Exercise 4

Juliane Harnisch - jharnisch@uni-potsdam.de (mailto:jharnisch@uni-potsdam.de)

Part 1 (Generate samples)

In [1]:
```
import numpy as np
import random

np.random.seed(1)

mean = 170
variance = 16
standev = 4
# sample of size=10000 from normal distribution with mean=loc=170 and standard deviation=scale=4
# reference: https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
population = np.random.normal(loc=mean, scale=standev, size=10000)
```

Part 2 (Estimate mean, variance and standard deviation)

In [2]:
```
# 50 random samples taken from population
# reference: https://note.nkmk.me/en/python-random-choice-sample-choices/
sampleset = random.sample(list(population),50)


# computing the empirical mean as defined in the lecture
def empirical_mean(sample):
    n = len(sample)
    return(sum(sample)/n)

mu = empirical_mean(sampleset)


# computing the empirical variance as defined in the lecture
def empirical_variance(sample):
    n = len(sample)
    return(sum((sample-empirical_mean(sample))**2)/(n-1))

sigma2 = empirical_variance(sampleset)


# computing the ecmpirical standard deviation S_n (biased, as shown in Exercise 2)
def empirical_sd(sample):
    return(np.sqrt(empirical_variance(sample)))

sigma = empirical_sd(sampleset)

# printing the computed values against the real values
print('\t\t mean \t\t variance \t standard deviation \n empirical\t %.2f \t %.2f \t\t %.2f \n rea
```

```
                mean            variance        standard deviation
 empirical      168.59          12.62           3.55
 real           170             16              4
```

In [3]:
```
# computing the average estimations in 1000 samples of size 50
its = 1000
av_mu = 0
av_sigma2 = 0
av_sigma = 0

for i in range(its):
    sampleset = random.sample(list(population),50)
    av_mu += empirical_mean(sampleset)
    av_sigma2 += empirical_variance(sampleset)
    av_sigma += empirical_sd(sampleset)

av_mu = av_mu / its
av_sigma2 = av_sigma2 / its
av_sigma = av_sigma / its

print('average empirical mean: \t %.2f' %av_mu)
print('average empirical variance: \t %.2f' %av_sigma2)
print('average empirical stand. dev.: \t %.2f' %av_sigma)
```

```
average empirical mean:          170.03
average empirical variance:      16.18
average empirical stand. dev.:   4.00
```

In [4]:
```
# computing the ratios of average estimations to real values
print('average empirical mean/real mean: \t\t %.6f' %(av_mu/mean))
print('average empirical variance/real variance: \t %.6f' %(av_sigma2/variance))
print('average empirical stand. dev./real stand. dev.:  %.6f' %(av_sigma/standev))
```

```
average empirical mean/real mean:               1.000151
average empirical variance/real variance:       1.011265
average empirical stand. dev./real stand. dev.: 1.000236
```

We know from lecture and problem sheet 2 exercise 2, that the empirical mean $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and the empirical variance $S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X_n})^2$ are unbiased estimators, while the empirical standard deviation $S_n = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X_n})^2}$ is a biased estimator.

Computing these values for 50 random samples, the empirical values do not hit the acual values due to the small sample size. But if we compute the average estimations for 1000 different sets of 50 random samples, we get a better idea of how close the estimators get to the real values and therefore if they are unbiased or biased. With this computation, the estimations are much closer to the real values, but this is also true for the emp. standard deviation, that is biased. I think that this is because the real value of the standard deviation is a bit smaller than the values for the variance and mean, so I also considered the ratio between the average estimated values and the real ones. There we can see, that for all runs, the ratio of the standard deviation is the smallest.

Part 3 (Determine median and order statistic)

In [5]:
```
# ordering the samples
sampleset_ord = sorted(sampleset)

# computing the median
def med(sample):
    n = len(sample)
    sort = sorted(sample) # sorting the sample
    # definition from the lecture
    if n%2==0:   # for even sample length
        return(0.5*(sort[n//2-1]+sort[n//2])) # shifted with -1 since we start counting from 0
    else:        # for uneven sample length
        return(sort[n//2])

median = med(sampleset)
print("median: %.4f \n" %(median))

# 31th order statistic
print("31th order statistic: %.4f" %(sampleset_ord[30])) # position 30 since we start with 0
```

```
median: 169.5007

31th order statistic: 171.1233
```

Part 4 (Determine truncated mean)

In [6]:
```
# determine the truncated mean for some k
def truncmean(sample, k):
    n = len(sample)
    sort = sorted(sample) # ordering the sample
    # definition from the lecture
    return(float(1)/(n-2*k)*sum(sort[k:n-k])) # sum goes from k+1 to n-k (counting the sample fro

# choosing k=1,...,24
k = 10
print("truncated mean for k=%d: %3.2f \n" %(k,truncmean(sampleset, k)))

# checking if truncated mean = median for some k
for k in range(1,25):
    if truncmean(sampleset,k) == med(sampleset):
        print("equality of truncated mean and median found for k=%d" %(k))
```

```
truncated mean for k=10: 169.69

equality of truncated mean and median found for k=24
```

The truncated mean for $k = 24$ ist equal to the median. This is to be expected, since we have an even sample size of 50 and therefore the median is computed as the average of the 25th and the 26th order statistics. The truncated mean for $k = 24$ is also computed as the average of the 25th and the 26th order statistics, so they are equal. This is true for all samples of even length n and for the truncated mean for $k = \frac{n}{2} - 1$.