Linear Regression Basics

your name

2024-10-24

This will download the dataset needed for this worksheet. If it does not work, notify the instructor.

Math 2265 Chapter 8. Linear Regression

- Work as a group!
- You will need to replace `"ans"` or `your_answer` in the source code
- Update your name in L3
- Add your group members' name below; students may lose one point if Question 0 is unanswered
- Make sure you save and `knit` your work (to html or pdf) before submitting it to Canvas

---

Goal

- Review the basic concepts in linear regression
- Meaning of line fitting
  - Residuals
- The R-value
- how to use the `lm` function in R to find the least squares line

---

Question 0. Who are your group members? (List their first names)

Answer:

1. `<name_1>`
2. `<name_2>`

If you need more time to get used to `Markdown`, use the `Visual` mode.

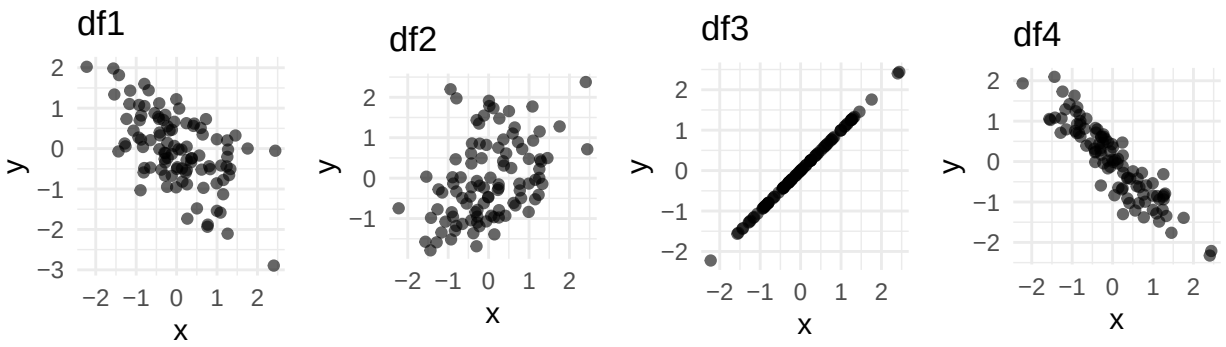
The icon is located in the upper-left corner next to `source`.

Correlation Coefficient R

The following snippet loads four data sets into data frames `df1, df2, df3, df4` (we do not need to understand the code as it is a bit involved). Each data frame consists of two variables, x and y, where the x variable is the same across all data frames.

You may need to install the package `gridExtra` to display the plots side-by-side. Save the file and check the pop-up notification at the top of this edit box.

```r
set.seed(0)
library(gridExtra)

# Read data from the CSV file
df <- read.csv("correlated_datasets.csv")

# We randomly choose 4 sets
target_correlations <- sample(c(2:9),4)

# Loop over the selected column positions
counter <- 1
for (i in target_correlations) {
  var_name <- paste0("df", counter)
  assign(var_name, data.frame(x = df$x, y = df[, i]))
  counter <- counter + 1
}


plots <- list()
# Loop through the four data frames (df1, df2, df3, and df4)
for (i in 1:4) {
  # Dynamically get each data frame by name
  df_name <- paste0("df", i)
  data <- get(df_name)

  # Create the scatter plot
  p <- ggplot(data, aes(x = x, y = y)) +
    geom_point(alpha = 0.6) +
    labs(title = df_name) +
    coord_fixed() +  # Maintain a 1:1 aspect ratio
    theme_minimal()

  plots[[i]] <- p
}

# Arrange all four plots in a single row
grid.arrange(grobs = plots, ncol = 4)
```



Question

Without computing the correlation coefficients (R-values), list the data frames in increasing order.

```r
# change the order of 1,2,3,4 accordingly
my_sorted_cor_coef <- c(1,2,3,4)
my_sorted_cor_coef
```

```
## [1] 1 2 3 4
```

Computing Correlation Coefficients with `cor`

Here is an example of computing the correlation coefficient of `df1`. Recall `$` is used to grab a variable (column) by name.

```r
cor_df1 <- cor(x = df1$x, y = df1$y)
cor_df1
```

```
## [1] -0.5808149
```

Task 1

Compute the other correlation coefficients (R-values) to confirm your answer.

```r
cor_df2 <- cor(x = df2$x, y = df2$y)
cor_df3 <- cor(x = df3$x, y = df3$y)
cor_df4 <- cor(x = df4$x, y = df4$y)
# display all correlation coefficients
c(cor_df1, cor_df2, cor_df3, cor_df4)
```

```
## [1] -0.5808149  0.4104509  1.0000000 -0.9146934
```

Check your answer to the question above.

Linear Regression (best fit line)

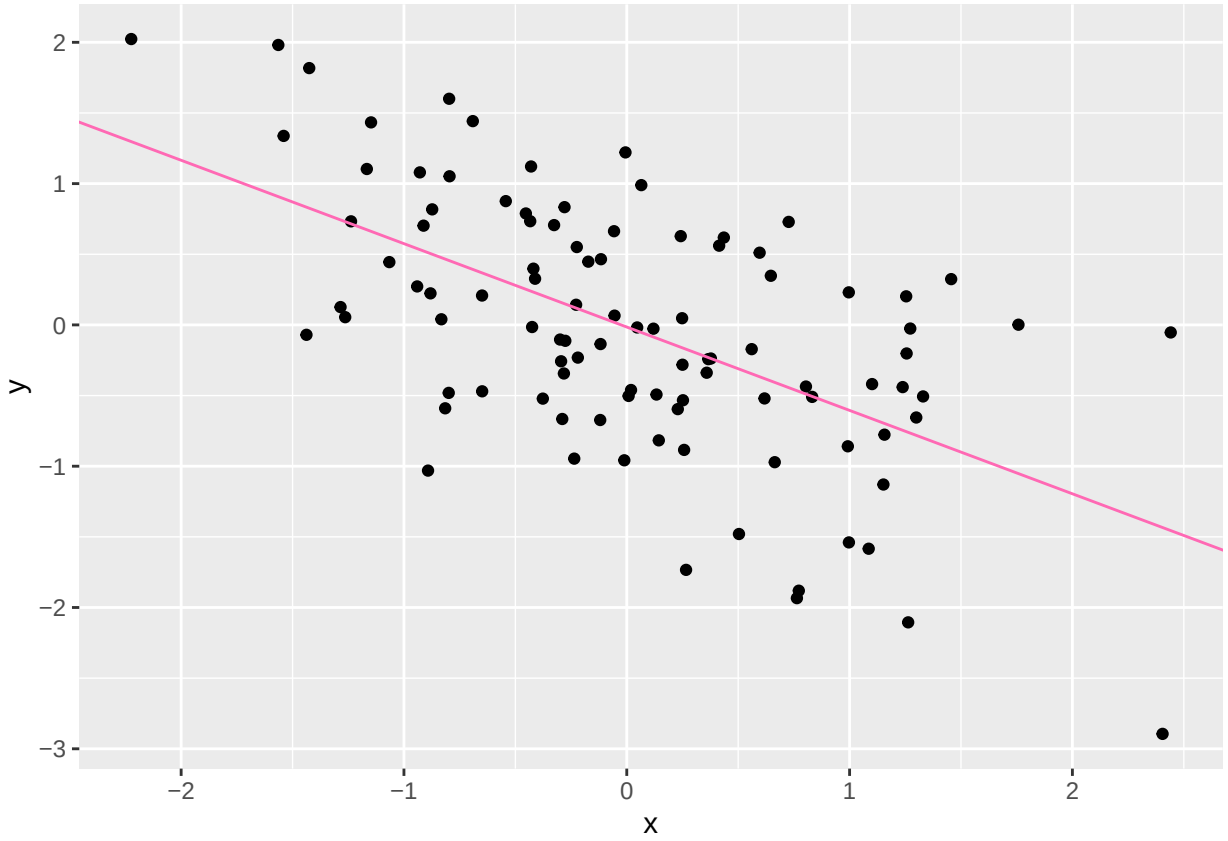
We will use the function `lm` to find the least squares line (the best fitting line): Use the first column for the intercept and slope and sketch its graph. Here is an example for `df1`.

```r
model <- lm(y~x, data=df1)
summary(model); print(paste("R-squared from cor:", cor_df1^2))
```

```
##
## Call:
## lm(formula = y ~ x, data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56154 -0.45499  0.02069  0.55828  1.40224
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01505    0.07340  -0.205    0.838
## x           -0.59011    0.08355  -7.063 2.38e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7337 on 98 degrees of freedom
## Multiple R-squared:  0.3373, Adjusted R-squared:  0.3306
## F-statistic: 49.89 on 1 and 98 DF,  p-value: 2.377e-10
```

```
## [1] "R-squared from cor: 0.337345943968659"
```

```
ggplot(data=df1, aes(x=x,y=y)) +
  geom_point() +
  # you may also copy the numbers in the first column instead of using coef(model)[n]
  geom_abline(intercept = coef(model)[1], slope = coef(model)[2], color='hotpink')
```



The sum of squared residuals We will compute the sum of squared residuals and save it to the variable `my_ssr`. We learned a way in the last worksheet. The `lm` function provides a convenient way.

```
# Computes the residuals of the least squares line
residuals(model)
```

```
##             1             2             3             4             5
## -1.344819e+00  5.289565e-01  2.935526e-01  7.401571e-01  8.203493e-01
##             6             7             8             9            10
##  4.440215e-01  5.466644e-01 -4.162663e-01  1.232589e+00 -1.460317e+00
##            11            12            13            14            15
## -1.468067e+00 -9.376185e-01  7.712663e-01 -8.216845e-01 -2.651774e-01
##            16            17            18            19            20
##  9.914702e-02 -3.693379e-01 -1.542322e+00  8.904579e-01  1.775106e-02
##            21            22            23            24            25
##  4.337975e-01 -7.175557e-06 -3.990206e-01  5.346017e-02  6.444749e-01
##            26            27            28            29            30
## -1.167649e+00 -9.282644e-01  1.049598e+00 -6.167420e-01  2.467507e-02
##            31            32            33            34            35
## -1.070438e+00  5.707558e-01  4.932288e-01 -1.602571e-01  1.172907e+00
##            36            37            38            39            40
```

```
## -4.348574e-01 -2.586912e-01  8.827911e-01  3.057014e-01  6.835674e-01
##            41            42            43            44            45
##  1.053946e+00  1.744530e-01  5.366341e-01 -4.360652e-01  4.301211e-01
##            46            47            48            49            50
## -1.693774e-01  1.073519e+00 -7.922972e-02 -3.330291e-03  2.363673e-02
##            51            52            53            54            55
## -1.561543e+00 -7.290187e-01  1.402239e+00  5.976019e-01  4.827024e-02
##            56            57            58            59            60
## -1.192326e-01 -1.404683e-01  3.608385e-01  7.258565e-01 -6.758509e-01
##            61            62            63            64            65
## -1.116387e-01 -9.496717e-01 -2.676870e-01  4.117563e-01 -1.056052e+00
##            66            67            68            69            70
##  7.863302e-01  9.917044e-01 -1.109014e-02  2.090600e-01  1.042972e+00
##            71            72            73            74            75
## -4.343367e-01 -7.177579e-01 -8.378653e-01 -7.283143e-01 -5.647154e-01
##            76            77            78            79            80
##  2.458938e-01 -7.170974e-01 -1.898954e-01  1.792357e-01 -9.034073e-01
##            81            82            83            84            85
##  1.145213e+00  9.573895e-01 -1.410369e+00 -3.459489e-01 -2.505275e-01
##            86            87            88            89            90
##  1.658696e-01 -9.359785e-01 -2.602555e-01  5.541265e-01  7.443792e-01
##            91            92            93            94            95
##  1.264076e-01  3.174379e-01 -4.826870e-01 -2.815592e-01  8.779639e-01
##            96            97            98            99           100
##  5.905887e-02 -4.952673e-01  1.198203e+00 -4.457607e-01  8.335493e-01
```

It is a vector, so we need to take the sum of their squares with a couple of commands

```
ssr <- sum( (residuals(model))^2 )
ssr
```

```
## [1] 52.75906
```

Task 2

Find the least squares line for `df2`, sketch the scatter plot and the line, and compute the sum of the squared residuals.

Least squares line

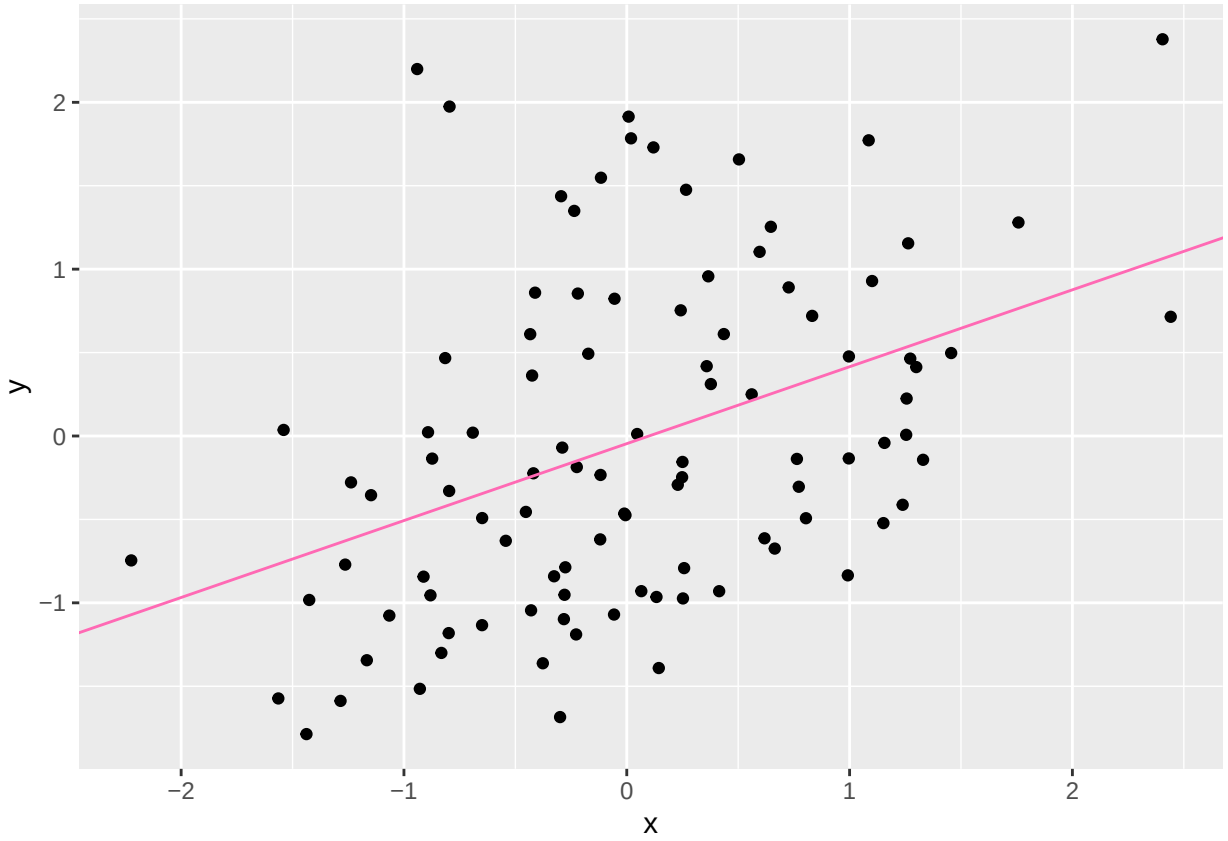
```
# linear model
model2 <- lm(y~x, data = df2)
summary(model2)
```

```
##
## Call:
## lm(formula = y ~ x, data = df2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5009 -0.7696 -0.1445  0.6022  2.6789
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.04597    0.09089  -0.506    0.614
```

```
## x            0.46102    0.10346   4.456 2.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9086 on 98 degrees of freedom
## Multiple R-squared:  0.1685, Adjusted R-squared:   0.16
## F-statistic: 19.86 on 1 and 98 DF,  p-value: 2.222e-05
```

Scatter plot

```
ggplot(data=df2, aes(x=x,y=y)) +
  geom_point() +
  geom_abline(intercept = coef(model2)[1], slope = coef(model2)[2], color='hotpink')
```



Sum of the squared residuals

```
# write your code below
my_ssr2 <- sum( residuals(model2)^2 )
my_ssr2
```

```
## [1] 80.91086
```

Plot residuals

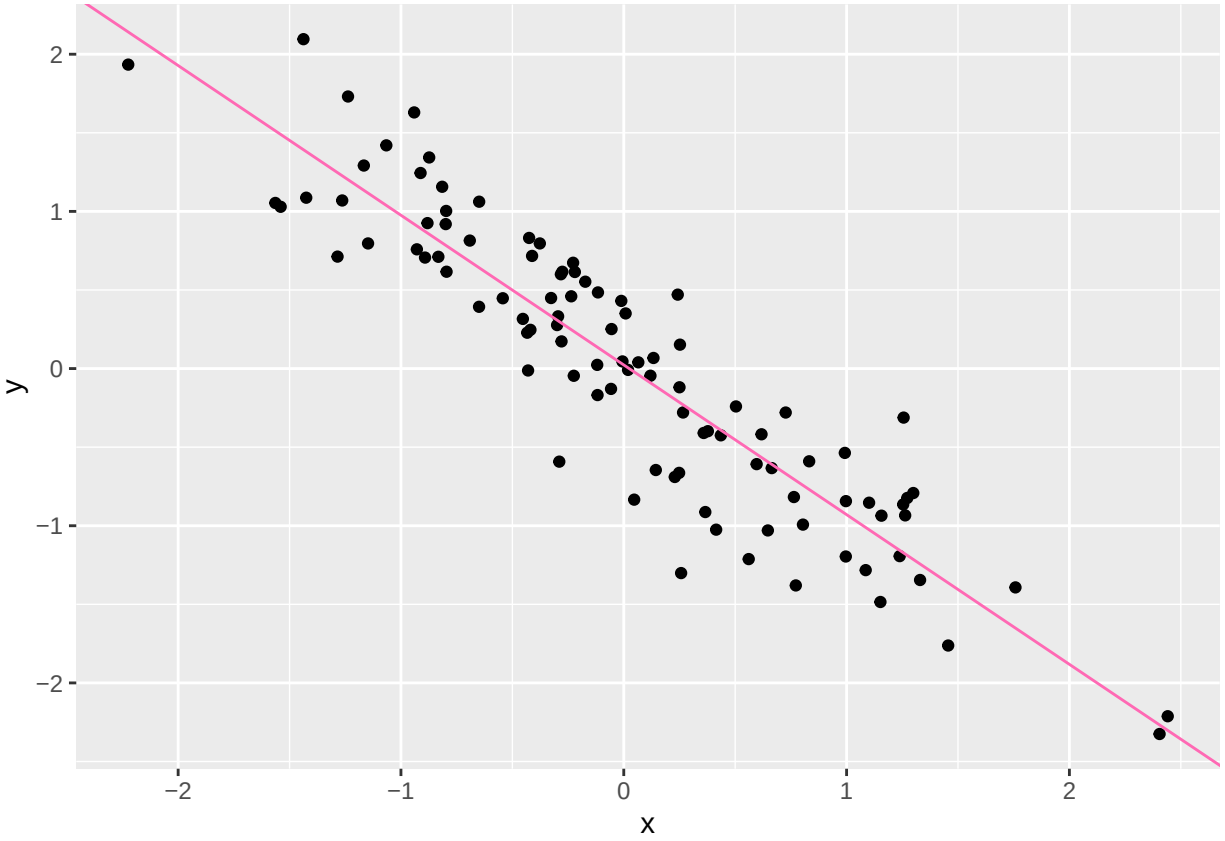
Task 3

First find the least squares line and save the residuals of `df4` as a variable

```
model4 <- lm(y~x, data=df4)
df4$res <- residuals(model4)
```

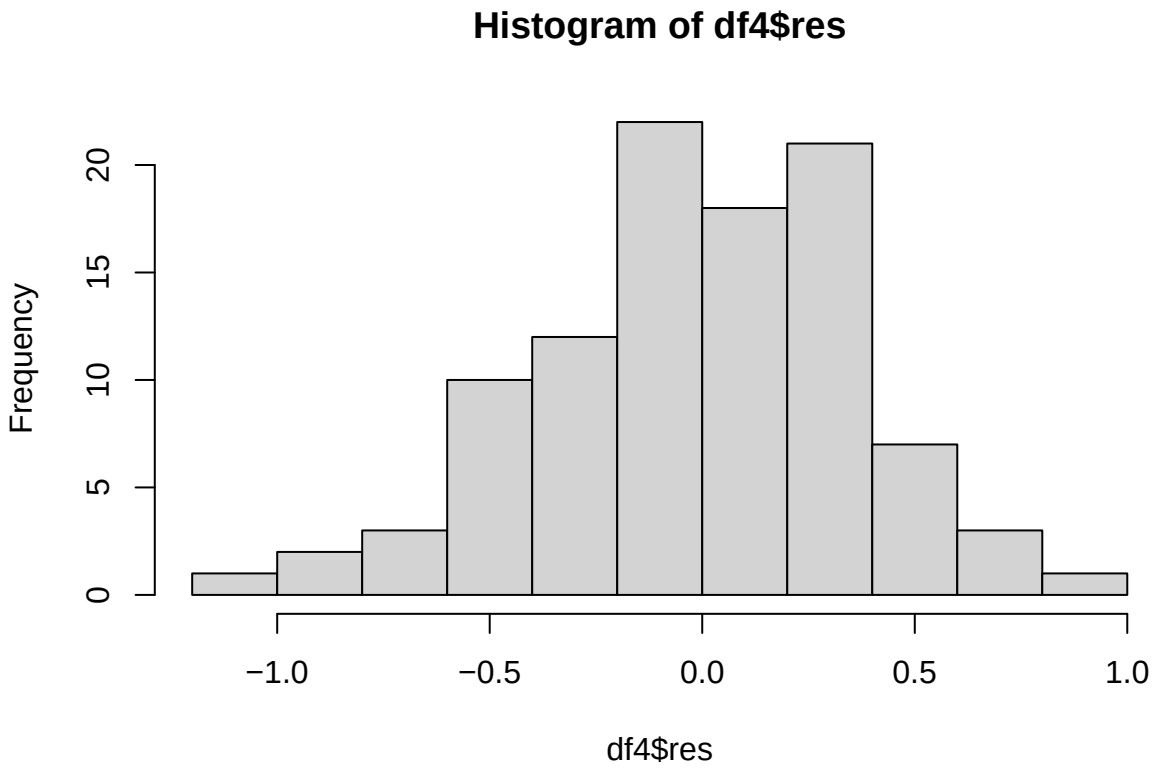
Add the intercept and slope of `model4`.

```
ggplot(data=df4, aes(x=x,y=y)) +
  geom_point() +
  geom_abline(intercept = coef(model4)[1], slope = coef(model4)[2], color='hotpink')
```



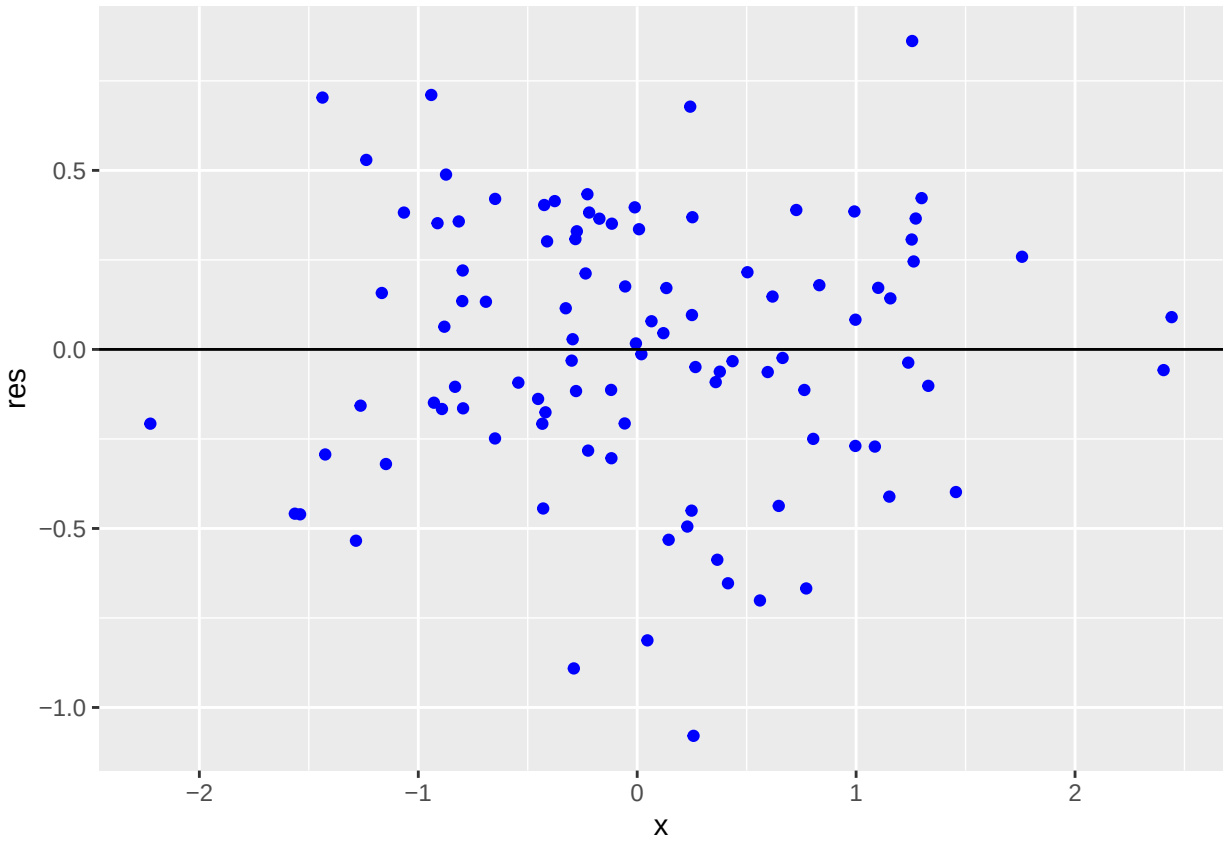
This histogram explains the term `nearly normal residuals`. That is the resudials follow a normal distribution.

```
hist(df4$res)
```



This is the scatter plot of the residuals.

```
ggplot(data=df4, mapping = aes(x = x, y = res)) +
  geom_point(color='blue') +
  geom_abline(intercept = 0, slope = 0)
```



Check your answers

TRUE: correct FALSE: incorrect

```
my_sorted_cor_coef == c(3,2,1,4)
```

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

```
my_ssr2 == sum( residuals(model2)^2 )  #
```

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

Share your work and help your group members before uploading your work to Canvas