Brian Holliday

Professor Yuce

Computational Statistics

30 April 2020w

Project 2: Bootstrap Works

1.

Bootstrap is a statistical method in which we measure a statistic from a sample. We do this in order to give us a parameter of that statistic. Bootstrapping works as a Monte Carlo method by resampling a sample of population with replacement. Each resample gets us closer to the correct parameter, because you are resampling each sample from the population. Eventually from those resamples, you can get a confidence interval of your parameter. This method is useful if you want to get a parameter for statistics that lack a standard calculation, such as a R^2 value, and for when the statistical parameters come from data that is non-normal.

There are several different bootstrap methods. The main bootstrap method is the plain bootstrap method, which subtracts the boot mean from the sample mean for the parameter. There is the Normal Bootstrap method, which gives a normal approximation than estimates bias and mean square error by dividing by the sample standard deviation. There is the studentized bootstrap which has the studentized pivot. This pivot divides the difference in the boot mean and the sample mean by the sample standard deviation. The wild bootstrap which samples the population totally differently by multiplying the resampled variables by a random Poisson variable. All these methods work in creating a parameter of a statistic. We will prove it works by looking at the bias and mean square error of the sample mean of two populations across all the methods.

We sampled from a random chi-square distribution with a mean of 8. We then measured the frequency at which the difference between the boot mean and the sample less that the value of the quantiles. From there we can subtract the difference in the value percentage quantiles to get the bias. We did the same process for the exponential distribution with a mean of one. This process was done for the Plain, Normal, Studentized and Wild Bootstrap methods. The base code for the bootstrap method and the quantiles can be seen in Figures 7 and 8.

The bias tells us how far away we are from correct quantile amount. If we check the graphs for the bias in Figures 1 and 2, we can see that for both the chi-squared chart and the exponential the boot strap method gets very close to the right value. We can confirm this because our mean square error for both the exponential and chi-squared boot method becomes smaller than the bias. This can be seen in Figures 3 and 4. This works across all methods. We can refer to Figure 5 to see some descriptive statistics for the mean, median, and standard deviation of the bias across methods. All the values for each statistic are low across methods. We know bootstrap works because the bias and mean square error between the original sample and the boot sample is small, therefore we know it works.

2.

Figures:

Figure 1: Chi8 Bias

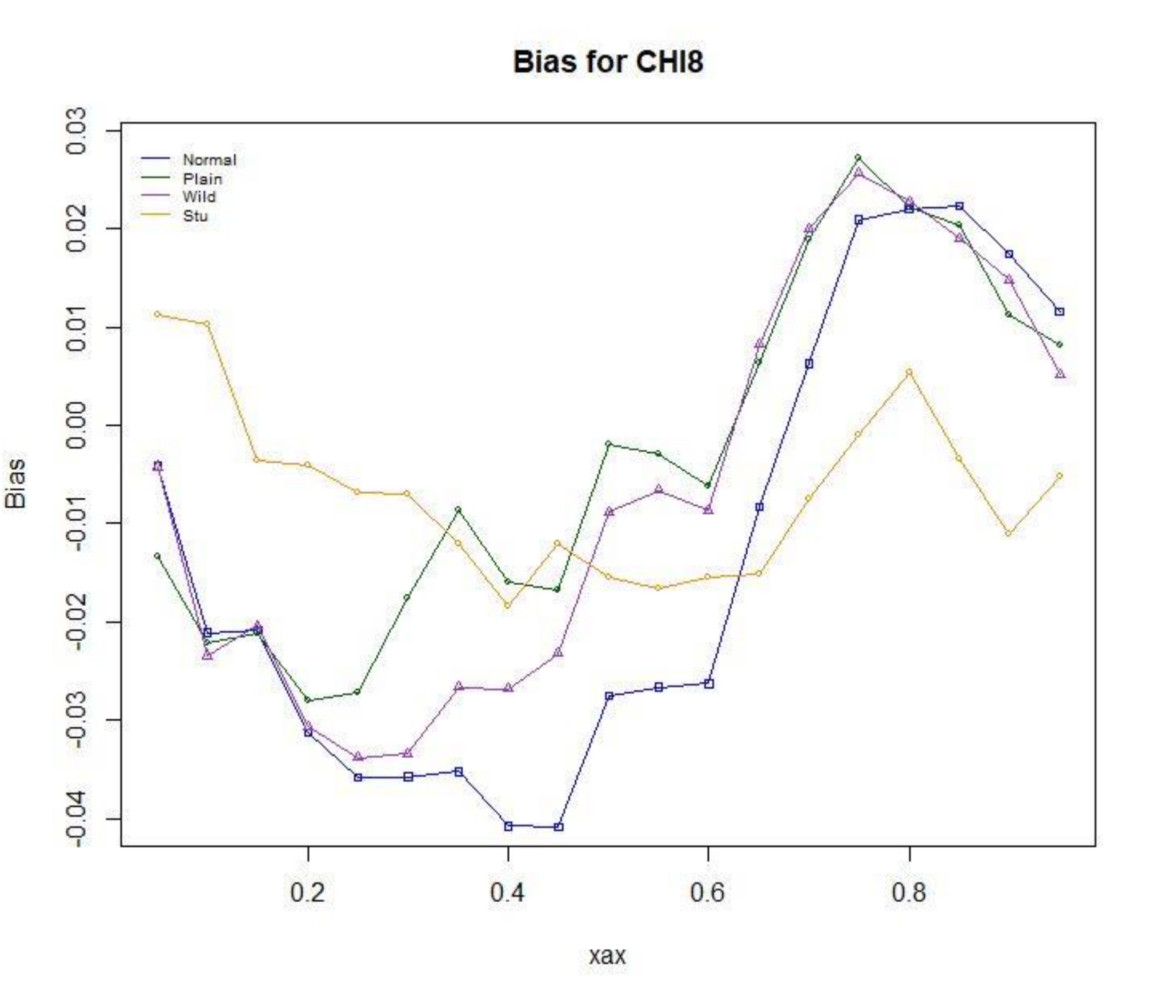


Figure 2: Exp1 Bias

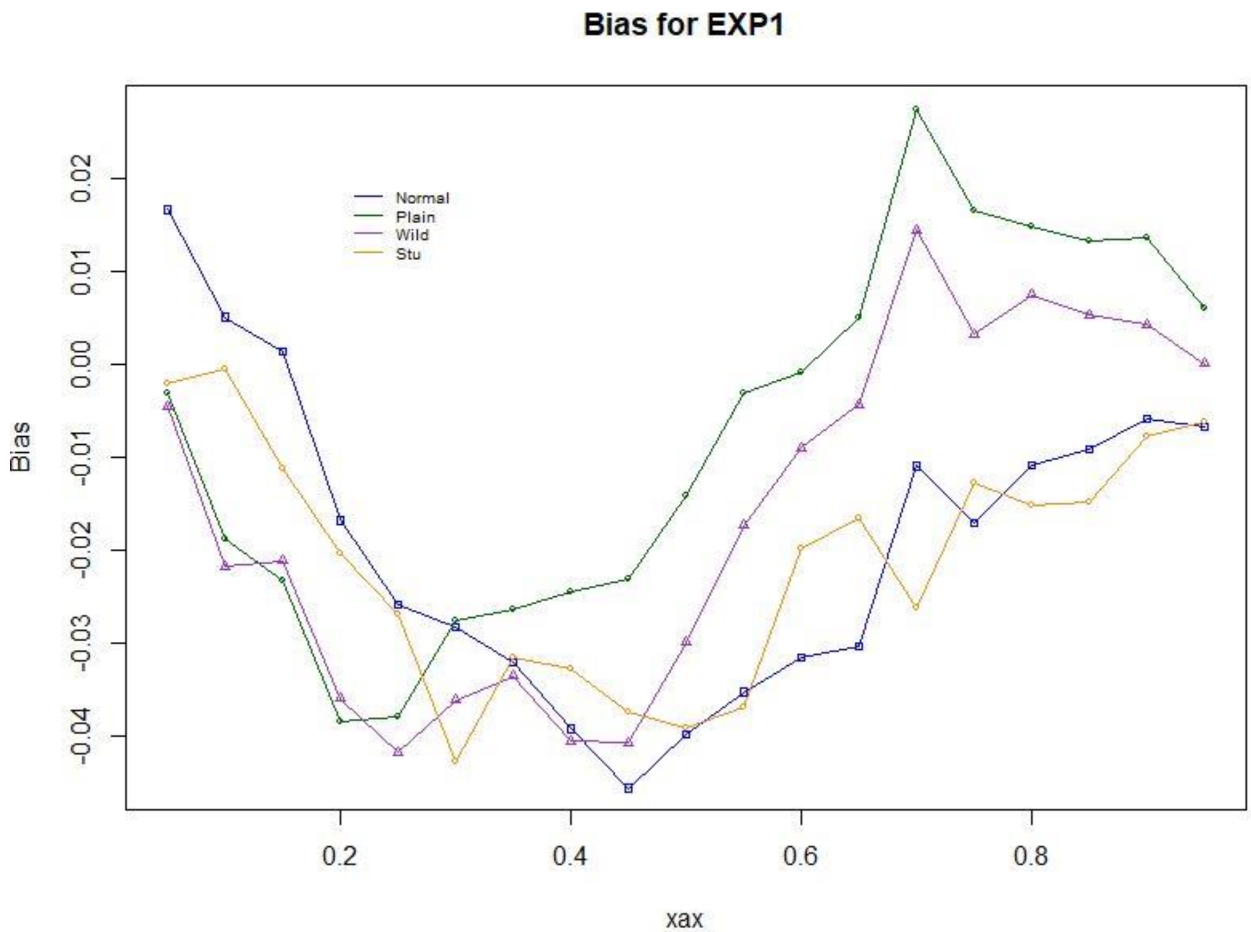


Figure 3: Chi8 MSE

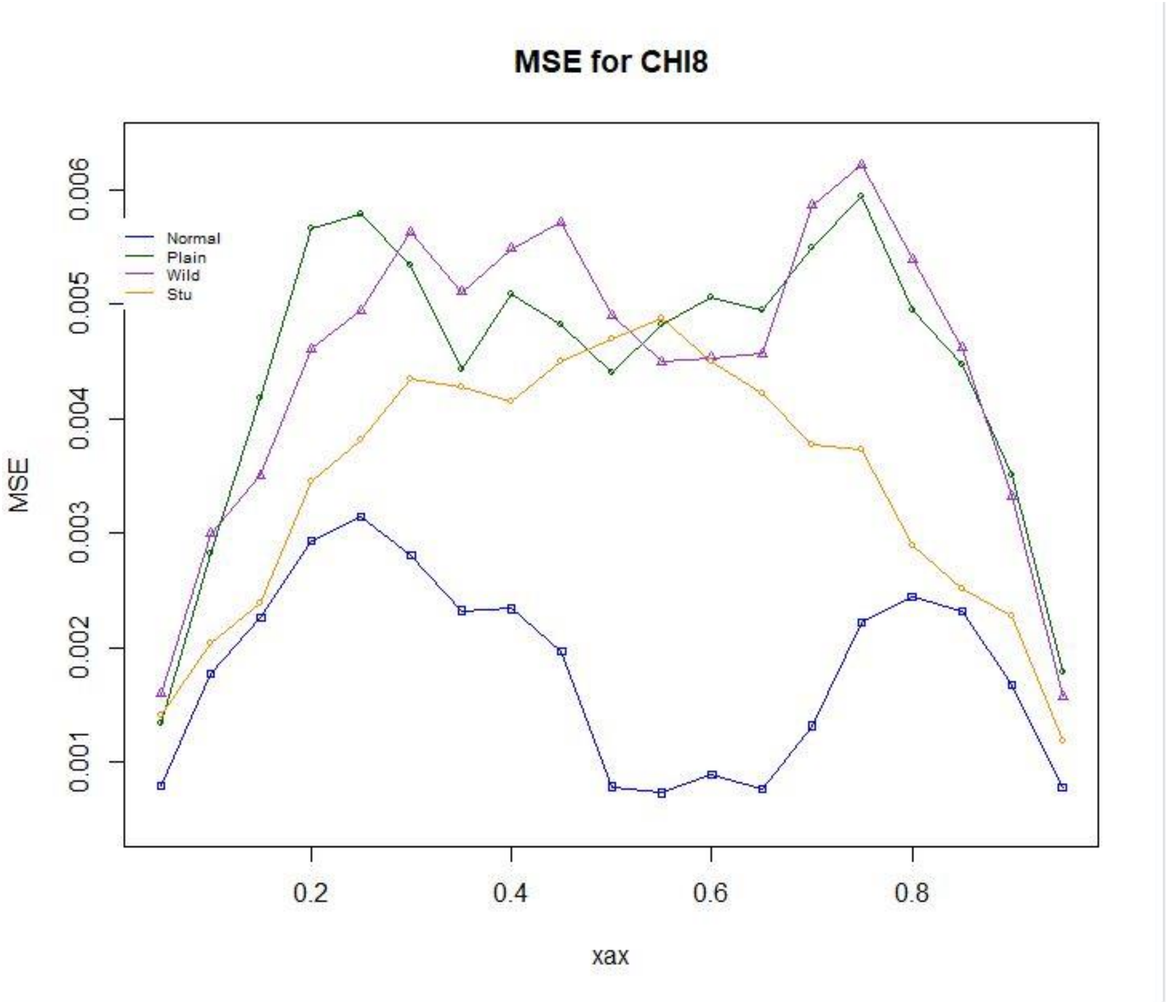
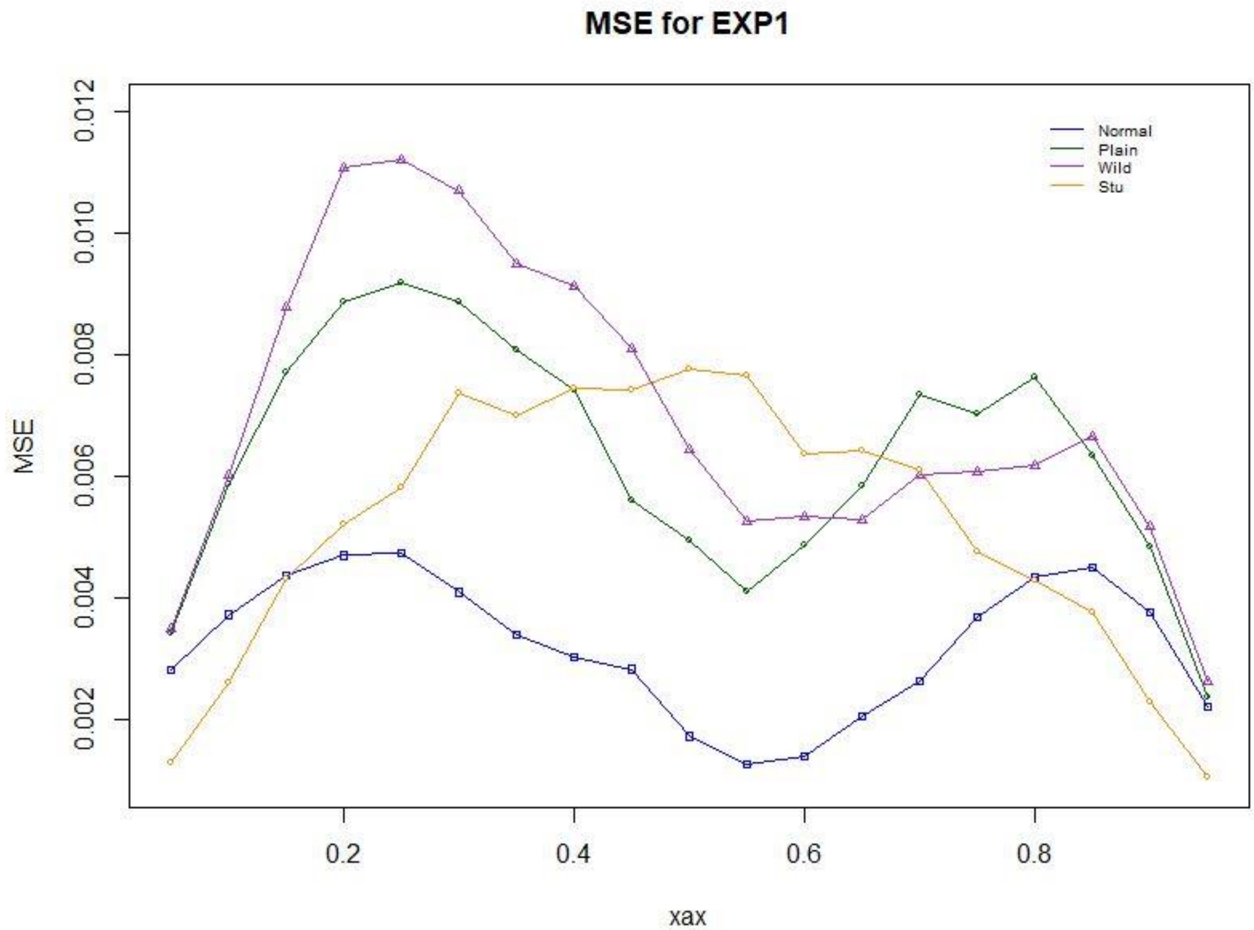


Figure 4: Exp1 MSE



3.

Figure 5: Chi8 Bias Descriptive Statistics

	method	mean	median	standard deviation
1	Plain BS Bias	-0.003568421	-0.00620000	0.017658899
2	Norm BS Bias	-0.013358959	-0.02109023	0.023234094
3	Stud BS Bias	-0.006715789	-0.00700000	0.008656229
4	Wild BS Bias	-0.006884211	-0.00860000	0.020661001

Figure 6: Exp1 Bias Descriptive Statistics

	method	mean	median	standard deviation
1	Plain BS Bias	-0.007663158	-0.0032000	0.01998333
2	Norm BS Bias	-0.019113311	-0.0171496	0.01702202

```
3  Stud BS Bias -0.021136842 -0.0198000         0.01304932
4  Wild BS Bias -0.015947368 -0.0174000         0.01904612
```

4.

Figure 7: Quantiles Code

```
### QUANTILES ###

q.exp1<-function(ns, nq)
{
  df <- double(nq)
  for(k in 1:nq) {
    s <- rexp(ns, 1)
    df[k] <- mean(s) - 1
  }
  return(quantile(df, 1:19/20))
}
```

Figure 8: EXP1 Boot Code

```
### EXP1 ###
#
bs.p.exp1<-function(ns, nr, nb, uu)
{
  s <- double(ns)        # S
  ss <- double(ns)       # S*
  pp <- matrix(double(nb * 19), nrow = nb)   # obs'd pk(S) across S's
  for(i in 1:nb) {
    s <- rexp(ns, 1)       # get S
    ms <- mean(s)              # M^
```

```
  fr <- rep(0, 19)          # place for frequencies, replicates 0 19-times.
  for(j in 1:nr) {
    ss <- sample(s, ns, rep = T)          # get S*
    dm <- mean(ss) - ms                   # get M* - M^
    fr <- fr + (dm < uu)                  # frequencies : WHY WORK?
  }
  pp[i,  ] <- fr/nr                       # pk(S), k = 1:19
}
err <- t(pp) - 1:19/20          # pk(S) - k/20, k = 1:19
bias <- apply(err, 1, mean)
mse <- apply(err^2, 1, mean)
cbind(bias, mse)
```