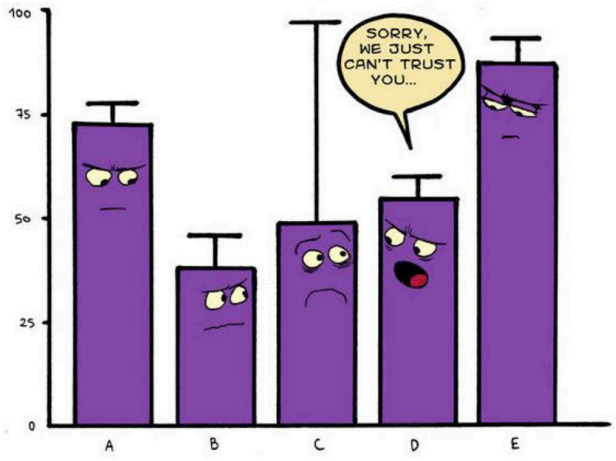
Introduction to basic data analysis and statistics in Earth sciences (using python 3)

Stephane Mazzotti, Université de Montpellier
Sept. 2017

"*It is easy to lie with statistics, but easier to lie without them*"
Frederick Mosteller



Reference: (https://www.pinterest.com/jillthompsong/stats-jokes/)

1- Why should we use statistics in Earth sciences?

- Statistics are useful to characterize and understand complex systems through the acquisition, analysis, and modeling of incomplete observations.

- Nearly all Earth science studies are based on limited, incomplete observations; statistics provide tools for quantitative analysis of the system attributes and their uncertainties.

- Statistics are only a set of tool to be used with others in the Earth science toolbox. When possible, statistics should be used to derive robust results in our studies, but they cannot be used as the only element.

- Example 1. You have measured two marker displacements of 12.1 m and 10.75 m along a fault trace. How do you characterize and analyze this data?
    - Answer 1 (bad): The average fault displacement must lie between 10.75 and 12.1 m.
    - Answer 2 (slightly better, but still bad): The average fault displacement is 11.425 m.
    - Answer 3 (better, but way too optimistic): The average fault displacement is 11.4 ± 0.7 m.
    - Answer 4 (correct, but not satisfying…): Ours observations suggest an average fault displacement near 11 – 12 m, but the sample is too small to derive robust estimations of its uncertainty.

- Example 2. Two granitic plutons are dated at 128 (± 10) Ma and 142 (± 5) Ma. Do they have the same age or do they represent two distinct phases of magmatic activity?
    - Answer 1 (bad): 128 Ma is clearly different from 142 Ma, so they must be distinct phases.
    - Answer 2 (slightly better): 128 ± 10 and 142 ± 5 don't overlap, so the ages are different.
    - Answer 3 (correct, but requires explanations…): The hypothesis that the plutons are of the same age can be rejected with 95% confidence, so the two ages can be considered different.
    - Answer 4 (also correct, but gives the opposite interpretation!): The hypothesis that the plutons are of the same age cannot be rejected with 99% confidence, so the two ages cannot be considered different.

- For most of you, this "Basic data analysis and statistics" course will be a refresher of high school and first-year university math (for the others, hang in there…), but we'll implement all the concepts using "**python 3**". So before we start, let's make sure that:
    - "python 3" (version 3.6 or close) is installed on your computer (if not, go to https://www.python.org/downloads)
    - You have installed the libraries "numpy", "scipy", and "matplotlib" (for example, using the python tool "pip". Cf. https://www.scipy.org/scipylib/download.html and https://matplotlib.org/users/installing.html)

- You can import the required libraries after launching python in your terminal (or GUI). For the rest of this course, python command will be given like this:

```
# text following a "#" is just comments
import numpy as np                  # imports "numpy" with short name "np"
from scipy import stats
import matplotlib.pyplot as plt
plt.ion()                           # makes graphics interactive
```

- **Learning outcomes**: If all goes well, at the end of this short course, you will be able to
  1. Describe a set of observations using
     - location (mean, median, …)
     - dispersion (standard deviation, …)
     - histogram
     - boxplot
  2. Describe circular data (modulo π or 2π).
  3. Analyze observations using simple statistical tests.
  4. Quote uncertainties and confidence intervals for basic data analysis (regressions, …).
  5. Handle data files, variables, arrays, and basic functions and graphs in python 3.

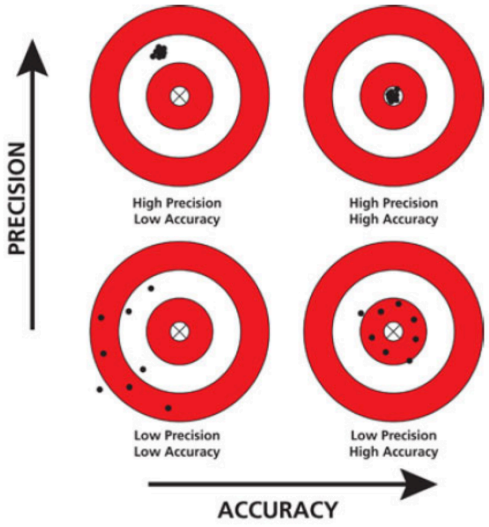
2- Before we start, a few definitions

- **Population**: the total set of all possible observations that can be made regarding a given system.
    - Example: studying the direction of extension during the Golf du Lion Eocene rifting, a population would be all the directions of extension on all the faults associated with this episode.

- **Sample**: a finite set of observations drawn from a population.
    - Example: In the case of Golf du Lion rifting, a sample would be a finite number of measurements of extension directions on a set of observed faults.

- **Accuracy ≠ Precision**
    - **Accuracy**: an estimation of the bias between a set of observations and the true value; i.e., how close to the true value the observations are.

    - **Precision**: an estimation of the dispersion / tightness of a set of observations; i.e., the ability to replicate the same result under the same condition. It is related to the number of <u>significant digits</u> in the numerical values.



    - Example: Relative to the measurement of π, observation *p1 = 3.1* is accurate but not precise, whereas observation *p2 = 3.523416* is not accurate but very precise.

- **Significant digits**: the numerals in a quantitative value that are <u>supported by the precision</u> of the observation. A common mistake consists overstating the significant digits, leading to a false sense of high precision in the result.
    - Example: Two tributary rivers have measured drainage basins of 130.1 km^2 and 231.5378 km^2. The total drainage area is 361.6 km^2 (not 361. 6378).

- **Hypothesis testing**: a formal statistical procedure used to accept or reject a working hypothesis with a specified significance (confidence) level. It typically consists in four stages: state the null and alternate hypotheses, formulate an analysis plan, analyze the sample data, and interpret the results**.**

    - **Null hypothesis**: a statement (usually denoted *H0*) that observations in a sample or in different samples show no relationship, i.e. that they result purely from chance. The null hypothesis is the "simplest" (no causal relationship) and should generally be considered true until proven otherwise.

- **Alternative hypothesis:** a statement (usually denoted *H1* or *Ha*) that sample observations are influenced by some non-random cause, i.e. that they show a relationship. H0 and H1 should be exclusive.

- **Significance level**: the probability (usually denoted α) of falsely rejecting a true hypothesis. It is chosen by the scientist to represent its tolerance for a false interpretation. Commonly, $\alpha = 0.05$ or $\alpha = 0.01$, i.e., we tolerate a 5% or 1% chance of error.

- **Confidence level**: the complement probability of the significance level, i.e. the probability that the interpretation is true. Commonly, it is set to 95% or 99% ($\alpha = 0.05$ or $\alpha = 0.01$).

- Example: In the case of the granites, the null hypothesis *H0* would be that they have the same age and the observed difference (128 ± 10 versus 142 ± 5) results from randomness (noise) in the date measurements. The alternate hypothesis *H1* would be that they are of different ages.

3 - Basic data characterization

3.1 - Data visualization

- Open the test dataset "datatest1.txt", load in the array variable "*samp*", look at it, and check how many elements it comprises.

```
samp = np.loadtxt("datatest1.txt")
samp                          # returns the content of the variable
len(samp)                     # returns the length of the variable
```

- Generate a histogram of the dataset. The histogram represents the number (default) or probability density / frequency (option "*normed=True*") of observations in each bin. In first example, the number of bins is defined automatically based on the dataset. In the second example, the number of bins is set to 50. Note that the command "*plt.hist*" also generates two arrays containing the counts and bin edges that can be saved in variables. Cf. "*matplotlib.pyplot.hist*".

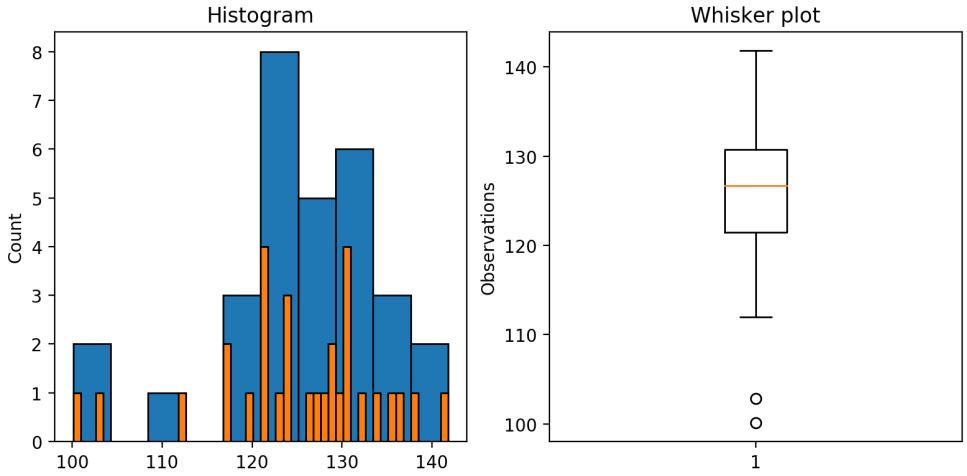
```
plt.hist(samp, edgecolor='black')
plt.hist(samp, edgecolor='black', bins=50)
count, bin, dummy = plt.hist(samp, bins=50)
```

- Generate a boxplot / whiskerplot of the dataset. The boxplot represents the non-parametric distribution of the dataset: median, quartiles, range, outliers (we'll get to that in section 3b). Cf. "*matplotlib.pyplot.boxplot*".

```
plt.boxplot(samp)
```

- Let's visualize both graphs together and spend a few minutes playing with the visualization. What can we say about the dataset "samp" and its distribution?

```
plt.subplot(1,2,1)
plt.hist(samp, edgecolor='black')
plt.title('Histogram')
plt.ylabel('Count')
plt.subplot(1,2,2)
plt.boxplot(samp)
plt.title('Whisker plot')
plt.ylabel('Observations')
```



A few words about graphs using "*matplotlib*" in python

The command "*plt.subplot*{x, y, z)" defines graphics regions for multiple graphics on a grid of *x* lines and *y* columns. The next graph will be drawn in *z* position (plot number).

The commands "*plt.title*" and "*plt.ylabel*" generate a title and a label on the y-axis of the graph. Guess what "*plt.xlabel*" does?

cf. "*matplotlib.pyplot*" for details.

3.2 - Location, dispersion, and outliers

- In statistics, a sample or population is commonly described by its location, which is a measure of its central tendency or average position, and its dispersion, which measures its tightness or spread around the location.

- Let's start with the location. The arithmetic **mean** (or average) μ is maybe the most common measure of location. It represents the central tendency or central value of the discrete set of observations. For a sample with *N* observations *x*, the arithmetic mean is

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

```
np.mean(samp)
np.round(np.mean(samp), 2)                    # for a cleaner output
```

NB: Other means exist (geometric, harmonic), but we won't consider them here.

- For a different quantification of the location, calculate the **median** Q_2, which is the value that separates the lower 50% from the upper 50% of the sample distribution. NB: $\mu \neq Q_2$.

```
np.median(samp)
```

- A third estimation of the sample location is its **mode**. The mode is the most frequent (most probable) value of the sample, but it's a tricky estimation. In our test dataset, there is no mode (all values occur only once) although the python command returns the first element… Using the histogram, check that the mode of "*samp*" should be ~123.

```
stats.mode(samp)
stats.mode([1,2,3,3,4,4,5,6,6,6,6,7])          # mode of a simpler vector
```

- Let's go one step beyond and consider dispersion by calculating the sample **quartiles**. The quartiles Q_1, Q_2, and Q_3 are the three values that divide the sample distribution into four quarters. They correspond to probabilities of 25, 50, and 75%. The second quartile Q_2 (50%) is the median.

```
np.percentile(samp, [25, 50, 75])
```

Note that in practice, any percentile can be calculated.

```
np.percentile(samp, [10, 30, 99])
```

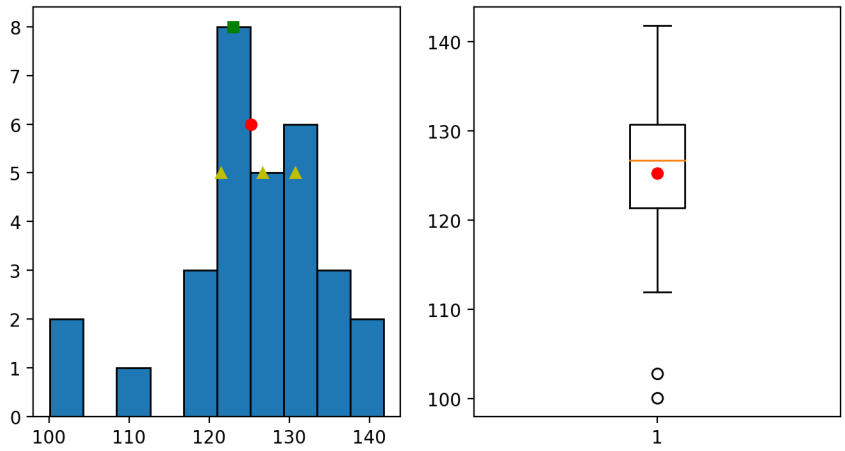
Using Q_1 and Q_3, we can calculate the **interquartile range** (also called midspread) $IQR = Q_3 - Q_1$, which is a common measure of the data dispersion.

```
np.percentile(samp, [75]) - np.percentile(samp, [25])
```

- The interquartile range is often used to find **outliers** in the data, i.e., values that are "too far" from the bulk of the sample. Outliers can correspond to experimental errors, or they can indicate complex (mixed) observations. Outliers are values that fall below $Q_1 - 1.5 * IQR$, or above $Q_3 - 1.5 * IQR$.

- Let's visualize these numbers for our test dataset. Note that the mean, median, and mode are not identical. Note also that the two observations around 100 – 105 are considered outliers.

```
plt.subplot(1,2,1)
plt.hist(samp, edgecolor='black')
plt.plot(np.mean(samp), [6], 'ro')
plt.plot(np.percentile(samp, [25, 50, 75]), [5, 5, 5], 'y^')
plt.plot([123], [8], 'gs')
plt.subplot(1,2,2)
plt.boxplot(samp, showmeans=True)
```



A few words about graphs using "matplotlib" in python

The command "*plt.plot*(x, y)" generates a scatter or line plot of the two vectors (or arrays) x and y of the same length (e.g., x = [1, 2, 3, 4], y = [1, 4, 9, 16] for the curve y = x^2 between 1 and 4).

- *plt.plot*(x, y) – with no additional argument will generate a line joining the array values
- plt.plot(x, y, *'ab…'*) with arguments *a*, *b, etc. between ' '* will generate a line or scatter plot using the argument letters to define the symbol color (r = red, g = green, y = yellow, …) and shape (o = circle, s = square, ^ = triangle, -- = dashed curve, …)

cf. "*matplotlib.pyplot.plot*" for details.

- So, how do we describe our dataset? Several options:
  - The sample mean and standard deviation are 125.3 and 9.1.

    print("The sample mean and standard deviation are", np.round(np.mean(samp),1), "and", np.round(np.std(samp), 1))

  - The sample mean is 125.3 ± 9.1 (standard deviation is implicit here).

    print("The sample mean is", np.round(np.mean(samp),1), "±", np.round(np.std(samp), 1))

  - The sample median and IQR are 126.7 and 9.3.

    print("The sample median and IQR are", np.round(np.median(samp),1), "and", np.round(np.percentile(samp, [75]) - np.percentile(samp, [25]) , 1))

3.3 – Very basic circular statistics

- In Earth sciences, many observations consist in circular (or directional) data, i.e. data that are defined modulo π (180 °) or 2π (360 °). Numerous mathematical tools exist to deal with this kind of data; we'll only check a couple of basic implementations in python 3.

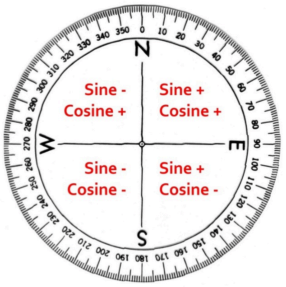
- Example: Let's say we measured four velocity directions (azimuth) at N010°, N013°, N005°, and N348°. Clearly, the average direction of this dataset is close to North, but a classic mean calculation yields N094°! We must treat them as a set of *N* angles α and transform them into polar coordinates (*X, Y*) to calculate the mean angle θ and the angular dispersion *r*. NB: The angular dispersion ranges from 0 (uniform dispersion over the circle) to 1 (concentration in the mean direction).

$$X = \frac{\sum_{i=1}^{N} \sin \alpha_i}{N}, Y = \frac{\sum_{i=1}^{N} \cos \alpha_i}{N}$$

$$r = \sqrt{X^2 + Y^2}, \cos \beta = \frac{X}{r}, \sin \beta = \frac{Y}{r}$$

$$\theta_T = \tan^{-1}\left(\frac{\sin \beta}{\cos \beta}\right)$$

$$\theta = \theta_T, si \sum \sin \alpha_i > 0 \; et \sum \cos \alpha_i > 0$$
$$\theta = 180 - \theta_T, si \sum \sin \alpha_i > 0 \; et \sum \cos \alpha_i < 0$$
$$\theta = 180 + \theta_T, si \sum \sin \alpha_i < 0 \; et \sum \cos \alpha_i < 0$$
$$\theta = 360 - \theta_T, si \sum \sin \alpha_i < 0 \; et \sum \cos \alpha_i > 0$$



- This is clearly more complex than simple non-circular statistics. The "*stats*" library in python has a few simple functions to obtain the circular mean and standard deviation of a dataset (with any upper and lower boundaries for the circular).

```
v = [10, 13, 5, 348]
np.mean(v), np.std(v)                  # classic (incorrect) μ and σ
stats.circmean(v, high = 360)          # μ modulo 360
stats.circstd(v, high = 360)           # σ modulo 360
```

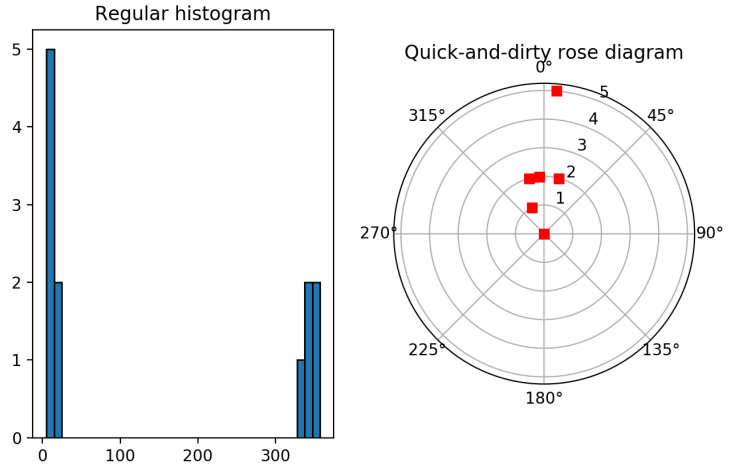
The correct mean and standard deviation of our set of directions are N004 ± 10 °.

- Be careful to differentiate "direction" and "orientation". The former (direction) refers to vector-type data (i.e., modulo 360 °), whereas the latter refers to line-type data (i.e. modulo 180 °).

- Example: A fault plane azimuth can be defined as an orientation (depending on the convention), such that N045° = N225° (i.e., NE - SW). In this case, you can use the "*stats*" functions with the option "*high = 180*".

- The problem with circular data is also evident with histograms. A regular histogram will not account for the circularity of the data, we must use a rose or wind diagram ("circular histogram"). The example below shows a set of velocity direction centered near N000° (North) using a hand-made rose diagram. There are several more elegant wind and rose diagrams in python, but they require specific libraries (cf. "windrose", https://pypi.python.org/pypi/windrose).

```
v = v + [5, 15, 357, 335, 13, 22, 341, 342]      # adds a few numbers
plt.subplot(1,2,1)
nb, bin, dummy = plt.hist(v, bins = 36, edgecolor='black')
plt.title("Regular histogram")
ax = plt.subplot(122, projection='polar')        # make a polar plot ...
ax.set_theta_direction(-1)                       # ... clockwise and rotated
ax.set_theta_offset(np.pi/2)                     # … and rotated 90 degres
bin = np.arange(5, 365, 10)
ax.plot(bin/360*2*np.pi, nb, 'rs')
plt.title("Quick-and-dirty rose diagram")
```



4 - Probability distributions (density functions)

4.1 - Normal distribution, standard deviation, uncertainty and confidence interval

- The normal (or Gaussian) distribution is one of the most common continuous probability distributions in Earth sciences. Most sets of independent observations follow the central limit theorem and converge to a normal distribution when the sample is sufficiently large.

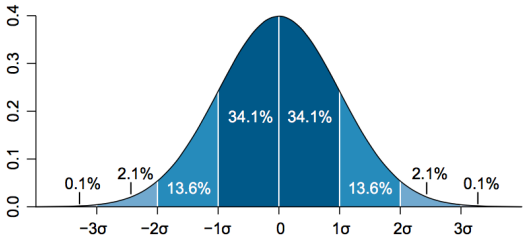
- A normal distribution is characterized by its mean μ and standard deviation σ. The standard deviation is a measure of the variation or dispersion of the data around the mean of normally distributed sample.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), -\infty < x < \infty$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$$

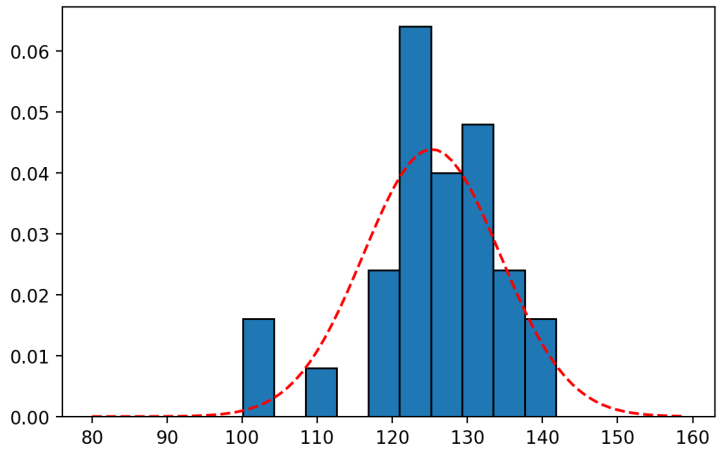
np.std(samp) *# calculates the standard deviation of "samp"*

- A few characteristics of the normal distribution:
  - It is symmetric about its mean.
  - mean = median = mode.
  - It is infinite on both ends.
  - 68.2% of the distribution is within +/- one standard deviation of the mean.
  - 95.4% of the distribution is within +/- two standard deviations of the mean.



- Getting back to our test dataset "*samp*", its mean and standard deviation are 125.3 and 9.1. Let's see how well it fits a normal distribution and discuss the result. Note how the observations around 100 – 105 are not explained by the normal distribution (reminds you of something?).

```
plt.hist(samp, edgecolor='black', normed=True)
mu = np.mean(samp)
sigma = np.std(samp)
x = np.arange(80,160,1)
pdf = 1 / (sigma*np.sqrt(2*np.pi)) * np.exp(-1 * (x-mu)**2 / (2*sigma**2))
plt.plot(x, pdf, '--r')
```



Woah, how did we do that?

The command "*np.arange*{x, y, z)" generates a 1D array (vector) of real values ranging from *x* (included) to *y* (excluded) every *z*. In the example above, we created an array from 80 to 159 every 1 (80, 81, 82, …).

The expression "*pdf* = …" calculates the theoretical normal probability distribution using

- standard "*numpy*" function (*np.pi*, *np.sqrt*, np.exp)
- the sample "*samp*" mean (*mu*) and standard deviation (*sigma*)
- the array *x* (80 to 159)

The calculation is performed on each element of *x*, resulting in a 1D array *pdf* of the same length with the normal distribution function.

- **Uncertainty and confidence interval**. We very commonly assume that a sample follows a normal distribution to quote the observation uncertainty in terms of standard deviation, 68%, or 95% confidence intervals.
  - Example: Our test dataset correspond to 30 measurements of ages on a granite pluton. Assuming the sample follows a normal distribution, we define the pluton average age as:
    - The pluton age is 125.3 ± 9.1 Ma ($\mu \pm \sigma$).
    - The pluton age is between 116.2 Ma and 134.4 Ma (68% confidence interval = ($\mu - \sigma$) – ($\mu + \sigma$)).
    - The pluton age falls within the range (107.1 – 143.5) Ma (95% confidence interval = ($\mu - 2*\sigma$) – ($\mu + 2*\sigma$)).

4.2 - Law of large numbers, minimum sample size for useful statistics

- Clearly, the dataset "*samp*" does not perfectly fit a normal distribution. It contains too many observations at the lower end (100 – 105) and around the mean (120 – 135), and it lacks observations in several ranges (110 – 120, upper end).

- In statistics, the law of large numbers states that the mean of sample with a large number of independent observations is close to the population mean, and will get closer as more observations are added. The central limit theorem further states that the sample will converge towards a normal distribution when it is sufficiently large…

- In practice, the minimum sample size depends on the applications and on the tolerated margin of error, which is gets much too detailed for us. Let's generate of few random samples from a normal distribution ($\mu = 100$, $\sigma = 10$) to get a general idea.
    - Repeat the last three steps to discuss how the results vary with different random samples.
    - Repeat with larger sample sizes (20, 50, 100, 1000).
    - What is the minimum size of a sample to get a "good" estimation of the mean and standard deviation of the parent population?

```
mu = 100
sigma = 10
x = np.arange(60,140,1)
pdf = 1 / (sigma*np.sqrt(2*np.pi)) * np.exp(-1 * (x-mu)**2 / (2*sigma**2))
plt.plot(x, pdf, '--r')
obs = np.random.normal(mu, sigma, 10)
plt.hist(obs, normed=True)
np.mean(obs), np.std(obs)
```

Generating random samples with "*numpy*"

The command "*np.random.normal*(x, y, z)" generates a 1D array of *z* elements randomly drawn from a normal distribution of mean *x* and standard deviation *y*. A whole variety of probability distributions (lognormal, chi2, …) exist in "*numpy*" to play with.

4.3 - Lognormal distribution, skewness, multimodal distribution, and other weird things

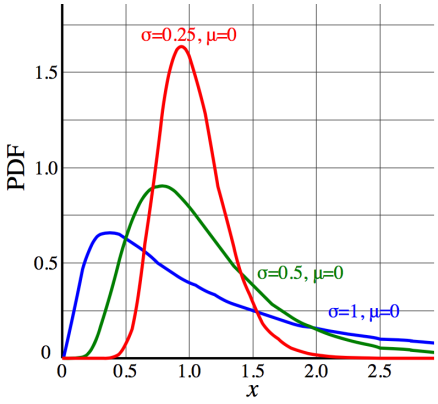
- Another very common continuous probability distribution in Earth sciences is the lognormal distribution. As its name suggests, it corresponds to the distribution of a variable whose logarithm is normally distributed.

- The lognormal distribution can represent processes driven by the accumulation of many small percentage changes, resulting in population with more frequent small values and less frequent large ones
    - Examples: daily rainfall, fault length, …

- A lognormal distribution is characterized by two parameters μ and σ (mean and standard deviation of the logarithm of the observations). σ is often referred to as the scale, which defines the width (and skewness) of the distribution.

$$f(x,\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right), 0 < x < \infty$$

- A few characteristics of the lognormal distribution:
    - It is asymmetric and skewed positively (to the right)
    - Its skewness increases with the scale
    - mean ≠ median ≠ mode
    - It strictly positive and infinite to the right



- Let's compare the normal and lognormal distributions applied to a new dataset. This time, we'll look at return periods of very large ($M_W \sim 9$) earthquakes in the Cascadia subduction zone.
    - Open the CSV file containing the data and load it into a new 1D array variable "*eq*"

```
eq = np.loadtxt("datatest2.csv", delimiter=',')          # note the ','
```

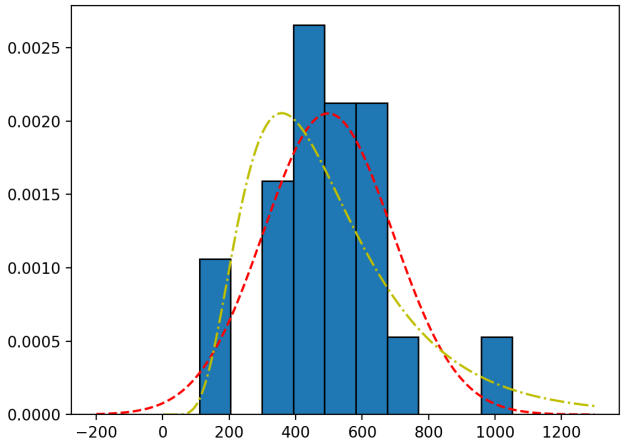
    - Find the parameters of the normal and lognormal distributions that best fit our dataset. Verify that these results fit with the mean and standard deviations of the sample.

```
mu, sigma = stats.norm.fit(eq)
logsigma, dummy, logmu = stats.lognorm.fit(eq, floc = 0)
```

    - Plot the data histogram, best-fit normal and lognormal distributions, and discuss the result.

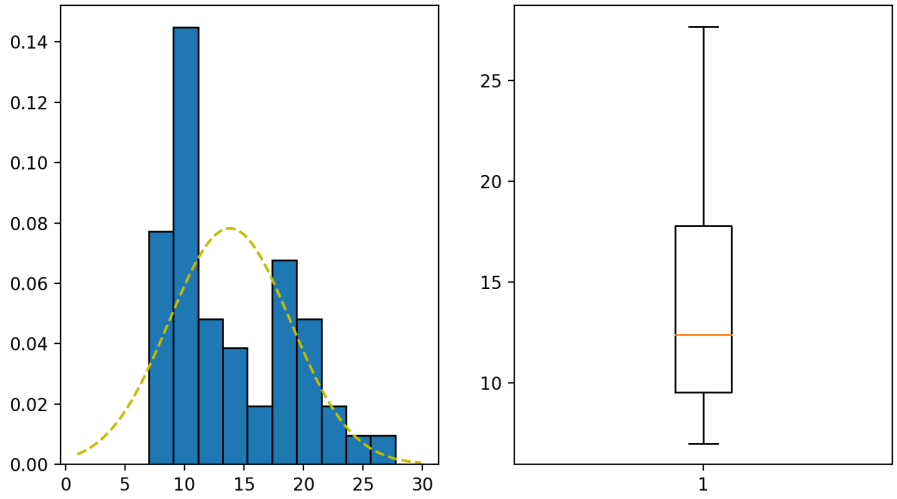
```
plt.hist(eq, normed=True, edgecolor='black')
x = np.arange(-200, 1300, 1)
norm = 1 / (sigma*np.sqrt(2*np.pi)) * np.exp(-1 * (x-mu)**2 / (2*sigma**2))
```

```
plt.plot(x, norm, '--r')
x = np.arange(1, 1300, 1)
mu = np.log(logmu)
sigma = logsigma
log = 1 / (x*sigma*np.sqrt(2*np.pi)) * np.exp(-1 * (np.log(x)-mu)**2 /
(2*sigma**2))
plt.plot(x, log, '-.y')
```



- More complex processes can lead to more complex distributions that can be difficult to deal with. Let's consider "*datatest3.txt*", which shows an example of a bimodal distribution. How well is it characterized by its mean and standard deviation?

```
samp = np.loadtxt("datatest3.txt")
plt.hist(samp, edgecolor='black')
# now you're on your own…
```



5 - Basic statistical analysis

5.1 - Does one (two) sample(s) follow a (the same) normal distribution? The Student's t test

- Let's get back to the issue of result accuracy. The following dataset (4.3, 4.9, 6.0, 6.2) represent four measurements of slip rate, in mm/a, on a given fault. We can state an average slip rate of 5.3 ± 0.8 mm/a ($\mu \pm \sigma$), but how confident are we that the result is accurate? I.e., can we reject the hypothesis that the true slip rate is much larger or smaller than 5.3 mm/a?

- The t test can be used to address this kind of question.
    - State the null and alternate hypotheses:
        - *H0* = the sample follows a normal distribution of mean 5.3 (i.e., the true slip rate is 5.3 mm/a).
        - *H1* = the sample true mean is different from 5.3.

- Calculate the p-value for a one-sample t test and compare with the desired significance level. The p-value can be viewed as the probability of making an error in accepting the null hypothesis.

```
slip = [4.3, 4.9, 6.0, 6.2]
np.mean(slip), np.std(slip)
stats.ttest_1samp(slip, 5.3)
```

  In this example, $p = 0.92$. Let's say we want a significance level $\alpha = 0.05$ (95% confidence level of not making a mistake). $p > \alpha$, so we cannot reject the null hypothesis (that the true mean is 5.3).

- Let's check a different *H0* =the sample follows a normal distribution of mean 6.0.

```
stats.ttest_1samp(slip, 6.0)
```

  In this case, $p = 0.25$. $p > \alpha$, and we cannot reject the null hypothesis that the true slip rate is 6.0 mm/a !

- One more round with a more extreme *H0* = the sample true mean is 10.0.

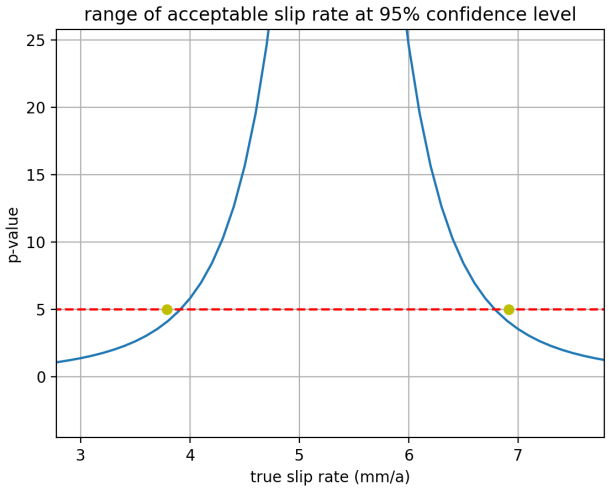
```
stats.ttest_1samp(slip, 10.0)
```

  Here, $p = 0.002 < \alpha$. In this case, we can reject the null hypothesis and state with 95% confidence that the true sample mean is not 10.0.

- Verify that any slip rate within the range (3.9 – 6.8) cannot be rejected with a 95% confidence. How does that compare with the sample mean and standard deviation?

What does that tell you about the using two standard deviations as a measure of the 95% confidence interval?

```
mrange = np.arange(0, 12, 0.1)
stat, p = stats.ttest_1samp(slip, mrange)
plt.plot(mrange, p*100)
plt.grid()
plt.plot([0, 12], [5, 5], '--r')
plt.plot([np.mean(slip) - 2 * np.std(slip), np.mean(slip) + 2 * np.std(slip)], [5, 5], 'yo')
```



- The t test is also commonly used to address questions such our very first example (two plutons with the same age or not). In this kind of problem, we use a two-sample t test to check if two independent samples have identical means.

- Do the two Cevennes plutons dated at 128 (± 10) Ma and 142 (± 5) Ma have the same age?
  - Null hypothesis *H0* = the two samples have identical expected means (i.e., they are drawn from the same population).

- In fact, the t test cannot answer this question without further information:

  - Case 1: We have the actual observations.

```
a1 = [116, 145, 127, 123, 128]
a2 = [141, 149, 137]
stats.ttest_ind(a1, a2)
```

The p-value $p = 0.079$ is larger than the significance level (set to $\alpha = 0.05$), so we cannot reject the null hypothesis with 95% confidence. In other words, we cannot exclude the fact that the two plutons have the same age and we have to use as a

working hypothesis that the ages are similar.

- Case 2: We don't have the actual observations, but we have the number of observations in both samples.

```
stats.ttest_ind_from_stats(128, 10, 5, 142, 5, 4)
```

The p-value *p = 0.039* is smaller than the significance level (α *= 0.05*) and we can reject the null hypothesis with 95% confidence. In other words, we are confident that that the two plutons have different ages.

- But wait, what if we had chosen a significance level α *= 0.01* in order to be sure at 99% of our interpretation? Then we could not have rejected the null hypothesis in both cases. Conclusion:
  - Low significance levels (high confidence levels) provide more drastic – but safer – interpretations.
  - Statistics are a useful tool, but in the end, they always require an arbitrary choice by the scientist (confidence level, threshold).
  - There is nothing magic at 95% confidence. In fact, recent discussions in scientific communities (especially medical sciences) propose to use more drastic confidence levels (99% or 99.9%) to avoid false interpretations.

5.2 - Correlation between two samples

- The coefficient of correlation is a measure of the statistical relationship or dependence between two random variables. There are several methods to calculate correlation. The most common is the Pearson product-moment correlation, which we'll just call "correlation" and note "*r*". For two samples of observations *x* and *y* of the same length *N*, specific mean $\mu_{x\,/\,y}$, and standard deviation $\sigma_{x\,/\,y}$:

$$r = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{(N-1)\sigma_x\sigma_y}$$

```
x = np.random.normal(0, 1, 10)
y = np.random.normal(0, 1, 10)
plt.plot(x, y, 'or')
plt.ylim([-5, 5]), plt.xlim([-5, 5])
np.corrcoef(x, y)[0, 1]
```

Depending on the random samples, the coefficient of correlation will vary between -1 and 1. *r = 0* indicates no relationship / dependence between the samples. *r = 1 (-1)* indicates a full positive (negative) relationship / dependence.

- The python function "*corrcoef*" allows calculating the correlation between several arrays

(x, y, z, …), and returns the respective coefficients of correlation r_{xy}, r_{xz}, etc.

```
x = np.random.normal(0, 1, 10)
y = np.random.normal(0, 1, 10)
z = np.random.normal(0, 1, 10)
np.corrcoef([x, y, z])
```

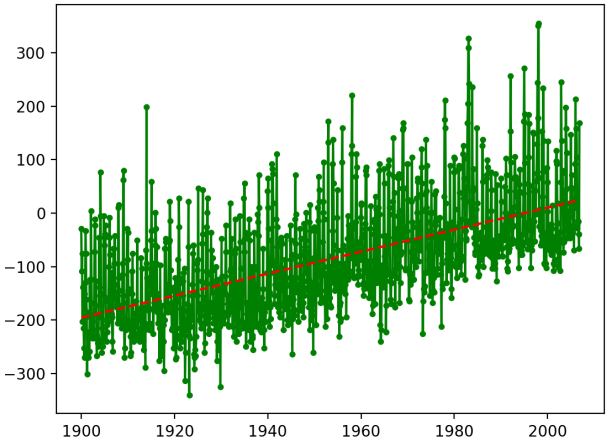
- The phrase "Correlation does not imply causation" is often found in statistics textbooks. It is meant to warn that inferring a causative relationship between two samples based on a strong coefficient of correlation is a logical fallacy. As seen above, two purely random samples can have a high correlation (say, $r > 0.5$) without any causal relationship.
    - Check that this is partly related to the sample size but rerunning the functions above with samples of 100 observations. You will have a much harder time finding correlation values larger than 0.2.

5.3 - Does a dataset show a significant trend? Simple linear regression

- Linear regressions are very classical problems in Earth sciences. They are used to study systems with linear rates of changes, expressed as $y = a * x + b$, with a the rate of change (slope, gradient, velocity, etc.) of a set of observations y explained by a set of explanatory values x.

- Example. The tide gauge in Seattle, USA, measures relative sea level height since 1900. We want to know the rate of relative sea level change during the 20th century using heights measured every month.

```
vanc = np.loadtxt("datatest4.txt")
date = vanc[:, 0]            # extracts the 1st column of the 2D array "vanc"
rsl = vanc[:, 1] * 1e3       # convert the 2nd column to mm
plt.plot(date, rsl, '.-g')
a, b, rval, pval, ste = stats.linregress(date, rsl)
x = np.array([1900, 2006])
plt.plot(x, a * x + b, '--r')
print("The 20th century rate of sea-level rise in Seattle is", np.round(a, 2),
"±", np.round(aste, 2), "mm/a.")
```



A few words on working with arrays in python

If we define *test = [10, 20, 30, 40]*, then *test[2]* returns the third element of *test* (30) . Remember that python starts counting at 0! The array can be subsampled over a range *a – b* using *test[a : b]*. For example *test[0:2]* returns [10, 20, 30] and *test[:]* returns the whole array.

2D arrays can be subsampled in the same fashion, but with two indices. For example, *test = np.array([10, 20, 30, 1, 2, 3]).reshape((2,3))* creates a 2D, 2 lines / 3 columns array. *test[1, :]* returns values in second line, all columns (1, 2, 3).

Arrays can be tricky things to play with. Let's wait for the "computing" course to find more.

- There are several indicators to measure the robustness of the linear regression.
  - The standard error on the slope (here *ste = 0.08 mm/a*). It is close to, but not identical, to a standard deviation (it is in fact the standard deviation of the sampling distribution, but let's not worry about that…). In most cases, we can use this standard error as we would the standard deviation of a normal distribution (68% confidence interval, etc., cf. section 4.1).

  - The p-value of a the test whose null hypothesis *H0* is that the slope is zero. In this example, the p-value is extremely small (*pval = .6 * 10^-24*) so we can reject the null hypothesis with a very high confidence.

  - The coefficient of correlation *R* and the associated coefficient of determination R^2. The latter measures the proportion of the variance in the dependent variable (*y*) that is predictable from the explanatory variable (*x*). In other words, in our example R^2 *= rval**2 = 0.35*, meaning that the linear model explains about 35% of the dispersion in the data.

- OK, so what does that tell us? Well, we can state the following:
  - The rate of sea-level rise is 2.06 ± 0.08 mm/a (well constrained, small uncertainty)
  - We can reject with over 99.9% confidence the hypothesis that the rate of sea-level change is 0 mm/a.
  - This linear trend only explains about 35% of the dispersion / spread in the data. As can be seen on from the graph, about 65% of the dispersion in not explained by / related to a linear trend.

- What if we want to study non-linear processes (logarithmic, exponential, sinusoidal, …)?
  - One way to deal with some non-linear problems is to linearize them by changing variables.
    - Example: You want to find the parameters of an exponential decay curve $y = b + a * exp(-x)$. By taking the logarithm of both sides of the equation, you can define the linear equation $y1 = a1 – x$, with $y1 = ln(y – b)$ and $a1 = ln(a)$, which you can solve using a linear regression function.

  - python also comprises non-linear regression functions, which can deal with more complex problems, and provide more correct statistics than linearization. Cf. for example "*curve_fit*" from the library "*scipy.optimize*".