PS01-HT

Kathryn Glen - 17331061

2/14/2022

Code from template given to begin with:

```
# remove objects
rm(list=ls())
# detach all libraries
detachAllPackages <- function() {
  basic.packages <- c("package:stats", "package:graphics", "package:grDevices
", "package:utils", "package:datasets", "package:methods", "package:base")
  package.list <- search()[ifelse(unlist(gregexpr("package:", search()))==1,
TRUE, FALSE)]
  package.list <- setdiff(package.list, basic.packages)
  if (length(package.list)>0)  for (package in package.list) detach(package,
character.only=TRUE)
}
detachAllPackages()

# load libraries
pkgTest <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[,  "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg,  dependencies = TRUE)
  sapply(pkg,  require,  character.only = TRUE)
}

# here is where you load any necessary packages
# ex: stringr
# lapply(c("stringr"),  pkgTest)

lapply(c(),  pkgTest)

## list()

library("rstudioapi")


# set wd for current folder
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
getwd()

## [1] "/Users/Kate/Desktop"
```

Question 1

```
set.seed(123)
data_vec <- (rcauchy(1000, location = 0, scale = 1))
```

Create a function using code given in the question. data_vec = g in function

```
my_KS <- function(g){
  ECDF <- ecdf(g)
  empiricalCDF <- ECDF(g)
  # generate test statistic
  D <- max(abs(empiricalCDF - pnorm(g)))
  return(D)
}
```

I next use the inbuilt R function for ks.test to check my work and compare.

```
ks.test(data_vec, "pnorm")

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  data_vec
## D = 0.13573, p-value = 2.22e-16
## alternative hypothesis: two-sided

my_KS(data_vec)

## [1] 0.1347281
```

The distance values between my_KS test and the inbuilt ks.test are essentially the same. I think the minute difference is due to calculation rounding. my_KS = 0.1347281. ks.test = 0.13573.

The p-value for the inbuilt ks.test is less than 0.05 so we reject the null hypothesis and the two distributions are not the same. This makes sense as one distribution is cauchy the other is pnorm.

I wasn't sure whether or not I needed to find the p-value myself.I attempted to find the p-value myself using the below code. However, the package used was built for an older version of R. When I tried to run it below it broke my computer and I needed to reboot.

I included it for notes to show my work but would not advise running it.

Unused code: "#install.packages("KSgeneral") library("KSgeneral") library("Rcpp") cont_ks_cdf(0.05, 1000)"

Based on research I think another option to find the p-value would be the use of approximation by comparing using Table 23 in Cambridge Statistical Tables but that is not R code.

Question 2

```
set.seed(123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)

y <- as.matrix(data[, "y"])
x <- as.matrix(data[, "x"])

intercept <- rep(1, nrow(x)) # column of ones for intercept
```

Coerce into matrices.

```
Y <- as.matrix(y)
X <- as.matrix(cbind(intercept, x))
```

Store the first part as a matrix called A. This isn't strictly necessary but it's convenient.

```
A <- solve(t(X) %*% X)
```

Find our OLS beta coefficient estimates.

```
betas <- A %*% t(X) %*% Y
betas

##                [,1]
## intercept 0.1391874
##           2.7266985

# As defined in example from lecture notes.
sigma <- sqrt(betas[1])

beta <- betas[2]

linear <- lm(y~x, data=data)
summary(linear)

##
## Call:
## lm(formula = y ~ x, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1906 -0.9374 -0.1665  0.8931  4.8032
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.13919    0.25276   0.551    0.582
## x            2.72670    0.04159  65.564   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.447 on 198 degrees of freedom
## Multiple R-squared:  0.956,  Adjusted R-squared:  0.9557
## F-statistic:  4299 on 1 and 198 DF,  p-value: < 2.2e-16

linear$coefficients

## (Intercept)           x
##   0.1391874   2.7266985

betas

##                [,1]
## intercept 0.1391874
##           2.7266985
```

As we can see here the linear coefficients and the beta coefficients are identical. However I need to use the BFGS method.

Code below as found in lecture notes.

```
theta <- c(beta, sigma)
Y <- as.matrix(y)
X <- as.matrix(x)

linear.lik <- function(theta, y, X){
  n <- nrow(X)
  k <- ncol(X)
  beta <- theta[1:k]
  sigma2 <- theta[k+1]^2
  e <- y - X%*%beta
  logl <- -.5*n*log(2*pi) - .5*n*log(sigma2) - ((t(e) %*% e)/(2*sigma2))
  return (-logl)
}
```

Using optim() function to enact BFGS method as asked in question.

```
Opts <-  optim(fn=linear.lik, par=c(1,1,1), hessian = TRUE,
               y=y, X =cbind(1, x), method = "BFGS")
Opts$par

## [1]  0.1398324  2.7265559 -1.4390716
```

Par from Opts (optim function) intercept = 0.1398324 This is slightly different to the intercept of OLS and lm because I used the Newton-Raphson method (BFGS specifically).

Intercept from lm = 0.1391874 Intercept from OLS = 0.1391874