RomanEmperors

July 16, 2021

1 Python script for duration of Roman empire

Source: - El Barmi, H., & McKeague, I. W. (2013). Empirical likelihood-based tests for stochastic ordering. Bernoulli, 19(1), 295- -307. - Khmaladze, E., Brownrigg, R. and Haywood, J. (2007). Brittle power: On Roman Emperors and exponential lengths of rule. Statist. Probab. Lett. 77 1248–1257. MR2392795

Durations of rule of Roman Emperors: First column: duration, in years Second column: Indicator of period

Period 1: 27 BC-235 AD Period 2: 235-284 AD Period 3: 284-395 AD, fase final, período recheado de déspotas

O artigo de Khmaladze, Brownrigg and Haywood (2007) analisaram os tempos de duração dos reinados dos imperadores romanos. Eles verificaram que uma distribuição exponencial ajusta-se bem aos dados implicando que seus reinados terminavam abruptamente. Os autores disseram que eles tinham um "brittle power" (poder fr'{a}gil). Para justificar esta afirmação lembre-se da propriedade "sem mem'{o}ria" da distribuição exponencial (ver seção ?? do livro-texto).

Verifique se o ajuste da distribuição exponencial é adequado usando os dados do arquivo RomanEmperors.txt. Ele possui duas colunas de dados. A primeira delas mostra uma lista ordenada da duração em anos dos 70 imperadores romanos, de Augusto, o primeiro deles filho adotivo de Júlio César, a Theodossius, o último grande imperator romano antes da derrocada final do império. Estes reinados cobrem o período de 27 AC a 395 DC. A seguir, use os dados dessa primeira coluna.

[2]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import scipy as scipy

# Reading this text file into a pandas dataframe
dados = pd.read_csv("RomanEmperors.txt", delimiter=r"\s+", header=0)
print(dados)
x = dados['duration']

scipy.__version__
```

```
    duration  period
0  41.585220       1
1  22.573580       1
```

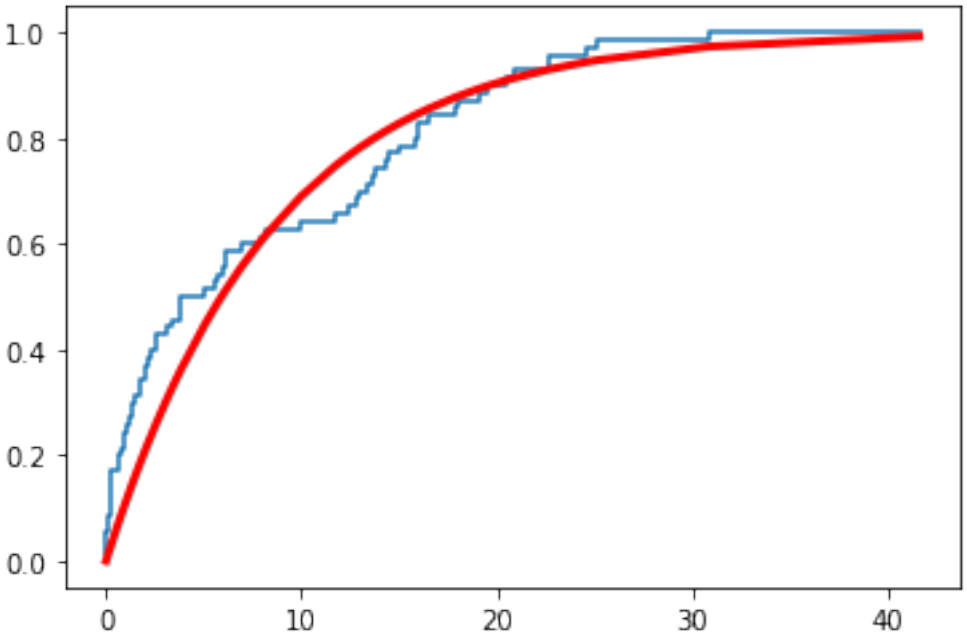
```
2    3.854894       1
3   13.716630       1
4   13.656400       1
..        …     …
65  11.726220       3
66  14.365500       3
67  16.002740       3
68  16.479120       3
69  15.994520       3

[70 rows x 2 columns]
```

[2]:
```
'1.3.1'
```

[3]:
```
# statsmodels is a Python module that provides classes and functions for the␣
 ↪estimation
# of many different statistical models, as well as for conducting statistical␣
 ↪tests,
# and statistical data exploration.

import statsmodels.api as sm # recommended import according to the docs

# empirical cumulative distribution function
ecdf = sm.distributions.ECDF(x)
```

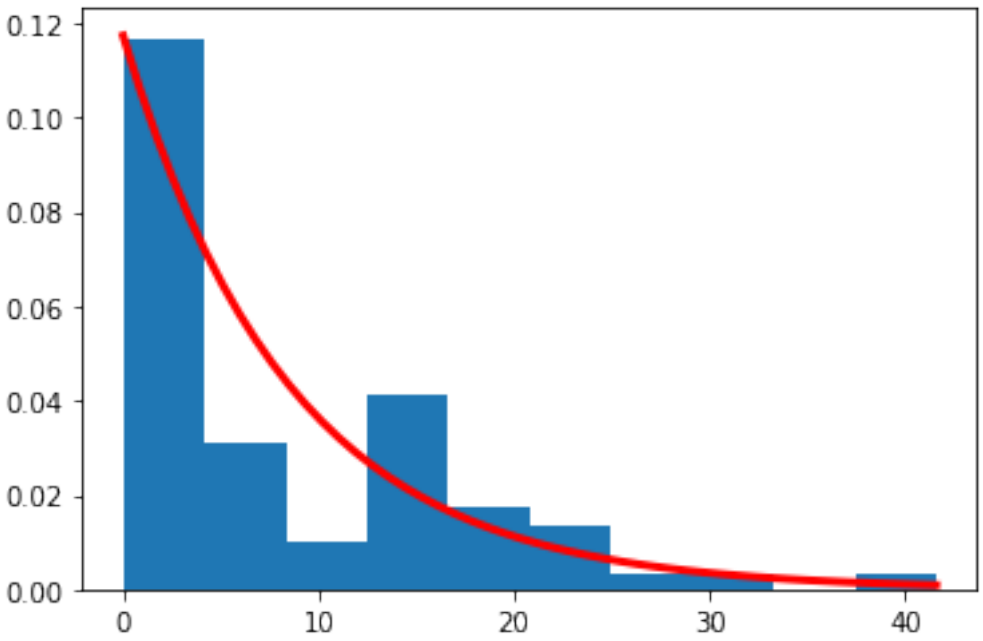
[4]:
```
# Visualizing the empirical and fitted cumulative functions
import matplotlib.pyplot as plt
%matplotlib inline

# plot of the empirical cumulative distribution function and the theoretical␣
 ↪(exponetial) F(x)
x2 = np.sort(x) # horizontal grid at the observed sample values
y2 = ecdf(x2) # value of the ecdf at x2 values
y3 = stats.expon.cdf(x2, scale=np.mean(x)) # valor de F(x) teorico (dist␣
 ↪exponencial) at x2
plt.step(x2, y2)
plt.plot(x2, y3, c ='r', linewidth=3)
```

[4]:
```
[<matplotlib.lines.Line2D at 0x1c17ba9350>]
```



[5]:
```
# Visualizing the histogram and the fitted exponential density

plt.hist(x, density=1) # normalized histogram (area = 1)
xd2 = np.linspace(min(x), max(x)) # horizontal grid
yd3 = stats.expon.pdf(xd2, scale=np.mean(x)) # valor de F(x) teorico (dist␣
 ↪exponencial) at xd2
plt.plot(xd2, yd3, c ='r', linewidth=3)

# The histograma and density visualization does not send us a clear message␣
 ↪about the fitness quality of the model
# It is a an example where the cumulative function seems to gives a more clear␣
 ↪answer (theta exp is a good fit)
# However the real answer is given by the Kolmogorov test
```

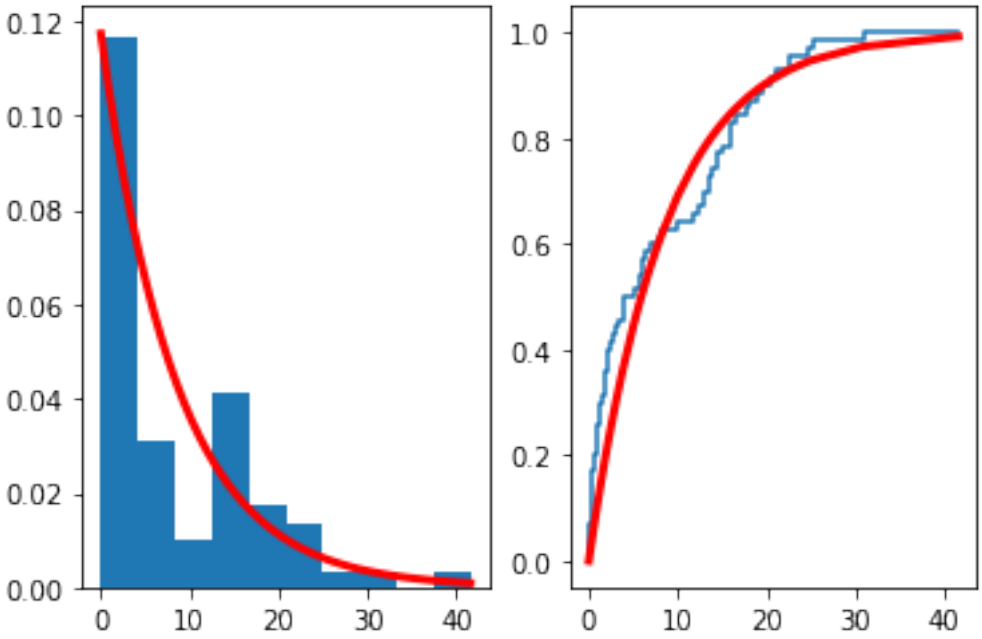
[5]:
```
[<matplotlib.lines.Line2D at 0x1c17f679d0>]
```



The histograma and density visualization does not send us a clear message about the fitness quality of the model. It is a an example where the cumulative function seems to gives a more clear answer (theta exp is a good fit). However the real answer is given by the Kolmogorov test.

[6]:
```
# The two plots in the same window
plt.figure(figsize=(20, 5))

plt.figure()
plt.subplot(121) # array 1 x 2 de subplots, drawing in subplot 2
plt.hist(x, density=1) # normalized histogram (area = 1)
plt.plot(xd2, yd3, c ='r', linewidth=3)

plt.subplot(122) # array 1 x 2 of subplots, drawing in subplot 1
plt.step(x2, y2)
plt.plot(x2, y3, c ='r', linewidth=3)
```

[6]:
```
[<matplotlib.lines.Line2D at 0x1c1828bc50>]

<Figure size 1440x360 with 0 Axes>
```



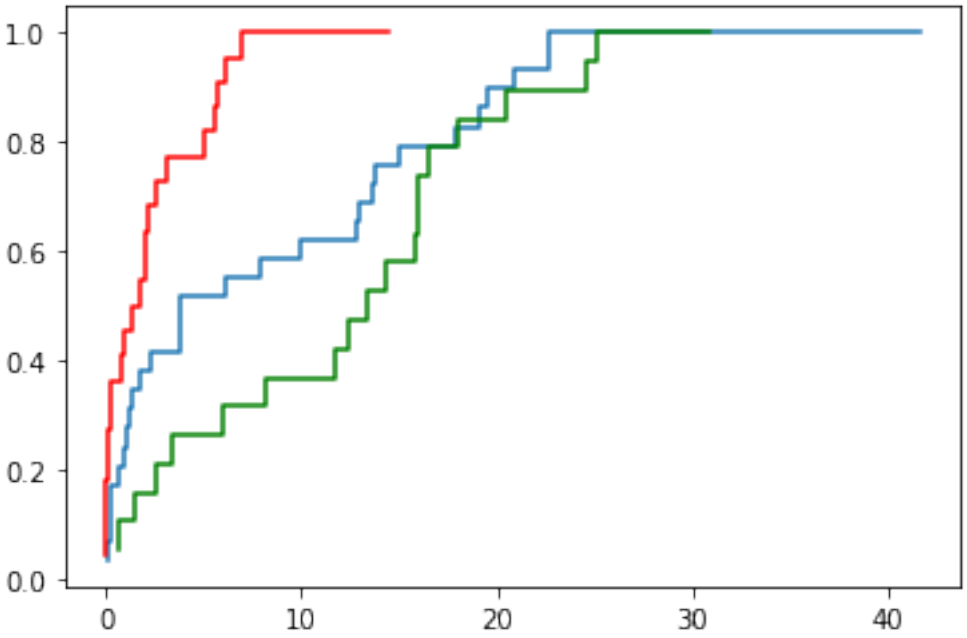
[7]:
```
# The Kolmogorov test
# function in the stats module of the scipy library

# Neste caso, temos E(X) = scale = aproximadamente a media aritmetica da amostra␣
 ↪de X
mx = np.mean(x)
print("A duração média dos 70 reinados foi de ",mx, "anos.")
kolmo = stats.kstest(x, cdf='expon', args=(0,mx)) # args requer o parametro␣
 ↪loc, nao pode passar apenas scale.
print(kolmo)
```

```
A duração média dos 70 reinados foi de  8.517532122857142 anos.
KstestResult(statistic=0.1559898592129607, pvalue=0.059263118369621726)
```

[8]:
```
# Teste de Kolomogorv-Smirnov
# A distribuição dos tempos de reinado nos 3 períodos do império romano seguiram␣
 ↪a mesma distribuição.
# Não estamos testando se a distribuição é exponencial.
# Testamos apenas se a distribuição do período 1 = distribuição do período 2,␣
 ↪não importa qual seja
# esta distribuição comum.

dados["period"].value_counts()

per1 = dados[dados['period']==1]['duration']
```

```
per2 = dados[dados['period']==2]['duration']
per3 = dados[dados['period']==3]['duration']

ecdf1 = sm.distributions.ECDF(per1)
ecdf2 = sm.distributions.ECDF(per2)
ecdf3 = sm.distributions.ECDF(per3)

# plot of the empirical cumulative distribution function and the theoretical␣
 ↪(exponetial) F(x)
x1 = np.sort(per1) # horizontal grid at the observed sample values
y1 = ecdf1(x1) # value of the ecdf at x2 values
x2 = np.sort(per2) # horizontal grid at the observed sample values
y2 = ecdf2(x2) # value of the ecdf at x2 values
x3 = np.sort(per3) # horizontal grid at the observed sample values
y3 = ecdf3(x3) # value of the ecdf at x2 values
plt.step(x1, y1)
plt.step(x2, y2, c = 'r')
plt.step(x3, y3, c = 'g')
```

[8]: [<matplotlib.lines.Line2D at 0x1c1838e990>]



Observe como o período 2 (em vermelho) teve tempos de reinado substancialmente mais curtos que os outrso dois períodos. O período 3 (em verde) parece ter reinados um pouco mais longos mas... será mesmo? Até onde a diferença entre os períodos 1 e 3 não pode ser devido ao mero acaso?

Isto é, será que a distribuição que gera os períodos de reinado dos períodos 1 e 3 não é a mesma e

a diferença que vemos nos dados foi uma mera flutuação aleatória?

[9]:
```
print(stats.ks_2samp(per2, per1)) # vermelho x azul
print(stats.ks_2samp(per2, per3)) # vermelho versus verde

print(stats.ks_2samp(per1, per3)) # azul x verde

# Na última versão do scipy podemos usar diretamente a funcão kstest (minha␣
 ↪versão
# não é a mais recente...
stats.kstest(per2, per3)
```

```
Ks_2sampResult(statistic=0.4373040752351097, pvalue=0.011317720649650753)
Ks_2sampResult(statistic=0.6913875598086124, pvalue=3.499600487877341e-05)
Ks_2sampResult(statistic=0.2885662431941924, pvalue=0.23392828983229197)
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call last)

       <ipython-input-9-28d848e9b5a6> in <module>
         5
         6 # Na última versão do scipy podemos usar diretamente a funcão kstest:
   ----> 7 stats.kstest(per2, per3)


       /Applications/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py␣
↪in kstest(rvs, cdf, args, N, alternative, mode)
      4795         vals = np.sort(rvs)
      4796         N = len(vals)
   -> 4797     cdfvals = cdf(vals, *args)
      4798
      4799     # to not break compatibility with existing code


       TypeError: 'Series' object is not callable
```