Python for Hydrology

Session 6 – Statics I

Objective:

Develop regression analysis, develop statistical distributions for different climate stations, and calculate return periods for rainfall.

Regression line

Start by creating the notebook and the folder of Session 6.

Import the libraries to be used in this Session.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime
import scipy.stats as st
```

Start creating random data:

- "n" represents the number of data
- "x" represents random values of the size "n"
- "y" represents the line equation, but here we are adding more random data to create a dispersion of values.

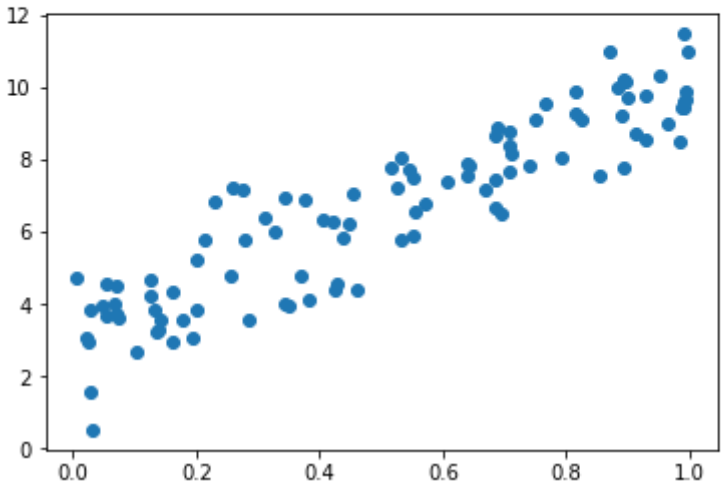
```
n = 100
x = np.random.rand(n)
y = 3 + 7*x + np.random.randn(n)
```

If we plot this, we get the following view:

```
plt.scatter(x,y)
```

```
n = 100
x = np.random.rand(n)
y = 3 + 7*x + np.random.randn(n)

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x1966c95aa90>
```



With Scipy, we can perform the line regression easily. But this returns five variables:

- Slope
- Intercept
- Correlation Coefficient
- P value
- Standard Error

```
slope, intercept, CorrelationCoefficient, Pvalue , StandardError =
st.linregress(x, y)
print(slope,intercept, CorrelationCoefficient, Pvalue , StandardError)
```

```
slope, intercept, CorrelationCoefficient, Pvalue , StandardError = st.linregress(x, y)
print(slope,intercept, CorrelationCoefficient, Pvalue , StandardError)
```

```
6.840948755412061 3.0324520601523846 0.890435018722499 2.8179426077637116e-35 0.3531976693677505
```

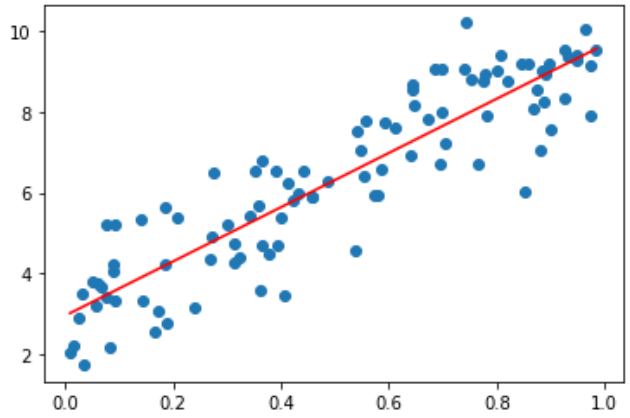
Now, we can make a quick plot of the regression line.

```
plt.scatter(x,y)
```

```
plt.plot([x.min(),x.max()],[slope*x.min()+intercept,slope*x.max()+inte
rcept],color='red')
```

```
plt.scatter(x,y)
plt.plot([x.min(),x.max()],[slope*x.min()+intercept,slope*x.max()+intercept],color='red')
```

```
[<matplotlib.lines.Line2D at 0x29835b4afa0>]
```



We can calculate the errors related to this regression. The following proceeding is useful when you have modeled vs. observed values.

First, we create an array with the errors associated with the actual Y values and the regression line's values.

```
eps = y - intercept - slope*x
```

```
eps = y - intercept - slope*x
eps
```

```
array([ 0.21923804,  0.83187471, -0.05992707, -0.58198945, -0.32840067,
        2.89070941, -0.52678126,  0.99535317,  0.68909029, -2.50372152,
       -1.37549594,  0.72194885, -0.74727915,  0.92974743, -0.84690296,
       -0.08780589,  0.76652254, -1.20475674, -1.18584037,  1.67690175,
        0.03042571, -1.75673198,  1.16895239, -0.62877243,  0.21582876,
       -0.6031651 ,  1.27526767, -1.12792441, -0.7659901 ,  2.29075046,
        0.46187366, -1.17539455, -1.91021019, -0.26540228, -0.34947568,
       -1.02207529, -0.58115539,  0.48900017,  1.27934121, -0.36349379,
       -0.69792249, -0.18882989, -1.19790459,  2.19774918,  0.39290707,
       -0.25553967, -0.75877445, -0.3943093 , -0.41499666, -0.76694252,
        0.27085211, -0.27245421, -0.71877769,  1.23709846, -0.46900131,
        1.34728483,  0.68484234, -0.09637445,  0.35232446, -0.65945175,
       -1.01357836,  0.37189509,  0.44442839,  0.07126696,  0.29959296,
        0.23993601, -0.71347411, -1.46347784,  0.07230814,  0.79563376,
       -1.33602943,  0.9617866 , -2.24518281, -0.8605711 , -1.33767622,
       -0.08301304,  2.25447222,  0.55336673,  0.3189165 , -0.93978658,
       -0.51576595,  0.84731782,  0.90241835, -0.36405923,  1.17314216,
       -1.13314384, -0.58008065,  1.67123092, -0.94212294,  1.06954936,
        1.44208228,  0.97224506,  0.91347636, -1.69158301,  1.54807505,
        0.23141513, -1.11269099,  0.38635294,  1.84176177,  0.42365205])
```

We need to create a range of values equal to the length of the graph

```
x1 = np.linspace(0, 1)
x1
```

```
x1 = np.linspace(0, 1)
x1
```

```
array([0.        , 0.02040816, 0.04081633, 0.06122449, 0.08163265,
       0.10204082, 0.12244898, 0.14285714, 0.16326531, 0.18367347,
       0.20408163, 0.2244898 , 0.24489796, 0.26530612, 0.28571429,
       0.30612245, 0.32653061, 0.34693878, 0.36734694, 0.3877551 ,
       0.40816327, 0.42857143, 0.44897959, 0.46938776, 0.48979592,
       0.51020408, 0.53061224, 0.55102041, 0.57142857, 0.59183673,
       0.6122449 , 0.63265306, 0.65306122, 0.67346939, 0.69387755,
       0.71428571, 0.73469388, 0.75510204, 0.7755102 , 0.79591837,
       0.81632653, 0.83673469, 0.85714286, 0.87755102, 0.89795918,
       0.91836735, 0.93877551, 0.95918367, 0.97959184, 1.        ])
```

We need to use the following formula to create the variance of the fitting error.

$$\sigma^2_{pred} = E[(Y_{pred} - \widehat{Y})^2] = \sigma^2_{\varepsilon}\left(1 + \frac{1}{n} + \frac{(X_0 - \overline{X})^2}{\sum_{i=1}^{n}(X - \overline{X})^2}\right).$$

```
e_pi = np.var(eps)*(1+1.0/n + (x1-x.mean())**2/np.sum((x-
x.mean())**2))
e_pi
```

```
e_pi = np.var(eps)*(1+1.0/n + (x1-x.mean())**2/np.sum((x-x.mean())**2))
e_pi
```

```
array([1.14746378, 1.14490889, 1.14245583, 1.14010461, 1.13785523,
       1.13570768, 1.13366196, 1.13171808, 1.12987604, 1.12813583,
       1.12649746, 1.12496093, 1.12352623, 1.12219336, 1.12096233,
       1.11983314, 1.11880578, 1.11788026, 1.11705657, 1.11633472,
       1.1157147 , 1.11519652, 1.11478018, 1.11446567, 1.114253  ,
       1.11414216, 1.11413316, 1.11422599, 1.11442066, 1.11471716,
       1.1151155 , 1.11561568, 1.11621769, 1.11692154, 1.11772722,
       1.11863474, 1.11964409, 1.12075528, 1.12196831, 1.12328317,
       1.12469986, 1.12621839, 1.12783876, 1.12956096, 1.131385  ,
       1.13331088, 1.13533859, 1.13746813, 1.13969951, 1.14203273])
```

We calculate the percent point function (PPF), which is the cumulative distribution function's inverse. We calculate for the lower probability equal to 0.01, keeping the size of the sample equal to "n-1"

```
zl = st.t.ppf(0.01, n-1)
zl
```

```
zl = st.t.ppf(0.01, n-1)
zl
```

```
-2.364605861435974
```

And the PPF for the lower probability equal to 0.90

```
zu = st.t.ppf(0.90, n-1)
zu
```

```
zu = st.t.ppf(0.90, n-1)
zu
```

```
1.2901614420275025
```

Then we calculate the line regarding the probability equal to 10%. We calculate its "Y" values based on the line regression parameters and the variance of the predicted values. As we calculated the squared variance, we need to calculate its squared root and multiply it by the PPF.

```
ll = intercept + slope*x1 + np.sqrt(e_pi)*zl
ul = intercept + slope*x1 + np.sqrt(e_pi)*zu
```

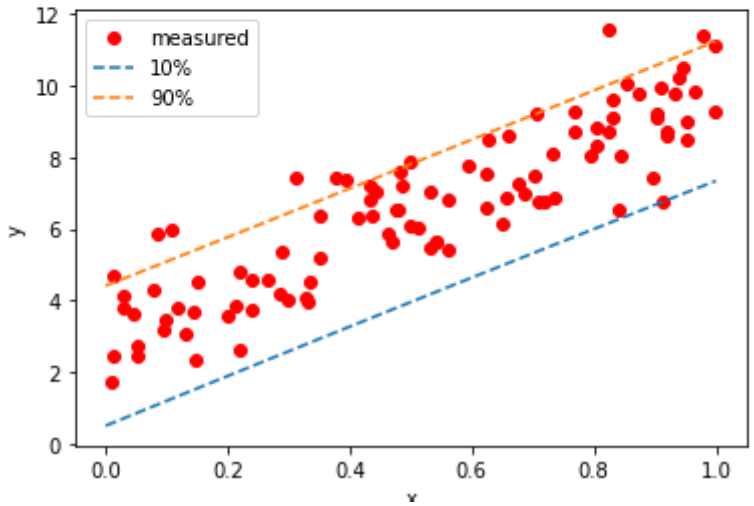

```
ll = intercept + slope*x1 + np.sqrt(e_pi)*zl
ul = intercept + slope*x1 + np.sqrt(e_pi)*zu
```


Then we can make a plot of both figures

```
fig, ax = plt.subplots()
ax.plot(x,y,'ro', label='measured')
ax.plot(x1,ll,'--', label='10%')
ax.plot(x1,ul,'--', label='90%')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend(loc='best')
```

```
fig, ax = plt.subplots()
ax.plot(x,y,'ro', label='measured')
ax.plot(x1,ll,'--', label='10%')
ax.plot(x1,ul,'--', label='90%')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x26f33ef71c0>
```



For our precipitation trends, linear regression is not enough because it varies a lot. We can see this by calculating a visualizing the regression line.

First, open the 'pp.csv' file

```
pp = pd.read_csv('pp.csv')
pp['Datetime'] = pd.to_datetime(pp['Datetime'])
pp = pp.set_index('Datetime')
pp.head()
```

```
pp = pd.read_csv('pp.csv')
pp['Datetime'] = pd.to_datetime(pp['Datetime'])
pp = pp.set_index('Datetime')
pp.head()
```

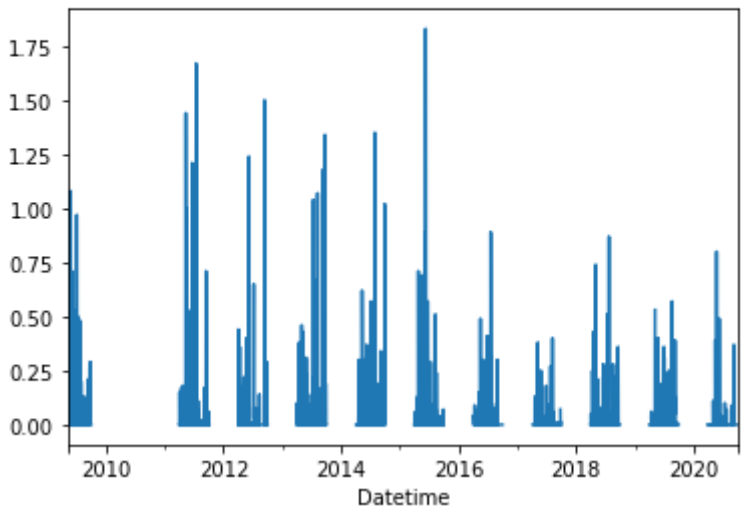
Datetime	Agency	SiteNumber	Precipitation_in	Code	Day	Month	Year
2009-05-05	USGS	393938104572101	0.00	A	5	5	2009
2009-05-06	USGS	393938104572101	0.00	A	6	5	2009
2009-05-07	USGS	393938104572101	0.00	A	7	5	2009
2009-05-08	USGS	393938104572101	0.01	A	8	5	2009
2009-05-09	USGS	393938104572101	0.08	A	9	5	2009

If you plot this file you could see how variant is the precipitation

```
pp['Precipitation_in'].plot()
```


```
pp['Precipitation_in'].plot()
```


```
<AxesSubplot:xlabel='Datetime'>
```

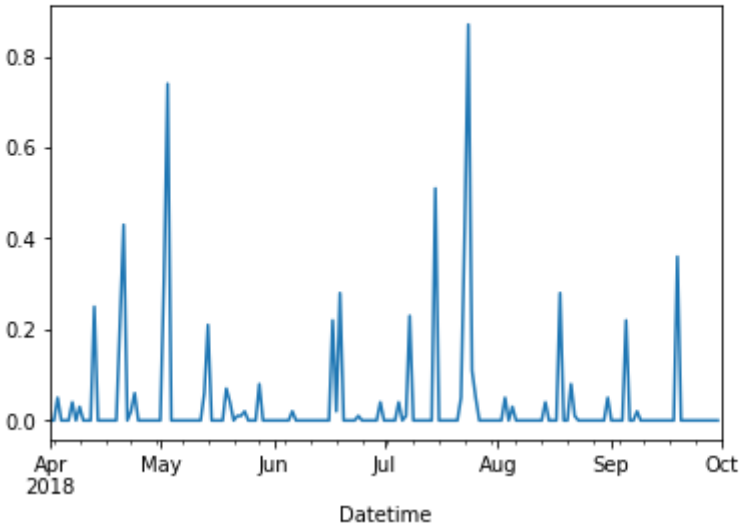


Even if we plot a specific range, the precipitation has short period peaks.

```
pp['Precipitation_in'].loc['2018-04-01':'2018-10-01'].plot()
```

```
pp['Precipitation_in'].loc['2018-04-01':'2018-10-01'].plot()
```

```
<AxesSubplot:xlabel='Datetime'>
```



Calculate its regression line

```
slope, intercept, CorrelationCoefficient, Pvalue , EstandardError =
st.linregress(np.arange(y2018.index.shape[0]),
y2018['Precipitation_in'])
slope, intercept
```

```
slope, intercept, CorrelationCoefficient, Pvalue , EstandardError = st.linregress(np.arange(y2018.index.shape[0]), y2018['Precipitati
slope, intercept
```

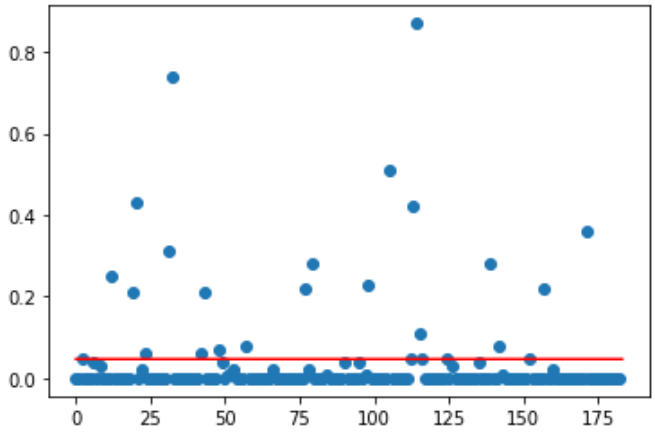
```
(-0.00011619528012970637, 0.0469672131147541)
```

And plot it, you could see a straight line.

```
plt.plot([0,len(y2018)],[slope*x.min()+intercept,slope*x.max()+interce
pt],color='red')
plt.scatter(np.arange(y2018.index.shape[0]),y2018['Precipitation_in'])
```

```
plt.plot([0,len(y2018)],[slope*x.min()+intercept,slope*x.max()+intercept],color='red')
plt.scatter(np.arange(y2018.index.shape[0]),y2018['Precipitation_in'])
```

```
<matplotlib.collections.PathCollection at 0x29835e27dc0>
```



A better approach to see the precipitation values is by distributions.

Distributions

To work with distributions, the main values you need are the mean and the standard deviation.

For example, if you want to calculate a normal distribution, you use the function "norm" of the library "scipy.stats.", this function returns a Scipy object. For the next example, we use a mean equal to 0 and a standard deviation equal to 5.

```
d = st.norm(loc=0, scale=5) # mean, standard deviation
d
```

```
d = st.norm(loc=0, scale=5) # mean, standard deviation
d
```

```
<scipy.stats._distn_infrastructure.rv_frozen at 0x26f340ad760>
```

Based on this distribution, we can calculate the Probability Density Function (PDF) or the Cumulative Density Function (CDF); for now and on, we calculate the PDF.

To calculate the PDF, you need to specify a range of values and pass them to it.

```
x = np.linspace(-50,50, 1000)
y1 = d.pdf(x)
```

```
x = np.linspace(-50,50, 1000)
y1 = d.pdf(x)
y1
```

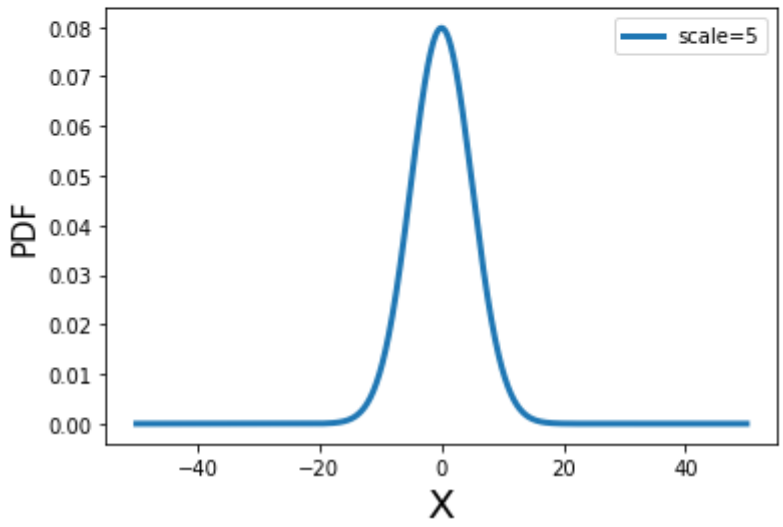
```
array([1.53891973e-23, 1.87964042e-23, 2.29487756e-23, 2.80072325e-23,
       3.41669970e-23, 4.16648052e-23, 5.07876170e-23, 6.18831270e-23,
       7.53724444e-23, 9.17653834e-23, 1.11678893e-22, 1.35859263e-22,
       1.65208862e-22, 2.00818334e-22, 2.44005330e-22, 2.96361103e-22,
       3.59806518e-22, 4.36659369e-22, 5.29715248e-22, 6.42344634e-22,
       7.78609433e-22, 9.43402787e-22, 1.14261674e-21, 1.38334322e-21,
       1.67411483e-21, 2.02519327e-21, 2.44891460e-21, 2.96010244e-21,
       3.57656214e-21, 4.31967172e-21, 5.21508805e-21, 6.29359051e-21,
       7.59208835e-21, 9.15482314e-21, 1.10348034e-20, 1.32955152e-20,
       1.60129623e-20, 1.92780953e-20, 2.31997070e-20, 2.79078776e-20,
       3.35580761e-20, 4.03360377e-20, 4.84635639e-20, 5.82054158e-20,
       6.98775031e-20, 8.38566086e-20, 1.00591931e-19, 1.20618780e-19,
       1.44574816e-19, 1.73219309e-19, 2.07455947e-19, 2.48359863e-19,
       2.97209624e-19, 3.55525092e-19, 4.25112214e-19, 5.08115966e-19,
       6.07082948e-19, 7.25035317e-19, 8.65558097e-19, 1.03290224e-18,
```

So lumping all together, we can plot the distribution function for a range of given values.

```
d = st.norm(loc=0, scale=5) # mean, standard deviation
x = np.linspace(-50,50, 1000)
y1 = d.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
d = st.norm(loc=0, scale=5) # mean, standard deviation
x = np.linspace(-50,50, 1000)
y1 = d.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
<matplotlib.legend.Legend at 0x26f33dcdac0>
```

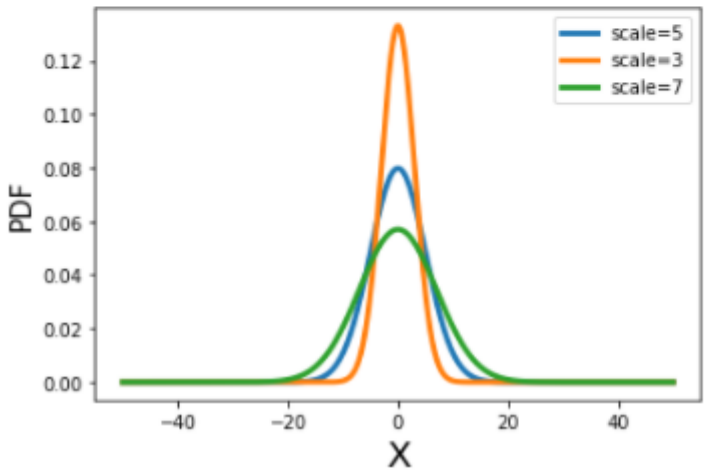


We can do this calculation for multiple standard deviations and see how they differ.

```
rv1 = st.norm(loc=0, scale=5) # mean, standard deviation
rv2 = st.norm(loc=0, scale=3)
rv3 = st.norm(loc=0, scale=7)

x = np.linspace(-50,50, 1000)
y1 = rv1.pdf(x)
y2 = rv2.pdf(x)
y3 = rv3.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.plot(x, y2, lw=3, label='scale=3')
ax.plot(x, y3, lw=3, label='scale=7')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
rv1 = st.norm(loc=0, scale=5) # mean, standard deviation
rv2 = st.norm(loc=0, scale=3)
rv3 = st.norm(loc=0, scale=7)

x = np.linspace(-50,50, 1000)
y1 = rv1.pdf(x)
y2 = rv2.pdf(x)
y3 = rv3.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.plot(x, y2, lw=3, label='scale=3')
ax.plot(x, y3, lw=3, label='scale=7')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
<matplotlib.legend.Legend at 0x26f36bef5b0>
```

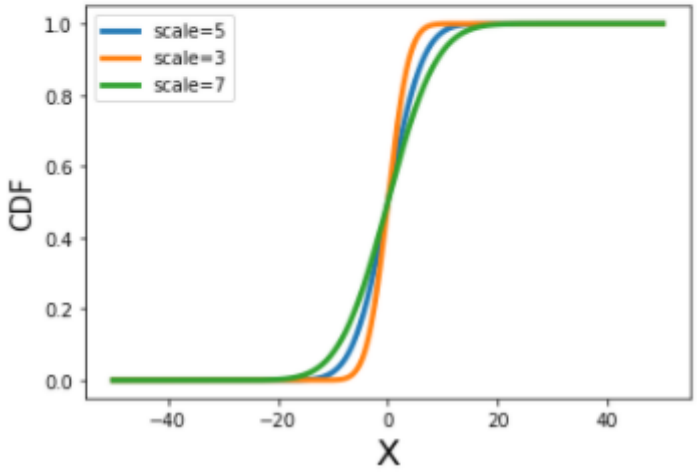


Just for the sake of visualizing a CDF calculate and plot it.

```
y1 = rv1.cdf(x)
y2 = rv2.cdf(x)
y3 = rv3.cdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.plot(x, y2, lw=3, label='scale=3')
ax.plot(x, y3, lw=3, label='scale=7')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('CDF', fontsize=15)
ax.legend()
```

```
y1 = rv1.cdf(x)
y2 = rv2.cdf(x)
y3 = rv3.cdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='scale=5')
ax.plot(x, y2, lw=3, label='scale=3')
ax.plot(x, y3, lw=3, label='scale=7')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('CDF', fontsize=15)
ax.legend()
```

```
<matplotlib.legend.Legend at 0x26f373f03a0>
```

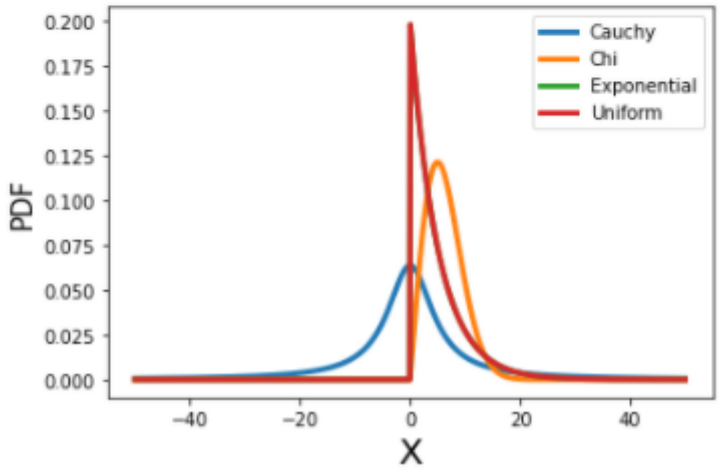


The normal distribution is not the only one available; you can find many kinds of distributions as follows:

```
rv1 = st.cauchy(loc=0, scale=5)
rv2 = st.chi(2, loc=0, scale=5)
rv3 = st.expon(loc=0, scale=5)
rv4 = st.uniform(loc=0, scale=5)

# compute pdf
y1 = rv1.pdf(x)
y2 = rv2.pdf(x)
y3 = rv3.pdf(x)
y4 = rv4.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='Cauchy')
ax.plot(x, y2, lw=3, label='Chi')
ax.plot(x, y3, lw=3, label='Exponential')
ax.plot(x, y3, lw=3, label='Uniform')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
rv1 = st.cauchy(loc=0, scale=5)
rv2 = st.chi(2, loc=0, scale=5)
rv3 = st.expon(loc=0, scale=5)
rv4 = st.uniform(loc=0, scale=5)

# compute pdf
y1 = rv1.pdf(x)
y2 = rv2.pdf(x)
y3 = rv3.pdf(x)
y4 = rv4.pdf(x)

fig,ax =plt.subplots()
ax.plot(x, y1, lw=3, label='Cauchy')
ax.plot(x, y2, lw=3, label='Chi')
ax.plot(x, y3, lw=3, label='Exponential')
ax.plot(x, y3, lw=3, label='Uniform')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
<matplotlib.legend.Legend at 0x26f33c7e130>
```



Let's work with the file "F_Wayne_Tmp_Pcp_Filter_xlsx"; you can find this file in the "Data" folder of this Session.

```
datos =
pd.read_excel('F_Wayne_Tmp_Pcp_Filter.xlsx',sheet_name='Sheet1',index_
col=0)
datos.head()
```

```
datos = pd.read_excel('F_Wayne_Tmp_Pcp_Filter.xlsx',sheet_name='Sheet1',index_col=0)
datos.head()
```

Date	Pcr (xx)	Pcp (mm)
1941-01-01	71	7.1
1941-01-02	15	1.5
1941-01-03	0	0.0
1941-01-04	0	0.0
1941-01-05	0	0.0

Let's rename the column "Pcp (mm)"

```
datos = datos.rename(columns={'Pcp (mm)': 'Ppt'})
datos.head()
```


```
datos = datos.rename(columns={'Pcp (mm)': 'Ppt'})
datos.head()
```

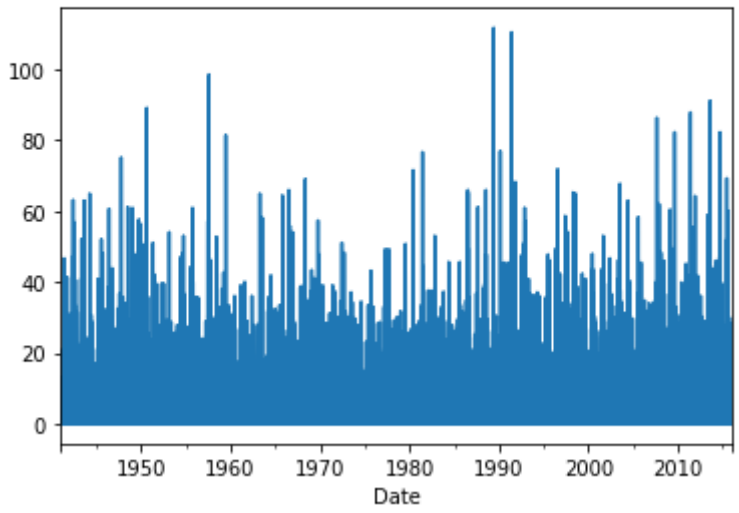

Date	Pcr (xx)	Ppt
1941-01-01	71	7.1
1941-01-02	15	1.5
1941-01-03	0	0.0
1941-01-04	0	0.0
1941-01-05	0	0.0

Plot the data:

```
datos['Ppt'].plot()
```

```
datos['Ppt'].plot()
```

```
<AxesSubplot:xlabel='Date'>
```



See it's statistics:

```
datos['Ppt'].describe()
```

```
datos['Ppt'].describe()
```

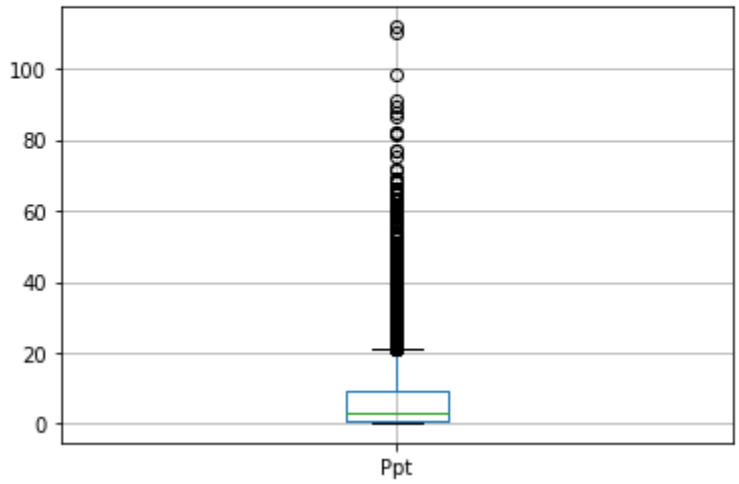
```
count    27393.000000
mean         2.567481
std          6.828232
min          0.000000
25%          0.000000
50%          0.000000
75%          1.300000
max        111.800000
Name: Ppt, dtype: float64
```

The boxplot reflects the statistics calculated previously.

```
datos[datos['Ppt']>0].boxplot('Ppt')
```

```
datos[datos['Ppt']>0].boxplot('Ppt')
```

```
<AxesSubplot:>
```

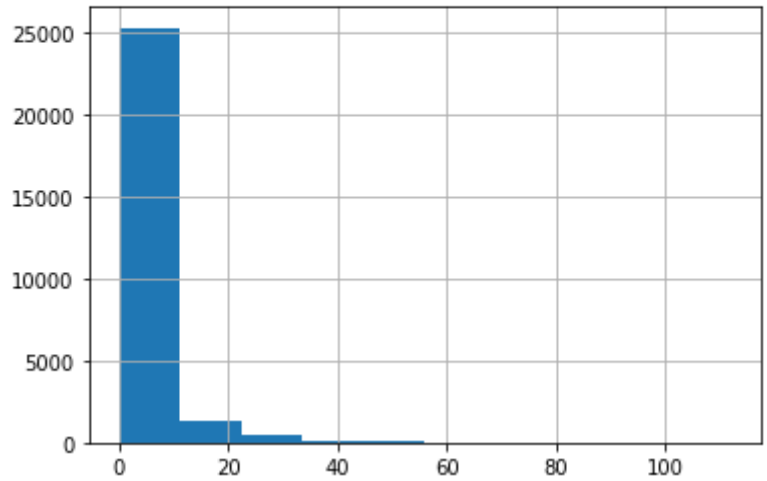


The histogram is a similar plot to a distribution.

```
datos['Ppt'].hist()
```

```
datos['Ppt'].hist()
```

```
<AxesSubplot:>
```



You always have to consider the date ranges with which you are working. The statistics can vary dramatically between one range of dates to another.

For instance:

```
datos['Ppt'].loc['1950-01-01':'1959-12-31'].describe()
```

```
# for 10 years: 1950 to 1960
datos['Ppt'].loc['1950-01-01':'1959-12-31'].describe()
```

```
count    3652.000000
mean        2.737979
std         7.080382
min         0.000000
25%         0.000000
50%         0.000000
75%         1.500000
max        98.600000
Name: Ppt, dtype: float64
```

```
# for 30 years: 1950 to 1980
datos['Ppt'].loc['1950-01-01':'1979-12-31'].describe()
```

```
# for 30 years: 1950 to 1980
datos['Ppt'].loc['1950-01-01':'1979-12-31'].describe()
```

```
count    10957.000000
mean         2.436789
std          6.319161
min          0.000000
25%          0.000000
50%          0.000000
75%          1.300000
max         98.600000
Name: Ppt, dtype: float64
```

To create distributions, we first need the mean and standard deviation of the information and store them inside variables.

```
promedio = datos['Ppt'].mean()
desviacion = datos['Ppt'].std()
promedio,desviacion
```

```
promedio = datos['Ppt'].mean()
desviacion = datos['Ppt'].std()
promedio,desviacion
```

```
(2.5674807432555764, 6.8282315143483645)
```

Now, we need to create an array with a range of data corresponding to the minimum and maximum precipitation

```
tabulaciones = np.arange(datos['Ppt'].min(),datos['Ppt'].max(),0.1)
```

Create the normal distribution for the precipitation values

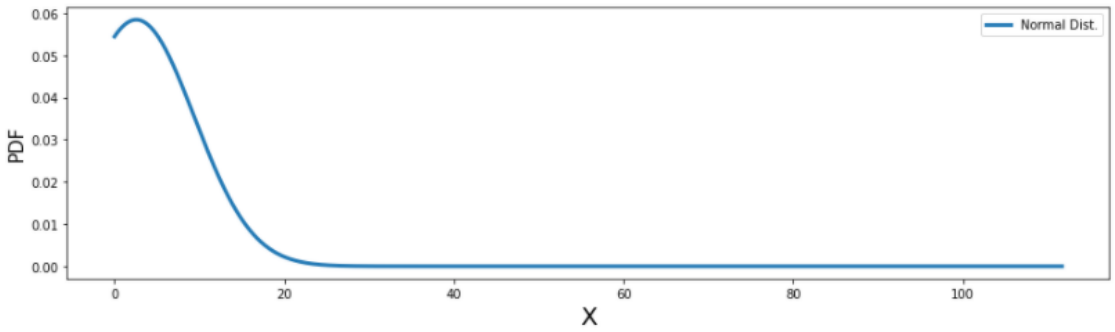
```
distnormal = st.norm.pdf(tabulaciones, loc=promedio, scale=desviacion)
```

Plot the range with the distribution.

```
fig,ax =plt.subplots()
ax.plot(tabulaciones, distnormal, lw=3, label='Normal Dist.')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
fig,ax =plt.subplots()
ax.plot(tabulaciones, distnormal, lw=3, label='Normal Dist.')
ax.set_xlabel('X', fontsize=20)
ax.set_ylabel('PDF', fontsize=15)
ax.legend()
```

```
<matplotlib.legend.Legend at 0x26f36bb9670>
```



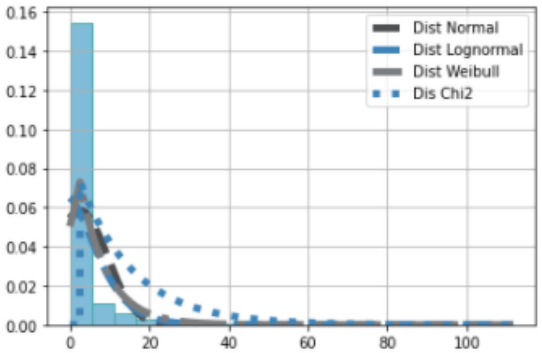
We can see the differences between different types of distributions

```
distnormal = st.norm.pdf(tabulaciones,
                         loc=promedio, scale=desviacion)
distlognormal = st.pearson3.pdf(tabulaciones,skew=1,
                                loc=promedio, scale=desviacion)
distweibull = st.dweibull.pdf(tabulaciones,c=1,
                                loc=promedio, scale=desviacion)
distchi2 = st.chi2.pdf(tabulaciones,df=2,
                                loc=promedio, scale=desviacion)
```

Then make a quick plot of them with the histogram. We have an extra argument "density" inside the plot of the histogram; this variable weights the amount of data; this is helpful to visualize the tendency of the histogram and the distributions.

```
datos['Ppt'].hist(bins=20, edgecolor='#4aaaaa',
color='#80BCD8',density=True)
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5,
linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5,
linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5,
linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5,
linestyle=':', label='Dis Chi2')
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 15, 4
```

```
# we plot the normalized histogram and the probabilistic distributions
datos['Ppt'].hist(bins=20, edgecolor='#4aaaaa', color='#80BCD8',density=True)
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5, linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5, linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5, linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5, linestyle=':', label='Dis Chi2')
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 15, 4
```



As you could see, most of the values are 0 precipitation. We could slice some values to perform a better graph or calculate the distribution for a specific month.

Let's try with another weather station that has more rainfall events. Open and format the file "station2.xlsx"

```
ws =
pd.read_excel('station2.xlsx',skiprows=2,index_col='FECHA',parse_dates
=True)
ws.head()
```

```
ws = pd.read_excel('station2.xlsx',skiprows=2,index_col='FECHA',parse_dates=True)
ws.head()
```

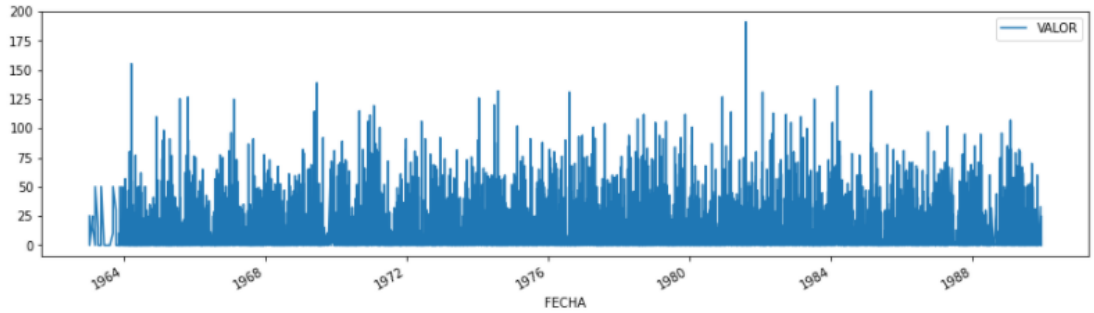
FECHA	ESTACION	OPERADOR	VARIABLE	HORA	VALOR	UNIDADMEDIDA
1963-01-11	PILCOPATA	SERVICIO NACIONAL METEOROLOGÍA E HIDROLOGÍA	PRECIPITACION ACU 1DIA	00:00:00	25.0	mm
1963-02-11	PILCOPATA	SERVICIO NACIONAL METEOROLOGÍA E HIDROLOGÍA	PRECIPITACION ACU 1DIA	00:00:00	25.0	mm
1963-03-11	PILCOPATA	SERVICIO NACIONAL METEOROLOGÍA E HIDROLOGÍA	PRECIPITACION ACU 1DIA	00:00:00	0.0	mm
1963-04-11	PILCOPATA	SERVICIO NACIONAL METEOROLOGÍA E HIDROLOGÍA	PRECIPITACION ACU 1DIA	00:00:00	15.1	mm
1963-05-11	PILCOPATA	SERVICIO NACIONAL METEOROLOGÍA E HIDROLOGÍA	PRECIPITACION ACU 1DIA	00:00:00	0.0	mm

See its values.

```
ws.plot()
```

```
ws.plot()
```

```
<AxesSubplot:xlabel='FECHA'>
```



Calculate the mean and standard deviation

```
promedio = ws['VALOR'].mean()
desviacion = ws['VALOR'].std()
promedio,desviacion
```

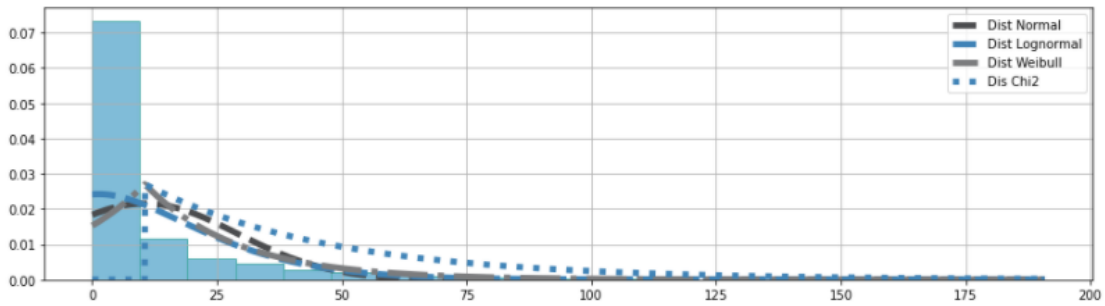
```
promedio = ws['VALOR'].mean()
desviacion = ws['VALOR'].std()
promedio,desviacion
```

```
(10.538268095342227, 18.48991989998275)
```

Calculate tabulations, the distributions, and plot the result. You could see almost the same distribution, but the X-axis elongates to 191 mm of precipitation, which is the maximum precipitation value. The curve of the distribution is not at 0, as we used to see. Now, it is located at 10.5 mm, which is the mean of all the time series

```
# now we determine the probabilistic distributions
tabulaciones = np.arange(ws['VALOR'].min(),ws['VALOR'].max(),0.1)

distnormal = st.norm.pdf(tabulaciones,
                         loc=promedio, scale=desviacion)
distlognormal = st.pearson3.pdf(tabulaciones,skew=1,
                                loc=promedio, scale=desviacion)
distweibull = st.dweibull.pdf(tabulaciones,c=1,
                                loc=promedio, scale=desviacion)
distchi2 = st.chi2.pdf(tabulaciones,df=2,
                                loc=promedio, scale=desviacion)

# we plot the normalized histogram and the probabilistic distributions
ws['VALOR'].hist(bins=20, edgecolor='#4aaaaa',
color='#80BCD8',density=True)
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5,
linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5,
linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5,
linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5,
linestyle=':', label='Dis Chi2')
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 15, 4
```

```
# now we determine the probabilistic distributions
tabulaciones = np.arange(ws['VALOR'].min(),ws['VALOR'].max(),0.1)

distnormal = st.norm.pdf(tabulaciones,
                         loc=promedio, scale=desviacion)
distlognormal = st.pearson3.pdf(tabulaciones,skew=1,
                                loc=promedio, scale=desviacion)
distweibull = st.dweibull.pdf(tabulaciones,c=1,
                                loc=promedio, scale=desviacion)
distchi2 = st.chi2.pdf(tabulaciones,df=2,
                                loc=promedio, scale=desviacion)

# we plot the normalized histogram and the probabilistic distributions
ws['VALOR'].hist(bins=20, edgecolor='#4aaaaa', color='#80BCD8',density=True)
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5, linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5, linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5, linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5, linestyle=':', label='Dis Chi2')
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 15, 4
```



Let's try with another weather station, open the data of the file "Precip_Est_Yauyos.csv"

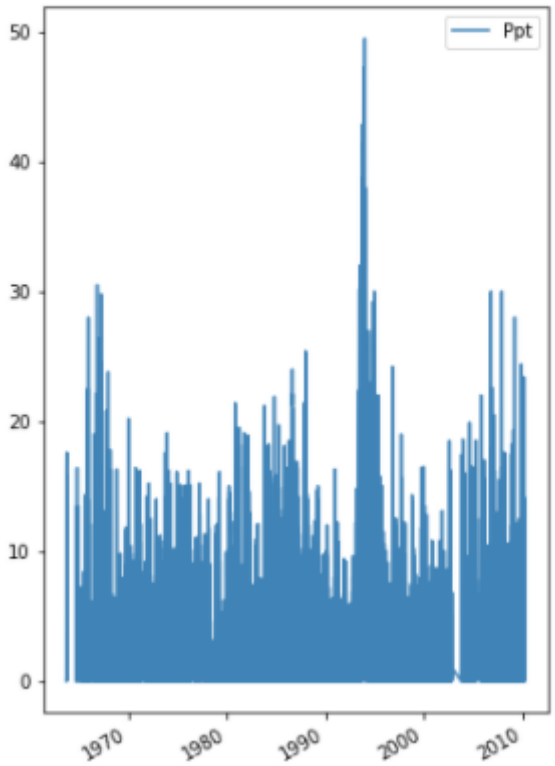
```
Yauyos = pd.read_csv('Precip_Est_Yauyos.csv', index_col=0,
parse_dates=True)
```

Plot its values


```
Yauyos = pd.read_csv('Precip_Est_Yauyos.csv', index_col=0,
parse_dates=True)
```

```
Yauyos.plot(color='#3F83B7')
```

```
<AxesSubplot:>
```



Calculate its mean and standard deviation, this time we are filtering the values greater than 0.

```
promedio = Yauyos[Yauyos>0].mean()
desviacion = Yauyos[Yauyos>0].std()
print(promedio, desviacion)
```

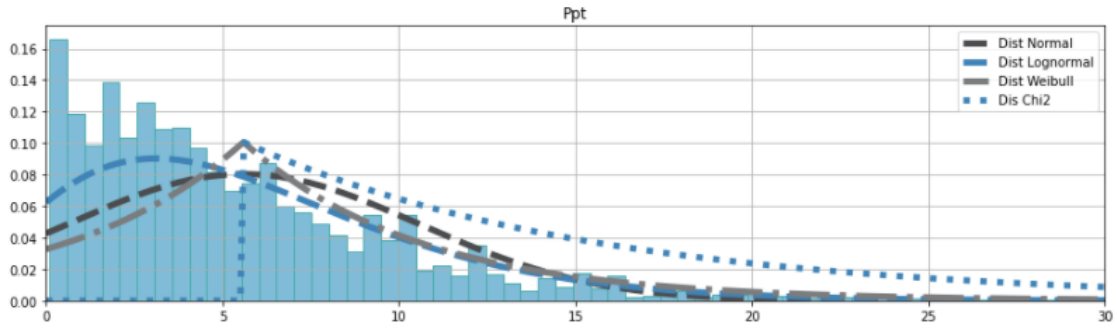
```
promedio = Yauyos[Yauyos>0].mean()
desviacion = Yauyos[Yauyos>0].std()
print(promedio, desviacion)
```

```
Ppt    5.589663
dtype: float64 Ppt    4.965991
dtype: float64
```

Then calculate its distributions and plot them.

```
#determinamos las regresiones estadisticas
tabulaciones = np.arange(Yauyos.Ppt.min(),Yauyos.Ppt.max(),0.1)
distnormal = st.norm.pdf(tabulaciones,
                         loc=promedio, scale=desviacion)
distlognormal = st.pearson3.pdf(tabulaciones,skew=1,
                                loc=promedio, scale=desviacion)
distweibull = st.dweibull.pdf(tabulaciones,c=1,
                              loc=promedio, scale=desviacion)
distchi2 = st.chi2.pdf(tabulaciones,df=2,
                       loc=promedio, scale=desviacion)

Yauyos[Yauyos>0].hist(bins=100, density=True, edgecolor='#4aaaaa',
color='#80BCD8')
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5,
linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5,
linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5,
linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5,
linestyle=':', label='Dis Chi2')
plt.xlim(0,30)
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 16,8
```


```
#determinamos las regresiones estadisticas
tabulaciones = np.arange(Yauyos.Ppt.min(),Yauyos.Ppt.max(),0.1)
distnormal = st.norm.pdf(tabulaciones,
                         loc=promedio, scale=desviacion)
distlognormal = st.pearson3.pdf(tabulaciones,skew=1,
                                loc=promedio, scale=desviacion)
distweibull = st.dweibull.pdf(tabulaciones,c=1,
                              loc=promedio, scale=desviacion)
distchi2 = st.chi2.pdf(tabulaciones,df=2,
                       loc=promedio, scale=desviacion)

Yauyos[Yauyos>0].hist(bins=100, density=True, edgecolor='#4aaaaa', color='#80BCD8')
plt.plot(tabulaciones,distnormal, color='#4B4C4E', linewidth=5, linestyle='--',label='Dist Normal')
plt.plot(tabulaciones,distlognormal, color='#3F83B7', linewidth=5, linestyle='--', label='Dist Lognormal')
plt.plot(tabulaciones,distweibull, color='#7B7C7E', linewidth=5, linestyle='-.', label='Dist Weibull')
plt.plot(tabulaciones,distchi2, color='#3F83B7', linewidth=5, linestyle=':', label='Dis Chi2')
plt.xlim(0,30)
plt.legend(loc='upper right')
plt.rcParams['figure.figsize'] = 16,8
```

Return periods

There are many ways to calculate return periods. The first method we use is based on the distributions. Here we pass a probability that is equal to the inverse of the number of years desired.

For instance, based on the previous distributions, we can calculate the return periods for 100 years.

```
p100y_norm = st.norm.ppf([1/100], loc=promedio, scale=desviacion)
p100y_lognorm = st.pearson3.ppf([1/100], skew=1, loc=promedio,
scale=desviacion)
p100y_weibull = st.dweibull.ppf([1/100], c=1, loc=promedio,
scale=desviacion)
p100y_chi2 = st.chi2.ppf([1/100], df=2, loc=promedio,
scale=desviacion)

print("Precipitation for T=100años Normal Dist =", p100y_norm[0])
print("Precipitation for T=100años Logormal Dist =", p100y_lognorm[0])
print("Precipitation for T=100años Weibull Dist =", p100y_weibull[0])
print("Precipitation for T=100años Chi2 Dist =", p100y_chi2[0])
```

```
p100y_norm = st.norm.ppf([1/100], loc=promedio, scale=desviacion)
p100y_lognorm = st.pearson3.ppf([1/100], skew=1, loc=promedio, scale=desviacion)
p100y_weibull = st.dweibull.ppf([1/100], c=1, loc=promedio, scale=desviacion)
p100y_chi2 = st.chi2.ppf([1/100], df=2, loc=promedio, scale=desviacion)

print("Precipitation for T=100años Normal Dist =", p100y_norm[0])
print("Precipitation for T=100años Logormal Dist =", p100y_lognorm[0])
print("Precipitation for T=100años Weibull Dist =", p100y_weibull[0])
print("Precipitation for T=100años Chi2 Dist =", p100y_chi2[0])
```

```
Precipitation for T=100años Normal Dist = -5.962958928335638
Precipitation for T=100años Logormal Dist = -2.298195808793828
Precipitation for T=100años Weibull Dist = -13.837406780891248
Precipitation for T=100años Chi2 Dist = 5.689482575259646
```

For 200 years would be the same proceeding:

```
p200y_norm = st.norm.ppf([1/200], loc=promedio, scale=desviacion)
p200y_lognorm = st.pearson3.ppf([1/200], skew=1, loc=promedio,
scale=desviacion)
p200y_weibull = st.dweibull.ppf([1/200], c=1, loc=promedio,
scale=desviacion)
```

```
p200y_chi2 = st.chi2.ppf([1/200], df=2, loc=promedio,
scale=desviacion)

print("Precipitation for T=200años Normal Dist  =", p200y_norm[0])
print("Precipitation for T=200años Logormal Dist =", p200y_lognorm[0])
print("Precipitation for T=200años Weibull Dist =", p200y_weibull[0])
print("Precipitation for T=200años Chi2 Dist =", p200y_chi2[0])
```

```
p500y_norm = st.norm.ppf([0.998], loc=promedio, scale=desviacion)
p500y_lognorm = st.pearson3.ppf([0.998], skew=1, loc=promedio, scale=desviacion)
p500y_weibull = st.dweibull.ppf([0.998], c=1, loc=promedio, scale=desviacion)
p500y_chi2 = st.chi2.ppf([0.998], df=2, loc=promedio, scale=desviacion)

print("Precipitación para T=200años Dist Normal =", p500y_norm[0])
print("Precipitación para T=200años Dist Logormal =", p500y_lognorm[0])
print("Precipitación para T=200años Dist Weibull =", p500y_weibull[0])
print("Precipitación para T=200años Dist Chi2 =", p500y_chi2[0])
```

```
Precipitación para T=200años Dist Normal = 19.882587070049723
Precipitación para T=200años Dist Logormal = 25.890733542281012
Precipitación para T=200años Dist Weibull = 33.0091860371984
Precipitación para T=200años Dist Chi2 = 67.31303406120736
```

Let's calculate specific probabilities for the maximum values of each year for the "Yauyos" DataFrame.

First, create a column of the years.

```
Yauyos['Year'] = Yauyos.index.year
Yauyos.head()
```

```
Yauyos['Year'] = Yauyos.index.year
Yauyos.head()
```

	Ppt	Year
1963-09-01	0.0	1963
1963-09-02	0.0	1963
1963-09-03	0.0	1963
1963-09-04	0.0	1963
1963-09-05	12.4	1963

Group by the years and calculate the maximum values.

```
GroupedYauyos = Yauyos.groupby('Year').max()
GroupedYauyos.head()
```

```
GroupedYauyos = Yauyos.groupby('Year').max()
GroupedYauyos.head()
```

Year	Ppt
1963	17.6
1964	16.4
1965	28.0
1966	30.5
1967	29.8

Sort from maximum to minimum all the yearly precipitation values.

```
GroupedYauyos = GroupedYauyos.sort_values("Ppt",ascending=False)
GroupedYauyos.head()
```

```
GroupedYauyos = GroupedYauyos.sort_values("Ppt",ascending=False)
GroupedYauyos.head()
```

Year	Ppt
1993	49.5
1966	30.5
1994	30.4
2007	30.0
2006	30.0

To calculate the probabilities, we divide the maximum value's ordinal position with the length of the data plus 1.

```
GroupedYauyos['Probability'] = [(i+1)/(len(GroupedYauyos)+1) for i in
range(len(GroupedYauyos))]
```

```
GroupedYauyos.head()
```

```
GroupedYauyos['Probability'] = [(i+1)/(len(GroupedYauyos)+1) for i in range(len(GroupedY
GroupedYauyos.head()
```

Year	Ppt	Probability
1963	17.6	0.020408
1964	16.4	0.040816
1965	28.0	0.061224
1966	30.5	0.081633
1967	29.8	0.102041

Then, the return period is equal to 1 divided by the probability

```
GroupedYauyos['ReturnPeriod'] = 1/GroupedYauyos['Probability']
GroupedYauyos.head()
```

```
GroupedYauyos['ReturnPeriod'] = 1/GroupedYauyos['Probability']
GroupedYauyos.head()
```

Year	Ppt	Probability	ReturnPeriod
1963	17.6	0.020408	49.000000
1964	16.4	0.040816	24.500000
1965	28.0	0.061224	16.333333
1966	30.5	0.081633	12.250000
1967	29.8	0.102041	9.800000

If you see the complete table, you could find all the list of return periods for our values

```
GroupedYauyos
```