R Notebook

```r
library(kernelboot)
# library(distributions3)
library(boot)
set.seed(123)
```

```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]))
}

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

# For Exponential Distribution
get_estimate_Expo <- function(data, indices) {
  return(mean(data[indices]))
}

# Function for printing results
print_coverage <- function(data, widths) {
  # Get mean coverage probabilities
  vec <- c()
  for (i in 1:ncol(data)) {
     vec <- c(vec, mean(data[,i]))
  }
  # Get mean widths
  vec2 <- c()
  for (i in 1:ncol(widths)) {
     vec2 <- c(vec2, mean(widths[,i]))
  }

  tmp.df <- data.frame(matrix(ncol=2, nrow=15))
  colnames(tmp.df) <- c('coverage probability', 'average width')
  rownames(tmp.df) <- c('t-based 10', 't-based 30', 't-based 100',
                        'perc boot 10', 'perc boot 30', 'perc boot 100',
                        'basic boot 10', 'basic boot 30', 'basic boot 100',
                        'norm boot 10', 'norm boot 30', 'norm boot 100',
```

```r
                        'smooth boot 10', 'smooth boot 30', 'smooth boot 100')
  tmp.df[,1] <- vec
  tmp.df[,2] <- vec2
  print(tmp.df)
}
```

Normal Distribution Simulation

```r
# Create simulation variables
nsims = 100
mu = 0

# Initialize Data Matrix
normies_coverage <- matrix(nrow = nsims, ncol = 15)
normies_width <- matrix(nrow = nsims, ncol = 15)

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()
  width.i <- c()

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

  # 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))

  width.i <- c(width.i, (upper10t - lower10t),
                (upper30t - lower30t),
                (upper100t - lower100t))
```

```r
#----------------------------------------------------------------------------
# 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 = 200
)
normies30.boot <- boot(
    data = normies30,
    statistic = get_estimate_NormPois,
    R = 200
)
normies100.boot <- boot(
    data = normies100,
    statistic = get_estimate_NormPois,
    R = 200
)

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

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))

  width.i <- c(width.i, (upper10 - lower10),
              (upper30 - lower30),
              (upper100 - lower100))
}

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

```r
  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]

  cov.i <- c(cov.i, (lower10 <= mu & upper10 >= mu),
             (lower30 <= mu & upper30 >= mu),
             (lower100 <= mu & upper100 >= mu))

  width.i <- c(width.i, (upper10 - lower10),
             (upper30 - lower30),
             (upper100 - lower100))
  # Set row of data matrix equal to the vector of data we calculated for this
  # individual simulation

  normies_coverage[i,] <- cov.i
  normies_width[i,] <- width.i
}

# Print Coverage rate
print_coverage(normies_coverage, normies_width)
```

```
##                  coverage probability average width
## t-based 10                       0.94     1.3252976
## t-based 30                       0.96     0.7461156
## t-based 100                      0.94     0.3944169
## perc boot 10                     0.91     1.1056763
## perc boot 30                     0.93     0.7134050
## perc boot 100                    0.93     0.3912454
## basic boot 10                    0.89     1.1056763
## basic boot 30                    0.96     0.7134050
## basic boot 100                   0.92     0.3912454
## norm boot 10                     0.89     1.1413487
## norm boot 30                     0.94     0.7210337
## norm boot 100                    0.92     0.3914425
## smooth boot 10                   0.89     1.0831327
## smooth boot 30                   0.91     0.6956929
## smooth boot 100                  0.93     0.3821216
```

Geometric Distribution Simulation

```r
# Create simulation variables
nsims = 100
p = 0.15
mean = (1-p)/p

# Initialize Data Matrix
geomies_coverage <- matrix(nrow = nsims, ncol = 15)
geomies_width <- matrix(nrow = nsims, ncol = 15)
```

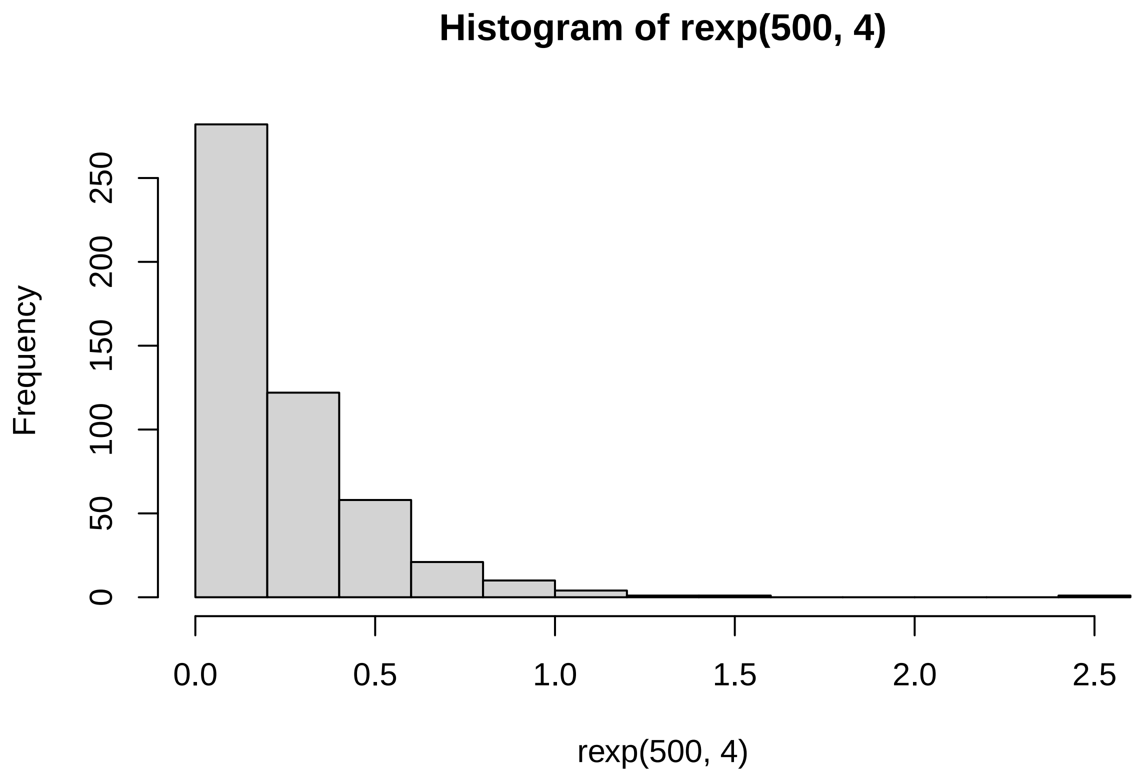
```r
for (i in 1:nsims) {
#i = 3
  # Generate data of different sample sizes
  geomies10 = rgeom(10, p)
  geomies30 = rgeom(30, p)
  geomies100 = rgeom(100, p)

  # Initialize row vector to be inserted into data matrix
  cov.i <- c()
  width.i <- c()
  #---------------------------------------------------------------------------
  # Calculate t-based confidence intervals
  #---------------------------------------------------------------------------

  # Construct t-based Lower and Upper bound CI for data of size n
  lower10t <- t.test(geomies10, conf.level = 0.95, )$conf.int[1]
  upper10t <- t.test(geomies10, conf.level = 0.95)$conf.int[2]

  # Do the same for the rest of the n sample sizes
  lower30t <- t.test(geomies30, conf.level = 0.95)$conf.int[1]
  upper30t <- t.test(geomies30, conf.level = 0.95)$conf.int[2]

  lower100t <- t.test(geomies100, conf.level = 0.95)$conf.int[1]
  upper100t <- t.test(geomies100, conf.level = 0.95)$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 <= mean & upper10t >= mean)),
            (lower30t <= mean & upper30t >= mean),
            (lower100t <= mean & upper100t >= mean))

  width.i <- c(width.i, (upper10t - lower10t),
               (upper30t - lower30t),
               (upper100t - lower100t))
  #---------------------------------------------------------------------------
  # 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
  geomies10.boot <- boot(
      data = geomies10,
      statistic = get_estimate_Geom,
      R = 200
  )
  geomies30.boot <- boot(
      data = geomies30,
      statistic = get_estimate_Geom,
      R = 200
  )
  geomies100.boot <- boot(
      data = geomies100,
```

```r
    statistic = get_estimate_Geom,
    R = 200
)

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

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

  if (sum(geomies10) == 0) {
    lower10 = 0
    upper10 = 0
  }
  else {
    # Calculate lower and upper bounds for boot CI as before
    lower10 = boot.ci(geomies10.boot, type = boot_types[j])[[4]][4-subt]
    upper10 = boot.ci(geomies10.boot, type = boot_types[j])[[4]][5-subt]
  }
  # Calculate lower and upper bounds for boot CI as before
  # lower10 = boot.ci(geomies10.boot, type = boot_types[j])[[4]][4-subt]
  # upper10 = boot.ci(geomies10.boot, type = boot_types[j])[[4]][5-subt]

  lower30 = boot.ci(geomies30.boot, type = boot_types[j])[[4]][4-subt]
  upper30 = boot.ci(geomies30.boot, type = boot_types[j])[[4]][5-subt]

  lower100 = boot.ci(geomies100.boot, type = boot_types[j])[[4]][4-subt]
  upper100 = boot.ci(geomies100.boot, type = boot_types[j])[[4]][5-subt]

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

  width.i <- c(width.i, (upper10 - lower10),
              (upper30 - lower30),
              (upper100 - lower100))
}

# Lastly, we calculate the 'Smooth' Bootstrap Confidence Intervals
boot10 <- kernelboot(data=geomies10, statistic=get_estimate_NormPois)
lower10 <- summary(boot10)[3]; upper10 <- summary(boot10)[5]

boot30 <- kernelboot(data=geomies30, statistic=get_estimate_NormPois)
lower30 <- summary(boot30)[3]; upper30 <- summary(boot30)[5]

boot100 <- kernelboot(data=geomies100, statistic=get_estimate_NormPois)
lower100 <- summary(boot100)[3]; upper100 <- summary(boot100)[5]

cov.i <- c(cov.i, (lower10 <= mean & upper10 >= mean),
             (lower30 <= mean & upper30 >= mean),
             (lower100 <= mean & upper100 >= mean))
```

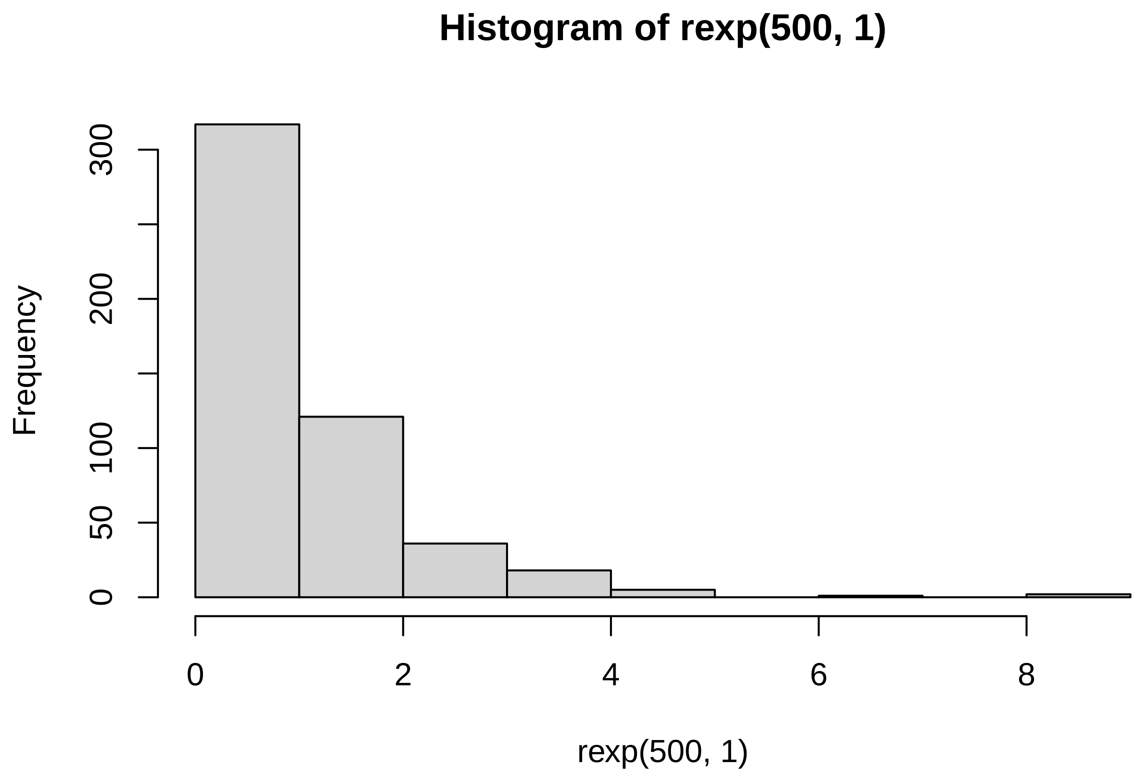
```r
  width.i <- c(width.i, (upper10 - lower10),
               (upper30 - lower30),
               (upper100 - lower100))
  # Set row of data matrix equal to the vector of data we calculated for this
  # individual simulation

  geomies_coverage[i,] <- cov.i
  geomies_width[i,] <- width.i
}

# Print Coverage rate
print_coverage(geomies_coverage, geomies_width)
```

```
##                  coverage probability average width
## t-based 10                       0.94      8.486601
## t-based 30                       0.91      4.327301
## t-based 100                      0.97      2.430591
## perc boot 10                     0.90      6.932344
## perc boot 30                     0.90      4.173570
## perc boot 100                    0.96      2.421253
## basic boot 10                    0.86      6.932344
## basic boot 30                    0.91      4.173570
## basic boot 100                   0.95      2.421253
## norm boot 10                     0.93      7.514485
## norm boot 30                     0.92      4.341231
## norm boot 100                    0.97      2.482763
## smooth boot 10                   0.88      6.886527
## smooth boot 30                   0.89      4.056863
## smooth boot 100                  0.97      2.367238
```

```r
hist(rexp(500, 4))
```




```r
hist(rexp(500, 1))
```




```
# hist(rgeom(50, 0.15))
```

Exponential Distribution Simulation

```
# Create simulation variables
nsims = 100
lambda = 4
mean = 1/lambda

# Initialize Data Matrix
expos_coverage <- matrix(nrow = nsims, ncol = 15)
expos_width <- matrix(nrow = nsims, ncol = 15)

for (i in 1:nsims) {
  # Generate data of different sample sizes
  expos10 = rexp(10, lambda)
  expos30 = rexp(30, lambda)
  expos100 = rexp(100, lambda)

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

```r
#-----------------------------------------------------------------------------
# Calculate t-based confidence intervals
#-----------------------------------------------------------------------------

# Construct t-based Lower and Upper bound CI for data of size n
lower10t <- t.test(expos10, conf.level = 0.95)$conf.int[1]
upper10t <- t.test(expos10, conf.level = 0.95)$conf.int[2]

# Do the same for the rest of the n sample sizes
lower30t <- t.test(expos30, conf.level = 0.95)$conf.int[1]
upper30t <- t.test(expos30, conf.level = 0.95)$conf.int[2]

lower100t <- t.test(expos100, conf.level = 0.95)$conf.int[1]
upper100t <- t.test(expos100, conf.level = 0.95)$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 <= mean & upper10t >= mean)),
          (lower30t <= mean & upper30t >= mean),
          (lower100t <= mean & upper100t >= mean))

width.i <- c(width.i, (upper10t - lower10t),
             (upper30t - lower30t),
             (upper100t - lower100t))
#-----------------------------------------------------------------------------
# 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
expos10.boot <- boot(
    data = expos10,
    statistic = get_estimate_Expo,
    R = 200
)
expos30.boot <- boot(
    data = expos30,
    statistic = get_estimate_Expo,
    R = 200
)
expos100.boot <- boot(
    data = expos100,
    statistic = get_estimate_Expo,
    R = 200
)

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

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(expos10.boot, type = boot_types[j])[[4]][4-subt]
    upper10 = boot.ci(expos10.boot, type = boot_types[j])[[4]][5-subt]

    lower30 = boot.ci(expos30.boot, type = boot_types[j])[[4]][4-subt]
    upper30 = boot.ci(expos30.boot, type = boot_types[j])[[4]][5-subt]

    lower100 = boot.ci(expos100.boot, type = boot_types[j])[[4]][4-subt]
    upper100 = boot.ci(expos100.boot, type = boot_types[j])[[4]][5-subt]

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

    width.i <- c(width.i, (upper10 - lower10),
                (upper30 - lower30),
                (upper100 - lower100))
  }

  # Lastly, we calculate the 'Smooth' Bootstrap Confidence Intervals
  kernel.type = 'epanechnikov'
  boot10 <- kernelboot(data=expos10, statistic=get_estimate_NormPois, kernel = kernel.type)
  lower10 <- summary(boot10)[3]; upper10 <- summary(boot10)[5]

  boot30 <- kernelboot(data=expos30, statistic=get_estimate_NormPois, kernel = kernel.type)
  lower30 <- summary(boot30)[3]; upper30 <- summary(boot30)[5]

  boot100 <- kernelboot(data=expos100, statistic=get_estimate_NormPois, kernel = kernel.type)
  lower100 <- summary(boot100)[3]; upper100 <- summary(boot100)[5]

  cov.i <- c(cov.i, (lower10 <= mean & upper10 >= mean),
             (lower30 <= mean & upper30 >= mean),
             (lower100 <= mean & upper100 >= mean))

  width.i <- c(width.i, (upper10 - lower10),
             (upper30 - lower30),
             (upper100 - lower100))
  # Set row of data matrix equal to the vector of data we calculated for this
  # individual simulation

  expos_coverage[i,] <- cov.i
  expos_width[i,] <- width.i
}

# Print Coverage rate
print_coverage(expos_coverage, expos_width)
```

```
##                  coverage probability average width
## t-based 10                       0.90    0.31175870
## t-based 30                       0.90    0.17524661
## t-based 100                      0.93    0.09884636
## perc boot 10                     0.86    0.25472272
```

```
## perc boot 30                     0.88    0.16587256
## perc boot 100                    0.95    0.09954611
## basic boot 10                    0.82    0.25472272
## basic boot 30                    0.88    0.16587256
## basic boot 100                   0.95    0.09954611
## norm boot 10                     0.87    0.27807644
## norm boot 30                     0.92    0.17302320
## norm boot 100                    0.96    0.10274836
## smooth boot 10                   0.84    0.25265672
## smooth boot 30                   0.89    0.16368318
## smooth boot 100                  0.91    0.09593230
```

Poisson Distribution Simulation

```r
# Create simulation variables
nsims = 100
lambda = 4

# Initialize Data Matrix
pois_coverage <- matrix(nrow = nsims, ncol = 15)
pois_width <- matrix(nrow = nsims, ncol = 15)

for (i in 1:nsims) {
  # Generate data of different sample sizes
  pois10 = rpois(10, lambda)
  pois30 = rpois(30, lambda)
  pois100 = rpois(100, lambda)

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

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

  # Construct t-based Lower and Upper bound CI for data of size n
  lower10t <- t.test(pois10, conf.level = 0.95)$conf.int[1]
  upper10t <- t.test(pois10, conf.level = 0.95)$conf.int[2]

  # Do the same for the rest of the n sample sizes
  lower30t <- t.test(pois30, conf.level = 0.95)$conf.int[1]
  upper30t <- t.test(pois30, conf.level = 0.95)$conf.int[2]

  lower100t <- t.test(pois100, conf.level = 0.95)$conf.int[1]
  upper100t <- t.test(pois100, conf.level = 0.95)$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 <= lambda & upper10t >= lambda)),
            (lower30t <= lambda & upper30t >= lambda),
```

```r
          (lower100t <= lambda & upper100t >= lambda))

width.i <- c(width.i, (upper10t - lower10t),
             (upper30t - lower30t),
             (upper100t - lower100t))
#----------------------------------------------------------------------------
# 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
pois10.boot <- boot(
    data = pois10,
    statistic = get_estimate_NormPois,
    R = 200
)
pois30.boot <- boot(
    data = pois30,
    statistic = get_estimate_NormPois,
    R = 200
)
pois100.boot <- boot(
    data = pois100,
    statistic = get_estimate_NormPois,
    R = 200
)

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

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(pois10.boot, type = boot_types[j])[[4]][4-subt]
  upper10 = boot.ci(pois10.boot, type = boot_types[j])[[4]][5-subt]

  lower30 = boot.ci(pois30.boot, type = boot_types[j])[[4]][4-subt]
  upper30 = boot.ci(pois30.boot, type = boot_types[j])[[4]][5-subt]

  lower100 = boot.ci(pois100.boot, type = boot_types[j])[[4]][4-subt]
  upper100 = boot.ci(pois100.boot, type = boot_types[j])[[4]][5-subt]

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

  width.i <- c(width.i, (upper10 - lower10),
              (upper30 - lower30),
              (upper100 - lower100))
}
```

```r
  # Lastly, we calculate the 'Smooth' Bootstrap Confidence Intervals
  boot10 <- kernelboot(data=pois10, statistic=get_estimate_NormPois)
  lower10 <- summary(boot10)[3]; upper10 <- summary(boot10)[5]

  boot30 <- kernelboot(data=pois30, statistic=get_estimate_NormPois)
  lower30 <- summary(boot30)[3]; upper30 <- summary(boot30)[5]

  boot100 <- kernelboot(data=pois100, statistic=get_estimate_NormPois)
  lower100 <- summary(boot100)[3]; upper100 <- summary(boot100)[5]

  cov.i <- c(cov.i, (lower10 <= lambda & upper10 >= lambda),
             (lower30 <= lambda & upper30 >= lambda),
             (lower100 <= lambda & upper100 >= lambda))

  width.i <- c(width.i, (upper10 - lower10),
            (upper30 - lower30),
            (upper100 - lower100))
  # Set row of data matrix equal to the vector of data we calculated for this
  # individual simulation

  pois_coverage[i,] <- cov.i
  pois_width[i,] <- width.i
}

# Print Coverage rate
print_coverage(pois_coverage, pois_width)
```

```
##                  coverage probability average width
## t-based 10                       0.93     2.7261470
## t-based 30                       0.97     1.4693531
## t-based 100                      0.96     0.7967708
## perc boot 10                     0.90     2.2508972
## perc boot 30                     0.96     1.4200941
## perc boot 100                    0.97     0.7927454
## basic boot 10                    0.89     2.2508972
## basic boot 30                    0.95     1.4200941
## basic boot 100                   0.97     0.7927454
## norm boot 10                     0.89     2.3061336
## norm boot 30                     0.95     1.4155437
## norm boot 100                    0.98     0.7945610
## smooth boot 10                   0.87     2.2261894
## smooth boot 30                   0.95     1.3745746
## smooth boot 100                  0.97     0.7791635
```

Binomial Distribution Simulation

```r
# Create simulation variables
nsims = 100
p = 0.5
size = 50
mean = size*p
```

```r
# Initialize Data Matrix
bin_coverage <- matrix(nrow = nsims, ncol = 15)
bin_width <- matrix(nrow = nsims, ncol = 15)

for (i in 1:nsims) {
  # Generate data of different sample sizes
  bin10 = rbinom(10, size, p)
  bin30 = rbinom(30, size, p)
  bin100 = rbinom(100, size, p)

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

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

  # Construct t-based Lower and Upper bound CI for data of size n
  lower10t <- t.test(bin10, conf.level = 0.95)$conf.int[1]
  upper10t <- t.test(bin10, conf.level = 0.95)$conf.int[2]

  # Do the same for the rest of the n sample sizes
  lower30t <- t.test(bin30, conf.level = 0.95)$conf.int[1]
  upper30t <- t.test(bin30, conf.level = 0.95)$conf.int[2]

  lower100t <- t.test(bin100, conf.level = 0.95)$conf.int[1]
  upper100t <- t.test(bin100, conf.level = 0.95)$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 <= mean & upper10t >= mean)),
             (lower30t <= mean & upper30t >= mean),
             (lower100t <= mean & upper100t >= mean))
  width.i <- c(width.i, (upper10t - lower10t),
              (upper30t - lower30t),
              (upper100t - lower100t))

  #-----------------------------------------------------------------------------
  # 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
  bin10.boot <- boot(
      data = bin10,
      statistic = get_estimate_Binom,
      R = 200
  )
  bin30.boot <- boot(
      data = bin30,
      statistic = get_estimate_Binom,
```

```
    R = 200
)
bin100.boot <- boot(
    data = bin100,
    statistic = get_estimate_Binom,
    R = 200
)

# 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(bin10.boot, type = boot_types[j])[[4]][4-subt]
  upper10 = boot.ci(bin10.boot, type = boot_types[j])[[4]][5-subt]

  lower30 = boot.ci(bin30.boot, type = boot_types[j])[[4]][4-subt]
  upper30 = boot.ci(bin30.boot, type = boot_types[j])[[4]][5-subt]

  lower100 = boot.ci(bin100.boot, type = boot_types[j])[[4]][4-subt]
  upper100 = boot.ci(bin100.boot, type = boot_types[j])[[4]][5-subt]

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

# Lastly, we calculate the 'Smooth' Bootstrap Confidence Intervals
boot10 <- kernelboot(data=bin10, statistic=get_estimate_NormPois)
lower10 <- summary(boot10)[3]; upper10 <- summary(boot10)[5]

boot30 <- kernelboot(data=bin30, statistic=get_estimate_NormPois)
lower30 <- summary(boot30)[3]; upper30 <- summary(boot30)[5]

boot100 <- kernelboot(data=bin100, statistic=get_estimate_NormPois)
lower100 <- summary(boot100)[3]; upper100 <- summary(boot100)[5]

cov.i <- c(cov.i, (lower10 <= mean & upper10 >= mean),
             (lower30 <= mean & upper30 >= mean),
             (lower100 <= mean & upper100 >= mean))
width.i <- c(width.i, (upper10 - lower10),
            (upper30 - lower30),
            (upper100 - lower100))
# Set row of data matrix equal to the vector of data we calculated for this
# individual simulation
```

```r
  bin_coverage[i,] <- cov.i
  bin_width[i,] <- width.i
}

# Print Coverage rate
print_coverage(bin_coverage, bin_width)
```

```
##                  coverage probability average width
## t-based 10                       0.99      4.966748
## t-based 30                       0.94      2.650000
## t-based 100                      0.96      1.377743
## perc boot 10                     0.93      4.132767
## perc boot 30                     0.93      2.520223
## perc boot 100                    0.96      1.379153
## basic boot 10                    0.92      4.132767
## basic boot 30                    0.92      2.520223
## basic boot 100                   0.95      1.379153
## norm boot 10                     0.93      4.086848
## norm boot 30                     0.91      2.478877
## norm boot 100                    0.93      1.354555
## smooth boot 10                   0.91      4.044048
## smooth boot 30                   0.93      2.491637
## smooth boot 100                  0.95      1.332514
```