STT 461 HW 6

Derien Weatherspoon

2023-04-05

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)
library(stats)
library(ggplot2)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

Question 1

Take the function(x, y, z) = x^2 - 3x + y^4 - 3y + z^2 + 10z + cos(cyz). Find the location of the minimum value. Use c(0,0,0) as the initialization location. Try different initialization locations. Can you find the different locations for minima? If you see multiple ones, which one seems to be a global minimum?

```
q1func <- function(xyz) {
  x <- xyz[1]
  y <- xyz[2]
  z <- xyz[3]
  return(x^2 - 3*x + y^4 - 3*y + z^2 + 10*z + cos(x*y*z))
}
x0 <- c(0, 0, 0)

result <- optim(x0, q1func)
min <- result$par
min
```

```
## [1]  0.9586131  0.7348568 -4.8938219
```

```
# Try different initialization locations
x1 <- c(1, 1, 1)
result1 <- optim(x1, q1func)
print(result1$par)
```

```
## [1]  2.406042  1.186217 -5.403300
```

```
x2 <- c(-1, -1, -1)
result2 <- optim(x2, q1func)
print(result2$minimum)
```

```
## NULL
```

Global minimum at the coordinates given from the code.

Question 2

The beta distribution on the interval $0 < x < 1 \ldots$ probability density: p(x) = *dbeta(x, alpha, beta)*

a) Take the sample data in d2l called beta_samp.csv. Construct the log-likelihood function using dbeta.

```
beta <- read.csv("beta_samp.csv")
log_likelihood <- function(params) {
  a <- params[1]
  b <- params[2]
  log_likelihoods <- dbeta(beta$sample, a, b, log = TRUE)
  all_log_likelihood <- sum(log_likelihoods)
  return(-all_log_likelihood)
}
```

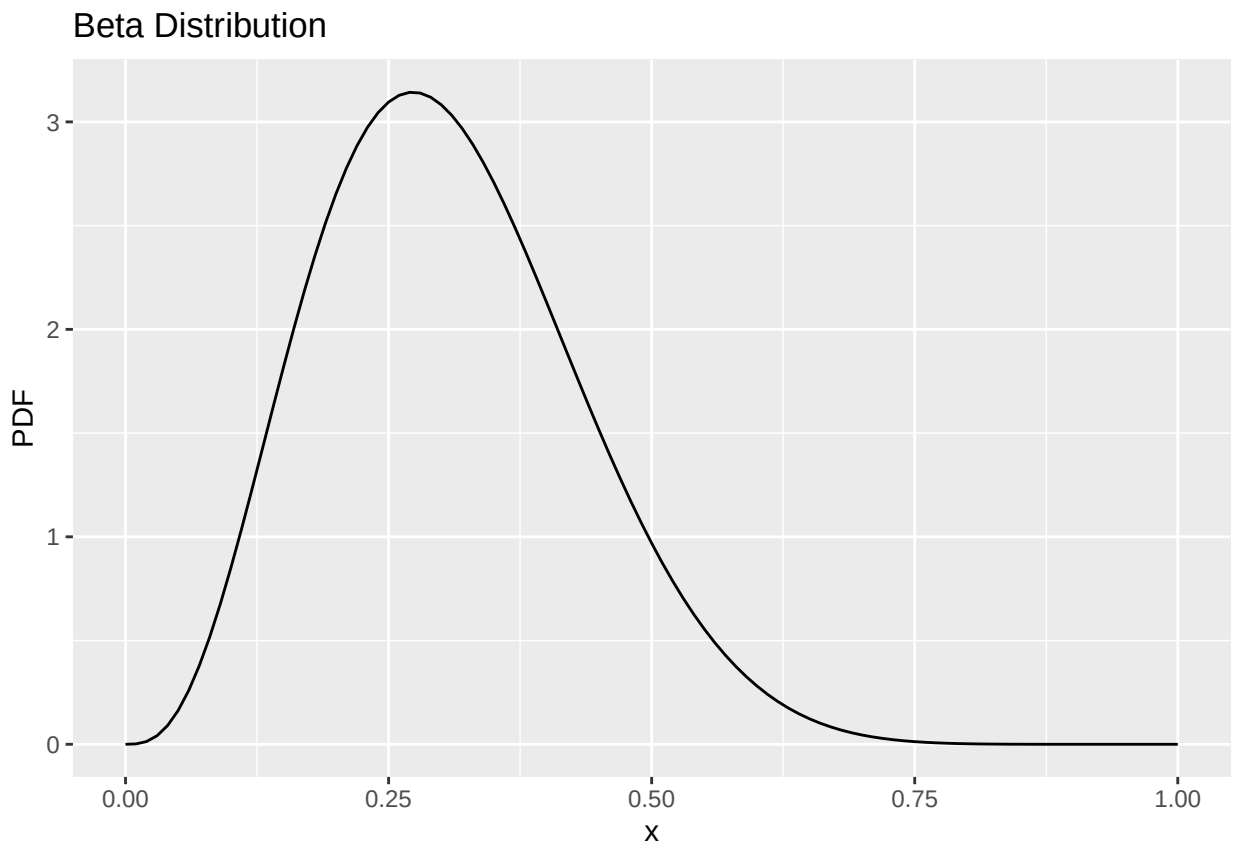
b) Use the optim function to find the MLE for the parameters.

```
params <- c(100, 90)
result <- optim(params, log_likelihood)
estimated_params <- result$par
estimated_params
```

```
## [1] 4.004677 9.000550
```

c) Graph the beta distribution for the given parameters between $0 < x < 1$.

```
alpha <- estimated_params[1]
beta <- estimated_params[2]
x <- seq(0, 1, by = 0.01)
pdf <- dbeta(x, alpha, beta)
dat <- data.frame(x = x, pdf = pdf)
ggplot(dat, aes(x = x, y = pdf)) +
  geom_line() +
  labs(title = "Beta Distribution",
       x = "x",
       y = "PDF")
```



Question 3

Referring back to problem 2 on homework 5, with the Boston dataset, redo the linear regressions in the following ways:

a) Construct a linear model with the variable with the strongest correlation, like previously, but this time do the minimization of RMSE directly, with the optim function. Check to make sure the result matches what you got in the previous homework.

```
df <- read.csv("boston.csv")

# Create a linear model with the variable with the strongest correlation
model <- lm(medv ~ lstat, data = df)

# Use the optim function to minimize RMSE
optim_model <- optim(par = coef(model), fn = function(x) {
  pred <- predict(model, newdata = df)
  rmse <- sqrt(mean((df$medv - pred)^2))
  return(rmse)
})
optim_model
```

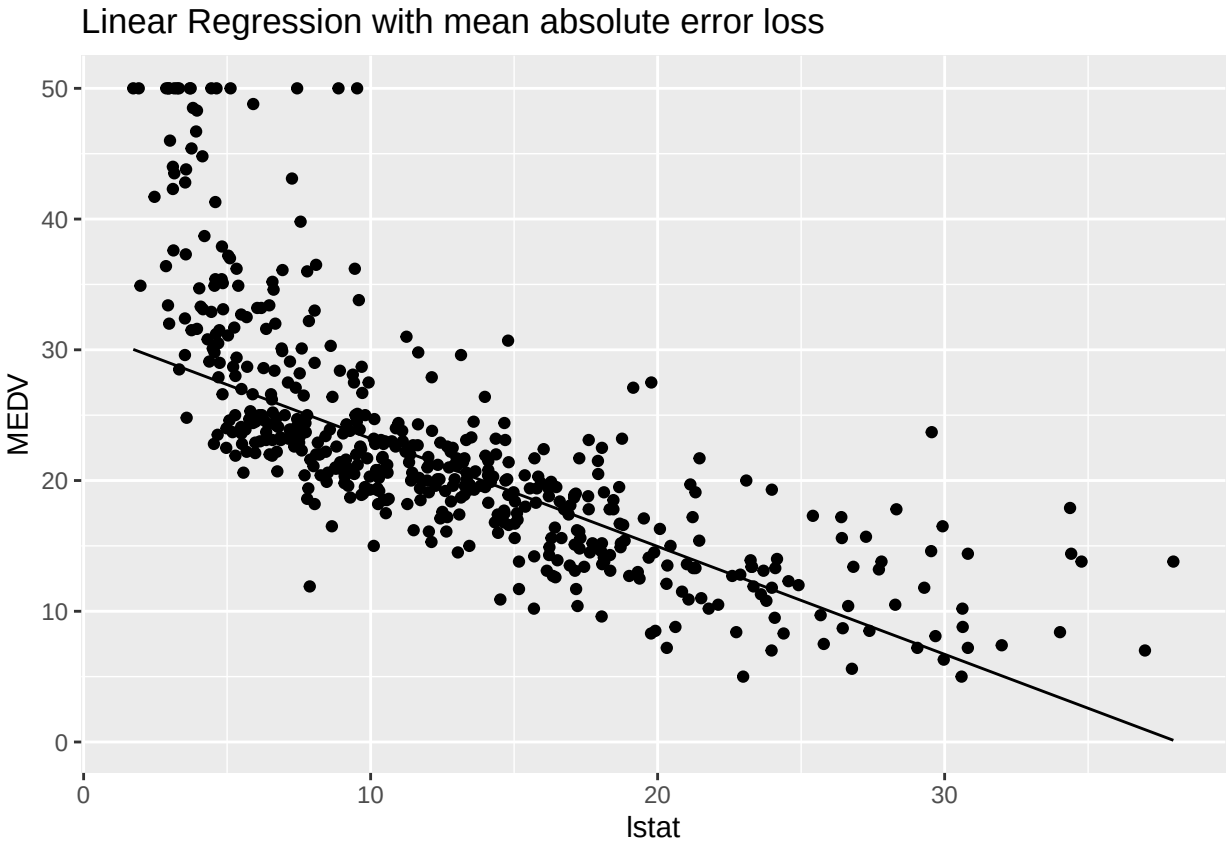
```
## $par
## (Intercept)       lstat
```

```
##  34.5538409  -0.9500494
##
## $value
## [1] 6.203464
##
## $counts
## function gradient
##        3       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

RMSE of 6.20, I believe this is the same as before.

b) Next construct a linear model of the same form, with mean absolute error as the loss function. Plot the regression lines and compare the model with what you did in the previous part.

```
mean_absolute_error <- function(beta) {
  predicted <- beta[1] + beta[2] * df$lstat
  residuals <- predicted - df$medv
  mean_absolute_error <- mean(abs(residuals))
  return(mean_absolute_error)
}
init_beta <- c(0, 0)
result <- optim(init_beta, mean_absolute_error)
intercept <- result$par[1]
slope <- result$par[2]
predicted <- intercept + slope * df$lstat
data <- data.frame(x = df$lstat, y = df$medv, predicted = predicted)
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  geom_line(aes(y = predicted)) +
  labs(title = "Linear Regression with mean absolute error loss",
       x = "lstat",
       y = "MEDV")
```


Linear Regression with mean absolute error loss
MEDV
lstat
50
40
30
20
10
0
0
10
20
30