Digital Skills

Christos Dimitrakakis

October 14, 2022

Contents

1 Introduction

This is a hands-on course that integrates introductory programming, statistics and data science. Through out this course, we will formulate scientific hypotheses, design experiments, and collect and analyse data visually and through formal models. You are expected to supplement this course with homework, self-study and other courses in descriptive statistics and python programming, but no prior knowledge is assumed. Formal concepts will be introduced through class activities and examples.

The course will focus neither on statistical theory nor on programming. We will only lightly build those skills within the course. Our main aim will

be to build scientific and statistical intuition through practical work. However, to achieve this you need to be able to independently pick up theoretical and programming skills. For that reason, it is necessary for you to do outside reading either (ideally) by taking statistics and programming courses in parallel, or through self-study.

1.1 Learning goals:

Graphical comprehension

1. Recognise structural elements in a statistical graph (e.g. axis, symbols, labels) and evaluate the effectiveness (for perception and judgment) and appropriateness (for the type of data) of structural element.
2. Translate relationships reflected in a graph to the data represented.
3. Recognise when one graph is more useful than another and organise/reorganise data to make an alternative representation.
4. Use context to make sense of what is presented in a graph and avoid reading too much into any relationships observed.
5. Express creative thinking via the production of an innovative graphical presentation.

Scientific process

1. Understanding the randomness, variability and uncertainty inherent in a problem.
2. Developing clear statements of the problem/scientific research question; understanding the purpose of the answer.
3. Be able to perform a basic experiment design.
4. Identify sources of bias in data collection and analysis.
5. Ensuring acquisition of high-quality data and not just a lot of numbers.
6. Understanding the process that produced the data, to provide proper context for analysis.
7. Allowing domain knowledge to guide both data collection and analysis.
8. Quantify uncertainty—and knowledge—visually.

9. Realise that all visualisations are model summaries.
10. Be able to write simple python programs for data science workflows.

1.2 Administration

1. Make sure you are registered on IS-Academia
2. Also register on Moodle: this is where the assignments will be
3. Clone this git repository

1.3 Assessment

The assessment is purely through in-class exercises, quizzes and homework assignments. There will be assignments spread over the semester, as well as a group project. The project will be performed in pairs.

For all assignment and the project, the following rubrik is used. Some of the assignments may not involve all parts.

Experiment design. The first stage any project, no matter how small, is the experiment design and analysis. This includes a plan for how to collect data, methodologies for analysing the data, and the development of a pipeline, preferrably in the form of a program, for collecting data and analysing it. In additional, the experiment design must be reproducible: This can be ensured by running the data collection and analysis pipeline on simulated data, and seeing if the results are as expected.

Computation. Here you must instantiate the experiment design and analysis with concrete computations. For reproducibility, the computations you perform should be independent of the data you actually have. Correctness of the computations is the most important aspect, here. However, you should also take care to document why and how how you are doing the computations.

Graphics. This addresses the creation of visualisations of your analysis. It is recommended to do this fully automatically, so that you can simply run your pipeline and get all the results you need. Be sure to quantify uncertainty.

Text. Here you should explain in text what the graphics mean. Point out any interesting things you can see in the visualisation and try to explain it. Do not be overconfident, but quantify uncertainty properly.

Synthesis. Here you should summarise the most important findings from your analysis. Be careful to not over-interpret your results. A lot of

results can be imaginary and can be attributed to insufficient data, biased sampling, improper modelling or p-value hacking. Again, be sure to quantify uncertainty.

Skill	Needs Improvement	Basic Level	Advanced Level
Experiment design: Data collection and analysis pipeline	Inappropriate sampling, non-reproducible analysis	Data collection biased or analysis not reproducibile.	Unbiased sampling and reproducibile experiment desgin and analysis.
Computation: Perform computations	Computations contain errors and extraneous code	Computations are correct but contain extraneous/unnecessary code.	Computations are correct and properly justified and explained.
Graphics: Communicate findings graphically clearly, precisely and concisely	Inappropriate choice of plots; poorly labelled plots; plots missing.	Plots convey information correctly but lack context for interpretation.	Plots convey information correctly with adequate and appropriate information
Text: Communicate findings clearly, precisely and concisely	Explanation is illogical, incorrect or incoherent.	Explanation is partially correct but incomplete or unconvincing	Explanation is correct, complete and convincing.
Synthesis: Identify key features of the analysiand interpret results	Conclusions are missing, incorrect, or not made based on analysis	Conclsions reasonable, but partially correct or incomplete.	Relevant conclusions explicitly connected to analysis and context.

Pass: All parts must be addressed, the 'default' grade is 75%. 5% is added for every 'advanced' skill and removed for every 'needs improvement skill'. Thus the passing grades are 50-100%.

Fail: If not all parts are explicitly addresed, the assignment is failed.

1.4 Data sources

This course will consider the following data sources in order of importance.

1.4.1 Synthetic data

This data is obtained through simulation, and it is useful in order to test whether a particular pipeline is working as intended. In particular, it is a great way to test the performance of a method as you vary the data generation process so that different assumptions are satisfied. This allows you to verify robustness.

1.4.2 UCI machine learning repository

The UCI repository has a large collection of datasets in an easy to access format. These have already been used in many academic papers, and are a

good starting point for you to look at real data. All the data is formatted in an easy-to-use some format, but some pre-processing may still be necessary.

1.4.3 Wikipedia and newspaper articles

Wikipedia has many interesting articles, from which you can extra tabular data, as well as more contextual information. It is possible to also discuss newspaper articles. Wikipedia and newspaper articles can be used in the context of some assignments.

1.4.4 Economics data

- FRED: Federal Reserve Economic Data
- OECD: Organisation for Economic Co-operation and Development

1.4.5 NASA Mars Challenge

`https://www.drivendata.org/competitions/97/nasa-mars-gcms/`

2 Module 1: Visualisation as models and data summary (4 weeks)

What is visualisation? It is a way to *summarise data*. It is also a way to view relationships between variables. Visualisation helps us to find patterns and understand the underlying laws behind how the data was generated. This is, in fact, the essence of modelling.

A model is *also* a way of summarising the essential features of the data. A visualisation differs from a model only in one sense: It easy to interpret visually.

Every data visualisation implicitly assumes a model of the data generating process. This is true for even the simplest visualisations, like histograms. There is no escape from the fact that any visualisation makes a lot of assumptions. We must emphasize what those assumptions are. What happens if they are not true?

Every data visusalisation, then, proceeds in three steps:

1. Data transformation
2. Model creation

3. Model visualisation

Parameters. Every model is defined by a number of parameters. This is what is displayed when we visualise data. You can think of the model as the underlying theory, and the visualisation as a way to explain the theory visually.

2.1 Histograms: model a distribution

Histograms are a simple tool for modelling distributions. In their simplest application, they are used to simply count the number of items in distinct bins of a dataset. While typically employed to represent the empirical distribution of one-dimensional variables, they can be generalised to multiple dimensions .

2.1.1 Bar graph activity

1. All students who are male raise their hand
2. All students who are female raise their hand.
3. We count, and draw a bar graph: the number of male, female and other students.
4. We count how many are in the BSc of DS of those
5. We also count how many are taking a programming course
6. We also count how many are taking a maths/stats course
7. What does the graph tell us about:
   (a) Computer Science students
   (b) Students at Neuchatel
   (c) Residents of Neuchatel
   (d) Other subsets of the population
8. Can we make a similar graph when measuring a continuous variable?

1. Definition of a bar graph THEORY

   Consider a set of k categories $C = \{1, \ldots, k\}$. Every individual belongs in one category. This can be defined through a function f assigning categories to individuals.

In particular, imagine a dataset of individuals D. There exists a membership function

$$f : D \to C$$

so that $f(x)$ is the category to which individual $x \in D$ belongs.

A bar graph is a visual representation of the following count vector,

$$n_c(D) = \sum_{i \in C} \mathbb{I}\{f(i) = c\},$$

where $\mathbb{I}$ is an indicator function:

$$\mathbb{I}\{A\} = \begin{cases} 1, & A \text{ is true} \\ 0, & A \text{ is false,} \end{cases}$$

so that the height of the c-th bar is proportional to n_c.

2.1.2 Introduction to histograms THEORY:STATISTICS

Assume data is in $\mathbb{R}$. Then split the real line into intervals $[a_i, b_i)$. For a given dataset D, for each interval i, count the amount of data $n_i(D)$ in the interval. We can also normalise to obtain $p_i(D) = n_i(D)/\sum_j n_i(D)$

More generlaly, a (counting) histogram is defined as a collection of disjoint sets called **bins**

$\{A_i | i = 1, \ldots, k\}$

with associated counts n_i, so that, given some data D,

$n_i(D) = \sum_{x \in D} \mathbb{I}[x \in A_i]$,

where n_i is the number of datapoints in A_i. Typically $A_i \subset R$.

We can use the histogram as the model of a distribution. For that, we use the relative frequency of points in each bin: $p_i(D) = n_i(D)/\sum_j n_j(D)$. The selection of bins influences the model.

See also: `https://en.wikipedia.org/wiki/Histogram`

1. Histogram activity ACTIVITY

   (a) Introduce the concept of a histogram on the board.

   (b) Split the students in two groups.

   (c) Have each group collect the height of every student.

   (d) How can we summarise the data of each group?

   (e) Now the students will individually draw a histogram from the data of their group.

(f) Show two different histograms from two people in the same group. Why are they different? Discuss in pairs and then in class.

(g) Now show a histogram from a person in another group. Why are the histograms in the two groups different? Discuss.

(h) Collect the data of all students in the online excel file.

(i) Now we shall plot a histogram of the students using the sheet. How does that differ?

[If there are not enough students, the exercise can be performed by adding random numbers using dice]

2. Measuring a discrete distribution ACTIVITY

(a) Toss a coin 10 times and record each one of the results, e.g. {0,1,1,0,0,0,0,1,1,1}.

(b) Count the number of times it comes heads or tails.

(c) We then summarise the result.

Let us denote the number of times you have heads by $N(x = k)$. It should be approximately true that $N(x = k) \approx 5$, however, this may not be true for everybody.

We can visualise this by plotting bars or lines, whose height is proportional to $N(x = k)$.

We typically assume that individual coin tosses are generated from The Bernoulli distribution. This means that the probability of heads or tails is fixed, and does not depend on the result of the previous tosses. Why might not that be the case?

If individual tosses are Bernoulli, then the distribution of the number of heads (or tails) is a binomial distribution.

We will now show how to achieve the same results programmatically.

2.1.3 General python help CODE:PYTHON

To help yourself understand python, you can always take a look at the documentation in English or French. To start with, check out the Tutorial. Then use the library reference for advanced usage.

Sometimes it is quicker to just use the **help** command in the python console or the **?** command in the jupyter notebook.

Python can be used as a simple calculator

```
$ python3
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> 2 * 3 + 1
7
>>> 2 * (3 + 1)
8
>>> exit()
```

This interactive console is the most usual way of playing with python in the beginning, but it is not useful in general.

2.1.4 A simple program

Python programs are executed one statement at a time. Statements are separated by newlines. Anything appearing after a # symbol is not executed.

```
print("Hello world") # first statement, with a comment
print("Goodbye, world.") # second statement!
# print("This is not printed") - it is a comment, you see
```

Python programs can be generate text, write and read from files, access the internet, generate and display or save plots to disc, play and record music, record images from a camera.....

Before we actually run the program, one of you can play the role of the python **interpreter**. The interpreter goes through each line of the program, interprets it and executes it.

When we execute a program in the console, we are assuming the role of the intpreter that steps through the program.

Run the above statements in the python interpreter

```
$ python3
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello world")
Hello world
>>> print ("Goodbye, world")
Goodbye, world
```

```
>>> exit()
$
```

The statements print() and exit() are called functions. Function names must always be used with parenthesis. The contents of the parentheses are called **arguments**. Changing the arguments to a function has a different effect.

We can now try and save the above commands in a file called "pythontest.py" and execute it via

```
python3 pythontest.py
```

2.1.5 Consoles, Scripts and Notebooks

Console input is used when you want to have a purely interactive session to test something

Script files are used to save your work and re-run it. They also allow you to build complex programs from multiple files, where each file has a different functionality.

Notebooks are something in between. They are script files with interactive output, and are very useful for rapid development and testing. They also save their state and output in between runs, so they help to document your code. We will use them a lot in class. There are two methods to use notebooks:

- Locally through e.g. jupyter-lab or jupyter-notebook
- Online through `https://colab.research.google.com/` `https://replit.com` or `https://noto.epfl.ch`

Most of the time, you want to be saving the code you write in a script file and executing it, instead of using the console. However, sometimes the interactivity of the console is helpful. This is when notebooks are used. After you are done developing something with the notebook, you can then extract what you need in a simple python script.

2.1.6 Python variables CODE

The python interpreter has a **state**. This includes the contents of a memory where variables are stored, and the current location of the code pointer, that is which line will be executed next.

Variables are alphanumeric references to simple or complex objects. Possible variable names:

- X
- NumberOfApples
- salary
- $\text{scratch}_{\text{variable}}$
- y2

A variable can be assigned a value with the = operator.

```
x = 2 # this gives the numeric value of '2' to the variable
```

Variables must be defined before they can be used for the first time

```
>>> x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>> x = 2
>>> x # typing the name of a variable in the console gives you its value
2
```

Numerical Python variables are very simple entities. Let us go through this is easy program for a warm-up.

- x=value; assigns a value to a variable named x
- print(); displays something in the terminal

```
x = 1 # a variable
y = 2 # another variable
print(x+y) # print the value of this variable sum
x = y # assignment operation: now x has the same value as y
print(x) #what would this value be?
y = 3
print(x) #is x changed?
```

2.1.7 Python lists CODE

A slightly more compex object are python lists. A list can contain anything, and is so very flexible. It can contain numbers, strings, or arbitrary 'objects'.

Now check out the first part of the Histogram example. For that we need one line of setup so we can plot stuff.

```
import matplotlib.pyplot as plt # this is used for plotting
X=[0,1,0,0,0,0,1];  # list of coin tosses
plt.hist(X) # plot a histogram - this automatically splits everything into bins
```

In reality, the histogram function creates a so-called bar plot

```
import matplotlib.pyplot as plt # this is used for plotting
X=[0,1,0,0,0,0,1];  # list of coin tosses
plt.bar(["heads","tails"], [sum(X), len(X) - sum(X)]) # do a bar plot!
```

The following source creates a list of four numbers and returns one element. Things to unpack here:

- **x[i]** returns the **(i+1)-th** element: we start counting **from 0**
- the **return** statement sends a value back to the whatever started the python program: in this case this .org file.

```
x = [1, 2, 3, 4]
return x[3] # returns the last element of the list
```

The following program assigns arrays-values to variables. Now x, y are both lists.

```
x = [1, 2, 3, 4]
y = [-1, -2]
x = y # assignment operation: now x is just a different name for y
y[0] = 1 # modify the 0th element of y
return x # what would the value of x be?
```

Lists are different in one respect: when we assign one list name to another, this does not copy any data. Both names refer to the same data. Consequently, if we change the data, it changes for both variable names. The way to avoid that is to use the **copy()** function.

```
x = [1, 2, 3, 4]
y = [-1, -2]
x = y.copy() # copy operation: now x has a copy of y's data
y[0] = 1 # modify the 0th element of y
return x # what would the value of x be?
```

2.1.8 Numpy arrays ADVANCED:CODE

Because lists are very flexible, they are a bit slow. A special type of object, an array, is used to handle lists of numbers. This is not defined in basic python, but only in one module called *numpy*. Even though basic Python has only a few commands, it has many modules that extend the language to perform complex tasks without having to code everything from scratch.

```
import numpy as np
x = np.array([1, 2, 3, 4])
y = np.array([-1, -2])
x = y # assignment operation:
y[0] = 1
return x
```

2.1.9 Pandas and Histograms PLOT:CODE

For this, we work on the Histogram example.

Pandas is a module for simple and efficient data I/O processing and visualisation. The following code snippet demonstrates a couple of features.

```
import pandas as pd # we need to load a library first
# loading data into pandas creates a data frame df
df['column-name'] # selects a column
df.hist() # creates a plot with many histograms
```

1. Coin example ACTIVITY:PLOT

   Plotting is also possible through the matplotlib. This is the module that pandas uses to plot stuff. It just has a simpler interface for doing so. But if you want to create custom plots, matplotlib is what you need to use.

   ```
   X = [1, 0, 1, 0, 1, 1, 0, 1, 0] # a sequence of coin tosses.
   import matplotlib.pyplot as plt # python has no default plot function, we must IMP
   plt.hist(X) # this function plots the histogram
   ```

   Each one of you should predict the result of a number of coin tosses. Let us do a histogram of the predictions. This is a binomial distribution.

   (a) The students record their data in the shared spreadsheet

   (b) Firstly, plot the histogram of the data with default settings.

(c) What is the eff

Let us look at the student data: see src/histograms/heights.ipynb

2. Heights example ACTIVITY

```
import pandas as pd
X = pd.read_csv("class-data.csv") # read the data into a DataFrame
X['Height (cm)'].hist() #directly plot the histogram
```

2.1.10 Histograms vs Pie Charts

While histograms are good visualisations of distributions on the real line, distributions over a discrete set of possible values are best-represented by a pie-chart. This especially if there is no relation between the different values. As an example, if the values are distinct categories, there is no particular reason to order them on an axis.

- What are the advantages and disadvantages of pie charts and histograms?

	Histogram	Pie Chart
To show proportions		
For more categories		
To compare relative size		
For real-valued data		

- Why is a 3D pie chart never a good idea?

  #+BEGIN$_{\text{SRC}}$: python plt.pie(counts) # plot counts #+END$_{\text{SRC}}$

2.1.11 Randomness CODE:ACTIVITY

Random algorithms using coins.

```
y = 0 # y is a variable, with the value zero currently
import numpy as np # this library has many useful functions
x = np.random.choice(100) # x takes values 'randomly'. It is a 'random variable'.
return x # let's see what value it takes
```

Uncertainty vs randomness: coin-flipping experiment

1. Everybody flips a coin 10 times.

2. Record each throw with 0, 1 in this spreadsheet: `https://docs.google.com/spreadsheets/d/1E4bs05HnKXf1GZe4g3v6RLnHsj-YcaWg3Qe_RQyfhHU/edit?usp=sharing`
3. Then record how you threw the coin and what coin it was.
4. Discuss if the coin is really random.
5. What is the distribution of coin throws for the first throw?
6. What is the distribution of recorded coin biases? Why do some coins appear more biased than others?
7. Does it make sense to aggregate all the results? What does that assume?

In the context of experiment design and data analysis, it is very common to have conditions like those in this example. Even though we wish there was such a thing as the 'repeated experiment', in practice it never is repeated. There is always some varying factor.

Pseudo-random numbers

Let us now repeat the experiment with data generated via a computer.

```
# here is a default way to generate 'random' numbers
import random
X = random.choices([0, 1], k=10) # uniformly choose 10 times between 0 and 1.
plt.hist(X) # everytime we run these commands, we get a different proportion
```

This python code is completely deterministic. A complicated calculation is used to generate the next 'random' number from the previous one. Consider this example:

```
import random
seed(5) #this sets the 'state' of the random number generating machine
print(random.uniform(0,1)) # the random number is a function of the state
print(random.uniform(0,1)) # the state changes after we generate a new number
print(random.uniform(0,1))
seed(5) # when we reset the state, we get the same sequence of numbers
print(random.uniform(0,1)) #
print(random.uniform(0,1))
print(random.uniform(0,1))
#+END_SRC python
```

```
For cryptographically strong random numbers you need to use the secrets module:
#+BEGIN_SRC python
import secrets
secrets.choice(range(100))
```

Physical sources of randomness

Let's go back to throwing coins now. Coins are completely deterministic. Whenever we have a specific coin to throw in the air, there are two things we do not know. The first is which side the coin will land on. Why is that? The second is uncertainty about the coin bias: is the probability of landing heads exactly 50%? How can we quantify this? What does it depend on? Discuss in class.

What physical source of randomness can we use instead of coins?

2.1.12 Uncertainty ACTIVITY

Probability is not only used to model random events. In fact, almost nothing can be said to be really random, unless we go into quantum physics. Even a die thrown in the air follows precise mechanical laws. Given enough information, it is possible to accurately predict the outcome of a throw.

For that reason, probability is best thought of as a way to model any residual uncertainty we have about an event. Then the probability of an event is simply a subjective measure of the likelihood.

While probability offers a nice mathematical formulation of uncertainty, when this uncertainty is subjective, the question arises: how can we elicit precise probabilities about uncertain events from individuals? Here is an example.

1. The number of immigrants ACTIVITY Consider the following question: how many immigrants live in Switzerland?

   (a) In-class discussion: what do we mean by that?

   (b) Now everybody can make a guess and record it on this form: `https://moodle.unine.ch/mod/evoting/view.php?id=295622`

   What does this distribution mean? Can we use it as an estimate of uncertainty?

   (a) Now let us create some confidence intervals. The procedure is as

follows. Let us take a first guess at an inteval, (say 5-10%) and ask: (a) Are you willing to take an even bet that the true number is between [5-10%]?

2.1.13 Probability background THEORY

The theory of probability is used to mathematically define processes with uncertain outcomes. The set of all possible outcomes depends on the process. For example, if we throw a die, this is the set of all possible ways, locations etc that the die can fall and land. However, we may only be interested in two events: whether the die lands showing a '6' or not. Formally, an event A is a subset of all the possible outcomes Ω. In our example, A can be the set of all ways in which the die can land so that its top shows "6". A probability measure P simply assigns a number between 0 and 1 to every subset A we might be interested in. This can be thought of as the area of A. Different probability measures P assign different areas to different sets.

For some more technical details, see Probability space.

See also:

- `https://en.wikipedia.org/wiki/Probability`

1. Probability space

   In probability theory, we typically define the set of all possible events that we care about as the algebra Σ, so that any possible event $A \in \Sigma$ and so that $A \subset \Omega$. The algebra has the property that it is closed under union and complement, that is:

   (a) If $A, B \in \Sigma$ then $A \cup B \in \Sigma$

   (b) If $A \in \Sigma$ then $\neg A \in \Sigma$.

   Here, $\neg A \triangleq \Omega \setminus A$, i.e. the subset of Ω not containing A.

   Together with a probability measure P, the tuple (Ω, Σ, P) defines a probability space. Simply put, $P(A)$ is the probability that event A happens.

2. Probability measures THEORY:PROBABILITY A probability measure P is a function from sets to the interval $[0, 1]$. Measuring the probability of a set is technically the same as measuring the area of a region, or the number of items in a given region. Formally, for a probability measure is defined on:

- A **universe** Ω of outcomes
- The **algebra** Σ of subsets of Ω (which we can think of as all the 'events' of interest) so that:

(a) If $A \in \Sigma$, then $A \subset \Omega$ (b) If $A, B \in \Sigma$ then $A \cup B \in \Sigma$. (b) If $A \in \Sigma$ then $\Omega \setminus A \in \Sigma$.

The axioms of probability A probability measure $P : \Sigma \to [0, 1]$ on Ω satisfies the following axioms:

(a) $P(\Omega) = 1$.

(b) If $A \cap B = \emptyset$ then $P(A \cup B) = P(A) + P(B)$.

From these, it also follows that $P(\emptyset) = 0$.

See also: `https://en.wikipedia.org/wiki/Probability_measure`

2.1.14 Example Distributions

We focus on distributions where there is a finite number of possible outcomes, and hence a finite number of possible events that we might care about. All such distributions are characterised by one or more *parameters*. The simplest such distribution is a distribution on only two outcomes, the family of Bernoulli distributions.

1. The Bernoulli distribution. DEFINITION:PROBABILITY Let us start with a simple example, the Bernoulli distribution with parameter $\theta \in [0, 1]$. This is the distribution over two outcomes $\{0, 1\}$, so that if x is a Bernoulli random variable, then:

$$\mathbb{P}(x = 1) = \theta, \qquad \mathbb{P}(x = 0) = 1 - \theta.$$

It is typical to think the distribution of heads and tails of a coin as being Bernoulli, with parameter $\theta = 1/2$.

Probability space Formally, if the underlying probability space is (Ω, Σ, P), with random outcomes $\omega \in \Omega$ and random variable $x : \Omega \to \{0, 1\}$ then

$$\mathbb{P}(x = 1) = P(\{\omega : x(\omega) = 1\}).$$

See also: `https://en.wikipedia.org/wiki/Bernoulli_distribution`

2. Binomial distribution DEFINITION:PROBABILITY If we repeat a Bernouli trial, we can also count the number of times the coin comes heads. The distribution of the counts is called the Binomial distribution. If y is a binomial random variable for n throws with parameter θ, then we can write it as the sum of n Bernoulli random variables $x_1, \ldots, x_n$, i.e.:

$$y = \sum_{t=1}^{n} x_t.$$

The probability of k heads after n throws is given by the formula:

$$\mathbb{P}(y = k) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

, where $\binom{n}{k}$ is the bimomial coefficient.

See also: `https://en.wikipedia.org/wiki/Binomial_distribution`

3. The Categorical/Multinomial distribution

A multinomial distribution is an extension of the Bernoulli and binomial distributions to $m \geq 2$ outcomes.

Categorial distributions Let us start with one trial, e.g. a single throw of a die. We can model this dice throw as the distribution where the probability that the die lands with its k-th face on top is θ_k,

$$\mathbb{P}(x = k) = \theta_k.$$

Thus, this distribution is parametrised by the vector $\theta = (\theta_1, \ldots, \theta_m)$. A random variable $x : \Omega \to \{1, \ldots, m\}$ obeying this distribution is called multinomial.

If the underlying probability space is (Ω, Σ, P), then

$$\mathbb{P}(x = k) = P(\{\omega : x(\omega) = k\})$$

See also: `https://en.wikipedia.org/wiki/Multinomial_distribution`

4. Uniform distributions

A special case of binomial and multinomial distributions is the uniform distribution. This is defined as follows.

Let $|A|$ be the size of a set A. Then a distribution P is uniform if it obeys

$$P(A) = \frac{|A|}{|\Omega|}.$$

This definition applies to continuous distributions as well. A standard example is the uniform distribution on the interval $[0, 1)$. Then the probability that we obtain an outcome in the set $[0, p)$ is always equal to p, i.e.

$$P([0, p)) = \mathbb{P}(\omega \in [0, p)) = p.$$

5. Random variables THEORY:PROBABILITY

   A real-valued random variable $f : \Omega \to \mathbb{R}$ is simply a function from the outcomes to the real numbers. Even though it is a fixed function, its values are random, because the actual value $\omega \in \Omega$ that will be used to calculate its value $f(\omega)$ is random.

   Random variables can be easily generalised to other domains than the real numbers.

   See also: `https://en.wikipedia.org/wiki/Random_variable` En francais: `https://fr.wikipedia.org/wiki/Variable_al%C3%A9atoire`

   (a) Example random variable EXAMPLE

   Take a 10-sided die. The outcomes of the die represent the space Ω. We can now create a Bernoulli variable from the die. For example, set $x(\omega) = 1$ when $\omega \in \{1, 5\}$ and $x(\omega) = 0$ when $\omega \in \{6, 10\}$.

   What is the distribution of x? Have you seen it before?

   (b) Generating a Bernoulli random variable. ACTIVITY:CODE

   In python, there are procedures for generating data from many types of random variables. However, all such methods for generating random numbers are not truly random. They rely on something called a pseudorandom number generator. The values output by this generator are then *transformed* so as to become similar to the random variable we want.

   i. Let us start by throwing a 10-sided die. How do you generate a Bernoulli random variable with $\theta = 0.6$ from the outcomes of the dice throw?

   ii. Now let us consider the following python code, which generates values from a Bernoulli random variable.

```
import numpy as np
return np.random.choice(2,p=[0.6, 0.4])
```

   Let us replicate it through a distribution of uniform randomv variables in $[0, 1]$.

```
import numpy as np
x = np.random.uniform() # returns a uniformly generated number in [0,1).
if x < 0.6:
  return 1
else:
  return 0
```

2.1.15 Group comparison ACTIVITY

Let us say we wish to compare the distribution among multiple groups. Perhaps we wish to compare the number of students who achieve a certain test like the table below.

	Success	Success
School	Male	Female
A	62%	82%
B	63%	68%
C	37%	34%
D	33%	35%
E	28%	24%
F	6%	7%
Average	45%	38%

1. Let us plot the success rate for females and males over different schools.
2. Does this show a bias? What information is missing?
3. Let us combine these two plots into one plot now for this we need to use the following code:

```
w=0.5
X = np.linspace(1,6*w,6)
M = ...
F = ...
plt.bar(X, M, align=edge, width=w/2)
plt.bar(X + w, F, align=edge, width=w/2)
```

2.2 Time-Series: model the evolution of a system

A time series $x_1, \ldots, x_t$ is simply a sequence of variables. We typically assume that this is random. How can we capture this dependency between variables?

Does the value of x_t depend only on the value of x_{t-1}? On all the previous values? Only on the time index t?

Frequently, sequential observations of a variable x_t are in fact noisy measurements of the true variable of interest, y_t, which we never observe. As an example, consider covid infections. There is a true, underlying, number of infections, but we only ever measure the number of positive cases detected in a day.

Generally, there are three tasks associate with time series modelling, always given data up to this point, i.e. $x_1, \ldots, x_t$.

1. Smoothing: What has happened in the past? Here we estimate y_{t-k} for $k > 0$.

2. Filtering: What is the current situation? To solve this problem we must estimate y_t.

3. Prediction: What will happen in the future? This involves predicting y_{t+k} for some $k > 0$.

These problems are all related and can be formalised in a statistical manner, and there are multiple algorithms that can be used to solve each problem. When $x_t = y_t$, then smoothing and filtering are trivial, but prediction is still an important problem. We focus here on a simple linear transformation such as the moving average as a basic solution method.

Smoothing For smoothing, a moving average filter is typically sufficient whenever $\mathbb{E}(x_t) = y_t$, i.e. x is just a zero-mean noisy measurement of y. Then we can construct the estimator

$$\hat{y}_t = \frac{1}{2n+1} \sum_{k=t-n}^{t+n} x_k.$$

Filtering When we wish to filter, at best we can take the moving average from the past n observations. If n is very large, then there is a a corresponding delay between our filtering and the final prediction.

$$\hat{y}_t = \frac{1}{n+1} \sum_{k=t-n}^{t} x_k.$$

The only way to remove the lag is to perform a more complex transformation of the original data. To see this, consider the problem of prediction.

Prediction Prediction means estimating something in the future. This task is never trivial, even with perfect observations, i.e. when $x_t = y_t$. In

this setting moving averages do not make sense. A simple idea is to *assume a linear trend*, e.g. that $y_{t+1} - y_t = y_t - y_{t-1}$. By re-arranging terms, we have that $y_{t+1} = 2y_t - y_{t-1}$. This gives us the estimator:

$$\hat{y}_{t+1} = 2x_t - x_{t-1}$$

2.2.1 Plotting lines

Here is a simple example of line plotting.

```
import numpy as np
X = [1, 2, 3, 4, 5, 4, 3, 2, 1] # define a small number of points
import matplotlib.pyplot as plt # import the plotting library
plt.plot(x) # perform a standard, simple plot
plt.savefig(f)
return f
```

What are such plots useful for?

2.2.2 Race times ACTIVITY

`https://en.wikipedia.org/wiki/1500_metres_world_record_progression`

Wikipedia has a table that shows the progression of 1500m world records.

1. Let us first show the records up to 1950 .
2. Try and predict the progrssion of world records on the board.
3. Let us now look at the actual graph. Is it what you expected?
4. How do you expect the progression to continue after 2020?
5. How do you explain this progression? Can you find data to validate or refute your explanation?

1. Scraping tables example :example:data-collection:

```
import pandas
tables=pandas.read_html("URL") # read a table
# convert date-string:
dt = datetime.datetime.strptime(string, '%Y-%m-%d').year
# string manipulation
string.replace("+", "0") # replaces a + with a 0
string.split(":") # splits a string into multiple strings
# data formats
float("12.2"); # converts a number into a float
```

2.2.3 Example: The inclination of Mars EXAMPLE

1. Plot Mars data
2. Show orbits
3. 3-body system, chaos and randomness

2.2.4 Example: Covid EXAMPLE

1. Plot covid data.
2. Smooth the data: moving average plots
3. Try and estimate past, current and future infections with simple tools.
4. Discuss: Are those simple tools sufficient? Is our visualisation consistent? Do we need something further?

2.2.5 Example: Stock market prices

See: Trading Economics

2.3 Scatterplots: model a relationship

Let us start with an example where we just have three variables. We can plot the relationship between any two of them.

```
X=[1, 2, 3, 4, 10, 6]
Y=[5, 2, 5, 3, 1, 2]
Z=[0, 1, 0, 1, 0, 1]
import matplotlib.pyplot as plt
plt.scatter(X,Y)
```

Variables are frequently in some array instead.

```
import numpy as np
n_data = 10
n_features = 3
data = np.random.uniform(size=[n_data, n_features]) # create some random data
plt.scatter(data[:,0], data[:,1]) #plot the first against the second column
# We can always take a 'slice' of the data:
data[:,[1,2]] # get columns 1 and 2 and all the rows
data[1:10, [0,2]] # get columns 0 and 2 and all rows 1-10
```

```
## : means everyhing
## a:b means everything from a to b
## [a,b,c] means a, b and c.
```

In dataframes, it we can deal with multiple variables by name

```
import pandas as pd
df = pd.DataFrame(data, columns =  ["Alcohol", "Caffeine", "Sugar"])
plt.scatter(df["Alcohol"], df["Caffeine"]) #plot the first against the second column
# getting slices is also possible in pandas dataframes, just slightly different:
df.loc[:,'Alcohol'] # get column Alcohol
df.loc[:,['Alcohol', 'Caffeine']] # get column Alcohol and Caffeine
```

2.3.1 Relationships as functions

A lot of relationships between two variables $x \in X$, $y \in Y$, can be described through some deterministic function $f : X \to Y$, i.e.

$$y = f(x).$$

If the relationship is one-to-one, then there exists an inverse function $f^{-1} : Y \to X$, so that

$$x = f^{-1}(y),$$

with

$$x = f^{-1}[f(x)].$$

Sometimes, however, the relationship betwen the two variables is not deterministic, that is the value of x does not uniquely determine the value of y, or the converse may occur... or both.

1. Physical relations EXAMPLE

   Many equations in physics relate two quantities. For example, there is the equation relating current I, voltage V and resistance R:

   $$V = IR.$$

   This relation can be inverted to obtain

   $$R = V/I, \qquad I = V/R.$$

   Let us say that the resistance R is fixed. By altering the voltage (e.g. by adding more batteries to a circuit) we can see that the current increases.

2.3.2 Relationships as joint distributions

The simplest way to model a stochastic relationship between two variables is to model the joint distribution $P(X, Y)$. Consider the example of heights versus weights: We can expect that the taller a person is, the heavier they will be. However, their weight will depend on the mass of their muscle and adipose tissue. These in turn depend on their age, sex, genetics and lifetime calorie expenditure and intake.

1. Weight and height distribution EXAMPLE

   Here we can plot the number of people having a certain height and weight combination. This can be done with a colour-map. This is not much different than a normal histogram - and is called hist2d in pyplot:

```
X = 100 / (1 + np.exp(-np.random.normal(size=100))) + 125
Y = X *  (1 + 0.1*abs(np.random.normal(size=100))) - 100
```

2.3.3 Relationships as conditional distribution.

Here we model the distribution of one variable given a fixed value for the other, e.g. $P(X|Y)$. The simplest thing is to only try to model the expected value $E[X|Y]$. How can we do this?

Method 1: polynomial fitting!

```
# returns the best fitting line to the data
a, b = np.polyfit(data_x, data_y, 1)
# Why is this the 'best' line? Because it minimises the total squared
# error between the predicted value and the actual ones.
# We can plot the line by this simple linear equation
ax = np.linspace(0,1)
plt.plot(ax, a * ax + b)
```

2.3.4 Example: Unemployment, GDP

Get some data financial data from FRED. This is time-series data. Can we actually make sense out of it in terms of correlations? Explore.

First, the unemployment rate: `https://fred.stlouisfed.org/series/UNRATE` Then, the GDP: `https://fred.stlouisfed.org/series/GDPC1` This has two different data frames.

```
# read the files
import pandas as pd
```

```
ur=pd.read_csv("UNRATE.csv")
gdp=pd.read_csv("GDPC1.csv")
# the date ranges are different, so we must try to merge them (inner join!)
merged = ur.merge(gdp)
```

3 Module 2: Experiment design (3 weeks)

3.1 Data collection and cleaning

3.2 Random sampling

1. Uniformly random sampling. How can we perform it?
2. Biased sampling, correcting for effects.
3. Importance sampling.

3.3 The data science pipeline

The experimental pipipeline has a number of different components.

1. Formulating the problem.
2. Deciding what type of data is needed.
3. Choosing the model and visualisation needed.
4. Designing the experimental protocol.
5. Generating data confirming to our assumptions.
6. Testing the protocol on synthetic data. Is it working as expected?

4 Module 3: Inference (2 weeks)

4.1 Expectation THEORY:PROBABILITY

Recall that a random variable f is a function $f : \Omega \to \mathbb{R}$. The expectation of a random variable with underlying distribution $P(\omega)$ is simply

$$\mathbb{E}_P[f] \triangleq \sum_{\omega \in \Omega} f(\omega) P(\omega).$$

There is nothing random about the variable itself, it is only the random input that makes its value random.

```
def random_variable(omega):
    return omega * omega
```

4.1.1 Centime exercise

A jar with coins is passed around the class.

1. The students are asked to guess how many coins it contains.
2. The students agree on a 50% confidence interval.
3. The students fit a normal distribution on this interval $[\mu - \frac{2}{3}\sigma, \mu + \frac{2}{3}\sigma]$.
4. Is this normal distribution a good choice? Are you 90\
5. Is a normal distribution generally appropriate?
6. Puzzle: Guess how many coins there are. If correct, then the class will share the money. If not, they will get nothing. What is the correct guess?

(If students have trouble with this, try with small numbers of coins and finite number of possibilities - demonstrate by playing the guessing game repeatedly)

4.2 Bayesian analysis THEORY:PROBABILITY

Recall the definition of Conditional probability:
$P(A|B) = P(A \cap B)/P(B)$,
i.e. the probability of A given B is the probability of A and B happening divided by the probability of B.
From this it follows that
$P(B|A) = P(A \cap B)/P(A)$.
Combining the two equations, we obtain:
$P(A|B) = P(B|A)P(A)/P(B)$.
So we can reverse the order of conditioning, i.e. relate to the probability of A given B to that of B given A.

4.2.1 The covid test problem ACTIVITY

10% of the class has covid, i.e. P(covid) = 0.1. Each one of you performs a covid test. If you have covid, the test is correct 80% of the time, i.e.

P(positive | covid) = 0.8. Conversely, if you do not have covid, there is still a 10% chance of a positive test, with P(positive | not-covid) = 0.1

How likely is it that you have covid if your test is positive or negative, i.e. P(covid | positive), vs. P(covid | negative)?

First of allzzzzzzzzzzzzzzzz, each one of you should independently generate a uniform random number between 1 and 10. For that, you can each throw a die, and record the outcome.

Then you throw a second die, and record that as well.

I will now pass over the tables and tell each one of you if they have a positive test.

Now, everybody with a positive test raises their hand. I expect it to be slightly more than 10% (but it depends).

4.2.2 The cards problem

1. Print out a number of cards, with either [A|A], [A|B] or [B|B] on their sides.
2. Get a card (say with face A), and ask what is the probability the other side is the same.
3. Have the students perform the experiment with:
   (a) Draw a random card.
   (b) Count the number of people with A.
   (c) Of those, count the number of people with A on the other side.
   (d) It should be clear that 1/3 of people have [A|A] and of those

4.2.3 The k-Meteorologists problem

Bayesian reasoning is most useful in the following setting:

- We have models of the world, $\{P_\theta | \theta \in \Theta\}$.
- We have a prior distribution $P(\theta)$ over the models.
- We obtain data D for whiche very model assigns a probabiltiy $P_\theta(D)$.
- We calculate the posterior distribution

$P(\theta|D) = P_\theta(D)P(\theta)/P(D)$.

- This tells us how likely each model is given the data.

In this example, we have k meteorological stations, each one of which gives us the probability that it will rain.

The table below gives the probability of rain according to each station.

Table 1: Rain probabilities and events

Station	Day 1	Day 2	Day 3
MeteoSuisse	70%		
Chris's Model	50%		
Actual rain			

The table below is our belief at the beginning of each day, about which station is overall best in predicting rain. What should our initial belief be?

Table 2: Belief at start of day

Belief	Day 1	Day 2	Day 3	Day 4
MeteoSuisse	90%			
Chris's Model	10%			

Write a program that updates the beliefs sequentially given observations and station predictions.

4.3 Hypothesis testing

4.3.1 Homework assignment: Define a data collection and analysis problem

5 Module 4: Advanced visualisation (2 weeks)

5.1 Geographical data

`https://scikit-learn.org/stable/auto_examples/neighbors/plot_species_kde.html`

5.1.1 Colour maps

1. Colour as a continuous variable.
2. Colour as a discrete variable.
3. Colour perception and interpretation.

5.1.2 Contour maps

1. Geographical contour
2. Density plots

5.2 Text data

6 Module 5: Data analysis in practice (2 weeks)

6.1 Survey data: The garden of many paths

6.2 Visualising fMRI data

6.3 Visualising GWAS data

6.3.1 Homework assignment: Visualisation of a project

7 Module 6: Project work and presentations (2 weeks)

8 Assignments

The course contains assignments and a project. The instructions for each assignment are given below. The assignments are largely done in class, but completed at home.

8.1 Table To Picture

Find a table in wikipedia on a topic of interest, and convert the table into a graph.

8.2 Plot deconstruction

TLDR: Take an existing plot from the web, re-create it, and try to improve it.

Find an inteesting plot from a web page on e.g. wikipedia. Try to identify some problem with the plot. To help you, ask yourself the following questions:

- Is the plot type appropriate?
- Is the data correct?

- Does the plot convey an appropriate message?
- Is thee more data somewhere that you could combine with the original to obtain a better picture?

After you have identified problems with the plot, data sources, or missing data, create a new plot, along with an explanation of how you addressed the original plot's deficiencies.

8.3 Newspaper article analysis

In this assignment you will read a newspaper article with some statistics and visualisations, and try to interpret what it says. You must study the article criticially. Are the conclusions supported by the data? Does the methodology make sens? Find primary sources that confirm or challenge the article to obtain a more rounded picture.

Here is a list of possible articles you can use. Feel free to suggest your own article and add it to the list.

`https://docs.google.com/spreadsheets/d/1QKj_L9f0UIH8Oqgs2kcjc8AU1eKZzsTOcYFSUO6HJBY/edit#gid=0`

8.4 Simulation study

For a simple visualisation problem, vary parameter values and simulate thousands of times under each set of conditions. Summarise your findings graphically.

8.5 Copy the master

You are given a visualisation constructed from a given dataset. You must create a similar visualisation from another dataset.

8.6 Open project

8.6.1 Project proposal

Propose a problem to solve, including:

- Hypotheses to test
- How to collect data
- How to analyse the collected data

8.6.2 Project Highlight

After you have started your project, each one of the project members presents a preliminary plot and explains it (5 minutes).

8.6.3 Project report

The completed project should include a report written by both students in the team. This should should address the points in the Assessment description.

9 Notation

For convenience, I include necessary mathematical notation

9.1 Sets

- $\mathbb{R}$: Real numbers
- $\mathbb{R}^d$: d-dimensional Euclidean space
- $\emptyset$: The empty set
- $A \subset B$: A is a subset of B.
- $A \cap B$: The intersection of A and B
- $A \cup B$: The union of A and B
- $A \setminus B$: Removing B from A
- Ω: The "universe"
- $A^c = \Omega \setminus A$: The complement of a set.
- $\{x|f(x) = 0\}$: The set of x so that $f(x) = 0$.

9.2 Analysis

- $\mathbb{I}\{x \in A\}$: indicator function (takes the value 1 if $x \in A$, 0 oterwise)
- $\sum_{x \in X} f(x) = f(x_1) + \cdots + f(x_n)$, with $X = \{x_1, \ldots, x_n\}$
- $d/dx f(x)$: derivative of f

- $\partial/\partial x f(x, y)$: partial derivative of f
- $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_n)$, vector of partial derivatives.

9.3 Probability

- $\mathbb{P}$: Probability (generally)
- $\mathbb{E}$: Probability
- P: A probability measure
- p: A probability density
- $P(A|B) = P(A \cup B)/P(B)$. Conditional probability, $A, B \subset \Omega$.
- θ: Parameter
- Θ: Parameter set
- $\{P_\theta | \theta \in \Theta\}$: A family of parametrised models
- $\mathbb{P}(x|y)$ conditional probability for random variables x, y (generally)

10 Graphics types

1. Histogram and 3D extensions
2. Density curve
3. Scatterplot
4. Smooth scatterplot
5. Violin plot
6. Line Plot
7. Confidence Intervals
8. Geographical/topological maps
9. Network graphs
10. Word cloud

See also: Catalogue of data visualisation

11 Schedule and links to other courses

The schedule of this and the other courses is in flux, but I do not expect it to change very much. In any case, the course will operate independently of the other courses. You should expect to cover the same topic more than once. However, this course will not focus on either statistical theory or programming.

Week	Statistics	Programming	In-Course	Homework
1 23 Sep	Course intro	Python intro	Histograms Randomness	Math score
2 30 Sep	R Intro Data manipulation Histograms Scatterplots Boxplots Variable types Mosaic plots Functions	Data types	Uncertainty Discrete Variables Continuous Variables	Form groups
3 7 Oct	Quantifying Variability Distribution Density function Histograms Skewness Quantiles	Control	Time-Series Linear functions Stock market prices Crime statistics S&P index World Records	Form groups
4 14 Oct	Qualitative vars in R Discrete vars in R	Structures	Scatterplots Unemployment	Proj. Proposal
5 21 Oct	Continous RV	Functions	Data Cleaning	Table2Picture
6 28 Oct	Continuous RV	Complements	Experiment design Random Sampling Undercounting Represnetative samples	Deconstruction
7 4 Nov	Continuous RV	Classes	Expectations	NewsPaper
8 11 Nov	Dependencies. Joint distribution. Conditional distribution.	Objects	Bayesian Inference	Project Highlight
9 18 Nov	Moments	Errors	Pipelines Simulation studies	Project
10 25 Nov	Covariance Correlation Scatterplots	Iterators	Hypothesis testing The garden of many paths	Project
11 2 Dec	Prices, returns	FP	Examples, project work	Project
12 9 Dec	Conditional expectations		Examples, project work	Project presentatio
13 16 Dec			Project presentations	Project work

12 Glossary

- Expectation: Espérance
- Histogram: Histogramme?
- Pie plot: ?
- Bar plot: ?
- Scatterplot: ?
- Randomness: Hasard
- Uncertainty: Incertitude
- Probability: Probabilité
- Stochastic: stochastique
- Random Variable: Variable aléatoire
- Sample: Échantillon
- Sampling: Échantillonnage
- Set: Ensemble
- Subset: Sous-ensemble
- Superset: Sur-ensemble
- Survey: Sondage
- σ-algebra: tribu, σ-algèbre

13 References

Python help: Use python's !help! function whenever you can.

```
help(print)
```

13.1 Some workshops

- `https://vishub.net/bach`
- `https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=22331`
- `https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=22351`

13.2 Books

- [CA] The Truthful Art. Cairo, Alberto :book:
- [GJ] Data Science Par La Pratique (Data Science from Scratch). Grus, Joel (Ch3: Visualisation, Ch6: Probability, Ch7: Inference, Ch9: Data collection, Ch10: Exploration, Ch14: Regression, Ch15: Regression+Bootstrap)