Answer Key: Problem Set 1

Applied Stats II

Jeffrey Ziegler

Instructions

- *Please show your work! You may lose points by simply writing in the answer. If the problem requires you to execute commands in* `R`*, please include the code you used to get your answers. Please also include the* `.R` *file that contains your code. If you are not sure if work needs to be shown for a particular problem, please ask.*
- *Your homework should be submitted electronically on GitHub in* `.pdf` *form.*
- *This problem set is due before class on Monday February 14, 2022. No late assignments will be accepted.*
- *Total available points for this homework is 80.*

Question 1

The Kolmogorov-Smirnov test uses cumulative distribution statistics test the similarity of the empirical distribution of some observed data and a specified PDF, and serves as a goodness of fit test. The test statistic is created by:

$$D = \max_{i=1:n} \left\{ \frac{i}{n} - F_{(i)}, F_{(i)} - \frac{i-1}{n} \right\}$$

where F is the theoretical cumulative distribution of the distribution being tested and $F_{(i)}$ is the ith ordered value. Intuitively, the statistic takes the largest absolute difference between the two distribution functions across all x values. Large values indicate dissimilarity and the rejection of the hypothesis that the empirical distribution matches the queried theoretical distribution. The p-value is calculated from the Kolmogorov-Smirnoff CDF:

$$p(D \leq x) \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2 \pi^2 / (8x^2)}$$

which generally requires approximation methods (see Marsaglia, Tsang, and Wang 2003). This so-called non-parametric test (this label comes from the fact that the distribution of the

test statistic does not depend on the distribution of the data being tested) performs poorly in small samples, but works well in a simulation environment. Write an `R` *function that implements this test where the reference distribution is normal. 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).*

As a hint, you can create the empirical distribution and theoretical CDF using this code:

```
    # create empirical distribution of observed data
    ECDF <- ecdf(data)
    empiricalCDF <- ECDF(data)
    # generate test statistic
    D <- max(abs(empiricalCDF - pnorm(data)))
```

First, we'll create our data using `rcauchy(1000, location = 0, scale = 1)`. Then, we can insert the above code into our function, which first calculates our test statistic and then iteratively calculates our p-value.

```
# create data
set.seed(5)
n <- 1000
empirical <- rcauchy(n, location = 0, scale = 1)
# create K-S function to compare data against normal distribution
ksTest <- 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
  # empty vector to be filled
  summed <- NULL
  for(i in 1:n){
    summed <- c(summed, exp((-(2 * i - 1)^2 * pi^2) / ((8 * D)^2)))
  }
  pValue <- sqrt(2*pi)/D * sum(summed)
  cat("D =", D, "\n")
  cat("p-value =", pValue, "\n")
}
# run "by hand" K-S test
ksTest(empirical)
```

```
D = 0.135986
p-value = 0.004404211
```

We can corroborate that the above output from our function operates similarly to the built-in function `ks.test()`, though there are some differences due to approximation and rounding.

```
# double check K-S test with R function
ks.test(empirical, "pnorm")
```

```
One-sample Kolmogorov-Smirnov test

data:  empirical
D = 0.13699, p-value < 2.2e-16
alternative hypothesis: two-sided
```

Question 2

Estimate an OLS regression in `R` *that uses the Newton-Raphson algorithm (specifically* `BFGS`*, which is a quasi-Newton method), and show that you get the equivalent results to using* `lm`*. Use the code below to create your data.*

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

First, we need to create our data, then we create write our log-likelihood function.

```
# create log-likelihood function
# we need to specify what the outcome will be,
# what the input variables are, and the starting values
# of the parameters to be estimated
norm_log_likelihood <- function(outcome, input, parameter) {
  # how many betas are we estimating? (# of columns)
  n <- ncol(input)
  beta <- parameter[1:n]
  # estimate of our variance as well
  sigma <- sqrt(parameter[1+n])
  # put all of those into our log-likelihood (since log=T)
  # remember, we're referencing normal distribution, so
  # that's why you see dnorm()
  -sum(dnorm(outcome, input %*% beta, sigma, log=TRUE))
}
```

Next, we can execute our function and confirm that we get the same answer using `lm()`.

```
# print our estimated coefficients (intercept and beta_1)
results_norm <- optim(fn=norm_log_likelihood, outcome=ex_data$y, input=cbind
    (1, ex_data$x), par=c(1, 1, 1), hessian=T)
# print our estimated coefficients (intercept and beta_1)
# get same results regardless of which log-likelihood function we use
round(results_norm$par, 2)[1:2]
```

```
  [1] 0.14 2.73
```

```
# confirm that we get the same thing with glm()
round(coef(lm(y~x, data=ex_data)), 2)
```

```
  [1] 0.14 2.73
```