCont       -29538       1977 -14.942  < 2e-16 ***
## gradeCont           105239       3025  34.792  < 2e-16 ***
## floorsCatflr_2      -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3      144356      19378   7.449 9.88e-14 ***
## viewCatview_1        86085       7356  11.702  < 2e-16 ***
## viewCatview_2       153218      10078  15.203  < 2e-16 ***
## conditionCatcnd_2    12350      17768   0.695   0.4870
## conditionCatcnd_3   117552      18688   6.290 3.25e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205800 on 15109 degrees of freedom
## Multiple R-squared:  0.6729, Adjusted R-squared:  0.6725
## F-statistic:  1636 on 19 and 15109 DF,  p-value: < 2.2e-16

# Same model

### Both ###
hyb <- step(lmp,scope=list(lower=lm0,upper=lmp),direction="both",k=2,trace=0)
summary(hyb)

##
## Call:
## lm(formula = price ~ sqft_living + sqft_lot + sqft_above + basementCat +
##     sqft_living15 + sqft_lot15 + renovated + latBin + longBin +
##     waterfront + bedroomsCont + bathroomsCont + gradeCont + floorsCat +
##     viewCat + conditionCat, data = HousingRed[train, c("price",
##     predictors)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1130252  -112318   -15645    90371  4336770
```

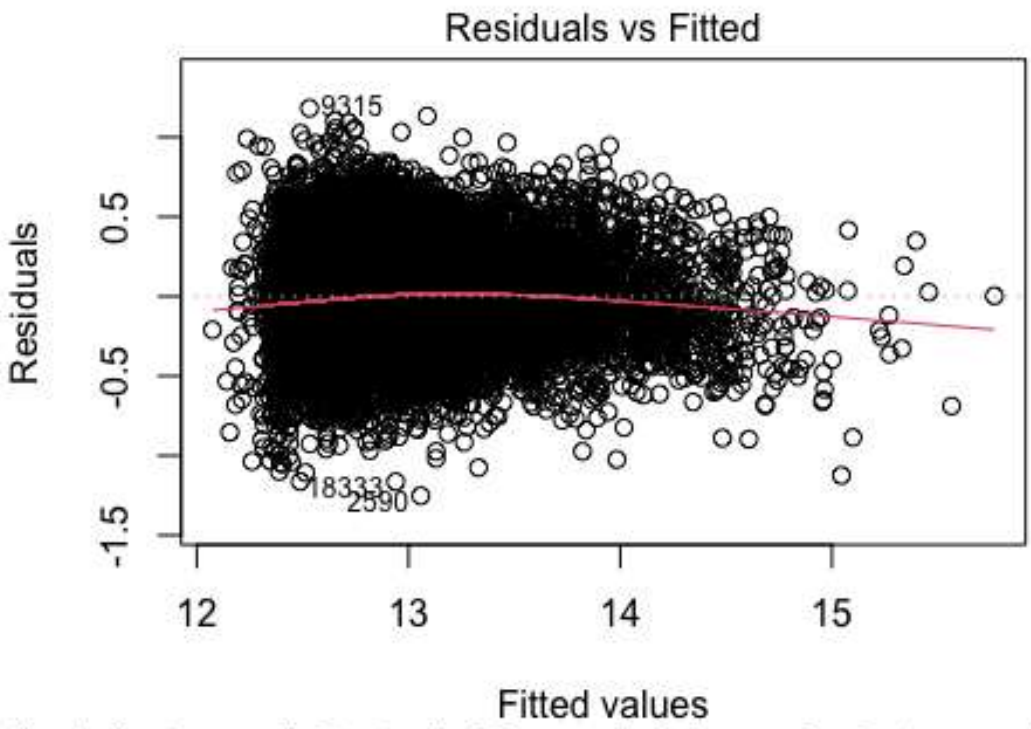
```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         418146      18098  23.104  < 2e-16 ***
## sqft_living         199147       7098  28.059  < 2e-16 ***
## sqft_lot              5779       2419   2.389   0.0169 *
## sqft_above          -43326       6420  -6.749 1.55e-11 ***
## basementCat1        -45880       8144  -5.633 1.80e-08 ***
## sqft_living15        14212       2829   5.023 5.14e-07 ***
## sqft_lot15          -14755       2475  -5.961 2.56e-09 ***
## renovated           112930       8452  13.362  < 2e-16 ***
## latBin              186726       3875  48.189  < 2e-16 ***
## longBin              49841       3849  12.950  < 2e-16 ***
## waterfront1         522723      20884  25.030  < 2e-16 ***
## bedroomsCont        -18306       2194  -8.345  < 2e-16 ***
## bathroomsCont       -29538       1977 -14.942  < 2e-16 ***
## gradeCont           105239       3025  34.792  < 2e-16 ***
## floorsCatflr_2      -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3      144356      19378   7.449 9.88e-14 ***
## viewCatview_1        86085       7356  11.702  < 2e-16 ***
## viewCatview_2       153218      10078  15.203  < 2e-16 ***
## conditionCatcnd_2    12350      17768   0.695   0.4870
## conditionCatcnd_3   117552      18688   6.290 3.25e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205800 on 15109 degrees of freedom
## Multiple R-squared:  0.6729, Adjusted R-squared:  0.6725
## F-statistic:  1636 on 19 and 15109 DF,  p-value: < 2.2e-16
```

```
# Same model
```

```
### Try modeling Ln(Price)

### Forward Stempwise ###
lmpln <- lm(log(price) ~ ., data = HousingRed[train,c('price',predictors)])
lm0ln <- lm(log(price) ~ 1, data = HousingRed[train,c('price',predictors)])
fwdln <- step(lm0ln, scope = list(lower = lm0ln, upper = lmpln), direction =
"forward", k = 2, trace = 0)
summary(fwdln)
```

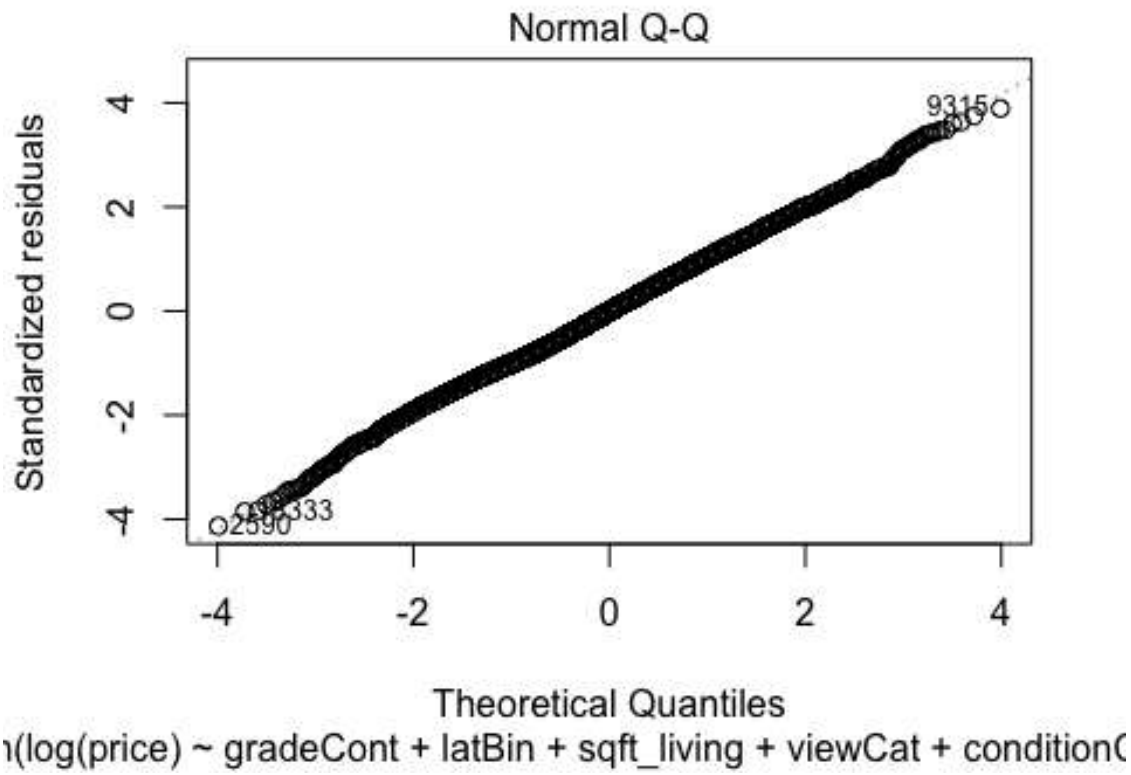
```
##
## Call:
## lm(formula = log(price) ~ gradeCont + latBin + sqft_living +
##     viewCat + conditionCat + renovated + sqft_living15 + waterfront +
##     sqft_above + floorsCat + longBin + sqft_lot + sqft_lot15 +
##     basementCat + bathroomsCont + bedroomsCont, data = HousingRed[train,
##     c("price", predictors)])
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## gradeCont          0.169536   0.004459  38.018  < 2e-16 ***
## latBin             0.293510   0.005713  51.379  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## viewCatview_1      0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2      0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2  0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3  0.362794   0.027550  13.168  < 2e-16 ***
## renovated          0.172885   0.012460  13.875  < 2e-16 ***
## sqft_living15      0.065507   0.004171  15.704  < 2e-16 ***
## waterfront1        0.358696   0.030788  11.651  < 2e-16 ***
## sqft_above        -0.098270   0.009465 -10.383  < 2e-16 ***
## floorsCatflr_2     0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3     0.215870   0.028568   7.556 4.39e-14 ***
## longBin            0.038033   0.005674   6.703 2.12e-11 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
## sqft_lot15        -0.017611   0.003649  -4.826 1.40e-06 ***
## basementCat1      -0.028493   0.012007  -2.373   0.0177 *
## bathroomsCont      0.008092   0.002914   2.776   0.0055 **
## bedroomsCont      -0.007710   0.003234  -2.384   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16

# Residuals are better, but definitely not heteroscedastic
plot(fwdln, which = 1)
```



ı(log(price) ~ gradeCont + latBin + sqft_living + viewCat + conditionC

```
plot(fwdln, which = 2)
```



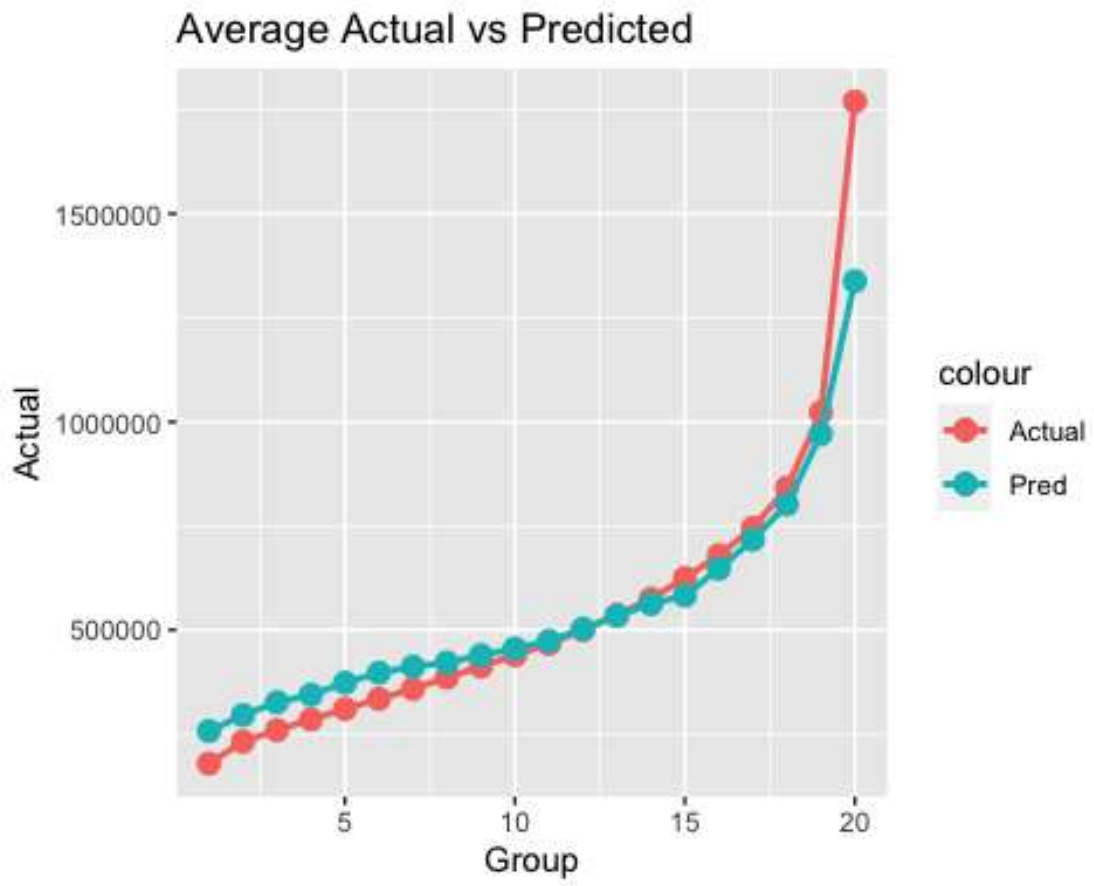
)(log(price) ~ gradeCont + latBin + sqft_living + viewCat + condition(

```
### Backward Stempwise ###
bckln <-
step(lmpln,scope=list(lower=lm0ln,upper=lmpln),direction="backward",k=2,trace
=0)
summary(bckln)

##
## Call:
## lm(formula = log(price) ~ sqft_living + sqft_lot + sqft_above +
##     basementCat + sqft_living15 + sqft_lot15 + renovated + latBin +
##     longBin + waterfront + bedroomsCont + bathroomsCont + gradeCont +
##     floorsCat + viewCat + conditionCat, data = HousingRed[train,
##     c("price", predictors)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
```

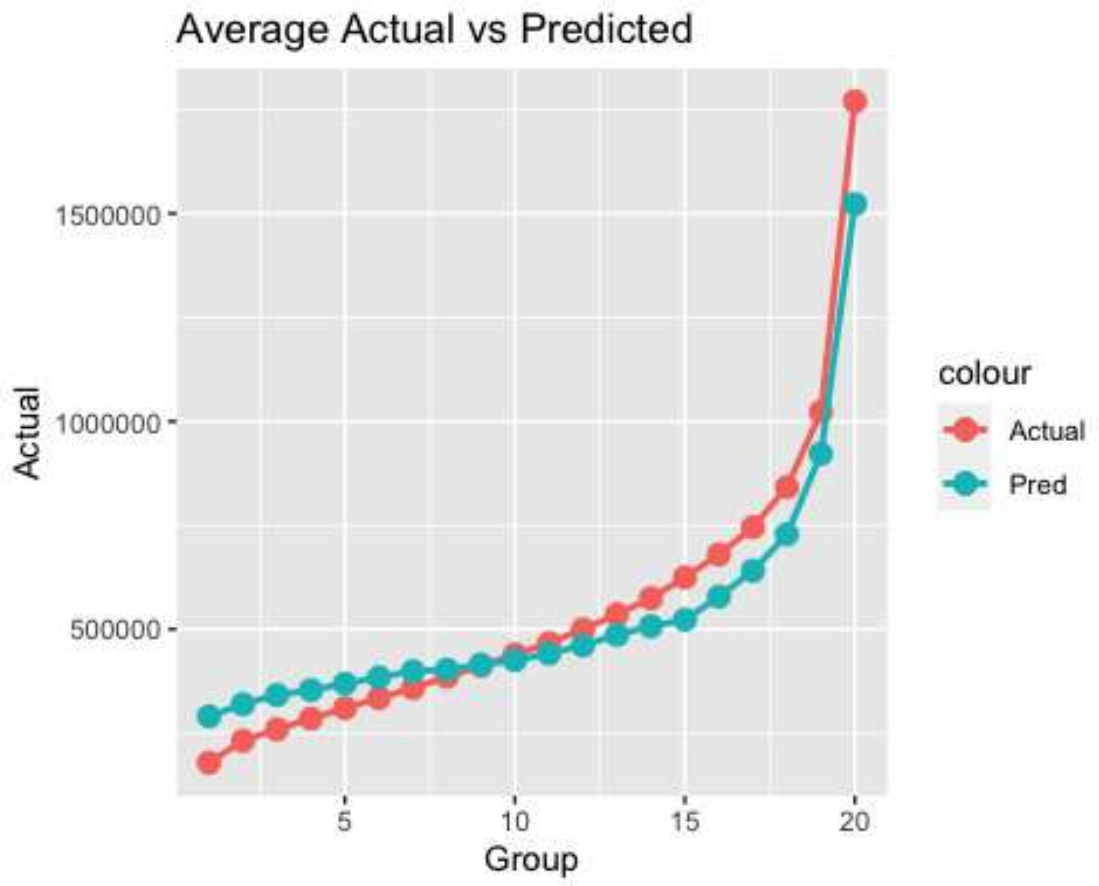
```
## sqft_above         -0.098270   0.009465 -10.383  < 2e-16 ***
## basementCat1       -0.028493   0.012007  -2.373   0.0177 *
## sqft_living15       0.065507   0.004171  15.704  < 2e-16 ***
## sqft_lot15         -0.017611   0.003649  -4.826 1.40e-06 ***
## renovated           0.172885   0.012460  13.875  < 2e-16 ***
## latBin              0.293510   0.005713  51.379  < 2e-16 ***
## longBin             0.038033   0.005674   6.703 2.12e-11 ***
## waterfront1         0.358696   0.030788  11.651  < 2e-16 ***
## bedroomsCont       -0.007710   0.003234  -2.384   0.0171 *
## bathroomsCont       0.008092   0.002914   2.776   0.0055 **
## gradeCont           0.169536   0.004459  38.018  < 2e-16 ***
## floorsCatflr_2      0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3      0.215870   0.028568   7.556 4.39e-14 ***
## viewCatview_1       0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2       0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2   0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3   0.362794   0.027550  13.168  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16

# Same model again

### Both ###
hybln <-
step(lmpln,scope=list(lower=lm0ln,upper=lmpln),direction="both",k=2,trace=0)
summary(hybln)

##
## Call:
## lm(formula = log(price) ~ sqft_living + sqft_lot + sqft_above +
##     basementCat + sqft_living15 + sqft_lot15 + renovated + latBin +
##     longBin + waterfront + bedroomsCont + bathroomsCont + gradeCont +
##     floorsCat + viewCat + conditionCat, data = HousingRed[train,
##     c("price", predictors)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
## sqft_above        -0.098270   0.009465 -10.383  < 2e-16 ***
## basementCat1      -0.028493   0.012007  -2.373   0.0177 *
```

```
## sqft_living15      0.065507   0.004171  15.704  < 2e-16 ***
## sqft_lot15        -0.017611   0.003649  -4.826 1.40e-06 ***
## renovated          0.172885   0.012460  13.875  < 2e-16 ***
## latBin             0.293510   0.005713  51.379  < 2e-16 ***
## longBin            0.038033   0.005674   6.703 2.12e-11 ***
## waterfront1        0.358696   0.030788  11.651  < 2e-16 ***
## bedroomsCont      -0.007710   0.003234  -2.384   0.0171 *
## bathroomsCont      0.008092   0.002914   2.776   0.0055 **
## gradeCont          0.169536   0.004459  38.018  < 2e-16 ***
## floorsCatflr_2     0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3     0.215870   0.028568   7.556 4.39e-14 ***
## viewCatview_1      0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2      0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2  0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3  0.362794   0.027550  13.168  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16
```

```
# Same model again

lm0.pred   <- predict(lm0, HousingRed[test,])
lmp.pred   <- predict(lmp, HousingRed[test,])
fwd.pred   <- predict(fwd, HousingRed[test,])
fwdln.pred <- exp(predict(fwdln, HousingRed[test,]))


# RMSE, intercept only, saturated, step, step (ln(y))
c(sqrt(mean((lm0.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((lmp.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((fwd.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((fwdln.pred - HousingRed[test,'price'])^2)))

## [1] 383969.3 215913.9 215913.9 231242.1

rankedPrediction(fwd.pred, HousingRed[test,'price'])
```

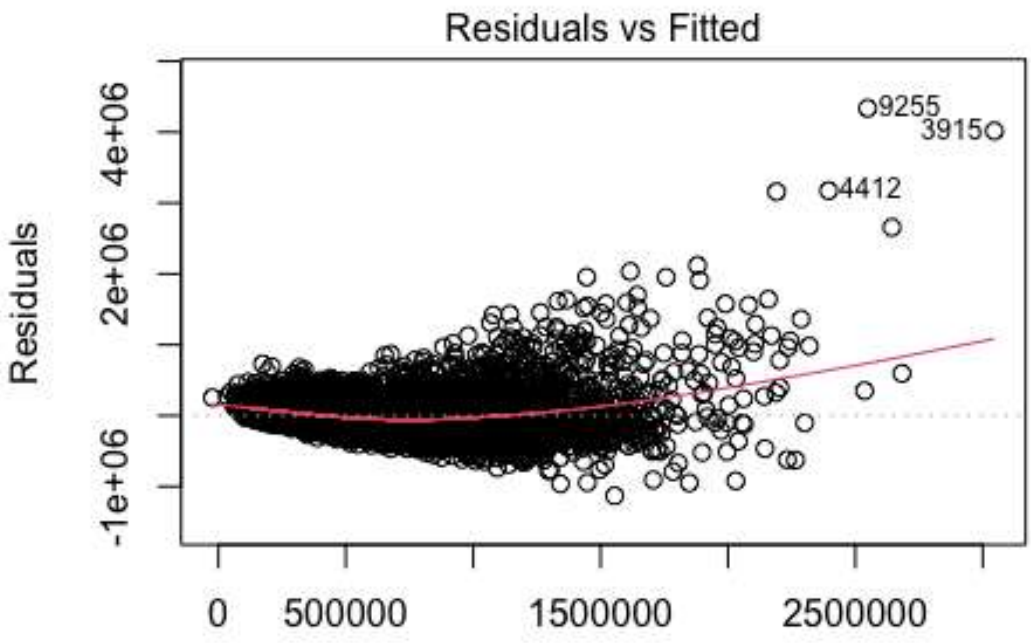



```
rankedPrediction(fwdln.pred, HousingRed[test,'price'])
```



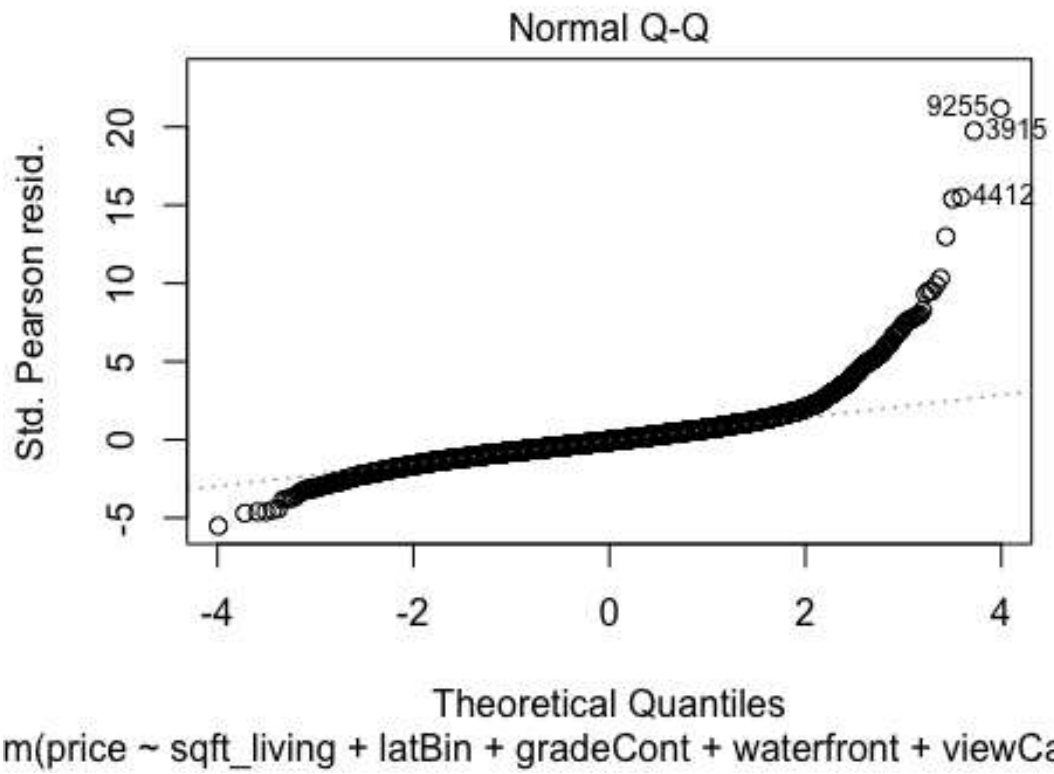

```
## ------------


### Forward Step-wise ###
lmp <- glm(price ~ ., data = HousingRed[train,c('price',predictors)], family
= gaussian)
lm0 <- glm(price ~ 1, data = HousingRed[train,c('price',predictors)], family
= gaussian)
fwd <- step(lm0, scope = list(lower = lm0, upper = lmp), direction =
"forward", k = 2, trace = 0)
summary(fwd)

##
## Call:
## glm(formula = price ~ sqft_living + latBin + gradeCont + waterfront +
##     viewCat + bathroomsCont + conditionCat + longBin + renovated +
##     floorsCat + bedroomsCont + sqft_lot15 + sqft_living15 + sqft_above +
##     basementCat + sqft_lot, family = gaussian, data = HousingRed[train,
##     c("price", predictors)])
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -1130252    -112318     -15645      90371    4336770
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         418146      18098  23.104  < 2e-16 ***
## sqft_living         199147       7098  28.059  < 2e-16 ***
## latBin              186726       3875  48.189  < 2e-16 ***
## gradeCont           105239       3025  34.792  < 2e-16 ***
## waterfront1         522723      20884  25.030  < 2e-16 ***
## viewCatview_1        86085       7356  11.702  < 2e-16 ***
## viewCatview_2       153218      10078  15.203  < 2e-16 ***
## bathroomsCont       -29538       1977 -14.942  < 2e-16 ***
## conditionCatcnd_2    12350      17768   0.695   0.4870
## conditionCatcnd_3   117552      18688   6.290 3.25e-10 ***
## longBin              49841       3849  12.950  < 2e-16 ***
## renovated           112930       8452  13.362  < 2e-16 ***
## floorsCatflr_2      -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3      144356      19378   7.449 9.88e-14 ***
## bedroomsCont        -18306       2194  -8.345  < 2e-16 ***
## sqft_lot15          -14755       2475  -5.961 2.56e-09 ***
## sqft_living15        14212       2829   5.023 5.14e-07 ***
## sqft_above          -43326       6420  -6.749 1.55e-11 ***
## basementCat1        -45880       8144  -5.633 1.80e-08 ***
## sqft_lot              5779       2419   2.389   0.0169 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 42372730359)
##
##     Null deviance: 1.9572e+15  on 15128  degrees of freedom
## Residual deviance: 6.4021e+14  on 15109  degrees of freedom
## AIC: 413159
##
## Number of Fisher Scoring iterations: 2

# Residuals not great
plot(fwd, which = 1)
```



m(price ~ sqft_living + latBin + gradeCont + waterfront + viewCat + l

```
plot(fwd, which = 2)
```



m(price ~ sqft_living + latBin + gradeCont + waterfront + viewCat + t

```
### Backward Stempwise ###
bck <-
step(lmp,scope=list(lower=lm0,upper=lmp),direction="backward",k=2,trace=0)
summary(bck)

##
## Call:
## glm(formula = price ~ sqft_living + sqft_lot + sqft_above + basementCat +
##     sqft_living15 + sqft_lot15 + renovated + latBin + longBin +
##     waterfront + bedroomsCont + bathroomsCont + gradeCont + floorsCat +
##     viewCat + conditionCat, family = gaussian, data = HousingRed[train,
##     c("price", predictors)])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1130252   -112318    -15645     90371   4336770
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        418146      18098  23.104  < 2e-16 ***
## sqft_living        199147       7098  28.059  < 2e-16 ***
## sqft_lot             5779       2419   2.389   0.0169 *
## sqft_above         -43326       6420  -6.749 1.55e-11 ***
```

```
## basementCat1         -45880       8144  -5.633 1.80e-08 ***
## sqft_living15         14212       2829   5.023 5.14e-07 ***
## sqft_lot15           -14755       2475  -5.961 2.56e-09 ***
## renovated            112930       8452  13.362  < 2e-16 ***
## latBin               186726       3875  48.189  < 2e-16 ***
## longBin               49841       3849  12.950  < 2e-16 ***
## waterfront1          522723      20884  25.030  < 2e-16 ***
## bedroomsCont         -18306       2194  -8.345  < 2e-16 ***
## bathroomsCont        -29538       1977 -14.942  < 2e-16 ***
## gradeCont            105239       3025  34.792  < 2e-16 ***
## floorsCatflr_2       -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3       144356      19378   7.449 9.88e-14 ***
## viewCatview_1         86085       7356  11.702  < 2e-16 ***
## viewCatview_2        153218      10078  15.203  < 2e-16 ***
## conditionCatcnd_2     12350      17768   0.695   0.4870
## conditionCatcnd_3    117552      18688   6.290 3.25e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 42372730359)
##
##     Null deviance: 1.9572e+15  on 15128  degrees of freedom
## Residual deviance: 6.4021e+14  on 15109  degrees of freedom
## AIC: 413159
##
## Number of Fisher Scoring iterations: 2

# Same model

### Both ###
hyb <- step(lmp,scope=list(lower=lm0,upper=lmp),direction="both",k=2,trace=0)
summary(hyb)

##
## Call:
## glm(formula = price ~ sqft_living + sqft_lot + sqft_above + basementCat +
##     sqft_living15 + sqft_lot15 + renovated + latBin + longBin +
##     waterfront + bedroomsCont + bathroomsCont + gradeCont + floorsCat +
##     viewCat + conditionCat, family = gaussian, data = HousingRed[train,
##     c("price", predictors)])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1130252   -112318    -15645     90371   4336770
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         418146      18098  23.104  < 2e-16 ***
## sqft_living         199147       7098  28.059  < 2e-16 ***
## sqft_lot              5779       2419   2.389   0.0169 *
```

```
## sqft_above          -43326       6420  -6.749 1.55e-11 ***
## basementCat1        -45880       8144  -5.633 1.80e-08 ***
## sqft_living15        14212       2829   5.023 5.14e-07 ***
## sqft_lot15          -14755       2475  -5.961 2.56e-09 ***
## renovated           112930       8452  13.362  < 2e-16 ***
## latBin              186726       3875  48.189  < 2e-16 ***
## longBin              49841       3849  12.950  < 2e-16 ***
## waterfront1         522723      20884  25.030  < 2e-16 ***
## bedroomsCont        -18306       2194  -8.345  < 2e-16 ***
## bathroomsCont       -29538       1977 -14.942  < 2e-16 ***
## gradeCont           105239       3025  34.792  < 2e-16 ***
## floorsCatflr_2      -10210       4284  -2.383   0.0172 *
## floorsCatfrl_3      144356      19378   7.449 9.88e-14 ***
## viewCatview_1        86085       7356  11.702  < 2e-16 ***
## viewCatview_2       153218      10078  15.203  < 2e-16 ***
## conditionCatcnd_2    12350      17768   0.695   0.4870
## conditionCatcnd_3   117552      18688   6.290 3.25e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 42372730359)
##
##     Null deviance: 1.9572e+15  on 15128  degrees of freedom
## Residual deviance: 6.4021e+14  on 15109  degrees of freedom
## AIC: 413159
##
## Number of Fisher Scoring iterations: 2
```

```
# Same model
```

```
### Try modeling Ln(Price)

### Forward Stempwise ###
lmpln <- lm(log(price) ~ ., data = HousingRed[train,c('price',predictors)],
family = gaussian)

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

lm0ln <- lm(log(price) ~ 1, data = HousingRed[train,c('price',predictors)],
family = gaussian)

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

fwdln <- step(lm0ln, scope = list(lower = lm0ln, upper = lmpln), direction =
"forward", k = 2, trace = 0)
```

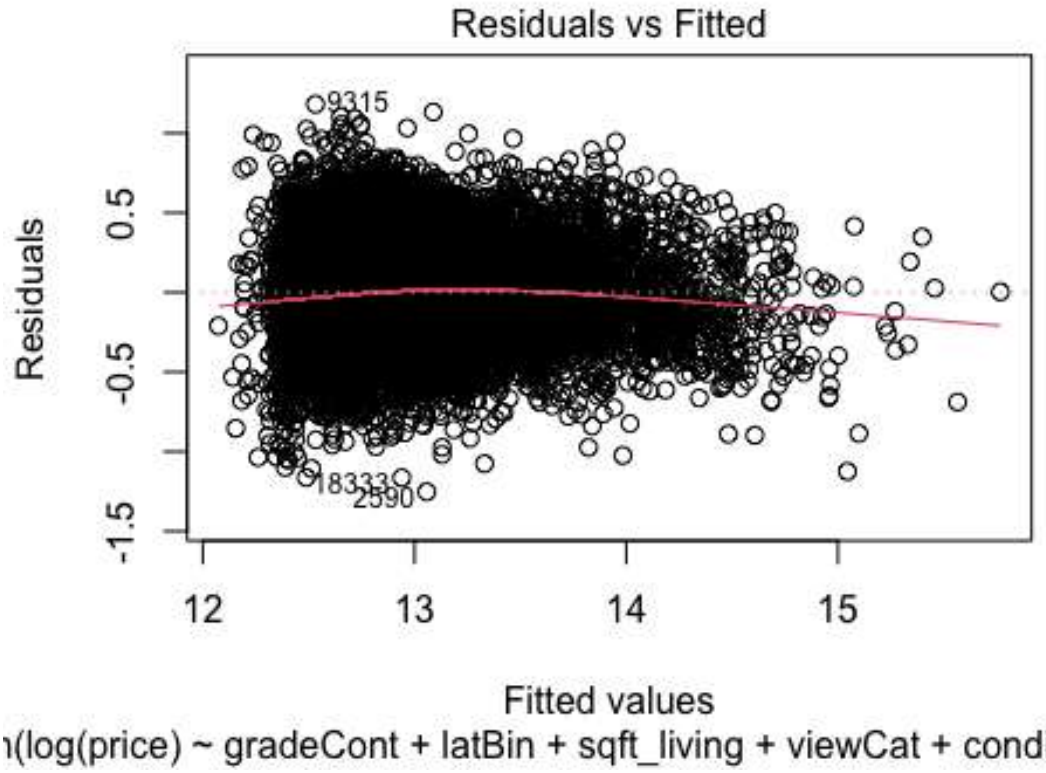
```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
```

```
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
:
##  extra argument 'family' will be disregarded
```

```
summary(fwdln)
```

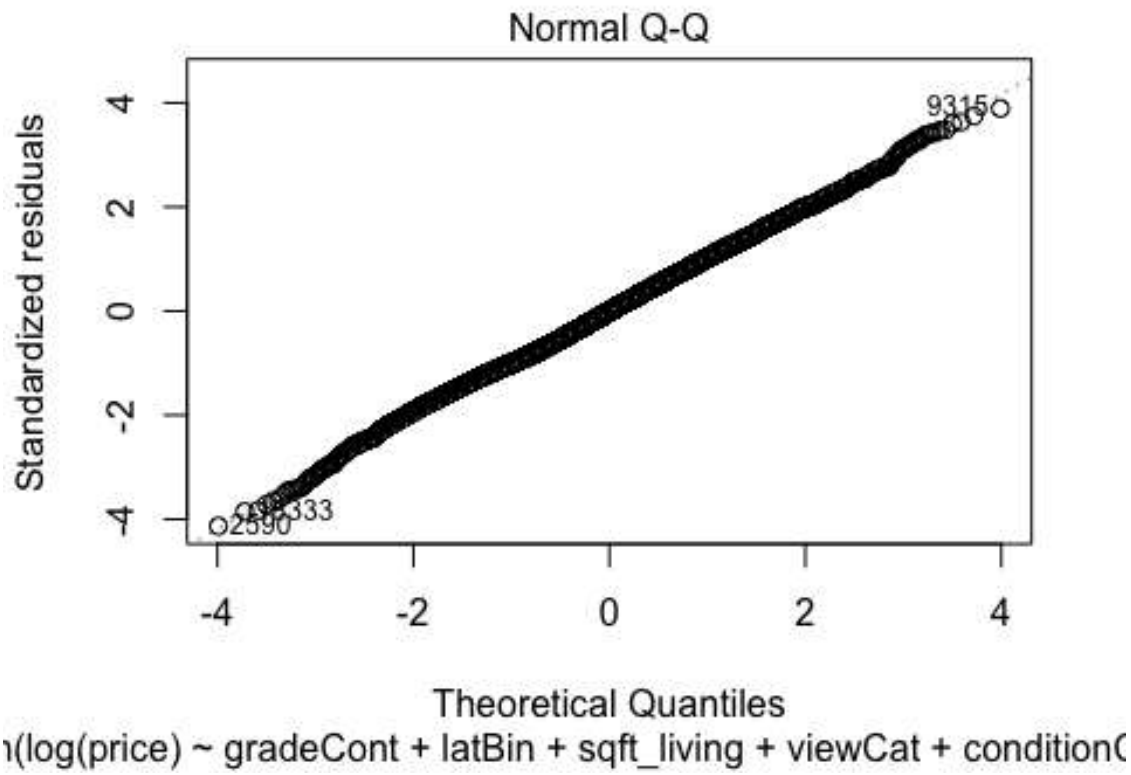
```
##
## Call:
## lm(formula = log(price) ~ gradeCont + latBin + sqft_living +
##     viewCat + conditionCat + renovated + sqft_living15 + waterfront +
##     sqft_above + floorsCat + longBin + sqft_lot + sqft_lot15 +
##     basementCat + bathroomsCont + bedroomsCont, data = HousingRed[train,
##     c("price", predictors)], family = gaussian)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## gradeCont          0.169536   0.004459  38.018  < 2e-16 ***
## latBin             0.293510   0.005713  51.379  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## viewCatview_1      0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2      0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2  0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3  0.362794   0.027550  13.168  < 2e-16 ***
## renovated          0.172885   0.012460  13.875  < 2e-16 ***
## sqft_living15      0.065507   0.004171  15.704  < 2e-16 ***
## waterfront1        0.358696   0.030788  11.651  < 2e-16 ***
## sqft_above        -0.098270   0.009465 -10.383  < 2e-16 ***
## floorsCatflr_2     0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3     0.215870   0.028568   7.556 4.39e-14 ***
## longBin            0.038033   0.005674   6.703 2.12e-11 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
## sqft_lot15        -0.017611   0.003649  -4.826 1.40e-06 ***
## basementCat1      -0.028493   0.012007  -2.373   0.0177 *
## bathroomsCont      0.008092   0.002914   2.776   0.0055 **
```

```
## bedroomsCont      -0.007710   0.003234  -2.384   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16

# Residuals are better, but definitely not heteroscedastic
plot(fwdln, which = 1)
```




```
ı(log(price) ~ gradeCont + latBin + sqft_living + viewCat + conditionC
```

```
plot(fwdln, which = 2)
```

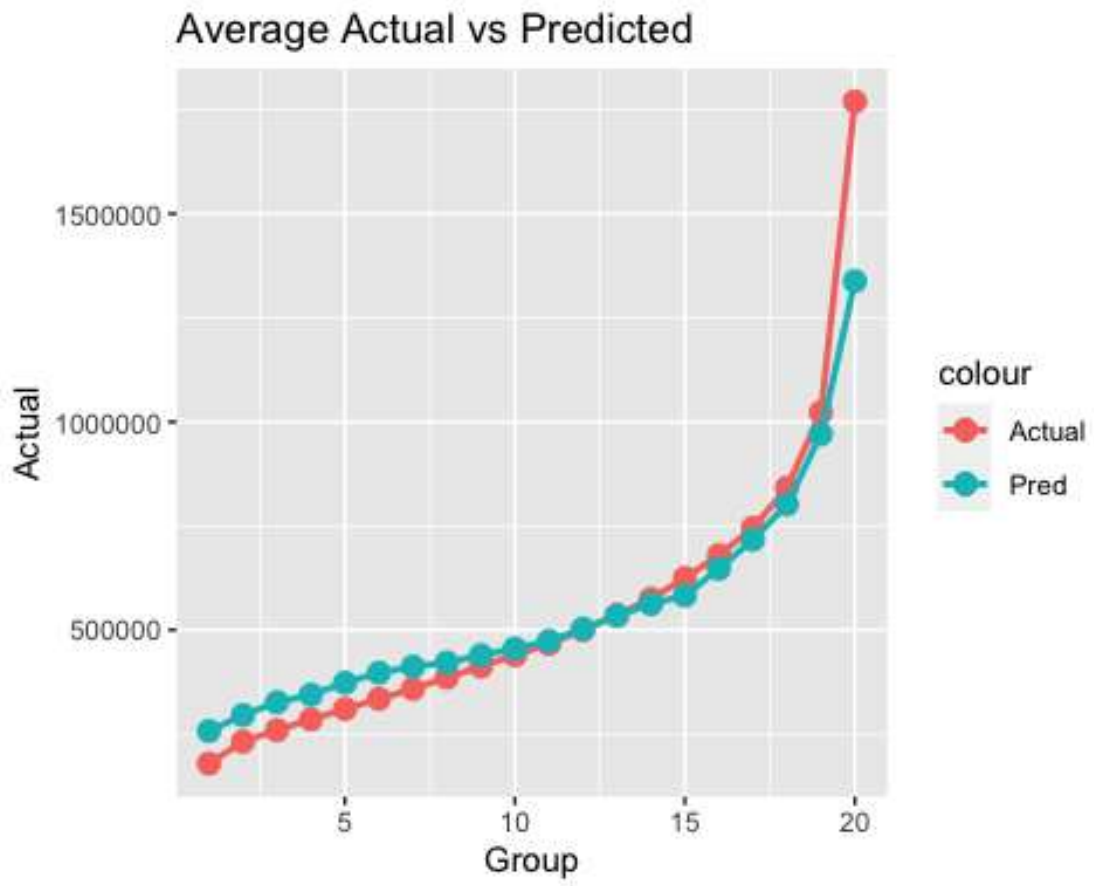


)(log(price) ~ gradeCont + latBin + sqft_living + viewCat + condition(

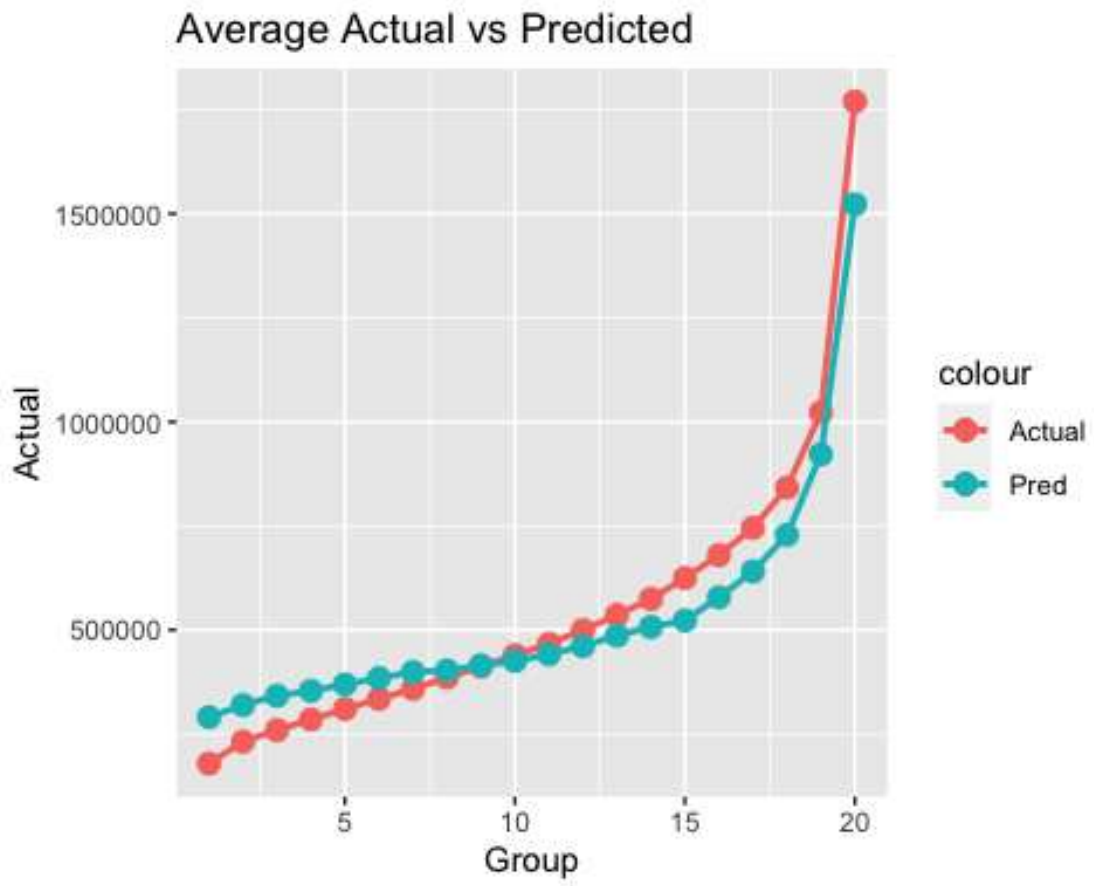
```
### Backward Stempwise ###
bckln <-
step(lmpln,scope=list(lower=lm0ln,upper=lmpln),direction="backward",k=2,trace
=0)
summary(bckln)

##
## Call:
## lm(formula = log(price) ~ sqft_living + sqft_lot + sqft_above +
##     basementCat + sqft_living15 + sqft_lot15 + renovated + latBin +
##     longBin + waterfront + bedroomsCont + bathroomsCont + gradeCont +
##     floorsCat + viewCat + conditionCat, data = HousingRed[train,
##     c("price", predictors)], family = gaussian)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
```

```
## sqft_above         -0.098270   0.009465 -10.383  < 2e-16 ***
## basementCat1       -0.028493   0.012007  -2.373   0.0177 *
## sqft_living15       0.065507   0.004171  15.704  < 2e-16 ***
## sqft_lot15         -0.017611   0.003649  -4.826 1.40e-06 ***
## renovated           0.172885   0.012460  13.875  < 2e-16 ***
## latBin              0.293510   0.005713  51.379  < 2e-16 ***
## longBin             0.038033   0.005674   6.703 2.12e-11 ***
## waterfront1         0.358696   0.030788  11.651  < 2e-16 ***
## bedroomsCont       -0.007710   0.003234  -2.384   0.0171 *
## bathroomsCont       0.008092   0.002914   2.776   0.0055 **
## gradeCont           0.169536   0.004459  38.018  < 2e-16 ***
## floorsCatflr_2      0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3      0.215870   0.028568   7.556 4.39e-14 ***
## viewCatview_1       0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2       0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2   0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3   0.362794   0.027550  13.168  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16

# Same model again

### Both ###
hybln <-
step(lmpln,scope=list(lower=lm0ln,upper=lmpln),direction="both",k=2,trace=0)
summary(hybln)

##
## Call:
## lm(formula = log(price) ~ sqft_living + sqft_lot + sqft_above +
##     basementCat + sqft_living15 + sqft_lot15 + renovated + latBin +
##     longBin + waterfront + bedroomsCont + bathroomsCont + gradeCont +
##     floorsCat + viewCat + conditionCat, data = HousingRed[train,
##     c("price", predictors)], family = gaussian)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25208 -0.21763 -0.00272  0.20993  1.18117
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.673376   0.026681 474.988  < 2e-16 ***
## sqft_living        0.219274   0.010464  20.956  < 2e-16 ***
## sqft_lot           0.019076   0.003566   5.349 8.98e-08 ***
## sqft_above        -0.098270   0.009465 -10.383  < 2e-16 ***
## basementCat1      -0.028493   0.012007  -2.373   0.0177 *
```

```
## sqft_living15      0.065507   0.004171  15.704  < 2e-16 ***
## sqft_lot15        -0.017611   0.003649  -4.826 1.40e-06 ***
## renovated          0.172885   0.012460  13.875  < 2e-16 ***
## latBin             0.293510   0.005713  51.379  < 2e-16 ***
## longBin            0.038033   0.005674   6.703 2.12e-11 ***
## waterfront1        0.358696   0.030788  11.651  < 2e-16 ***
## bedroomsCont      -0.007710   0.003234  -2.384   0.0171 *
## bathroomsCont      0.008092   0.002914   2.776   0.0055 **
## gradeCont          0.169536   0.004459  38.018  < 2e-16 ***
## floorsCatflr_2     0.061480   0.006316   9.734  < 2e-16 ***
## floorsCatfrl_3     0.215870   0.028568   7.556 4.39e-14 ***
## viewCatview_1      0.129796   0.010845  11.968  < 2e-16 ***
## viewCatview_2      0.136116   0.014858   9.161  < 2e-16 ***
## conditionCatcnd_2  0.196985   0.026195   7.520 5.79e-14 ***
## conditionCatcnd_3  0.362794   0.027550  13.168  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3035 on 15109 degrees of freedom
## Multiple R-squared:  0.665,  Adjusted R-squared:  0.6646
## F-statistic:  1579 on 19 and 15109 DF,  p-value: < 2.2e-16

# Same model again

lm0.pred   <- predict(lm0, HousingRed[test,])
lmp.pred   <- predict(lmp, HousingRed[test,])
fwd.pred   <- predict(fwd, HousingRed[test,])
fwdln.pred <- exp(predict(fwdln, HousingRed[test,]))

# RMSE, intercept only, saturated, step, step (ln(y))
c(sqrt(mean((lm0.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((lmp.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((fwd.pred - HousingRed[test,'price'])^2)),
  sqrt(mean((fwdln.pred - HousingRed[test,'price'])^2)))

## [1] 383969.3 215913.9 215913.9 231242.1

rankedPrediction(fwd.pred, HousingRed[test,'price'])
```




```
rankedPrediction(fwdln.pred, HousingRed[test,'price'])
```


Average Actual vs Predicted
1500000
1000000
500000
Actual
5
10
15
20
Group
colour
Actual
Pred