An evaluation of various ATE estimators through Non-parametric bootstrapping

Josiah Davis

4/28/2017

Define Functions

```r
get_data <- function(dir, file_in){
  # Get the data-frame containing predicted propensity, observed and counterfactual outcomes
  #
  # Args:
  #   dir: working directory as character vector
  #   file_in: name of the file as character vector
  #
  # Returns:
  #   dataframe object with no transformations

  df <- read.csv(paste0(dir, file_in), stringsAsFactors = FALSE)
  return(df)
}

calc_sse <- function(df){
  # Calculate the simple substitution estimator
  #
  # Args:
  #   df: data-frame containing the conditional probabiliy of leaving under both exposures
  #
  # Returns:
  #   The average treatement effect using the simple substitution estimator

  # Simple sibstitution estimator of the ATE
  psi_hat_ss <- mean(df$Q_bar_1W - df$Q_bar_0W)
  return(psi_hat_ss)
}

calc_iptw <- function(df){
  # Calculate the IPTW estimator
  #
  # Args:
  #   df: data-frame containing the conditional probabiliy of leaving under both exposures
  #
  # Returns:
  #   The average treatement effect using IPTW estimator of the ATE

  num1 <- mean(as.numeric(df$salary==1)*df$left/df$g_hat_AW)
  num2 <- mean(as.numeric(df$salary==0)*df$left/df$g_hat_AW)
  psi_hat_iptw <- num1 - num2
  return(psi_hat_iptw)
}
```

```
calc_iptw_stable <- function(df){
  # Calculate the stabilized IPTW estimator
  #
  # Args:
  #   df: data-frame containing the conditional probabiliy of leaving under both exposures
  #
  # Returns:
  #   The average treatement effect using the stablized IPTW estimator of the ATE

  wt <- 1 / df$g_hat_AW
  num1 <- mean( wt*as.numeric(df$salary==1)*df$left)/mean( wt*as.numeric(df$salary==1))
  num2 <- mean( wt*as.numeric(df$salary==0)*df$left)/mean( wt*as.numeric(df$salary==0))
  psi_hat_iptw_stable <- num1 - num2
  return(psi_hat_iptw_stable)
}

calc_tmle <- function(df){
  # Calculate the Targeted Maximum Likelihood Estimator (TMLE)
  #
  # Args:
  #   df: data-frame containing the conditional probabiliy of leaving under both exposures
  #
  # Returns:
  #   The average treatement effect using IPTW estimator of the ATE

  # Use propensity score estiamtes to create the clever covariate
  H_AW <- as.numeric(df$salary==1)/df$g_hat_AW - as.numeric(df$salary==0)/df$g_hat_AW
  H_1W <- 1/df$g_hat_AW
  H_0W <- -1/df$g_hat_AW

  # Update the initial estimates using logistic regression
  # Note: -1 supresses the intercept
  logit_update <- glm(df$left ~ -1 + offset(qlogis(df$Q_bar_AW)) + H_AW, family='binomial')
  eps <- logit_update$coef
  Q_bar_AW_star <- plogis(qlogis(df$Q_bar_AW) + eps*H_AW)
  Q_bar_1W_star <- plogis(qlogis(df$Q_bar_1W) + eps*H_1W)
  Q_bar_0W_star <- plogis(qlogis(df$Q_bar_0W) + eps*H_0W)

  # Substitute the updated fits into the target parameter mapping:
  psi_hat_tmle <- mean(Q_bar_1W_star) - mean(Q_bar_0W_star)
  return(psi_hat_tmle)
}

calc_ci <- function(m, method){
  # Calculate a confidence interval using a normality assumption
  #
  # Args:
  #   m: Matrix with bootstrapped iterations across rows and estimators across columns
  #   method: String indicating how to compute the CIs (i.e., 'normal' OR 'quantile')
  #
  # Returns:
  #   Confidence intervals for all estimators in a data-frame
  if(method == 'Normal'){
    cis <- apply(m, 2, function(x){
      mean(x) + c(-1.96 * sqrt(var(x)), 1.96 * sqrt(var(x)))
    })
  }else if(method == 'Quantile'){
    cis <- apply(m, 2, function(x){
      quantile(x, c(0.025, 0.975))
    })
  }
  cis <- data.frame(t(cis))
  colnames(cis) <- c('lower', 'upper')
  cis$method <- rep(method, 4)
  cis$estimator <- c('SSE', 'IPTW', 'IPTW.ST', 'TMLE')
  cis$coverage <- cis$upper - cis$lower
  cis$variable <- factor(cis$estimator, levels = c('SSE', 'IPTW', 'IPTW.ST', 'TMLE'), labels = PSI_LABS
  return(cis[,c('estimator', 'lower', 'upper', 'coverage', 'method', 'variable')])
}

run_boot <- function(df, n_boot){
  # Conduct bootstrapping of various ATE estimators
  #
  # Args:
  #   df: The observed data along with counterfactual prediction
  #       probabilities and treatment mechanism predictions
  #   n_boot: The number of boostrapping iterations to use.
  #
  # Returns:
  #   Bootstrapped estimates of the various ATE estimators as a matrix

  n <- nrow(df)
  est_boot <- matrix(nrow = n_boot, ncol = 4)
  for(b in 1:n_boot){
    d_boot <- df[sample(1:n, replace = TRUE),]
    est_boot[b,] <- c(calc_sse(d_boot)
                      , calc_iptw(d_boot)
                      , calc_iptw_stable(d_boot)
                      , calc_tmle(d_boot))
  }
  return(est_boot)
}

format_boot <- function(est){
  # Format results into data-frame for plotting
  #
  # Args:
  #   est: Bootstrapped estimators as a matrix
  #
  # Returns:
  #   formatted data-frame for easier plotting

  est <- data.frame(estimators)
  colnames(est) <- c('SSE', 'IPTW', 'IPTW.ST', 'TMLE')
  est <- est %>% melt()
```

```r
  est$variable <- factor(est$variable
                         , levels = c('SSE', 'IPTW', 'IPTW.ST', 'TMLE')
                         , labels = PSI_LABS)
  return(est)
}

visualize_boot <- function(est, ci){
  # Visualize bootstrapped results of various ATE estimators.
  #
  # Args:
  #   est: Bootstrapped estimators as a long data-frame
  #   ci: Confidence intervals as a data-frame
  #
  # Returns:
  #   ggplot object with the ditributions highlighting the
  #   variability of the estimators.

  # Format the estimators for plotting
  est <- format_boot(est)

  # Create the plot
  plt <- ggplot(est, aes(x = value, fill = variable)) +
    geom_density() +
    geom_vline(data = ci, aes(xintercept = lower, linetype = method)) +
    geom_vline(data = ci, aes(xintercept = upper, linetype = method)) +
    facet_wrap(~variable, ncol = 1, labeller = label_parsed)  +
    scale_fill_manual(values = PSI_COLS[1:4], guide=FALSE) +
    theme(legend.title=element_blank()) +
    labs(title = expression(paste('Comparison of Estimators '
                                  , hat(Psi)(P[n])
                                  , ' using Non-parametric Bootstrap'))
         , x = expression(hat(Psi)(P[boot]))
         , y = '')
  return(plt)
}
```

Define Constants

```r
DIR <- '../../data/'
FILE_IN_NAME <- 'SL_output.csv'
FILE_OUT_NAME <- 'est_output.csv'

PSI_COLS <- c('#1f78b4', '#b2df8a', '#33a02c', '#fb9a99'
              , '#e31a1c','#fdbf6f', '#ff7f00', '#cab2d6', '#6a3d9a', '#a6cee3')

PSI_LABS <- c('hat(Psi)[SSE](P[n]^b)', 'hat(Psi)[IPTW](P[n]^b)' , 'hat(Psi)[IPTW.ST](P[n]^b)', 'hat(Psi
```

Run everything

```r
library(reshape2)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

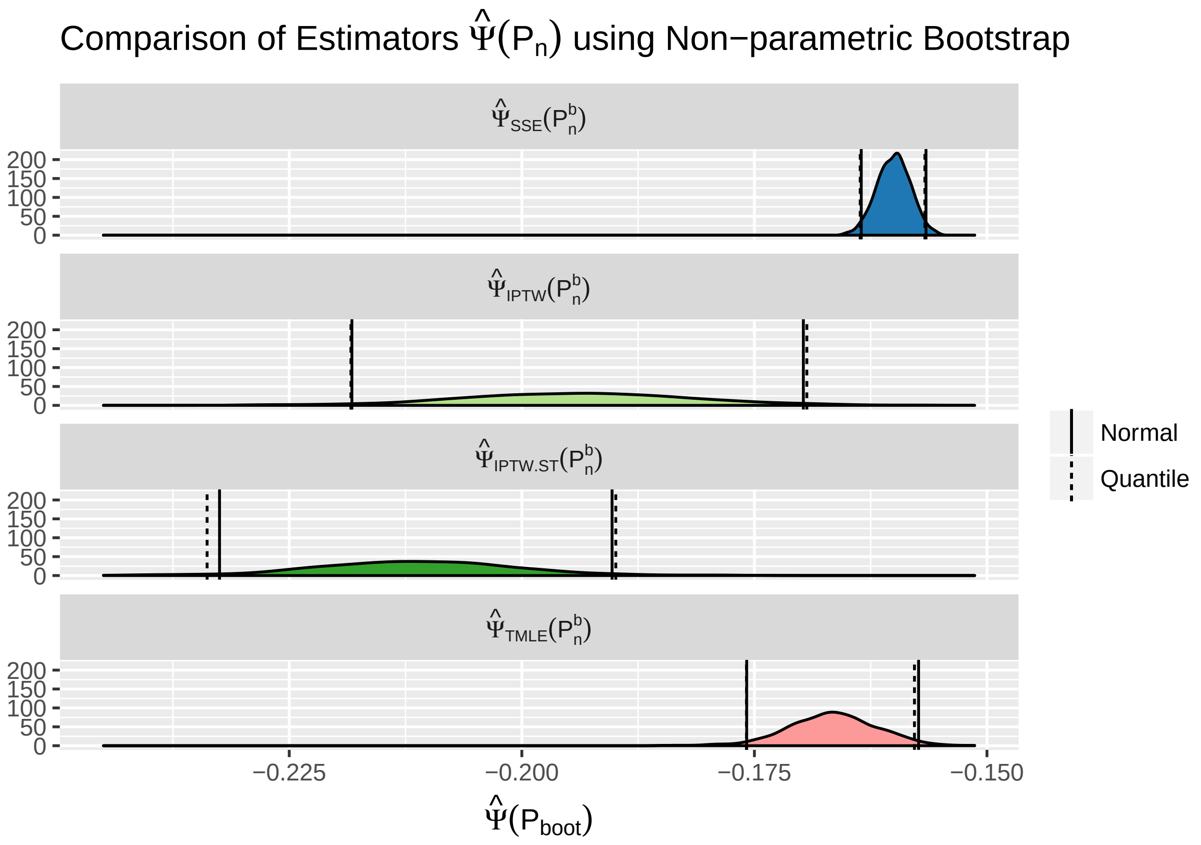
```r
# Get the data
d <- get_data(DIR, FILE_IN_NAME)

# Run the bootstrapping
estimators <- run_boot(d, 500)

# Calculate the confidence intervals
cis <- rbind(calc_ci(estimators, 'Normal'), calc_ci(estimators, 'Quantile'))

# Visualize the densities
visualize_boot(estimators, cis)
```

```
## No id variables; using all as measure variables
```



```r
# Write to file
cis %>% select(-variable) %>% write.csv(paste0(DIR, 'confidence_intervals.csv'), row.names = FALSE)
```