

Exploring The Percentile Interval

Student Name

Instructions: In this activity we explore the performance of the percentile interval for inferring on the population standard deviation. You'll want to make sure to load the tidyverse before beginning in the setup chunk above.

- a. First run the chunk of code below to save the function for generating a percentile interval to your environment.

```
# function for calculating percentile interval
percentileFUN <- function(origsample, statistic, B, alpha, alternative){
  ## create matrix where each column is a bootstrap sample
  b <- matrix(sample(origsample, replace = TRUE, size = length(origsample)*B), ncol = B)
  ## calculate bootstrap sample statistics
  stats <- apply(b, 2, function(x) match.fun(statistic)(x))
  ## return appropriate interval
  if(alternative == "two.sided"){return(c(sort(stats)[(B+1)*(alpha/2)],
                                           sort(stats)[(B+1)*(1-(alpha/2))]))}
  if(alternative == "less"){return(c(-Inf, sort(stats)[(B+1)*(1-alpha)]))}
  if(alternative == "greater"){return(c(sort(stats)[(B+1)*(alpha)], Inf))}
}
```

- b. Test out the function using the chunk of code below and verify that you get an interval (-0.03690115, 0.34605626) using the same seed. Here we take a sample of size $N = 60$ from a Normal(0,1) population and construct two-sided 95% intervals for the population median using 999 bootstrap samples.

```
# test function out
set.seed(4567)
percentileFUN(origsample = rnorm(60), statistic = "median", B = 999, alpha = 0.05,
              alternative = "two.sided")
```

```
## [1] -0.03690115 0.34605626
```

- c. Construct a 90% two-sided percentile interval for the population standard deviation using the same sample as above and 999 bootstrap samples. Report the interval below.

```
# get percentile interval
set.seed(4567)
percentileFUN(origsample = rnorm(60), statistic = "sd", B = 999, alpha = 0.10,
              alternative = "two.sided")
```

```
## [1] 0.7513912 1.0106975
```

- d. What value should the interval contain? Does it contain this value?

The interval should contain the value 1 since the sample comes from a Normal distribution with mean 0 and standard deviation 1.

To check the assumptions of the percentile interval in a variety of situations, we can generate sampling distributions of the standard deviation from different underlying populations. Let's take the Chi-Square distributions (right-skewed) as our example. Don't worry if you don't understand all of the code below. You can run the entire chunk at once and move forward.

```
# degrees of freedom and sample size
k <- 1:9
n <- 20

# matrix to save results in - for each population, Chi-square(k),
# we will obtain 1000 sample standard deviations
sampsds <- matrix(0, nrow = 1000, ncol = length(k))

for(i in 1:length(k)){ # for each population (i.e. value of k)

  # generate 1000 samples of size n from a Chi-square(k) distribution
  sampmat <- matrix(rchisq(n*1000, k[i]), ncol = 1000)

  # calculate sample standard deviations
  sampsds[,i] <- apply(sampmat, 2, sd)
}

# save as dataframe for use in ggplot
sd_data <- as.data.frame(sampsds)

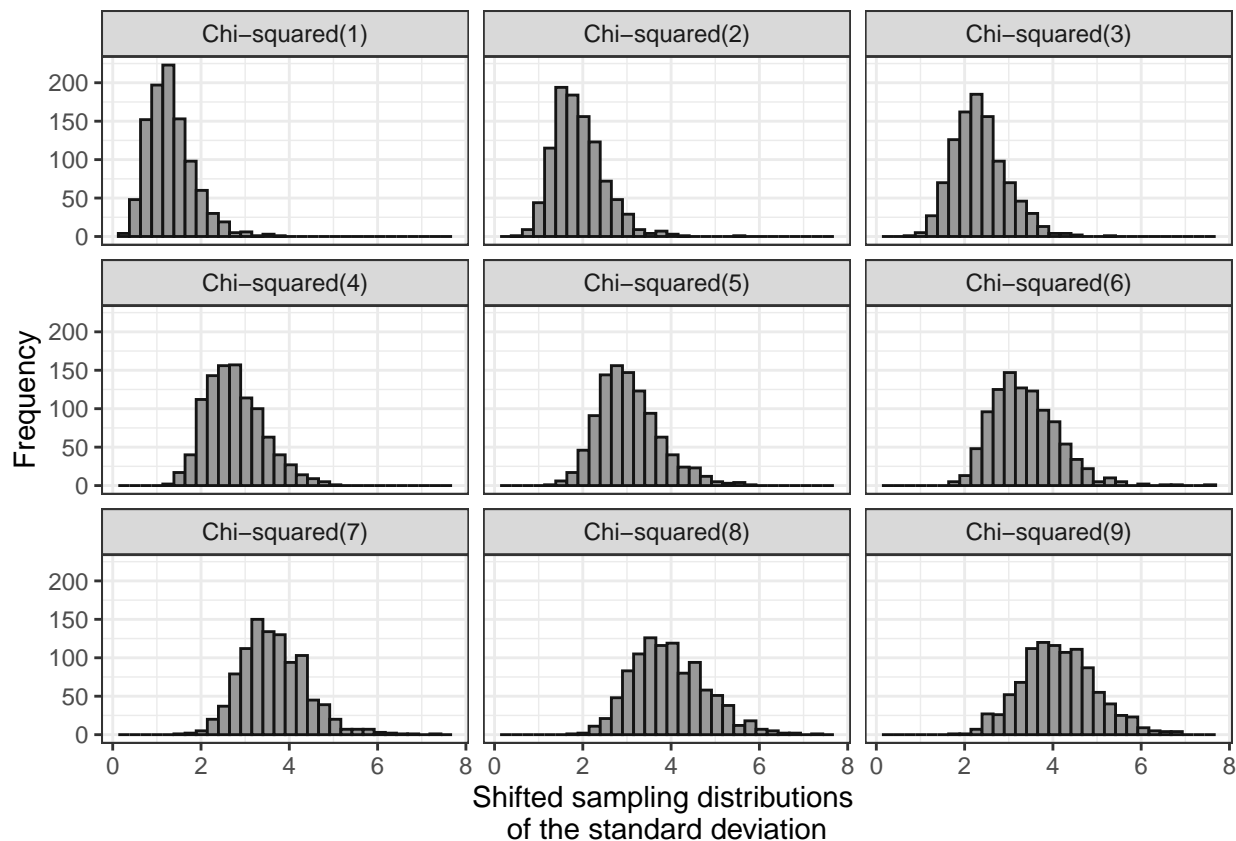
# change column names to specify population
colnames(sd_data) <- paste("Chi-squared(", k, ")", sep = "")

# gather column names and values to create one factor variable giving population
# and one giving sample sd
sd_data <- gather(sd_data, key = "population", value = "sd")
```

Now we can plot the distributions using the code below.

```
# plot distributions
ggplot(sd_data) +
  geom_histogram(aes(sd), fill = "gray60", color = "gray8") +
  facet_wrap(~population) + # facet by population
  theme_bw() +
  labs(x = "Shifted sampling distributions \nof the standard deviation", y = "Frequency")
```

```
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



- e. How would you describe the plot generated above? Does it seem that the assumptions of the percentile interval are met in any of these cases?

There is clear right skew when the degrees of freedom are smaller but as this parameter increases the distribution becomes more approximately normal.

We'll now explore how the percentile interval performs in the long-run if we apply it in one of these cases. Below we generate 1000 90% percentile intervals and calculate the proportion of them that contains the true parameter. Note that the standard deviation of a $\text{Chisquare}(k)$ distribution is $\sqrt{2k}$

```
# return to the Chi-squared(2) case and generate 1000 samples of size n
k <- 2
chisq2mat <- matrix(rchisq(n*1000, k), ncol = 1000)

# for each sample, calculate the percentile interval
percentile_ints <- t(apply(chisq2mat, 2, percentileFUN, statistic = "sd", B = 999,
                          alpha = 0.10, alternative = "two.sided"))

# what proportion of intervals contain the true parameter?
mean(percentile_ints[,1] <= sqrt(2*k) & percentile_ints[,2] >= sqrt(2*k))
```

```
## [1] 0.67
```

- f. What proportion of intervals should have contained the true parameter? What proportion actually did?

The proportion should be 90% but it was actually 64.8%.

- g. Independent Exploratory Simulations: Select 3 values for k and 3 increasing values for n . Using the code above calculate the coverage proportion for each combination of k and n (there should be 9 combinations). Discuss the results that you get and whether these were expected.

Results for this portion will vary but students should realize that the performance improves with the sample size and increasing k .