R Notebook

```r
# Define functions for calculating estimates (to simplify code)

# For Normal and Poisson
get_estimate_NormPois <- function(data, indices) {
  return(mean(data[indices]))
}

# For Binomial Distribution
get_estimate_Binom <- function(data, indices) {
  return(mean(data[indices])/length(data[indices]))
}

# For Geometric Distribution
get_estimate_Geom <- function(data, indices) {
  return(1/mean(data[indices]))
}
```

```r
library(kernelboot)
library(boot)
```

Normal Distribution Simulation

```r
# Set simulation parameters
nsims = 100
mu = 0

# Initialize Data Matrix
normies_coverage <- matrix(nrow = nsims, ncol = 12)

for (i in 1:nsims) {
  # Generate data of different sample sizes
  normies10 = rnorm(10)
  normies30 = rnorm(30)
  normies100 = rnorm(100)

  # Initialize row vector to be inserted into data matrix
  cov.i <- c()

  #---------------------------------------------------------------------------
  # Calculate t-based confidence intervals
  #---------------------------------------------------------------------------

  #results <- as.numeric(nsims)
```

```r
#t.int <- matrix(FALSE, sims, 2)

# Construct t-based Lower and Upper bound CI for data of size n
lower10t <- t.test(normies10, conf.level = 0.95)$conf.int[1]
upper10t <- t.test(normies10, conf.level = 0.95)$conf.int[2]

# Do the same for the rest of the n sample sizes
lower30t <- t.test(normies30, conf.level = 0.95)$conf.int[1]
upper30t <- t.test(normies30, conf.level = 0.95)$conf.int[2]

lower100t <- t.test(normies100, conf.level = 0.95)$conf.int[1]
upper100t <- t.test(normies100, conf.level = 0.95)$conf.int[2]

# Get bool on whether confidence intervals contain the true parameter
# and append that to cov.i, the row matrix we will add to the entire simulation's
# Data Matrix
cov.i <- c(cov.i, as.numeric((lower10t <= mu & upper10t >= mu)),
          (lower30t <= mu & upper30t >= mu),
          (lower100t <= mu & upper100t >= mu))

#------------------------------------------------------------------------------
# Calculate Boot confidence intervals
#------------------------------------------------------------------------------

# Create boot objects for the three sample sizes
# These boot objects are used for the percentile, basic, and normal boot methods
normies10.boot <- boot(
    data = normies10,
    statistic = get_estimate_NormPois,
    R = 100
)
normies30.boot <- boot(
    data = normies30,
    statistic = get_estimate_NormPois,
    R = 100
)
normies100.boot <- boot(
    data = normies100,
    statistic = get_estimate_NormPois,
    R = 100
)

# This whole loop calculates the three boot types in the list below
boot_types <- c("perc", "basic", "norm")

for (j in 1:(length(boot_types))) {
  subt <- 0
  if (boot_types[j] == "norm") {subt = 2}

  # Calculate lower and upper bounds for boot CI as before
  lower10 = boot.ci(normies10.boot, type = boot_types[j])[[4]][4-subt]
  upper10 = boot.ci(normies10.boot, type = boot_types[j])[[4]][5-subt]
```

```
    lower30 = boot.ci(normies30.boot, type = boot_types[j])[[4]][4-subt]
    upper30 = boot.ci(normies30.boot, type = boot_types[j])[[4]][5-subt]

    lower100 = boot.ci(normies100.boot, type = boot_types[j])[[4]][4-subt]
    upper100 = boot.ci(normies100.boot, type = boot_types[j])[[4]][5-subt]

    # Append whether the true parameter is covered to dataframe
    cov.i <- c(cov.i, (lower10 <= mu & upper10 >= mu),
               (lower30 <= mu & upper30 >= mu),
               (lower100 <= mu & upper100 >= mu))
  }

  # Lastly, we calculate the 'Smooth' Bootstrap Confidence Intervals
  boot10 <- kernelboot(data=normies10, statistic=get_estimate_NormPois)
  lower10 <- summary(boot10)[3]; upper10 <- summary(boot10)[5]

  boot30 <- kernelboot(data=normies30, statistic=get_estimate_NormPois)
  lower30 <- summary(boot30)[3]; upper30 <- summary(boot30)[5]

  boot100 <- kernelboot(data=normies100, statistic=get_estimate_NormPois)
  lower100 <- summary(boot100)[3]; upper100 <- summary(boot100)[5]

  # Set row of data matrix equal to the vector of data we calculated for this
  # individual simulation
  normies_coverage[i,] <- cov.i
}

# Print Coverage rate
for (i in 1:ncol(normies_coverage)) {
  print(mean(normies_coverage[,i]))
  }
```

```
## [1] 1
## [1] 0.92
## [1] 0.91
## [1] 0.95
## [1] 0.91
## [1] 0.91
## [1] 0.94
## [1] 0.92
## [1] 0.95
## [1] 0.95
## [1] 0.88
## [1] 0.91
```