Estimación por Cuadrados Mínimos

ejemplos.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

inputfile = "../Datasets/S1MME_week43.csv"
df = pd.read_csv(inputfile)
sig=df.S1_mode_combined_attach_request_times_SEQ
df.plot()
plt.title("timeseries: "+inputfile)
plt.xlabel("time")
plt.grid()
plt.show()


dataset=sig
interval=1
diff = list()
for i in range(interval, len(dataset)):
    value = dataset[i] - dataset[i - interval]
    diff.append(value)

plt.plot(sig)
plt.plot(diff)
plt.legend(['sig','diff'])
plt.show()

diff1=diff
dataset=diff
interval=1
diff = list()
for i in range(interval, len(dataset)):
    value = dataset[i] - dataset[i - interval]
    diff.append(value)

diff2=diff

plt.plot(sig)
plt.plot(diff1)
plt.plot(diff2)
plt.legend(['sig','diff','diff2'])
plt.show()


######################################################################
# fit a straight line to the economic data
from numpy import arange
from pandas import read_csv
```

```
from scipy.optimize import curve_fit
from matplotlib import pyplot

# define the true objective function
def objective(x, a, b):
    return a * x + b

# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/longley.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
# choose the input and output variables
x, y = data[:, 4], data[:, -1]
# curve fit
popt, _ = curve_fit(objective, x, y)
# summarize the parameter values
a, b = popt
print('y = %.5f * x + %.5f' % (a, b))
# plot input vs output
pyplot.scatter(x, y)
# define a sequence of inputs between the smallest and largest known inputs
x_line = arange(min(x), max(x), 1)
# calculate the output for the range
y_line = objective(x_line, a, b)
# create a line plot for the mapping function
pyplot.plot(x_line, y_line, '--', color='red')
pyplot.show()


########################################################################
# fit a second degree polynomial to the economic data
from numpy import arange
from pandas import read_csv
from scipy.optimize import curve_fit
from matplotlib import pyplot

# define the true objective function
def objective(x, a, b, c):
    return a * x + b * x**2 + c

# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/longley.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
# choose the input and output variables
x, y = data[:, 4], data[:, -1]
# curve fit
popt, _ = curve_fit(objective, x, y)
# summarize the parameter values
a, b, c = popt
print('y = %.5f * x + %.5f * x^2 + %.5f' % (a, b, c))
# plot input vs output
pyplot.scatter(x, y)
# define a sequence of inputs between the smallest and largest known inputs
x_line = arange(min(x), max(x), 1)
# calculate the output for the range
y_line = objective(x_line, a, b, c)
# create a line plot for the mapping function
```

```
pyplot.plot(x_line, y_line, '--', color='red')
pyplot.show()

#########################################################################
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

inputfile = "../Datasets/S1MME_week43.csv"
df = pd.read_csv(inputfile)
sig=df.S1_mode_combined_attach_request_times_SEQ

N=24 #hours
sample=4 #sampled period number

dataframe = pd.Series(sig)
ts=pd.DataFrame(dataframe.values)
rows=int(len(ts)/N)
data = ts.values.reshape(rows,N)

#########################################################################
## estimación por valor medio de cada hora
#########################################################################

betas=data.mean(axis=0)
plt.plot(betas)
plt.plot(data[sample,:])
plt.plot(data[sample,:]-betas)
plt.legend(['betas','data','data-betas'])
plt.show()

#########################################################################
## repetir para todo el largo de la serie
#########################################################################

est=np.tile(betas,rows)

plt.plot(sig)
plt.plot(est)
plt.plot(sig-est)
plt.legend(['sig','est','sig-est'])
plt.show()

#########################################################################
## ejemplo diferenciando
#########################################################################

dataset=sig
interval=1
diff = list()
for i in range(interval, len(dataset)):
```

```
    value = dataset[i] - dataset[i - interval]
    diff.append(value)

plt.plot(sig)
plt.plot(diff)
plt.legend(['sig','diff'])
plt.show()

#repito la estimacion pero para la diferenciada
N=24 #hours
dataframe = pd.Series(pd.concat([pd.Series(diff[0]),pd.Series(diff)], axis=0))
ts=pd.DataFrame(dataframe.values)
rows=int(len(ts)/N)
data = ts.values.reshape(rows,N)

betas=data.mean(axis=0)
plt.plot(betas)
plt.plot(data[sample,:])
plt.plot(data[sample,:]-betas)
plt.legend(['betas','data','data-betas'])
plt.show()

##############################################################################
## repetir para todo el largo de la serie
##############################################################################

est=np.tile(betas,rows)

plt.plot(sig)
plt.plot(est)
plt.plot(sig-est)
plt.legend(['sig','est','sig-est'])
plt.show()
```