Comparison of confidence intervals in high dimensions

Jean Morrison

December 28, 2016

In this document we will compare several different approaches to building confidence intervals in high dimensions using the example in Section 1.5 of "Rank conditional coverage and confidence intervals in high dimensional problems" by Jean Morrison and Noah Simon. At the end of this document you will find the code that generates the results shown in the paper exactly. We will make use of the `rcc` package which accompanies the paper and can be found at github.com/jean997/rcc and the `rccSims` package which is at github.com/jean997/rccSims.

Generate data

For this exploration we will use setting 2 from section 1.5. We have 1000 parameters generated from a $N(0,1)$ distribution. Each of our observed statistics Z_i is generated as

$$Z_i \sim N(\theta_i, 1).$$

We will rank these statistics based on their absolute value.

```
library(ggplot2)
library(rcc)
library(rccSims)
set.seed(1e7)
theta <- rnorm(1000)
Z <- rnorm(n=1000, mean=theta)
j <- order(abs(Z), decreasing=TRUE)
rank <- match(1:1000, j)
```

Naive confidence intervals

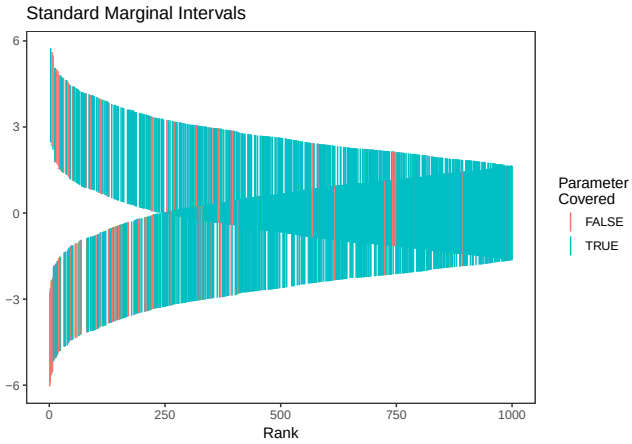
Now lets construct standard marginal confidence intervals for the parameters associated with each of these estimates. We will use $\alpha = 0.1$ to give an expected rate of coverage of 90%.

```
ci.naive <- cbind(Z - qnorm(0.95), Z + qnorm(0.95))
#Average coverage
sum(ci.naive[,1] <= theta & ci.naive[,2] >= theta)/1000
```

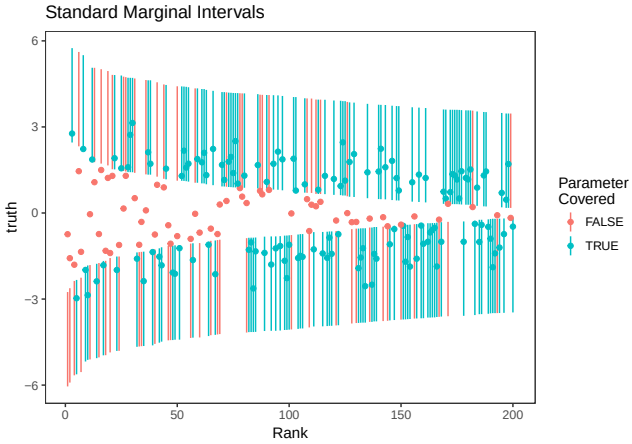
```
## [1] 0.894
```

We can plot these intervals and color them by whether or not they cover their target parameter. Notice that even though the overall coverage rate is close to the nominal level, a lot of the non-covering intervals are constructed for the most extreme statistics.

```
rccSims::plot_cis(rank, ci.naive, theta) + ggtitle("Standard Marginal Intervals")
```



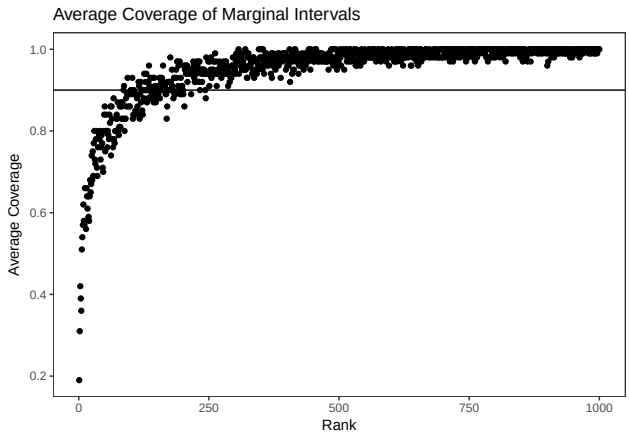
Here is the same plot for only the top 20% of statistics. In this plot, points show the value of the true parameter.



This is just one data set. To see what the expected coverage rates at each rank are we'll simulate 100 more data sets:

```
zsim <- replicate(n=100, expr={rnorm(n=1000, mean=theta)})
naive.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  j <- order(abs(zz), decreasing=TRUE)
  ci <- cbind(zz - qnorm(0.975), zz + qnorm(0.975))
  covered <- (ci[,1] <= theta & ci[,2] >= theta)[j]
  return(covered)
})
```

Plotting the average coverage at each rank:



Selection adjusted confidence intervals

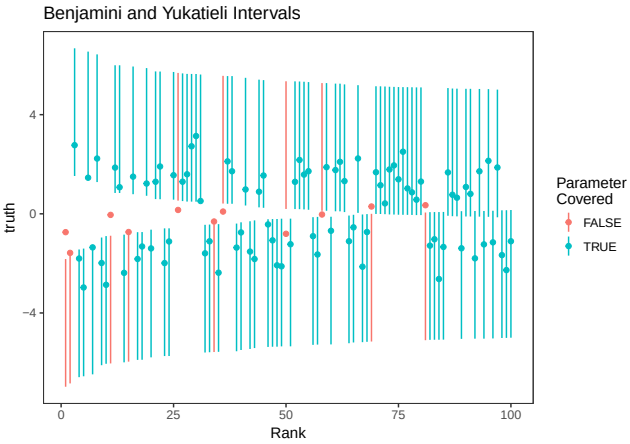
Now lets look at the intervals we get out of the methods proposed by Benjamini and Yekutieli (2005), Weinstein, Fithian, and Benjamini (2013), and Reid, Taylor, and Tibshirani (2014). These approaches all require us to make a selection before constructing intervals and we will get different intervals if we select a different subset of estimates. For this exploration, we select the top ten percent of estimates based on the absolute value ranking.

Benjamini and Yekutieli (2005) Intervals

Using this selection rule, the Benjamini and Yekutieli (2005) intervals are equivalent to the $1 - \frac{100*0.1}{1000} = 0.99$ coverage marginal intervals:

```
Z.sel <- Z[rank <= 100]
theta.sel <- theta[rank <= 100]
ci.by <- cbind(Z.sel - qnorm(0.995) , Z.sel + qnorm(0.995))
sum(ci.by[,1] <= theta.sel & ci.by[,2] >= theta.sel)/100
```

```
## [1] 0.89
```

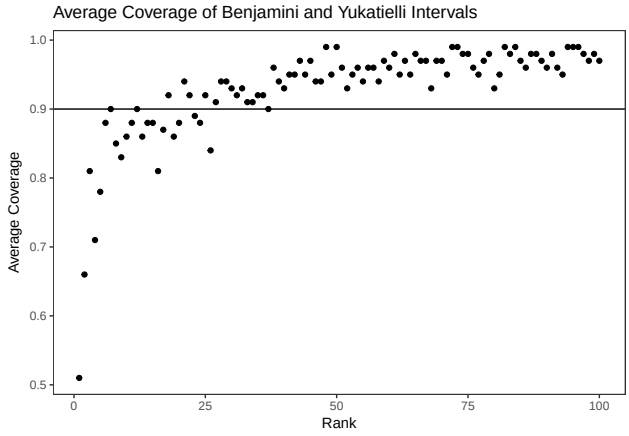


It can be hard to see patterns in just one data set so lets look at the average over 100 data sets:

```
by.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  j <- order(abs(zz), decreasing=TRUE)
  ci <- cbind(zz[j][1:100] - qnorm(0.995), zz[j][1:100] + qnorm(0.995))
```

```
  covered <- (ci[,1] <= theta[j][1:100] & ci[,2] >= theta[j][1:100])
  return(covered)
})
#The average coverage in each simulation is close to the nominal level
summary(colMeans(by.coverage))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8600  0.9100  0.9300  0.9282  0.9500  0.9800
```



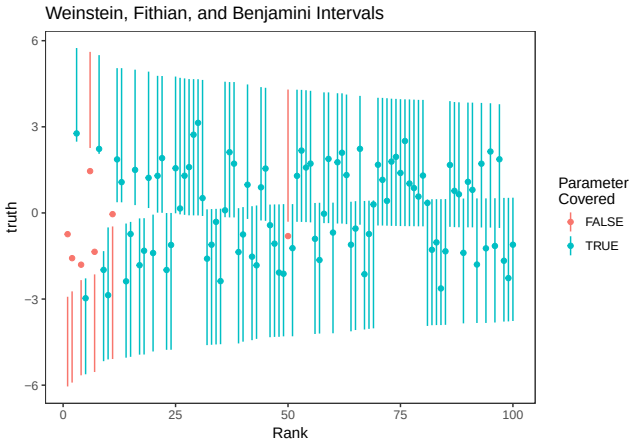
These intervals have reduced coverage for the top ranked estimates even though the average coverage in the selected set is correct.

Weinstein, Fithian, and Benjamini (2013) Intervals

For the Weinstein, Fithian, and Benjamini (2013) intervals, we use code that accompanied that paper. This code is included in the `rcc` package for convenience. These intervals are asymetric and narrower than the Benjamini and Yekutieli (2005) intervals but still control the average coverage in the selected set. To 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 Z
ct <- abs(Z[rank==101])
wfb <- lapply(Z, 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=1000)[rank <= 100,]
sum(ci.wfb[,1] <= theta[rank <=100] & ci.wfb[,2]>= theta[rank<=100])/100
```

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



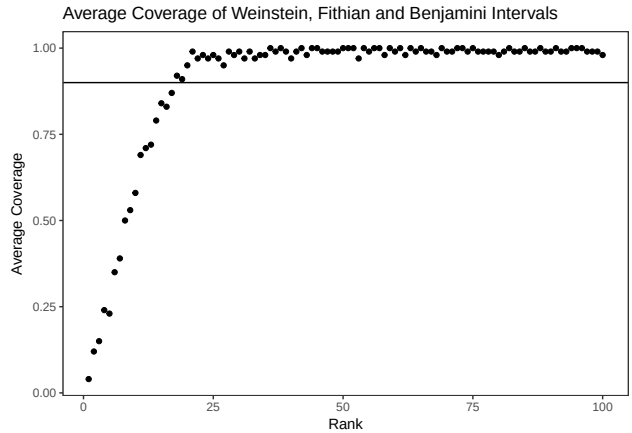
Most of the non-covering intervals are still for the parameters with the most significnt estimates. This pattern is clear when we look at the average coverage over 100 data sets:

```
system.time(wfb.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  j <- order(abs(zz), decreasing=TRUE)
  ct <- abs(zz[j][101])
  wfb <- lapply(zz[j][1:100], 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 <- matrix(unlist(wfb), byrow=TRUE, nrow=100)
  covered <- (ci[,1] <= theta[j][1:100] & ci[,2] >= theta[j][1:100])
  return(covered)
}))
```

```
##    user  system elapsed
##   5.176   0.000   5.176
```

```
#The average coverage in each simulation is close to the nominal level
summary(colMeans(wfb.coverage))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8400  0.8900  0.9100  0.9053  0.9200  0.9600
```

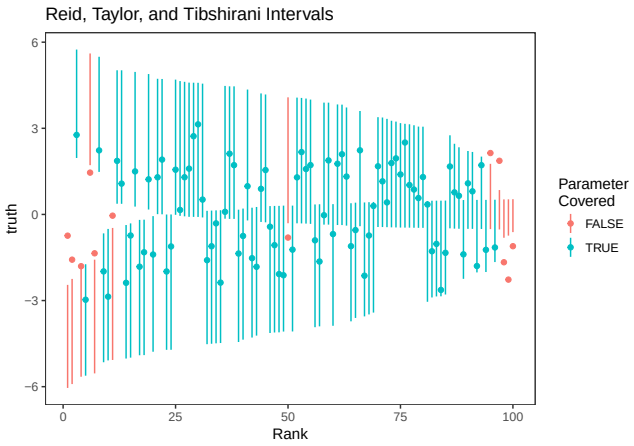


Reid, Taylor, and Tibshirani (2014) Intervals

Finally, lets look at the intervals of Reid, Taylor, and Tibshirani (2014). These are implemented in the `selectiveInference` R package

```
library(selectiveInference)
M <- manyMeans(y=Z, k=100, alpha=0.1, sigma=1)
ci.rtt <- matrix(nrow=1000, ncol=2)
ci.rtt[M$selected.set, ] <- M$ci
ci.rtt <- ci.rtt[rank <= 100,]
sum(ci.rtt[,1] <= theta[rank <=100] & ci.rtt[,2]>= theta[rank<=100])/100
```

```
## [1] 0.88
```



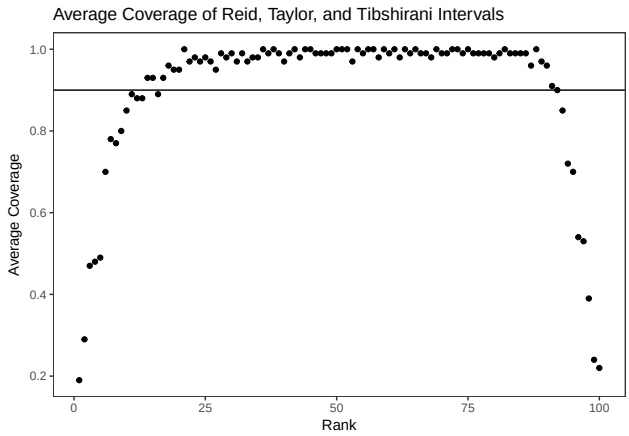
For the most part, these intervals are narrower than the Weinstein, Fithian, and Benjamini (2013) intervals. Non-covering intervals tend to be concentrated at the two extremes – parameters associated with the most and least significant estimates tend to go uncovered. We can see this by looking at the average over 100 simulations:

```
system.time(rtt.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  j <- order(abs(zz), decreasing=TRUE)
  M <- manyMeans(y=zz, k=100, alpha=0.1, sigma=1)
  ci <- matrix(nrow=1000, ncol=2)
  ci[M$selected.set, ] <- M$ci
  ci <- ci[j,][1:100,]
  covered <- (ci[,1] <= theta[j][1:100] & ci[,2] >= theta[j][1:100])
  return(covered)
}))
```

```
##    user  system elapsed
##   8.808   0.000   8.810
```

```
#The average coverage in each simulation is close to the nominal level
summary(colMeans(rtt.coverage))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8400  0.8800  0.9000  0.9017  0.9225  0.9600
```



Parametric Bootstrap

Now we will construct intervals for the same problem using the parametric bootstrap described in Section 2.3. Since we are ranking based on absolute value, we will use the variation given in Supplementary Algorithm 2. The procedure is implemented in the `par_bs_ci` function in the `rcc` package but here we go through the steps explicitly. First we estimate the average bias at each rank by bootstrapping:

```
set.seed(13421)
B <- replicate(n = 500, expr = {
    w <- rnorm(n=1000, mean=Z, sd=1)
    k <- order(abs(w), decreasing=TRUE)
    sign(w[k])*(w[k]-Z[k])
})
dim(B)
```

```
## [1] 1000  500
```

Next we calculate the 0.05 and 0.95 quantiles of the bias for each rank.

```
qs <- apply(B, MARGIN=1, FUN=function(x){quantile(x, probs=c(0.05, 0.95))})
```

Finally, we construct the intervals by pivoting

```
ci.boot <- cbind(Z[j]-qs[2,], Z[j]-qs[1,])
which.neg <- which(Z[j] < 0)
ci.boot[ which.neg , ] <- cbind(Z[j][which.neg] + qs[1,which.neg], Z[j][which.neg]+qs[2,which.neg])
#Get CI's in the same order as estimates
ci.boot <- ci.boot[rank,]
sum(ci.boot[,1] <= theta & ci.boot[,2] >= theta)/1000
```

```
## [1] 0.941
```

For comparison, here is how to get the same intervals using the `rcc` package.

```
set.seed(13421)
ci.boot2 <- rcc::par_bs_ci(beta=Z, n.rep=500)
head(ci.boot2)
```

```
##          beta se rank   ci.lower  ci.upper debiased.est
## 1  0.90690069  1  527 -0.8494054 1.9118492    0.5630546
## 2  0.02609555  1  989 -1.4182257 1.5079924    0.0142163
## 3 -1.05117530  1  455 -1.9420156 0.8520663   -0.6277091
## 4 -1.79176154  1  208 -2.5073640 0.2531810   -1.0707158
```

```
## 5 -0.42610396  1  780 -1.6345952 1.1359413   -0.2044789
## 6  0.51937390  1  729 -0.8277951 1.6480942    0.4228082
```

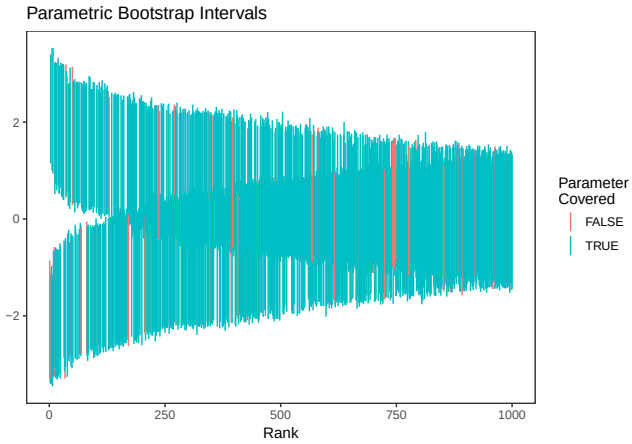
```
all.equal(ci.boot2$ci.lower, ci.boot[,1])
```

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

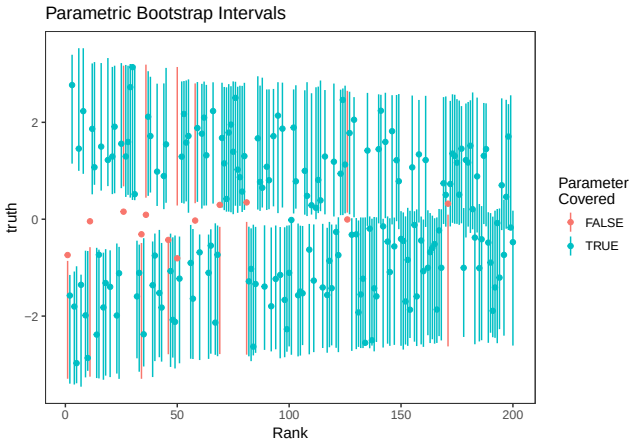
```
all.equal(ci.boot2$ci.upper, ci.boot[,2])
```

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

Here are the parametric bootstrap intervals for all ranks:



and for just the top 20% of ranks



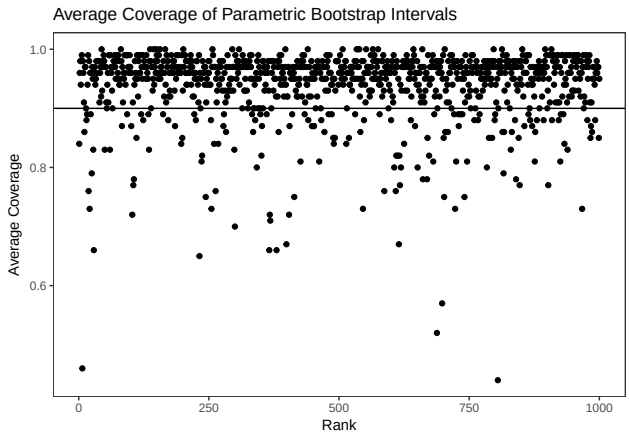
Looking at the average coverage over 100 data sets we see that the parametric bootstrap intervals are slightly conservative but that the average coverage for each rank is close to the nominal level.

```
system.time(boot.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  ci <- rcc::par_bs_ci(beta=zz)$ci
  covered <- (ci[,1] <= theta & ci[,2] >= theta)
  return(covered)
}))
```

```
##    user  system elapsed
##  32.904   0.072  32.976
```

```
summary(colMeans(boot.coverage))
```

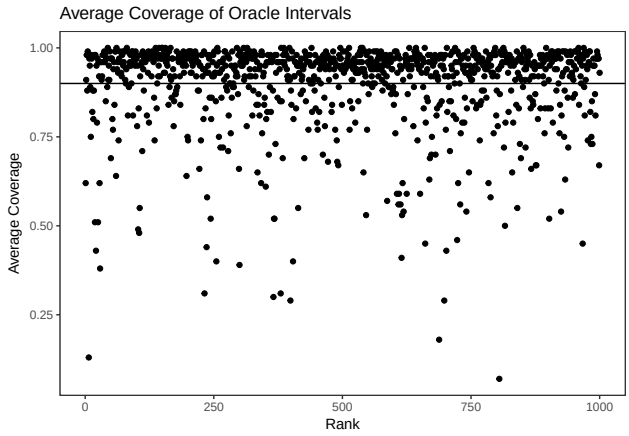
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9180  0.9330  0.9370  0.9377  0.9440  0.9540
```



The difference between the observed average coverage at each rank and the nominal level is a result of using Z_i as an estimate for θ_i in the bootstrapping step. If we knew θ_i we could generate the oracle bootstrap intervals which achieve exactly the right coverage at each rank (and are much smaller):

```
oracle.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  ci <- rcc::par_bs_ci(beta=zz, theta=theta)[, c("ci.lower", "ci.upper")]
  covered <- (ci[,1] <= theta & ci[,2] >= theta)
  return(covered)
})
summary(colMeans(oracle.coverage))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8780  0.8930  0.8990  0.8987  0.9042  0.9230
```



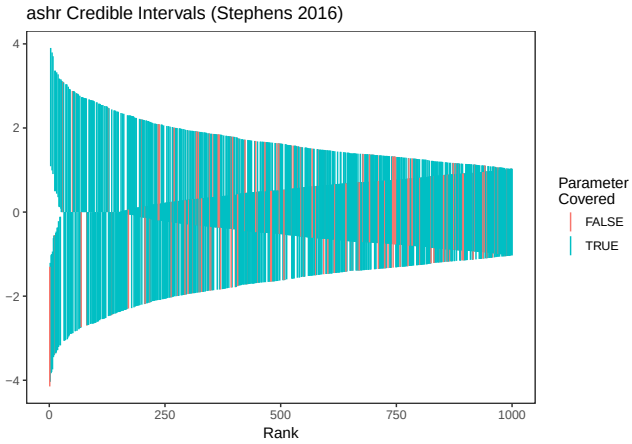
Empirical Bayes Credible Intervals

There are also empirical Bayes (EB) proposals for estimating praameters in high dimensional settings. Here we will look at the credible intervals generated using the method of Stephens (2016), which is implemented in the `ashr` package. This method assumes that θ_i are drawn from a unimodal distribution and that $Z_i \sim N(\theta_i, \sigma_i)$ where σ_i is known. These assumptions both hold in this case so `ashr` does well. Unlike the selection adjusted methods, `ashr` also has an RCC close to the nominal level at every rank.

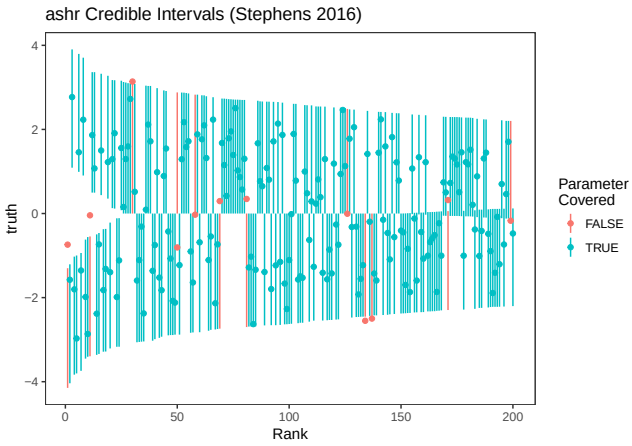
```
library(ashr)
ash.res <- ash(betahat = Z, sebetahat = rep(1, 1000), mixcompdist = "normal")
ci.ash <- ashci(ash.res, level=0.9, betaindex = 1:1000, trace=FALSE)
sum(ci.ash[,1]<= theta & ci.ash[,2] >= theta)/1000
```

```
## [1] 0.877
```

Here are the ashr intervals at all ranks



and for just the parameters with estimates in the top 20%



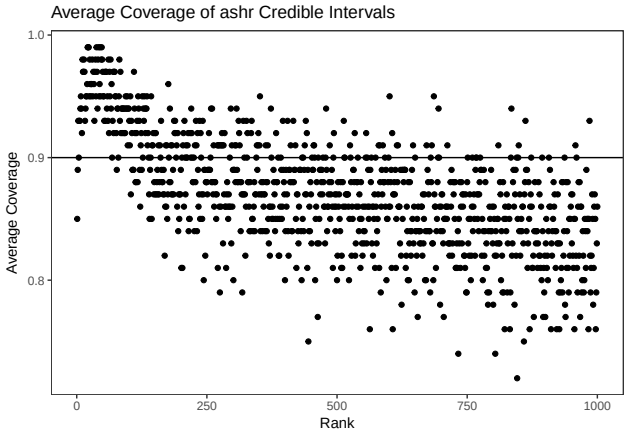
Here we can look at the average over 100 simulations. It is worth noting that I have conducted the simulations from more of a frequentist point of view — the parameters were fixed at the beginning and in each simulation we simply generate new statistics. If we wanted to really test the coverage of the credible intervals we should generate a new parameter vector each time as well. This might explain why we get overall slight undercoverage from the `ashr` intervals in this setting. `ashr` also takes substantially longer to run than the parametric bootstrap (on a normal laptop, the code below took nearly 7 minutes while the parametric bootstrap took only 30 seconds.) In this example, `ashr` does a good job controlling the RCC for the top parameters. This is, in part, because the true parameters are sparse. We found in the paper that when the parameters are not sparse, performance is much worse.

```
system.time(ash.coverage <- apply(zsim, MARGIN=2, FUN=function(zz){
  j <- order(abs(zz), decreasing = TRUE)
  ash.res <- ash(betahat = zz, sebetahat = rep(1, 1000), mixcompdist = "normal")
  ci <- ashci(ash.res, level=0.9, betaindex = 1:1000, trace=FALSE)
  covered <- (ci[,1] <= theta & ci[,2] >= theta)[j]
  return(covered)
}))
```

```
##    user  system elapsed
## 409.288   0.848 321.426
```

```
summary(colMeans(ash.coverage))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8220  0.8592  0.8740  0.8712  0.8840  0.9120
```



Generating simulation results in the paper

All of the interval construction methods described in the previous section (except for the Benjamini and Yekutieli (2005) intervals) are included in the `example_sim` function in the `rcc` package. In Section 1.5, we look at four sets of true parameters:

1. All the parameters are equal to zero
2. All the parameters were genreated from a $N(0, 1)$ distribution
3. 900 of the parameters are equal to 0 and 100 are equal to 3
4. 900 of the parameters are equal to 0 and 100 are drawn from a $N(0, 1)$ distribution

First we generate the four vectors of parameters

```
set.seed(14590424)
titles <- c("All Zero", "All N(0, 1)", "100 Effects=3", "100 Effects N(0, 1)")
example_params <- cbind(rep(0, 1000),
                        rnorm(n=1000),
                        rep(c(0, 3), c(900, 100)),
                        c(rep(0, 900), rnorm(100)))
titles <- c("All Zero", "All N(0, 1)", "100 Effects=3", "100 Effects N(0, 1)")
```

This matrix is also included as a builtin data set in the `rccSims` package. We generate simulation results using the `example_sim` function. This function just repeatedly executes the steps in the previous sections and records the coverage and width of the intervals.

```
set.seed(6587900)
sim.list <- list()
for(i in 1:4){
  sim.list[[i]] <- example_sim(example_params[,i], n=100, use.abs=TRUE)
}
```

These results are also included as a built-in data set to the `rccSims` package. Now we can make plots using the `plot_coverage` and `plot_width` functions in the `rccSims` package.

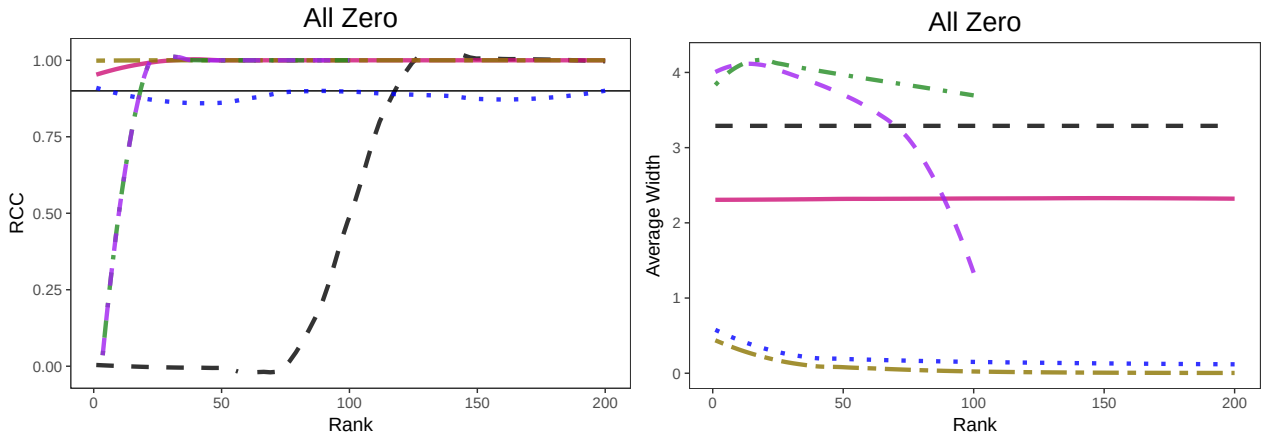
```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:intervals':
```

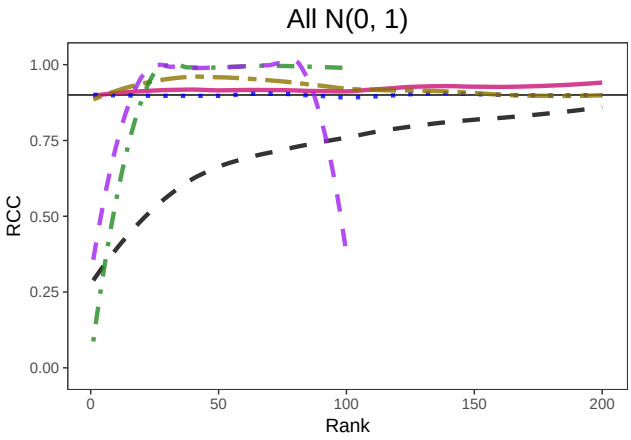
```
##
##     expand

## The following object is masked from 'package:Matrix':
##
##     expand
```

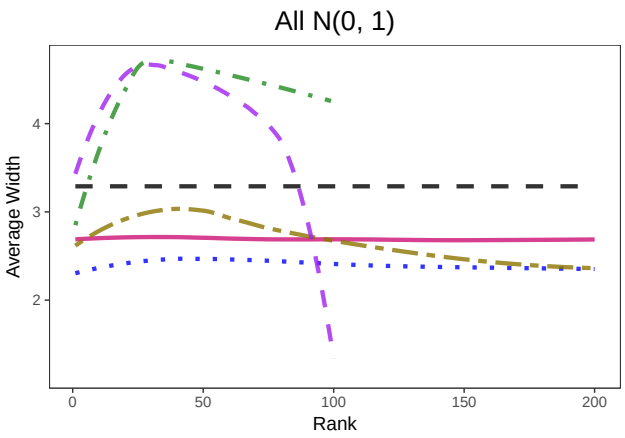
```
data("sim.list", package="rccSims")
covplots <- list()
widthplots <- list()
titles <- c("All Zero", "All N(0, 1)", "100 Effects=3", "100 Effects N(0, 1)")
for(i in 1:4){
  lp <- "none"
  covplots[[i]] <- plot_coverage(sim.list[[i]], proportion=0.2,
        cols=c("black",  "deeppink3",  "blue", "gold4", "forestgreen", "purple"),
        simnames=c("naive",  "par",    "oracle", "ash", "wfb", "selInf1"),
        ltys= c(2, 1, 3, 6, 4, 2), span=0.5, main=titles[i], y.range=c(-0.02, 1.02),
        legend.position = lp) + theme(plot.title=element_text(hjust=0.5))
  widthplots[[i]] <- plot_width(sim.list[[i]], proportion=0.2,
        cols=c("black",  "deeppink3",  "blue", "gold4", "forestgreen", "purple"),
        simnames=c("naive",  "par", "oracle", "ash", "wfb", "selInf1"),
        ltys= c(2, 1, 3, 6, 4, 2), span=0.5, main=titles[i],
        legend.position = lp)+ theme(plot.title=element_text(hjust=0.5))
}
legend <- rccSims:::make_sim_legend(legend.names = c("Marginal", "Parametric\nBootstrap",
                                                     "Oracle", "ashr", "WFB", "RTT"),
              cols=c("black",  "deeppink3",  "blue", "gold4", "forestgreen", "purple"),
              ltys= c(2, 1, 3, 6, 4, 2))
```

```
covplots[[1]]
widthplots[[1]]
```

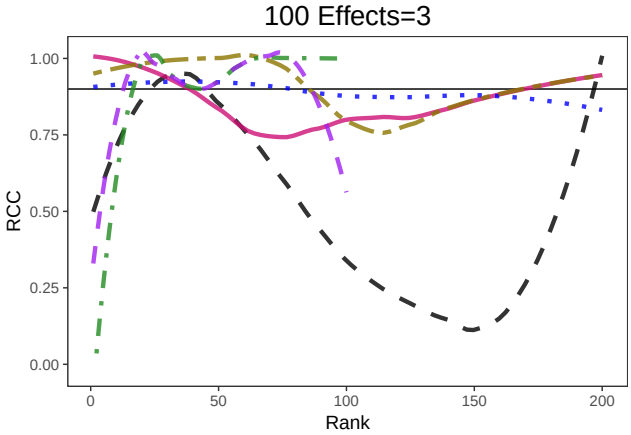


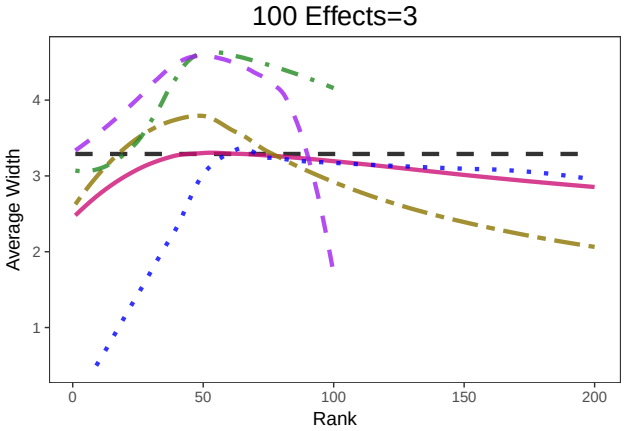

```
covplots[[2]]
widthplots[[2]]
```



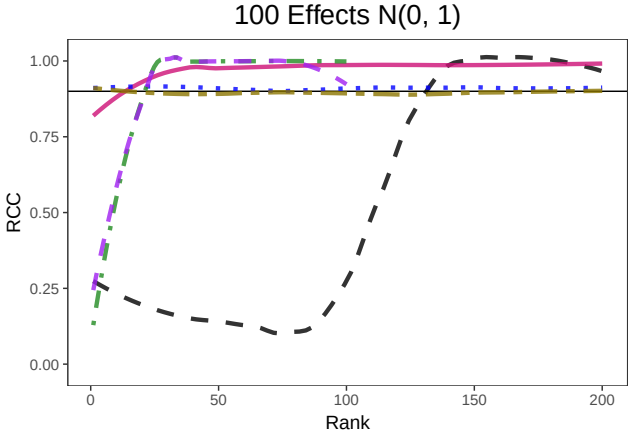


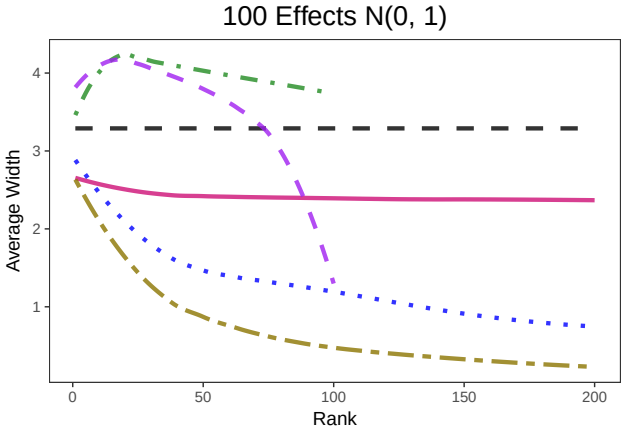

```
covplots[[3]]
widthplots[[3]]
```






```
covplots[[4]]
widthplots[[4]]
```





References

Benjamini, Yoav, and Daniel Yekutieli. 2005. "False Discovery Rate–Adjusted Multiple Confidence Intervals for Selected Parameters." *Journal of the American Statistical Association* 100 (469). Taylor & Francis: 71–81.

doi:10.1198/016214504000001907.

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*, May. http://arxiv.org/abs/1405.3340.

Stephens, Matthew. 2016. “False discovery rates: a new deal.” *Biostatistics*, October. doi:kxw041. doi:10.1093/biostatistics/kxw041.

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.