lab4.R

Feipeng Huang

2022-10-05

```
#Q1
r_mean = 10.4
r_sd = 2.4

norm_17 = rnorm(17, mean = r_mean, sd = r_sd)
norm_30 = rnorm(30, mean = r_mean, sd = r_sd)
norm_300 = rnorm(300, mean = r_mean, sd = r_sd)
norm_3000 = rnorm(3000, mean = r_mean, sd = r_sd)
#Q2
require(here)
```

```
## Loading required package: here
```

```
## here() starts at /Users/stonehuang/Documents/environmental_data
```

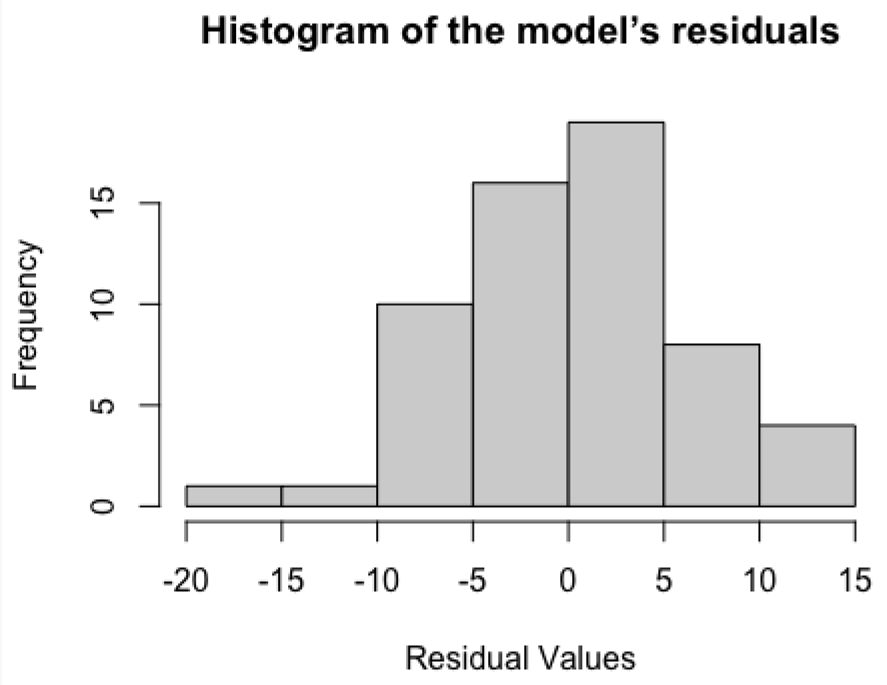
```
png(
  filename = here("lab_04_hist_01.png"),
  width = 1500, height = 1600,
  res = 180, units = "px")

par(mfrow = c(2, 2))

hist(norm_17, main = "Histogram 17 nd random numbers")
hist(norm_30, main = "Histogram 30 nd random numbers")
hist(norm_300, main = "Histogram 300 nd random numbers")
hist(norm_3000, main = "Histogram 3000 nd random numbers")

dev.off()
```

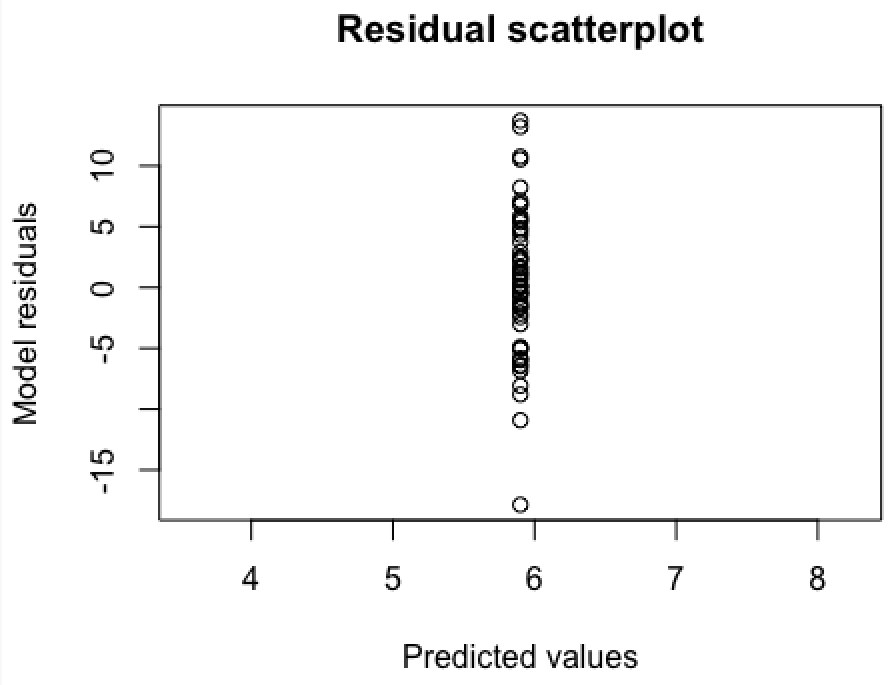
```
## quartz_off_screen
##                 2
```

```
#Q4
#Histograms of 17 and 30 normally-distributed random numbers are more skewed.
Histograms of 300 and 3000 normally-distributed random numbers are less
skewed and better resemble the shape of normal distribution.
#Q5
#The shapes of the histograms are different due to difference in sample size.
A larger sample size can better represent the normal distribution.
#Q6
#mean = 0
#standard deviation = 1
```

```r
#Q7
x = seq(0, 20, length.out = 100)
y = dnorm(x, mean = 10.4, sd = 2.4, log = FALSE)
pdf(
  file = here("norm_1.pdf"),
  width = 7, height = 7
  )
plot(x, y, xlim = c(0, 20), main = "Normal PDF: mean = 10.4, sd = 2.4", type
= "l")
abline(h = 0)
dev.off()

## quartz_off_screen
##                 2

#Q9
norm_1 = rnorm(59, mean = 5.9, sd = 5.9)
norm_2 = rnorm(5959, mean = 5.9, sd = 5.9)
unif_1 = runif(n = 59, min = 5, max = 9)
unif_2 = runif(n = 5959, min = 5, max = 9)
png(
  filename = here("lab_04_Q10.png"),
  width = 1500, height = 1600,
  res = 180, units = "px")
par(mfrow = c(2, 2))
hist(norm_1, main = "59 random normally-distributed numbers", col =
rgb(1,1,0))
hist(norm_2, main = "5959 random normally-distributed numbers", col =
rgb(1,0.5,0))
hist(unif_1, main = "59 random uniform numbers", col = rgb(1,1,0))
hist(unif_2, main = "5959 random uniform numbers", col = rgb(1,0.5,0))
dev.off()

## quartz_off_screen
##                 2

#Q11
# Calculates the value of y for a linear function, given the coordinates
# of a known point (x1, y1) and the slope of the line.
line_point_slope = function(x, x1, y1, slope)
{
  get_y_intercept =
    function(x1, y1, slope)
      return(-(x1 * slope) + y1)

  linear =
    function(x, yint, slope)
      return(yint + x * slope)

  return(linear(x, get_y_intercept(x1, y1, slope), slope))
}
```

```
library(here)
n_pts = 59
x_min = 5
x_max = 9
x_random = runif(n = n_pts, min = x_min, max = x_max)
y_random = rnorm(n = n_pts, mean = 5.9, sd = 5.9)
dat_random = data.frame(x = x_random, y = y_random)
png(
  filename = here("lab_04_Q12.png"),
  width = 1500, height = 1600,
  res = 180, units = "px")
plot(y ~ x, data = dat_random, pch = 8)
guess_x = 7
guess_y = 5.9
guess_slope = 0
curve(line_point_slope(x, guess_x, guess_y, guess_slope), add = T)
dev.off()

## quartz_off_screen
##                  2

line_point_slope(dat_random$x, guess_x, guess_y, guess_slope)

##  [1] 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9
5.9 5.9
## [20] 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9
5.9 5.9
## [39] 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9
5.9 5.9
## [58] 5.9 5.9

#Q13

dat_random$y_predicted = line_point_slope(dat_random$x, guess_x, guess_y,
guess_slope)
dat_random$resids = dat_random$y - dat_random$y_predicted

#Q14
```


Histogram of the model's residuals
Frequency
0
5
10
15
-20
-15
-10
-5
0
5
10
15
Residual Values


Residual scatterplot
Model residuals
-15
-5
0
5
10
4
5
6
7
8
Predicted values