COMPSCIX 415.2 Homework 7

Sanatan Das

March 17, 2018

Contents

Code and Documents Git Repository

All the work can be found in the below Git repository location:

https://github.com/sanatanonline/compscix-415-2-assignments

Load packages (prerequisites to run the code in this document)

```
library(tidyverse)
library(broom)
```

Analysis of Ames Housing dataset and predicting the price

Exercise 1

Load the train.csv dataset into R. How many observations and columns are there?

Answer

```
# Read from train.csv file
train <- read_csv("C:/view/opt/apps/git/R/compscix-415-2-assignments/train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Id = col_integer(),
##   MSSubClass = col_integer(),
##   LotFrontage = col_integer(),
##   LotArea = col_integer(),
##   OverallQual = col_integer(),
##   OverallCond = col_integer(),
##   YearBuilt = col_integer(),
##   YearRemodAdd = col_integer(),
##   MasVnrArea = col_integer(),
##   BsmtFinSF1 = col_integer(),
##   BsmtFinSF2 = col_integer(),
##   BsmtUnfSF = col_integer(),
##   TotalBsmtSF = col_integer(),
##   `1stFlrSF` = col_integer(),
##   `2ndFlrSF` = col_integer(),
##   LowQualFinSF = col_integer(),
##   GrLivArea = col_integer(),
##   BsmtFullBath = col_integer(),
##   BsmtHalfBath = col_integer(),
##   FullBath = col_integer()
##   # ... with 18 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
# glimpse train
glimpse(train)
```

```
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60,...
## $ MSZoning      <chr> "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RL", ...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, ...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10...
## $ Street        <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", ...
## $ Alley         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ LotShape      <chr> "Reg", "Reg", "IR1", "IR1", "IR1", "IR1", "Reg",...
## $ LandContour   <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl",...
## $ Utilities     <chr> "AllPub", "AllPub", "AllPub", "AllPub", "AllPub"...
## $ LotConfig     <chr> "Inside", "FR2", "Inside", "Corner", "FR2", "Ins...
## $ LandSlope     <chr> "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl",...
## $ Neighborhood  <chr> "CollgCr", "Veenker", "CollgCr", "Crawfor", "NoR...
## $ Condition1    <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm",...
## $ Condition2    <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", ...
## $ BldgType      <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", ...
## $ HouseStyle    <chr> "2Story", "1Story", "2Story", "2Story", "2Story"...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, ...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, ...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, ...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, ...
## $ RoofStyle     <chr> "Gable", "Gable", "Gable", "Gable", "Gable", "Ga...
## $ RoofMatl      <chr> "CompShg", "CompShg", "CompShg", "CompShg", "Com...
## $ Exterior1st   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Sdng", "Vin...
## $ Exterior2nd   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Shng", "Vin...
## $ MasVnrType    <chr> "BrkFace", "None", "BrkFace", "None", "BrkFace",...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, ...
## $ ExterQual     <chr> "Gd", "TA", "Gd", "TA", "Gd", "TA", "Gd", "TA", ...
## $ ExterCond     <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ Foundation    <chr> "PConc", "CBlock", "PConc", "BrkTil", "PConc", "...
## $ BsmtQual      <chr> "Gd", "Gd", "Gd", "TA", "Gd", "Gd", "Ex", "Gd", ...
## $ BsmtCond      <chr> "TA", "TA", "TA", "Gd", "TA", "TA", "TA", "TA", ...
## $ BsmtExposure  <chr> "No", "Gd", "Mn", "No", "Av", "No", "Av", "Mn", ...
## $ BsmtFinType1  <chr> "GLQ", "ALQ", "GLQ", "ALQ", "GLQ", "GLQ", "GLQ",...
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851,...
## $ BsmtFinType2  <chr> "Unf", "Unf", "Unf", "Unf", "Unf", "Unf", "Unf",...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ BsmtUnfSF     <int> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140,...
## $ TotalBsmtSF   <int> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952,...
## $ Heating       <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", ...
## $ HeatingQC     <chr> "Ex", "Ex", "Ex", "Gd", "Ex", "Ex", "Ex", "Ex", ...
## $ CentralAir    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y"...
## $ Electrical    <chr> "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SB...
## $ `1stFlrSF`    <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022...
## $ `2ndFlrSF`    <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, ...
## $ LowQualFinSF  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ GrLivArea     <int> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, ...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, ...
## $ BsmtHalfBath  <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ FullBath      <int> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, ...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, ...
```

```
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, ...
## $ KitchenQual   <chr> "Gd", "TA", "Gd", "Gd", "Gd", "TA", "Gd", "TA", ...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5,...
## $ Functional    <chr> "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ",...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, ...
## $ FireplaceQu   <chr> NA, "TA", "TA", "Gd", "TA", NA, "Gd", "TA", "TA"...
## $ GarageType    <chr> "Attchd", "Attchd", "Attchd", "Detchd", "Attchd"...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, ...
## $ GarageFinish  <chr> "RFn", "RFn", "RFn", "Unf", "RFn", "Unf", "RFn",...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, ...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205...
## $ GarageQual    <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ GarageCond    <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ PavedDrive    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y"...
## $ WoodDeckSF    <int> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, ...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, ...
## $ `3SsnPorch`   <int> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ ScreenPorch   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PoolQC        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ Fence         <chr> NA, NA, NA, NA, NA, "MnPrv", NA, NA, NA, NA, NA,...
## $ MiscFeature   <chr> NA, NA, NA, NA, NA, "Shed", NA, "Shed", NA, NA, ...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0,...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, ...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, ...
## $ SaleType      <chr> "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", ...
## $ SaleCondition <chr> "Normal", "Normal", "Normal", "Abnorml", "Normal...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, ...
```

So there are 1460 observations with 81 columns (variables)

Exercise 2

Normally at this point you would spend a few days on EDA, but for this homework we will get right to fitting some linear regression models. Our first step is to randomly split the data into train and test datasets. We will use a 70/30 split. There is an R package that will do the split for you, but let's get some more practice with R and do it ourselves by filling in the blanks in the code below.

```r
# load packages
library(tidyverse)
library(broom)
# When taking a random sample, it is often useful to set a seed so that
# your work is reproducible. Setting a seed will guarantee that the same
# random sample will be generated every time, so long as you always set the
# same seed beforehand
set.seed(29283)
# This data already has an Id column which we can make use of.
# Let's create our training set using sample_frac. Fill in the blank.
train_set <- train %>% sample_frac(____)
# let's create our testing set using the Id column. Fill in the blanks.
test_set <- train %>% filter(!(____ %in% ____$Id))
```

Answer

Let's fill in the blanks.

```r
# When taking a random sample, it is often useful to set a seed so that
# your work is reproducible. Setting a seed will guarantee that the same
# random sample will be generated every time, so long as you always set the
# same seed beforehand
set.seed(29283)
# This data already has an Id column which we can make use of.
# Let's create our training set using sample_frac. Fill in the blank.
train_set <- train %>% sample_frac(0.7)
# Print train set
train_set
```

```
## # A tibble: 1,022 x 81
##       Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape
##    <int>      <int> <chr>          <int>   <int> <chr>  <chr> <chr>
##  1    22         45 RM                57    7449 Pave   Grvl  Reg
##  2   637         30 RM                51    6120 Pave   <NA>  Reg
##  3   121         80 RL                NA   21453 Pave   <NA>  IR1
##  4   575         80 RL                70   10500 Pave   <NA>  Reg
##  5  1423        120 RM                37    4435 Pave   <NA>  Reg
##  6  1169         70 RL               120   13728 Pave   <NA>  Reg
##  7  1261         60 RL                NA   24682 Pave   <NA>  IR3
##  8  1319         20 RL                NA   14781 Pave   <NA>  IR2
##  9   116        160 FV                34    3230 Pave   Pave  Reg
## 10  1125         80 RL                NA    9125 Pave   <NA>  IR1
## # ... with 1,012 more rows, and 73 more variables: LandContour <chr>,
## #   Utilities <chr>, LotConfig <chr>, LandSlope <chr>, Neighborhood <chr>,
## #   Condition1 <chr>, Condition2 <chr>, BldgType <chr>, HouseStyle <chr>,
## #   OverallQual <int>, OverallCond <int>, YearBuilt <int>,
## #   YearRemodAdd <int>, RoofStyle <chr>, RoofMatl <chr>,
## #   Exterior1st <chr>, Exterior2nd <chr>, MasVnrType <chr>,
## #   MasVnrArea <int>, ExterQual <chr>, ExterCond <chr>, Foundation <chr>,
## #   BsmtQual <chr>, BsmtCond <chr>, BsmtExposure <chr>,
## #   BsmtFinType1 <chr>, BsmtFinSF1 <int>, BsmtFinType2 <chr>,
## #   BsmtFinSF2 <int>, BsmtUnfSF <int>, TotalBsmtSF <int>, Heating <chr>,
## #   HeatingQC <chr>, CentralAir <chr>, Electrical <chr>, `1stFlrSF` <int>,
## #   `2ndFlrSF` <int>, LowQualFinSF <int>, GrLivArea <int>,
## #   BsmtFullBath <int>, BsmtHalfBath <int>, FullBath <int>,
## #   HalfBath <int>, BedroomAbvGr <int>, KitchenAbvGr <int>,
## #   KitchenQual <chr>, TotRmsAbvGrd <int>, Functional <chr>,
## #   Fireplaces <int>, FireplaceQu <chr>, GarageType <chr>,
## #   GarageYrBlt <int>, GarageFinish <chr>, GarageCars <int>,
## #   GarageArea <int>, GarageQual <chr>, GarageCond <chr>,
## #   PavedDrive <chr>, WoodDeckSF <int>, OpenPorchSF <int>,
## #   EnclosedPorch <int>, `3SsnPorch` <int>, ScreenPorch <int>,
## #   PoolArea <int>, PoolQC <chr>, Fence <chr>, MiscFeature <chr>,
## #   MiscVal <int>, MoSold <int>, YrSold <int>, SaleType <chr>,
## #   SaleCondition <chr>, SalePrice <int>
```

```r
# let's create our testing set using the Id column. Fill in the blanks.
test_set <- train %>% filter(!(train$Id %in% train_set$Id))
# Print test set
test_set
```

```
## # A tibble: 438 x 81
##       Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape
```

```
##    <int>      <int> <chr>          <int>   <int> <chr>  <chr> <chr>
##  1     1         60 RL                65    8450 Pave   <NA>  Reg
##  2     2         20 RL                80    9600 Pave   <NA>  Reg
##  3     3         60 RL                68   11250 Pave   <NA>  IR1
##  4     4         70 RL                60    9550 Pave   <NA>  IR1
##  5    14         20 RL                91   10652 Pave   <NA>  IR1
##  6    23         20 RL                75    9742 Pave   <NA>  Reg
##  7    27         20 RL                60    7200 Pave   <NA>  Reg
##  8    38         20 RL                74    8532 Pave   <NA>  Reg
##  9    40         90 RL                65    6040 Pave   <NA>  Reg
## 10    42         20 RL               115   16905 Pave   <NA>  Reg
## # ... with 428 more rows, and 73 more variables: LandContour <chr>,
## #   Utilities <chr>, LotConfig <chr>, LandSlope <chr>, Neighborhood <chr>,
## #   Condition1 <chr>, Condition2 <chr>, BldgType <chr>, HouseStyle <chr>,
## #   OverallQual <int>, OverallCond <int>, YearBuilt <int>,
## #   YearRemodAdd <int>, RoofStyle <chr>, RoofMatl <chr>,
## #   Exterior1st <chr>, Exterior2nd <chr>, MasVnrType <chr>,
## #   MasVnrArea <int>, ExterQual <chr>, ExterCond <chr>, Foundation <chr>,
## #   BsmtQual <chr>, BsmtCond <chr>, BsmtExposure <chr>,
## #   BsmtFinType1 <chr>, BsmtFinSF1 <int>, BsmtFinType2 <chr>,
## #   BsmtFinSF2 <int>, BsmtUnfSF <int>, TotalBsmtSF <int>, Heating <chr>,
## #   HeatingQC <chr>, CentralAir <chr>, Electrical <chr>, `1stFlrSF` <int>,
## #   `2ndFlrSF` <int>, LowQualFinSF <int>, GrLivArea <int>,
## #   BsmtFullBath <int>, BsmtHalfBath <int>, FullBath <int>,
## #   HalfBath <int>, BedroomAbvGr <int>, KitchenAbvGr <int>,
## #   KitchenQual <chr>, TotRmsAbvGrd <int>, Functional <chr>,
## #   Fireplaces <int>, FireplaceQu <chr>, GarageType <chr>,
## #   GarageYrBlt <int>, GarageFinish <chr>, GarageCars <int>,
## #   GarageArea <int>, GarageQual <chr>, GarageCond <chr>,
## #   PavedDrive <chr>, WoodDeckSF <int>, OpenPorchSF <int>,
## #   EnclosedPorch <int>, `3SsnPorch` <int>, ScreenPorch <int>,
## #   PoolArea <int>, PoolQC <chr>, Fence <chr>, MiscFeature <chr>,
## #   MiscVal <int>, MoSold <int>, YrSold <int>, SaleType <chr>,
## #   SaleCondition <chr>, SalePrice <int>
```

Now, we have separated our train data set and test data set.

Exercise 3

Our target is called SalePrice. First, we can fit a simple regression model consisting of only the intercept (the average of SalePrice). Fit the model and then use the broom package to

- **take a look at the coefficient,**
- **compare the coefficient to the average value of SalePrice, and**
- **take a look at the R-squared.**

Use the code below and fill in the blanks.

```r
# Fit a model with intercept only
mod_0 <- lm(SalePrice ~ 1, data = _____)
# Double-check that the average SalePrice is equal to our model's coefficient
mean(train_set$SalePrice)
tidy(____)
# Check the R-squared
glance(____)
```

Answer

Let's fill in the blanks.

```
# Fit a model with intercept only
mod_0 <- lm(SalePrice ~ 1, data = train_set)
# Double-check that the average SalePrice is equal to our model's coefficient
mean(train_set$SalePrice)
```

```
## [1] 182176
```

```
tidy(mod_0)
```

```
##          term estimate std.error statistic p.value
## 1 (Intercept)   182176  2492.072  73.10222       0
```

```
# Check the R-squared
glance(mod_0)
```

```
##   r.squared adj.r.squared    sigma statistic p.value df    logLik      AIC
## 1         0             0 79668.37        NA      NA  1 -12983.57 25971.13
##        BIC     deviance df.residual
## 1 25980.99 6.480338e+12        1021
```

Exercise 4

Now fit a linear regression model using GrLivArea, OverallQual, and Neighborhood as the features. Don't forget to look at data_description.txt to understand what these variables mean. Ask yourself these questions before fitting the model:

- **What kind of relationship will these features have with our target?**
- **Can the relationship be estimated linearly?**
- **Are these good features, given the problem we are trying to solve?**

After fitting the model, output the coefficients and the R-squared using the broom package.

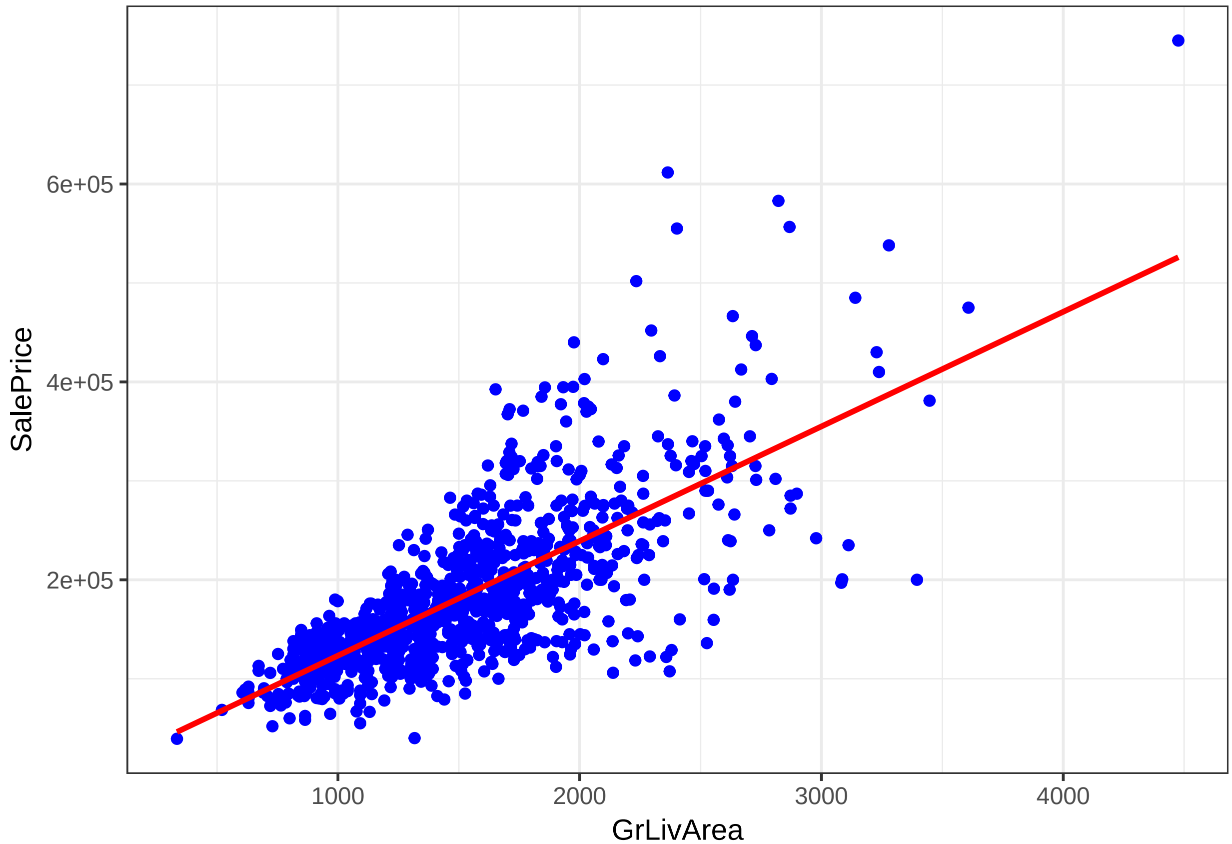
Answer these questions:

- **How would you interpret the coefficients on GrLivArea and OverallQual?**
- **How would you interpret the coefficient on NeighborhoodBrkSide?**
- **Are the features significant?**
- **Are the features practically significant?**
- **Is the model a good fit (to the training set)?**

Answer

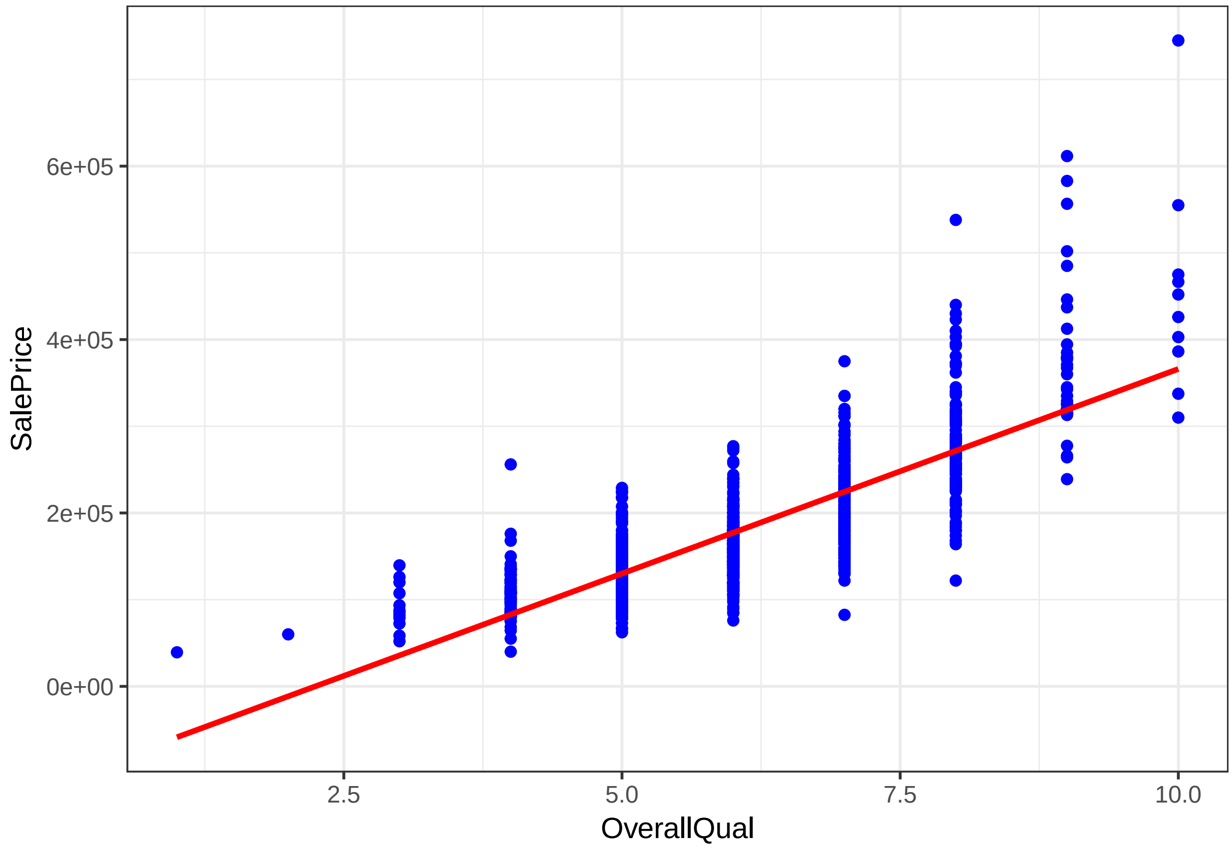
Let's plot the graph to see the relationship between SalePrice and GrLivArea, OverallQual, and Neighborhood.

```
ggplot(train_set, aes(x = GrLivArea, y = SalePrice)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  theme_bw()
```



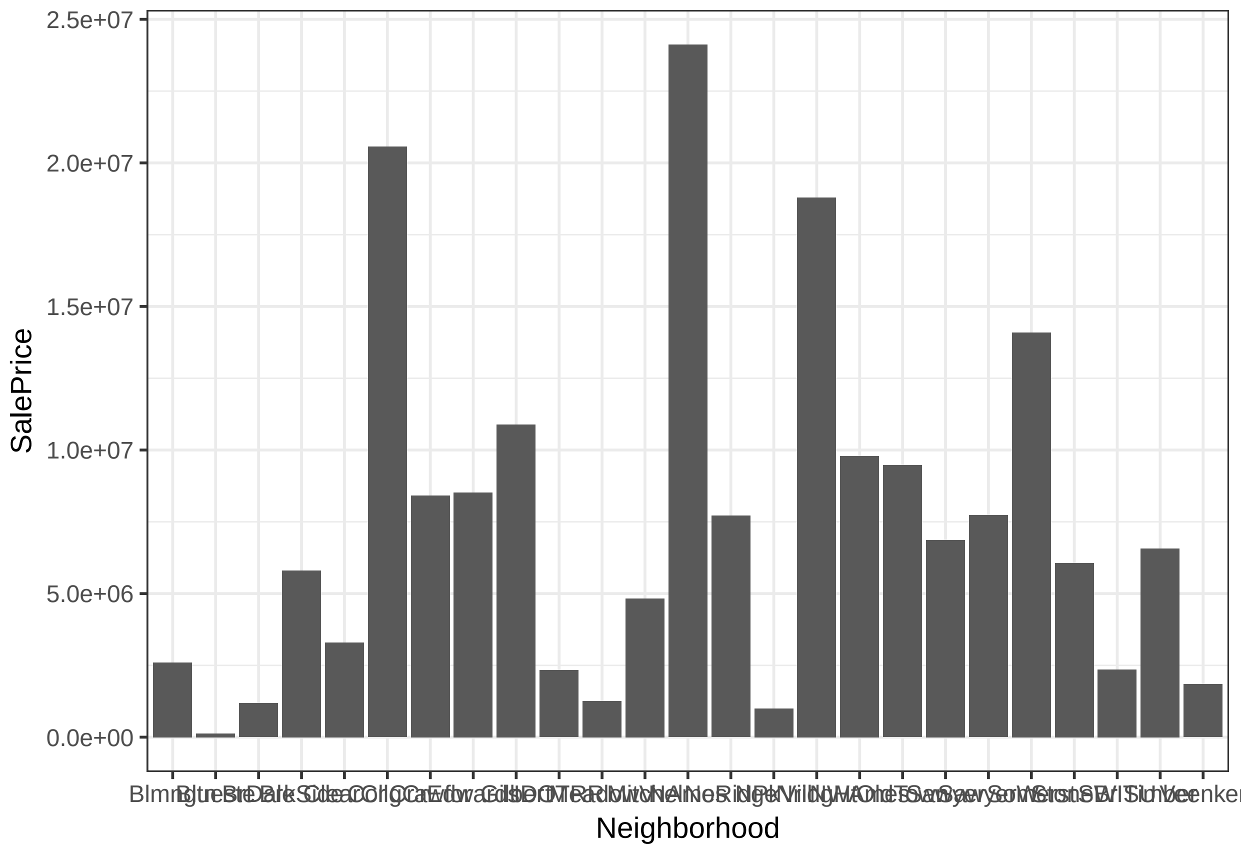
The relationship between SalePrice and GrLivArea is linear. It can be estimated linearly. It is an important feature to estimate as it impacts the SalePrice.

```
ggplot(train_set, aes(x = OverallQual, y = SalePrice)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  theme_bw()
```



The relationship between SalePrice and OverallQual is linear. It can be estimated linearly. It is an important feature to estimate as it impacts the SalePrice.

```
train_set %>%
ggplot() +
  geom_bar(aes(x = Neighborhood, y = SalePrice), stat = 'identity') +
  theme_bw()
```



There is a relationship between SalePrice and Neighborhood. But the Neighborhood is categorical, so no linear relationship with SalePrice. It is an important feature to estimate as it impacts the SalePrice. We will factor it and add to our regression model.

Let's create the models and see the values.

```
# Fit a model for GrLivArea
lm_1 <- lm(SalePrice ~ GrLivArea, data = train_set)
mean(train_set$SalePrice)
```

```
## [1] 182176
```

```
tidy(lm_1)
```

```
##          term estimate   std.error statistic       p.value
## 1 (Intercept) 7518.567 5334.586903   1.40940  1.590217e-01
## 2   GrLivArea  115.833    3.354991  34.52558 1.189915e-173
```

```
glance(lm_1)
```

```
##   r.squared adj.r.squared    sigma statistic       p.value df logLik   AIC
## 1 0.5388821     0.5384301 54125.85  1192.016 1.189915e-173  2 -12588 25182
##        BIC   deviance df.residual
## 1 25196.79 2.9882e+12        1020
```

```
# Fit a model for OverallQual
lm_2 <- lm(SalePrice ~ OverallQual, data = train_set)
mean(train_set$SalePrice)
```

```
## [1] 182176
```

```r
tidy(lm_2)
```

```
##          term  estimate std.error statistic       p.value
## 1 (Intercept) -105872.5  6893.466 -15.35839  4.801210e-48
## 2 OverallQual   47192.3  1102.649  42.79902 5.918164e-230
```

```r
glance(lm_2)
```

```
##   r.squared adj.r.squared    sigma statistic       p.value df    logLik
## 1 0.6423257      0.641975 47669.72  1831.756 5.918164e-230  2 -12458.19
##        AIC      BIC     deviance df.residual
## 1 24922.38 24937.17 2.317851e+12        1020
```

```r
# Fit a model for Neighborhood
train_set <- train_set %>% mutate(Neighborhood_fct = factor(Neighborhood, ordered = FALSE))
lm_3 <- lm(SalePrice ~ Neighborhood_fct, data = train_set)
mean(train_set$SalePrice)
```

```
## [1] 182176
```

```r
tidy(lm_3)
```

```
##                       term    estimate std.error  statistic      p.value
## 1              (Intercept)  200308.462  14880.25 13.4613631 4.487217e-38
## 2  Neighborhood_fctBlueste  -76308.462  55676.80 -1.3705612 1.708203e-01
## 3   Neighborhood_fctBrDale  -92908.462  21979.59 -4.2270339 2.585277e-05
## 4  Neighborhood_fctBrkSide  -76713.249  16812.68 -4.5628209 5.675519e-06
## 5  Neighborhood_fctClearCr   19853.672  20330.29  0.9765561 3.290259e-01
## 6  Neighborhood_fctCollgCr   -4491.700  15774.54 -0.2847437 7.758997e-01
## 7  Neighborhood_fctCrawfor    4905.221  17077.14  0.2872390 7.739890e-01
## 8  Neighborhood_fctEdwards  -69405.077  16300.50 -4.2578500 2.259005e-05
## 9  Neighborhood_fctGilbert   -9247.988  16490.05 -0.5608225 5.750446e-01
## 10  Neighborhood_fctIDOTRR  -94072.098  18768.65 -5.0121942 6.365074e-07
## 11 Neighborhood_fctMeadowV -103085.385  21043.85 -4.8985984 1.125722e-06
## 12 Neighborhood_fctMitchel  -44658.462  17727.84 -2.5191151 1.192047e-02
## 13   Neighborhood_fctNAmes  -54161.231  15455.33 -3.5043723 4.780249e-04
## 14 Neighborhood_fctNoRidge  135254.147  18616.48  7.2652908 7.499696e-13
## 15 Neighborhood_fctNPkVill  -59058.462  25152.21 -2.3480422 1.906702e-02
## 16 Neighborhood_fctNridgHt  118035.386  16438.06  7.1806164 1.355530e-12
## 17  Neighborhood_fctNWAmes  -12271.923  16636.63 -0.7376449 4.609038e-01
## 18 Neighborhood_fctOldTown  -72380.651  16134.44 -4.4860949 8.097746e-06
## 19  Neighborhood_fctSawyer  -63192.622  16703.04 -3.7833005 1.639801e-04
## 20 Neighborhood_fctSawyerW  -20425.787  16981.27 -1.2028425 2.293228e-01
## 21 Neighborhood_fctSomerst   34565.772  16413.31  2.1059604 3.545731e-02
## 22 Neighborhood_fctStoneBr  118543.907  19311.16  6.1386206 1.199918e-09
## 23   Neighborhood_fctSWISU  -52975.087  20033.15 -2.6443711 8.312879e-03
## 24  Neighborhood_fctTimber   52345.885  18224.51  2.8722792 4.161598e-03
## 25 Neighborhood_fctVeenker   63620.110  25152.21  2.5294039 1.157881e-02
```

```r
glance(lm_3)
```

```
##   r.squared adj.r.squared    sigma statistic       p.value df    logLik
## 1 0.5571452     0.5464847 53651.51  52.26259 1.011167e-157 25 -12567.35
##       AIC      BIC     deviance df.residual
## 1 25186.7 25314.87 2.869849e+12         997
```

```r
# Fit a model with all three variables
lm_4 <- lm(SalePrice ~ GrLivArea + OverallQual + Neighborhood_fct, data = train_set)
mean(train_set$SalePrice)
```

```
## [1] 182176
```

```r
tidy(lm_4)
```

```
##                        term     estimate    std.error  statistic
## 1               (Intercept) -45017.87483 12933.341808 -3.4807612
## 2                 GrLivArea     62.77735     3.006033 20.8837885
## 3               OverallQual  21692.23178  1353.714104 16.0242342
## 4  Neighborhood_fctBlueste -38288.88063 36531.907177 -1.0480942
## 5   Neighborhood_fctBrDale -43314.05372 14524.693991 -2.9820975
## 6  Neighborhood_fctBrkSide -14064.37052 11318.850018 -1.2425618
## 7  Neighborhood_fctClearCr  27839.00662 13561.346871  2.0528202
## 8  Neighborhood_fctCollgCr   4297.67432 10372.304467  0.4143413
## 9  Neighborhood_fctCrawfor   7423.05573 11371.511784  0.6527765
## 10 Neighborhood_fctEdwards -15284.11495 10994.287187 -1.3901870
## 11 Neighborhood_fctGilbert  -8357.55930 10894.173472 -0.7671586
## 12  Neighborhood_fctIDOTRR -32689.43085 12603.712743 -2.5936350
## 13 Neighborhood_fctMeadowV -14446.06504 14190.148622 -1.0180348
## 14 Neighborhood_fctMitchel   1922.31487 11788.608170  0.1630655
## 15   Neighborhood_fctNAmes  -7719.67883 10375.956174 -0.7439969
## 16 Neighborhood_fctNoRidge  47685.16790 12567.432633  3.7943444
## 17 Neighborhood_fctNPkVill -20240.71145 16548.664867 -1.2231024
## 18 Neighborhood_fctNridgHt  63872.80848 10880.456671  5.8704161
## 19  Neighborhood_fctNWAmes -12279.33299 11047.502893 -1.1115030
## 20 Neighborhood_fctOldTown -36107.07577 10849.170903 -3.3280954
## 21  Neighborhood_fctSawyer  -4121.92502 11252.369778 -0.3663162
## 22 Neighborhood_fctSawyerW  -5391.97074 11230.758221 -0.4801075
## 23 Neighborhood_fctSomerst  18700.96725 10772.212794  1.7360377
## 24 Neighborhood_fctStoneBr  65712.45881 12745.312907  5.1558137
## 25   Neighborhood_fctSWISU -45451.86707 13564.792586 -3.3507233
## 26  Neighborhood_fctTimber  27925.08619 11985.325857  2.3299397
## 27 Neighborhood_fctVeenker  54913.12768 16521.075497  3.3238228
##         p.value
## 1  5.216927e-04
## 2  1.337222e-80
## 3  1.389020e-51
## 4  2.948497e-01
## 5  2.932566e-03
## 6  2.143221e-01
## 7  4.035110e-02
## 8  6.787135e-01
## 9  5.140512e-01
## 10 1.647830e-01
## 11 4.431692e-01
## 12 9.636216e-03
## 13 3.089089e-01
## 14 8.705000e-01
## 15 4.570540e-01
## 16 1.569690e-04
## 17 2.215806e-01
```

```
## 18 5.917964e-09
## 19 2.666204e-01
## 20 9.064637e-04
## 21 7.142070e-01
## 22 6.312565e-01
## 23 8.286672e-02
## 24 3.045915e-07
## 25 8.363074e-04
## 26 2.000859e-02
## 27 9.203087e-04
```

```
glance(lm_4)
```

```
##   r.squared adj.r.squared   sigma statistic p.value df    logLik      AIC
## 1 0.8099927     0.8050277 35178.1  163.1401       0 27 -12134.95 24325.91
##        BIC     deviance df.residual
## 1 24463.93 1.231311e+12         995
```

How would you interpret the coefficients on GrLivArea and OverallQual?

Answer

How would you interpret the coefficient on NeighborhoodBrkSide?

Answer

Are the features significant?

Answer

Are the features practically significant?

Answer

Is the model a good fit (to the training set)?

Answer

Exercise 5

Evaluate the model on test_set using the root mean squared error (RMSE). Use the predict function to get the model predictions for the testing set.

Hint: use the sqrt() and mean() functions:

```
test_predictions <- predict(NAME_OF_YOUR_MODEL_HERE, newdata = test_set)
rmse <- sqrt(mean((___ - ___)^2))
```

Answer

Let's predict on the test data and evaluate the model.

```
test_set <- test_set %>% mutate(Neighborhood_fct = factor(Neighborhood, ordered = FALSE))
test_predictions <- predict(lm_4, newdata = test_set)
rmse <- sqrt(mean((test_set$SalePrice - mean(test_set$SalePrice))^2))
rmse
```

```
## [1] 78835.85
```

Exercise 7

One downside of the linear model is that it is sensitive to unusual values because the distance incorporates a squared term. Fit a linear model to the simulated data below, and visualise

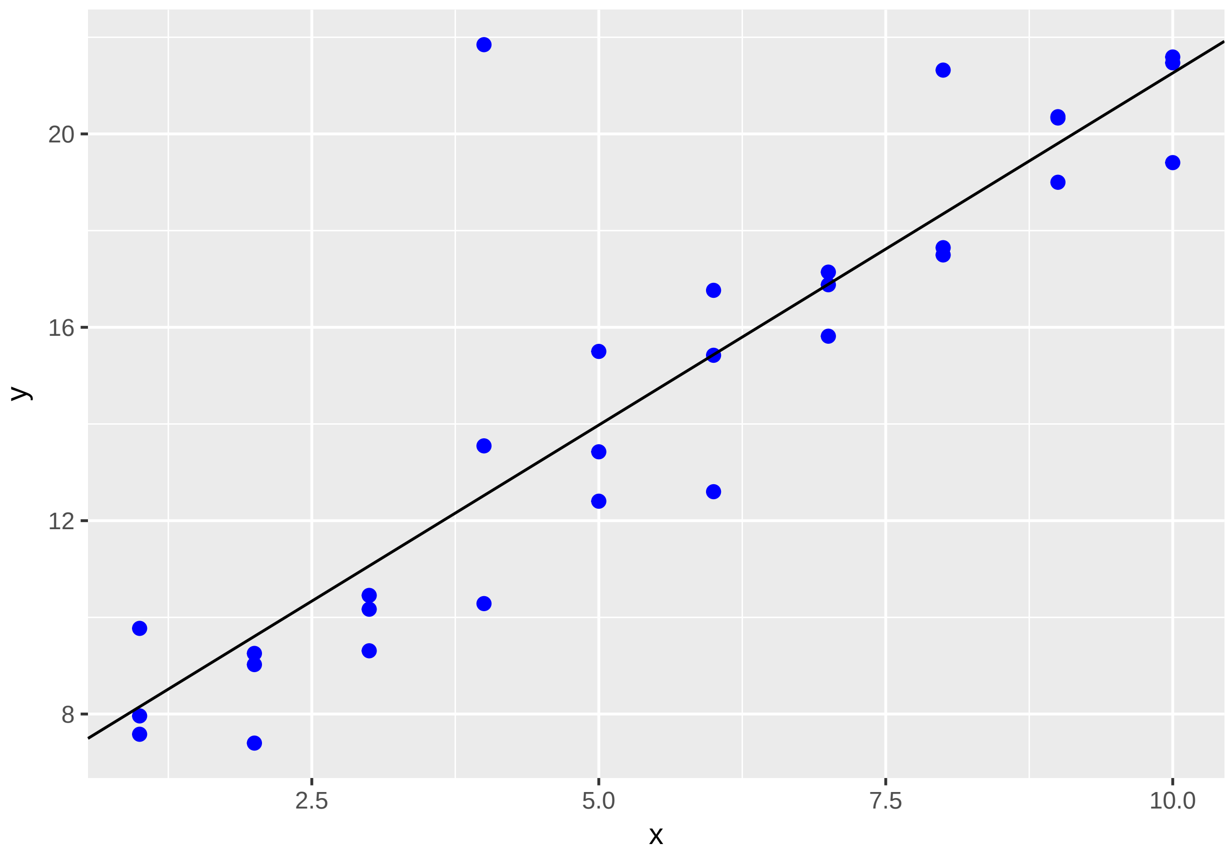
the results. Rerun a few times to generate different simulated datasets. What do you notice about the model?

```
sim1a <- tibble(
x = rep(1:10, each = 3),
y = x * 1.5 + 6 + rt(length(x), df = 2)
)
```

Answer

Lets create a model and run on the simulated data and visualize it.

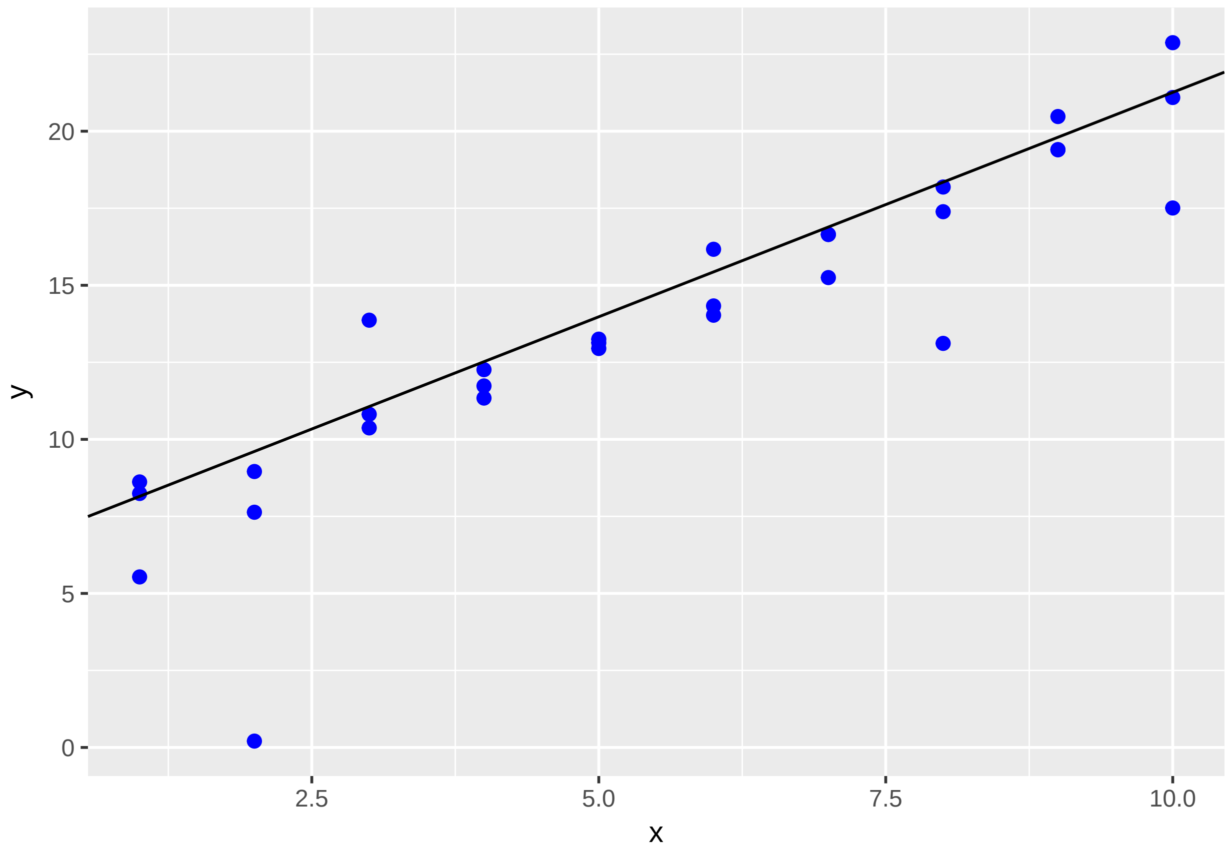
```
sim1a <- tibble(
x = rep(1:10, each = 3),
y = x * 1.5 + 6 + rt(length(x), df = 2)
)

mod_5 <- lm(y~x, data = sim1a)
ggplot(sim1a,aes(x,y))+
  geom_point(size = 2, color = "blue")+
  geom_abline(intercept = mod_5$coefficients[1],slope = mod_5$coefficients[2])
```



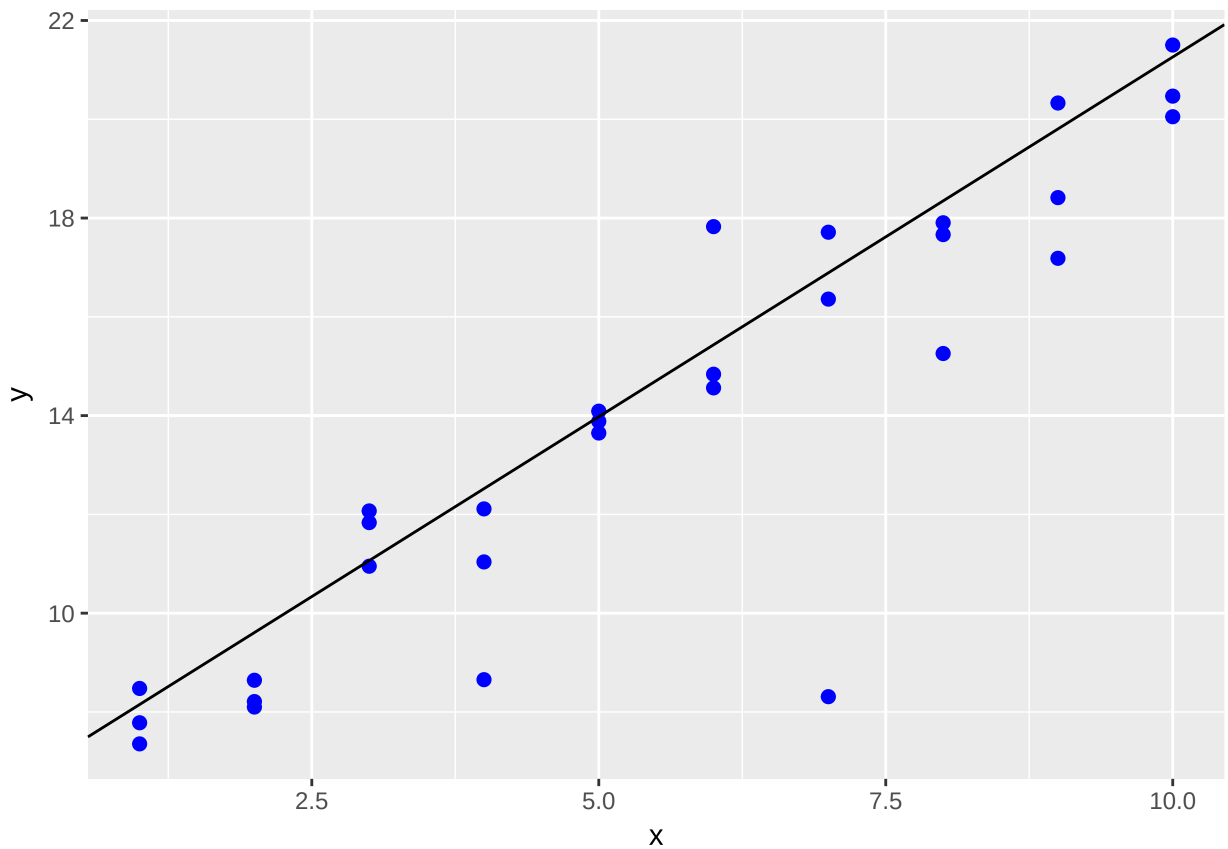
Now, let's run for few times.

```
sim2a <- tibble(
x = rep(1:10, each = 3),
y = x * 1.5 + 6 + rt(length(x), df = 2)
)
```

```
mod_5 <- lm(y~x, data = sim1a)
ggplot(sim2a,aes(x,y))+
  geom_point(size = 2, color = "blue")+
  geom_abline(intercept = mod_5$coefficients[1],slope = mod_5$coefficients[2])
```




```
sim3a <- tibble(
x = rep(1:10, each = 3),
y = x * 1.5 + 6 + rt(length(x), df = 2)
)

mod_5 <- lm(y~x, data = sim1a)
ggplot(sim3a,aes(x,y))+
  geom_point(size = 2, color = "blue")+
  geom_abline(intercept = mod_5$coefficients[1],slope = mod_5$coefficients[2])
```



Conclusion: Sometimes, one single abnormal value forces the fitted line deviate from the “intutively” best lines.

End of Homework 7