Assignment #1

Simulation of Random Data from Different Models

Responsible: Mats Gustafsson

Contents

1 GOALS

After completing this homework you are expected to be able to explain about and simulate data (using Python) from the following different types of statistical models commonly used in various technological applications:

- PART A: Discrete stochastic variables - urn models
- PART B: Continuous valued stochastic variables - pdf models
- PART C: Markov and Hidden Markov Models with discrete time and discrete states
- PART D: Linear relationships embedded in measurement noise

The overarching task here is to learn how to use Python to generate data from different statistical models commonly used in various technological applications.

Remark.
This will make it clear that each such model is defined by a set of parameters that for a particular application usually have unknown values. Therefore one usually has to come up with estimates of these parameters based on a collected dataset available. Since the number of observations used to obtain estimates always are relatively few, these estimates always come with uncertainty, which is one central topic for later parts of the course.

2 REMINDER - MOUNT YOUR DISK TO GOOGLE COLAB

This assignment should be performed by means of the freely available Google Colab. You first need to create a personal gmail account unless you do not already have one that you prefer to use. You may easily create a personal gmail account at

www.google.com

Google Colab is available here.

https://colab.research.google.com/notebooks/intro.ipynb

where it offers an online notebook, which is more or less identical to the Jupyter notebook (for those who are familiar with this notebook). For a 3 minutes long video introduction to Colab see:

https://youtu.be/inN8seMm7UI

Remark. If you do not want to use your current gmail account (if you already have one), you can create a new gmail account to be used only during this course.

Follow the instructions on the Colab introduction page in order to create and open a new Colab notebook to be used in this assignment. After this step, rename the created notebook by clicking on the text on the top line and then changing the filename to make it include you personal name, like `Assignment_1_Mats_Gustafsson.ipynb`. **Note:** Your Notebook files will be stored automatically on your personal Google Drive, under a subfolder called `Colab Notebooks`.

2.1 Mount your personal Google Drive to Google Colab

You should mount your personal Google Drive to Colab using the following code inside your notebook:

```
#Mount your personal Google Drive to Colab
from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir('drive')
os.chdir('MyDrive')
os.listdir()
```

Note: This will require an small additional intermediate step where you manually have to approve the mounting (this is straightforward, just follow the instructions that automatically appear on the screen).

2.2 Create a course specific subfolder in Google Colab

You should also create a subfolder called SI_for_TA and make it your current working directory:

```
if not os.path.isdir("SI_for_TA"):  #create subfolder SI_for_TA does
  os.mkdir("SI_for_TA")             ## if it does not already exist
os.chdir("SI_for_TA") #move new subfolder=current working directory
os.getcwd()    #check which is the current working directory
```

In summary, this code makes subfolder SI_for_TA on your Google Drive your working directory.

2.3 Recommended videos to watch

Here are two suggested videos to watch related to Colab and Python programming, respectively.

Video about Colab and its notebook: https://youtu.be/inN8seMm7UI

Video about Python programming: https://www.youtube.com/watch?v=N4mEzFDjqtA

3 INTRODUCTION

NOTE: Always use numpy! In order to avoid difficulties and have a chance to get help form your teachers if needed, throughout this exercise you should always use **numpy** for generation of the random numbers needed! Here follows some code snippets that can be a helpful.

3.1 How to draw samples from an Urn Model using numpy

```
#RESAMPLING FROM URN USING NUMPY
import numpy as np

#Create urn
np_values=np.asarray([1,2,3])
np_counts=np.asarray([3,2,1])
np_urn=np.repeat(np_values, np_counts)
print(np_urn)
#Draw with replacement
N=6
np_draw_with_replacement=np.random.choice(np_urn, size=N, p=None, replace=True)
print("Draw with replacement",np_draw_with_replacement)
```

From this code you also get the tip that when you create a new variable, it is a good idea to have a prefix “np_ ” to indicate that the variable is a numpy array. Similarly, when/if needed, you could use “list_ ” as a prefix for lists , “pd_” for pandas variables, etc.

3.2 How to draw data from normal distributions using Numpy

Here is an example of how to generate N=10 examples from a univariate normal distribution.

```
#RESAMPLING FROM UNIVARIATE NORMAL USING NUMPY
import numpy as np
mu=0
sigma=1
N=10
np_onedimensional_normal_data=np.random.normal(mu, sigma, N)
```

Here is an example of how to generate N=10 examples from a bivariate (2-variable) normal distribution.

```
#RESAMPLING FROM MULTIVARIATE NORMAL USING NUMPY
import numpy as np
list_mu=[1, 2]
list_SIGMA= [[1, 0], [0, 1]]
N=10
np_2dimensional_normal_samples_as_rows= np.random.multivariate_normal(list_mu,list_SIGMA,N)
print("Data:",np_2dimensional_normal_samples_as_rows)
print("Datatype of np_2dimensional_normal_data_as_rows:",type(np_2dimensional_normal_samples_as_rows)
    )
print("Shape of np_2dimensional_normal_data_as_rows:",np_2dimensional_normal_samples_as_rows.shape)
```

4 PART A: Discrete stochastic variables - urn models

If you want to draw data from a probability distribution with N possible outcomes where the i:th outcome has probability p_i, then you can create an urn model (remember, using numpy!) containing balls of Ndifferent identities where the fractions of balls with identity i equals exactly p_i, see Fig.1.

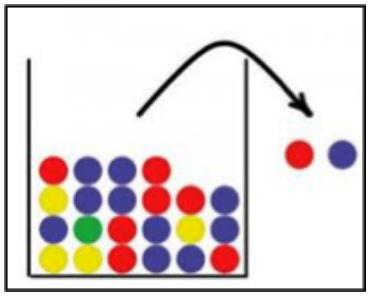


Figure 1: A draw from an urn model.

4.1 Exercise A:1 - Urn model of discrete probability distribution

Assume you want to draw 100 samples from the following distribution that corresponds to the three possible outcomes 1,2 or 3: $P(1) = 1\%$, $P(2) = 1.5\%$, $P(3) = 97.5\%$. Write Python code that simulates this. Solution: Create an urn consisting of 200 balls where two of them are of type 1, three of them are of type 2 and 195 of them are of type 3. Then draw 100 balls with replacement from this urn.

4.2 Exercise A:2 - Urn model for lab instrument processing times

A lab instrument for analysis and/or processing of items has a cassette that can take 3 items at a time. For each cassette, the instrument performs the analys/processing sequentially for the three samples inside. Assume that there are 5 type of samples that can arrive and that they have the following different analysis/processing times: 54, 66, 70, 72 and 97 minutes. Assume 3 of the 5 are selected randomly (with replacement) and put into the cassette. What is the probability that the analysis will not be completed after 200 minutes?

Remark. Since the samples are selected with replacement, it means that the cassette can be filled with 3 copies of the same type of samples.

Tip: Solve the problem by creating an urn that contains the 5 values and draw three values from the urn with replacement. Then calculate the resulting analysis time and perform this procedure in total 1000 times. Then determine the fraction of all the resulting analysis times that are longer than 200 minutes.

4.3 Exercise A:3 - Urn model of a pipetting robot with success rate 93%

Assume you are working with a pipetting robot that is claimed by the manufacturer have a success rate of 93% and that you are using it for liquid transferring to all wells in 96-well plates. What is the distribution of correctly filled wells in such 96-well plates? On average, there should be $0.93 \cdot 96 \approx 89$ successful wells in a 96-well plate, but this will of course vary due to random chance. In order to solve this problem, write your own Python code so that it can draw with replacement from an urn that contains 93red balls and 7 white balls. If you now draw 96 balls with replacement from this urn, you are simulate a pipetting experiment where the Robot is pipetting to 96 wells and where each red ball drawn corresponds to a correct pipetting event while each white ball drawn corresponds to a well where the pipetting failed. For example, if 92 of the drawn balls are red and 4 balls are white, this result corresponds to an experiment where the robot made 92 correct liquid transfers, while the pipetting failed in 4 wells.

Now simulate a case where the Robot is pipetting 1000 different 96-well plates. For each plate, determine the number of correct wells the Robot produced. Finally make a histogram of these numbers in such a way that you get a figure similar to histograms shown in Fig. 2.

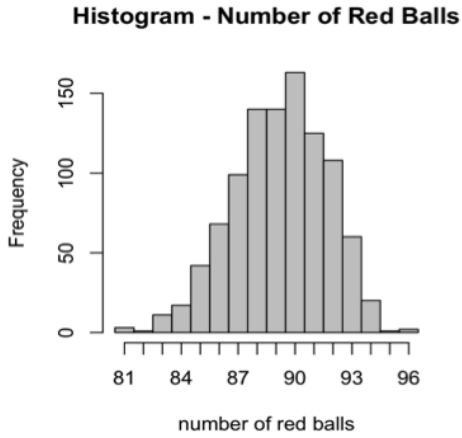


Figure 2: Histogram reflecting the number of correct pipetting attempts.

As expected, the average number of correct wells will be close to 89 (89 corresponds to 93% of 96). From this graph we can also see that if one performs a quality control experiment and finds that there are less than 80 successful wells, then the Robot success rate must be lower than 93%: This is because such an event is more or less impossible according to the histogram!

5 PART B: Real values stochastic variables - pdf models

When stochastic observations are real valued their distribution is described by a probability density function (pdf). In the following problems, some of the most commonly encountered pdfs will be considered, including how to draw samples from mixtures of pdfs. In addition, here the multivariate normal pdf is also studied.

Remark
Always keep in mind that there is a big difference between a pdf, which is a deterministic function, and the stochastic observations observed from such a pdf. Thus, if generate a very large number of observations from a pdf and use these observations to create a normalized histogram, then this normalized histogram will be a good approximation of the pdf from which the examples were drawn.

5.1 Exercise B:1 - Classical univariate pdfs

Here a few examples of pdfs commonly encountered in in many technological applications are considered.

Normal $N(\mu, \sigma)$
This is the classical Normal distribution:

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Examples of members from this family are shown in Fig. 3.

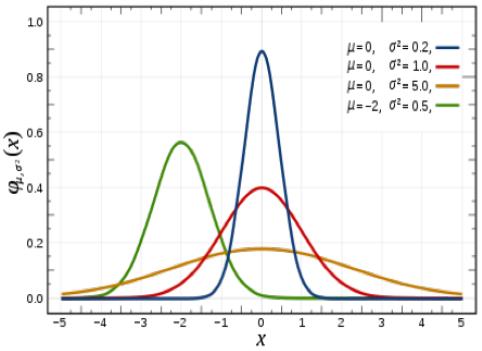


Figure 3: Examples of different normal pdfs.

Beta(α, β)
The beta-distribution is a pdf that only lives on the interval $[0, 1]$ and is therefore of relevance for example in various forms of survival analyses where the outcomes are values restricted to this interval.

$$f(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du}$$

Examples of members from this family are shown in Fig. 4.

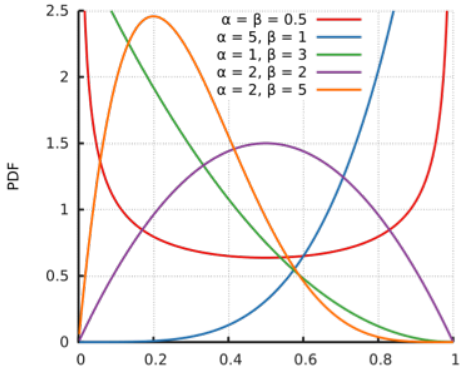


Figure 4: Examples of different beta pdfs.

(a) Use Python to make graphs showing the pdfs for the following stochastic variables, and for some other parameter selection of our choice (for example the ones used in the figures above):
$N(\mu = 0, \sigma = 2)$
$Beta(\alpha = 2, \beta = 5)$

(b) Use Python to generate 10 samples from the distributions above.

5.2 Exercise B:2 - Data from a mixture of multiple pdfs

Often observed values come from a mixture of underlying distributions.

Example 1: You are measuring the size (area) x of objects in images containing a mixture of different objects, like different manufactured products a conveyor belt, different human cell types in a test tube.

Example 2: You are measuring the effect x of a particular antibiotic in a population consisting of children, adults and and elderly.

Example 3: You measure vibrations from a population of machines that consists of 3 subtypes of machines.

In all these examples it is reasonable to assume the individuals belonging to one of the subpopulations are generating observations from a common pdf. When there is a mixture of underlying distributions, the observed distribution $p(x)$ of observed values x will be weighted sum of the different underlying pdfs. More specifically, if there are 3 different object types, each with its on distribution $p_i(x)$, then

$$p(x) = w_1 \cdot p_1(x) + w_2 \cdot p_2(x) + w_3 \cdot p_3(x)$$

where the weights w_i sum to one: $w_1 + w_2 + w_3 = 1$. Examples of members from this family are shown in Fig. 5.

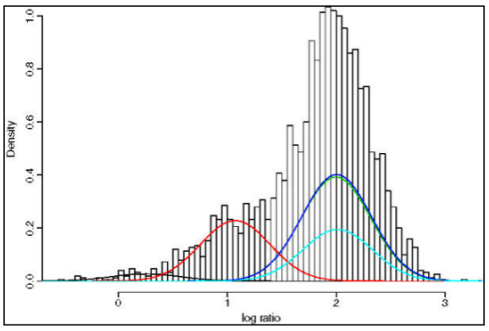


Figure 5: Examples of a mixture pdf and an associated histogram.

A 2-step algorithm: In order to draw a random samples from such a mixture of multiple pdfs, one has to run the following 2-step algorithm:

1. Draw one sample from a discrete 3 class distribution with probabilities (w_1, w_2, w_3).
2. If the outcome is class i, then draw a sample from $p_i(x)$.

Now consider the special case $p(x) = w_1 \cdot p_1(x) + w_2 \cdot p_2(x) + w_3 \cdot p_3(x)$ where $w_1 = w_2 = 0.05$, $w_3 = 0.9$ and $p_1(x) \sim N(\mu = -2, \sigma = 0.1)$, $p_2(x) \sim N(\mu = 2, \sigma = 0.1)$ and $p_3(x) \sim N(\mu = 0, \sigma = 1)$.

(a) Draw a graph showing the pdf $p(x)$ on the interval $[-3, 3]$.

(b) Draw 1000 samples from this distribution and then make and display normalized histogram, or a kernel density estimate, so that you can compare with the graph created in (a).

Tip: In order to draw integers 1,2,3 with probabilities $w_1 = w_2 = 0.05$, $w_3 = 0.9$, just generate a random number r from a uniform distribution on the interval $[0, 1]$ and choose 1 if $r \leq 0.05$, choose 2 if $0.05 \leq r \leq 0.10$, otherwise choose 3. Alternatively, first create an urn containing balls labelled 1,2 and 3 with the corresponding proportions $w_1 = w_2 = 0.05$, $w_3 = 0.9$ and then draw one ball randomly from it.

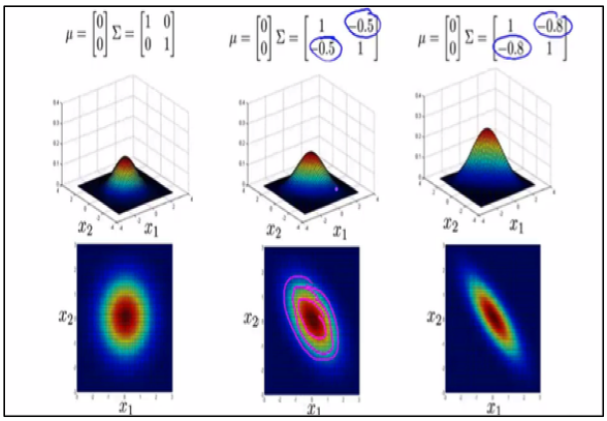
Remark
By means of an appropriately selected mixture distribution, one can approximate quite well almost any pdf of interest.

5.3 Exercise B:3 - Data from multivariate normal distributions

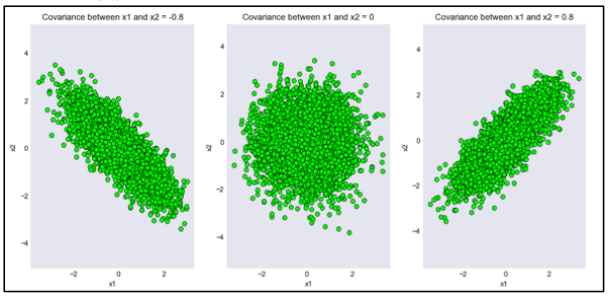
A multivariate normal distribution $N(\boldsymbol{\mu}, \Sigma)$ is defined by a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ. The pdf $f(\mathbf{x})$ for a d-dimensional vector $\mathbf{x}$ may be written :

$$f(\mathbf{x}) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{(2\pi)^{d/2}|\Sigma|^{1/2}} = \frac{\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}))}{\sqrt{(2\pi)^d \det(\Sigma)}}$$

Some aspects of this family of functions are illustrated for $d = 2$ dimensions in Fig. 6. You should pay attention to how the structure of the covariance matrix Σ is affecting the shape and orientation of the distribution.



(a) Two-dimensional normal distributions.



(b) Samples drawn from 2-dimensional normal distributions.

Figure 6: Illustrations of two-dimensional normal distributions.

(a) Generate 1000 samples from a bi-variate normal distribution with zero mean vector the following parameters in the covariance matrix: $\Sigma_{11} = 1$, $\Sigma_{12} = \Sigma_{21} = 0.7$, $\Sigma_{22} = 3$. Then make a 2D scatterplot to show the 1000 two-dimensional data points generated.

(b) Repeat the task in (a) but with the following parameter values: $\Sigma_{11} = 1$, $\Sigma_{12} = \Sigma_{21} = -0.7$, $\Sigma_{22} = 3$.

(c) Repeat the task in (a) but with the following parameter values: $\Sigma_{11} = 1$, $\Sigma_{12} = \Sigma_{21} = 0$, $\Sigma_{22} = 3$.

Remark.
(i) When the two variables x_1 and x_2 co-vary negatively (are negatively correlated), then the elongated cloud of points has a negative slope as illustrated in (b) above.
(ii) When the two variables x_1 and x_2 have covariance zero (uncorrelated), then the elongated cloud of points is aligned with the coordinate axes.

6 PART C: Markov and Hidden Markov Models: Discrete time and discrete states

Here your task is be to generate random sequences of states or symbols using Markov and Hidden Markov Models.

In the following problems, you will get an idea about how Markov models can be used to describe and simulate the stochastic dynamics of the cell cycle of a living biological cell and how Hidden Markov models can be used to describe and simulate families of DNA (and Protein) sequences. First, Figures 7 and 8 show some other examples of Markov Models found in other application areas related to industrial/financial analyses.

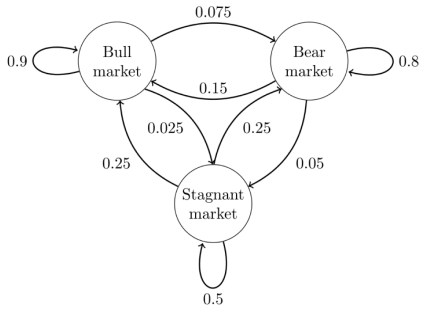


Figure 7: A Markov model simulating transitions between different markets (Bull, Bear, Stagnant).

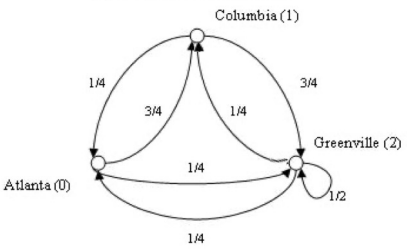


Figure 8: A Markov model simulating a maintenance person or an expensive instrument having to travel between different cities due to randomly occurring demands.

6.1 Exercise C:1 - State sequence from a 3-state cell cycle Markov Model

(a) The task here is to simulate data that reflects the stochastic process of living biological cells, that are randomly switching between 3 main different states G, M and A. More specifically, the task is to write Python code to simulate 2000 steps from the following 3-state Markov model, where state G corresponds to growth, state M to mitosis, and state A to cell-cycle arrest, respectively. The simulation should start from state G.

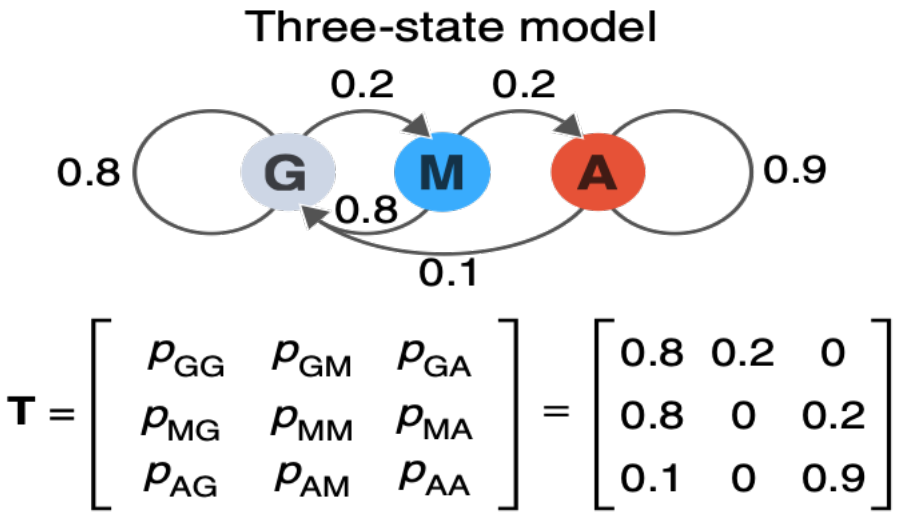


Figure 9: A 3-state Markov model having the states G, M, and A. The associated transition probabilities are indicated next to the arrows in the upper part of the picture. The lower part shows the transition matrix T, which contains the transition probabilities P_{XY} of jumping from state X to state Y.

This should result in a list consisting of 2000 state values, for example starting as

```
G,G,G,M,G,G,G,G,M,A,A,A,A,A,A,G,G,G,M,...
```

Tip : When in state S, draw a random number r from a uniform distribution on the unit interval $[0, 1]$. Then make a jump accordingly. For example, when S=G stay in state G if $r \leq 0.8$, otherwise move to state M (if $r \geq 0.8$). Alternatively, create an urn for each state that reflects the transition probabilites related to the next move (which for state S_i corresponds to row i in the transition matrix T).

(b) Make a normalized histogram using the last 1000 states of the sequence generated and confirm that the resulting probabilities agree quite well with the theoretical limiting probability distribution $[P_G, P_M, P_A] = [0.625, 0.125, 0.25]$. These are the probabilites of observing each state if one runs the Markov Model for very long and then randomly sample one observation (whcih will be the current state) from it.

6.2 Exercise C:2 - Generate symbol sequences from a Hidden Markov Model

Starting in the start state S0, use Python code to simulate 10 realizations from the Hidden Markov model shown in Fig. 10.

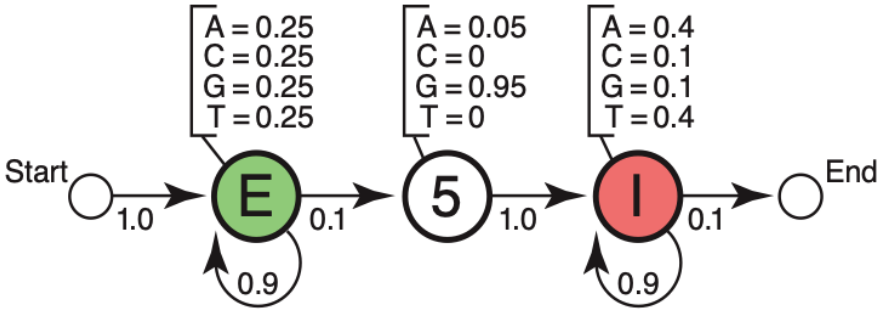


Figure 10: A hidden Markov model with the five states $S_0 = \text{Start}$, $S_1 = \text{E}$, $S_2 = 5$, $S_3 = \text{I}$ and $S_4 = \text{End}$ and four emitting symbols A, C, G and T having different emission probabilities for each state. The state transition probabilities and the emission probabilities are written out explicitly.

Thus you should get 10 lists resembling the following:

```
C,T,T,C,A,T,G,T,G,A,A,A,G,C,A,G,...
```

Remark #1.
For a given set of parameter values (transition probabilities and emission probabilities per state), a HMM defines a family/distribution of sequences. Such a family can for example approximate:
(i) A family of sound/microphone sequences obtained for a particular word expressed by a set of different persons. Notably, most of us have different accents and spend more/less time on different sounds when pronouncing a single word.
(ii) A family of DNA sequences obtained for a particular gene across different species.
(iii) A family of amino acid sequences for a particular family of proteins.
(iv) A family of sound/microphone sequences obtained for a particular mechanical machine failure generated by a set of different industrial machines or industrial drilling equipments.
(v) A family of interest rate sequences obtained for a particular market transition model (which describes random transitions/jumps between bear, bull and stagnant markets).
(vi) A family of repair cost sequences for a maintenance person traveling between different cities/factories due to randomly occuring demands.

Remark #2.
All these Markov Models and Hidden Markov Models obviously have many parameters that have to be selected before the simulation. Using the Maximum Likelihood method and similar methods, it is possible to estimate these parameters from a set of training examples in the form of time series data collected. Intuitively, the idea with the parameter estimation is to find the particular parameter values that make the stochastic model of interest generate new sequences being ”most similar” to the already collected data.

Using such approaches, it is possible to use a set of observed sequences (often called training examples/sequences) to find Markov Models and Hidden Markov Models with reasonable parameters, provided that the number of observed sequences are sufficiently many. Two key questions in this context are: (1) How uncertain will we be about the true parameter values (assuming that the real world can be described perfectly by such a model)? (2) How well does new data generated with the parameters selected agree with independent test examples generated from the same source as the training examples.

7 PART D: Linear relationships embedded in measurement noise

Here your task is to simulate data from linear relationships and then add experimental noise to them. This is in order to be reminded and understand that observations always come with experimental noise/-variability. Therefore, even if there is a perfect underlying linear functional relationship between the variables studied, it will not be a perfect linear relationship between the actual values observed, and usually there is experimental variability associated with both variables studied.

7.1 Exercise D:1 - Linear relationship embedded in experimental noise for both x and y

(a) Simulate 10 examples $(x(n), y(n))$ from the following model where $x(n)$ is drawn from a uniform distribution on the unit interval $[0, 1]$ and $y(n)$ is determined as

$$y(n) = 2x(n) + 4$$

(b) Simulate 10 examples $(x(n), y(n))$ as above, but in addition, also simulate that both $y(n)$ and $x(n)$ are observed in additive noise so that the observed pair $(x_{OBS}(n), y_{OBS}(n))$ is

$$\begin{cases} x_{OBS}(n) = x(n) + e_x(n) \\ y_{OBS}(n) = y(n) + e_y(n) \end{cases}$$

where $e_x(n)$ and $e_y(n)$ are random numbers drawn from $N(\mu = 0, \sigma = 0.3)$.

NOTE: For each observation $(x_{OBS}(n), y_{OBS}(n))$, one has to generate a new pair of random numbers $(e_x(n), e_y(n))$, in order to simulate a unique pair of random measurement errors, which only occurs at the particular time point n.

8 DEADLINE AND EXAMINATION

This examination will consists of the following parts:

(1) Answers and Results: Upload you Colab code to Studium and prepare for discussing your code during the mandatory seminar. Deadline: You must **submit before the first seminar** that belong to this assignment. Thus if there are 3 seminars where this assignment will be discussed, you must submit before the first one, even if you join a later session. This is in order to be fair and not having everyone coming to the last session.

NOTE: If you are working in pairs, each person has to submit their own document, even if it is almost identical. You are of course always responsible for your own submission and must be able to explain all of it during seminars and similar examinations.

(2) Python code as one single Colab notebook: The notebooks should be ready to run in Google Colab and it should be submitted after running it so that results (printouts and figures) are visible when opening the notebook. If your code consists of more than one notebook (and/or some additional code called from inside the notebook) your instructions how to run the code should be provided as a text file called `readme.txt`. Make sure that all files you are submitting have your first name and surname as part of the file name, like:

```
Assignment_1_Mats_Gustafsson.ipynb
```

(3) Personal reflections and feedback as pdf: An uploaded pdf document to Studium providing personal reflection and feedback:

```
Assignment_1_reflections_Mats_Gustafsson.pdf
```

This pdf document should contain reflective answers to the following questions:
(a) Did you have sufficient background knowledge to perform this exercise? If not, what background was missing?
(b) Make sure you understand that a pdf may have (but not always have) values that are lager than one.
(c) Make sure you understand and remember how to draw a sample from a pdf that is written as a weighted sum of at least two other pdfs, for example a so called Gaussian mixture.
(d) Make sure you understand that when $f()$ is a multivariate normal pdf, the function value $f(\mathbf{x})$ is a scalar but that any observation $\mathbf{x}$ from this pdf is a vector.
(e) What did you personally find interesting and/or useful an/or informative with the assignment, and why? If you actually did not find it enlightening, in that case we would like to know about why.
(f) What did you find frustrating and/or difficult, and/or unnecessary, and why?
(g) What would you suggest as an improvement of this assignment for next year, and why?
(h) Did you spend more than 20 hours to complete this exercise? Why?

NOTE: If you are working in pairs, each person has to send in a personal reflection.

(4) Mandatory Seminar: You will be discussing/presenting your results during the associated mandatory seminar. You do NOT have to prepare any separate slide presentation, it will be fine to show your Colab notebook if/when you are requested to present something.