Treatment Effects in Nested Subgroups

Jean Morrison

January 1, 2017

This document accompanies the paper "Rank conditional coverage and confidence intervals in high dimensional problems." by Jean Morrison and Noah Simon. Here we walk through the simulations in section 3.2 and reproduce the results shown in the paper. We make use of the `rcc` and `rccSims` packages which accompanies the paper and can be found at github.com/jean997/rcc and github.com/jean997/rccSims.

Simulation Set-up

Here we consider data from a clinical trial examining the effect of a a treatment on an outcome Y. For each participant we have also collected a biomarker W and suspect that the treatment has a greater effect for patients with larger values of W. We are interested in establishing a cut-point in W to guide enrolment of a future trial so we will estimate the treatment effect in subgroups defined by the biomarker.

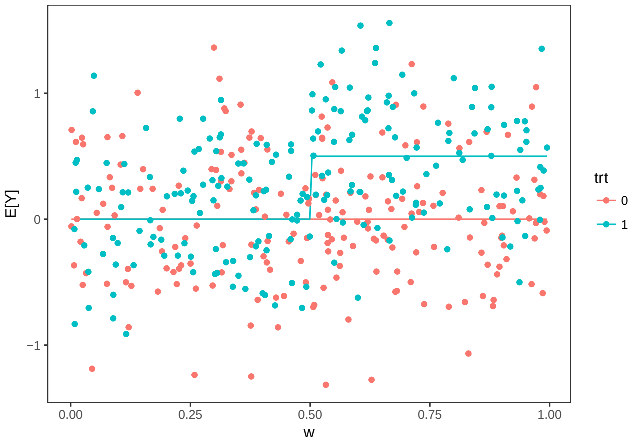
The relationship between Y, the treatment, and the biomarker is given by

$$E[Y|trt, W] = \begin{cases} 0 & W < 0.5 \\ \frac{1}{2} \cdot trt & W \geq 0.5 \end{cases}.$$

For participant i, the observed value of Y is $Y_i = E[Y|W_i, trt_i] + \epsilon$ where $\epsilon \sim N(0, 1/4)$.

In each simulation, we generate data for 400 participants — 200 in each treatment arm. The vlaue biomarker is uniformly distributed between 0 and 1. Here we generate data for one simulation:

```r
library(ggplot2)
library(rcc)
library(rccSims)
set.seed(1e7)
n <- 2*200
mean_outcome <- function(x, trt){
  if(x < 0.5) return(0)
  return(0.5*trt)
}
dat <- data.frame("trt"=rep(c(0, 1), each=n/2), "w"=runif(n=n))
dat$Ey <- apply(dat, MARGIN=1, FUN=function(z){mean_outcome(z[2],z[1])})
dat$y <- rnorm(n=n, mean=dat$Ey, sd=0.5)
dat$trt <- as.factor(dat$trt)
ggplot(dat) + geom_line(aes(x=w, y=Ey, group=trt, col=trt)) +
  geom_point(aes(x=w, y=y, col=trt, group=trt)) +
  ylab("E[Y]") + theme_bw() + theme(panel.grid = element_blank())
```



We will consider 100 cutopoints between 0.1 and 0.9, $w_1, \ldots, w_100$. For each cutpoint, $j \in 1, \ldots, 100$, we estimate the difference in average treatment effect for individuals above and below the cutpoint:

$$\beta_j = (E[Y|trt = 1, W \geq w_j] - E[Y|trt = 0, W \geq w_j]) - (E[Y|trt = 1, W < w_j] - E[Y|trt = 0, W < w_j])$$

We can estimate β_j by fitting the parameters in the regression

$$Y = \beta_0 + \alpha_1 trt + \alpha_2 1_{W > w_j} + \beta_j trt 1_{W > w_j} + \epsilon$$

by least squares. Here we calculate $\hat{\beta}_j$ for each cutpoint:

```
n.cutpoints <- 100
cutpoints <- seq(0.1, 0.9, length.out=n.cutpoints)
#Indicator variables
ix <- sapply(cutpoints, FUN=function(thresh){as.numeric(dat$w >= thresh)})
#Run the linear regressions
stats <- apply(ix, MARGIN=2, FUN=function(ii){
      f <- lm(y~trt*ii, data=dat)
      summary(f)$coefficients[4, 1:3]
    })
stats <- data.frame(matrix(unlist(stats), byrow=TRUE, ncol=3))
names(stats) <- c("beta", "se", "tstat")
stats$cutpoint <- cutpoints
j <- order(abs(stats$tstat), decreasing=TRUE)
stats$rank <- match(1:nrow(stats), j)
head(stats)
```

```
##        beta        se    tstat  cutpoint rank
## 1 0.3857081 0.1694744 2.275907 0.1000000   58
## 2 0.4327279 0.1639602 2.639225 0.1080808   50
## 3 0.4194834 0.1570891 2.670354 0.1161616   47
## 4 0.3477664 0.1529919 2.273104 0.1242424   60
## 5 0.3417814 0.1501157 2.276786 0.1323232   57
## 6 0.3852470 0.1489406 2.586582 0.1404040   52
```

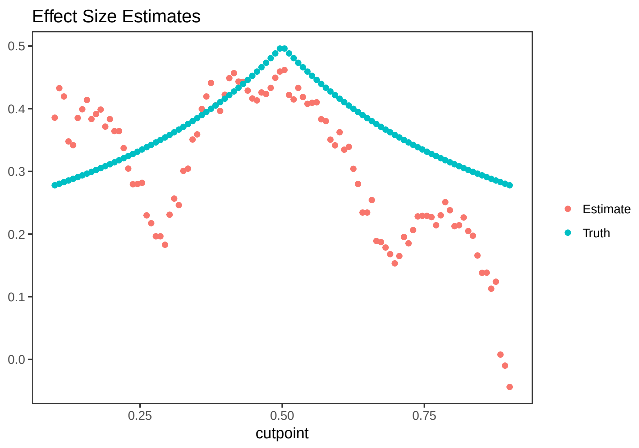
The true value of β_j is

$$0.5 * \frac{min(1 - w_j, 0.5)}{(1 - w_j)} - 0.5\frac{max(0, w_j - 0.5)}{w_j}$$

```
stats$truth <-sapply(cutpoints, FUN=function(wj){
  0.5*(min(1-wj, 0.5)/(1-wj)) - 0.5*max(0, wj-0.5)/wj})
```

Plotting effect size estimates and true parameter values vs. cutpoint:

```r
library(tidyr)
stats_long <- gather(stats[, c("cutpoint", "beta", "truth")], "type", "value", -cutpoint)
ggplot(stats_long) + geom_point(aes(x=cutpoint, y=value, group=type, color=type)) +
  scale_color_discrete(labels=c("Estimate", "Truth")) + ggtitle("Effect Size Estimates") +
  theme_bw() + theme(panel.grid=element_blank(), axis.title.y=element_blank(), legend.title = element_bl
```



Because we use the same individuals to estimate each parameter, the estimates are highly correlated.

Confidence Intervals

Non-Parametric Bootstrap

The non-parametric bootstrap is the most appropriate choice for these data since the estimates are correlated. Only the non-parametric bootstrap can model this correlation. To use the `nonpar_bs_ci` function in the `rcc` package to compute the nonparametric boostrap confidence intervals, we must supply an analysis function. We will use a data object that has trt, w, $E[Y]$, and Y as the first four columns and the indicator variables $1_{W>w_j}$ as the next 100 columns as input.

```r
mydata <- cbind(dat, ix)
analysis.func <- function(data){
  y <- data[,4]
  trt <- data[,1]
  stats <- apply(data[, 5:(n.cutpoints + 4)], MARGIN=2, FUN=function(ii){
      f <- lm(y~trt*ii)
      if(nrow(summary(f)$coefficients) < 4) return(rep(0, 3))
      summary(f)$coefficients[4, 1:3]
    })
  stats <- data.frame(matrix(unlist(stats), byrow=TRUE, ncol=3))
  names(stats) <- c("estimate", "se", "statistic")
  return(stats)
}
```

Here we calculate the non-parametrci bootstrap confidence intervals (Algorithm 3 in the paper):

```r
library(parallel)
library(rcc)
ci.nonpar <- nonpar_bs_ci(data=mydata, analysis.func=analysis.func, n.rep=500, parallel=TRUE)
head(ci.nonpar)
```

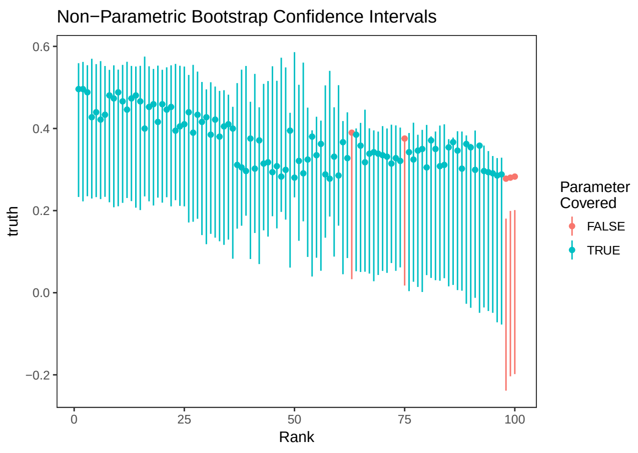
```
##         est statistic rank  ci.lower  ci.upper debiased.est
## 1 0.3857081  2.275907   58 0.1855063 0.5404221    0.3631941
## 2 0.4327279  2.639225   50 0.2322796 0.5859051    0.4103919
## 3 0.4194834  2.670354   47 0.1968628 0.5723828    0.3832292
## 4 0.3477664  2.273104   60 0.1633408 0.5054773    0.3265293
## 5 0.3417814  2.276786   57 0.1339574 0.5050506    0.3179952
## 6 0.3852470  2.586582   52 0.1739200 0.5604974    0.3584845
```

```
ci.nonpar <- ci.nonpar[, c("ci.lower", "ci.upper")]
sum(ci.nonpar[,1] <= stats$truth & stats$truth <= ci.nonpar[,2])/n.cutpoints
```

```
## [1] 0.95
```

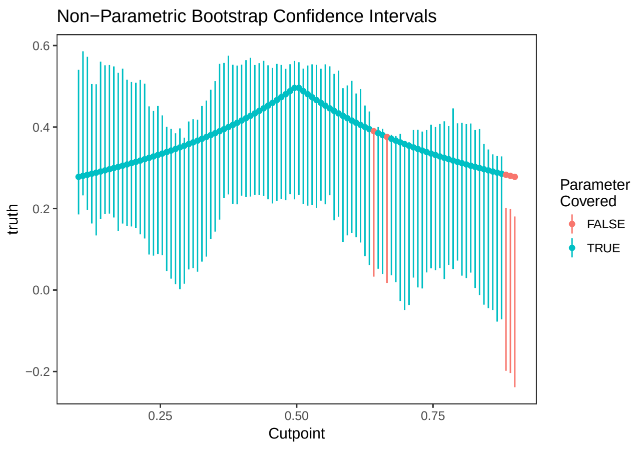
Here are the intervals plotted versus rank. Colored points indicate the true value of the parameter.

```
library(rccSims)
plot_cis(stats$rank, ci.nonpar, stats$truth, plot.truth=TRUE) +
  ggtitle("Non-Parametric Bootstrap Confidence Intervals")
```



Here they are plotted versus cutpoint:

```
plot_cis(cutpoints, ci.nonpar, stats$truth, plot.truth=TRUE) +
  ggtitle("Non-Parametric Bootstrap Confidence Intervals") + xlab("Cutpoint")
```



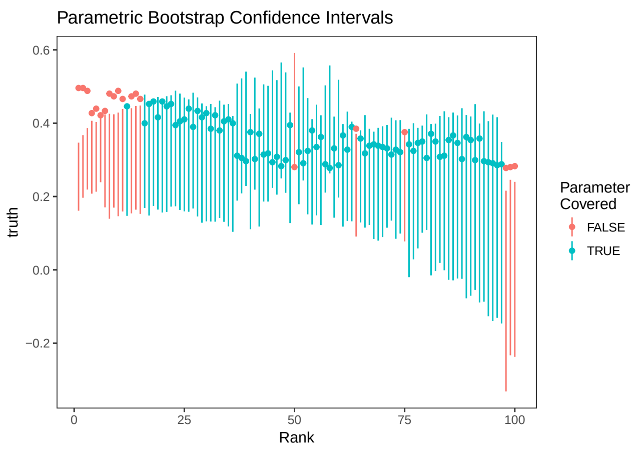
Parametric Bootstrap

We could instead use the parametric bootstrap (Algorithm 2 and Supplemental Algorithm 2):

```
ci.par <- rcc::par_bs_ci(beta=stats$beta, se=stats$se, n=500, use.abs=TRUE)[, c("ci.lower", "ci.upper")]
```

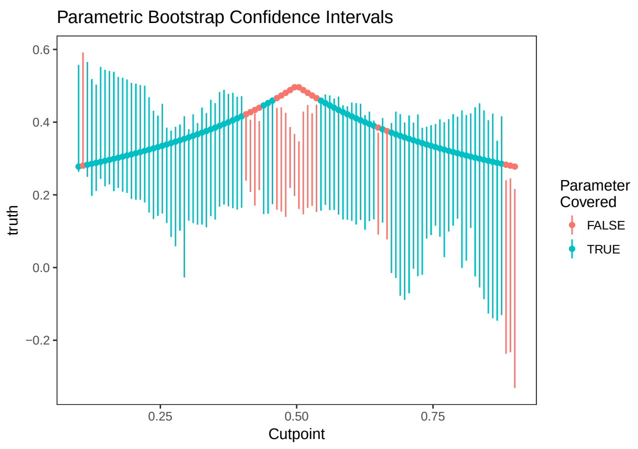
Here are the intervals plotted versus rank. Colored points indicate the true value of the parameter.

```
plot_cis(stats$rank, ci.par, stats$truth, plot.truth=TRUE) +
  ggtitle("Parametric Bootstrap Confidence Intervals")
```



Here they are plotted versus cut-point:

```
plot_cis(cutpoints, ci.par, stats$truth, plot.truth=TRUE) +
  ggtitle("Parametric Bootstrap Confidence Intervals") + xlab("Cutpoint")
```



The parametric bootstrap confidence intervals perform more poorly than the non-parametric confidence intervals because estimates are highly correlated but this isn't modeled in the basic parametric bootstrapping algorithm. These intervals tend to undercover parameters associated with the most and least significant estimates.

Marginal Confidence Intervals

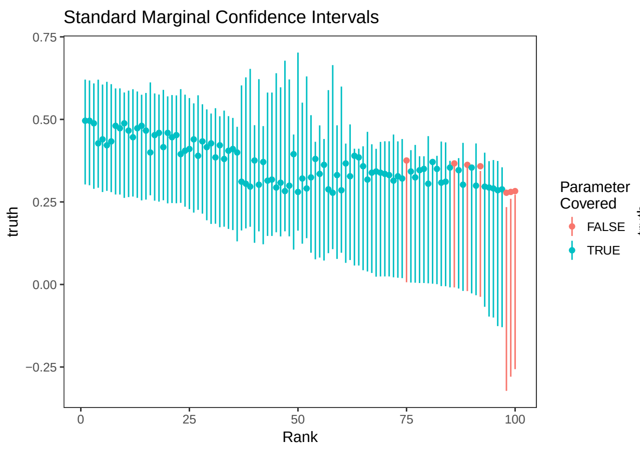
For comparison, let's look at the naive confidence intervals:

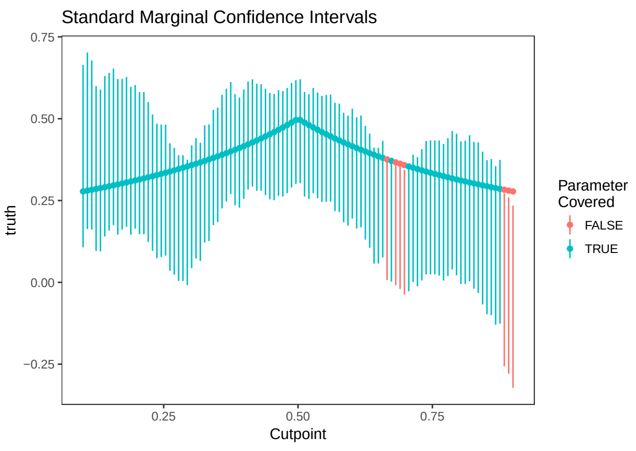
```
 ci.naive <- cbind(stats$beta-stats$se*qnorm(0.95), stats$beta + stats$se*qnorm(0.95))
mean(ci.naive[,1] <= stats$truth & stats$truth <= ci.naive[,2])
```

```
## [1] 0.93
```

```
plot_cis(stats$rank, ci.naive, stats$truth, plot.truth=TRUE) +
  ggtitle("Standard Marginal Confidence Intervals")
```

```
plot_cis(cutpoints, ci.naive, stats$truth, plot.truth=TRUE) +
  ggtitle("Standard Marginal Confidence Intervals") + xlab("Cutpoint")
```





These do pretty well since the estimates are so correlated.

Selection Adjusted Confidence Intervals

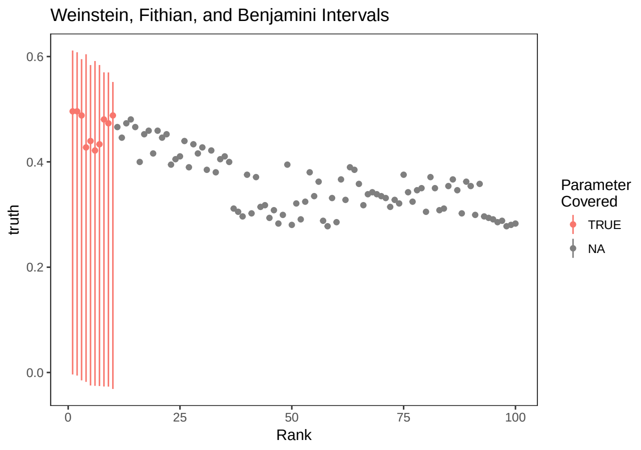
Here are the selection adjusted intervals of the intervals of Weinstein, Fithian, and Benjamini (2013) after selecting the top 10 (10%) parameters. o make running simulations easier, we have included the code distributed by Weinstein, Fithian, and Benjamini (2013) in the `rccSims` package. The `Shortest.CI` function used below is part of this code.

```
#We need to give this method the "cutpoint" or minimum value of the test statistic
ct <- abs(stats$tstat[stats$rank==11])
wfb <- lapply(stats$tstat, FUN=function(x){
              if(abs(x) < ct) return(c(NA, NA))
              ci <- try(rccSims::Shortest.CI(x, ct=ct, alpha=0.1), silent=TRUE)
              if(class(ci) == "try-error") return(c(NA, NA)) #Sometimes WFB code produces errors
              return(ci)
             })
ci.wfb <- matrix(unlist(wfb), byrow=TRUE, nrow=n.cutpoints)
ci.wfb[,1]<- ci.wfb[,1]*stats$se
ci.wfb[,2]<- ci.wfb[,2]*stats$se
mean(ci.wfb[,1] <= stats$truth & ci.wfb[,2]>= stats$truth, na.rm=TRUE)
```

```
## [1] 1
```

Here are the 10 WFB intervals which all cover their respective parameters

```
## Warning: Removed 90 rows containing missing values (geom_linerange).
```

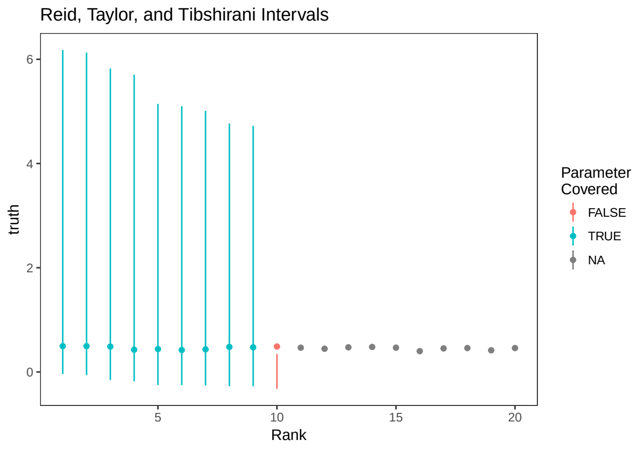


Here are the selection adjusted confidence intervals of Reid, Taylor, and Tibshirani (2014). These are implemented in the `selectiveInference` R package.

```
library(selectiveInference)
M <- manyMeans(y=stats$tstat, k=0.1*n.cutpoints, alpha=0.1, sigma=1)
ci.rtt <- matrix(nrow=n.cutpoints, ncol=2)
ci.rtt[M$selected.set, ] <- M$ci
mean(ci.rtt[,1] <= stats$truth & ci.rtt[,2]>= stats$truth, na.rm=TRUE)
```

```
## [1] 0.9
```

```
## Warning: Removed 10 rows containing missing values (geom_linerange).
```



This method gives the 10th ranked parameter an very short confidence interval!

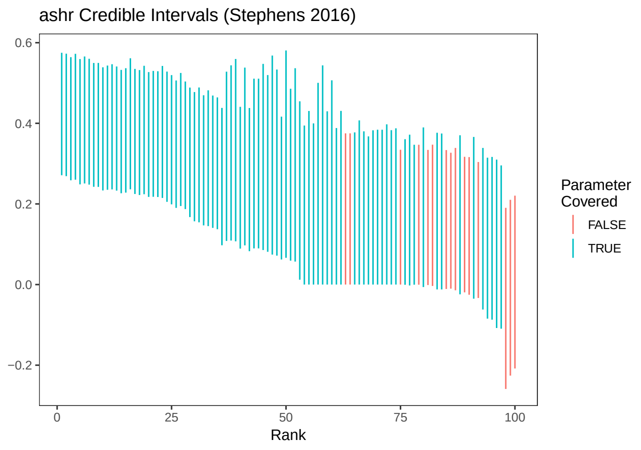
Empirical Bayes Confidence Intervals

Here are the credible intervals we get out of ashr ((**???**))

```
library(ashr)
ash.res <- ash(betahat = stats$beta, sebetahat = stats$se, mixcompdist = "normal")
ci.ash <- ashci(ash.res, level=0.9, betaindex = 1:n.cutpoints, trace=FALSE)
mean(ci.ash[,1]<= stats$truth & ci.ash[,2] >= stats$truth)
```

```
## [1] 0.85
```

Here are the ashr intervals at all ranks



Replicate the results in the paper

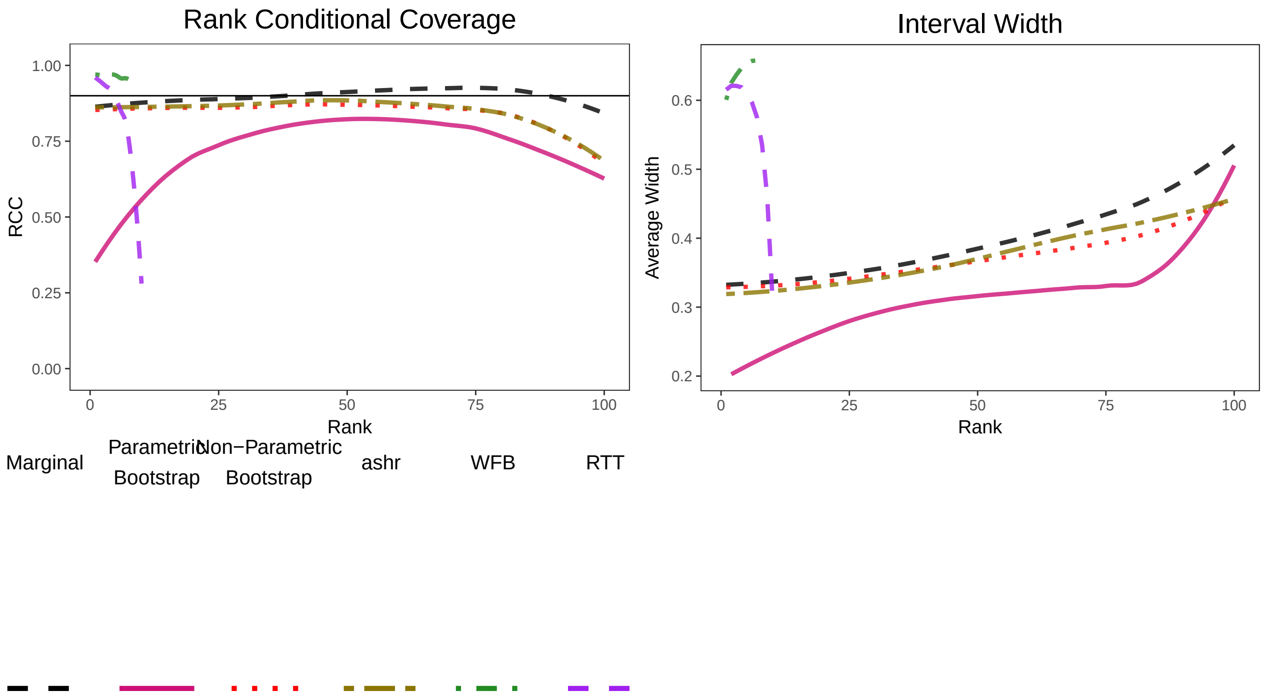
The steps above are all implemented by the `biomarker_sim` function in the `rccSims` package. This code will generate the results in package:

```
library(parallel)
biomarker_results <- biomarker_sim(n=400, n.rep = 400, n.cutpoints = 100, seed =1e7)
```

These results are also included as a built-in data set in the `rccSims` package.

This generates the plots shown in the paper:

```
data("biomarker_results", package="rccSims")
coverageplot <- rccSims::plot_coverage(biomarker_results, proportion=1,
      cols=c("black",  "deeppink3",  "red", "gold4", "forestgreen", "purple"),
      simnames=c("naive",  "par",    "nonpar", "ash", "wfb", "selInf1"),
      ltys= c(2, 1, 3, 6, 4, 2),
      span=0.5, main="Rank Conditional Coverage", y.range=c(-0.02, 1.02),
      legend.position = "none") + theme(plot.title=element_text(hjust=0.5))
widthplot <- rccSims::plot_width(biomarker_results, proportion=1,
      cols=c("black",  "deeppink3",  "red", "gold4", "forestgreen", "purple"),
      simnames=c("naive",  "par",    "nonpar", "ash", "wfb", "selInf1"),
      ltys= c(2, 1, 3, 6, 4, 2), span=0.5, main="Interval Width",
      legend.position = "none") + theme(plot.title=element_text(hjust=0.5))
legend <- rccSims::make_sim_legend(legend.names = c("Marginal", "Parametric\nBootstrap",
                                                 "Non-Parametric\nBootstrap", "ashr", "WFB", "RTT"),
            cols=c("black",  "deeppink3",  "red", "gold4", "forestgreen", "purple"),
            ltys= c(2, 1, 3, 6, 4, 2))
coverageplot
widthplot
legend$plot
```



Reid, Stephen, Jonathon Taylor, and Robert Tibshirani. 2014. "Post selection point and interval estimation of signal sizes in Gaussian samples." *ArXiv Preprint ArXiv:1405.3340*, no. 1 (May): 1–22. http://arxiv.org/abs/1405.3340.

Weinstein, Asaf, William Fithian, and Yoav Benjamini. 2013. "Selection Adjusted Confidence Intervals With More Power to Determine the Sign." *Journal of the American Statistical Association* 108 (501). Taylor & Francis Group: 165–76. doi:10.1080/01621459.2012.737740.