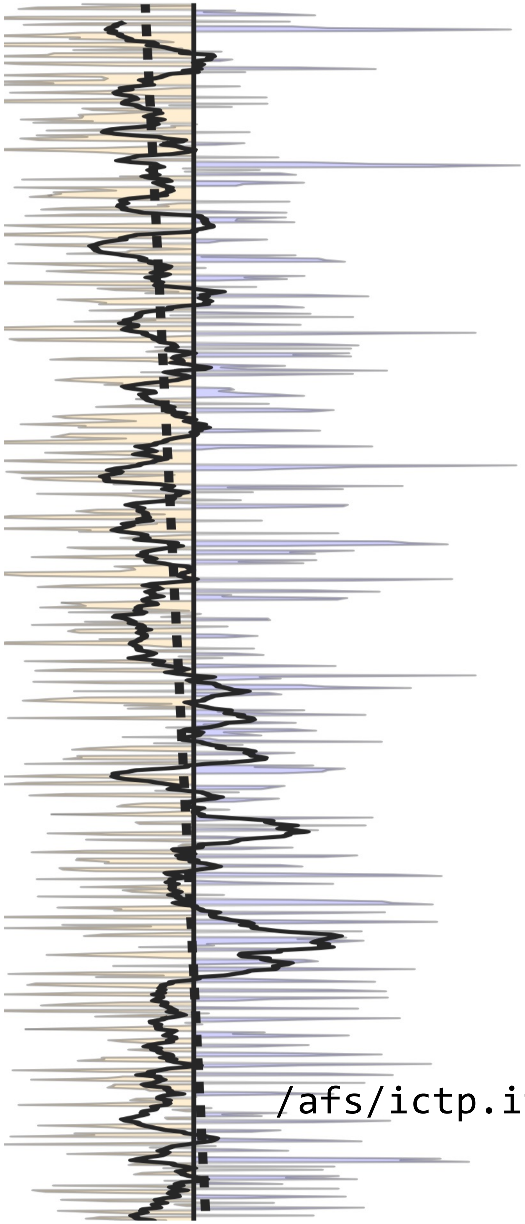


Data Analysis in Geosciences

Lecture Notes, 2023/2024

Maria Vittoria Guarino
(mguarino@ictp.it)

/afs/ictp.it/public/m/mguarino/WORLD/Data_Analysis_2023_2024/DA_Lecture*.pdf

Lecture 1: Introduction to data analysis and descriptive statistics

Lecture 2: Statistical significance tests and linear regression

Lecture 3: Gridded data and interpolation methods

Lecture 4: Fourier Transform and spectral analysis

Lecture 5: Commonly used indexes in geosciences and python plotting

1.5h classes split into: 1h frontal teaching + 30 min computer-based exercises

Reading

“*Environmental Data Analysis with MATLAB or Python: Principles, Applications, and Prospects*” by William Menke, Elsevier. Available from the ICTP Library.

“*A Hands-On Introduction to Using Python in the Atmospheric and Oceanic Sciences*” by Johnny Wei-Bing Lin. https://www.johnny-lin.com/pyintro/

“*Statistical Methods for Climate Scientists*” by DelSole T. and Tippett M., Cambridge University Press. https://www.cambridge.org/core/books/statistical-methods-for-climate-scientists/85F85ED46389BBD726E41F2EE8AA6824

“*Basic Numerical Methods in Meteorology and Oceanography*” by Doos K. et al., Stockholm University Press. https://www.stockholmuniversitypress.se/site/books/m/10.16993/bbs/

“*A Student's Guide to Fourier Transforms*” by J.F. James, Cambridge University Press. https://www.cambridge.org/core/books/students-guide-to-fourier-transforms/19FC459B947887F3816F56AE827E6722

Python Tutorials:
https://docs.python.org/3/tutorial/
https://matplotlib.org/stable/plot_types/index

Python Environment

ICTP Jupyter Notebook: https://jupyter.ictp.it/

```
import numpy as np
import scipy
import pandas as pd

import matplotlib.pyplot as plt

import iris
from mpl_toolkits.basemap import Basemap
```

NEED INSTALLING

To install iris (needs Conda): `conda install -c conda-forge iris`

https://scitools-iris.readthedocs.io/en/v3.0.1/installing.html

To install iris: `conda install -c conda-forge basemap`

https://matplotlib.org/basemap/stable/users/installation.html

Python Environment

If using the ICTP Jupyter Notebook: README_install_iris.txt

1) Upload the ‘jeditor.py’ file provided

2) File -> New -> Terminal

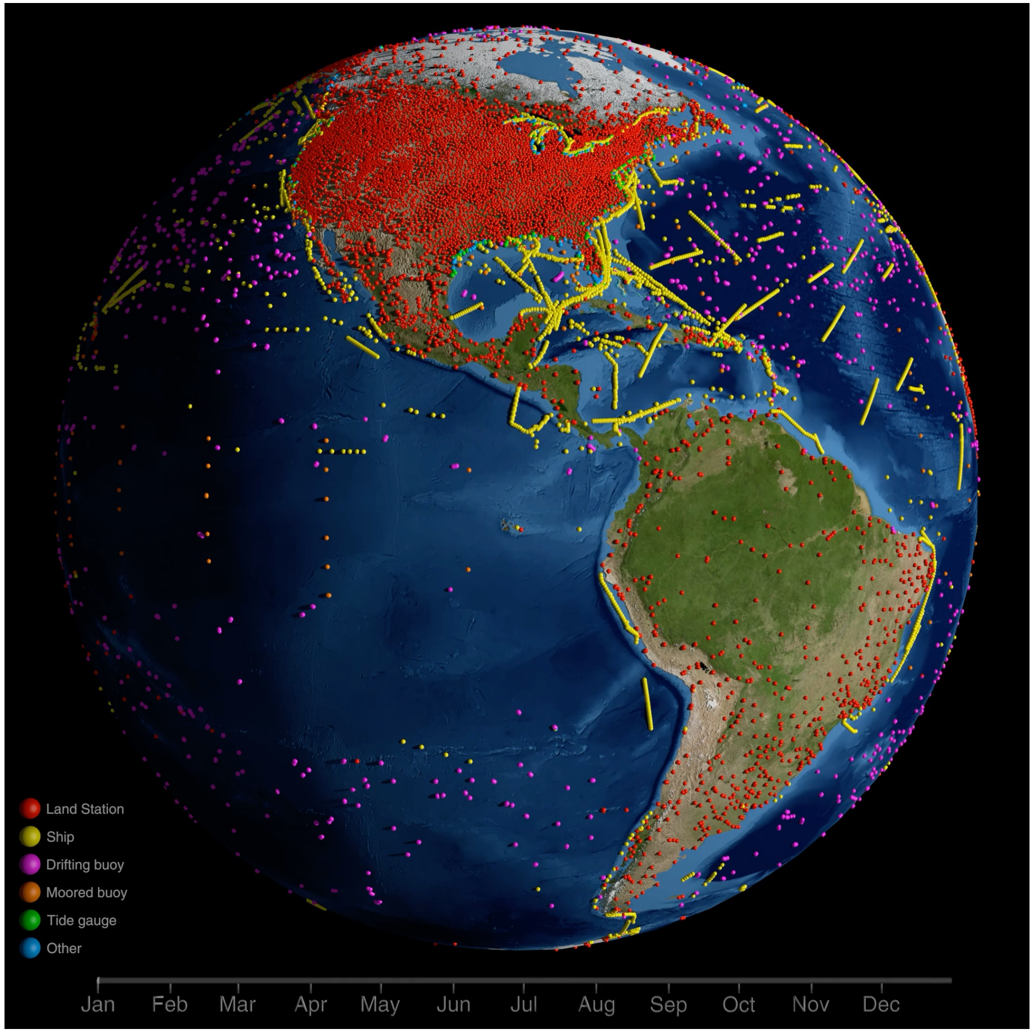
3) Type :

```
conda create -y -n iris
source activate iris
conda install -p $HOME/.conda/envs/iris -c
conda-forge conda ipykernel iris
python -m ipykernel install --user --name=iris
python3 jeditor.py
```

To install another packet (for example, Basemap):

```
source activate iris
conda install -p $HOME/.conda/envs/iris -c conda-forge basemap
```

Lecture 1
Introduction to data analysis and descriptive statistics



We want to monitor and analyze all domains of the Earth System:

LAND
ATMOSPHERE
OCEAN
ICE

Where does data come from?

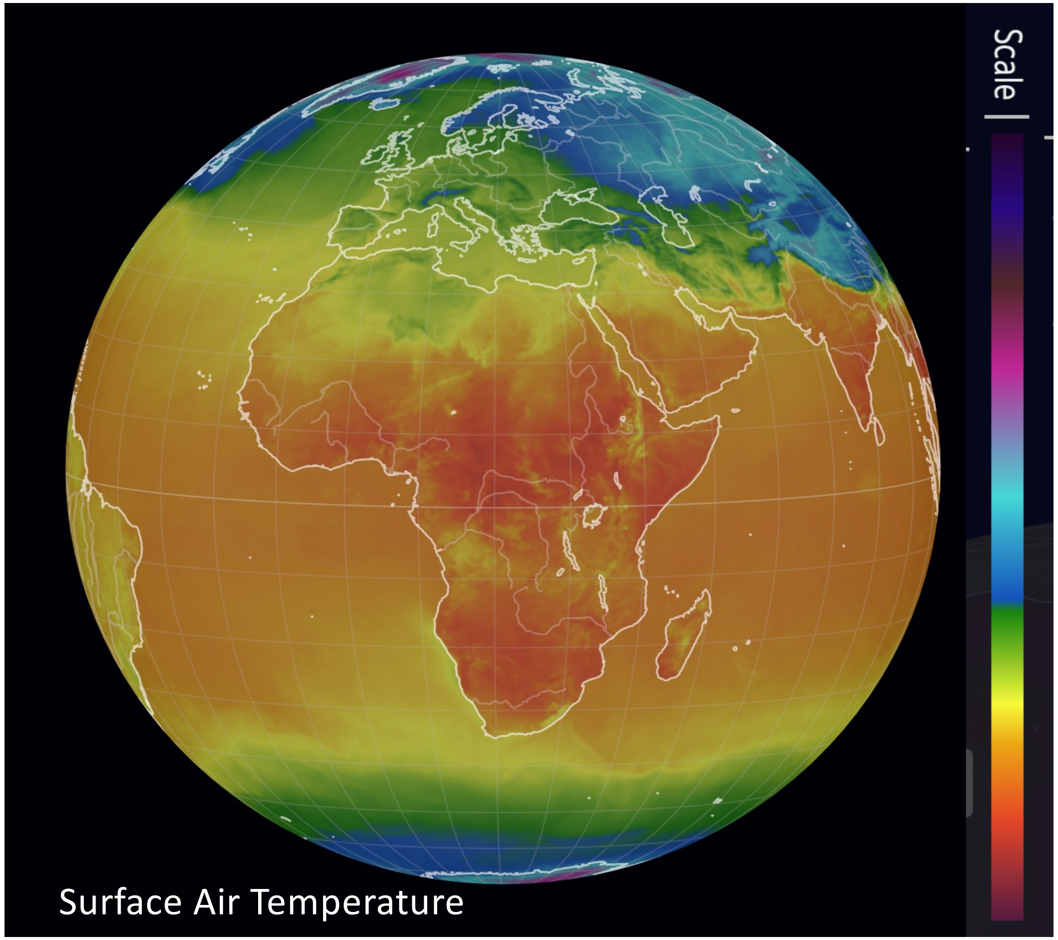
Ground Stations (meteorological stations, seismographs)
Weather Radars
Buoys
Satellites
etc.

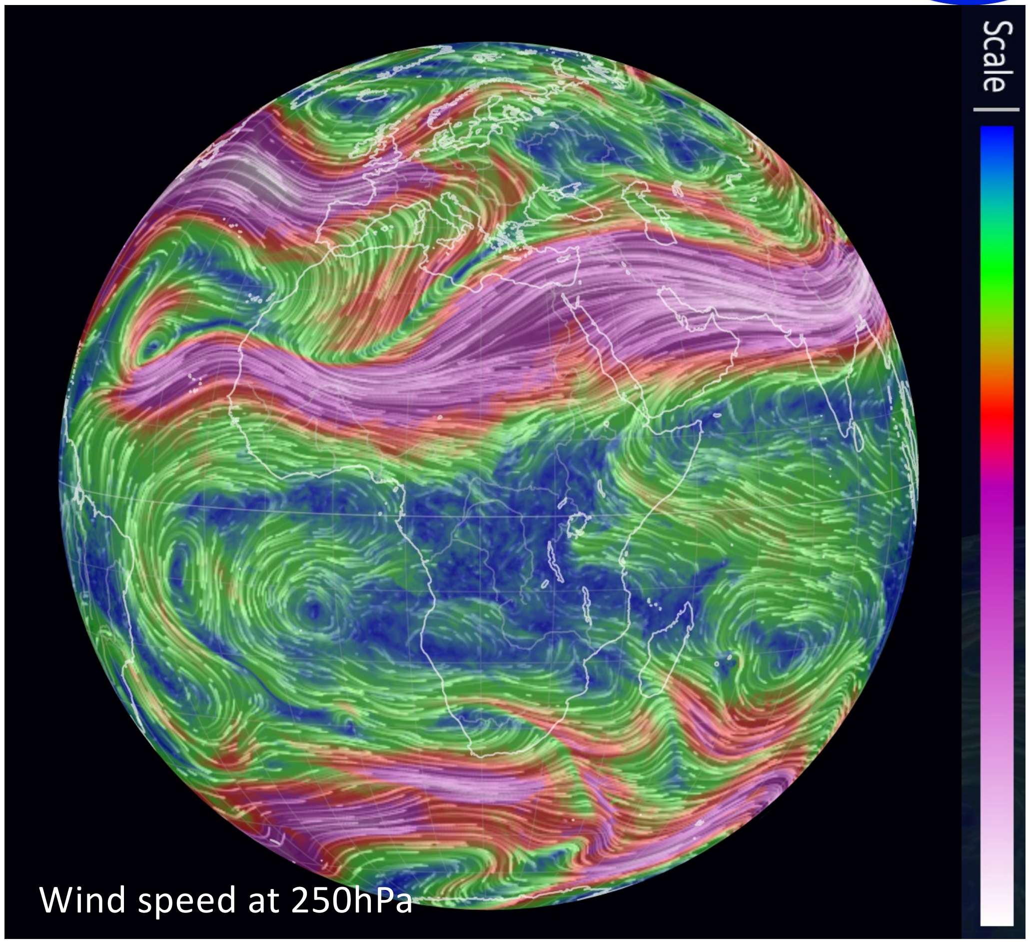
Data format: ASCII, gridded (netcdf), binary, GeoTIFF, Vector and shape files (*ESRI*), etc..

NASA Scientific Visualization Studio: https://svs.gsfc.nasa.gov/5208/

What do we analyze data for?

- To see what they look like





Source: https://earth.nullschool.net

What do we analyze data for?

- To see what they look like
- Detect patterns
- Test hypotheses
- Study relationships among variables
- Study temporal evolution (predict future/past behaviour)
- Assess regional differences

The quantitative approach

We can describe data numerically using statistical indexes: **DESCRIPTIVE STATISTICS.**

Descriptive statistics can be used to answer the following questions:

-What do data look like from a distribution point of view? (i.e. what is the shape of the distribution?)	$MEAN, MEDIAN, MODE$	**CENTRAL TENDENCY**
-How variable is the data? (i.e. what is the spread among data-points?)	$STANDARD\ DEVIATION,$ $STANDARD\ ERROR, etc.$	**SPREAD**
-Can any *statistical* relationship be inferred for two given variables?	$COVARIANCE,$ $CORRELATION\ COEFF., etc.$	**RELATIONS**

DESCRIPTIVE STATISTICS

Arithmetic Mean

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$x = (2, 2, 3, 4, 9)$

$Mean \; = \; 4$

$Median \; = \; 3$ *Middle value separating the greater and lesser halves of a dataset [2,2,**3**,4,9]*

$Mode \; = \; 2$ *Most frequent value in a dataset (i.e. the one will be more frequently sampled)*

$x = (2, 3, 4, 6, 8, 9, 13)$

$Mean \; = 6.429$

$Median \; = (4 + 6 + 8)/3 = 6$

$Mode \; = None$

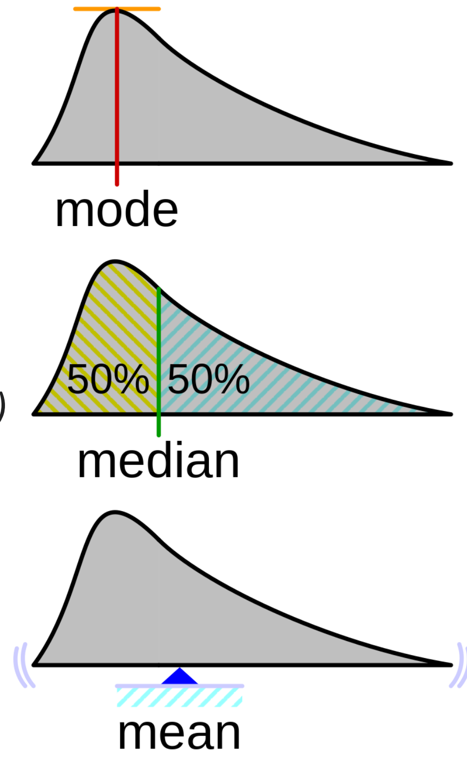


figure from: Wikipedia

DESCRIPTIVE STATISTICS

Geometric Mean

$$\bar{x}_g = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

Quadratic Mean / Root Mean Square (RMS)

$$\bar{x}_q = \sqrt{\frac{\sum_{i=1}^{n} (x_i)^2}{n}}$$

Harmonic Mean

$$\bar{x}_h = n / \sum_{i=1}^{n} \frac{1}{x_i}$$

DESCRIPTIVE STATISTICS

Weighted Arithmetic Mean

$$\overline{x_w} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

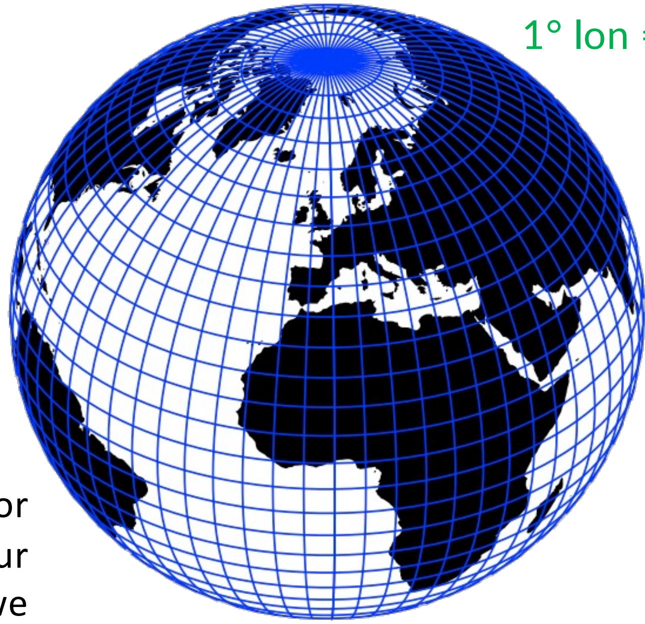
Note that if all elements have the same weight, then:
weighted mean = regular arithmetic mean

Grid-cells near the Poles cover a smaller area than grid-cells near the Equator because of **the meridian convergence**:

POLE 1° lat ~111.6 km
1° lon = 0 km

60°
1° lon ~ 56 km

EQ.
1° lon ~ 110.5 km
1° lat ~ 110.5 km



On a regular latitude-longitude grid the grid-spacing is not constant!
We must always use weighted means when computing global means or spatial means over large areas, if we fail to do so the results of our analysis will be hugely distorted in favor of the Poles (i.e. we overrepresent the contribution from the polar regions to the total mean).

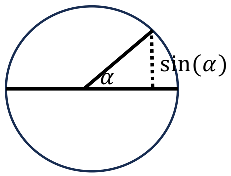
DESCRIPTIVE STATISTICS

Area Weighted Arithmetic Mean for Gridded data

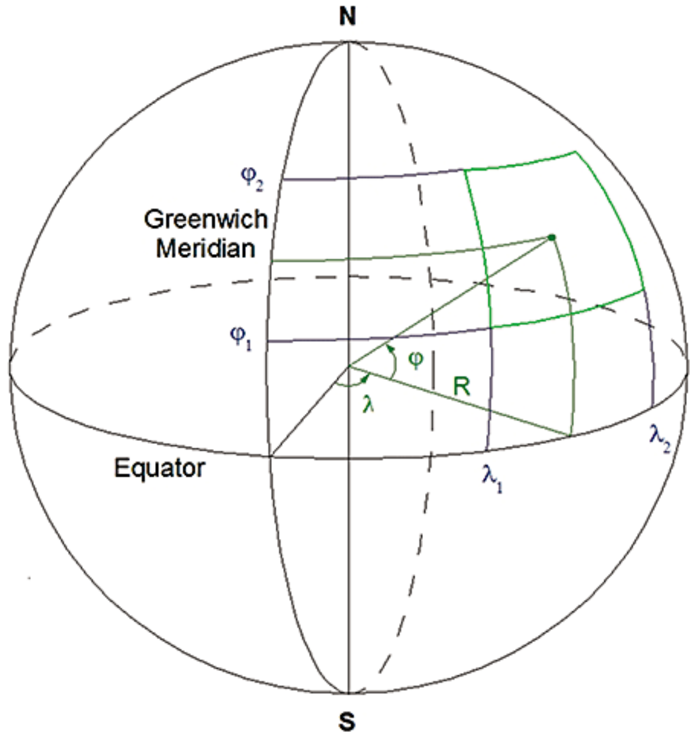
$$\overline{x_{aw}} = \frac{\sum_{i=1}^{n} x_i a_i}{A}$$

a_i : $grid - cell\ area$
A: $total\ area$

$$a_i = R^2(\lambda_2 - \lambda_1)(\sin(\varphi_2) - \sin(\varphi_1))$$



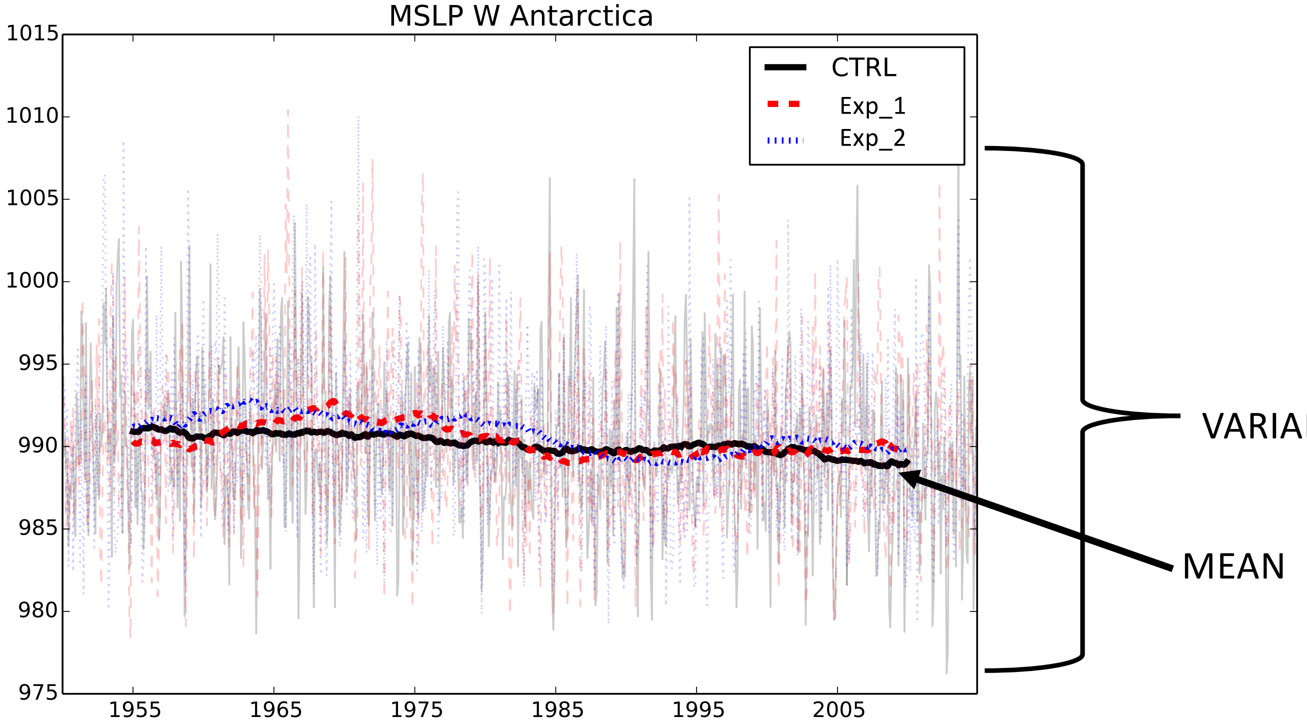
R: $Earth's\ radius$
λ: $longitude$
φ: $latitude$



Santini et al., 2010. Figure 2.
https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1467-9671.2010.01200.x

DESCRIPTIVE STATISTICS

How variable is my data?



Mean Sea Level Pressure (MSLP) in hPa over West Antarctica from 1950 to 2014. Thick line is annual mean with a 11-year running mean applied, gray dashed line is monthly mean.

Standard Deviation

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1\,*}}$$

SAMPLE

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

POPULATION

DESCRIPTIVE STATISTICS

$\boldsymbol{Variance}$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1\ *}$$

SAMPLE

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

POPULATION

$\boldsymbol{Coefficient\ of\ Variation\ (CV)}$

$$CV = \frac{s}{\bar{x}}$$

$$CV = \frac{\sigma}{\mu}$$

*for sample quantities we divide by $(n-1)$ instead of n to correct a bias due to the fact that we work with a sample and not with the population: the standard deviation (and the variance) depend on $\bar{x}$, we have lost one degree of freedom by using this piece of information.

DESCRIPTIVE STATISTICS

Pearson Correlation Coefficient (LINEAR)

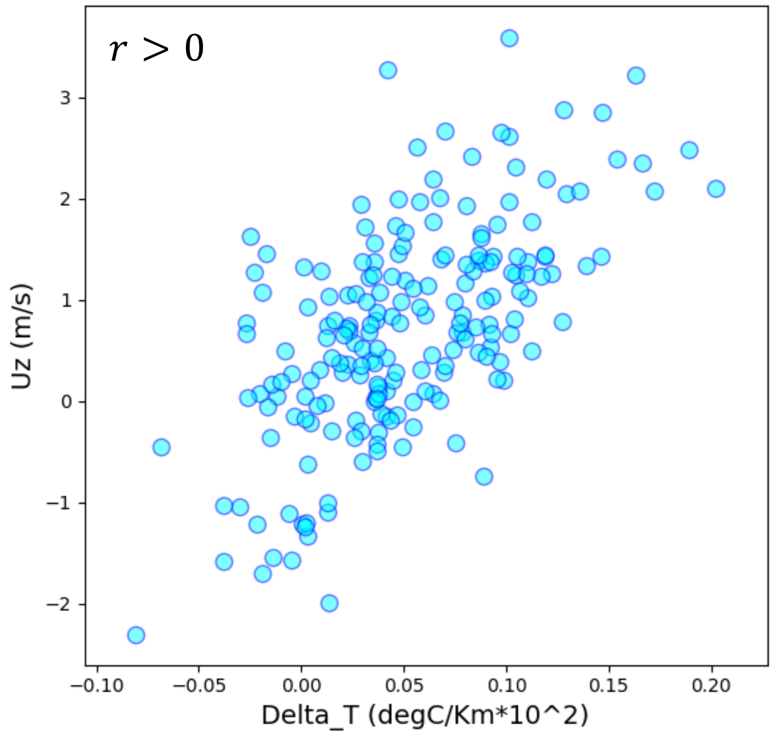
Do x_1 and x_2 vary linearly?
Is the change in x_1 proportional to the change in x_2 ?

$$r = \frac{\sum_i (x_{i1} - \bar{x}_1)(x_{i2} - \bar{x}_2)}{\sqrt{\sum_i (x_{i1} - \bar{x}_1)^2 \quad \sum_i (x_{i2} - \bar{x}_2)^2}}$$

$r = [-1,1]$ $\quad r > 0$ *positive correlation*
$r < 0$ *negative correlation*

$|r| > 0.6 - 7$ *good correalation*

$|r| > 0.8 - 9$ *strong correalation*



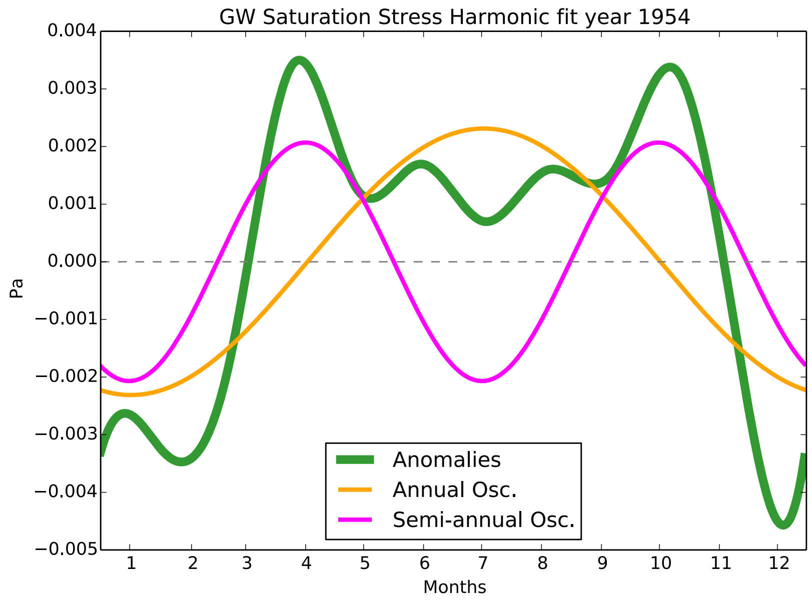
Scatter plot showing simulation outputs for: anomalies of zonal mean zonal wind (Uz, m/s) and meridional temperature gradient (°C/km) across the North Atlantic basin (area weighted mean) for a 200-year run.

DESCRIPTIVE STATISTICS

Spearman Correlation Coefficient

Do x_1 and x_2 vary monotonically?
x_1 and x_2 are still correlated but the rate of change is not necessarily the same.

Harmonic Fit performed on atmospheric gravity wave forcing over Antarctica for year 1954.
Is the anomaly signal better represented by the Annual (AO) or Semiannual (SAO) Oscillation?

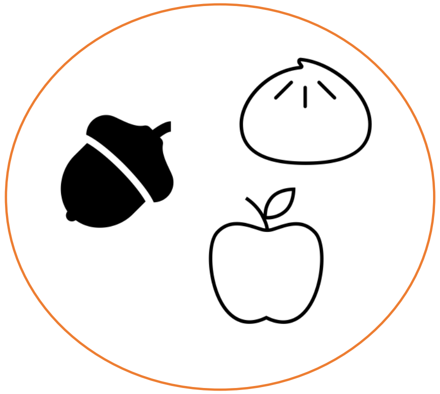


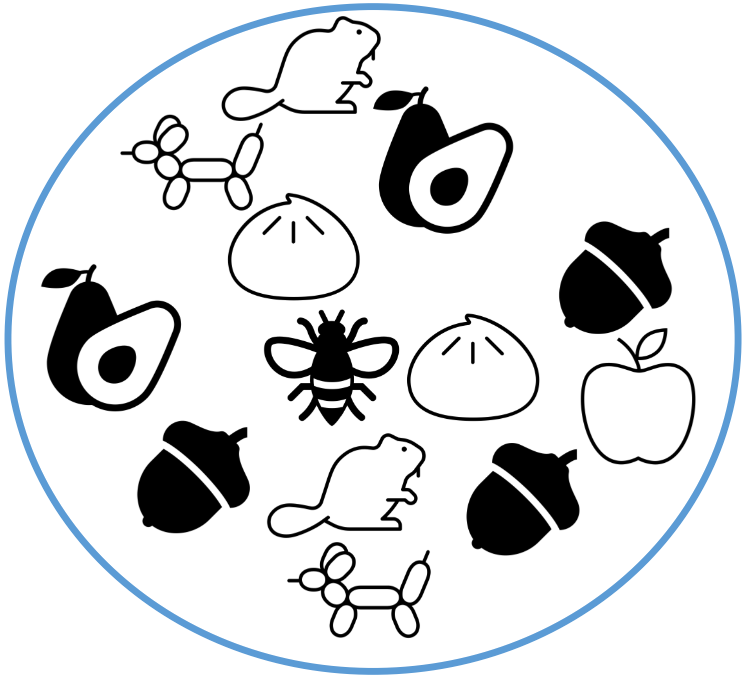

```
Pearson corr coeff AO 0.690717265664
Pearson corr coeff SAO 0.618165969483

Spearman corr coeff AO 0.506782685089
Spearman corr coeff SAO 0.701923632173
```

DESCRIPTIVE STATISTICS

Probability density function (p.d.f):

SAMPLE





POPULATION

Every time we collect data we work with samples and never with the whole population!

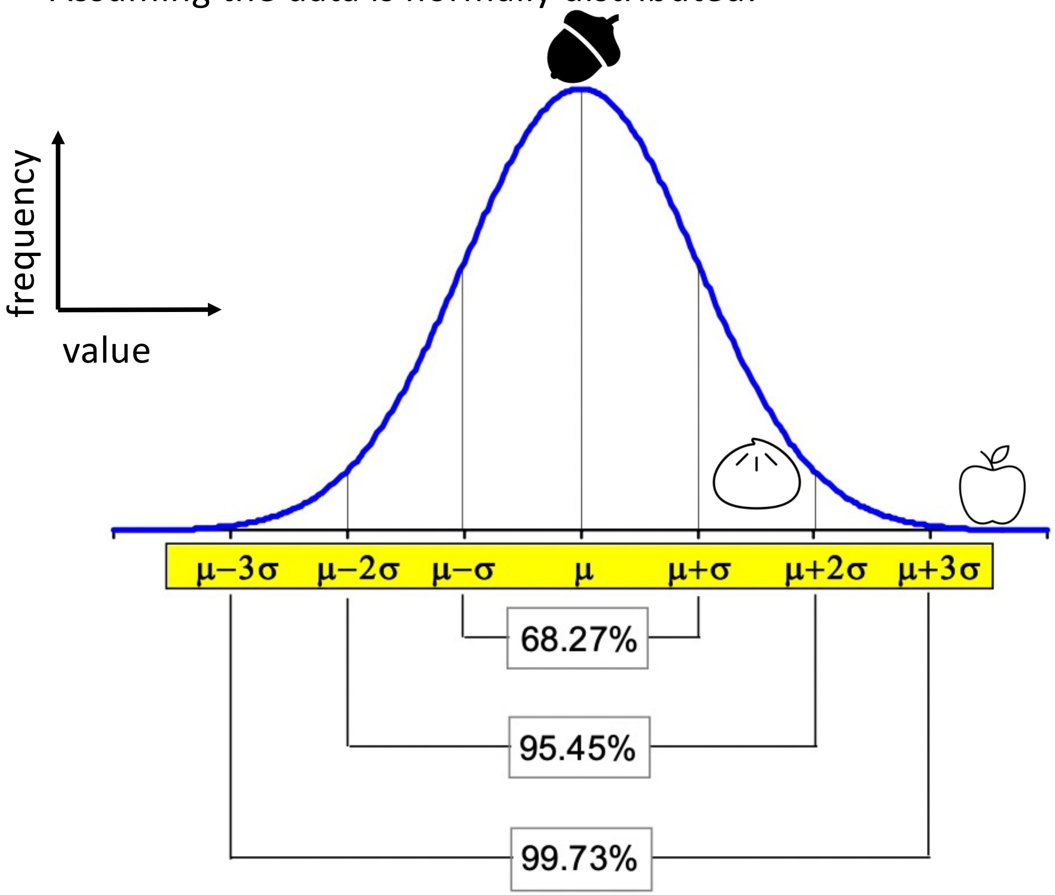
μ : population mean
$\bar{x}$: sample mean

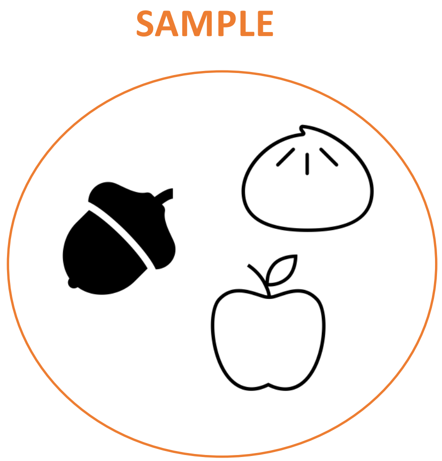
p.d.fs help us drawing universal conclusions from sampled data.
For example:
How lucky have I been to sample an acorn? And an apple?

DESCRIPTIVE STATISTICS

How lucky have I been to sample an acorn? And an apple?

Assuming the data is normally distributed:





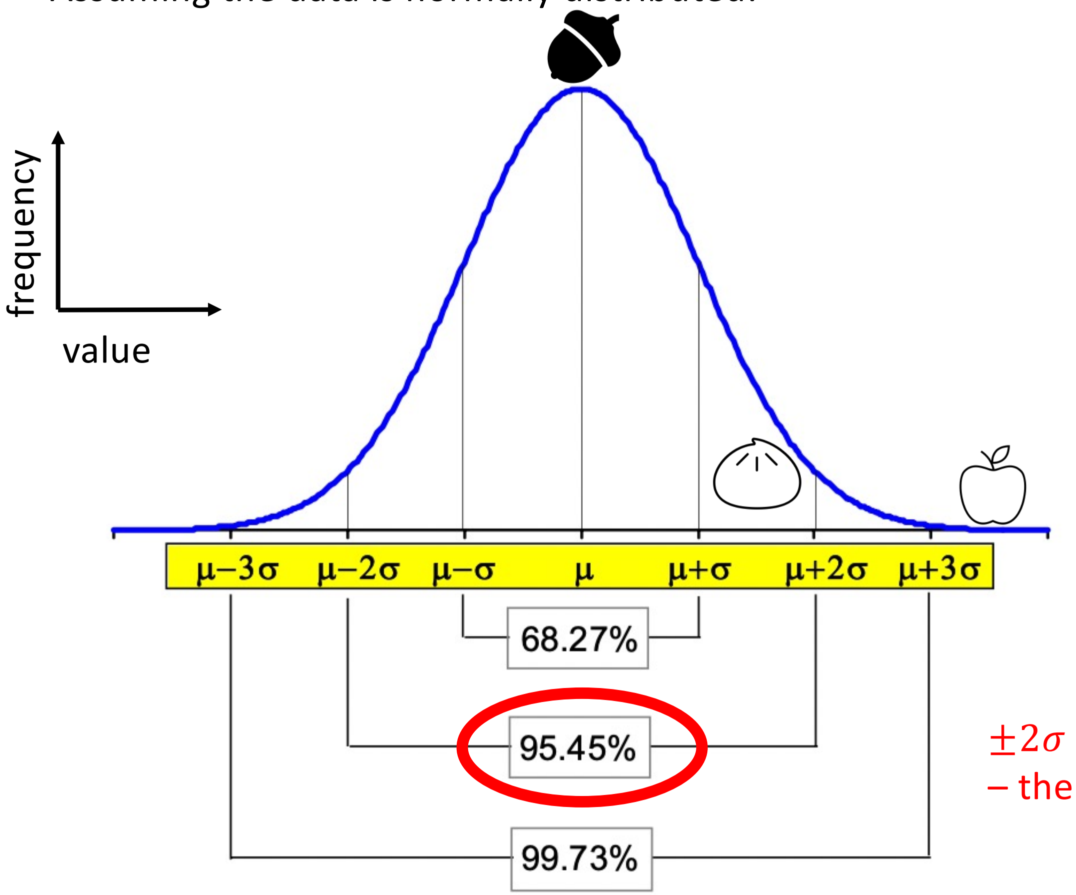
LAW OF LARGE NUMBERS:
For a given sample of independent variables following the same distribution, the sample mean converges to the true mean as the number of observations increases:

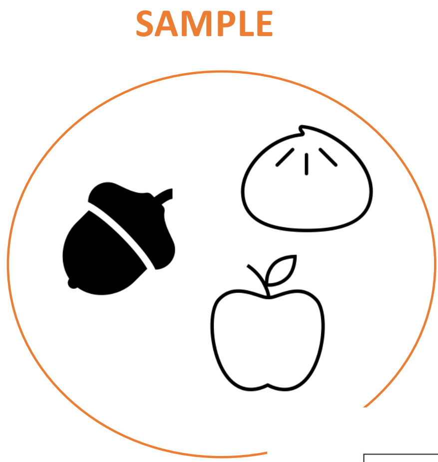
$$\bar{x}_n \underset{n \to \infty}{\longrightarrow} \mu$$

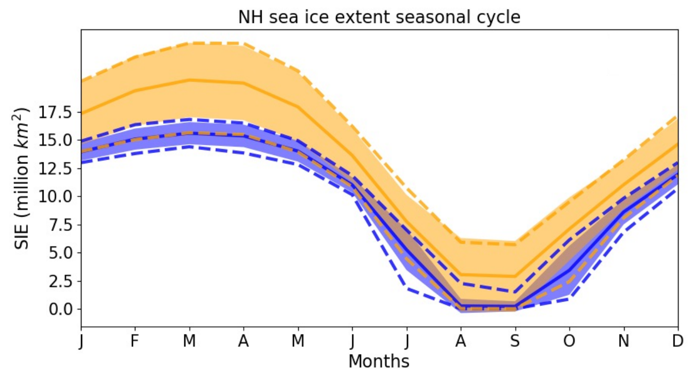
DESCRIPTIVE STATISTICS

How lucky have I been to sample an acorn? And an apple?

Assuming the data is normally distributed:







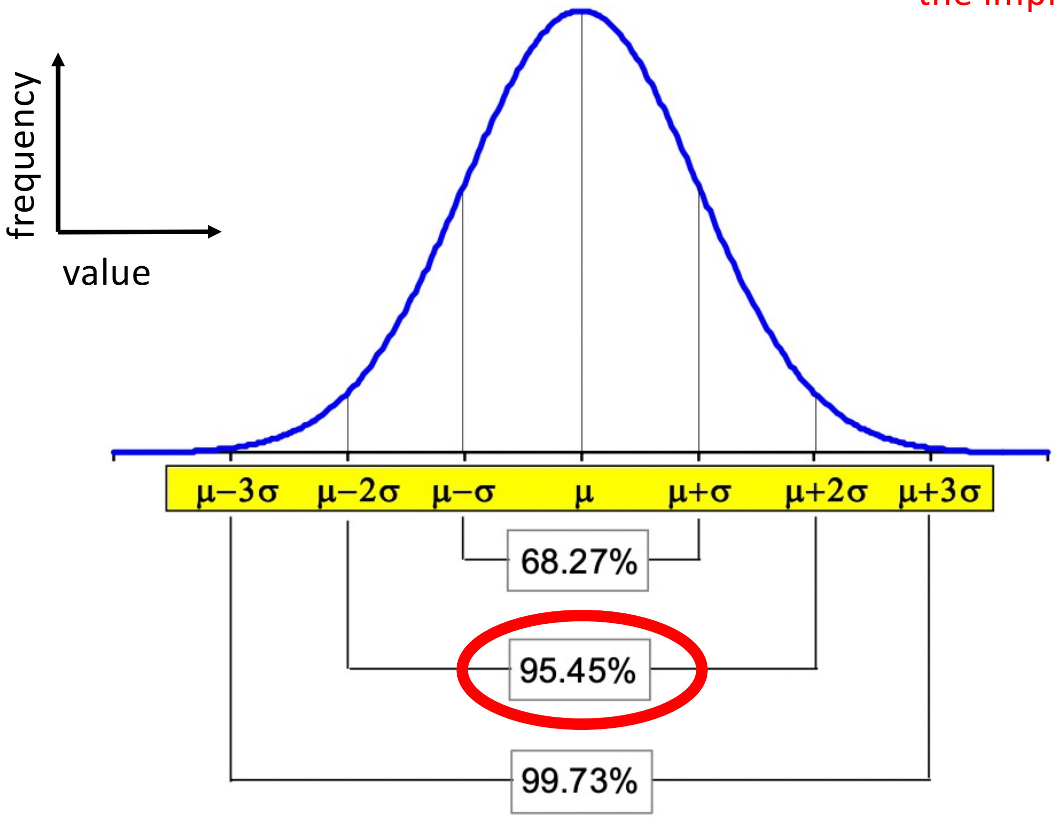
$\pm 2\sigma$ often used when presenting data
– the implicit assumption made is that the data is normally distributed

WHY?

DESCRIPTIVE STATISTICS

Assuming the data is normally distributed:

$\pm 2\sigma$ often used when presenting data
– the implicit assumption made is that the data is normally distributed



CENTRAL LIMIT THEOREM (CLT):

The CLT states that as the **number of samples increases**, the distribution of the sample means **approaches a normal distribution**, regardless of the shape of the original distribution.

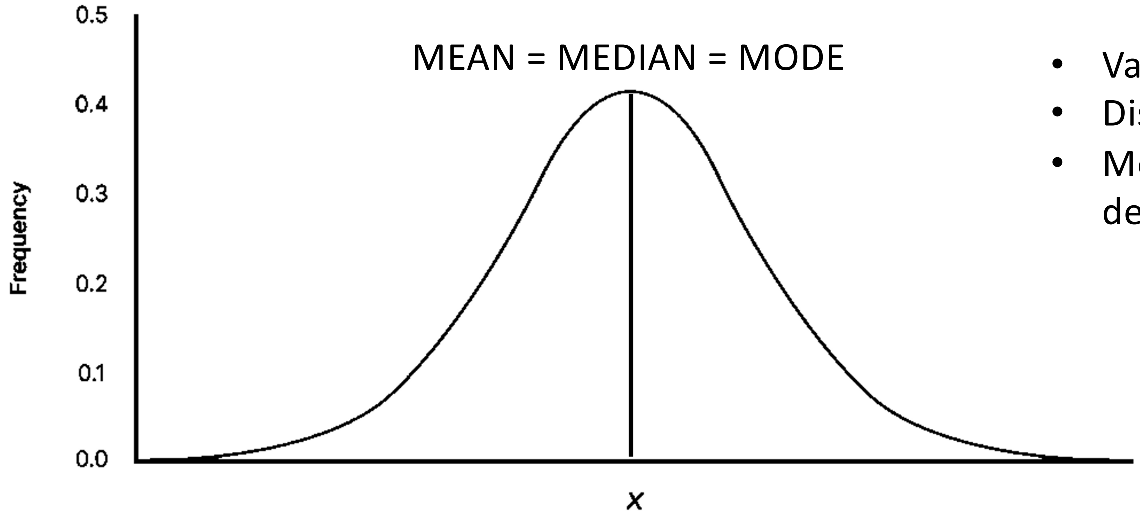
That is why the normal distribution is common in nature!
But note that the CLT is valid for a large number of samples:
the more data-points/samples we have the better.

DESCRIPTIVE STATISTICS

Distributions:

NORMAL / GAUSSIAN / BELL-SHAPED $N(\mu, \sigma)$

$$y = \frac{1}{\sqrt{2\pi \cdot \sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



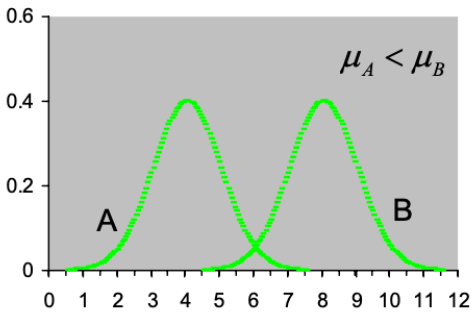
- Values around the mean are the most frequently occurring
- Distribution is symmetrical about the mean (no skew)
- Mean (μ) and standard deviation (σ) are enough to describe the distribution: $N(\mu, \sigma)$

DESCRIPTIVE STATISTICS

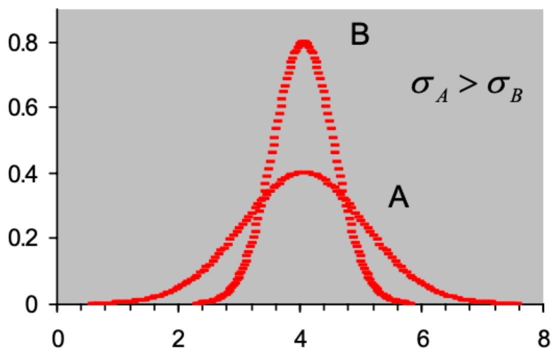
Distributions:

NORMAL / GAUSSIAN / BELL-SHAPED $N(\mu, \sigma)$

$$y = \frac{1}{\sqrt{2\pi \cdot \sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Same standard deviation, different mean



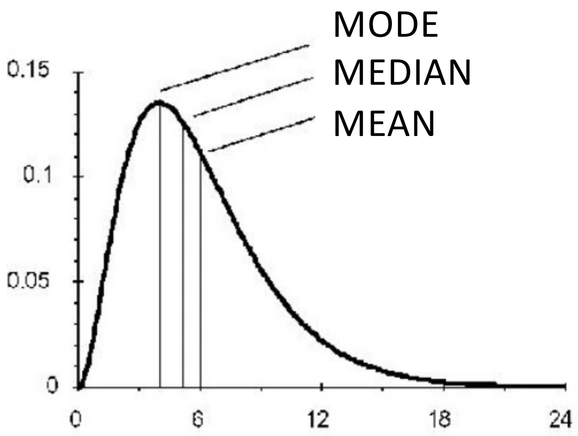
Different standard deviation, same mean

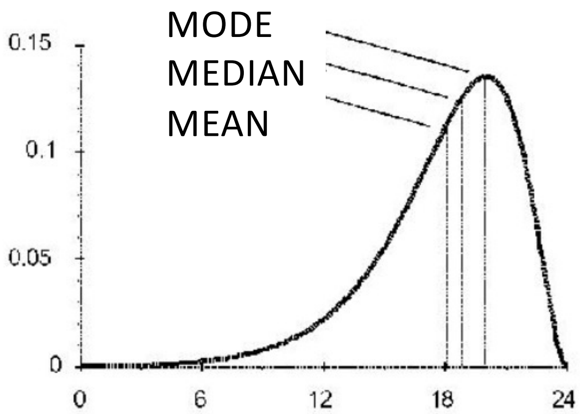
- Values around the mean are the most frequently occurring
- Distribution is symmetrical about the mean (no skew)
- Mean (μ) and standard deviation (σ) are enough to describe the distribution: $N(\mu, \sigma)$

DESCRIPTIVE STATISTICS

Distributions:

ASYMMETRICAL





MODE < MEDIAN < MEAN
Positive skew: the right tail is longer

MODE > MEDIAN > MEAN
Negative skew: the left tail is longer

Pearson's first coefficient of skewness (SK_1):

$$SK_1 = \frac{mean - mode}{std\_dev}$$

(others: Pearson's second coefficient of skewness (SK_2), Fisher-Pearson coefficient of skewness.)

HEAVLY SKEWED DISTRIBUTION MEANS MORE EXTREME VALUES BUT WITH LOW PROBABILITY

DESCRIPTIVE STATISTICS

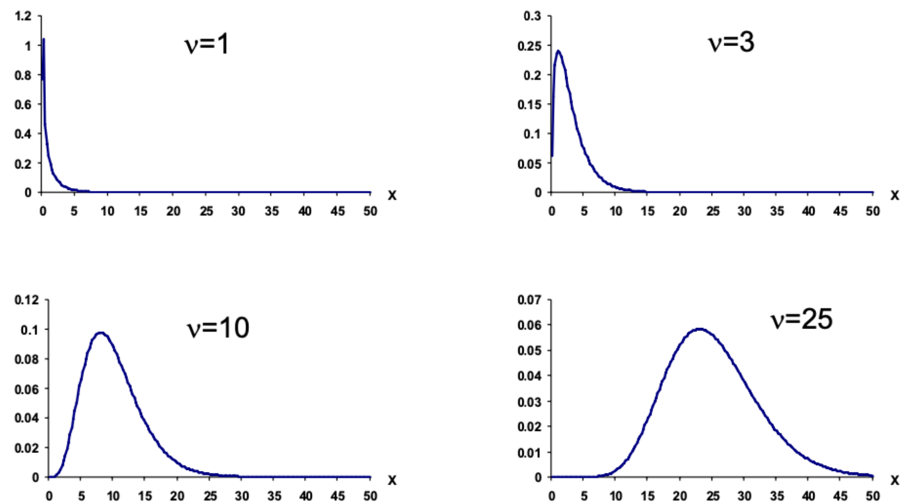
Distributions:

χ^2 CHI-SQUARED DISTRIBUTION

$$\chi^2_{(n)} = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^2} = \sum_{i=1}^{n} z_i^2$$

$$\nu = \mathrm{n}$$

$$\text{degrees of freedom} = \text{number of observations}$$



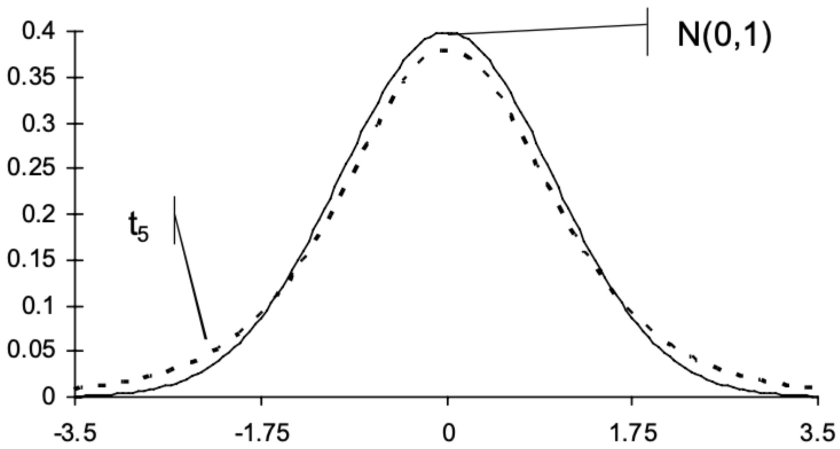
- Given n random variables, independent and normally distributed: $x_1, x_2, \ldots, x_n$, χ^2 is given by the sum of their squares.
- The shape of the χ^2 distribution is dependent on the degrees of freedom (ν) only.
- χ^2 takes different shapes from $\nu = 1$ to $\nu = 30$ when it becomes approximately normal.

DESCRIPTIVE STATISTICS

Distributions:

t DISTRIBUTION (Student's Distribution)

$$t_{n-1} = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}} \qquad (\nu = \mathrm{n} - 1)$$



Comparison between a Normal distribution (N(0,1)) and a t distribution with 5 degrees of freedom (t_5).

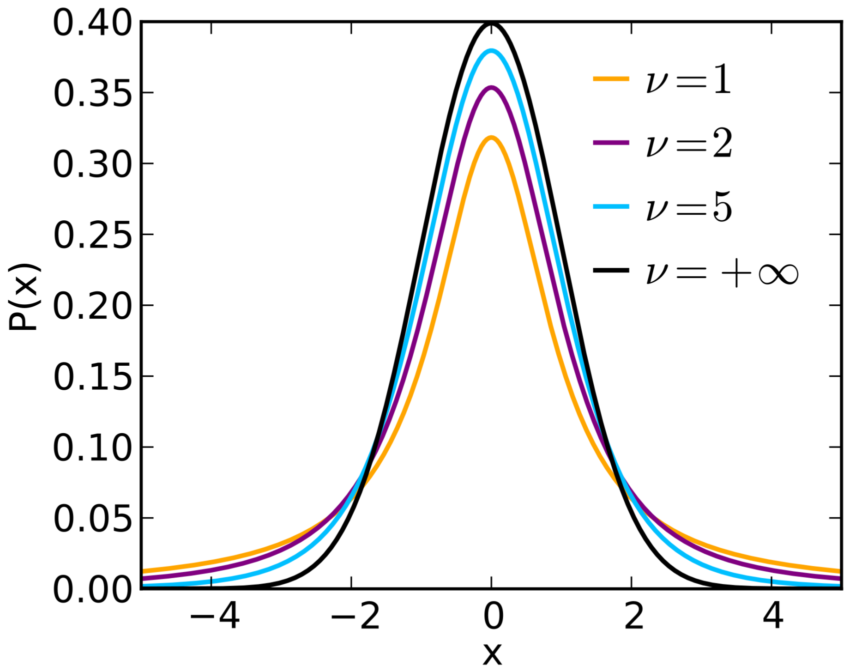
- It assumes that the observations come from a normally distributed population.
- It is for small sample sizes when the population variance is not known: it uses the sample mean ($\bar{x}$) and stdev (s).
- Depends on the degrees of freedom: $\nu = n - 1$.
- Similar to the normal distribution N(0,1) but with "heavier tails": extreme values occur more often.

DESCRIPTIVE STATISTICS

Distributions:

t DISTRIBUTION (Student's Distribution)

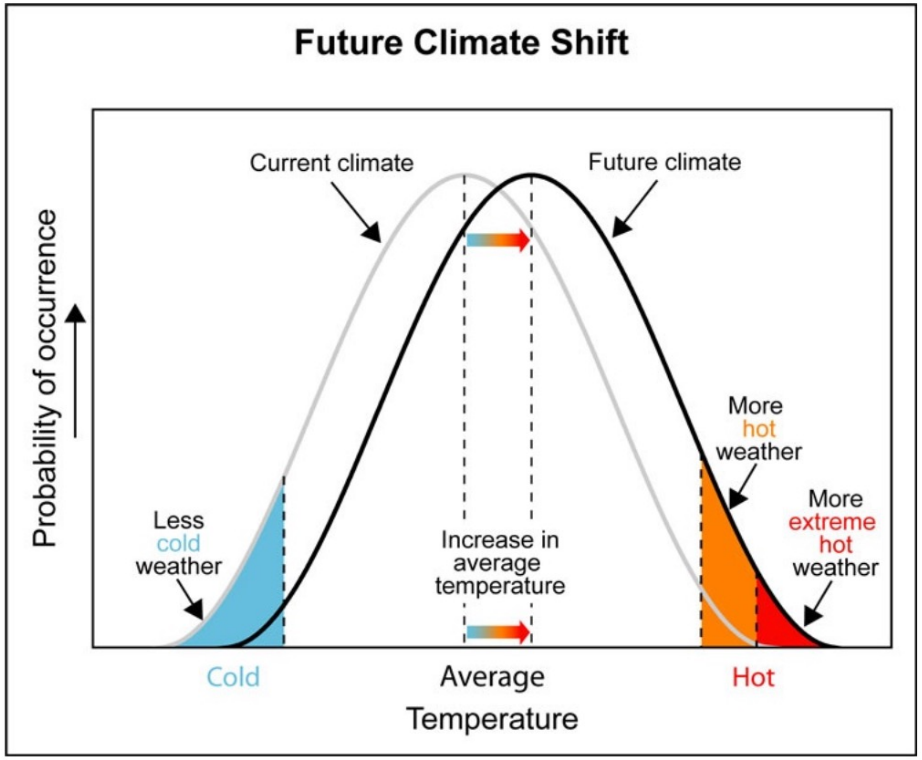
$$t_{n-1} = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}} \qquad (\nu = \mathrm{n} - 1)$$



t distribution for different degrees of freedom
(figure from: Wikipedia)

- It assumes that the observations come from a normally distributed population.
- It is for small sample sizes when <u>the population variance is not known</u>: it uses the sample mean ($\bar{x}$) and stdev (s).
- Depends on the degrees of freedom: $\nu = n - 1$.
- Similar to the normal distribution N(0,1) but with "heavier tails": extreme values occur more often.
- The amount of probability mass within the tails is decided by ν. For $\nu \rightarrow \infty$ the t distribution tends to a Normal distribution.

T-TEST (based on the t distribution) is the MOST WIDELY USED STATISTICAL SIGNIFICANCE TEST (`scipy.stats.ttest_ind`)



Source: US Climate Change Science Program.

Skeptical Science:
https://skepticalscience.com/Review-Rough-Winds-Extreme-Weather-Climate-Change-James-Powell.html

Paçal, Aytaç, et al. "Detecting extreme temperature events using Gaussian mixture models." *Journal of Geophysical Research: Atmospheres* 128.18 (2023): e2023JD038906.

https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2023JD038906

Python Coding: Tips and Tricks

- Always ***look at*** your dataset and visualize it before start writing any code.

What is the shape? the length? How many dimensions? NaNs? Infs? Masked?
Do you have a clear picture in your mind of what type of data you are dealing with?

- **`print`** is your friend.

`print print print print !!` always check your code is doing what you wanted it to do.

- Use **`for`** loops only if strictly necessary – looping is time consuming for large datasets.

- Make your code tidy and as universal as possible: i.e., define your functions (**`def my_function()`**) and call them as you analyze your data.

- The problem you are having someone else already had: ***stackoverflow*** ? (python community is huge and help is available online).

- Create your own **`GitHub`** repository (?)

Exercises

1. Use the provided dataset (a time-series) to compute the mean (example provided) and the standard deviation writing your own functions: `ERA5_2m_SAT_TS_1990_2023.txt`

2. Compare your results with the outputs from the python built-in functions: `np.average` and `np.std`

1. Now repeat the above two steps but using the following input dataset: `ERA5_2m_SAT_TS_1990_2023_nan.txt`
   How does np.average behave when nan values are present in the input array? (use np.nanmean instead)

4. In case of nan values, what alternative do we have besides using np.nanmean np.nanstdev? (tip: mask invalid data-points and use again np.average and np.stdev).

5. Make a line plot of the input time-series.

Exercises

numpy.average (weights are allowed) https://numpy.org/doc/stable/reference/generated/numpy.average.html

numpy.mean https://numpy.org/doc/stable/reference/generated/numpy.mean.html

numpy.nanmean https://numpy.org/doc/stable/reference/generated/numpy.nanmean.html

numpy.std https://numpy.org/doc/stable/reference/generated/numpy.std.html

numpy.nanstd https://numpy.org/doc/stable/reference/generated/numpy.nanstd.html

Note: both `numpy.average` and `numpy.stdev` deal with masked data-points by ignoring them (i.e. masked entries are not used in the calculations). If NaNs are present you can mask them before passing the dataset to np.average and np.stdev. The result will be the same as using np.nanmean np.nanstd.

Exercises

1. Use the provided dataset (a time-series) to compute the mean and the standard deviation writing your own functions: `ERA5_2m_SAT_TS_1990_2023.txt`

ERA5 2m Air Temperature for Trieste from 1990 to 2023:

```
Year    month   T(K)
[1990]   [1]   276.74615
[1990]   [2]   280.38818
[1990]   [3]   282.6789
[1990]   [4]   283.4358
[1990]   [5]   289.6418
[1990]   [6]   291.34448
[1990]   [7]   293.84976
[1990]   [8]   294.12073
[1990]   [9]   289.6087
[1990]   [10]   286.8277
[1990]   [11]   281.59503
[1990]   [12]   276.80756
[1991]   [1]   276.79822
[1991]   [2]   275.0808
[1991]   [3]   282.33533
[1991]   [4]   283.1449
[1991]   [5]   285.22668
[1991]   [6]   291.24957
```

....

Exercises

Example Script: **DA_exercise_1.py**

```
import iris

import numpy as np
import matplotlib.pyplot as plt

import scipy
from scipy import stats

###### Define Your Functions  Here ############

 …

############### Load datasets and call functions #############
#Load txt file and call functions
data_in=np.loadtxt('ERA5_2m_SAT_TS_1990_2023.txt', usecols=2)
#print the first 15 lines of file
print (data_in[:15])

 mean=compute_mean(data_in)
 stdev=compute_stdev(data_in, mean)
```

Exercises

Example Function for the mean:

```
###### Define Function to compute mean of input dataset ############
#working with ASCII files
def compute_mean(data_in):
```

Method 1

```
 mean=np.sum(data_in)/len(data_in)
 print (mean)
```

Method 2 (your own function)

```
 i_sum=0
 for i in range (0,len(data_in)):
  i_sum=data_in[i]+i_sum

 mean=i_sum/len(data_in)
 print (mean)
```

Method 3

```
 mean=np.average(data_in)
 #mean=np.nanmean(data_in)
 print (mean)


 return(mean)
```

Exercises

The same file but as a NETCDF: ERA5_2m_SAT_TS_1990_2023.nc

```
>>> import iris
>>> cube=iris.load_cube('ERA5_2m_SAT_TS_1990_2023.nc')
>>> print (cube)
air_temperature / (K)                 (time: 408)
    Dimension coordinates:
        time                               x
    Scalar coordinates:
        latitude                      45.75 degrees
        longitude                     13.75 degrees
    Attributes:
        CDI                           'Climate Data Interface version 2.2.1 (https://mpimet.mpg.de/cdi)'
        CDO                           'Climate Data Operators version 2.2.0 (https://mpimet.mpg.de/cdo)'
        Conventions                   'CF-1.7'
        cds_magics_style_name         'near-surface-air-temperature'
        comment                       'near-surface (usually, 2 meter) air temperature'
        history                       'Mon Mar 04 14:33:57 2024: cdo mergetime ERA5_2m_SAT_TS_1990_1999.nc E
        institution                   'European Centre for Medium-Range Weather Forecasts'
        source                        'ECMWF'
        type                          'real'
```

Exercises

The same file but as a NETCDF: ERA5_2m_SAT_TS_1990_2023.nc

```
#working with NetCDF files
def compute_mean_nc(data_in):

 i_sum=0
 for i in range (0,len(data_in.data)):
  i_sum=data_in[i].data+i_sum

 mean=i_sum/len(data_in.data)
 print (mean)


 mean=data_in.collapsed('time', iris.analysis.MEAN)
 print (mean.data)


 return(mean)
```

```
############### Load datasets and call functions #############

#use iris to load netcdf file
data_in=iris.load_cube('ERA5_2m_SAT_TS_1990_2023.nc')
print (data_in)
mean=compute_mean_nc(data_in)
stdev=compute_stdev_nc(data_in)
```