MetropolitanHousingPropertiesCoding

PrabhTalwar

2024-01-31

Appendix

```
#reading the data
housing <- AmesHousing::make_ames()
```

```
#top few rows of the data
head(housing,3)
```

```
## # A tibble: 3 x 81
##   MS_SubClass             MS_Zoning Lot_Frontage Lot_Area Street Alley Lot_Shape
##   <fct>                   <fct>            <dbl>    <int> <fct>  <fct> <fct>
## 1 One_Story_1946_and_New~ Resident~          141    31770 Pave   No_A~ Slightly~
## 2 One_Story_1946_and_New~ Resident~           80    11622 Pave   No_A~ Regular
## 3 One_Story_1946_and_New~ Resident~           81    14267 Pave   No_A~ Slightly~
## # i 74 more variables: Land_Contour <fct>, Utilities <fct>, Lot_Config <fct>,
## #   Land_Slope <fct>, Neighborhood <fct>, Condition_1 <fct>, Condition_2 <fct>,
## #   Bldg_Type <fct>, House_Style <fct>, Overall_Qual <fct>, Overall_Cond <fct>,
## #   Year_Built <int>, Year_Remod_Add <int>, Roof_Style <fct>, Roof_Matl <fct>,
## #   Exterior_1st <fct>, Exterior_2nd <fct>, Mas_Vnr_Type <fct>,
## #   Mas_Vnr_Area <dbl>, Exter_Qual <fct>, Exter_Cond <fct>, Foundation <fct>,
## #   Bsmt_Qual <fct>, Bsmt_Cond <fct>, Bsmt_Exposure <fct>, ...
```

Glimpse of the data

Here, we see the glimpse of our data. Some of the variables are factors and some are numerical.

```
glimpse(housing)
```

```
## Rows: 2,930
## Columns: 81
## $ MS_SubClass        <fct> One_Story_1946_and_Newer_All_Styles, One_Story_1946~
## $ MS_Zoning          <fct> Residential_Low_Density, Residential_High_Density, ~
## $ Lot_Frontage       <dbl> 141, 80, 81, 93, 74, 78, 41, 43, 39, 60, 75, 0, 63,~
## $ Lot_Area           <int> 31770, 11622, 14267, 11160, 13830, 9978, 4920, 5005~
## $ Street             <fct> Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pav~
## $ Alley              <fct> No_Alley_Access, No_Alley_Access, No_Alley_Access, ~
## $ Lot_Shape          <fct> Slightly_Irregular, Regular, Slightly_Irregular, Re~
## $ Land_Contour       <fct> Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, HLS, Lvl, Lvl, L~
```

```
## $ Utilities          <fct> AllPub, AllPub, AllPub, AllPub, AllPub, AllPub, All~
## $ Lot_Config         <fct> Corner, Inside, Corner, Corner, Inside, Inside, Ins~
## $ Land_Slope         <fct> Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, G~
## $ Neighborhood       <fct> North_Ames, North_Ames, North_Ames, North_Ames, Gil~
## $ Condition_1        <fct> Norm, Feedr, Norm, Norm, Norm, Norm, Norm, Norm, No~
## $ Condition_2        <fct> Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm, Nor~
## $ Bldg_Type          <fct> OneFam, OneFam, OneFam, OneFam, OneFam, OneFam, Twn~
## $ House_Style        <fct> One_Story, One_Story, One_Story, One_Story, Two_Sto~
## $ Overall_Qual       <fct> Above_Average, Average, Above_Average, Good, Averag~
## $ Overall_Cond       <fct> Average, Above_Average, Above_Average, Average, Ave~
## $ Year_Built         <int> 1960, 1961, 1958, 1968, 1997, 1998, 2001, 1992, 199~
## $ Year_Remod_Add     <int> 1960, 1961, 1958, 1968, 1998, 1998, 2001, 1992, 199~
## $ Roof_Style         <fct> Hip, Gable, Hip, Hip, Gable, Gable, Gable, Gable, G~
## $ Roof_Matl          <fct> CompShg, CompShg, CompShg, CompShg, CompShg, CompSh~
## $ Exterior_1st       <fct> BrkFace, VinylSd, Wd Sdng, BrkFace, VinylSd, VinylS~
## $ Exterior_2nd       <fct> Plywood, VinylSd, Wd Sdng, BrkFace, VinylSd, VinylS~
## $ Mas_Vnr_Type       <fct> Stone, None, BrkFace, None, None, BrkFace, None, No~
## $ Mas_Vnr_Area       <dbl> 112, 0, 108, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6~
## $ Exter_Qual         <fct> Typical, Typical, Typical, Good, Typical, Typical, ~
## $ Exter_Cond         <fct> Typical, Typical, Typical, Typical, Typical, Typica~
## $ Foundation         <fct> CBlock, CBlock, CBlock, CBlock, PConc, PConc, PConc~
## $ Bsmt_Qual          <fct> Typical, Typical, Typical, Typical, Good, Typical, ~
## $ Bsmt_Cond          <fct> Good, Typical, Typical, Typical, Typical, Typical, ~
## $ Bsmt_Exposure      <fct> Gd, No, No, No, No, No, Mn, No, No, No, No, No, No,~
## $ BsmtFin_Type_1     <fct> BLQ, Rec, ALQ, ALQ, GLQ, GLQ, GLQ, ALQ, GLQ, Unf, U~
## $ BsmtFin_SF_1       <dbl> 2, 6, 1, 1, 3, 3, 3, 1, 3, 7, 7, 1, 7, 3, 3, 1, 3, ~
## $ BsmtFin_Type_2     <fct> Unf, LwQ, Unf, Unf, Unf, Unf, Unf, Unf, Unf, Unf, U~
## $ BsmtFin_SF_2       <dbl> 0, 144, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1120, 0~
## $ Bsmt_Unf_SF        <dbl> 441, 270, 406, 1045, 137, 324, 722, 1017, 415, 994,~
## $ Total_Bsmt_SF      <dbl> 1080, 882, 1329, 2110, 928, 926, 1338, 1280, 1595, ~
## $ Heating            <fct> GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA, Gas~
## $ Heating_QC         <fct> Fair, Typical, Typical, Excellent, Good, Excellent,~
## $ Central_Air        <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, ~
## $ Electrical         <fct> SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SB~
## $ First_Flr_SF       <int> 1656, 896, 1329, 2110, 928, 926, 1338, 1280, 1616, ~
## $ Second_Flr_SF      <int> 0, 0, 0, 0, 701, 678, 0, 0, 0, 776, 892, 0, 676, 0,~
## $ Low_Qual_Fin_SF    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Gr_Liv_Area        <int> 1656, 896, 1329, 2110, 1629, 1604, 1338, 1280, 1616~
## $ Bsmt_Full_Bath     <dbl> 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, ~
## $ Bsmt_Half_Bath     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Full_Bath          <int> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 3, 2, ~
## $ Half_Bath          <int> 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, ~
## $ Bedroom_AbvGr      <int> 3, 2, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 2, 1, 4, 4, ~
## $ Kitchen_AbvGr      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ Kitchen_Qual       <fct> Typical, Typical, Good, Excellent, Typical, Good, G~
## $ TotRms_AbvGrd      <int> 7, 5, 6, 8, 6, 7, 6, 5, 5, 7, 7, 6, 7, 5, 4, 12, 8,~
## $ Functional         <fct> Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, T~
## $ Fireplaces         <int> 2, 0, 0, 2, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, ~
## $ Fireplace_Qu       <fct> Good, No_Fireplace, No_Fireplace, Typical, Typical,~
## $ Garage_Type        <fct> Attchd, Attchd, Attchd, Attchd, Attchd, Attchd, Att~
## $ Garage_Finish      <fct> Fin, Unf, Unf, Fin, Fin, Fin, Fin, RFn, RFn, Fin, F~
## $ Garage_Cars        <dbl> 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, ~
## $ Garage_Area        <dbl> 528, 730, 312, 522, 482, 470, 582, 506, 608, 442, 4~
## $ Garage_Qual        <fct> Typical, Typical, Typical, Typical, Typical, Typica~
```

```
## $ Garage_Cond        <fct> Typical, Typical, Typical, Typical, Typical, Typica~
## $ Paved_Drive        <fct> Partial_Pavement, Paved, Paved, Paved, Paved, Paved~
## $ Wood_Deck_SF       <int> 210, 140, 393, 0, 212, 360, 0, 0, 237, 140, 157, 48~
## $ Open_Porch_SF      <int> 62, 0, 36, 0, 34, 36, 0, 82, 152, 60, 84, 21, 75, 0~
## $ Enclosed_Porch     <int> 0, 0, 0, 0, 0, 0, 170, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Three_season_porch <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Screen_Porch       <int> 0, 120, 0, 0, 0, 0, 0, 144, 0, 0, 0, 0, 0, 0, 140, ~
## $ Pool_Area          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Pool_QC            <fct> No_Pool, No_Pool, No_Pool, No_Pool, No_Pool, No_Poo~
## $ Fence              <fct> No_Fence, Minimum_Privacy, No_Fence, No_Fence, Mini~
## $ Misc_Feature       <fct> None, None, Gar2, None, None, None, None, None, Non~
## $ Misc_Val           <int> 0, 0, 12500, 0, 0, 0, 0, 0, 0, 0, 0, 500, 0, 0, 0, ~
## $ Mo_Sold            <int> 5, 6, 6, 4, 3, 6, 4, 1, 3, 6, 4, 3, 5, 2, 6, 6, 6, ~
## $ Year_Sold          <int> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 201~
## $ Sale_Type          <fct> WD , WD , WD , WD , WD , WD , WD , WD , WD , WD , W~
## $ Sale_Condition     <fct> Normal, Normal, Normal, Normal, Normal, Normal, Nor~
## $ Sale_Price         <int> 215000, 105000, 172000, 244000, 189900, 195500, 213~
## $ Longitude          <dbl> -93.61975, -93.61976, -93.61939, -93.61732, -93.638~
## $ Latitude           <dbl> 42.05403, 42.05301, 42.05266, 42.05125, 42.06090, 4~
```

5-Number Summary Statistics

We are going to see the distribution of our data by diving into the summary of the data. By looking at the output of the summary of the data, we can see that there are no NA's in the data, so there will be no need of imputing the data. Also, we can see the various levels of our categorical variables in the summary like MS subclass, MS Zoning, Alley and many more. Upon analyzing the numerical data, we can see skewness in some variables like lot area, lot frontage and sale price as well.

```
summary(housing, head(2))
```

```
##                                MS_SubClass                      MS_Zoning
##  One_Story_1946_and_Newer_All_Styles:1079   Residential_Low_Density:2273
##  (Other)                            :1851   (Other)                 : 657
##
##
##
##
##   Lot_Frontage       Lot_Area       Street                 Alley
##  Min.   :  0.00   Min.   :  1300   Grvl:  12   No_Alley_Access:2732
##  1st Qu.: 43.00   1st Qu.:  7440   Pave:2918   (Other)        : 198
##  Median : 63.00   Median :  9436
##  Mean   : 57.65   Mean   : 10148
##  3rd Qu.: 78.00   3rd Qu.: 11555
##  Max.   :313.00   Max.   :215245
##    Lot_Shape     Land_Contour    Utilities      Lot_Config     Land_Slope
##  Regular:1859   Lvl    :2633   AllPub :2927   Inside :2140   Gtl    :2789
##  (Other):1071   (Other): 297   (Other):   3   (Other): 790   (Other): 141
##
##
##
##
##      Neighborhood   Condition_1    Condition_2     Bldg_Type
##  North_Ames: 443   Norm   :2522   Norm   :2900   OneFam :2425
```

```
##  (Other)   :2487   (Other): 408   (Other):  30   (Other): 505
##
##
##
##
##     House_Style    Overall_Qual   Overall_Cond    Year_Built   Year_Remod_Add
##  One_Story:1481   Average: 825   Average:1654   Min.   :1872   Min.   :1950
##  (Other)  :1449   (Other):2105   (Other):1276   1st Qu.:1954   1st Qu.:1965
##                                                 Median :1973   Median :1993
##                                                 Mean   :1971   Mean   :1984
##                                                 3rd Qu.:2001   3rd Qu.:2004
##                                                 Max.   :2010   Max.   :2010
##    Roof_Style     Roof_Matl     Exterior_1st   Exterior_2nd   Mas_Vnr_Type
##  Gable  :2321   CompShg:2887   VinylSd:1026   VinylSd:1015   None   :1775
##  (Other): 609   (Other):  43   (Other):1904   (Other):1915   (Other):1155
##
##
##
##
##   Mas_Vnr_Area      Exter_Qual     Exter_Cond     Foundation     Bsmt_Qual
##  Min.   :   0.0   Typical:1799   Typical:2549   PConc  :1310   Typical:1283
##  1st Qu.:   0.0   (Other):1131   (Other): 381   (Other):1620   (Other):1647
##  Median :   0.0
##  Mean   : 101.1
##  3rd Qu.: 162.8
##  Max.   :1600.0
##    Bsmt_Cond    Bsmt_Exposure  BsmtFin_Type_1  BsmtFin_SF_1   BsmtFin_Type_2
##  Typical:2616   No     :1906   GLQ    : 859   Min.   :0.000   Unf    :2499
##  (Other): 314   (Other):1024   (Other):2071   1st Qu.:3.000   (Other): 431
##                                               Median :3.000
##                                               Mean   :4.177
##                                               3rd Qu.:7.000
##                                               Max.   :7.000
##   BsmtFin_SF_2      Bsmt_Unf_SF     Total_Bsmt_SF     Heating
##  Min.   :   0.00   Min.   :   0.0   Min.   :   0   GasA   :2885
##  1st Qu.:   0.00   1st Qu.: 219.0   1st Qu.: 793   (Other):  45
##  Median :   0.00   Median : 465.5   Median : 990
##  Mean   :  49.71   Mean   : 559.1   Mean   :1051
##  3rd Qu.:   0.00   3rd Qu.: 801.8   3rd Qu.:1302
##  Max.   :1526.00   Max.   :2336.0   Max.   :6110
##      Heating_QC   Central_Air   Electrical    First_Flr_SF    Second_Flr_SF
##  Excellent:1495   N: 196      SBrkr  :2682   Min.   : 334.0   Min.   :   0.0
##  (Other)  :1435   Y:2734      (Other): 248   1st Qu.: 876.2   1st Qu.:   0.0
##                                              Median :1084.0   Median :   0.0
##                                              Mean   :1159.6   Mean   : 335.5
##                                              3rd Qu.:1384.0   3rd Qu.: 703.8
##                                              Max.   :5095.0   Max.   :2065.0
##  Low_Qual_Fin_SF     Gr_Liv_Area   Bsmt_Full_Bath   Bsmt_Half_Bath
##  Min.   :   0.000   Min.   : 334   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:   0.000   1st Qu.:1126   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :   0.000   Median :1442   Median :0.0000   Median :0.00000
##  Mean   :   4.677   Mean   :1500   Mean   :0.4311   Mean   :0.06109
##  3rd Qu.:   0.000   3rd Qu.:1743   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :1064.000   Max.   :5642   Max.   :3.0000   Max.   :2.00000
```

```
##    Full_Bath       Half_Bath      Bedroom_AbvGr   Kitchen_AbvGr
##  Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:1.000
##  Median :2.000   Median :0.0000   Median :3.000   Median :1.000
##  Mean   :1.567   Mean   :0.3795   Mean   :2.854   Mean   :1.044
##  3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:1.000
##  Max.   :4.000   Max.   :2.0000   Max.   :8.000   Max.   :3.000
##   Kitchen_Qual  TotRms_AbvGrd      Functional     Fireplaces
##  Typical:1494   Min.   : 2.000   Typ    :2728   Min.   :0.0000
##  (Other):1436   1st Qu.: 5.000   (Other): 202   1st Qu.:0.0000
##                 Median : 6.000                  Median :1.0000
##                 Mean   : 6.443                  Mean   :0.5993
##                 3rd Qu.: 7.000                  3rd Qu.:1.0000
##                 Max.   :15.000                  Max.   :4.0000
##        Fireplace_Qu   Garage_Type   Garage_Finish   Garage_Cars
##  No_Fireplace:1422   Attchd :1731   Unf    :1231   Min.   :0.000
##  (Other)     :1508   (Other):1199   (Other):1699   1st Qu.:1.000
##                                                    Median :2.000
##                                                    Mean   :1.766
##                                                    3rd Qu.:2.000
##                                                    Max.   :5.000
##   Garage_Area      Garage_Qual    Garage_Cond    Paved_Drive
##  Min.   :   0.0   Typical:2615   Typical:2665   Paved  :2652
##  1st Qu.: 320.0   (Other): 315   (Other): 265   (Other): 278
##  Median : 480.0
##  Mean   : 472.7
##  3rd Qu.: 576.0
##  Max.   :1488.0
##   Wood_Deck_SF     Open_Porch_SF    Enclosed_Porch    Three_season_porch
##  Min.   :   0.00   Min.   :  0.00   Min.   :   0.00   Min.   :  0.000
##  1st Qu.:   0.00   1st Qu.:  0.00   1st Qu.:   0.00   1st Qu.:  0.000
##  Median :   0.00   Median : 27.00   Median :   0.00   Median :  0.000
##  Mean   :  93.75   Mean   : 47.53   Mean   :  23.01   Mean   :  2.592
##  3rd Qu.: 168.00   3rd Qu.: 70.00   3rd Qu.:   0.00   3rd Qu.:  0.000
##  Max.   :1424.00   Max.   :742.00   Max.   :1012.00   Max.   :508.000
##   Screen_Porch   Pool_Area          Pool_QC          Fence       Misc_Feature
##  Min.   :  0   Min.   :  0.000   No_Pool:2917   No_Fence:2358   None   :2824
##  1st Qu.:  0   1st Qu.:  0.000   (Other):  13   (Other) : 572   (Other): 106
##  Median :  0   Median :  0.000
##  Mean   : 16   Mean   :  2.243
##  3rd Qu.:  0   3rd Qu.:  0.000
##  Max.   :576   Max.   :800.000
##     Misc_Val           Mo_Sold         Year_Sold      Sale_Type
##  Min.   :    0.00   Min.   : 1.000   Min.   :2006   WD     :2536
##  1st Qu.:    0.00   1st Qu.: 4.000   1st Qu.:2007   (Other): 394
##  Median :    0.00   Median : 6.000   Median :2008
##  Mean   :   50.63   Mean   : 6.216   Mean   :2008
##  3rd Qu.:    0.00   3rd Qu.: 8.000   3rd Qu.:2009
##  Max.   :17000.00   Max.   :12.000   Max.   :2010
##  Sale_Condition   Sale_Price       Longitude         Latitude
##  Normal :2413   Min.   : 12789   Min.   :-93.69   Min.   :41.99
##  (Other): 517   1st Qu.:129500   1st Qu.:-93.66   1st Qu.:42.02
##                 Median :160000   Median :-93.64   Median :42.03
##                 Mean   :180796   Mean   :-93.64   Mean   :42.03
```

```
##                 3rd Qu.:213500   3rd Qu.:-93.62   3rd Qu.:42.05
##                 Max.   :755000   Max.   :-93.58   Max.   :42.06
```

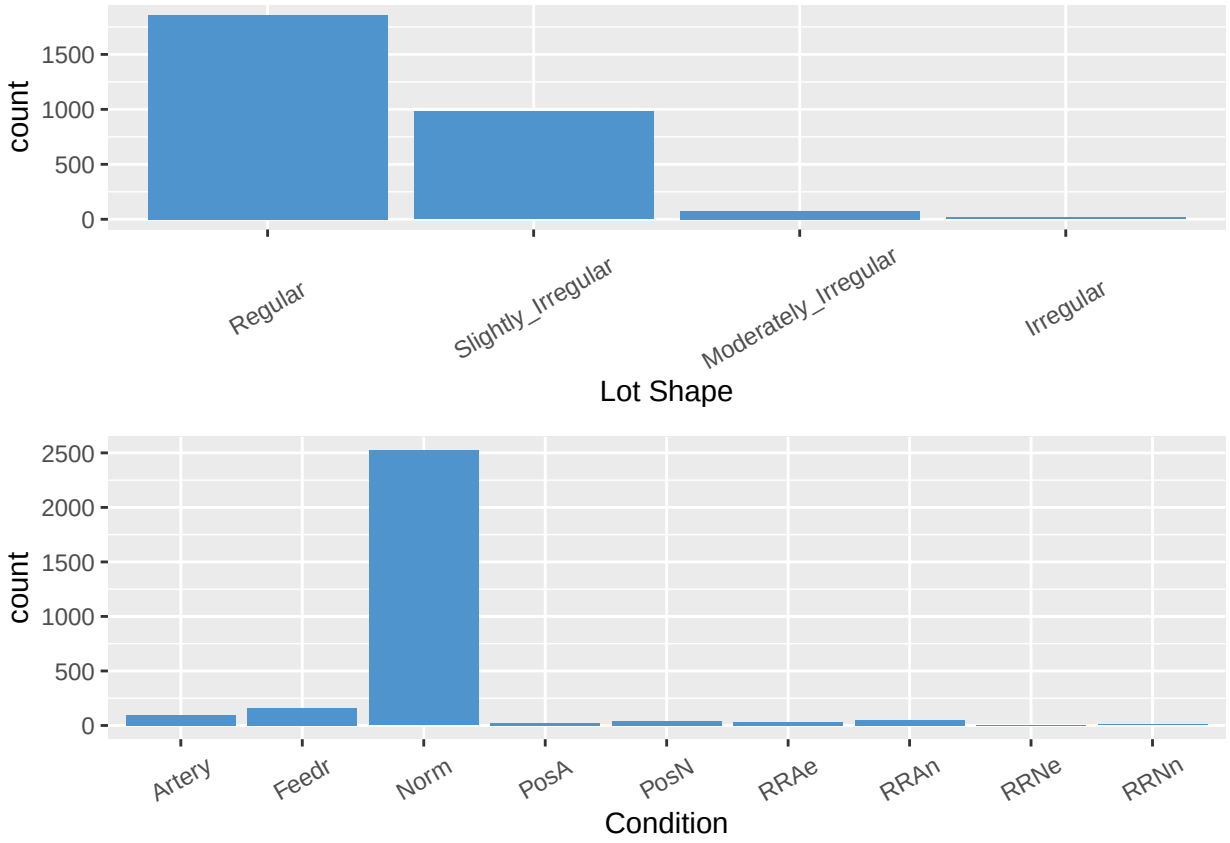
```
g1 <- ggplot(housing, aes(x = Lot_Shape))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5)) +
  labs(x = "Lot Shape")

g2 <- ggplot(housing, aes(x = Lot_Config))+
  geom_bar(fill='steelblue3') +
  labs(x = "Lot Configuration")

g3 <- ggplot(housing, aes(x = Condition_1))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5)) +
  labs(x = "Condition")

g4 <- ggplot(housing, aes(x = Bldg_Type))+
  geom_bar(fill='steelblue3') +
  labs(x = "Building Type")

g5 <- ggplot(housing, aes(x = House_Style))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 60, vjust = 0.5)) +
  labs(x = "House Style")

g6 <- ggplot(housing, aes(x = Roof_Style))+
  geom_bar(fill='steelblue3') +
  labs("Roof Style")

g7 <- ggplot(housing, aes(x = Mas_Vnr_Type))
  geom_bar(fill='steelblue3')
```

```
## geom_bar: just = 0.5, width = NULL, na.rm = FALSE, orientation = NA
## stat_count: width = NULL, na.rm = FALSE, orientation = NA
## position_stack
```

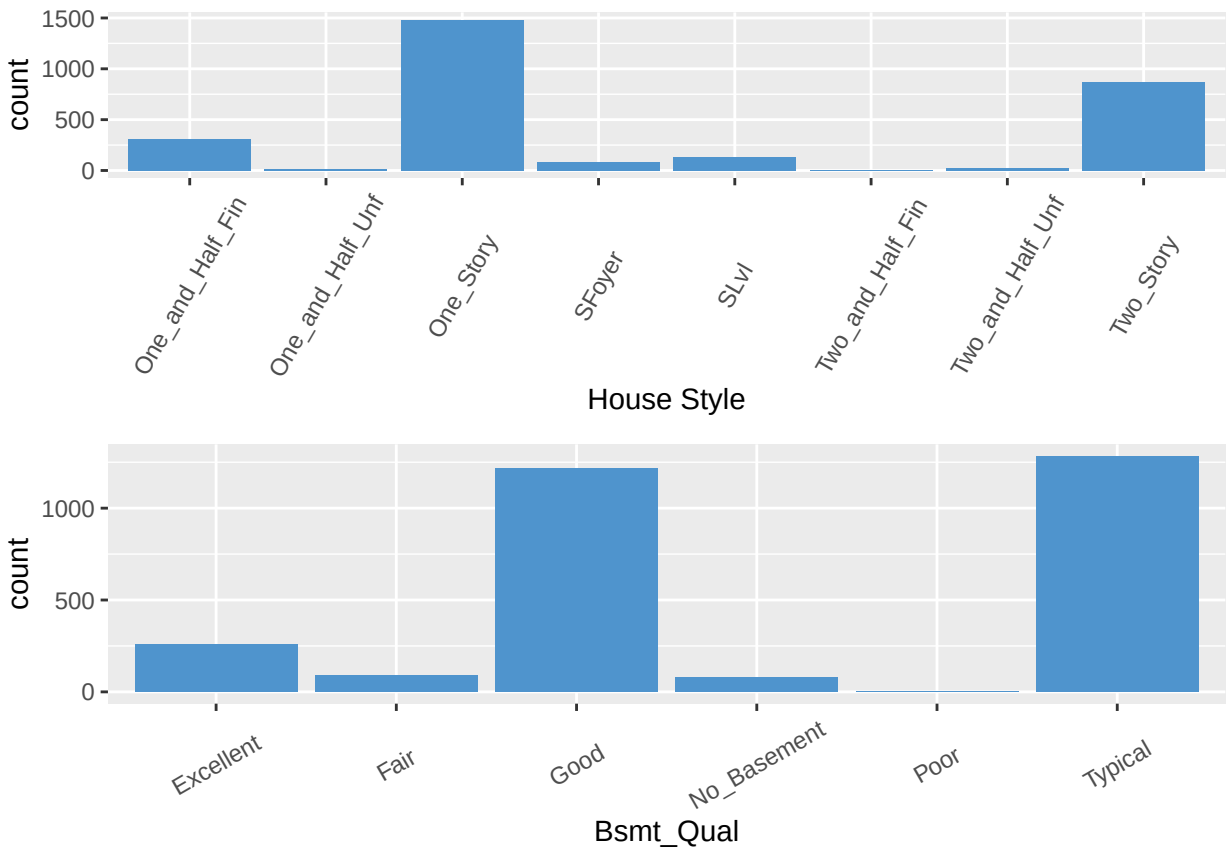
```
g8 <- ggplot(housing, aes(x = Exter_Qual)) +
  geom_bar(fill='steelblue3') +
  labs("Exterior Quality")

g9 <- ggplot(housing, aes(x = Exter_Cond))+
  geom_bar(fill='steelblue3')

g10 <- ggplot(housing, aes(x = Foundation))+
  geom_bar(fill='steelblue3') +
  labs("Foundation")

g11 <- ggplot(housing, aes(x = Bsmt_Qual))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5))

g12 <- ggplot(housing, aes(x = Bsmt_Exposure))+
```

```
  geom_bar(fill='steelblue3')

g13 <- ggplot(housing, aes(x = BsmtFin_Type_1))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5))

g14 <- ggplot(housing, aes(x = Heating_QC))+
  geom_bar(fill='steelblue3')

g15 <- ggplot(housing, aes(x = Central_Air))+
  geom_bar(fill='steelblue3')

g16 <- ggplot(housing, aes(x = Electrical))+
  geom_bar(fill='steelblue3')

g17 <- ggplot(housing, aes(x = Kitchen_Qual))+
  geom_bar(fill='steelblue3')

g18 <- ggplot(housing, aes(x = Fireplace_Qu))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5))

g19 <- ggplot(housing, aes(x = Garage_Type))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 60, vjust = 0.5))

g20 <- ggplot(housing, aes(x = Garage_Finish))+
  geom_bar(fill='steelblue3')

g21 <- ggplot(housing, aes(x = Garage_Qual))+
  geom_bar(fill='steelblue3')

g22 <- ggplot(housing, aes(x = Garage_Cond))+
  geom_bar(fill='steelblue3')

g23 <- ggplot(housing, aes(x = Paved_Drive))+
  geom_bar(fill='steelblue3')

g24 <- ggplot(housing, aes(x = Fence))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5))

g25 <- ggplot(housing, aes(x = Sale_Type))+
  geom_bar(fill='steelblue3') +
  theme(axis.text.x=element_text(angle = 30, vjust = 0.5))

g26 <- ggplot(housing, aes(x = Sale_Condition))+
  geom_bar(fill='steelblue3')

g27 <- ggplot(housing, aes(x = Kitchen_Qual))+
  geom_bar(fill='steelblue3')

g28 <- ggplot(housing, aes(x = Kitchen_Qual))+
```

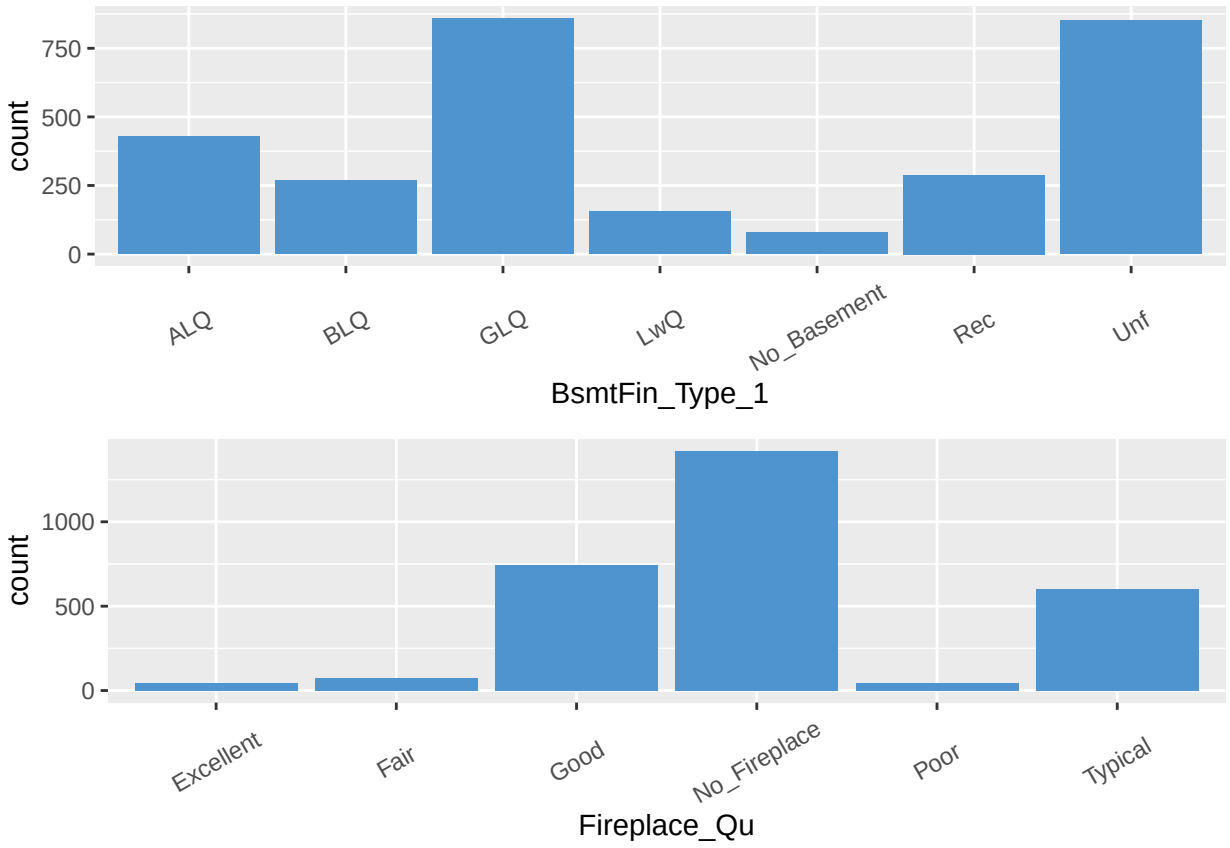
```
  geom_bar(fill='steelblue3')

g29 <- ggplot(housing, aes(x = MS_SubClass))+
  geom_bar(fill='steelblue3')+
  coord_flip()

g30 <- ggplot(housing, aes(x = MS_Zoning))+
  geom_bar(fill='steelblue3')+
  coord_flip()

g31 <- ggplot(housing, aes(x = Exterior_1st))+
  geom_bar(fill='steelblue3')+
  coord_flip()

g32 <- ggplot(housing, aes(x = Exterior_2nd))+
  geom_bar(fill='steelblue3')+
  coord_flip()

g33 <- ggplot(housing, aes(x = Neighborhood))+
  geom_bar(fill='steelblue3')+
  coord_flip()

grid.arrange(g1, g3, ncol = 1, nrow = 2)
```



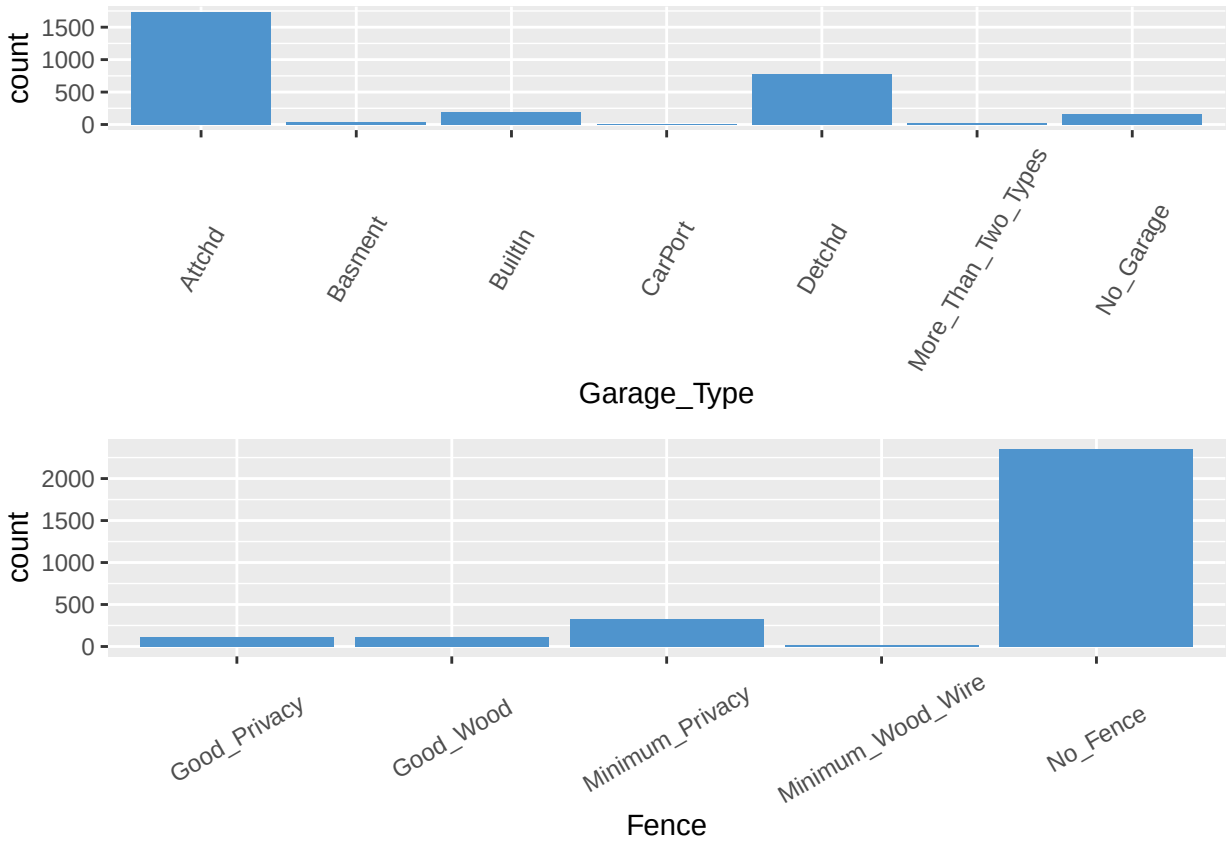
```
grid.arrange(g5, g11, ncol = 1, nrow = 2)
```



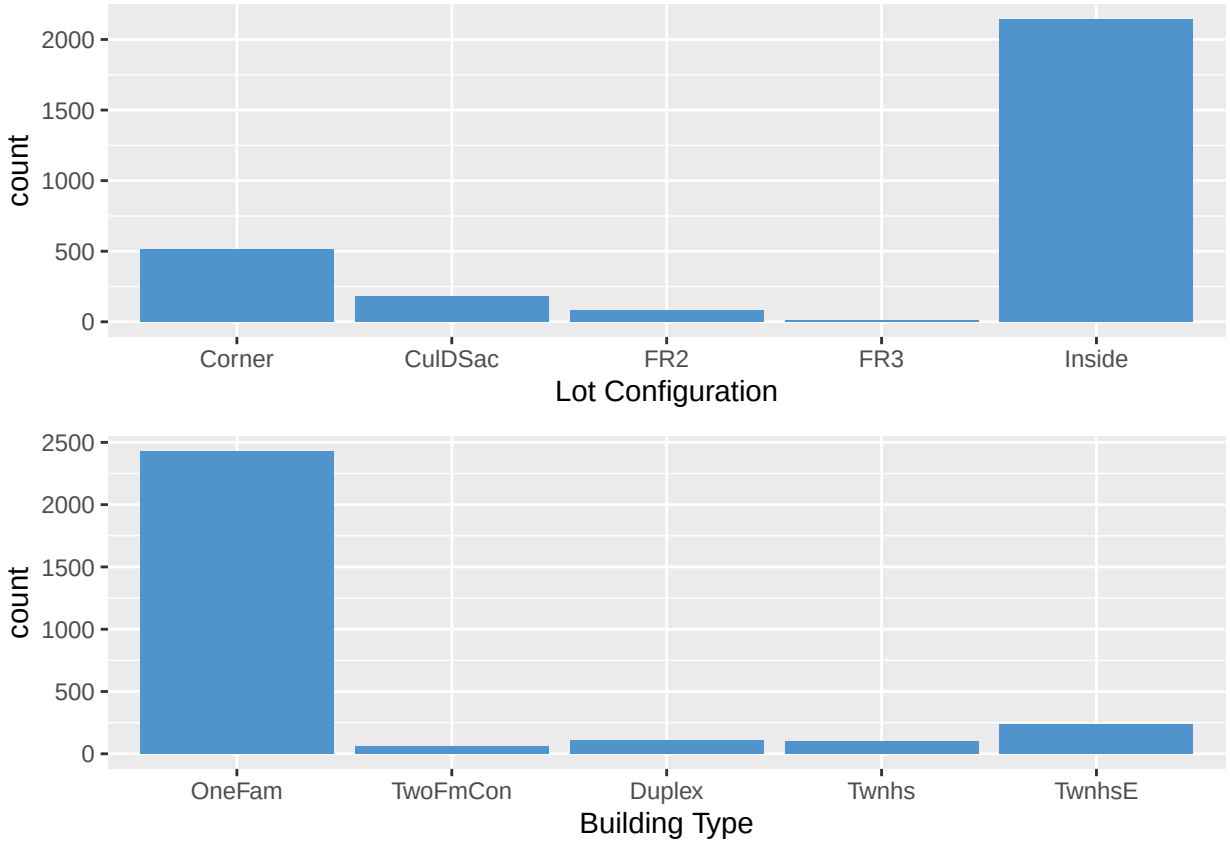

```
grid.arrange(g13, g18, ncol = 1, nrow = 2)
```



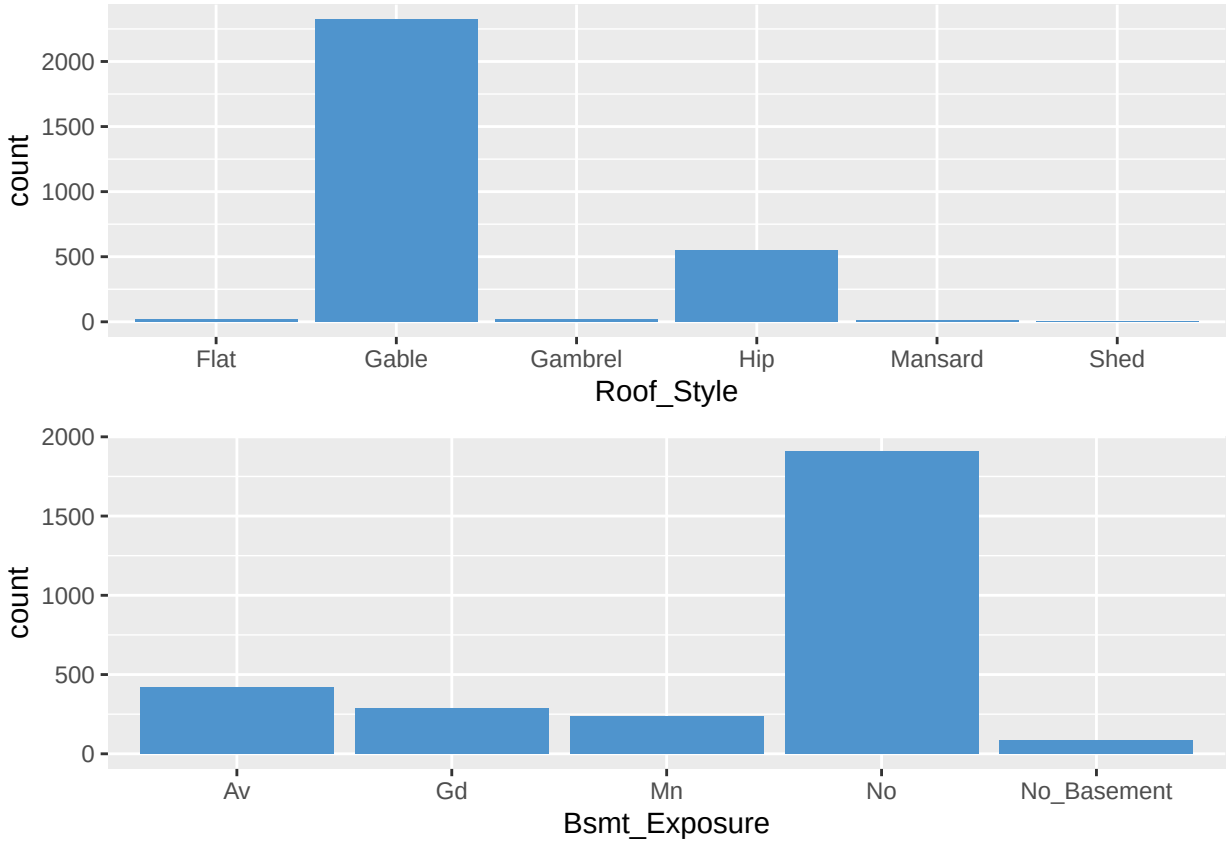

```
grid.arrange(g19, g24, ncol = 1, nrow = 2)
```



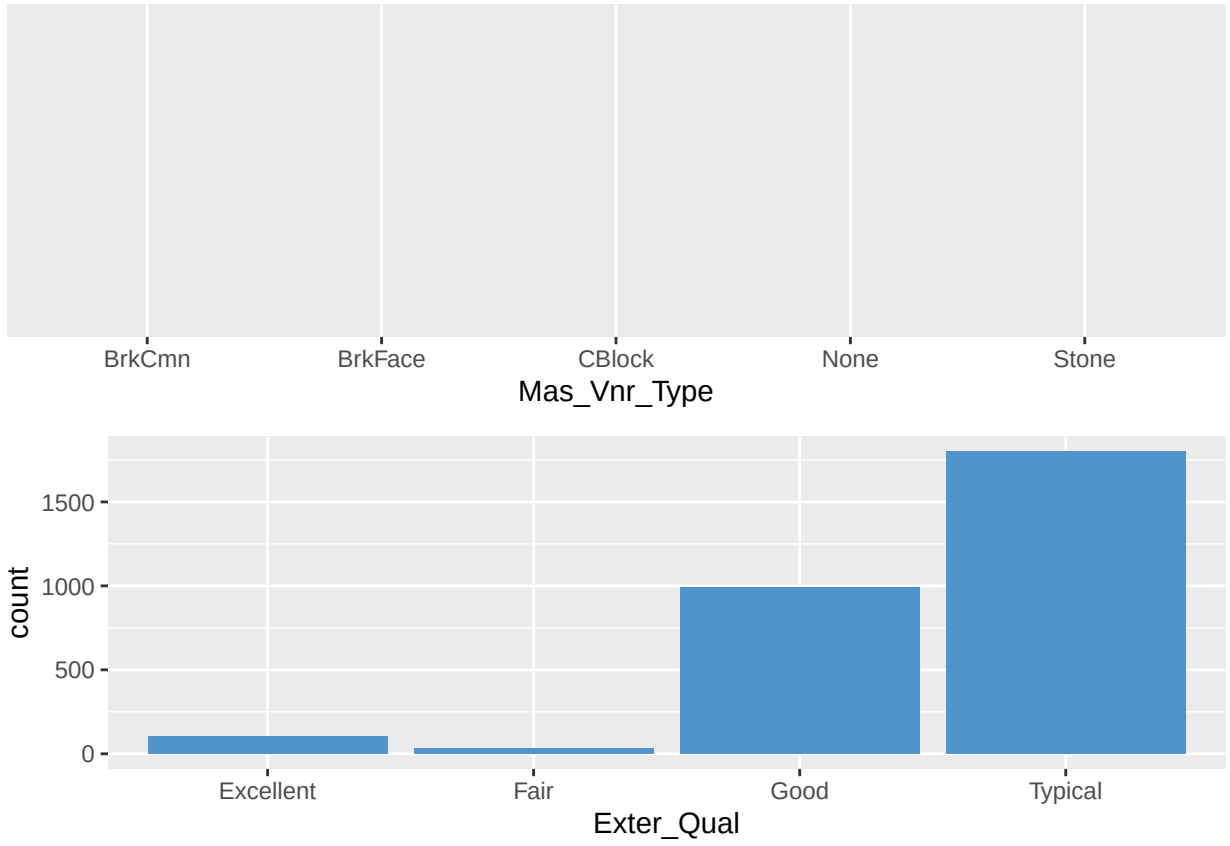

```
grid.arrange(g2, g4,  ncol = 1, nrow = 2)
```



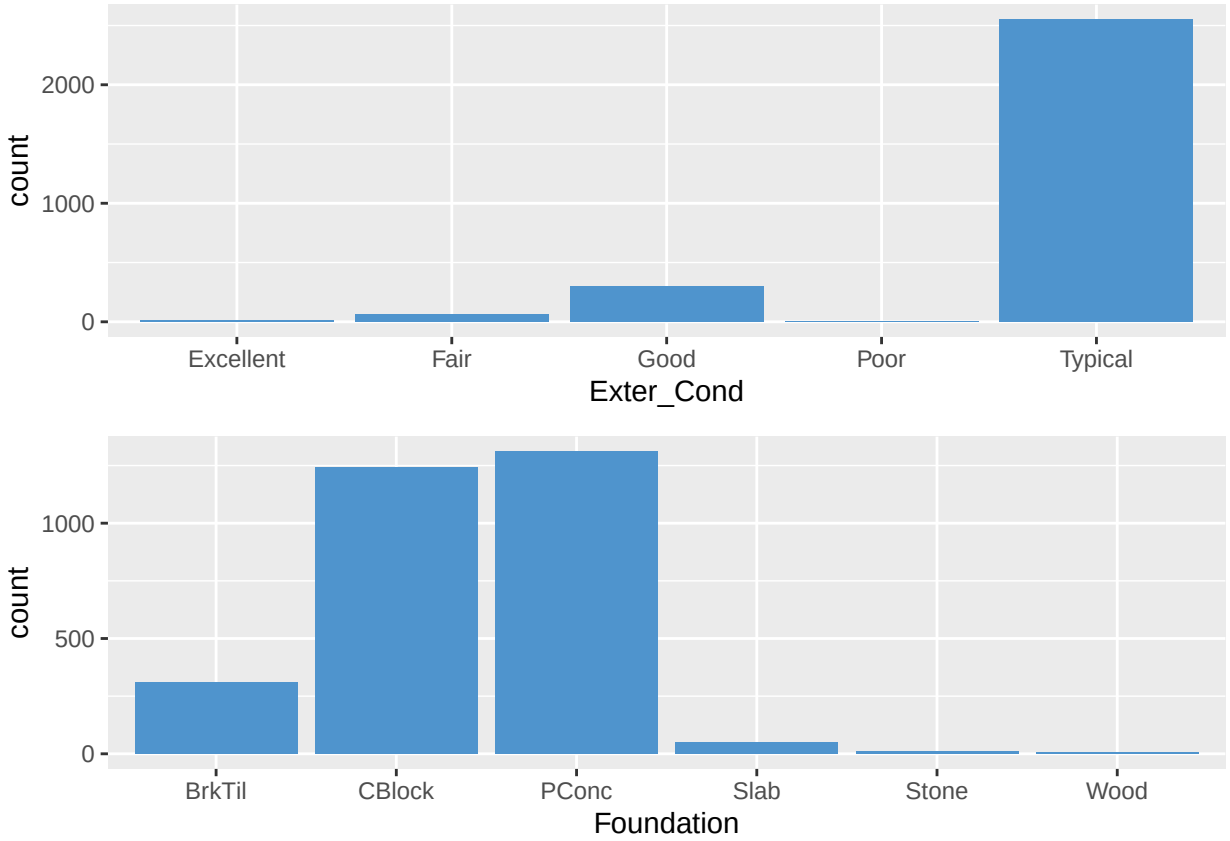

```
grid.arrange(g6, g12, ncol = 1, nrow = 2)
```



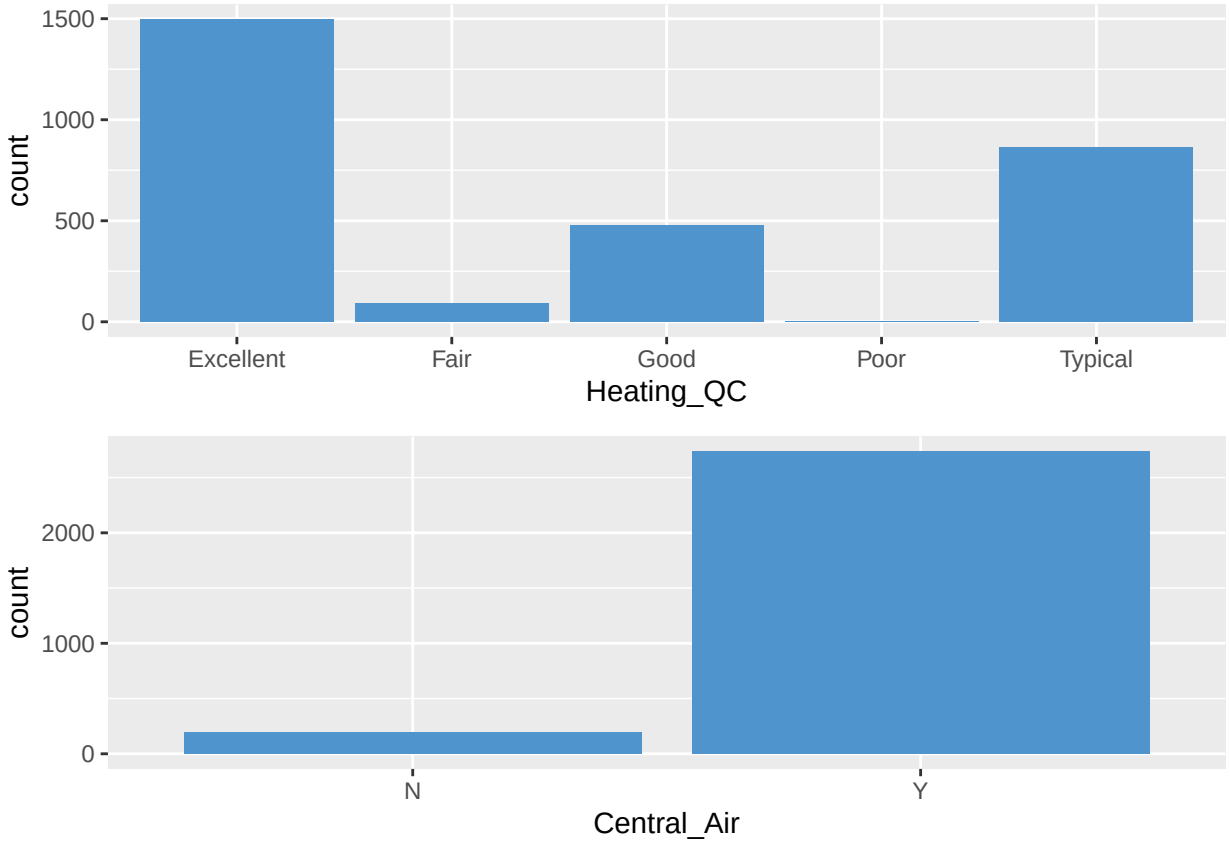

```
grid.arrange(g7, g8,  ncol = 1, nrow = 2)
```



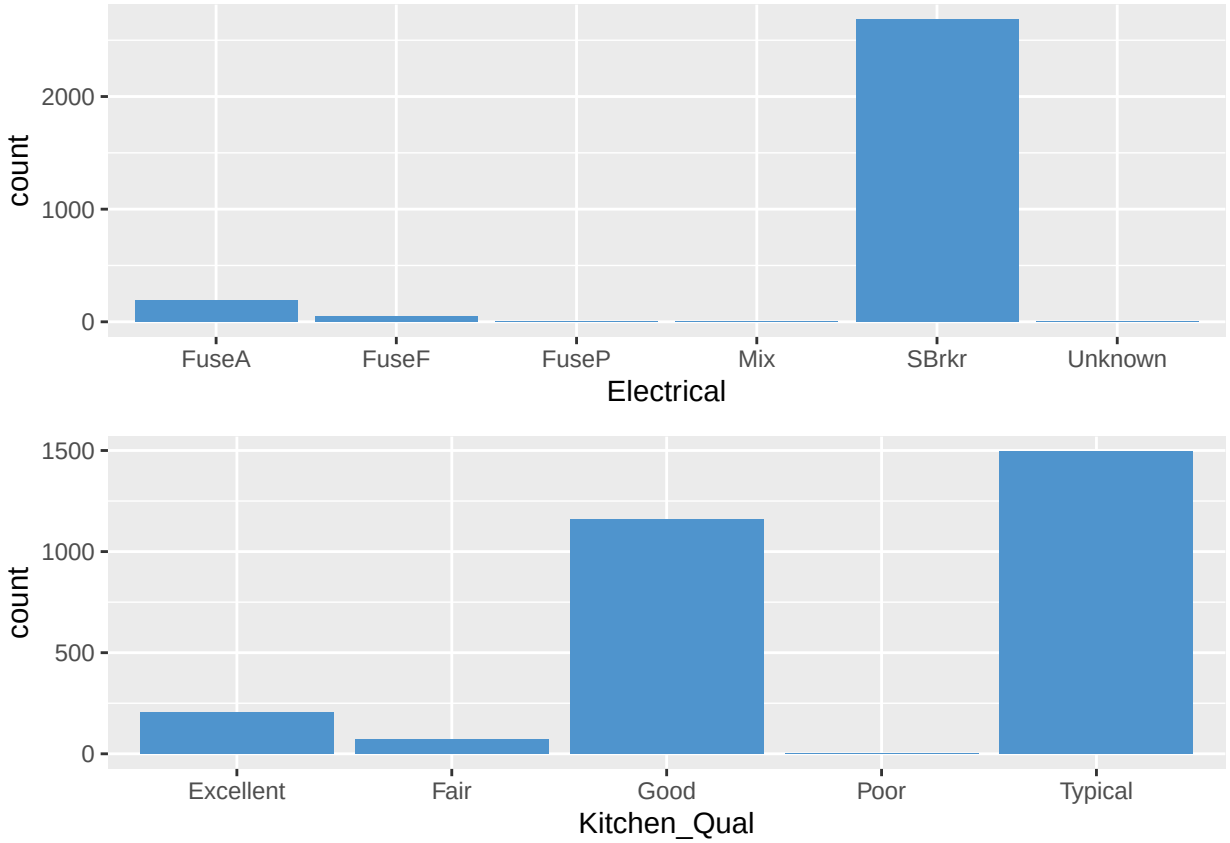

```
grid.arrange(g9, g10, ncol = 1, nrow = 2)
```



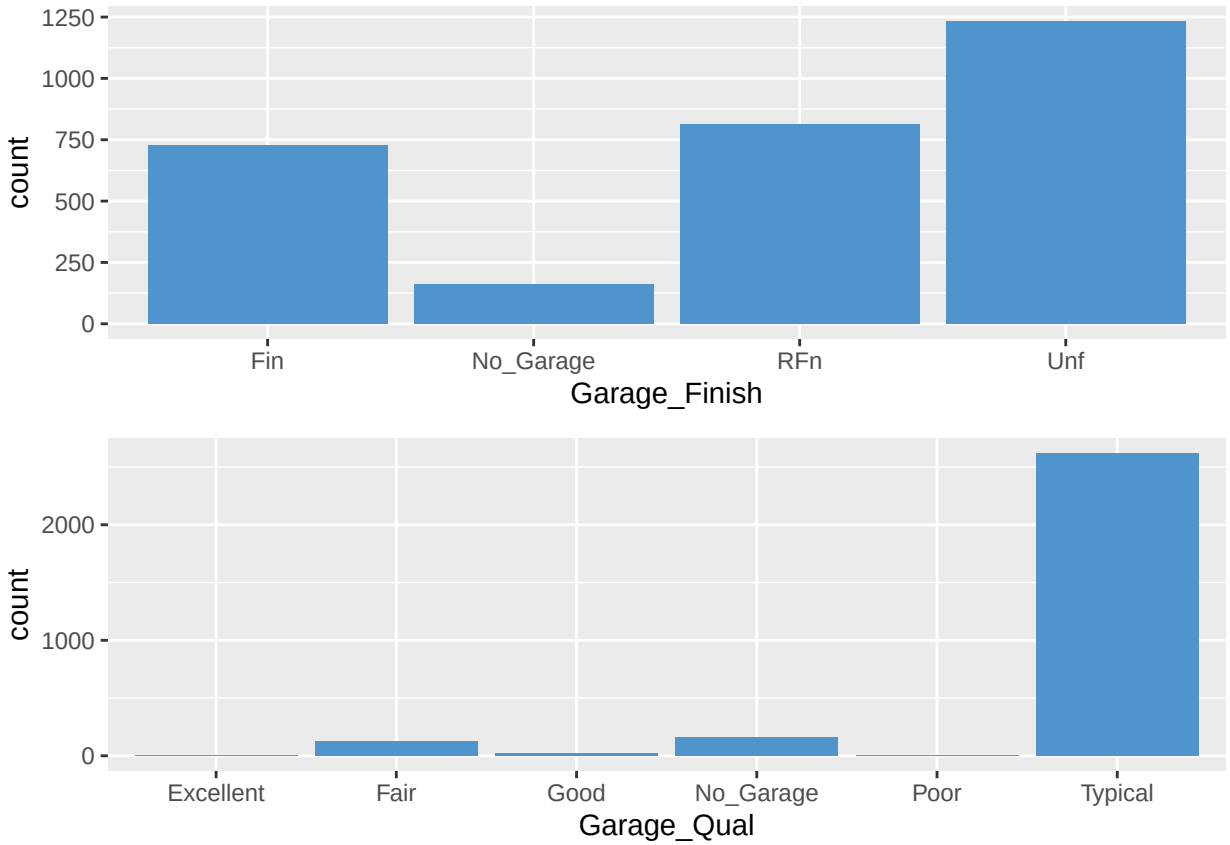

```
grid.arrange(g14, g15, ncol = 1, nrow = 2)
```



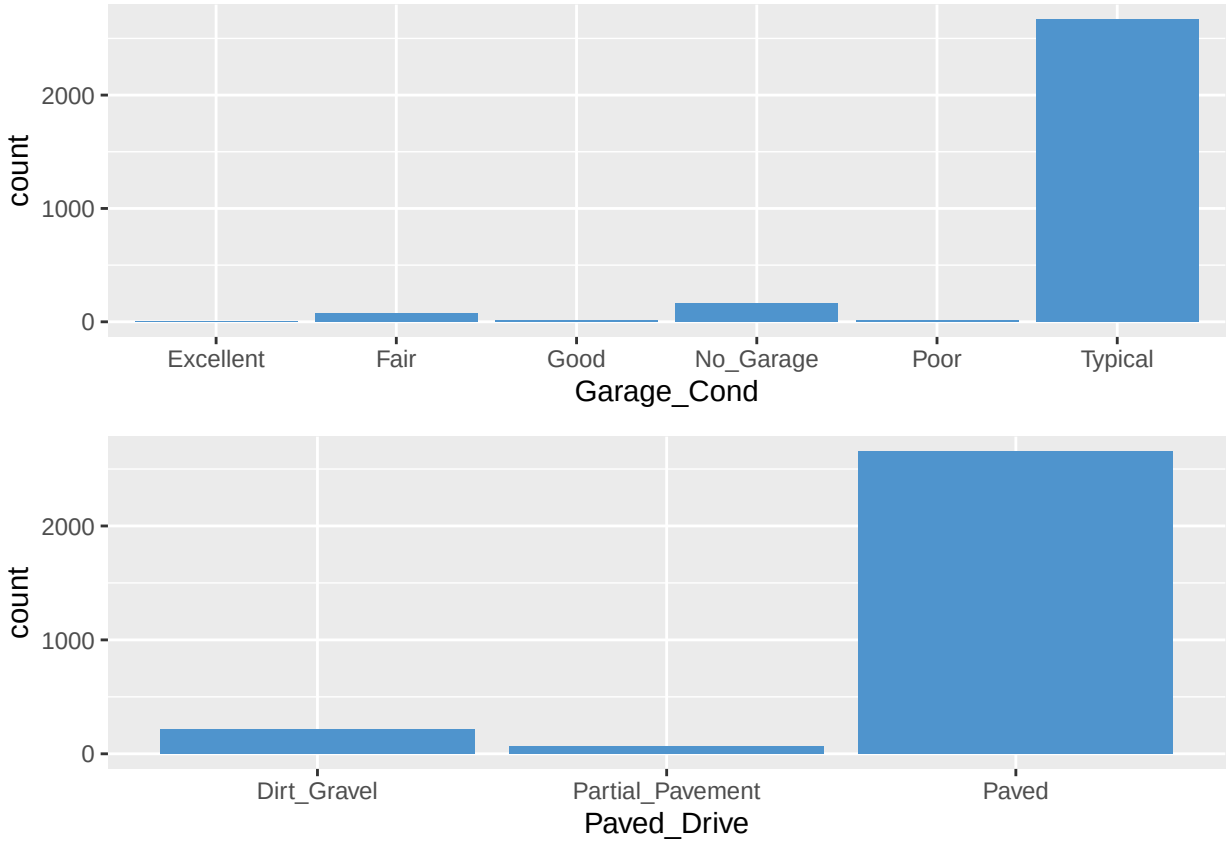

```
grid.arrange(g16, g17, ncol = 1, nrow = 2)
```



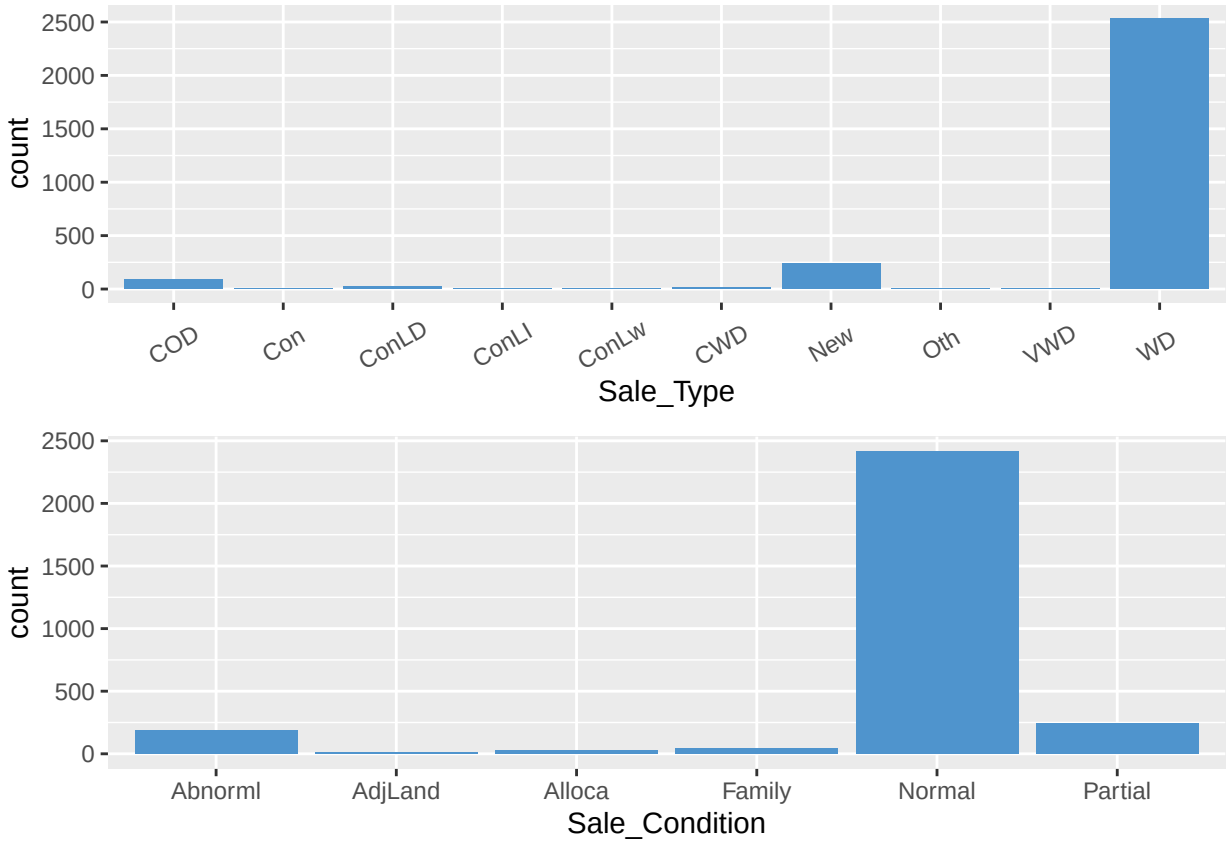

```
grid.arrange(g20, g21,ncol = 1, nrow = 2)
```



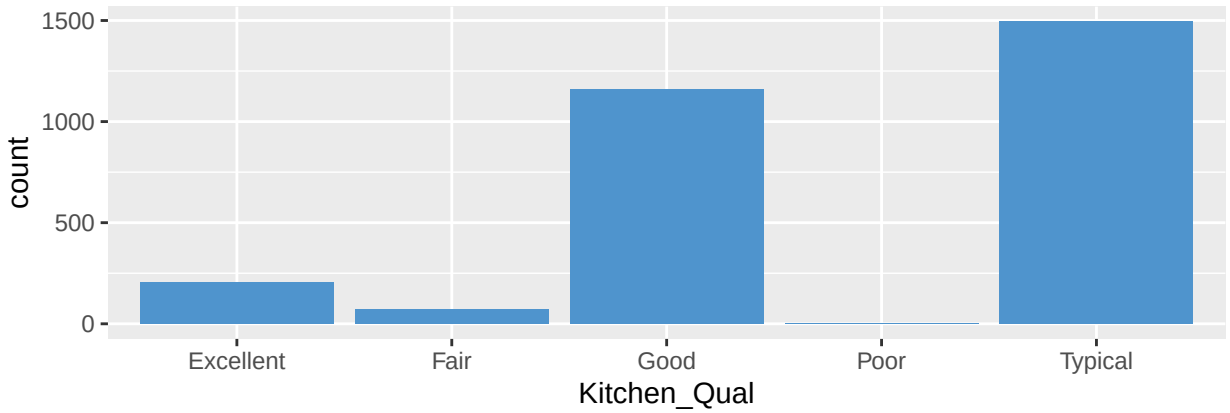

```
grid.arrange(g22, g23, ncol = 1, nrow = 2)
```



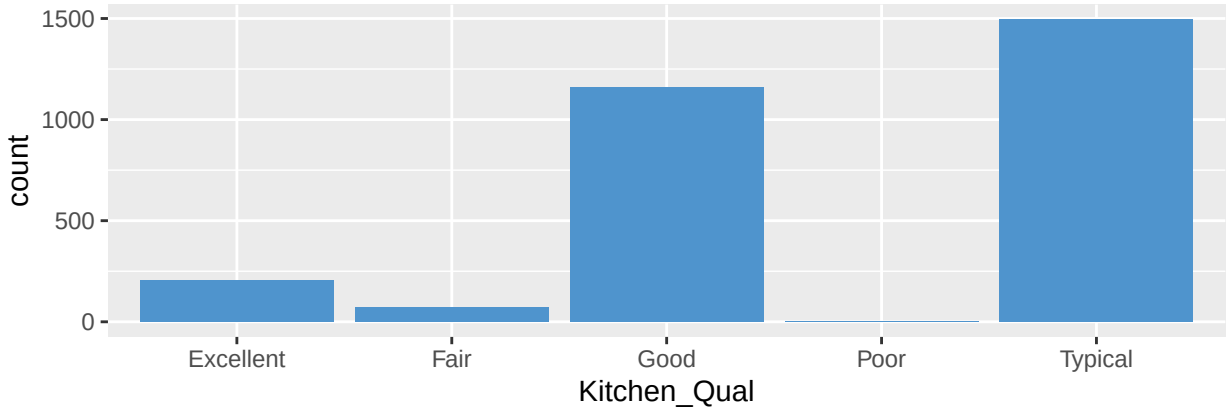

```
grid.arrange(g25, g26, ncol = 1, nrow = 2)
```



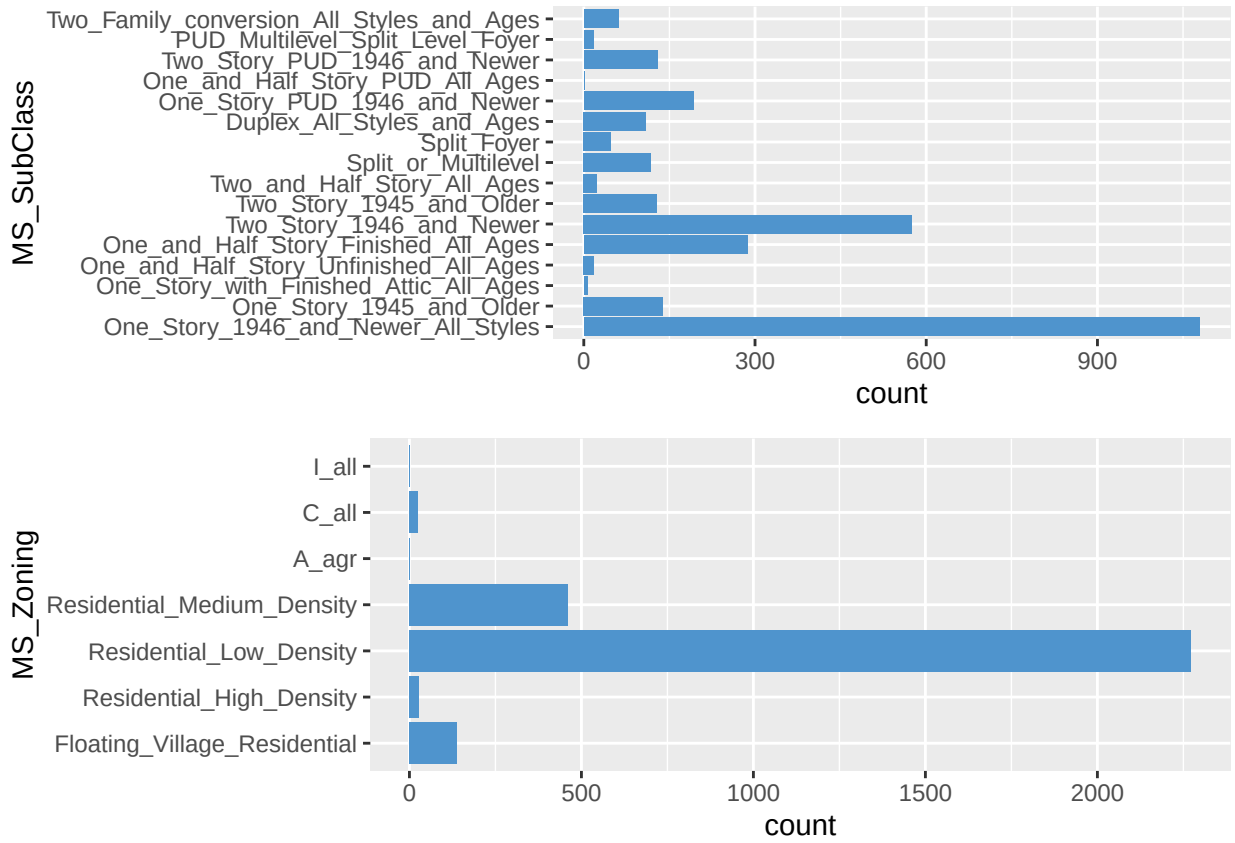

```
grid.arrange(g27, ncol = 1, nrow = 2)
```



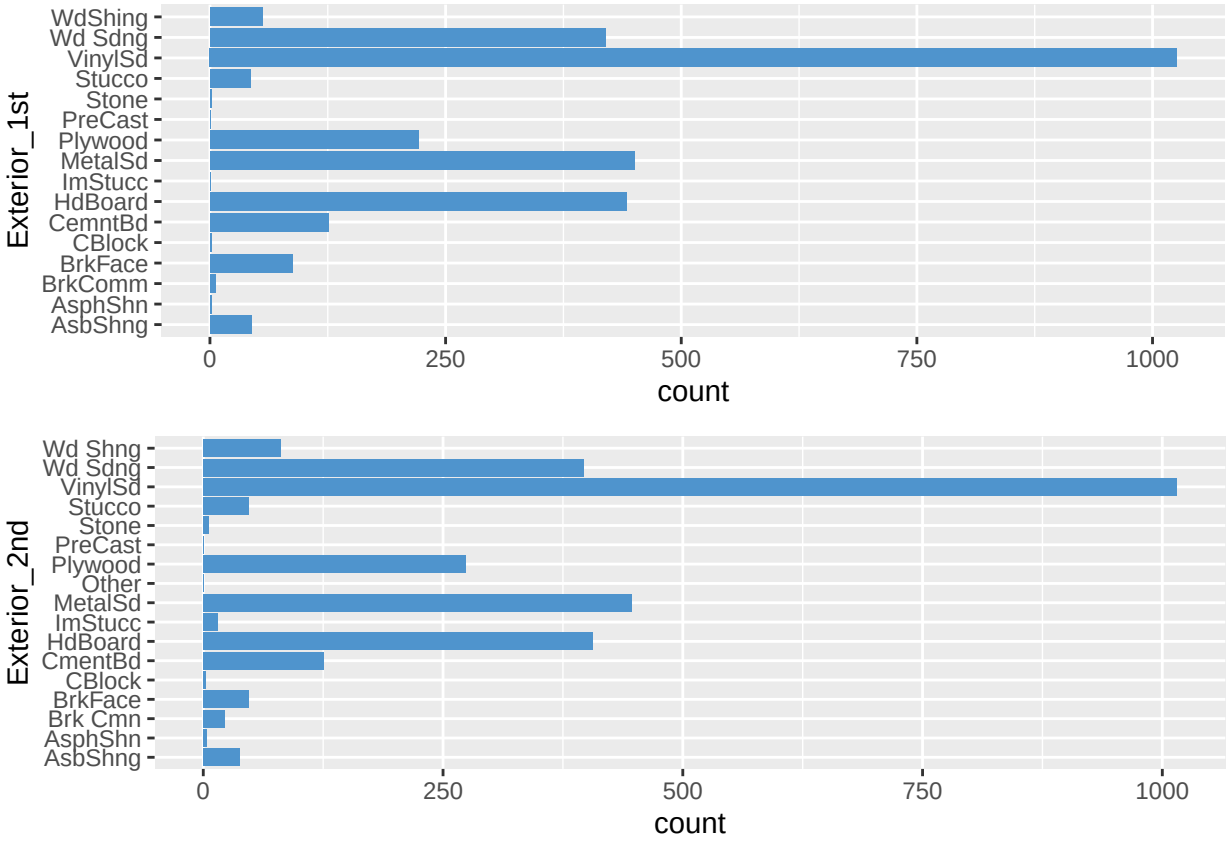

```
grid.arrange(g28, ncol = 1, nrow = 2)
```



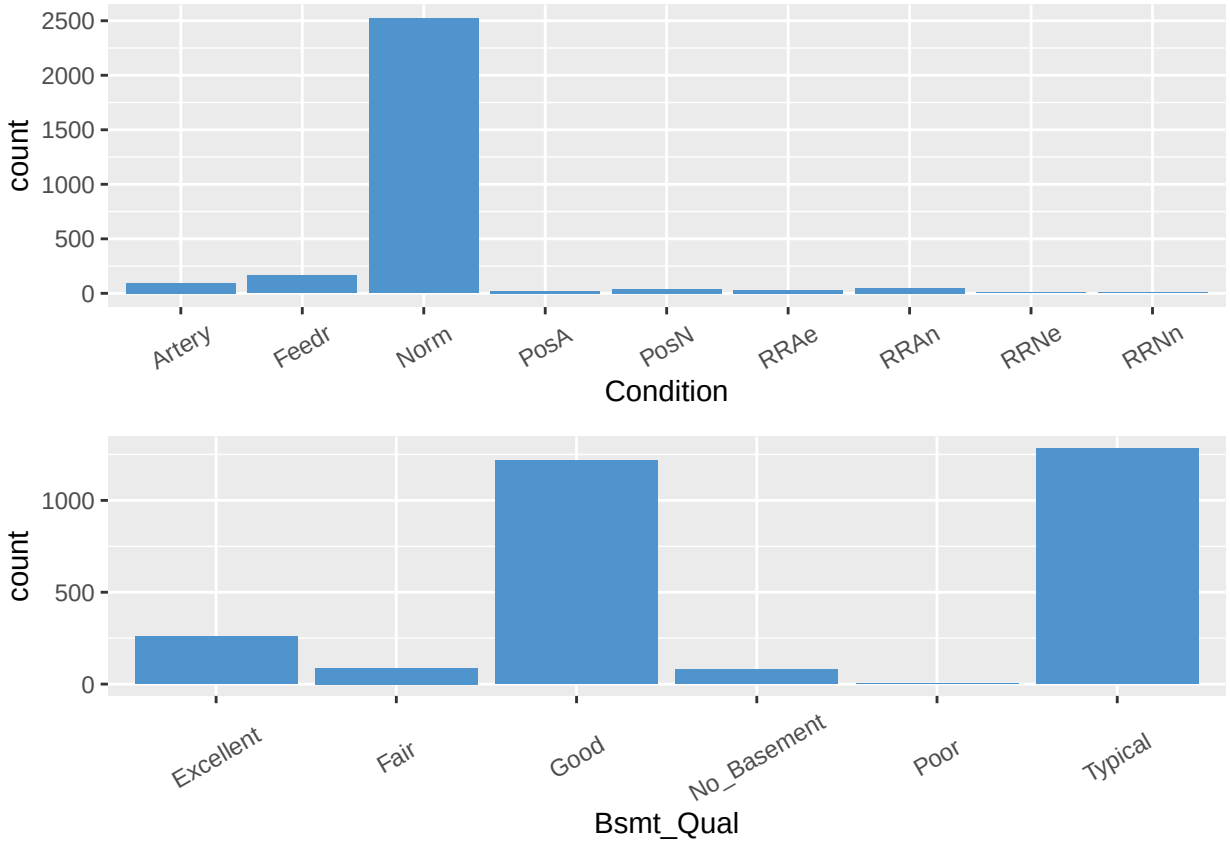

```
grid.arrange(g29, g30,  ncol = 1, nrow = 2)
```




```
grid.arrange(g31, g32, ncol = 1, nrow = 2)
```

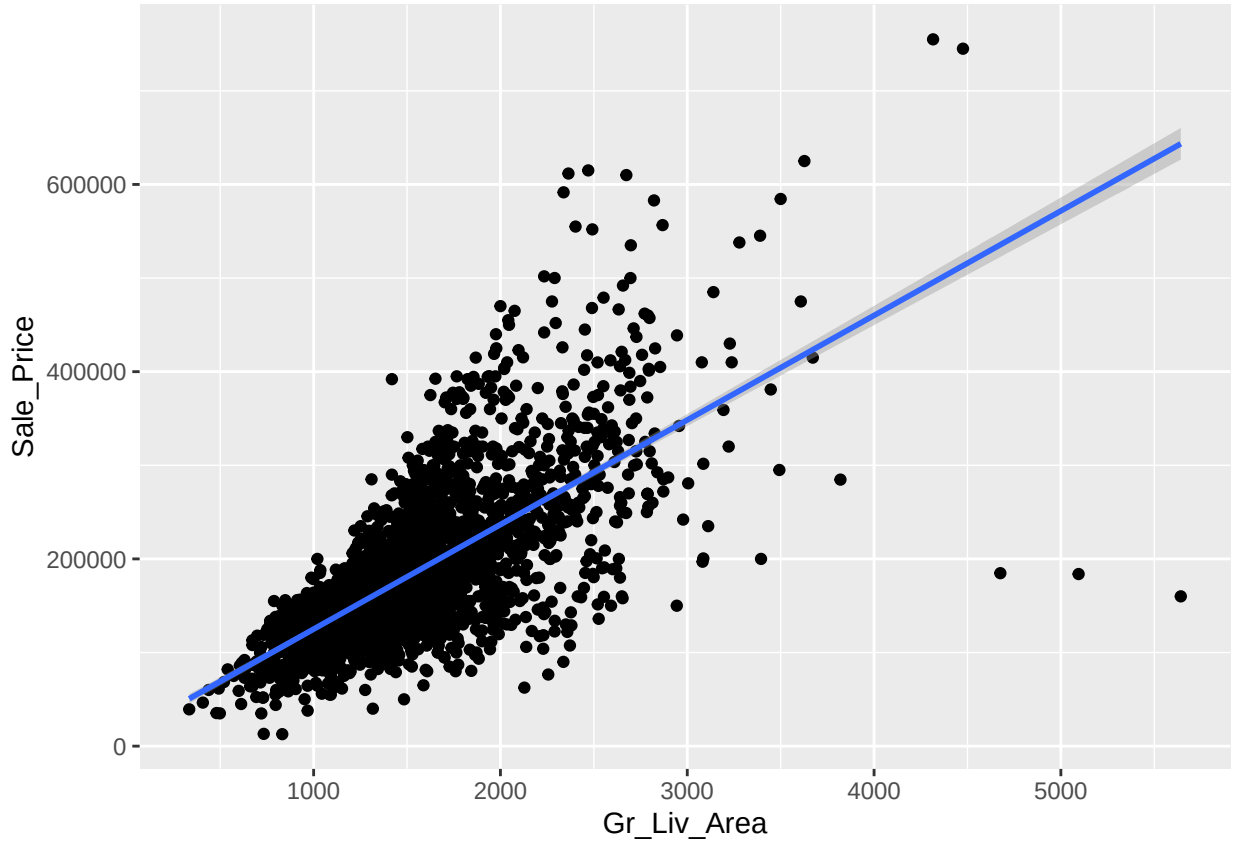



```
grid.arrange(g3, g11, ncol = 1, nrow = 2)
```

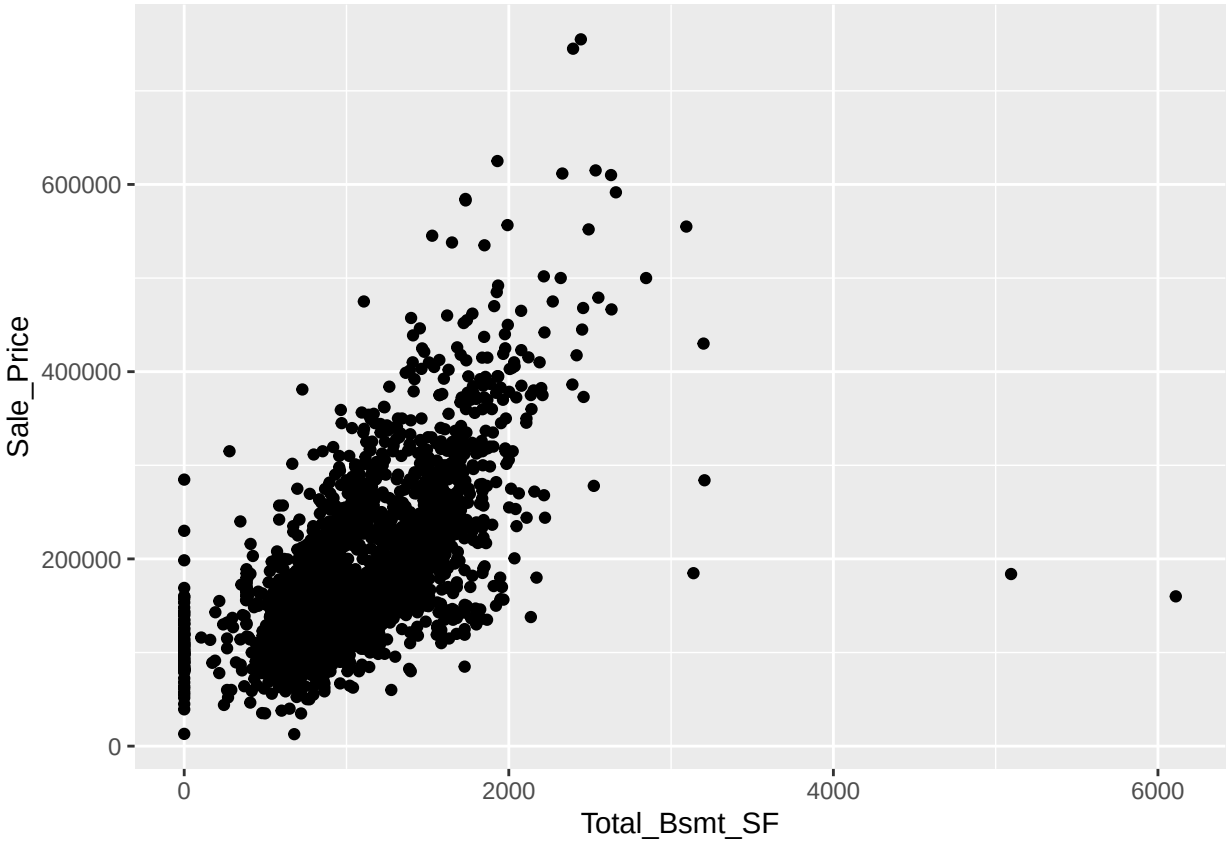



```
ggplot(data=housing,mapping = aes(x=Gr_Liv_Area,y=Sale_Price))+
  geom_point()+
  geom_smooth(method = "lm")
```

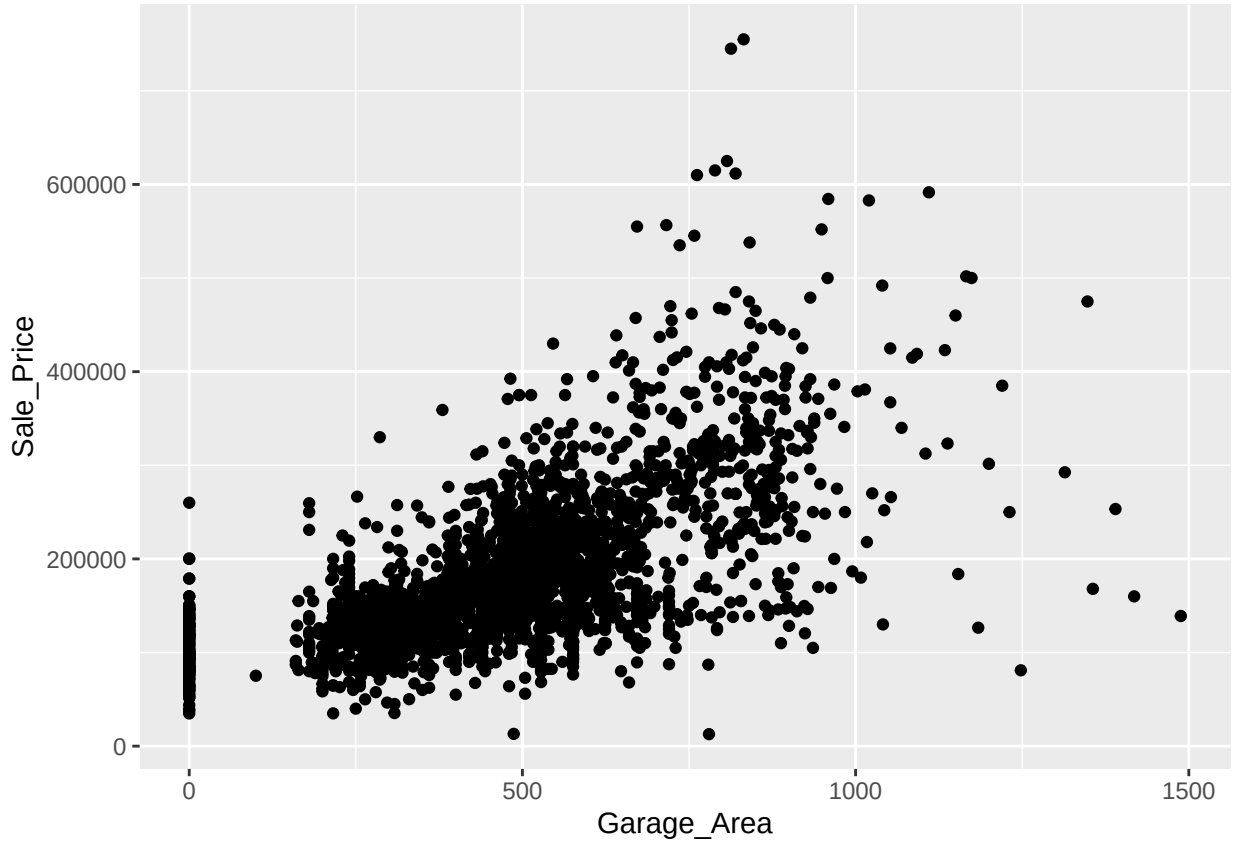
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(data=housing,mapping = aes(x=Total_Bsmt_SF ,y=Sale_Price))+
  geom_point()
```



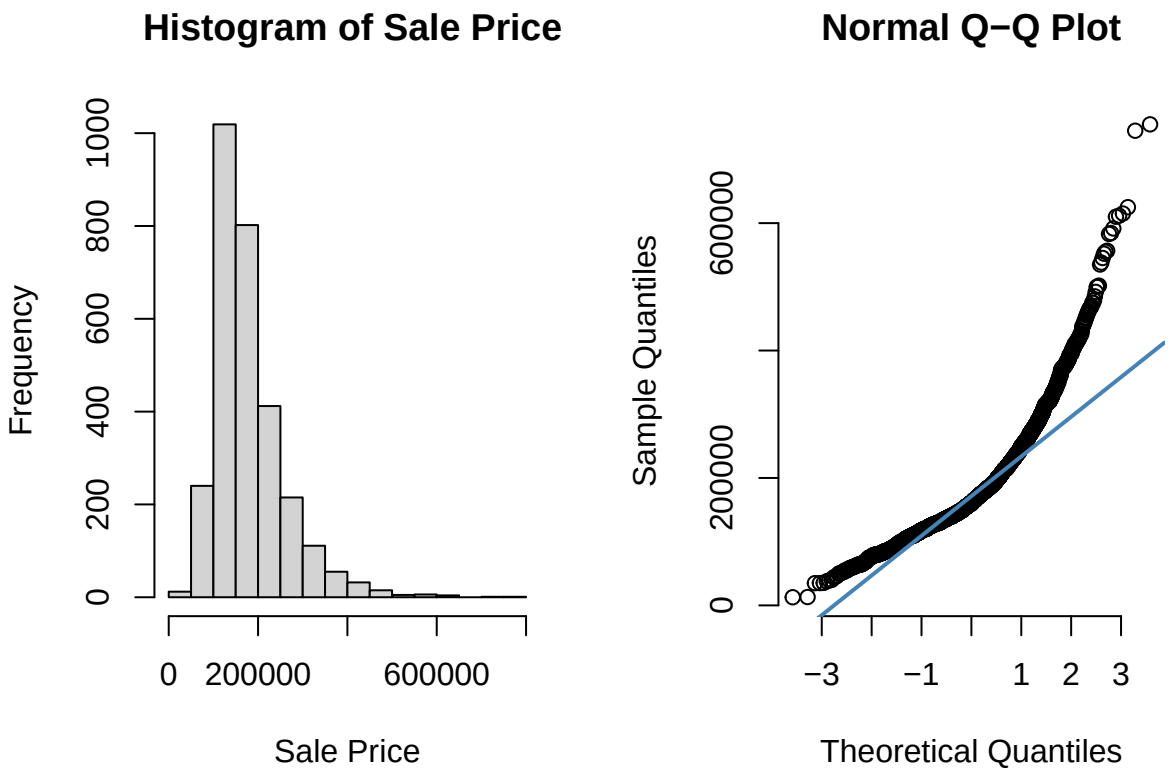

```
ggplot(data=housing,mapping = aes(x=Garage_Area,y=Sale_Price))+
  geom_point()
```



Preprocessing of the data

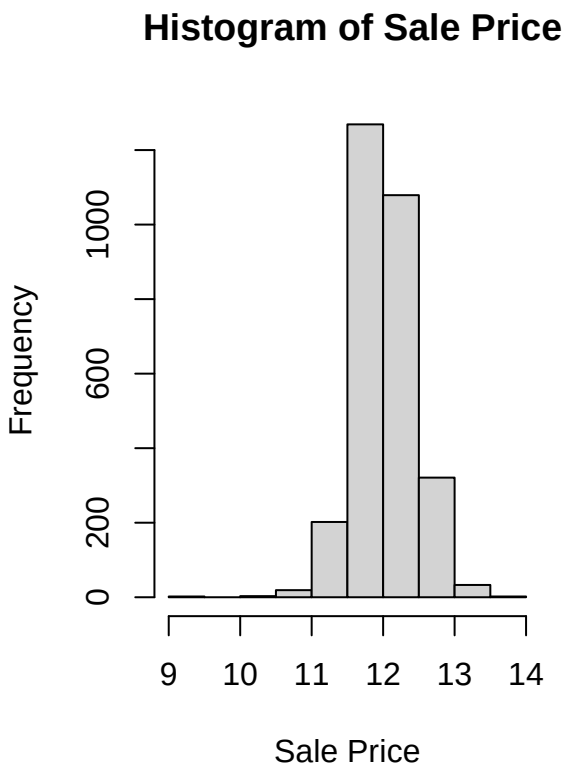
This is a histogram of the sale prices of our Chicago properties. We can see that some properties have really high sale prices which are skewing the histogram of sale price. In order to improve the predictive performance of our model, we will do the log transformation of the sale price of the houses in order to make them normally distributed. QQplot also confirms the skewness in sale price of the houses. We can see the deviations of the sale prices from the qqline which signifies that sale price is not normally distributed.

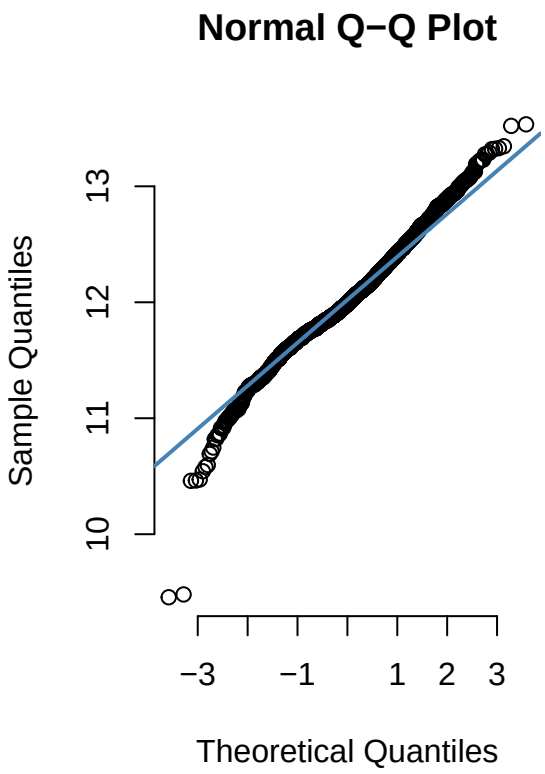
```
par(mfrow=c(1,2))
hist(housing$Sale_Price,xlab = "Sale Price",main=" Histogram of Sale Price")
qqnorm(housing$Sale_Price,pch = 1, frame = FALSE)
qqline(housing$Sale_Price, col = "steelblue", lwd = 2)
```



Transforming the sale price

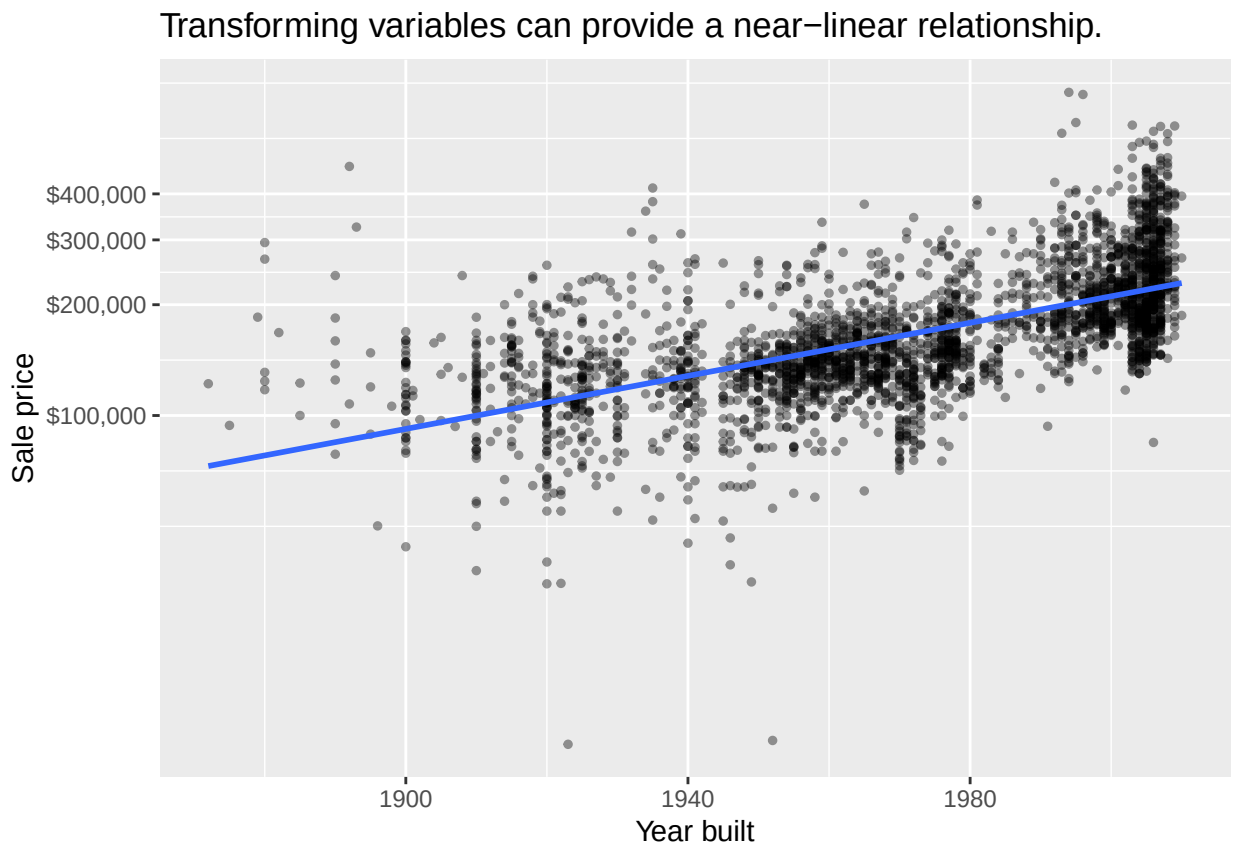
```
par(mfrow=c(1,2))
hist(log(housing$Sale_Price),xlab = "Sale Price",
     main=" Histogram of Sale Price")
qqnorm(log(housing$Sale_Price),pch = 1, frame = FALSE)
qqline(log(housing$Sale_Price), col = "steelblue", lwd = 2)
```






```
ggplot(housing, aes(Year_Built, Sale_Price)) +
  geom_point(size = 1, alpha = .4) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_y_log10("Sale price", labels = scales::dollar,
                breaks = seq(0, 400000, by = 100000)) +
  xlab("Year built") +
  ggtitle(paste("Transforming variables can provide a near-linear relationship."))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Selecting the variables via Correlation

As, we saw there are 80 variables in the data set. We cannot include all of them in model as some of them are categorical which need encoding and that would make the dimension of the dataset high which might result in to multicollinearity.

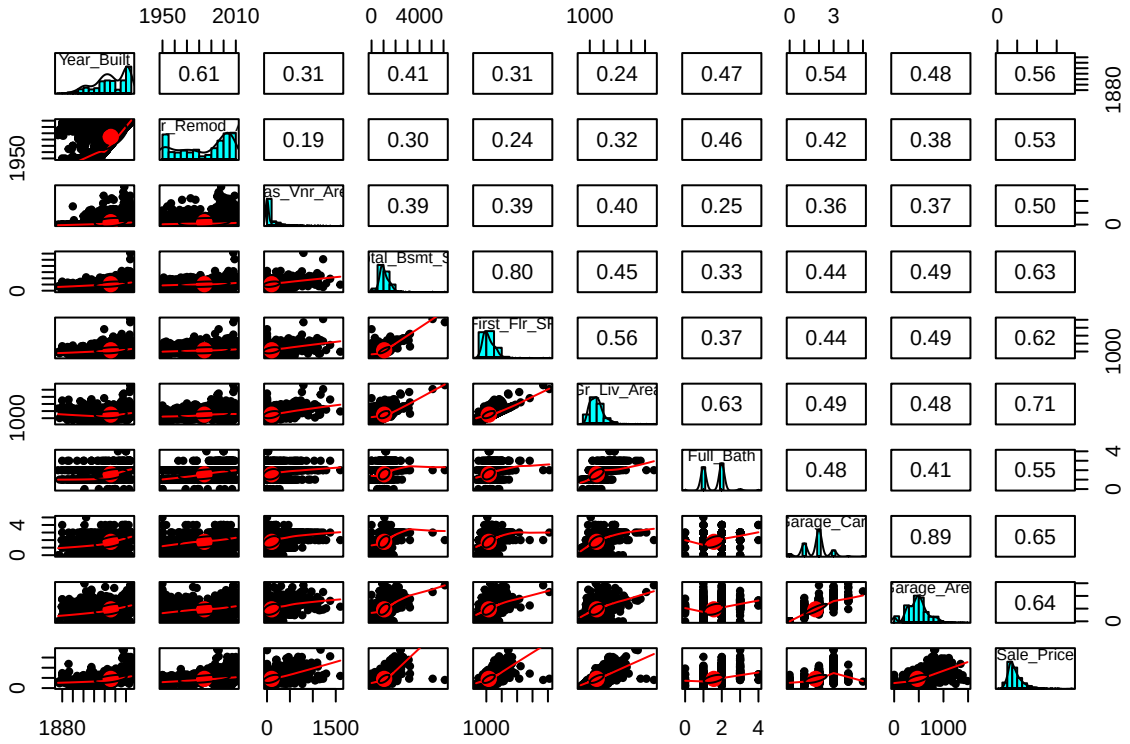
So, we will try selecting some variables which might help us in predicting the sale price of the house. We will use linear correlation technique to determine the variables having strong correlation with the sale price.

```
#selecting numeric variables
num_cols <- unlist(lapply(housing, is.numeric))# Identify numeric columns
#correlation matrix of numerical variables
correlations <- cor(housing[ , num_cols])
#variables having correlation greater than 0.5
imp_correations <- correlations>0.5
```

From the correlation matrix, we got to know the important variables which affect the sale price of the house significantly.

FRom the output, we can see that sale price is highly correlated with both garage area and garage cars as we saw that above in the scatterplot. Also, ground living area plays a important role in determining the sale price of the house.

```
pairs.panels(housing[,c("Year_Built","Year_Remod_Add","Mas_Vnr_Area",
"Total_Bsmt_SF","First_Flr_SF","Gr_Liv_Area","Full_Bath","Garage_Cars","Garage_Area",
"Sale_Price")])
```



Model Building

Now, we have some factors which affect the sale price of the house and we will be using them to predict the sale price of the house.

Splitting the data

```
#setting the seed
set.seed(123)
#splitting the data
chicago_split <- initial_split(data = housing,split=0.7,strata=Sale_Price)
#training set
Chicago_train <- training(chicago_split)
#testing set
Chicago_test <- testing(chicago_split)
```

preprocessing steps

In the pairs.panels(), we saw some variables were having nearly correlation with the sale price of the house, so we selected some variables out of them which explains independently sale price of the house As, it is redundant to keep the variable which are doing the same job in explaining the target variable.

Selected variables were:

- Year built: Date the house was built
- Total basement: Total square feet of basement area
- Ground living area
- Garage Area: Size of garage

We will log transform our response variable as discussed earlier, center and scale the data in order to have same scale of the variables.

```
Chicago_blueprint <- recipe(Sale_Price~Year_Built+Total_Bsmt_SF+Gr_Liv_Area+
                              Garage_Area,
                            data = Chicago_train) %>%
  step_log(all_outcomes())%>%
  step_center(all_numeric(), -all_outcomes()) %>%
  step_scale(all_numeric(), -all_outcomes())
```

```
#preparing the data
prepare_chicago <- prep(Chicago_blueprint, training = Chicago_train)
#baking the data
baked_chicago <- bake(prepare_chicago, new_data = Chicago_train)
```

Cross Validating the model

Cross validation is the technique where we select the paramters of the model by training and testing the data on different folds of the data. K-fold cross validation will divide the data in to 10 folds and will use one fold for validating the model and use remaining folds to train the data. This will let us know how our model will perform in future.

We will include preprocessing steps as well during cross validating the model.

```
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5
)

#Building the cross validated model
lm_model<- train(Chicago_blueprint,
  data = Chicago_train,
  method = "lm",
  trControl = cv
)
```

From the output of the model, we can see that our model has been repeatedly cross validated and has an R square of 74% meaning our variables explain 74 percent of the variability in the sale price of a house. Our residuals also look normally distributed from 5 summary statistics.

AAll the varibles turns out ot be significant in predicting sale price of the house. And, overall p value of the model is also significant. RMSE of the model is 0.211 which means our model will mispredict the sale price of the house price on average by 0.211. This RMSE is right now on log scale, we will convert it back on normal scale for interpretation.

```
lm_model
```

```
## Linear Regression
##
## 2197 samples
##   80 predictor
##
## Recipe steps: log, center, scale
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1976, 1978, 1977, 1977, 1977, 1976, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2065356  0.7505782  0.1345534
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
summary(lm_model)
```

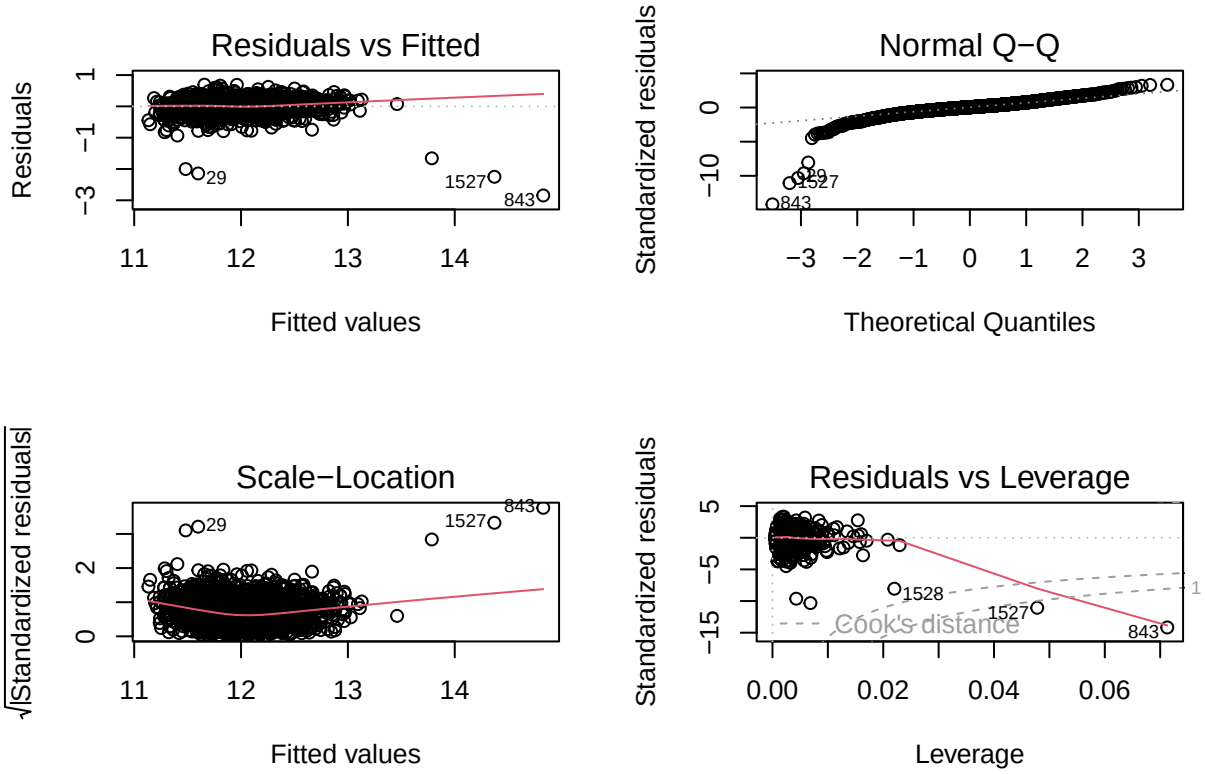
```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.84569 -0.08372  0.01151  0.10047  0.69330
##
## Coefficients:
##                Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   12.018561   0.004444 2704.19 <0.0000000000000002 ***
## Year_Built     0.143394   0.005174   27.71 <0.0000000000000002 ***
## Total_Bsmt_SF  0.078846   0.005456   14.45 <0.0000000000000002 ***
## Gr_Liv_Area    0.176291   0.005337   33.03 <0.0000000000000002 ***
## Garage_Area    0.074960   0.005759   13.02 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2083 on 2192 degrees of freedom
## Multiple R-squared:  0.7432, Adjusted R-squared:  0.7428
## F-statistic:  1586 on 4 and 2192 DF,  p-value: < 0.00000000000000022
```

Diagnostic plots

Here we will test or linear model assumptions.

1st plot: In the first plot, we can see random pattern in our errors. There is homoscedasticity in our error plot. There is a bit non linearity in our data. 2nd plot: This plot is used to check the normality of our errors. We can see little bit right skewness in our data. Overall, it is normalish. 3rd plot: This plot also checks the randomness in our errors. Turns out, there is no pattern observed in the plot. 4th plot: This plot is used to check the outliers in our data. 1537 and 1328 observation were turned out to be outliers which have to be diagnosed in order to see if they are problematic for us.

```
par(mfrow = c(2,2))
plot(lm_model$finalModel)
```



Testing the model

We will do same preprocessing steps on our test data, as we did on our training data. And, then we will bake our data and use it for testing the model.

```
baked_chicagotest <- bake(prepare_chicago, new_data = Chicago_test)
```

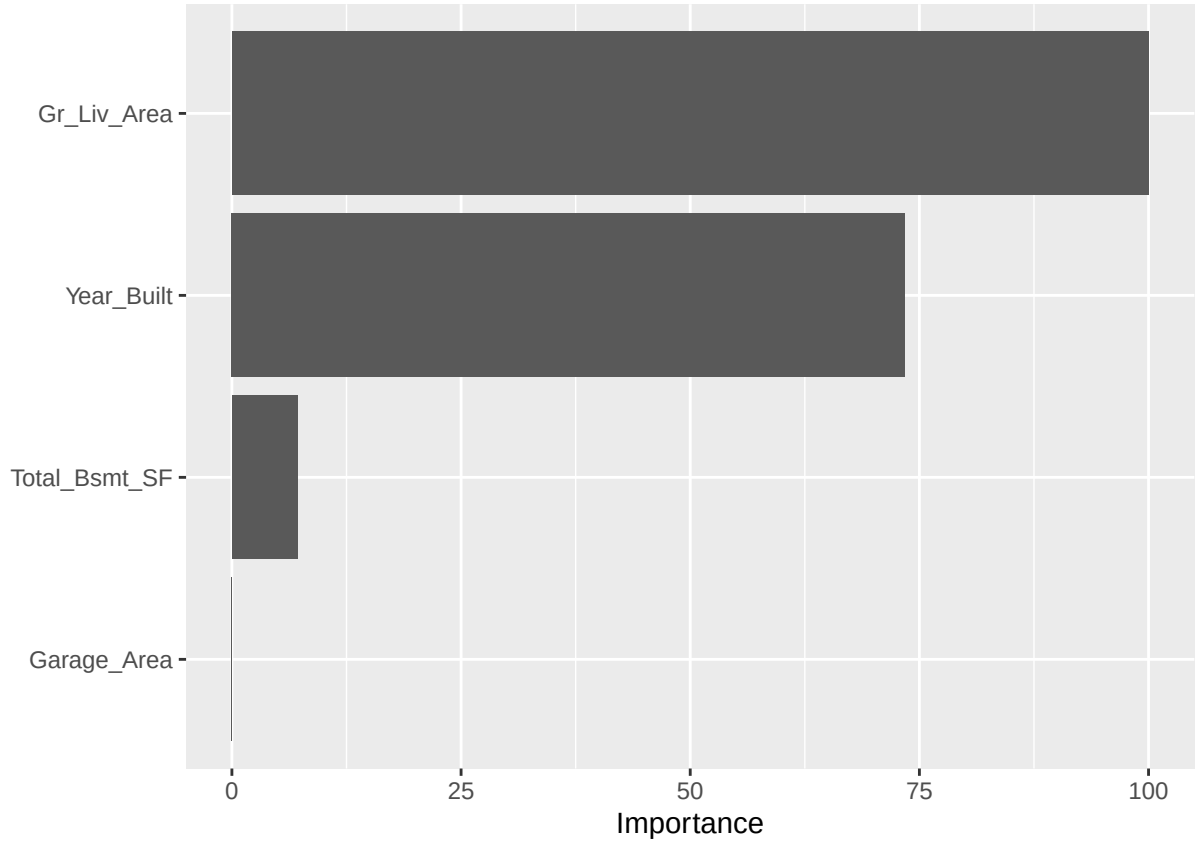
From the output, we can see that our model on average will predict the sale prices of the houses off by $198,953.

```
predict_chicagotest <- predict(lm_model,baked_chicagotest)

rmse(Chicago_test$Sale_Price, predict_chicagotest)
```

```
## [1] 198952.6
```

This is variable importance plot where we can see which variables plays an important role in model in predicting sale price of the houses. Most important variable comes out to be ground living area which plays most important role in determining sale price of the house followed by year built, total basement square footage and

```
vip(lm_model)
```



Random Forest

```
set.seed(123)
ind <- sample(2, nrow(housing), replace = TRUE, prob = c(0.7, 0.3))
train <- housing[ind==1,]
test <- housing[ind==2,]
```

```
model <- randomForest(Sale_Price ~ ., data = train, proximity = TRUE)

model
```

```
##
## Call:
##  randomForest(formula = Sale_Price ~ ., data = train, proximity = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 668054861
##                     % Var explained: 89.46
```

The random forest is build to predict the sales price and uses regression. The number of trees used by default by the model is 500. For further analysis we will be checking to see if 500 trees is enough for optimal regression. The models tells the number of variables that were consider at each split i.e. 26. Regression trees have a default setting of the number of variables divided by 3. As, we don't know if 26 is the best value, we will fiddle with this parameter later on.

```
sqrt(668054861)
```

```
## [1] 25846.76
```

Here is the MSE is 668054861. And the RMSE is 25846.76 The % variance explained by the model is a measure of how well OOB predictions explain the target variance of the training set which is 89.46 % in our case. It means the goodness of the outside prediction explained targeted variance in the training data set.

To test this hypothesis, we will make a random forest with 1000 trees

```
model1 <- randomForest(Sale_Price ~ ., data = housing, ntree = 1000, proximity = TRUE)

model1
```

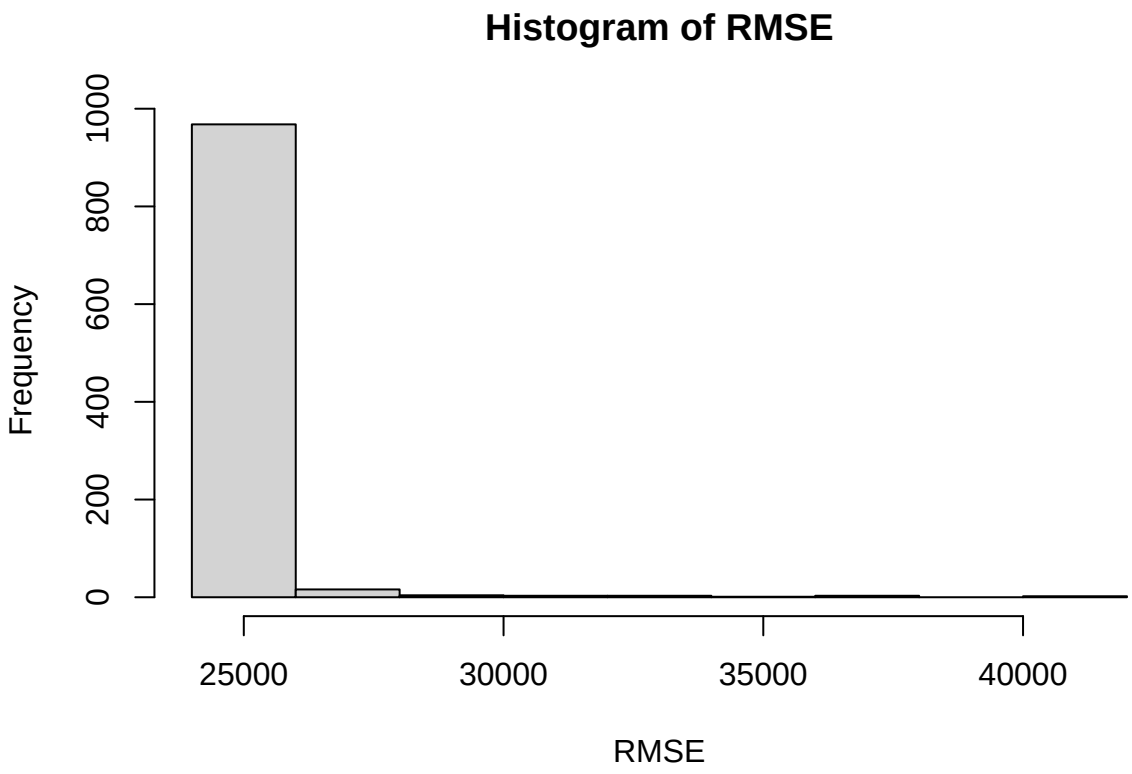
```
##
## Call:
##  randomForest(formula = Sale_Price ~ ., data = housing, ntree = 1000,      proximity = TRUE)
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 588928625
##                     % Var explained: 90.77
```

```
sqrt(588928625)
```

```
## [1] 24267.85
```

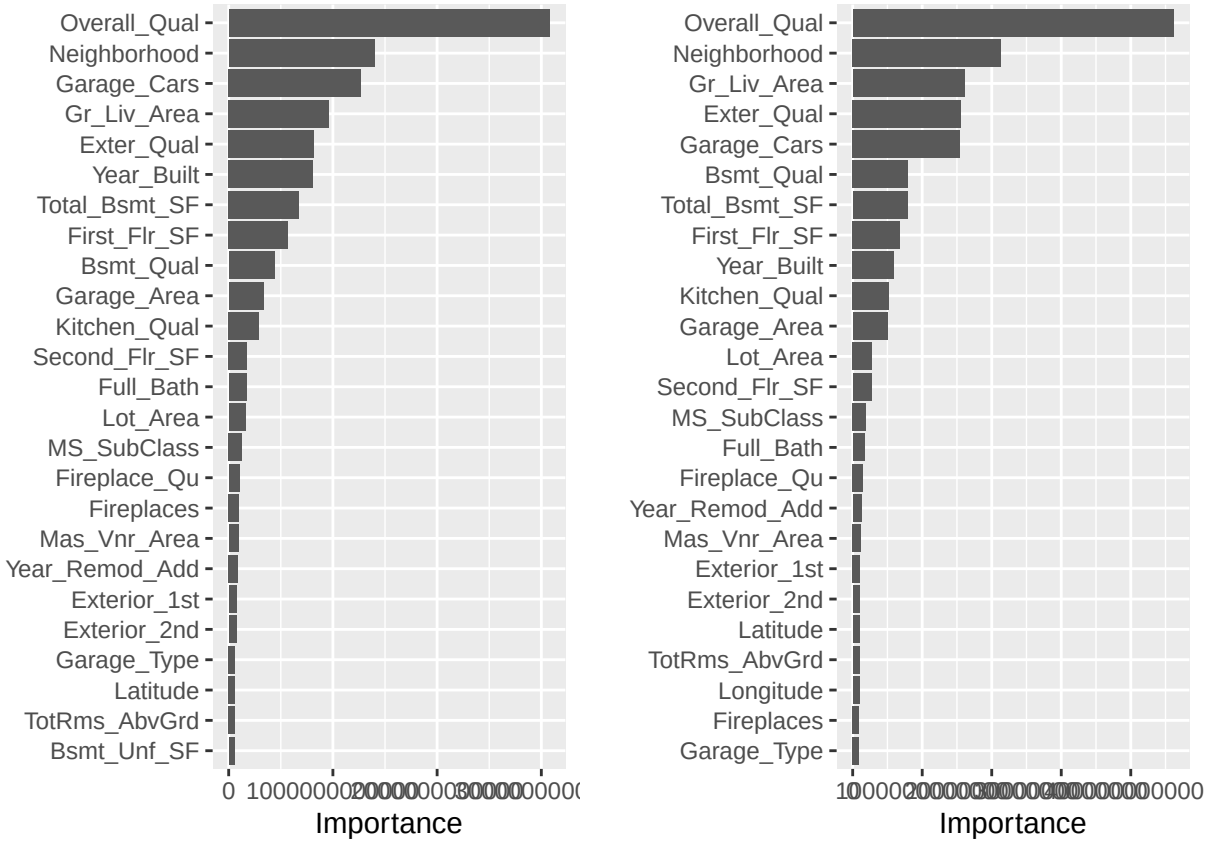
Here is the MSE is 588928625 And the RMSE is 24267.85 The % variance explained by the model is a measure of how well OOB predictions explain the target variance of the training set which is 90.77 % in our case. It means the goodness of the outside prediction explained targeted variance in the training data set. Ther is not much difference in RMSE by increasing number of trees.

```
RMSE <- sqrt(model1$mse)

hist(RMSE)
```



Currently, the best random forest model we have found uses mtry = 26, terminal node size of 26 observations, and a sample size of 80%. We see that our expected error ranges between 24000 to 26000

```
vip_m <- vip::vip(model, num_features = 25, importance  = TRUE)


vip_m1 <- vip::vip(model1, num_features = 25, importance  = TRUE)

gridExtra::grid.arrange(vip_m, vip_m1, nrow = 1)
```



Here, are the Top 25 Important Variables from two models. Some of the variables are common in both of them like "Overall_Qual", "Neighborhood", "Gr_Liv_Area", "Garage_Area", etc.

Gradient boost for regression

When gradient boost is used to predict the continuous variable, it is called gradient boost for regression, which in this case is sale price. Gradient boost builds decision tree based on the errors of previous tree and then scales the results and continues to build the tree until adding additional trees significantly reduce the size of the residuals.

Firstly, it is going to predict the average sale price of the houses and then it builds the tree based on errors from first tree. The errors from previous trees were difference of observed values and predicted values. A new tree is made using all predictors to predict the residuals from previous tree. Learning rate in this model plays an important role in scaling the tree. It takes step in right direction in order to reduce the error. gradient boost will continously improve on errors made by previous tree until adding other trees reduce the error rate in predicting the sale price of the house.

Building gradient boost regression model

We will gradient boost model to predict the sale price of the house using 5000 trees and will cross validate the model using 10 fold cross validation.

```
set.seed(123)  # for reproducibility
chicago_gbm <- gbm(
  formula = Sale_Price ~ .,
```

```
  data = Chicago_train,
  distribution = "gaussian",  # SSE loss function
  n.trees = 5000,
  shrinkage = 0.1,
  interaction.depth = 3,
  n.minobsinnode = 10,
  cv.folds = 10
)
```

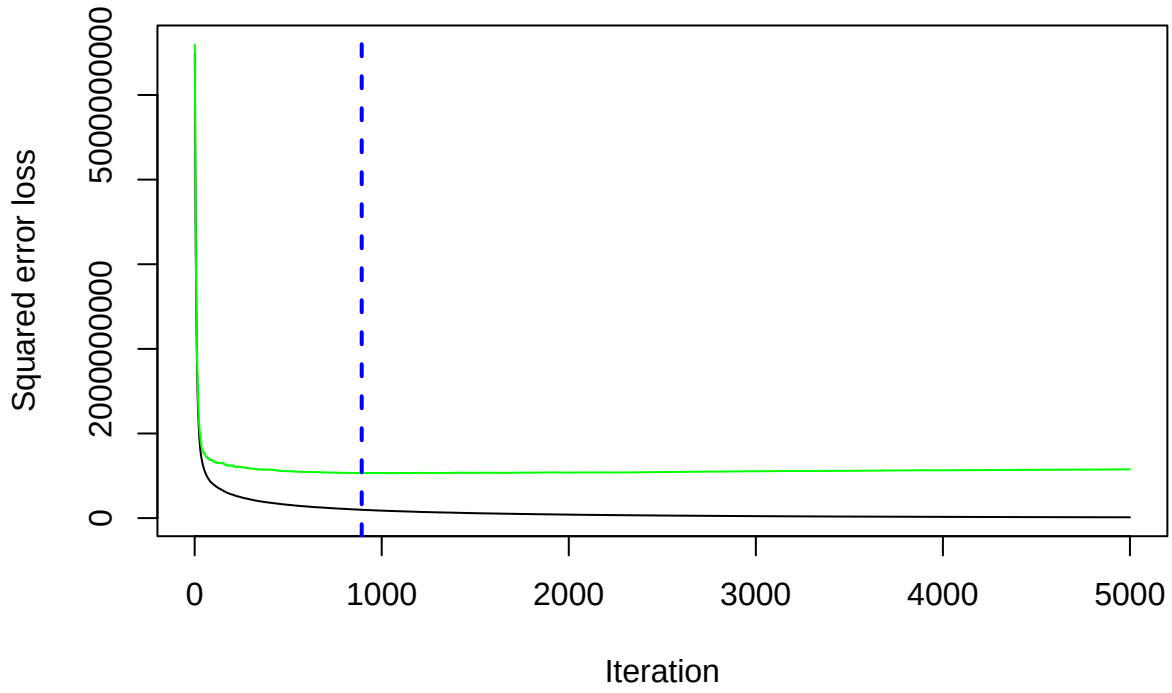
```
# find index for number trees with minimum CV error
best <- which.min(chicago_gbm$cv.error)
```

```
sqrt(chicago_gbm$cv.error[best])
```

```
## [1] 23034.56
```

Here, we see our model used 893 trees in order to predict the sale price of the house and our predictions for sale price will go off by 23000 on average, which is significant improvement in comparison to other model results.

```
gbm.perf(chicago_gbm, method = "cv")
```

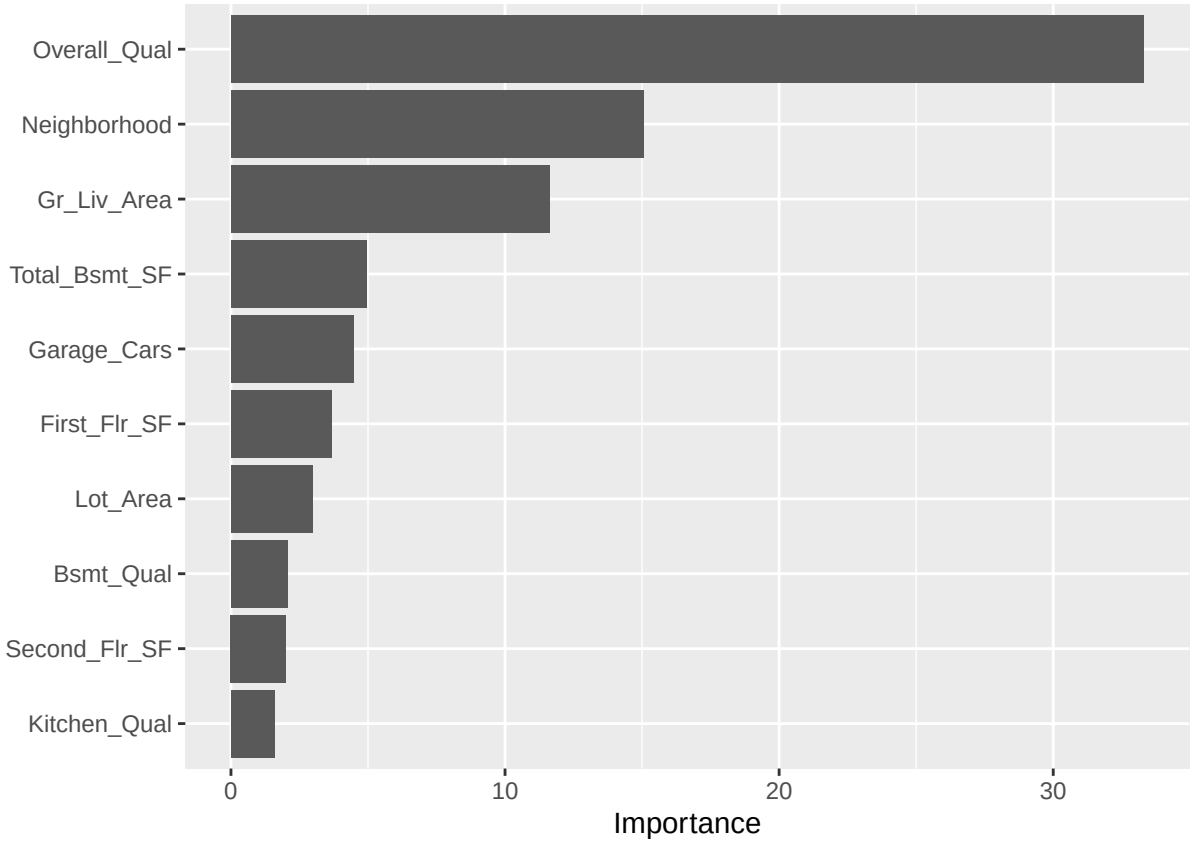



```
## [1] 893
```

Feature Importance

We can check here which variables were most important in determining sale price of the houses from top to bottom in descending order. Most important varaible turns out to be over all quality of the house, neighborhood, ground living area, nd so on.

```
vip(chicago_gbm)
```



Predicting the sale price using gbm model

We will make predictions on new data using our gbm model. As, we were having labelled data so we can check the error rate off model on unseen data. So, RMSE on the test data was nearly 20000 using 893 trees.

```
chicago_predict_gbm <- predict(chicago_gbm,Chicago_test)
```

```
## Using 893 trees...
```

```
rmse(Chicago_test$Sale_Price,chicago_predict_gbm)
```

```
## [1] 20564.01
```