Problem Set 1: Applied Stats II

Caitlín Cooney

Due: February 19, 2023

Question 1

Using `R` generate 1,000 Cauchy random variables (`rcauchy(1000, location = 0, scale = 1)`) and perform the test (remember, use the same seed, something like `set.seed(123)`, whenever you're generating your own data).

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

Write an `R` function that implements the Kolmogorov-Smirnov test where the reference distribution is normal.

```
kolsmir.test <- function(data) {

  # Create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

  # Generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))

  # Calculate p-value
  p.value <- 1 - pnorm(sqrt(length(data)) * D)

  return(list(D = D, p.value = p.value))
}
```

State your null and alternate hypotheses:

H0: The data comes from the specified distribution.

H1: At least one value does not match the specified distribution.

```
# Run the function
kolsmir.test(data)
```

The results are:

```
$D
[1] 0.1347281
$p.value
[1] 1.019963e-0}
```

As D is not greater than the critical value, we fail to reject the null hypothesis that the data comes from the specified distribution (or in other words, that P = P0).

Question 2

Estimate an OLS regression in `R` that uses the Newton-Raphson algorithm (specifically `BFGS`, which is a quasi-Newton method).

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

```r
  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)
  }

```

Find parameters that specify the point.

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

  linear.MLE$par
```

The results show:

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

Show that you get the equivalent results to using `lm`.

```r
  summary(lm(y~x, data))
```

The results show:

```
  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 ***
```

As expected, the estimates for `lm` are the same as the parameters found using the Newton-Raphson algorithm, because ordinary least squares is equivalent to maximum likelihood for a linear model, so it makes sense that `lm` would give us the same answers.