Data Analysis and Hypothesis Testing Using the Python ecosystem

Distributions

Stavros Demetriadis
Assc. Prof., School of Informatics,
Aristotle University of Thessaloniki

sdemetri@csd.auth.gr
http://mlab.csd.auth.gr/sdemetri

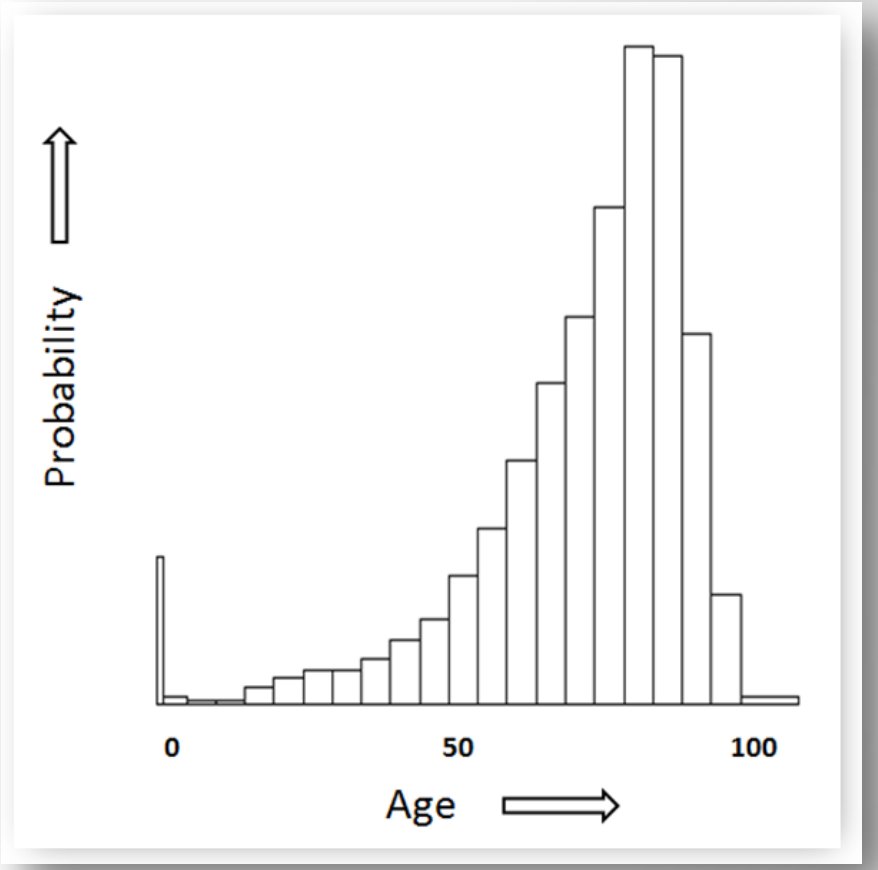
Overview

- Distributions
- The normal distribution
- The t distribution
- The f distribution

What is a distribution

- A **probability distribution** (or simply 'distribution') of a variable is an expression of the ***probability*** *that when measuring the variable the result will lie within a specific range of values.*

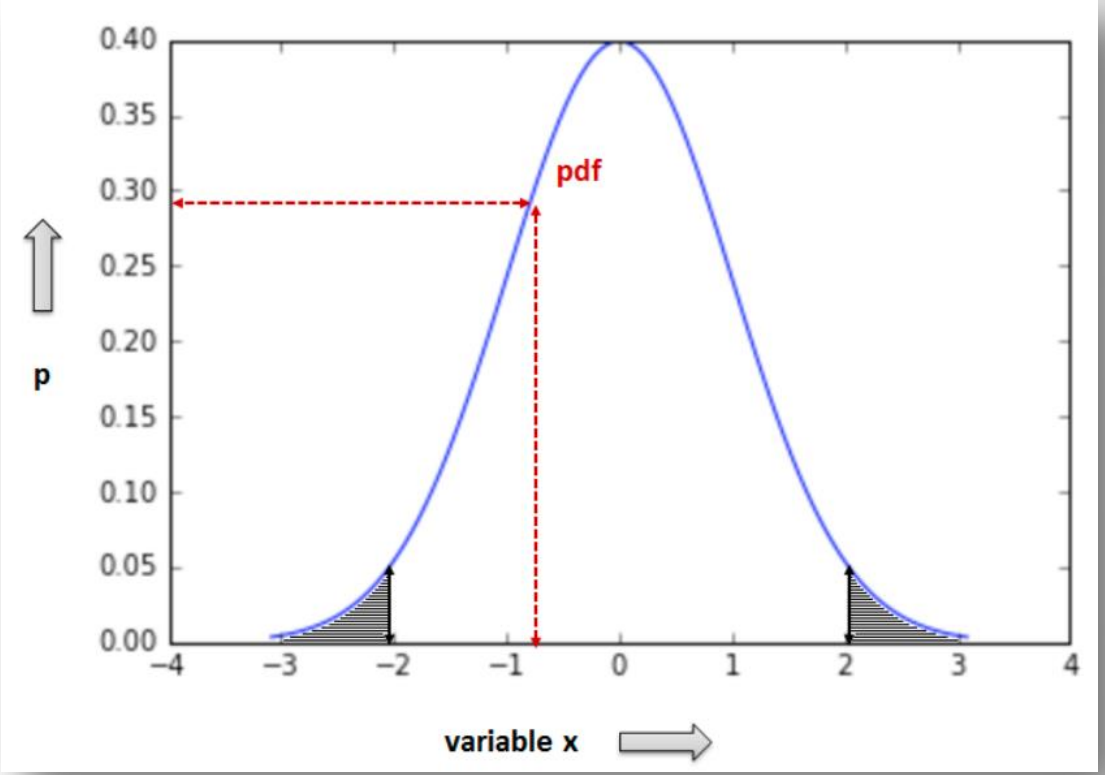
- *Discrete:* The variable can take *only specific countable levels of values*
- Continuous: The variable can take *any value ('varies continuously')* between a min and a max value



Probability density function (pdf)

- **pdf** returns *the probability that the variable x lies between a range of values*

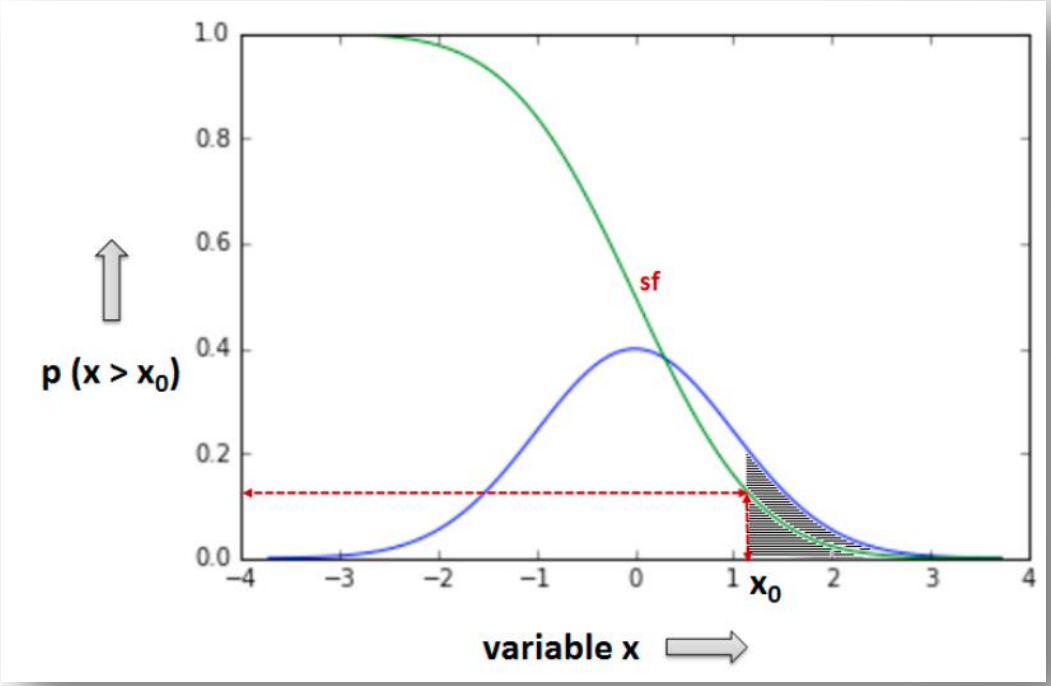
- For example, pdf(2, +inf) returns the probability that x lies in the range (+2, +inf)



Survival function (sf)

- **sf** returns the probability that *the variable x gets a value that is greater than a specific value* x_0

- For example, sf(x_0) returns the probability that x lies in the range (x_0, +inf)

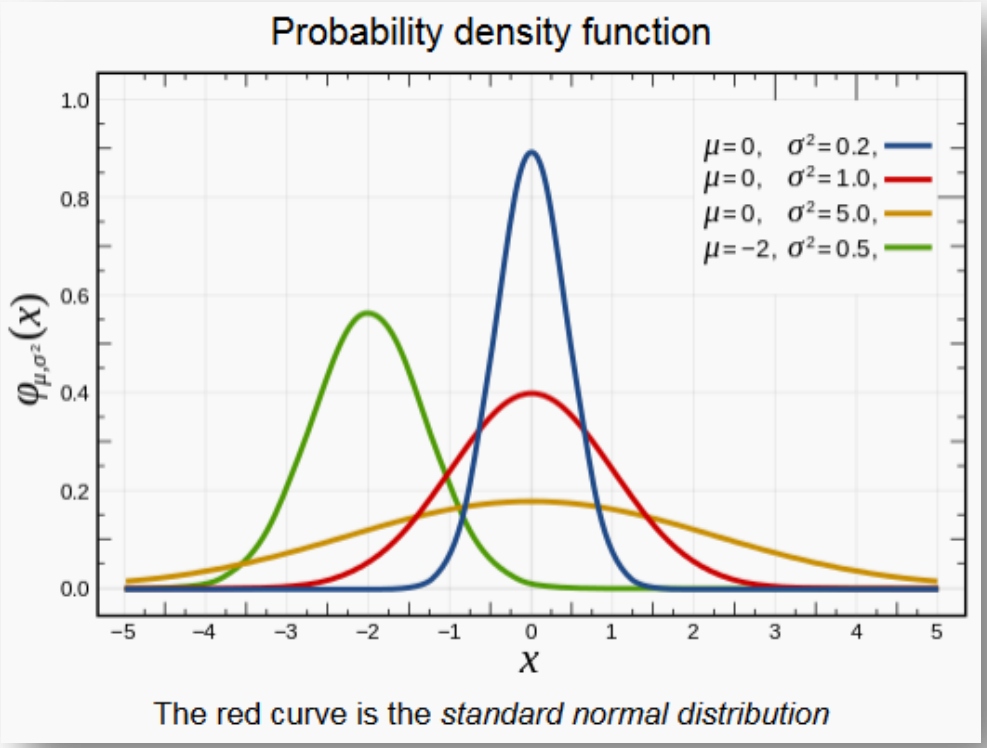


Please note that in this graph the vertical axis has values from 0 to 1

(which is different from the probability values in the pdf graph in the previous slide)

The normal distribution

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



The red curve is the *standard normal distribution*

- The probability density of the normal distribution, where:
- **μ**: the *mean* of the distribution
- **σ**: standard deviation
- **σ²**: variance

- Useful because of the **central limit theorem**: averages of random variables **become normally distributed** when the number of random variables is sufficiently large.
- See example representation in Wikipedia

Normal distribution in Scipy

- Using the Scipy tools = Practicing scientific computing in the Python ecosystem
- Distributions are part of the **scipy.stats** module
- To import scipy:
  - **import scipy as sc**
  - **import scipy.stats as stats**
  - **from scipy.stats import norm**

Normal distribution in Scipy

- Simple statistical functions
  - **median** → *median()*
  - **mean** → *mean()*
  - **standard deviation** → *std()*
  - **variance** → *var()*

```
from scipy.stats import norm

# get median, mean, standard deviation and var
print(norm.median(), norm.mean(), norm.std(), norm.var())

# or use stats() function
m, v, s, k = norm.stats(loc=0, scale=1, moments='mvsk')
print(m, v, s, k)
```

Normal distribution in Scipy

- **Standardized** distribution: using the z scores
  - Z = (x-M)/SD
  - So, for a standardized distribution: **Mean = 0 , SD = 1**
  - Mean is referred to as 'loc', std → 'scale'

- **'Frozen'** distribution
  - Construct a distribution with **specific M and SD to use**

```
rv = norm(loc=2, scale=4)    # Use 'rv'
m, v, s, k = rv.stats(moments='mvsk')
print( m, v, s, k)
```

```
2.0 16.0 0.0 0.0
```

Normal distribution in Scipy

- **rvs** : get random values from the distribution

```
from scipy.stats import norm
obs = norm.rvs(loc=0, scale=1, size=10)
obs
```

- **Plotting**: How to plot a distribution using pdf()

```
# A better approach

rv = norm(loc=0, scale=1)                    # Freeze the norm
x = np.linspace(rv.ppf(0.0001), rv.ppf(0.9999), 100)
y = rv.pdf(x)
plt.plot(x,y)
```

Normal distribution in Scipy

- How to find whether the probability of appearance for a specific value is **less than a threshold probability 'a'**

```
a=0.05
nv = st.norm(loc=2, scale=1.2)
x = nv.rvs(1)
p = nv.sf(x)

if p <= a:
    print('significant')
```

- For any specific value in the distribution (x in the example) call the **sf()** function to return the value of probability p
- Compare p to threshold a

Review exercises

- Distributions

- Preparation:

```
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
%matplotlib inline
```

- 1. Construct a normal distribution ‘nv’ (from: normal variate’) with M = 2 and SD = 1.2. Then get 20 random values from this distribution and print them onscreen *(hint: use st.norm() and rvs())*

- 2. Use *describe()* to get some statistics of the above 20 values. Is the mean and the variance close enough to the preset values of the ‘nv’ distribution? Why? What would you do to get a closer estimation?

- 3. Alternative use stats() to get some statistics of ‘nv’. Are these close enough to the presets? Why don’t we observe here any divergence as before with describe()?

- 4. Plot the distribution ‘nv’ as follows:
  - A) Construct a linspace() with start = -5 and stop = +5. What do you observe in the plot?
  - B) Then use ppf() to improve construction of the linspace()
  - C) Finally, use xticks() to present on the horizontal axis the –sd, M and +sd points

- 5. Write a loop of 100 iterations. In each iteration a random value from the ‘nv’ distribution is returned and its probability is compared to a threshold value a = 0.05. If p <= a then print the random value. At the end of the loop print how many such values were drawn during the iterations. What is the percentage of these values (within the total 100 iterations) when compared to threshold a?

- 6. Repeat the above 1-5 steps with the t and F distribution. In these distributions you will also need to assign a value to the **‘degrees of freedom’** parameter.