Problem Set 2, Question 4

Statistics 506, Fall 2017

Question 4

Design a Monte Carlo study to compare the coverage probabilities and width of the following two confidence intervals for the median:

i. the non-parametric bootstrap with 1,000 bootstrap samples;

ii. the robust estimator $\bar{\theta} \pm z_{\alpha/2}\bar{\sigma}/\sqrt{n}$ where $\bar{\theta}$ is the sample median and $\bar{\sigma}$ is 1.49 times the median absolute deviation.

Compare the estimators for a variety (4-6) of distributions and (3-5) sample sizes and report your results.

Solution

We will carry out the simulation for sample sizes 30, 100, and 1000 and compare the following distributions:

- Standard Normal $N(0, 1)$ (symmetric)
- Cauchy (symmetric, heavy-tailed)
- Chi-sq with 1 df (asymmetric)
- Beta(2, 2) (symmetric, bounded)
- Beta(.5, .5) (symmetric, bounded, bimodal).

Results

The results of the simulation are summarized in the figure and table below.

```
# libraries
library(tidyverse)

# load results
foo = load('compare_medians_16Oct2017.RData')

# order distributions by median width of BS interval
ord = {results %>%
  group_by(Name,type) %>%
  summarize(m=median(width)) %>%
  filter(type=='boot') %>%
  arrange(desc(m))}$Name

## Caption for figure
cap = 'Boxplots showing widths of 95% confidence intevals from 1,000 Monte
Carlo replications. Bootstrap intervals were based on 1,000 bootstrap samples.
The "robust" estimator appears to have smaller width on average.'

## Plot widths by sample size / distribution
results %>%
  mutate(type=ifelse(type=='boot','Bootstrap', 'Robust'),
         Name = factor(Name,ord)
         ) %>%
```

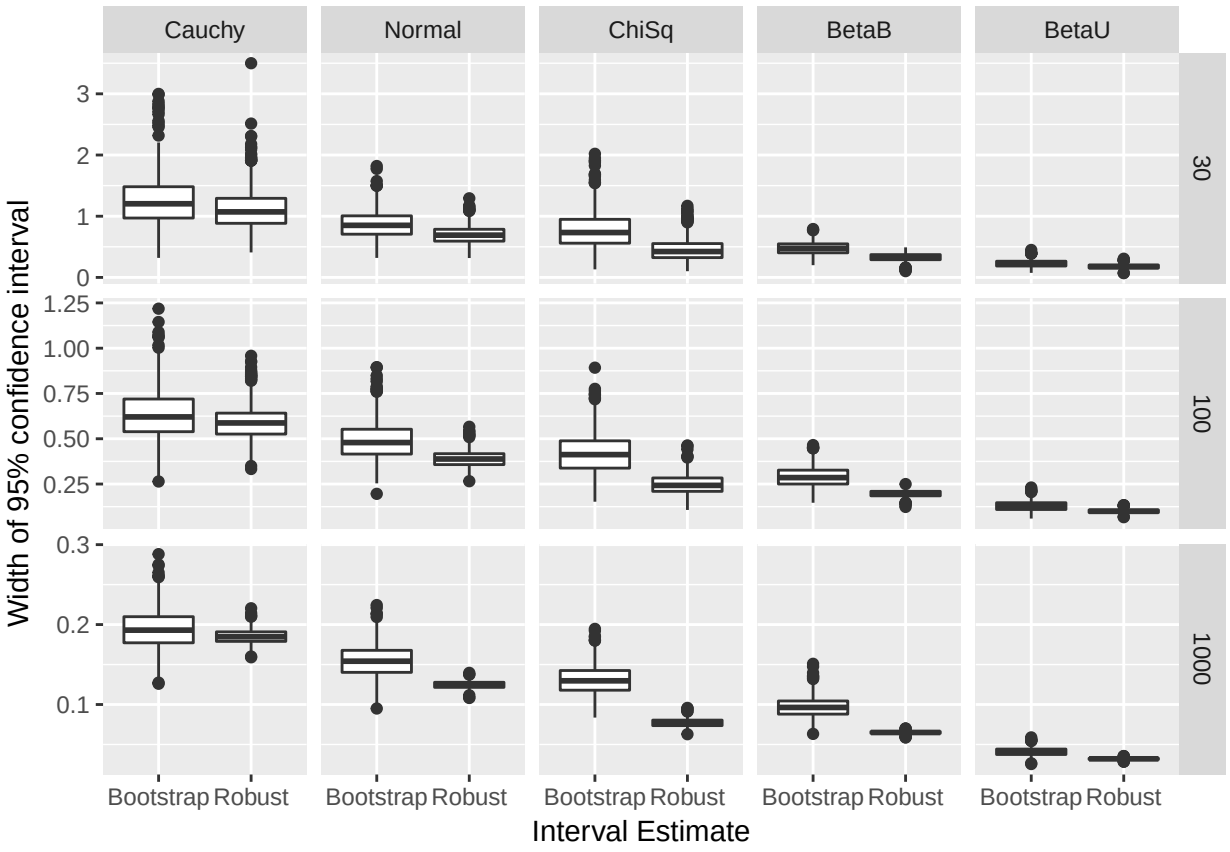

Figure 1: Boxplots showing widths of 95% confidence intevals from 1,000 Monte Carlo replications. Bootstrap intervals were based on 1,000 bootstrap samples. The "robust" estimator appears to have smaller width on average.

```
  ggplot(aes(x=type, y=width)) +
  geom_boxplot() +
  facet_grid(n~Name, scales='free_y') +
  xlab('Interval Estimate') +
  ylab('Width of 95% confidence interval') +
  scale_y_continuous()
```

```
# Compute estimate and CI from each set of Monte Carlo samples
cover_str = function(x){
  p = mean(x)
  n = length(x)
  moe = qnorm(.975)*sqrt(p*{1-p}/n)
  str = sprintf('%4.3f (%4.3f, %4.3f)',p,p-moe,p+moe)

  # italicize under coverage
  # bold conservative coverage
  if(p+moe < .95){
    str=paste0('_',str,'_')
  } else{
    if(p-moe > .95){
      str=paste0('__',str,'__')
```

```
    }
  }
  str
}
width_str = function(x){
  p = mean(x)
  n = length(x)
  moe = qnorm(.975)*sd(x)/sqrt(n)
  sprintf('%3.2f (%3.2f, %3.2f)',p,p-moe,p+moe)
}

## Construct a table
tab0 =
  results %>%
  mutate(type=ifelse(type=='boot','Bootstrap', 'Robust'),
         Name = factor(Name,ord)
         ) %>%
  group_by(Name, n, type) %>%
  summarize(Coverage=cover_str(cover), Width=width_str(width))

# reorganize
tab_cover = tab0 %>% select(-Width) %>%
  spread(type,Coverage) %>%
  ungroup

tab_width = tab0 %>% select(-Coverage) %>%
  spread(type,Width) %>%
  ungroup
```

From the table we see that the 'robust' estimator generally has less than 95% coverage.

```
tab_cover %>%
#  left_join(tab_width,by=c('Name','n')) %>%
  mutate(Name=ifelse(n > 30, "", as.character(Name))) %>%
  knitr::kable(col.names=c('Dist','n','BS Coverage', 'R Coverage'))
```

Dist	n	BS Coverage	R Coverage
Cauchy	30	0.942 (0.928, 0.956)	*0.925 (0.909, 0.941)*
	100	0.953 (0.940, 0.966)	*0.931 (0.915, 0.947)*
	1000	0.960 (0.948, 0.972)	0.949 (0.935, 0.963)
Normal	30	0.943 (0.929, 0.957)	*0.867 (0.846, 0.888)*
	100	0.945 (0.931, 0.959)	*0.881 (0.861, 0.901)*
	1000	0.950 (0.936, 0.964)	*0.883 (0.863, 0.903)*
ChiSq	30	0.938 (0.923, 0.953)	*0.750 (0.723, 0.777)*
	100	0.941 (0.926, 0.956)	*0.758 (0.731, 0.785)*
	1000	0.948 (0.934, 0.962)	*0.742 (0.715, 0.769)*
BetaB	30	0.956 (0.943, 0.969)	*0.759 (0.732, 0.786)*
	100	0.947 (0.933, 0.961)	*0.772 (0.746, 0.798)*
	1000	0.953 (0.940, 0.966)	*0.810 (0.786, 0.834)*
BetaU	30	0.936 (0.921, 0.951)	*0.853 (0.831, 0.875)*
	100	0.961 (0.949, 0.973)	*0.866 (0.845, 0.887)*
	1000	0.949 (0.935, 0.963)	*0.863 (0.842, 0.884)*

Script

The following script took approximately 5 minutes to run on mario using 8 cores when invoked using: `time R CMD BATCH --vanilla CompareMedians.R`.

```
file = 'CompareMedians.R'
cat(readChar(file,file.size(file)) )
```

```
## A script for comparing two methods for obtaining a 95% CI for
## the median: the bootstrap and a "robust" estimate.
##
## Author: James Henderson (xxx @ xxxx.edu)
## Date: October 16, 2017

## libraries
.libPaths('~/Rlib/')
library(dplyr); library(doParallel); library(doRNG)

## number of cores for parallelisim
ncores = 8

## Monte Carlo and boot strap replicates
mcrep = 1e3
nboot = 1e3

## random seed for reporoducibility
seed = 16+10*2017

## Sample sizes to compare
nvec = c(30, 100, 1000)

## Distributions to compare and their target medians
distr_list = list(
  Normal = list(rdist=rnorm, target=0, name='Normal'),
  Cauchy = list(rdist=function(x) rt(x, df=1), target=0, name='Cauchy'),
  ChiSq  = list(rdist=function(x) rchisq(x, df=1), target=qchisq(.5, df=1),
                name='ChiSq'),
  BetaU  = list(rdist=function(x) rbeta(x, 2, 2), target=.5, name='BetaU'),
  BetaB  = list(rdist=function(x) rbeta(x, .5, .5), target=.5, name='BetaB')
)

# Function to compute robust CI for a single data-set
robust_ci = function(x){
  m = median(x)
  mad = median(abs(x-m))
  m + c('lwr'=-1, 'upr'=1)*qnorm(.975)*1.49*mad / sqrt(n)
}

# Function to compute medians for all boot_strap samples in
# from single data set and extract the relevant quantiles.
boot_ci = function(boot_mat){
  quantile(apply(boot_mat, 2, median), c(.025, .975))
}

## Function to generate data and both sets of CIs
```

```
compare_medians = function(n, mcrep, rdist, target, nboot=1e3){
  # compare functions
  sim_data = rdist(n*mcrep)
  dim(sim_data) = c(n, mcrep)  # each column is a data set

  ## generate indices of bootstrap subsamples
  boot_ind = sample(1:n, nboot*n, replace=TRUE)
  dim(boot_ind) = c(n, nboot)

  ## construct boot strap samples
  boot_data = sim_data[boot_ind,]
  dim(boot_data) = c(n, nboot, mcrep)
  # Check that we've reshaped correctly
  #all.equal(boot_data[1:30,2,1],sim_data[boot_ind[,2],1])

  ## bootstrap intervals
  ci_boot = lapply(1:mcrep, function(i) boot_ci(boot_data[,,i]))
  ci_boot = do.call('rbind', ci_boot) ## this line won't run on Mario (why?)
  ci_boot = tibble(lwr=ci_boot[,1], upr=ci_boot[,2]) %>%
    mutate(width=upr-lwr, cover=upr>target & lwr<target, type='boot')

  ## robust intervals
  ci_robust = apply(sim_data,2,robust_ci)
  ci_robust = tibble(lwr=ci_robust[1,], upr=ci_robust[2,]) %>%
    mutate(width=upr-lwr, cover=upr>target & lwr<target, type='robust')

  bind_rows(ci_robust, ci_boot)
}

# Do the simulation in parallel
cl = makeCluster(ncores)
registerDoParallel(cl)
quiet = clusterEvalQ(cl,.libPaths('~/Rlib'))

# see 5.1 in: vignette("doRNG")
# for setting random seeds with nested loops
rng = RNGseq(length(distr_list)*length(nvec), seed)
p = length(nvec)

results = foreach(d=1:length(distr_list), .packages='dplyr', .combine='bind_rows') %:%
  foreach(n=iter(nvec), r=rng[{d-1}*p + 1:p]) %dopar% {

    # set seed
    rngtools::setRNG(r)

    # get distribution
    distr=distr_list[[d]]

    # return CIs
    compare_medians(n=n, mcrep=mcrep, rdist=distr$rdist, target=distr$target, nboot=nboot) %>%
      mutate(Name=distr$name, n=n)
  }

stopCluster(cl)
```

```
## Save the results
today = format.Date(Sys.time(), '%d%b%Y')
save(results, file=sprintf('./compare_medians_%s.RData',today))
```