Applied Statistical Programming - Spring 2022

Alma Velazquez

Problem Set 2

Due Wednesday, February 16, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered in this Rmarkdown document with R code to accompany the R output. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded. Once your work is finished, submit the Rmd file as well as the knitted PDF to the appropriate problem set module on Canvas.
2. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
3. If you have any questions regarding the Problem Set, contact the TA or use office hours.
4. For students new to programming, this may take a while. Get started.

Benford's law

Recent work in political science has proposed Benford's law as a method for identifying electoral fraud. The idea is that specific integer totals should appear in the *first significant digit* a known number of times if the data is being generated "naturally.' '

1. Calculating violations

Two ways of testing violations of Benford's law are proposed below. Let X_i represent the observed proportional frequency of the integer i in observed vote totals. So, for example, X_1 would represent the proportion vote totals where the integer 1 appears in the first significant digit.

- Leemis' m statistic

$$m = max_{i=1}^{9}\Big\{(X_i) - log_{10}(1 + 1/i)\Big\}$$

- Cho-Gains' d

$$d = \sqrt{\sum_{i=1}^{9}\Big((X_i) - log_{10}(1 + 1/i)\Big)^2}$$

Write a function to calculate these statistics. The function should take as an input (i) a matrix or vector of election returns and (ii) an option (or options) that controls whether the m statistic should be calculated, the d statistic should be calculated, or both. The output should be a list containing the results, *including the full digit distribution.*

Define Function

```
benfords_stats <- function(x, d = TRUE, m = TRUE) {

    # Input must satisfy these things for the rest to work
    criteria <- is.numeric(x) & (is.vector(x) | is.matrix(x))
    if (!criteria) {
        stop("Error: x must be a vector or matrix input of type numeric.")
    }

    # Include logical inputs to define a method later
    stats <- list(d = d, m = m)

    # Extract the first significant figure from input Use table() and
    # prop.table() to get the proportional frequency of each digit
    first_sigfigs <- prop.table(table(factor(as.integer(substr(as.character(as.integer(signif(x,
        1))), 1, 1)), levels = 1:9)))

    # Reformat into a data frame and include it in the output
    dist <- data.frame(digit = as.numeric(names(first_sigfigs)), proportion = as.numeric(first_sigfigs)
    stats <- append(stats, list(digit_distribution = dist))

    # This difference per digit is common to both statistics of interest
    diff <- dist$proportion - log10(1 + (1/dist$digit))

    # Use logical options to calculate statistic(s), add to output
    if (m) {
        leemis <- max(diff)
        stats <- append(stats, list(m_statistic = leemis))
    }

    if (d) {
        cho_gains <- sqrt(sum((diff)^2))
        stats <- append(stats, list(d_statistic = cho_gains))

    }

    # Give the list object the 'benfords' class, to be used by our method
    class(stats) <- "benfords"
    return(stats)

}
```

Test Function

```
# Test with county-level vote data from the 2000 presidential election in
# Georgia
library(faraway)
data(gavote)

gore_votes <- gavote$gore
benf_gore <- benfords_stats(gore_votes)

bush_votes <- gavote$bush
```

```r
benf_bush <- benfords_stats(bush_votes)

# Test with population data - we shouldn't be able to observe fraud here
counties <- read.csv("https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/counties/total
    header = T)

births <- counties$BIRTHS2011
benf_births <- benfords_stats(births)
```

2. Critical values

For each statistic, we can reject the null hypothesis of *no fraud* if the statistic reaches the critical values in the table below.

	$\alpha = 0.10$	$\alpha = 0.05$	$\alpha = 0.01$
Leemis' m	0.341	0.691	0.875
Cho-Gains' d	0.391	0.651	0.933

Create a new function called `print.benfords()` that will output a table containing:

- The name of each statistic
- The statistic as it was calculated
- The relevant number of asterisk's (e.g., one star for significance at the $\alpha = .10$ level, etc.)
- A legend at the bottom explaining the asterisk's (similar to what you see when you print an `lm` object.).

You can provide this output in any way you like, but it must be clearly organized and easy to understand. Don't forget to document your code.

```r
print.benfords <- function(x) {
    # Need benfords class to contain at least one statistic to print
    criteria <- x$d | x$m

    if (!criteria) {
        stop("Statistics empty, only digit distribution in object. Access with <ObjectName>$digit_distr
    }

    # Define the symbols and the significance levels
    sym <- c(" ", "*", "**", "***")
    alpha <- c(1, 0.1, 0.05, 0.01, 0)

    # Create a custom legend the way it's done in symnum()
    legend <- paste(c(rbind(sapply(alpha, format), c(sQuote(sym), ""))), collapse = " ")

    # Initiate a vector
    output <- c()

    # Use logicals in the class object to determine outputs
    if (x$d) {
        # Define cutpoints for this statistic
        cpD <- c(0, 0.391, 0.651, 0.933, 1)
        # Use the symnum() function to get symbols given cutpoints
        symbolsD <- symnum(x$d_statistic, cpD, sym, legend = FALSE)
        # Include in output
        output[1] <- paste0(x$d_statistic, symbolsD)
    } else {
```

```
        output[1] <- "Empty"
    }

    # Repeat for the second statistic
    if (x$m) {
        cpM <- c(0, 0.341, 0.691, 0.875, 1)
        symbolsM <- symnum(x$m_statistic, cpM, sym, legend = FALSE)
        output[2] <- paste0(x$m_statistic, symbolsM)
    } else {
        output[2] <- "Empty"
    }

    # Give outputs names
    names(output) <- c("Cho-Gains' D Statistic", "Leemis' M Statistic")

    # Print statistics and the significance code legend
    print(noquote(output))
    cat("\nSignificance:", legend)
}


print(benf_births)
```

```
## Cho-Gains' D Statistic    Leemis' M Statistic
##     0.118190244013423     0.0551117484631158
##
## Significance: 1 ' ' 0.1 '*' 0.05 '**' 0.01 '***' 0
```

```
print(benf_gore)
```

```
## Cho-Gains' D Statistic    Leemis' M Statistic
##     0.154566053970107      0.100638300692746
##
## Significance: 1 ' ' 0.1 '*' 0.05 '**' 0.01 '***' 0
```

Neither of these tests yield values that allow us to reject the null hypothesis; this is especially weird for the population data.