Problem Set 1

Applied Stats II

Due: February 11, 2024

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 23:59 on Sunday February 11, 2024. No late assignments will be accepted.

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:

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

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)))
```

Set up null hypothesis(H0) and alternative hypothesis(H1):

Null Hypothesis (H0): The observed data follows a specified theoretical distribution (in this problem, the specified distribution is the standard normal distribution).

Alternative Hypothesis (H1): The observed data does not follow the specified theoretical distribution.

```
# Define function for Kolmogorov-Smirnov test with normal reference
    distribution
kolmogorov_smirnov_test <- function(sample_size, seed = 123) {
  # Set seed for reproducibility
  set.seed(seed)

  # Generate Cauchy random variables, where sample_size is 1000
  data <- rcauchy(sample_size, location = 0, scale = 1)

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

  # Generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))  # Generate test statistic, which
    is the maximum difference between the empirical distribution function of
    the observed data and the theoretical cumulative distribution function of
    the standard normal distribution

  # Calculate p-value using Kolmogorov-Smirnov CDF
  p_value <- sqrt(2 * pi) / D * sum(exp(-((2 * (1:length(data)) - 1)^2 * pi^2)
    / (8 * D^2)))

```

```r
  # Return test statistic and p-value
  return(list(D = D, p_value = p_value))
}

# Example usage
result <- kolmogorov_smirnov_test(1000)
print(result)

# $D
# [1] 0.1347281

# $p_value
# [1] 5.652523e-29


# Compare with ks.test() function in R to validate custom implementation
ks_result <- ks.test(data, "pnorm")
print(ks_result)

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

Conclusion: The p-value obtained from the Kolmogorov-Smirnov test is extremely small (below $= 0.05$, and close to zero), indicating strong evidence against the null hypothesis. Therefore, we reject the null hypothesis and conclude that the observed data does not follow the specified theoretical distribution (standard normal distribution).

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.

Solution Approach:
1. First, generate the dataset.
2. Define the negative log-likelihood function, which is used to minimize the sum of squared residuals during fitting.
3. Optimize the negative log-likelihood function using the BFGS algorithm to obtain the estimates of regression coefficients.
4. Use the lm() function to perform ordinary least squares linear regression and obtain the estimates of regression coefficients.
5. Compare the coefficients obtained from both methods to validate their equivalence.

```r
# Step 1: Generate data
```

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

# Define negative log-likelihood function
linear.lik <- function(theta, y, X) {
  n <- nrow(X)  # Number of observations
  k <- ncol(X)  # Number of predictors
  beta <- theta[1:k]  # Regression coefficients
  sigma2 <- theta[k+1]^2   # Variance
  e <- y - X %*% beta   # Residuals

  # Negative log-likelihood function
  logl <- -0.5 * n * log(2 * pi) - 0.5 * n * log(sigma2) - sum((t(e) %*% e) /
    (2 * sigma2))
  return(-logl)
}

# Optimize negative log-likelihood function using BFGS algorithm
opt_result <- optim(par = c(0, 0, 1), fn = linear.lik, y = data$y, X = cbind
    (1, data$x), method = "BFGS")
# Extract estimated coefficients from BFGS algorithm
bfgs_coefs <- optim_result$par

# Print coefficients from BFGS algorithm
cat("Coefficients from BFGS algorithm:")
print(bfgs_coefs)
# 0.139187 2.726699


# Estimated coefficients using BFGS algorithm
beta_hat <- opt_result$par
cat("Estimated coefficients using BFGS algorithm:", beta_hat)
# Estimated coefficients using BFGS algorithm: -0.4065592 -0.4595849 438.0535

# Compare with lm() function, Ordinary Least Squares Linear Regression
summary (lm(y ~ x, data))

#
# 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
```

Conclusion: Based on the summary statistics of the regression model, the linear regression model performs well in explaining the dependent variable. The estimated coefficients are statistically significant as indicated by their very small p-values. The high value of multiple R-squared, close to 1, indicates that the model can explain a large proportion of the variance in the dependent variable. The p-value of the F-statistic, approaching zero, suggests that the overall model fit is significant.

The OLS regression coefficients estimated using the Newton-Raphson algorithm with the BFGS optimization method are very close to those obtained using the lm() function, indicating their equivalence.