

## STAT230-SPRING 2023-Final Computational

Upload to Moodle: the codes for all problems in a *single* text file with comments, clearly specifying which code answers which question; the plots from the simulations; the typed write-up containing mathematical solutions and comments on the meaning of things. You can hand in to me a paper with the mathematical solutions if you do not want to type it.

### Problem 1

Consider the distribution with density

$$f(x) = \begin{cases} 2\nu x & 0 < x < 1 \\ (1 - \nu)\lambda/x^{\lambda+1} & x \geq 1 \end{cases} \quad (1)$$

where  $0 \leq \nu \leq 1$  and  $\lambda > 0$ .

- a) Compute the cdf and then employ it to obtain a sampler from the distribution.
- b) The meaning of the two parameters  $\nu, \lambda$ .  
Use the result in part a) to determine what  $\nu$  is.  
As for the meaning of  $\lambda$ , show that  $E(\log X | X > 1) = 1/\lambda$ .
- c) The method-of-moments estimator of  $\lambda$ ,  $\hat{\lambda}_{MM}$ , using a sample of size  $n$ , when  $\nu$  is known.  
Show or argue why there is no method-of-moments estimator if the true value of  $\lambda$  is smaller than or equal to 1. Then assume  $\lambda > 1$  and find  $\hat{\lambda}_{MM}$  using the first sample moment.
- d) Obtain the MLE estimator of  $\lambda$ ,  $\hat{\lambda}_{MLE}$ , using a sample of size  $n$  assuming that  $\nu$  is known.
- e) Comparison of the two estimators  $\hat{\lambda}_{MLE}$  and  $\hat{\lambda}_{MM}$ .

First address some questions mathematically.

Do the two estimators always exist for all possible values of the true parameter  $\lambda$ ? Do the estimators exist when  $\nu = 0$ ? Do they exist when  $\nu = 1$ ? Recall the meaning of  $\nu$  obtained in part b) and comment.

Then compare the two estimators using the MSE, or, in fact, a Monte Carlo estimate of the MSE, since it is difficult to compute the MSE analytically. Monte Carlo estimation is another name for estimation by simulations: drawing a large number of times samples from the true distribution and employing these samples to estimate what is of interest. [This is justified by the law of large numbers]. More precisely, consider the following 60 simulation schemes, corresponding to all possible triples  $(n, \nu, \lambda)$ , with  $\nu = 0, 0.25, 0.5, 0.75, 0.9$ ;  $\lambda = 1.2, 2, 3$ ; sample sizes  $n = 50, 100, 500, 1000$ . For each case defined by  $(n, \nu, \lambda)$ , using the sampler obtained in part a), draw from the distribution independently  $n$  times to get  $(x_1, \dots, x_n)$ ; evaluate the estimators  $\hat{\lambda}_{MLE}$  and  $\hat{\lambda}_{MM}$  on such sample; do so  $N$  times (say,  $N = 200000$ ) and then use the resulting  $N$  estimates of each estimator, to compute (the estimates of) its variance, its bias squared and its MSE. Namely, the MSE

$$R = E[(\hat{\lambda} - \lambda)^2]$$

is approximated as follows:

$$E[(\hat{\lambda} - \lambda)^2] \approx \frac{1}{N} \sum_{j=1}^N (\hat{\lambda}(\mathbf{x}_j) - \lambda)^2$$

where  $\mathbf{x}_j = (x_1, \dots, x_n)_j$  is a size- $n$  sample drawn from the distribution given in (1) and  $\hat{\lambda}(\mathbf{x}_j)$  is the estimate of  $\lambda$  using the formulae obtained in c) and d). Since the question asks to compute also bias squared  $(\lambda - E(\hat{\lambda}))^2$  and variance  $Var(\hat{\lambda})$ , compute these quantities first, using the  $N$  estimates  $\{\hat{\lambda}(\mathbf{x}_j)\}_{j=1}^N$ , and then use the MSE in the following form

$$R = E[(\hat{\lambda} - \lambda)^2] = (\lambda - E(\hat{\lambda}))^2 + Var(\hat{\lambda}).$$

after showing that this decomposition does hold. Plot the MSE, variance and bias “curves”. Comment on the results: is one estimator *always* better than the other? which one would you choose? do the estimators become more accurate when  $n$  increases? [Pay attention to the scale of the MSE values, if they are small you should phrase your statement with caution.]

## Problem 2

The attached file `p2data.txt` contains data  $\mathbf{d} = (x_1, \dots, x_{50})$  consisting of 50 values that were drawn independently from some distribution  $F$ . You do not know this generating distribution and I will not reveal it to you (or, not yet, at least).

The concern in this problem is the estimation of the true median  $\theta$  of the distribution. You will probably estimate  $\theta$  using the median of the sample,  $\hat{\theta}$ . Naturally, it is important to assess how accurate an estimate the sample median is of the true median and, for this, it is necessary to employ some measure of error. But how can you compute the standard error of  $\hat{\theta}$  or a confidence interval (CI) for the true median  $\theta$  if the distribution  $F$  of the data is not known to you and in this case not even the family to which it may belong? Here’s a method (known as the *bootstrap*) that allows you to do that. The method is quite general and works for any quantity  $\theta$  of interest and any estimate  $\hat{\theta}$  of it.

The method uses the one sample one has,  $\mathbf{d}$ , from the unknown  $F$  to find an approximation  $\hat{F}$  of  $F$ . Then  $\hat{F}$  is used as if it were the true  $F$  and Monte Carlo estimates of the standard deviation of  $\hat{\theta}$  (as in problem 1) and even CIs for  $\theta$  can be computed. Precisely, draw  $n$  values independently from  $\hat{F}$ , where  $n$  is the number of values in the data  $\mathbf{d}$  (the sample so obtained,  $\mathbf{d}^*$ , is called a bootstrap sample); compute the statistic of interest on this sample (here the median); repeat these two steps a large number of times  $B$ . The procedure therefore yields  $B$  replicates of the median:  $\hat{\theta}^{*1} = s(\mathbf{d}^{*1}), \dots, \hat{\theta}^{*B} = s(\mathbf{d}^{*B})$ , where  $s$  is the algorithm that outputs the median of the bootstrap sample. These replicates are then employed to get an measure of error for the statistic  $\hat{\theta}$  (here the median of the sample  $\mathbf{d}$ ), and a CI for  $\theta$  (the true median).

### What to do with the bootstrap replications?

The fundamental goal of the bootstrap is that of obtaining a standard error for  $\hat{\theta}$ . This is estimated as the sample standard deviation of the  $B$  replications,  $(\hat{\theta}^{*1}, \dots, \hat{\theta}^{*B})$ : *i.e.* the square root of the sample variance

$$\hat{Var}(\hat{\theta}) = \sum_{b=1}^B \frac{(\hat{\theta}^{*b} - \bar{\hat{\theta}})^2}{B} \quad (2)$$

with  $\bar{\hat{\theta}} = \sum_{b=1}^B \hat{\theta}^{*b} / B$  the sample mean of the  $B$  replicates. [You can divide by  $B - 1$  in place of  $B$  in the formula (2), if you prefer the usual sample variance].  $\sqrt{\hat{Var}(\hat{\theta})}$  is known as the bootstrap estimate  $\hat{\sigma}_{\hat{\theta}}^{boot}$  of the standard error of the statistic  $\hat{\theta}$ . It is just the Monte Carlo estimate of the the standard deviation

of  $\hat{\theta}$  by using not the true distribution  $F$ , which is unknown, but its available approximation  $\hat{F}$ .

The bootstrap replicates can also be used to compute the CI of the parameter of interest  $\theta$  of which the statistic  $\hat{\theta}$  is an estimator. There exist different versions of bootstrap CIs. We will consider two of them.

One is the bootstrap percentile interval which uses the sample percentiles of the bootstrap replicates. Namely, the  $100(1 - \alpha)\%$  bootstrap two-sided CI is

$$(\hat{\theta}_{\alpha/2}^*, \hat{\theta}_{1-\alpha/2}^*)$$

where  $\hat{\theta}_{\beta}^*$  is the  $\beta$  sample percentile of the bootstrap replicates: the value that separates the  $100\beta\%$  smaller values of the set  $(\hat{\theta}^{*1}, \dots, \hat{\theta}^{*B})$  from the rest.

Another version of the bootstrap CI uses the bootstrap estimate of the standard error:

$$\hat{\theta} \pm z_{\alpha/2} \hat{\sigma}_{\hat{\theta}}^{boot}.$$

This is known as the normal-based  $100(1 - \alpha)\%$  bootstrap CI since it uses the normal cutoff points  $z_{\alpha/2}$  [ $P(Z \geq z_{\alpha}) = \alpha$  where  $Z \sim \mathcal{N}(0, 1)$ ]. [Notice that  $\hat{\theta}$  is the statistic evaluated on the original data  $\mathbf{d} = (x_1, \dots, x_n)$  not on bootstrap samples.]

## How to approximate $F$ ?

There are two versions of the bootstrap method: they differ in how  $F$  is approximated.

### Version 1. Non-parametric bootstrap

This version of the method assumes nothing about  $F$ .

The estimate  $\hat{F}$  of  $F$  obtained from its one sample  $\mathbf{d} = (x_1, \dots, x_n)$  is the discrete distribution having masses of equal values at  $x_1, \dots, x_n$ . Mathematically the pmf of the approximate distribution is

$$P(X = x_i) = \frac{1}{n}, \quad i = 1, \dots, n.$$

This is known as the empirical distribution. This is tantamount to considering the values we have observed as the only possible values, and each equally likely (at least if the  $x_i$  are distinct).

In other words, a non-parametric bootstrap sample is obtained as follows. Draw *with replacement*  $n$  balls from an urn that contains the  $n$  balls  $(x_1, \dots, x_n)$  (each ball is labeled by the values of the data). The  $n$  values you draw are the bootstrap sample. Therefore in a bootstrap sample some values may appear more than once.

### Version 2. The parametric bootstrap

The parametric bootstrap is a different version of the method, which is applied when some information about the true distribution  $F$  of the data is known. Specifically it requires that  $F = F_{\lambda}$  be known up to some parameter(s)  $\lambda$ . The data  $\mathbf{d}$  are employed to estimate the unknown parameter(s)  $\lambda$ , the only missing information, and  $F$  is approximated by  $\hat{F} = F_{\hat{\lambda}}$ . For  $\hat{\lambda}$  one can use the MLE (or some other estimator). Since  $F_{\hat{\lambda}}$  is completely known, one can sample as many times as one wants and generate the bootstrap samples.

What changes in the two versions is how  $F$  is estimated. The procedure is otherwise the same. To sum up, draw  $B$  data sets of (generally and for this exercise) the same size as the original data set from the bootstrap approximation  $\hat{F}$  to  $F$ ; evaluate the statistic of interest on the  $B$  bootstrap samples to get  $B$  bootstrap replicates.

## Computations

- a) Using the provided data, employ the non-parametric bootstrap and the parametric bootstrap to compute: the standard error of sample median; both the normal and percentile 95% bootstrap CI for the true median. You may use  $B = 10000$ . Save the bootstrap replicates since they are needed for part b).

To apply the parametric bootstrap you need to know something about the true distribution. The data were generated from the distribution (1) of problem 1 with  $\nu = 0.2$ . Use this value of  $\nu$  and  $\lambda = \hat{\lambda}_{MLE}$ , computed from the attached data, and generate the (parametric) bootstrap samples using the sampler you have obtained in problem 1.

- b) Draw the two histograms of the bootstrap replications  $(\hat{\theta}^{*1}, \dots, \hat{\theta}^{*B})$  of  $\hat{\theta}$  (for the parametric and nonparametric bootstrap) obtained in a). These histograms are estimates of the sampling distribution of the sample median  $\hat{\theta}$ . Draw also the pdf of the true sampling distribution of  $\hat{\theta}$ . In fact, unless you want to have fun and obtain mathematically the true pdf of the sample median of  $n$  samples (good luck!), just approximate it via simulations. You know the family of the distribution of the data, that is, equation (1); the true value of  $\nu$ ,  $\nu = 0.2$ ; you only need to know the true value of  $\lambda$ . I will reveal it to you now:  $\lambda = 2$ . Now draw a sample of size  $n$  ( $n = 50$  in this problem) and compute its median. Repeat a very large number of times, say  $N = 10^6$ . The histograms of these values is the Monte Carlo approximation to the true sampling distribution of the random variable median of  $n$  samples from  $F$ .
- c) Which of the two methods (parametric and non parametric bootstrap) is performing better? Answer this question, using the output of your simulations in a) and in b). Namely: comparing the parametric bootstrap CIs and non-parametric bootstrap CIs with each other and with the true median  $\theta$  (which you can read off from some computation of problem 1 and using the true values of  $\nu = 0.2$  and  $\lambda = 2$ ); comparing the histograms of the two bootstrap replications with that of the estimate of the true sampling distribution.

If one method is performing better than the other, why do you think that method is better in this specific problem?

- d) Show that the probability that a given observation from the data  $\mathbf{d}$ , *e.g.*,  $x_i$ , will occur in a non parametric bootstrap sample  $\mathbf{d}^{*j}$ , is approximately  $1 - \exp(-1)$ , that is, around 0.632. [Not graded question]
- e) Pause and Ponder. [Yes, this is an exercise.]  
In the bootstrap method there are two approximations.

1. The approximation of  $F$  by  $\hat{F}$ .

What we want to compute is  $Var_F(\hat{\theta})$  (the variance of  $\hat{\theta}$  with respect to the true distribution  $F$ ). We can't compute it, thus we approximate it by  $Var_{\hat{F}}(\hat{\theta})$ .

2.  $Var_{\hat{F}}(\hat{\theta})$  can be difficult to compute if  $\hat{\theta}$  is a complicated function. Thus we use simulations:

$$Var_{\hat{F}}(\hat{\theta}) \approx \frac{1}{B} \sum_{i=1}^B \left( \hat{\theta}(\mathbf{d}^{*i}) - \frac{1}{B} \sum_{k=1}^B \hat{\theta}(\mathbf{d}^{*k}) \right)^2$$

where  $\mathbf{d}^{*i}$  is vector of values that are drawn independently from  $\hat{F}$ .

The first approximation is worse than the second: indeed while  $B$  can be chosen as large as one wants (you can draw from  $\hat{F}$  as many times as you want), the first approximation depends on the number  $n$  of samples we have, which can't be increased to improve the approximation.

### Sample median, percentiles, $z_\alpha$ in Python and Java

For your implementation, this may be useful.

Python

The numpy function `np.median(x)` computes the sample median of the values  $x$  (an array). The numpy function `np.percentile(x, q)` computes the sample  $q$ -th percentile point(s) of the values  $x$ , where  $q$  is a value (possibly an array of values) in  $[0, 100]$ . [Thus be careful that  $q = 0.01$  is not  $q = 1$ .] When  $q = 50$  you get the 50-th sample percentile, that is the median.

For  $z_\alpha$

```
from scipy.stats import norm
rv = norm(loc=0, scale=1)
zalpha=rv.ppf(1-alpha)
```

Java

With the library hipparchus and the module `stat`.

Median

```
import org.hipparchus.stat.descriptive.rank.Median;
Median.evaluate(double[] values, int begin, int length)
```

The  $q$ -th percentile point, where  $q \in [0, 100]$  (which you can also use for the median with  $q = 50$ ).

```
import org.hipparchus.stat.descriptive.rank.Percentile;
Percentile.evaluate(double[] values, double q);
```

[Be careful that  $q = 0.01$  does not give you the first percentile point, which is obtained with  $q = 1$ .]

For example, if  $x$  is the array of the (double) values

```
Median Med =new Median();
double med= Med.evaluate(x, 0, x.length); //the median
Median Per =new Percentile();
double q= Per.evaluate(x, 95); //95th percentile point
```

For  $z_\alpha$

```
import org.hipparchus.distribution.continuous.NormalDistribution;
NormalDistribution ndistro= new NormalDistribution (0.0, 1.0);
double zalpha=ndistro.inverseCumulativeProbability(1-alpha);
```

## Problem 3

The goal of this problem: study testing of hypotheses about the parameter of a distribution; understand the power function and the  $p$ -value via simulations.

We will consider a particular test, the two-sample  $t$ -test, but most of the conclusions you will reach and things you will discover are valid for all tests. The following is a mini-introduction to the two-sample  $t$  test, which we have not covered in class.

Two sets of data are collected:

$$(x_1, \dots, x_{n_x}) \quad \text{and} \quad (y_1, \dots, y_{n_y}).$$

We assume that each  $x_i$  comes from  $\mathcal{N}(\mu_X, \sigma^2)$  and each  $y_j$  from  $\mathcal{N}(\mu_Y, \sigma^2)$  and all are independent, and we want to test the difference of the true means  $\mu_X - \mu_Y$ . For example, we can ask ourselves if there is a difference between the true means of the two populations  $\mu_X = \mu_Y$ ; or if the second distribution tends toward larger values than the first  $\mu_X - \mu_Y \leq 0$ . In this problem, we will consider the following one-sided test:

$$H_0 : \mu_X - \mu_Y \leq 0 \quad \text{v.} \quad H_a : \mu_X - \mu_Y > 0$$

The obvious test statistic is  $\bar{X} - \bar{Y}$ , the difference of the sample means, which has distribution:

$$\bar{X} - \bar{Y} \sim \mathcal{N}\left(\mu_X - \mu_Y, \sigma^2 \left(\frac{1}{n_x} + \frac{1}{n_y}\right)\right)$$

First question: why is this the correct distribution?

We do not know  $\sigma^2$ . In its place we can use the so called pooled sample variance:

$$\hat{\sigma}^2 = \frac{(n_x - 1)S_X^2 + (n_y - 1)S_Y^2}{n_x + n_y - 2}$$

with  $S_X^2 = \sum_{i=1}^{n_x} (X_i - \bar{X})^2 / (n_x - 1)$  and  $S_Y^2 = \sum_{i=1}^{n_y} (Y_i - \bar{Y})^2 / (n_y - 1)$ . After standardizing, the statistic to test the hypotheses is:

$$T = \frac{\bar{X} - \bar{Y}}{\sqrt{\hat{\sigma}^2 \left(\frac{1}{n_x} + \frac{1}{n_y}\right)}}. \quad (3)$$

Under the assumptions of the data distributions, when  $\mu_X = \mu_Y$  and for any value of  $\sigma^2$ , the test statistic (3) has a  $t$  distribution with  $n_x + n_y - 2$  degrees of freedom:  $T \sim T_{n_x + n_y - 2}$ . This is the reason of the name two-sample  $t$ -test. [There are other versions of this test that do not assume that the true variances of  $X$  and  $Y$  are equal.]

- a) Obtain (mathematically) the rejection region in term of the test statistic  $T$  (3) at a significance level equal to  $\alpha$ .
- b) Computing the (Monte Carlo) estimate of the power function (as a function of  $\Delta = \mu_X - \mu_Y$ ).

Consider the test in a) and by simulation compute the power function. Use the same value of  $\sigma^2$ ,  $n_x = 20$ ,  $n_y = 25$ , and  $\alpha = 0.1$ . Choose 9 pairs  $(\mu_X, \mu_Y)$  so that  $\Delta$  covers both the null and alternative region of the space of the parameter  $\Delta$ . One pair must correspond to  $\Delta = 0$ . I suggest to use one value for  $\mu_Y$ , say, and vary  $\mu_X$  to get the different  $\Delta$  values. Do not choose negative values of  $\Delta$  that are too small otherwise the estimated probabilities will be too small. [How small is small depends on how you choose  $\sigma$  (since it is the statistic scale).] For each simulation setting, draw the data, then put your researcher hat on and reach a decision about the hypothesis, repeat these steps a large number of times, say,  $N = 200000$ , so as to obtain (an estimate of) the probability of rejecting  $H_0$  which is what the power function is, as  $\Delta$  varies.

Write a few lines explaining what you have done (without repeating what is written in the text above). Specifically, make clear which parameters you are choosing (or I have set up for you) as Nature, which ones as a researcher. Comment on the probability estimates you obtain. Do they make sense? Which values correspond to type I error probabilities? Plot the curve of the probability estimates versus  $\Delta$ .

- c) Write down the  $p$ -value for the test (as a general formula that can be fed to a computer) in terms of the statistic  $T$  (3).
- d) Using the formula in c), simulate the distribution of the  $p$ -value when the true parameter  $\Delta = \mu_X - \mu_Y$  is 0 and then for four other values of  $\Delta$ , two of which must be in the region of the parameter space corresponding to  $H_0$  and two in the region corresponding to  $H_a$ . Thus, for each value of  $\Delta$ , you need to simulate the data  $X, Y$  according to their true distributions and use them to compute the  $p$ -value applying the formula in c). You will need to use functions in python/java (see below) to compute the probability. [The simulation is needed for the distribution of the  $p$ -value, but not to compute a single  $p$ -value.] Draw then the histograms of the computed  $p$ -values. Comment on the distribution of the  $p$ -value for the chosen values of  $\Delta$ . Do they look as you expected them?

## Some useful functions in python and java

Python

The following about the  $t$ -distribution may be useful

```
from scipy.stats import t
rv = t(df=nu)      #define the rv  t_nu  #nu =degrees of freedom
rv.pdf(x)          #f(x)
rv.cdf(q)          #F(q)= P(X <=q)
rv.sf(q)           #P(X >= q)
rv.ppf(p)          #F-1(p)
```

For the variance, recall that to compute the usual (unbiased) estimator the degrees of freedom must be set to 1: `np.var(x, ddof=1)`.  
For how to sample from a normal, see the slides of part 4.

Java

With the library `hipparchus` and the module `stat`: mean and variance, and `core` for the normal and  $t$  distribution

If  $x$  is the array of the values (`double[]`).

```
import org.hipparchus.stat.descriptive.moment.Variance;
import org.hipparchus.stat.descriptive.moment.Mean;
Variance var =new Variance();
double s2= var.evaluate(x, 0, x.length); // sample var using all values of x
Mean mn= new Mean();
double mn=Mean.evaluate(x, 0, x.length) //sample mean
```

For the  $t$ -distribution

```
import org.hipparchus.distribution.continuous.TDistribution;
TDistribution td= new TDistribution(double degreesOfFreedom);
td.cumulativeProbability(double x); // P(X<=x) double
td.inverseCumulativeProbability(double p); // F-1(p) double
```

For how to sample from a normal, see the slides of part 4.