Chapter 4

Simulation Based Statistics

Contents

4.1 Probability and Simulation

4.1.1 Introduction

One of the really big gains for statistics and modelling of random phenomena, provided by computer technology during the last decades, is the ability to simulate random systems on the computer, as we have already seen much in use in Chapter 2. This provides possibilities to obtain results that otherwise from a mathematical analytical point of view would be impossible to calculate. And, even in cases where the highly educated mathematician/physicist might be able to find solutions, simulation is a general and simple calculation tool allowing solving complex problems without a need for deep theoretical insight.

An important reason for including this subject in an introductory statistics course, apart from using it as a pedagogical tool to aide the understanding of random phenomena, is the fact that the methods we are usually introducing in basic statistics are characterized by relying on one of two conditions:

1. The original data population density is assumed to be a normal distribution
2. Or: The sample size n is large enough to make this assumption irrelevant for what we do

And in real settings it may be challenging to know for sure whether any of these two are really satisfied, so to what extend can we trust the statistical conclusions that we make using our basic tools, as e.g. the one- and two-sample statistical methods presented in Chapter 3. And how should we do the basic statistical analysis if we even become convinced that none of these two conditions are fulfilled? Statistical data analysis based on simulation tools is a valuable tool to complete the tool box of introductory statistics. It can be used to do statistical computing for other features than just means, and for other population distributions than the normal. It can also be used to investigate whether some of our assumptions appear reasonable. We already saw an example of this in relation to the qq-plots in Chapter 3.1.9.

In fact, it will become clear that the simulation tools presented here will make us rapidly able to perform statistical analysis that goes way beyond what historically has been introduced in basic statistics classes or textbooks. Unfortunately, the complexity of real life engineering applications and data analysis challenges can easily go beyond the settings that we have time to cover within an introductory exposition. With the general simulation tool in our tool box, we have

a multi-tool that can be used for (and adapted to) basically almost any level of complexity that we will meet in our future engineering activity.

The classical statistical practice would be to try to ensure that the data we're analyzing behaves like a normal distribution: symmetric and bell-shaped histogram. In Chapter 3 we also learned that we can make a normal q-q plot to verify this assumption in practice, and possibly transform the data to get them closer to being normal. The problem with small samples is that it even with these diagnostic tools can be difficult to know whether the underlying distribution really is ”normal” or not.

And in some cases the assumption of normality after all simply may be obviuosly wrong. For example, when the response scale we work with is far from being quantitative and continuous - it could be a scale like ”small”, ”medium” and ”large” - coded as 1, 2 and 3. We need tools that can do statistical analysis for us WITHOUT the assumption that the normal distribution is the right model for the data we observe and work with.

Traditionally, the missing link would be covered by the so-called non-parametric tests. In short this is a collection of methods that make use of data at a more coarse level, typically by focusing on the rank of the observations instead of the actual values of the observations. So in a paired t-test setup, for example, one would just count how many times the observations in one sample is bigger than in the other – instead of calculating the differences. In that way you can make statistical tests without using the assumption of an underlying normal distribution. There are a large number of such non-parametric tests for different setups. Historically, before the computer age, it was the only way to really handle such situations in practice. These tests are all characterized by the fact that they are given by relatively simple computational formulas which in earlier times easily could be handled. For small sample statistics with questionable distributional settings, these tools maintain to offer a robust set of basic statistical procedures.

The simulation based methods that we now present instead have a couple of crucial advantages to the traditional non-parametric methods:

- Confidence intervals are much easier to achieve
- They are much easier to apply in more complex situations
- They scale better to modern time big data analysis

4.1.2 Simulation as a general computational tool

Basically, the strength of the simulation tool is that one can compute arbitrary functions of random variables and their outcomes. In other words one can find probabilities of complicated outcomes. As such, simulation is really not a statistical tool, but rather a probability calculus tool. However, since statistics essentially is about analysing and learning from real data in the light of certain probabilities, the simulation tool indeed becomes of statistical importance, which we will exemplify very specifically below. Before starting with exemplifying the power of simulation as a general computational tool, we refer to the introduction to simulation in Chapter 2 – in particular read first Section 2.6, Example **??** and thereafter Section 2.6.

|||| Example 4.1 Rectangular plates

A company produces rectangular plates. The length of plates (in meters), X is assumed to follow a normal distribution $N(2, 0.01^2)$ and the width of the plates (in meters), Y are assumed to follow a normal distribution $N(3, 0.02^2)$. We're hence dealing with plates of size 2×3 meters, but with errors in both length and width. Assume that these errors are completely independent. We are interested in the area of the plates which of course is given by $A = XY$. This is a non-linear function of X and Y, and actually it means that we, with the theoretical tools we presented so far in the material, cannot figure out what the mean area really is, and not at all what the standard deviation would be in the areas from plate to plate, and we would definitely not know how to calculate the probabilities of various possible outcomes. For example, how often such plates have an area that differ by more than 0.1 m^2 from the targeted 6 m^2? One statement summarizing all our lack of knowledge at this point: we do not know the probability distribution of the random variable A and we do not know how to find it! With simulation, it is straightforward: one can find all relevant information about A by just simulating the X and Y a high number of times, and from this compute A just as many times, and then observe what happens to the values of A. The first step is then given by:

```
# Number of simulations
k = 10000

# Simulate X and Y, then A
X = stats.norm.rvs(loc=2,scale=0.01,size=k)
Y = stats.norm.rvs(loc=3,scale=0.02,size=k)
A = X * Y
```

The Python object `A` now contains 10.000 observations of A. The expected value

and the standard deviation for A are simply found by calculating the average and standard deviation for the simulated A-values:

```
# The mean and std. deviation of the simulated values
print(A.mean())

6.000707518857636

print(A.std(ddof=1))

0.050187379229233574
```

and the desired probability, $P(|A-6| > 0.1) = 1 - P(5.9 \leq A \leq 6.1)$ is found by counting how often the incident actually occurs among the k outcomes of A:

```
1*(abs(A-6) > 0.1).mean()

np.float64(0.0454)
```

The code `abs(A-6) > 0.1` creates an array with values `TRUE` or `FALSE` depending on whether the absolute value of $A-6$ is greater than 0.1 or not. When you multiply by 1 the `TRUE` is automatically translated into 1 and `FALSE` automatically translated to 0. To find the probability, we sum these binary values and divide by number of simulations k. This is equivalent to finding the mean of the binary values, and therefore we use the mean method.

Note, that if you do this yourself without using the same seed value you will not get exactly the same result. It is clear that this simulation uncertainty is something we must deal with in practice. The size of this will depend on the situation and on the number of simulations k. We can always get a first idea of it in a specific situation simply by repeating the calculation a few times and note how it varies. Indeed, one could then formalize such an investigation and repeat the simulation many times, to get an evaluation of the simulation uncertainty. We will not pursue this further here. When the target of the computation is in fact a probability, as in the latter example here, you can alternatively use standard binomial statistics, which is covered in Chapter 2 and Chapter 7. For example, with $k = 100000$ the uncertainty for a calculated proportion of around 0.044 is given by: $\sqrt{\frac{0.044(1-0.044)}{100000}} = 0.00065$. Or for example, with $k = 10000000$ the uncertainty is 0.000065. The result using such a k was 0.0455 and because we're a bit unlucky with the rounding position we can in practice say that the exact result rounded to 3 decimal places are either 0.045 or

0.046. In this way, a calculation which is actually based on simulation is turned into an exact one in the sense that rounded to 2 decimal places, the result is simply 0.05.

4.1.3 Propagation of error

Within chemistry and physics one may speak of measurement errors and how measurement errors propagate/accumulate if we have more measurements and/or use these measurements in subsequent formulas/calculations. First of all: The basic way to "measure an error", that is, to quantify a measurement error is by means of a standard deviation. As we know, the standard deviation expresses the average deviation from the mean. It is clear it may happen that a measuring instrument also on average measures wrongly (off the target). This is called "bias", but in the basic setting here, we assume that the instrument has no bias.

Hence, reformulated, an error propagation problem is a question about how the standard deviation of some function of the measurements depends on the standard deviations for the individual measurement: let $X_1, \ldots, X_n$ be n measurements with standard deviations (average measurement errors) $\sigma_1, \ldots, \sigma_n$. As usual in this material, we assume that these measurement errors are independent of each other. There are extensions of the formulas that can handle dependencies, but we omit those here. We must then in a general formulation be able to find

$$\sigma^2_{f(X_1,\ldots,X_n)} = \mathrm{V}(f(X_1, \ldots, X_n)). \tag{4-1}$$

Remark 4.2

[For the thoughtful reader: Measurement errors, errors and variances] Although we motivate this entire treatment by the *measurement error* terminology, often used in chemistry and physics, actually everything is valid for *any kind of* errors, be it "time-to-time" production errors, or "substance-to-substance" or "tube-to-tube" errors. What the relevant kind of errors/variabilities are depends on the situation and may very well be mixed together in applications. But, the point is that as long as we have a relevant error variance, we can work with the concepts and tools here. It does not have to have a "pure measurement error" interpretation.

Actually, we have already in this course seen the linear error propagation rule,

in Theorem in 2.56, which then can be restated here as

$$\text{If } f(X_1,\ldots,X_n) = \sum_{i=1}^{n} a_i X_i, \quad \text{then} \quad \sigma^2_{f(X_1,\ldots,X_n)} = \sum_{i=1}^{n} a_i^2 \sigma_i^2.$$

There is a more general non-linear extension of this, albeit theoretically only an approximate result, which involves the partial derivative of the function f with respect to the n variables:

|||| Method 4.3 The non-linear approximative error propagation rule

If $X_1,\ldots,X_n$ are independent random variables with variances $\sigma_1^2,\ldots,\sigma_n^2$ and f is a (potentially non-linear) function of n variables, then the variance of the f-transformed variables can be approximated linearly by

$$\sigma^2_{f(X_1,\ldots,X_n)} = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_i^2, \tag{4-2}$$

where $\frac{\partial f}{\partial x_i}$ is the partial derivative of f with respect to the i'th variable

In practice one would have to insert the actual measurement values $x_1,\ldots,x_n$ of $X_1,\ldots,X_n$ in the partial derivatives to apply the formula in practice, see the example below. This is a pretty powerful tool for the general finding of (approximate) uncertainties for complicated functions of many measurements or for that matter: complex combinations of various statistical quantities. When the formula is used for the latter, it is also in some contexts called the "delta rule" (which is mathematically speaking a so-called first-order (linear) Taylor approximations to the non-linear function f). We bring it forward here, because as an alternative to this approximate formula one could use simulation in the following way:

Method 4.4 Non-linear error propagation by simulation

Assume we have actual measurements $x_1, \ldots, x_n$ with known/assumed error variances $\sigma_1^2, \ldots, \sigma_n^2$:

1. Simulate k outcomes of all n measurements from assumed error distributions, e.g. $N(x_i, \sigma_i^2)$: $X_i^{(j)}, j = 1 \ldots, k$.
2. Calculate the standard deviation directly as the observed standard deviation of the k simulated values of f:

$$s^{\text{sim}}_{f(X_1,\ldots,X_n)} = \sqrt{\frac{1}{k-1}\sum_{j=1}^{k}(f_j - \bar{f})^2}, \tag{4-3}$$

where

$$f_j = f(X_1^{(j)}, \ldots, X_n^{(j)}). \tag{4-4}$$

Example 4.5

Let us continue the example with $A = XY$ and X and Y defined as in the example above. First of all note, that we already above used the simulation based error propagation method, when we found the standard deviation to be 0.04957 based on the simulation. To exemplify the approximate error propagation rule, we must find the derivatives of the function $f(x, y) = xy$ with respect to both x and y

$$\frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x.$$

Assume, that we now have two specific measurements of X and Y, for example $x = 2.00$ m and $y = 3.00$ m the error propagation law would provide the following approximate calculation of the "uncertainty error variance of the area result" $2.00\text{ m} \cdot 3.00\text{ m} = 6.00\text{ m}^2$, namely

$$\sigma_A^2 = y^2 \cdot 0.01^2 + x^2 \cdot 0.02^2 = 3.00^2 \cdot 0.01^2 + 2.00^2 \cdot 0.02^2 = 0.0025.$$

So, with the error propagation law we are managing a part of the challenge without simulating. Actually, we are pretty close to be able to find the correct theortical variance of $A = XY$ using tools provided in this course. By the definition and the following fundamental relationship

$$\mathrm{V}(X) = \mathrm{E}(X - \mathrm{E}(X))^2 = \mathrm{E}(X^2) - \mathrm{E}(X)^2. \tag{4-5}$$

So, one can actually deduce the variance of A theoretically, it is only necessary to know in addition that for independent random variables: $\mathrm{E}(XY) = \mathrm{E}(X)\,\mathrm{E}(Y)$ (which by the way then also tells us that $\mathrm{E}(A) = \mathrm{E}(X)\,\mathrm{E}(Y) = 6$)

$$\begin{aligned}
\mathrm{V}(XY) &= \mathrm{E}\left[(XY)^2\right] - \mathrm{E}(XY)^2 \\
&= \mathrm{E}(X^2)\,\mathrm{E}(Y^2) - \mathrm{E}(X)^2\,\mathrm{E}(Y)^2 \\
&= \left[\mathrm{V}(X) + \mathrm{E}(X)^2\right]\left[\mathrm{V}(Y) + \mathrm{E}(Y)^2\right] - \mathrm{E}(X)^2\,\mathrm{E}(Y)^2 \\
&= \mathrm{V}(X)\,\mathrm{V}(Y) + \mathrm{V}(X)\,\mathrm{E}(Y)^2 + \mathrm{V}(Y)\,\mathrm{E}(X)^2 \\
&= 0.01^2 \cdot 0.02^2 + 0.01^2 \cdot 3^2 + 0.02^2 \cdot 2^2 \\
&= 0.00000004 + 0.0009 + 0.0016 \\
&= 0.00250004.
\end{aligned}$$

Note, how the approximate error propagation rule actually corresponds to the two latter terms in the correct variance, while the first term – the product of the two variances is ignored. Fortunately, this term is the smallest of the three in this case. It does not always have to be like that. If you want to learn how to make a theoretical derivation of the density function for $A = XY$ then take a course in probability calculation.

Note, how we in the example actually found the "average error", that is, the error standard deviation by three different approaches:

1. The simulation based approach
2. The analytical, but approximate, error propagation method
3. A theoretical derivation

The simulation approach has a number of crucial advantages:

1. It offers a simple way to compute many other quantities than just the standard deviation (the theoretical derivations of such other quantities could be much more complicated than what was shown for the variance here)
2. It offers a simple way to use any other distribution than the normal – if we believe such better reflect reality
3. It does not rely on any linear approximations of the true non-linear relations

4.2 The parametric bootstrap

4.2.1 Introduction

Generally, a confidence interval for an unknown parameter μ is a way to express uncertainty using the sampling distribution of $\hat{\mu} = \bar{x}$. Hence, we use a distribution that expresses how our calculated value would vary from sample to sample. And the sampling distribution is a theoretical consequence of the original population distribution. As indicated, we have so far no method to do this if we only have a small sample size ($n < 30$), and the data cannot be assumed to follow a normal distribution. In principle there are two approaches for solving this problem:

1. Find/identify/assume a different and more suitable distribution for the population (”the system”)
2. Do not assume any distribution whatsoever

The simulation method called bootstrapping, which in practice is to simulate many samples, exists in two versions that can handle either of these two challenges:

1. Parametric bootstrap: simulate multiple samples from the assumed distribution.
2. Non-parametric bootstrap: simulate multiple samples directly from the data.

Actually, the parametric bootstrap handles in addition the situation where data could perhaps be normally distributed, but where the calculation of interest is quite different than the average, for example, the coefficient of variation (standard deviation divided by average) or the median. This would be an example of a non-linear function of data – thus not having a normal distribution nor a t-distribution as a sampling distribution. So, the parametric bootstrap is basically just an example of the use of simulation as a general calculation tool, as introduced above. Both methods are hence very general and can be used in virtually all contexts.

In this material we have met a few of such alternative continuous distributions, e.g. the log-normal, uniform and exponential distributions. But if we think about it, we have not (yet) been taught how to do any statistics (confidence intervals and/or hypothesis testing) within an assumption of any of these. The

parametric bootstrap is a way to do this without relying on theoretical derivations of everything. As for the theoretical variance deduction above, there are indeed methods for doing such general theoretical derivations, which would make us able to do statistics based on any kind of assumed distribution. The most welknown, and in many ways also optimal, overall approach for this is called *maximum likelihood* theory. The general theory and approach of maximum likelihood is not covered in this course, however it is good to know that, in fact, all the methods we present are indeed also maximum likelihood methods assuming normal distributions for the population(s).

4.2.2 One-sample confidence interval for μ

|||| Example 4.6 Confidence interval for the exponential rate or mean

Assume that we observed the following 10 call waiting times (in seconds) in a call center

$$32.6,\ 1.6,\ 42.1,\ 29.2,\ 53.4,\ 79.3,\ 2.3,\ 4.7,\ 13.6,\ 2.0.$$

If we model the waiting times using the exponential distribution, we can estimate the mean as

$$\hat{\mu} = \bar{x} = 26.08,$$

and hence the rate parameter $\lambda = 1/\beta$ in the exponential distribution as (cf. 2.48)

$$\hat{\lambda} = 1/26.08 = 0.03834356.$$

However, what if we want a 95% confidence interval for either $\mu = \beta$ or λ? We have not been tought the methods, that is, given any formulas for finding this. The following few lines of Python-code, a version of the simulation based error propagation approach from above, will do the job for us:

```
# Read the data
x = np.array([32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0])
n = len(x)
rate = 1/x.mean()

# Set the number of simulations
k = 100000

# 1. Simulate k samples each with n=10 observations from  an
#    exponential distribution with the estimated rate
simsamples = stats.expon.rvs(scale=1/rate, size=(k,n))

# 2. Compute the mean in each of the k samples
simsamples = pd.DataFrame(simsamples)
simmeans = simsamples.mean(axis=1)

# 3. Find the two relevant quantiles of the k generated means
print(np.quantile(simmeans,[0.025,0.975],
method='averaged_inverted_cdf'))

[12.575 44.563]
```

Explanation: we use `stats.expon.rvs` to generate 100.000 bootstrap samples each with 10 observations from an exponential distribution with the estimated mean value, and the results are collected in a 10×100.000 matrix. Then in a single call the 100.000 averages are calculated and subsequently the relevant quantiles found.

So the 95%-confidence interval for the mean μ is (in seconds)

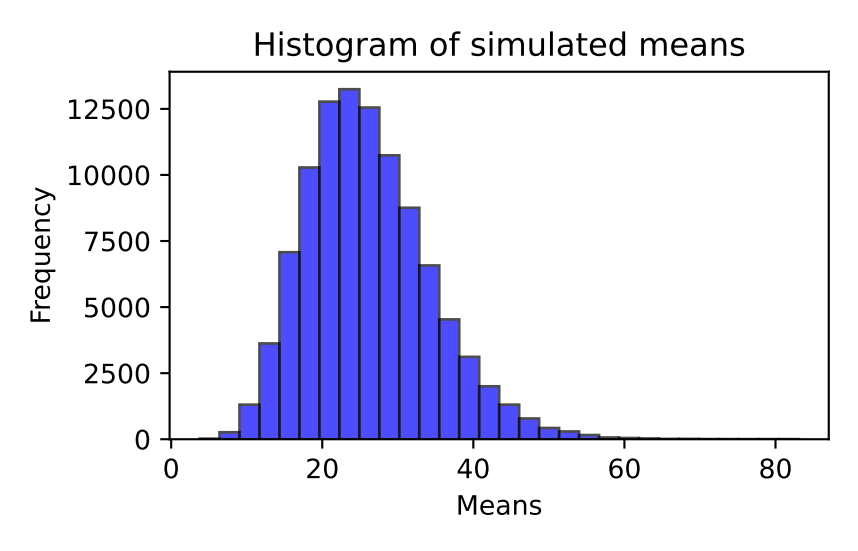
$$[12.6,\ 44.6].$$

And for the rate $\lambda = 1/\mu$ it can be found by a direct transformation (remember that the quantiles are 'invariant' to monotonic transformations, c.f. Chapter 3)

$$[1/44.6,\ 1/12.6] \Leftrightarrow [0.022,\ 0.0794].$$

The simulated sampling distribution of means that we use for our statistical analysis can be seen with the histogram:

```
# Histogram of the simulated means
plt.hist(simmeans, bins=30, edgecolor='black', color='blue', alpha=0.7)
plt.xlabel('Means')
plt.ylabel('Frequency')
plt.title('Histogram of simulated means')
plt.tight_layout()
plt.show()
```



We see clearly that the sampling distribution in this case is not a normal nor a t-distribution: it has a clear right skewed shape. So $n = 10$ is not quite large enough for this exponential distribution to make the Central Limit Theorem take over.

The general method which we have used in the example above is given below as Method 4.7.

4.2.3 One-sample confidence interval for any feature assuming any distribution

We saw in the example above that we could easily find a confidence interval for the rate $\lambda = 1/\mu$ assuming an exponential distribution. This was so, since the rate was a simple (monotonic) transformation of the mean, and the quantiles of simulated rates would then be the same simple transformation of the quantiles of the simulated means. However, what if we are interested in something not expressed as a simple function of the mean, for instance the median, the

coefficient of variation, the quartiles, Q_1 or Q_3, the IQR=$Q_3 - Q_1$ or any other quantile? Well, a very small adaptation of the method above would make that possible for us. To express that we now cover any kind of statistic one could think of, we use the general notation, the Greek letter θ, for a general feature of the distribution. For instance, θ could be the true median of the population distribution, and then $\hat{\theta}$ is the sample median computed from the sample taken.

|||| Method 4.7 Confidence interval for any feature θ by parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$ and assume that they stem from some probability distribution with density (pdf) f:

1. Simulate k samples of n observations from the assumed distribution f where the mean is set to $\bar{x}$ [a]
2. Calculate the statistic $\hat{\theta}$ in each of the k samples $\hat{\theta}_1^*, \ldots, \hat{\theta}_k^*$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%},\ q^*_{100(1-\alpha/2)\%}\right]$

[a] *(Footnote: And otherwise chosen to match the data as good as possible: some distributions have more than just a single mean related parameter, e.g. the normal or the log-normal. For these one should use a distribution with a variance that matches the sample variance of the data. Even more generally the approach would be to match the chosen distribution to the data by the so-called maximum likelihood approach)*

Please note again, that you can simply substitute the θ with whatever statistics that you are working with. This then also shows that the method box includes the often occurring situation, where a confidence interval for the mean μ is the aim.

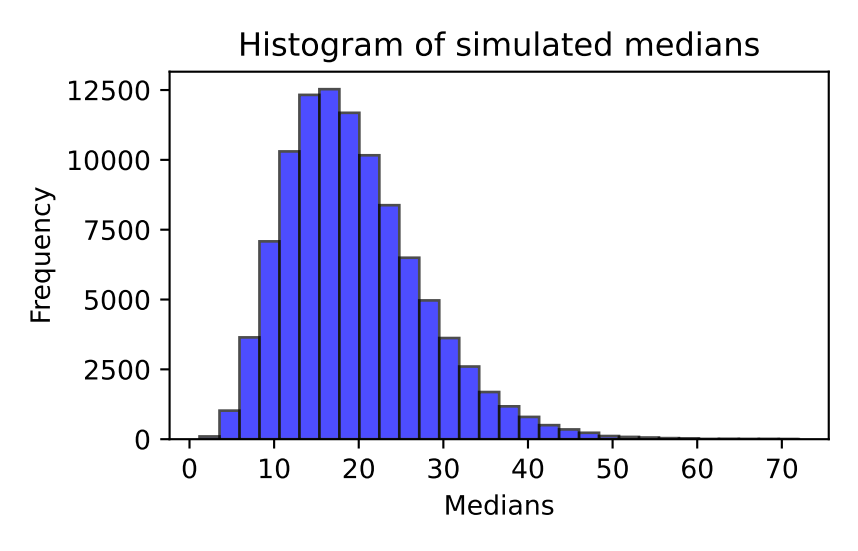
|||| Example 4.8 Confidence interval for the median assuming an exponential distribution

Let us look at the exponential data from the previous section and find the confidence interval for the median:

```
# Read the data
x = np.array([32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0])
n = len(x)
rate = 1/x.mean()

# Set the number of simulations
k = 100000

# 1. Simulate k samples each with n=10 observations from  an
#    exponential distribution with the estimated rate
simsamples = stats.expon.rvs(scale=1/rate, size=(k,n))

# 2. Compute the median in each of the k samples
simsamples = pd.DataFrame(simsamples)
simmedians = simsamples.median(axis=1)

# 3. Find the two relevant quantiles of the k generated medians
print(np.quantile(simmedians,[0.025,0.975],
method='averaged_inverted_cdf'))

[ 7.093 38.333]
```

The simulated sampling distribution of medians that we use for our statistical analysis can be studied by the histogram:

```
# Histogram of the simulated medians
plt.hist(simmedians, bins=30, edgecolor='black', color='blue',
alpha=0.7)
plt.xlabel('Medians')
plt.ylabel('Frequency')
plt.title('Histogram of simulated medians')
plt.tight_layout()
plt.show()
```



We see again clearly that the sampling distribution in this case is not a normal nor a t-distribution: it has a clear right skewed shape.

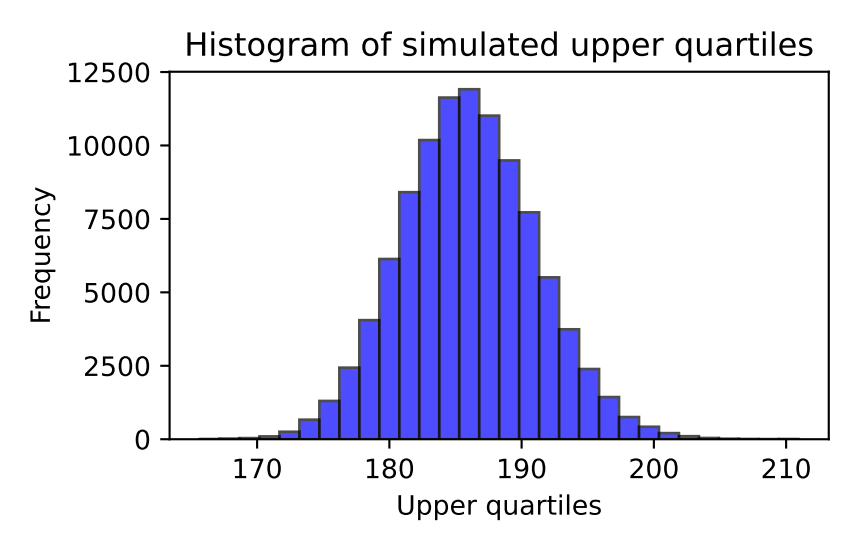
|||| Example 4.9 Confidence interval for Q_3 assuming a normal distribution

Let us look at the heights data from the previous chapters and find the 99% confidence interval for the upper quartile: (Please note that you will find NO theory nor analytically expressed method boxes in the material to solve this challenge). We proceed like in the previous example:

```
# Read the data
x = np.array([168, 161, 167, 179, 184, 166, 198, 187, 191, 179])
n = len(x)
mu = x.mean()
sd = x.std(ddof=1)

# Set the number of simulations
k = 100000

# 1. Simulate k samples each with n=10 observations from  a
#    normal distribution with the estimated parameters
simsamples = stats.norm.rvs(loc=mu, scale=sd, size=(k,n))

# 2. Compute the upper quartile in each of the k samples
simsamples = pd.DataFrame(simsamples)

# A version using the quantile-function from Pandas.
# Note: Pandas does not use our method of calculating quantiles.
simUQs = simsamples.quantile(0.75, axis=1)

# A version using the quantile-function from NumPy.
simUQs =  np.quantile(simsamples, 0.75, axis=1,
method="averaged_inverted_cdf")

# 3. Find the two relevant quantiles of the k generated medians
print(np.quantile(simUQs,[0.005,0.995],method='averaged_inverted_cdf'))


[173.481 199.813]
```

The simulated sampling distribution of upper quartiles that we use for our statistical analysis can be studied by the histogram:

```
# Histogram of the simulated upper quartiles
plt.hist(simUQs, bins=30, edgecolor='black', color='blue', alpha=0.7)
plt.xlabel('Upper quartiles')
plt.ylabel('Frequency')
plt.title('Histogram of simulated upper quartiles')
plt.tight_layout()
plt.show()
```



In this case the Q_3 of $n = 10$ samples of a normal distribution appear to be rather symmetric and nicely distributed, so maybe one could in fact use the normal distribution, also as an approximate sampling distribution in this case.

4.2.4 Two-sample confidence intervals assuming any distributions

In this section we extend what we learned in the two previous sections to the case where the focus is a comparison between two (independent) samples. We present a method box which is the natural extensions of the method box from above, comparing any kind of feature (hence including the mean comparison):

|||| Method 4.10 Two-sample confidence interval for any feature comparison $\theta_1 - \theta_2$ by parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$ and assume that they stem from some probability distributions with density f_1 and f_2:

1. Simulate k sets of 2 samples of n_1 and n_2 observations from the assumed distributions setting the means to $\hat{\mu}_1 = \bar{x}$ and $\hat{\mu}_2 = \bar{y}$, respectively [a]
2. Calculate the difference between the features in each of the k samples $\hat{\theta}^*_{x1} - \hat{\theta}^*_{y1}, \ldots, \hat{\theta}^*_{xk} - \hat{\theta}^*_{yk}$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

[a] *(Footnote: And otherwise chosen to match the data as good as possible: some distributions have more than just a single mean related parameter, e.g. the normal or the log-normal. For these one should use a distribution with a variance that matches the sample variance of the data. Even more generally the approach would be to match the chosen distribution to the data by the so-called maximum likelihood approach)*

|||| Example 4.11 CI for the difference of two means from exponential distributed data

Let us look at the exponential data from the previous section and compare that with a second sample of $n = 12$ observations from another day at the call center

$$9.6,\ 22.2,\ 52.5,\ 12.6,\ 33.0,\ 15.2,\ 76.6,\ 36.3,\ 110.2,\ 18.0,\ 62.4,\ 10.3.$$

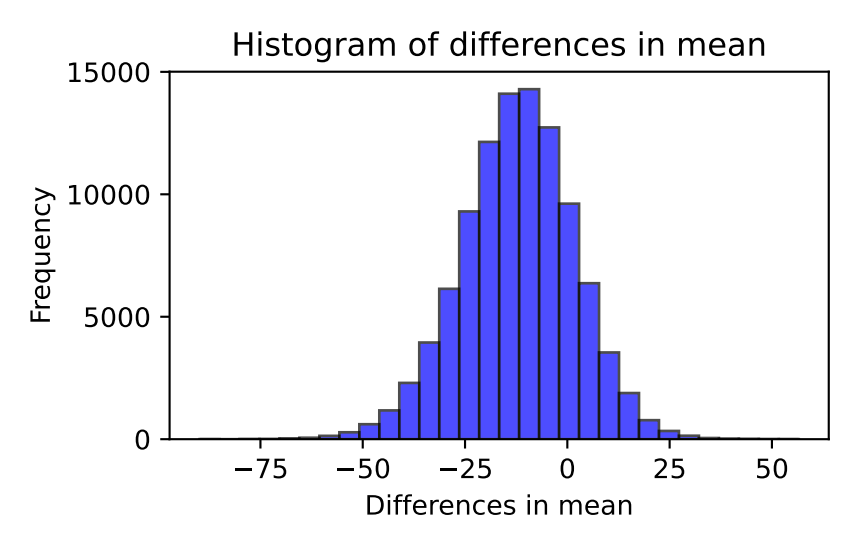
Let us quantify the difference between the two days and conclude whether the call rates and/or means are any different on the two days:

```
# Read the data
x = np.array([32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0])
y = np.array([9.6, 22.2, 52.5, 12.6, 33.0, 15.2, 76.6, 36.3, 110.2,
18.0, 62.4, 10.3])
n1 = len(x)
n2 = len(y)
rate1 = 1/x.mean()
rate2 = 1/y.mean()

# Set the number of simulations
k = 100000

# 1. Simulate k samples each with n1=10 observations from an
#    exponential distribution with the estimate rate of X
simXsamples = stats.expon.rvs(scale=1/rate1, size=(k,n1))
simXsamples = pd.DataFrame(simXsamples)

# 2. Simulate k samples each with n2=12 observations from  an
#    exponential distribution with the estimated rate of Y
simYsamples = stats.expon.rvs(scale=1/rate2, size=(k,n2))
simYsamples = pd.DataFrame(simYsamples)

# 3. Compute the difference between the two simulated means - k times
simDifmeans = simXsamples.mean(axis=1) - simYsamples.mean(axis=1)

# 4. Find the two relevant quantiles of the k generated differences in
mean
print(np.quantile(simDifmeans,[0.025,0.975],
method='averaged_inverted_cdf'))

[-40.521  14.028]
```

Thus, although the mean waiting time was higher on the second day ($\bar{y} = 38.24$ s), the range of acceptable values (the confidence interval) for the difference in means is $[-40.5,\ 14.0]$ – a pretty large range and including 0, so we have no evidence of the claim that the two days had different mean waiting times (nor call rates then) based on the current data.

Let us, as in previous examples take a look at the distribution of the simulated samples. In a way, we do not really need this for doing the analysis, but just out of curiosity, and for the future it may give a idea of how far from normality the relevant sampling distribution really is:

```
# Histogram of the simulated differences
plt.hist(simDifmeans, bins=30, edgecolor='black', color='blue',
alpha=0.7)
plt.xlabel('Differences in mean')
plt.ylabel('Frequency')
plt.title('Histogram of differences in mean')
plt.tight_layout()
plt.show()
```



In this case the differences of means of exponential distributions appears to be rather symmetric and nicely distributed, so maybe one could in fact use the normal distribution, also as an approximate sampling distribution in this case.

|||| Example 4.12 Nutrition study: comparing medians assuming normal distributions

Let us compare the median energy levels from the two-sample nutrition data from Example 3.46. And let us do this still assuming the normal distribution as we also assumed in the previous example. First we read in the data:

```
# Read the data
xA = np.array([7.53, 7.48, 8.08, 8.09, 10.15, 8.4, 10.88, 6.13, 7.9])
xB = np.array([9.21, 11.51, 12.79, 11.85, 9.97, 8.79, 9.69, 9.68,
9.19])
nA = len(xA)
nB = len(xB)
```

Then we do the two-sample median comparison by the parametric, normal based, bootstrap:

```
# Set the number of simulations
k = 100000

# 1. Simulate k samples each with nA=9 observations from a
#    normal distribution with the estimate parameters for group A
simAsamples = stats.norm.rvs(loc=xA.mean(), scale=xA.std(ddof=1) ,
size=(k,nA))
simAsamples = pd.DataFrame(simAsamples)

# 2. Simulate k samples each with nB=9 observations from a
#    normal distribution with the estimate parameters for group B
simBsamples = stats.norm.rvs(loc=xB.mean(), scale=xB.std(ddof=1) ,
size=(k,nB))
simBsamples = pd.DataFrame(simBsamples)

# 3. Compute the difference between the two simulated medians - k
times
simDifmedians = simAsamples.median(axis=1) - simBsamples.median(axis=1)


# 4. Find the two relevant quantiles of the k generated differences in
medians
print(np.quantile(simDifmedians,[0.025,0.975],
method='averaged_inverted_cdf'))

[-3.617 -0.401]
```

Thus, we accept that the difference between the two medians is somewhere between 0.4 and 3.6, and confirming the group difference that we also found in the means, as the 0 is not included in the interval.

Note the differences in the Python code compared to the previous bootstrapping example: we use the `stats.expon.rvs`-function instead of the `stats.norm.rvs`-function and change the method from `.mean` to `.median`.

|||| Remark 4.13 Hypothesis testing by simulation based confidence intervals

We have also seen that even though the simulation method boxes given are providing confidence intervals: we can also use this for hypothesis testing, by using the basic relation between hypothesis testing and confidence intervals. A confidence interval includes the 'acceptable' values, and values outside the confidence interval are the 'rejectable' values.

4.3 The non-parametric bootstrap

4.3.1 Introduction

In the introduction to the parametric bootstrap section above it was discussed that another approach instead of finding the 'right' distribution to use is to not assume any distribution at all. This can be done, and a way to do this simulation based is called the *non-parametric bootstrap* and is presented in this section. The section is structured as the parametric bootstrap section above – including the similar subsections and similar method boxes. So there will be two method boxes in this section: one for the one-sample analysis and one for the two-sample analysis.

In fact, the non-parametric approach could be seen as the parametric approach but substituting the density/distribution used for the simulation by the observed distribution of the data, that is, the empirical cumulative distribution function (ecdf), cf. Chapter 1. In practice this is carried out by (re)-sampling the data we have again and again: To get the sampling distribution of the mean (or any other feature) based on the n observations that we have in our given sample, we simply again and again take new samples with n observations from the one we have. This is done "with replacement" such that the "new" samples, from now on called the *bootstrap samples* would contain some of the original observations in duplicates (or more) and others will not be there.

4.3.2 One-sample confidence interval for μ

We have the sample: $x_1, \ldots, x_n$.

The $100(1-\alpha)\%$ confidence interval for μ determined by the non-parametric bootstrap is first exemplified:

|||| Example 4.14 Women's cigarette consumption

In a study women's cigarette consumption before and after giving birth is explored. The following observations of the number of smoked cigarettes per day were observed:

before	after	before	after
8	5	13	15
24	11	15	19
7	0	11	12
20	15	22	0
6	0	15	6
20	20		

This is a typical paired t-test setup, as discussed in Section 3.2.3, which then was handled by finding the 11 differences and thus transforming it into a one-sample setup. First we read the observations into Python and calculate the differences by:

```
# # Read the data and calculate the difference for each woman before
and after
x1 = np.array([8, 24, 7, 20, 6, 20, 13, 15, 11, 22, 15])
x2 = np.array([5, 11, 0, 15, 0, 20, 15, 19, 12, 0, 6])
dif = x1-x2
print(dif)

[ 3 13  7  5  6  0 -2 -4 -1 22  9]
```

There is a random-sampling function in the NumPy package (which again is based on a uniform random number generator): `np.random.choice`. Eg. you can get 5 repeated samples with replacement by: (Note that the argument `replace` is `true` by default. Sampling without replacement can thus be done by specifying `replace=false`.)

```
np.random.choice(dif,size=(5,len(dif)))

array([[-4,  5,  0, -1,  7, -2, -2,  5,  0, -2,  3],
       [ 3, -1,  5, -2,  9, 13, -2, -1,  0, 13,  9],
       [ 6, -2, 22,  7, -4,  7,  7,  5,  9,  5, 22],
       [-1,  5,  6, -1,  5, -1,  7,  3,  3,  6,  6],
       [-4,  7, -2, -2,  3,  7, -2, -2,  9, 13, 22]])
```

Explanation: the first argument, `dif`, defines the sampling space, and the second argument, `size=(5,len(dif))`, defines the number of bootstrap samples, 5, and their respective sizes, `len(dif)`.

One can then run the following to get a 95% confidence interval for μ based on $k = 100000$:

```
# Set the number of simulations
k = 100000

# Simulate k samples each with 11 observations by
# sampling with replacement from the data
simsamples = np.random.choice(dif,size=(k,len(dif)))

# Compute the mean in each of the k samples
simmeans = simsamples.mean(axis=1)

# Find the two relevant quantiles of the k generated meeans
print(np.quantile(simmeans,[0.025,0.975],
method='averaged_inverted_cdf'))

[1.364 9.818]
```

Explanation: The np.random.choice-function is called 100.000 times and the results collected in an 11×100.000 matrix. Then in a single call the 100.000 averages are calculated and subsequently the relevant quantiles found.

Note, that we use the similar three steps as above for the parametric bootstrap, with the only difference that the simulations are carried out by the re-sampling the given data rather than from some probability distribution.

4.3.3 One-sample confidence interval for any feature

What we have just done can be more generally expressed as follows:

Method 4.15 Confidence interval for any feature θ by non-parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$:

1. Simulate k samples of size n by randomly sampling among the available data (with replacement)
2. Calculate the statistic $\hat{\theta}$ in each of the k samples $\hat{\theta}_1^*, \ldots, \hat{\theta}_k^*$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

Example 4.16

Let us find the 95% confidence interval for the median cigarette consumption change in the example from above:

```
# The 95% CI for the median change
k = 100000
simsamples = pd.DataFrame(np.random.choice(dif,size=(k,len(dif))))
simmedians = simsamples.median(axis=1)
print(np.quantile(simmedians,[0.025,0.975],
method='averaged_inverted_cdf'))

[-1.000  9.000]
```

4.3.4 Two-sample confidence intervals

We now have two random samples: $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$. The $100(1-\alpha)\%$ confidence interval for $\theta_1 - \theta_2$ determined by the non-parametric bootstrap is defined as:

Method 4.17 Two-sample confidence interval for $\theta_1 - \theta_2$ by non-parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$:

1. Simulate k sets of 2 samples of n_1 and n_2 observations from the respective groups (with replacement)
2. Calculate the difference between the features in each of the k samples $\hat{\theta}^*_{x1} - \hat{\theta}^*_{y1}, \ldots, \hat{\theta}^*_{xk} - \hat{\theta}^*_{yk}$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

Example 4.18 Teeth and bottle

In a study it was explored whether children who received milk from bottle as a child had worse or better teeth health conditions than those who had not received milk from the bottle. For 19 randomly selected children it was recorded when they had their first incident of caries:

bottle	age	bottle	age	bottle	Age
no	9	no	10	yes	16
yes	14	no	8	yes	14
yes	15	no	6	yes	9
no	10	yes	12	no	12
no	12	yes	13	yes	12
no	6	no	20		
yes	19	yes	13		

One can then run the following to obtain a 95 % confidence interval for $\mu_1 - \mu_2$ based on $k = 100000$:

```
# Reading in "no bottle" group
x = np.array([9, 10, 12, 6, 10, 8, 6, 20, 12])
# Reading in "yes bottle" group
y = np.array([14,15,19,12,13,13,16,14,9,12])
# Number of simulations
k = 100000
# Simulate each sample k times
simxsamples = pd.DataFrame(np.random.choice(x,size=(k,len(x))))
simysamples = pd.DataFrame(np.random.choice(y,size=(k,len(y))))
# Calculate the sample mean differences
simmeandifs = simxsamples.mean(axis=1) - simysamples.mean(axis=1)
# Quantiles of the differences gives the CI
print(np.quantile(simmeandifs,[0.025,0.975],
method='averaged_inverted_cdf'))

[-6.211 -0.122]
```

|||| Example 4.19

Let us make a 99% confidence interval for the difference of medians between the two groups in the tooth health example:

```
# CI for the median differences
simmediandifs = simxsamples.median(axis=1) - simysamples.median(axis=1)

print(np.quantile(simmediandifs,[0.005,0.995],
method='averaged_inverted_cdf'))

[-8.000  0.000]
```

|||| Remark 4.20 Warning: Bootstrapping may not always work well for small sample sizes!

The bootstrapping idea was presented here rather enthusiastically as an almost magic method that can do everything for us in all cases. This is not the case. Some statistics are more easily bootstrapped than others and generally non-parametric bootstrap will not work well for small samples. The inherent lack of information with small samples cannot be removed by any magic trick. Also, there are more conceptually difficult aspects of bootstrapping for various purposes to improve on some of these limitations, see the next section. Some of the "naive bootstrap" CI interval examples introduced in this chapter is likely to not have extremely good properties – the coverage percentages might not in all cases be exactly at the aimed nominal levels.

Glossaries

Empirical cumulative distribution [Empirisk fordeling] The empirical cumulative distribution function F_n is a step function with jumps i/n at observation values, where i is the number of identical observations at that value 23

Exponential distribution [Eksponential fordelingen] The usual application of the exponential distribution is for describing the length (usually time) between events which, when counted, follows a Poisson distribution 9, 10, 12, 20

Histogram [Histogram] The default histogram uses the same width for all classes and depicts the raw frequencies/counts in each class. By dividing the raw counts by n times the class width the density histogram is found where the area of all bars sum to 1 2, 11, 14, 16

Maximum likelihood [Estimator baseret på maximum likelihood metoden] 10, 13, 18

Median [Median, stikprøvemedian] The median of population or sample (note, in text no distinguishment between *population median* and *sample median*) 9, 12, 13

Non-parametric (test) [Ikke-parametriske (tests)] 2, 9, 23, 26

Normal distribution [Normal fordeling] 1–3, 9, 10, 17, 20

Acronyms

ANOVA Analysis of Variance *Glossary:* Analysis of Variance

cdf cumulated distribution function *Glossary:* cumulated distribution function

CI confidence interval 9, 10, 12–15, 18, 22, 23, 26, *Glossary:* confidence interval

CLT Central Limit Theorem 12, *Glossary:* Central Limit Theorem

IQR Inter Quartile Range 13, *Glossary:* Inter Quartile Range

LSD Least Significant Difference *Glossary:* Least Significant Difference

pdf probability density function *Glossary:* probability density function