Ciências ULisboa

Modelação Numérica
Aula 9

Ajuste de parâmetros de distribuições

Algoritmo de otimização

GitHub - pjmateus/monte-carlo-annealing: Monte-Carlo search for the minimum of the multidimensional "cost" function

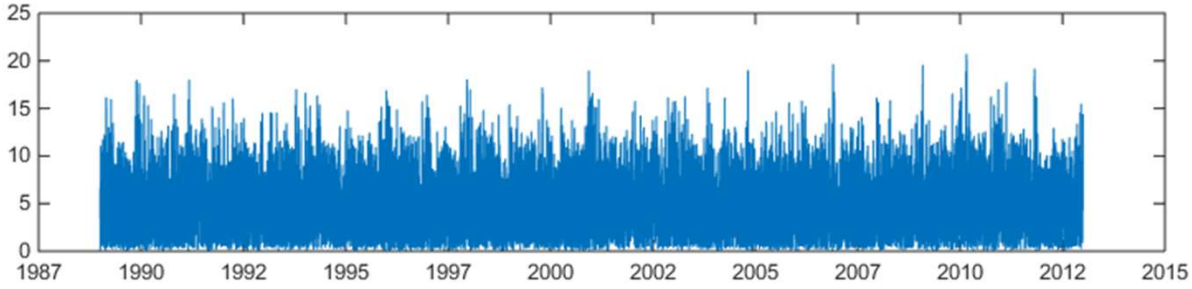
Artigo correspondente:

Miranda, P. M. A., & Mateus, P. (2022). Improved GNSS water vapor tomography with modified mapping functions. Geophysical Research Letters, 49, e2022GL100140. https://doi.org/10.1029/2022GL100140

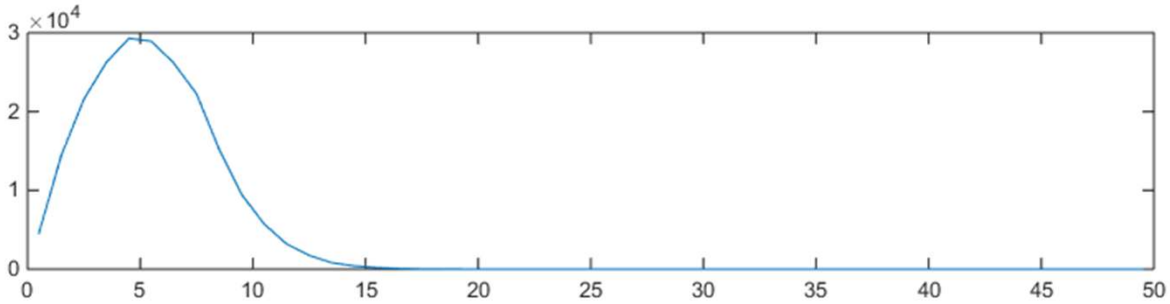
Distribuições

Alguns dados são bem descritos por distribuições estatísticas, que apesar de não permitirem prever dados específicos, definem a probabilidade de ocorrência de certos valores.

Vento simulado em Lisboa



Histograma do vento simulado



Distribuição de Weibull

No caso do vento é sabido que a sua função densidade de probabilidade se aproxima de uma distribuição de **Weibull**

$$\begin{cases} p(x) = \left(\frac{k}{\lambda}\right)\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^{k}}, x \geq 0 \\ p(x) = 0, x < 0 \end{cases}$$

configurada com 2 parâmetros: k $(shape)$ e λ $(scale)$, sendo estes parâmetros variáveis de local para local. O conhecimento de k, λ permite, por exemplo caracterizar completamente a potência eólica num dado local, se se admitir que a distribuição é estacionária (invariante no tempo).

Forma generalizada da distribuição de Weibull

A função de Weibull pode incluir um terceiro parâmetro livre $\tau(threshold)$:

$$\begin{cases} p(x) = \left(\frac{k}{\lambda}\right)\left(\frac{x-\tau}{\lambda}\right)^{k-1} e^{-\left(\frac{x-\tau}{\lambda}\right)^{k}}, x-\tau \geq 0 \\ p(x) = 0, x-\tau < 0 \end{cases}$$

No caso do vento, variável positiva definida, $\tau = 0$.

Nos exemplos que se segue vamos considerar o caso do vento, mas escrever o método de cálculo para o caso mais geral com 3 variáveis a estimar.

Otimização com pequenas alterações

```
def annealD(costX,vmin,vmax,Jmin,minvstep,maxITER,maxPERT,COOL,kappa,T,outITER,iseed=0):
    path=[]; sh=np.shape(vmin);ndim=;sh[0];vstep=2*(vmax-vmin)
    minxstep=minvstep[0];xstep=vstep[0];
    if iseed!=0:
        np.random.seed(iseed)
    rrr=np.random.sample(sh)
    V=vmin+(vmax-vmin)*rrr; VI=np.zeros(sh);J=cost(V);iTER=0;nHIT=1;kPERT=0
    while (iTER<maxITER and J>Jmin and xstep>minxstep):
        nHIT=0;iP=0
        while iP<maxPERT:
            kPERT=kPERT+1
            for idim in range(ndim):
                rr=np.random.rand()-0.5
                VI[idim]=V[idim]+vstep[idim]*rr
                while VI[idim]<vmin[idim] or VI[idim]>vmax[idim]:
                    rr=np.random.rand()-0.5
                    VI[idim]=V[idim]+vstep[idim]*rr
            JI=costX(VI);
            if JI<J:
                J=np.copy(JI);V=np.copy(VI);path.append(V)
            iP=iP+1
        iTER=iTER+1;T=T*COOL
        vstep=np.maximum(minvstep,vstep*np.exp(-kappa/T))
        xstep=vstep[0]
    return ndim,V,iTER,path
```

Distribuição de Weibull

$$\int_0^{\infty} weibull(x)dx = 1$$

```
def weibull(shape,scale,x,threshold=0):
    w=shape/scale*((x-threshold)/scale)**(shape-1)\
       *np.exp(-((x-threshold)/scale)**shape)
    return w
def costW(V): #2 dimensões
    shape,scale=V
    custo=np.mean((weibull(shape,scale,tH)-wH)**2)
    return custo
```

Leitura dos dados (24 anos, 1h, simulados)

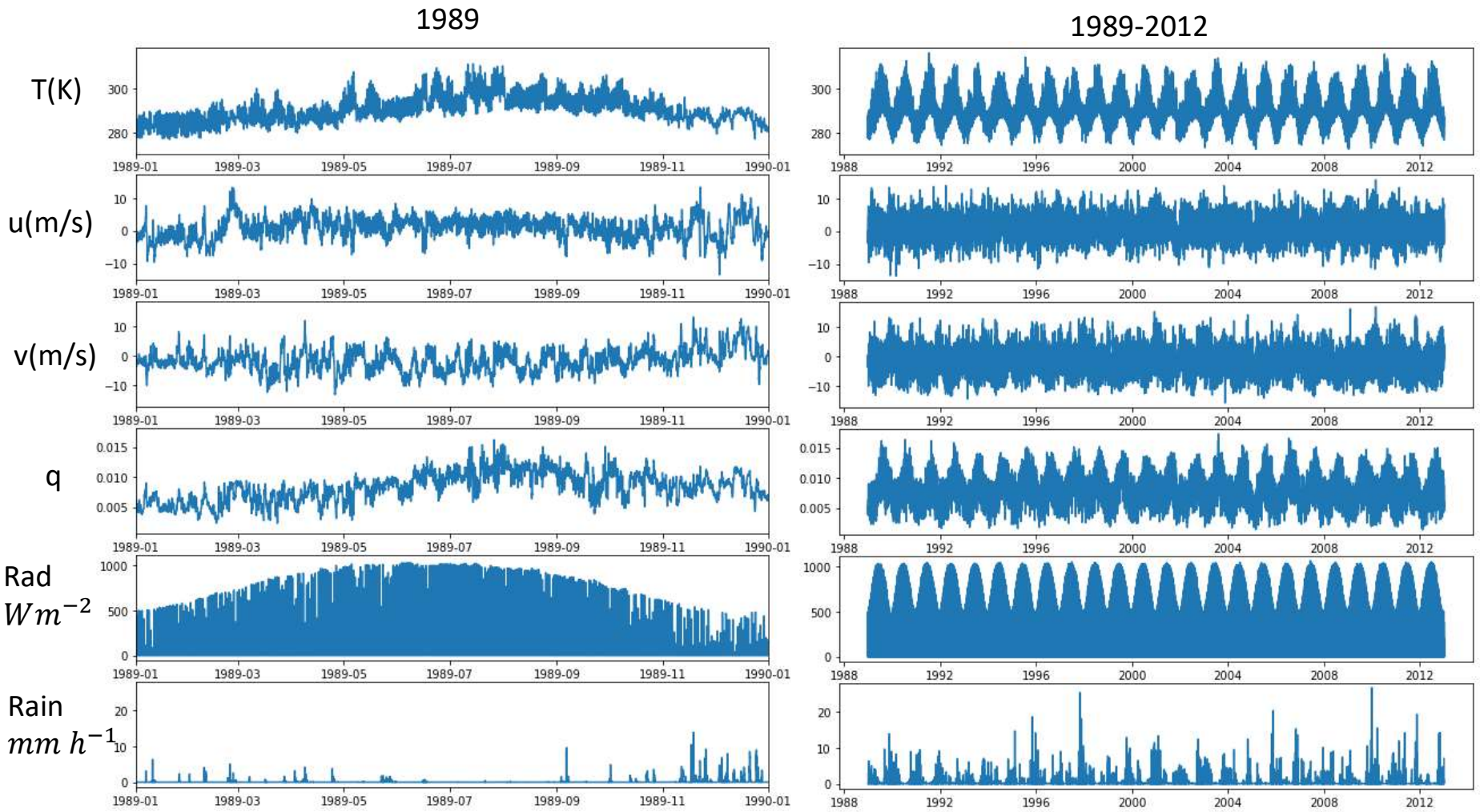
```
import datetime
time0=datetime.datetime(1989,1,1)
tt=np.copy(time0)
Dados=np.loadtxt('wrf_9km_Lisboa.dat')
times=[]
for k in range(len(Dados)):
    tt=tt+datetime.timedelta(hours=1)
    times.append(tt)
Temp=Dados[:,1]
u=Dados[:,2]
v=Dados[:,3]
q=Dados[:,4]
Rad=Dados[:,5]
Rain=Dados[:,6]
plt.figure(figsize=(10,12))
for var in range(1,7):
    plt.subplot(6,1,var)
    plt.plot(times,Dados[:,var])
    plt.xlim(datetime.datetime(1989,1,1)\
            ,datetime.datetime(1990,1,1))
```

ines,ncols,kP)

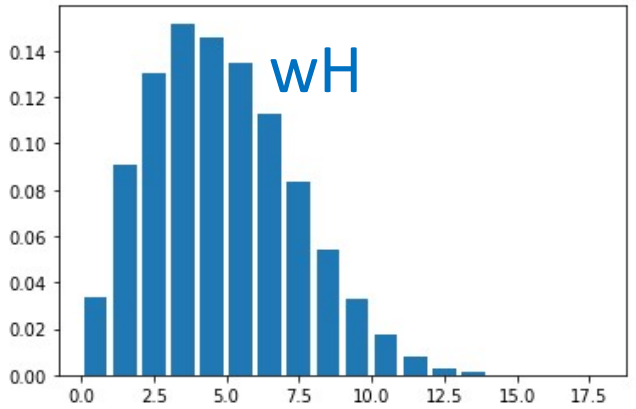
Dados - NumPy object array

	0	1	2	3	4	5	6
5	5.5	278.979	-3.75273	-3.64323	0.00504894	0	0
6	6.5	278.499	-3.4544	-3.45303	0.00483176	0	0
7	7.5	278.037	-2.94369	-3.3844	0.00465613	0	0
8	8.5	277.677	-3.2118	-3.39994	0.00456067	92.207	0
9	9.5	278.27	-3.42279	-3.23903	0.00456729	248.117	0
10	10.5	279.31	-3.57335	-2.75074	0.00454747	354.755	0
11	11.5	280.609	-3.27782	-2.41573	0.00452356	423.995	0
12	12.5	282.208	-3.07168	-1.89334	0.00447765	448.76	0
13	13.5	283.931	-2.76009	-1.31284	0.00466629	383.954	0
14	14.5	285.214	-1.996	-1.07618	0.00495157	279.727	0
15	15.5	285.908	-1.46128	-1.08457	0.00512272	180.842	0
16	16.5	286.112	-1.08831	-1.07285	0.00523031	82.8436	0
17	17.5	285.415	-1.09803	-1.12118	0.00547156	21.3852	0
18	18.5	284.97	-1.12912	-1.14843	0.00560579	0	0

Horas Temp u v q Rad Chuva


1989
1989-2012
T(K)
u(m/s)
v(m/s)
q
Rad
Wm^{-2}
Rain
$mm\ h^{-1}$

Leitura dos dados (24 anos, 1h, simulados)



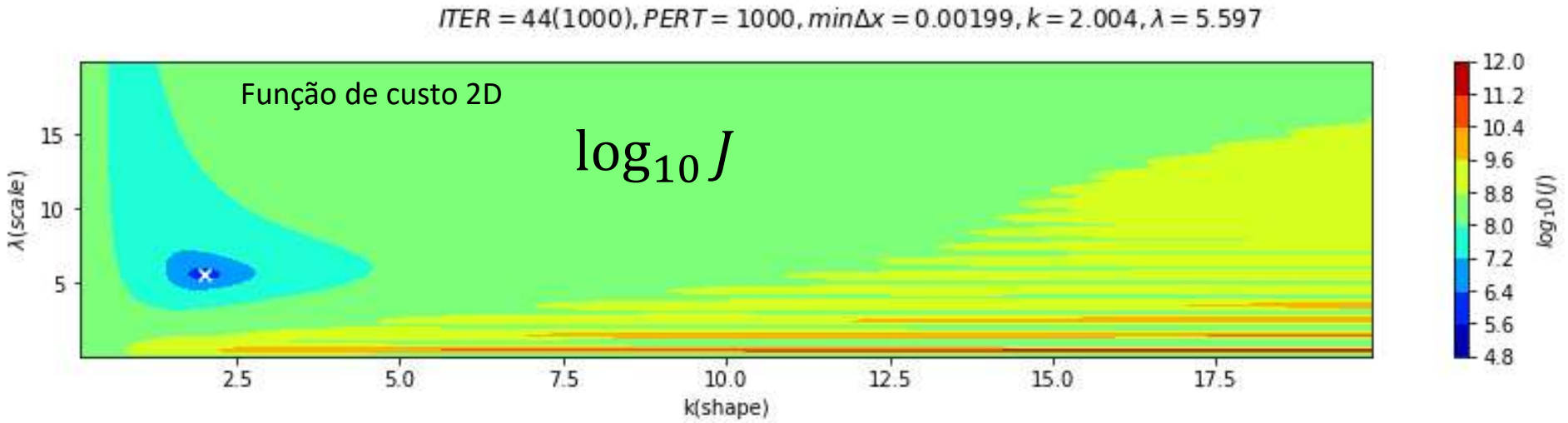

```
...
wE=np.sqrt(u**2+v**2);nE=len(wind)
step=1;low=0;high=18
tH=np.arange(low+step/2,high,step) #centro dos bins
bins=np.arange(low,high+step,step) #fronteiras dos bins
wH=np.histogram(wE,bins=bins)[0]/nE/step
plt.figure()
plt.bar(tE,wH)
V=np.zeros((2))
vnames=np.array(['k(shape)',r'$\lambda(scale)$'])
vmin=np.array([0.1,0.1])
vmax=np.array([20,20])
```

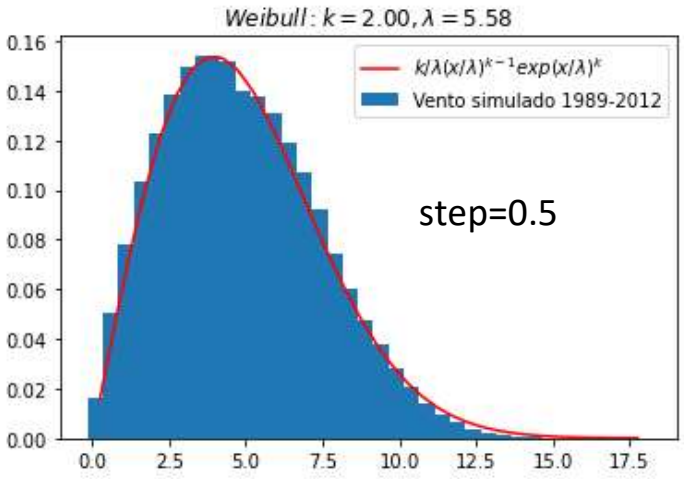
```
In [7]: np.sum(wH*step)
Out[7]: 0.9999952467868279
```

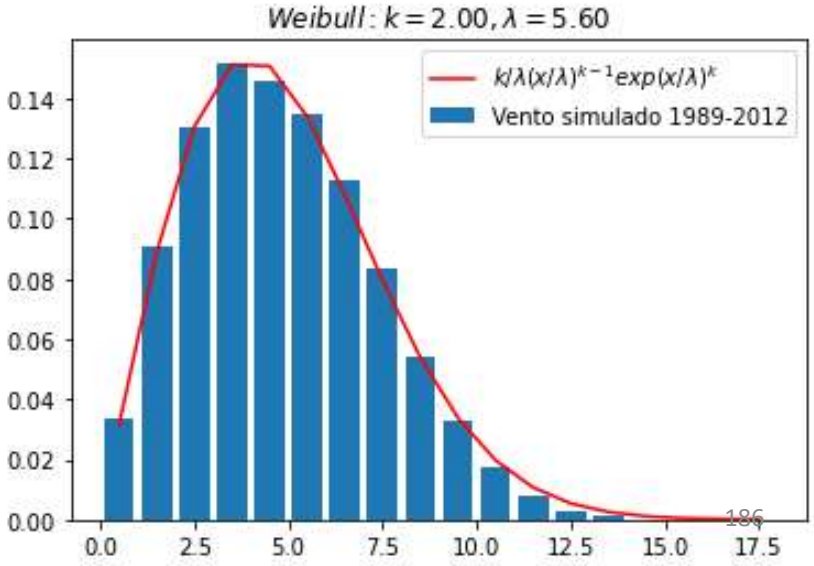
$$\int_{low}^{high} wH(x)dx = 1$$

Otimização bin a bin

```
Jmin=-10.
minvstep=(vmax-vmin)/10000
kappa=np.float64(0.1);T=10.;outITER=1
maxITER=1000
maxPERT=1000
COOL=0.9
n,V,iTER,path=annealD(costW,vmin,vmax,Jmin,minvstep,maxITER,maxPERT,\
                      COOL,kappa,T,outITER,iseed=2)
plt.figure()
plt.bar(tH,wH/len(wind),label='Vento simulado 1989-2012')
plt.plot(tH,weibull(V[0],V[1],tH),color='red'\
         ,label=r'$k/\lambda (x/\lambda)^{k-1} exp{{(x/\lambda)}}^k $')
plt.legend()
plt.title(r'$Weibull:k=%3.2f, \lambda=%3.2f$' % (V[0],V[1]))
```

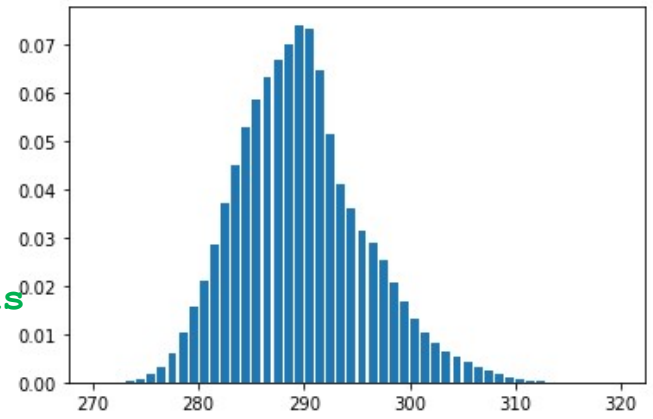

ITER = 44(1000), PERT = 1000, minΔx = 0.00199, k = 2.004, λ = 5.597
Função de custo 2D
$\log_{10} J$
λ(scale)
k(shape)
log10(J)


Weibull : k = 2.00, λ = 5.58
$k/\lambda(x/\lambda)^{k-1}exp(x/\lambda)^k$
Vento simulado 1989-2012
step=0.5


Weibull : k = 2.00, λ = 5.60
$k/\lambda(x/\lambda)^{k-1}exp(x/\lambda)^k$
Vento simulado 1989-2012

Ajuste de uma distribuição normal (gaussiana) à temperatura

```
def gauss(mu,sigma,x):
    nor=1/(sigma*np.sqrt(2*np.pi))*np.exp(-0.5*((x-mu)/sigma)**2)
    return nor
def costG(V):
    mu=V[0];sigma=V[1]
    custo=np.mean((gauss(mu,sigma,tH)-wH)**2)
    return custo

wE=np.copy(Temp);nE=len(wE)
step=1;low=270;high=320
tH=np.arange(low+step/2,high,step) #centro dos bins
bins=np.arange(low,high+step,step) #fronteira dos bins
wH=np.histogram(wE,bins=bins)[0]/nE/step
plt.figure()
plt.bar(tH,wH)
```

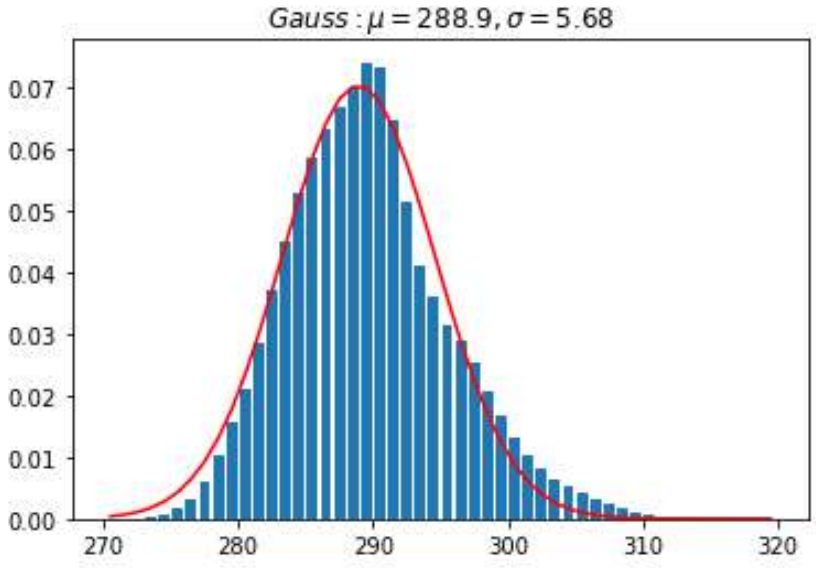


K

Ajuste de uma distribuição normal (gaussiana)

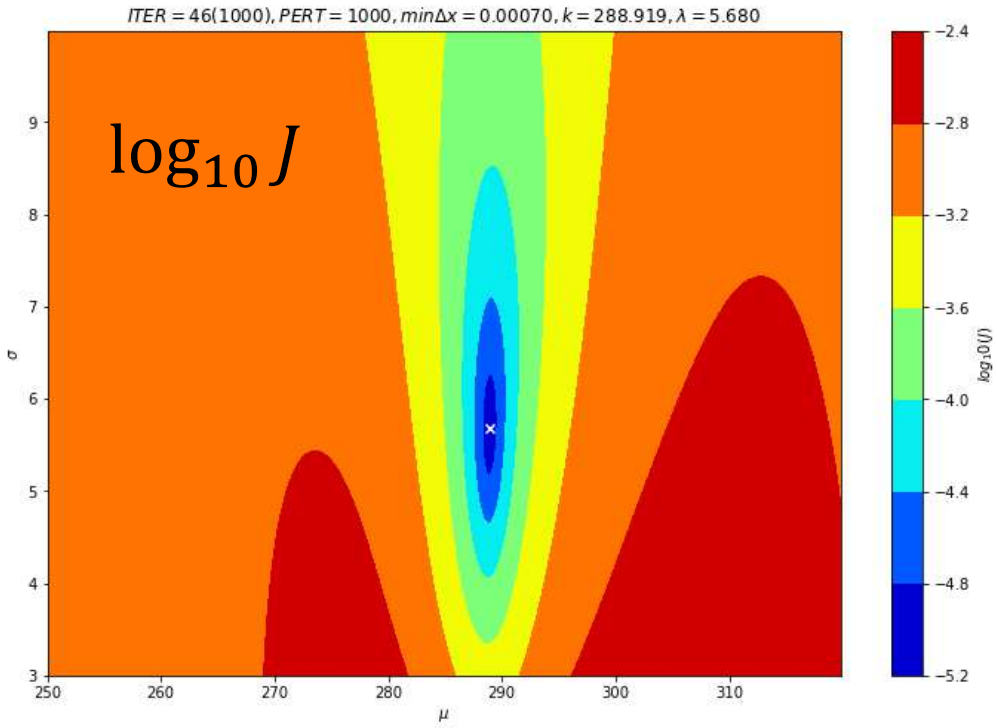
```
V=np.zeros((2))
vnames=np.array([r'$\mu$',r'$\sigma$'])
vmin=np.array([250,0])
vmax=np.array([320,10])
Jmin=-10.
minvstep=(vmax-vmin)/100000
kappa=np.float64(0.1)
T=10.
outITER=1
maxITER=1000
maxPERT=1000
COOL=0.9
n,V,iTER,path=annealD(costG,vmin,vmax,Jmin,minvstep,maxITER,maxPERT,COOL,kappa,\
                      T,outITER,iseed=101)
print(V)
plotcost(costG,vmin,vmax,V,vnames,10,tit=\
         r'$ITER=%3i(%4i),PERT=%5i,min\Delta x=%6.5f,k=%4.3f,\lambda=%4.3f$'%\
          (iTER,maxITER,maxPERT,minvstep[0],V[0],V[1]))
plt.figure()
plt.bar(tH,wH)
gH=gauss(V[0],V[1],tH)
plt.plot(tH,gH,color='red')
plt.title(r'$Gauss:\mu=%3.1f, \sigma=%3.2f$' % (V[0],V[1]))
```

Ajuste gaussiano à Temperatura horária

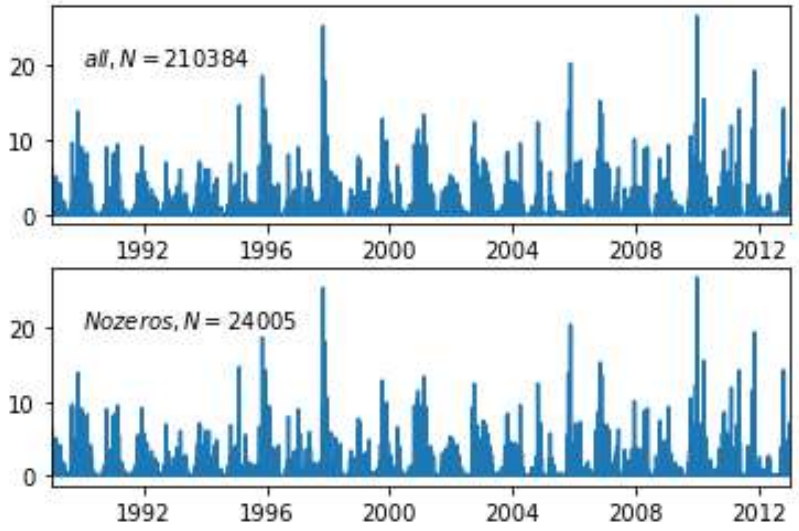


A distribuição desvia-se da normal

```
In [2]: np.mean(Temp)
Out[2]: 289.67671692776065

In [3]: np.std(Temp)
Out[3]: 5.98823346717542
```



Ajuste da precipitação(>0) a uma distribuição Γ




```
index=np.argwhere(R>1e-2)
wE=np.squeeze(R[index])
nE=len(wE)
times=np.array(times)
plt.figure()
plt.subplot(2,1,1)
plt.xlim(datetime.datetime(1989,1,1),datetime.datetime(2013,1,1))
plt.text(datetime.datetime(1990,1,1),20,R'$all, N=%5i$'% len(R))
plt.plot(times,R)
plt.subplot(2,1,2)
plt.plot(times[index],R[index])
plt.xlim(datetime.datetime(1989,1,1),datetime.datetime(2013,1,1))
plt.text(datetime.datetime(1990,1,1),20,\
                        r'$No zeros, N=%5i$' % len(wE))
```

Distribuição Γ

$$GD = \frac{\lambda^{\alpha} x^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)}$$

Onde $\Gamma(\alpha)$ é uma "função especial"

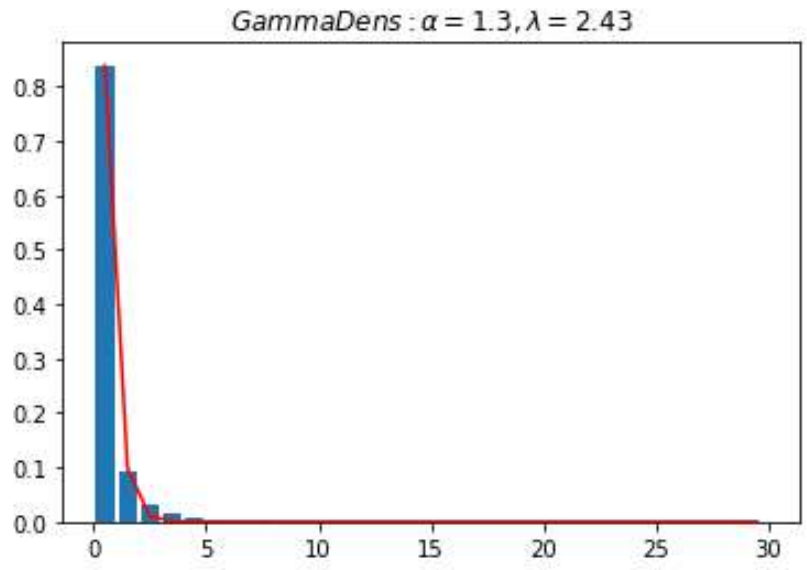
```
from scipy.special import gamma
def gammadens(alpha,lamb,x):
    gd=lamb**alpha*x**(alpha-1)*np.exp(-lamb*x)/gamma(alpha)
    return gd
def costGD(V):
    alpha,lamb=V
    custo=np.mean((gammadens(alpha,lamb,tH)-wH)**2)
    return custo
```

Otimização da distribuição GAMMA

```
step=1;low=0;high=30
tH=np.arange(low+step/2,high,step) #centro
bins=np.arange(low,high+step,step) #fronteira
wH=np.histogram(wE,bins=bins)[0]/nE/step
plt.figure()
plt.bar(tH,wH)
V=np.zeros((2))
vnames=np.array([r'$\alpha$',r'$\lambda$'])
vmin=np.array([0,0])
vmax=np.array([10,10])
Jmin=-10.
minvstep=(vmax-vmin)/100000
kappa=np.float64(0.1);T=10.;outITER=1
maxITER=1000
maxPERT=1000
COOL=0.9

n,V,iTER,path=annealD(costGD,vmin,vmax,Jmin,minvstep,maxITER,maxPERT,COOL,kappa,T,outITER,
iseed=0)
```

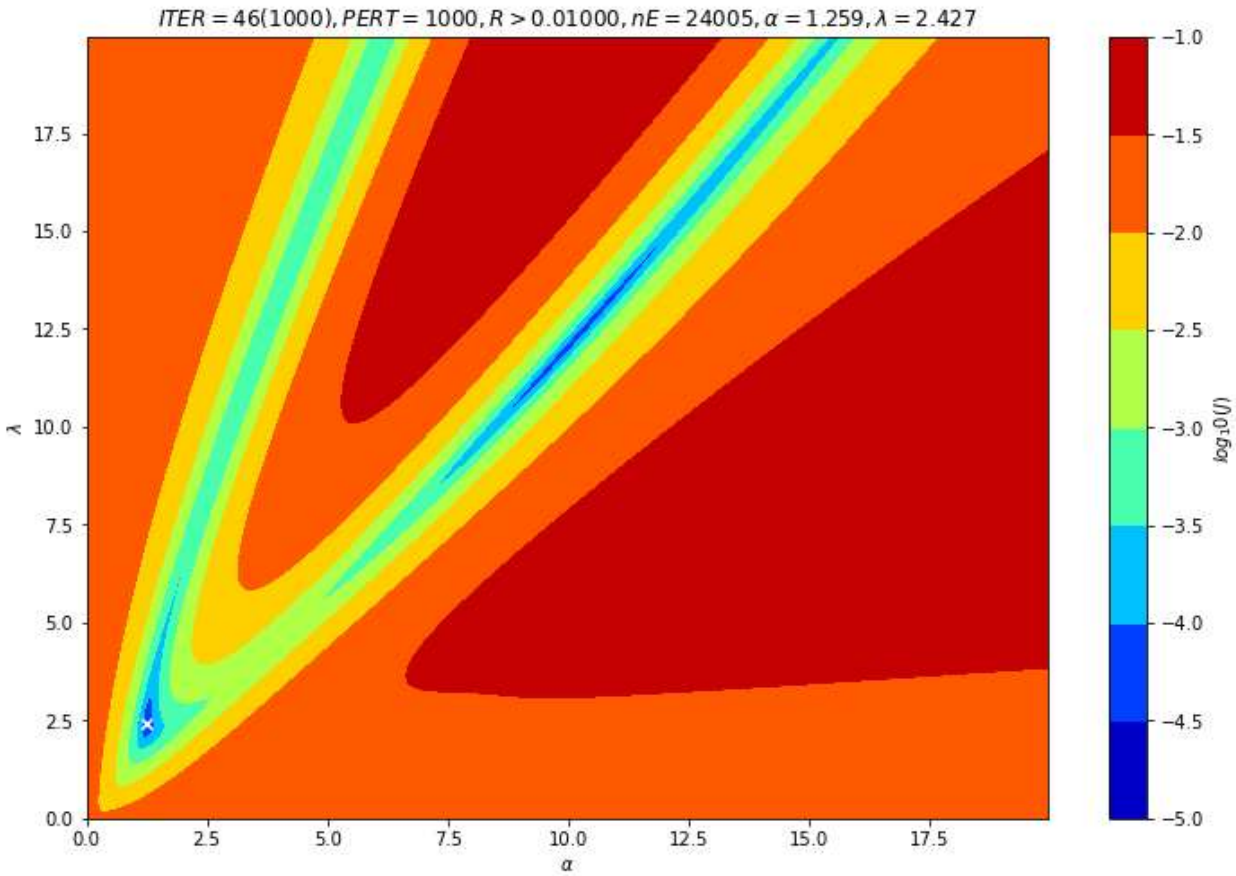
Não inclui “0”

Ajuste GD



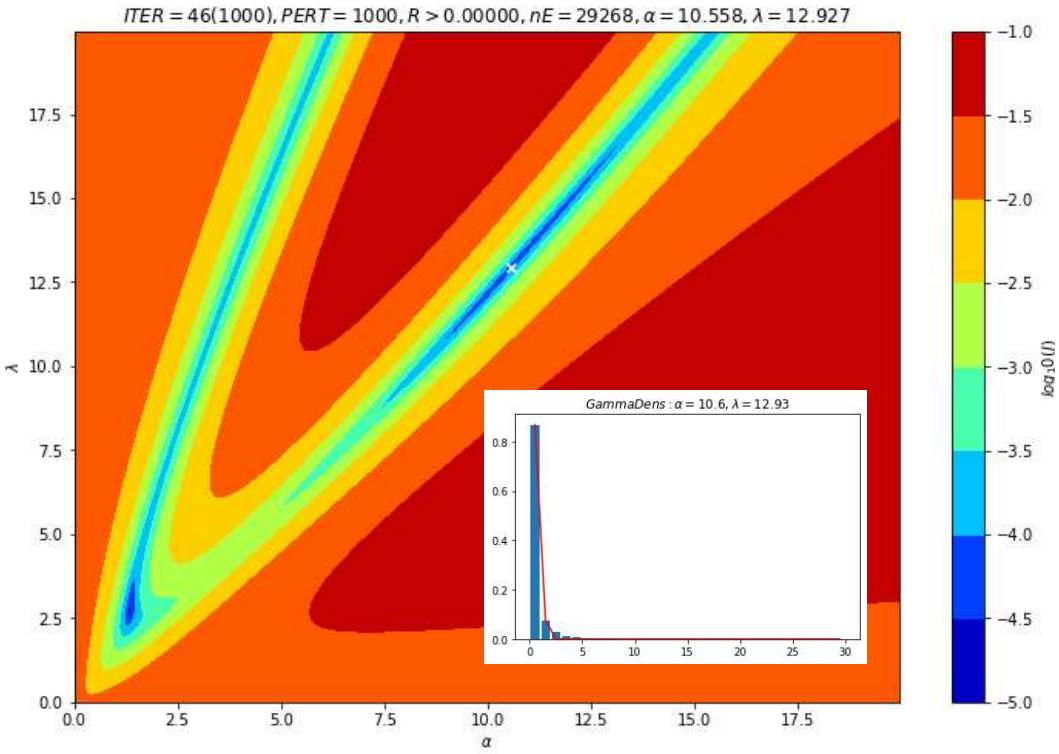

```
plt.figure()
plt.bar(tH,wH)
gH=gammadens(V[0],V[1],tH)
plt.plot(tH,gH,color='red')

plt.title(r'$GammaDens:\alpha=%3.1f, \lambda=%3.2f$' % (V[0],V[1]))
```

Função de custo (geometria complicada)

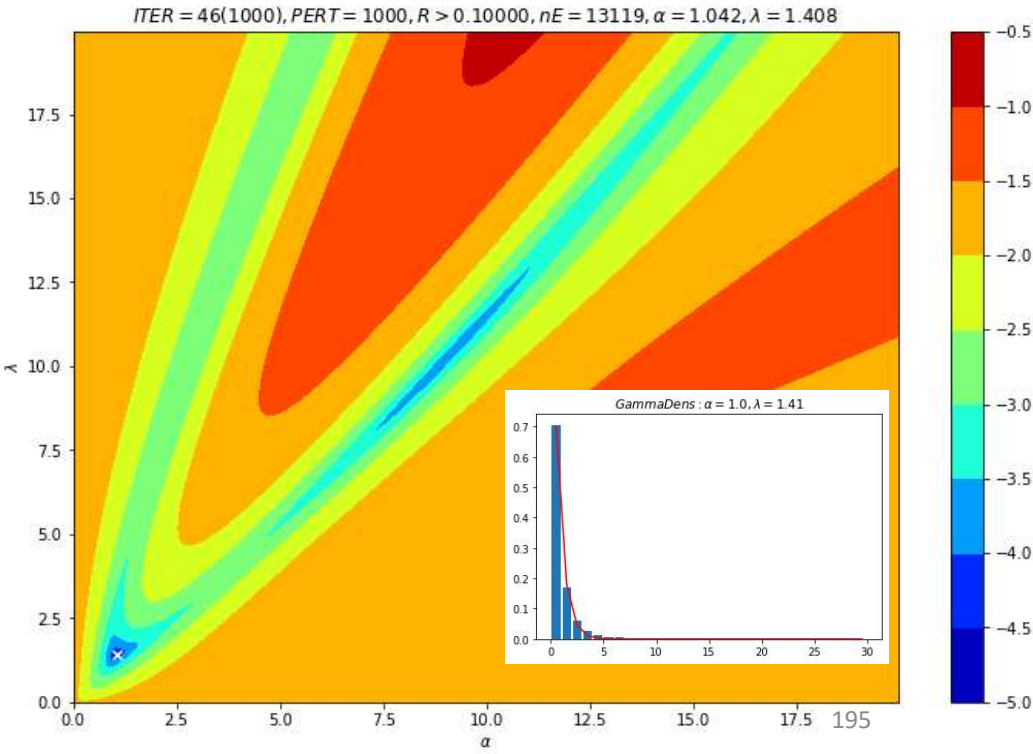


Teste de sensibilidade: existem mínimos locais…

index=np.argwhere(R>0)



index=np.argwhere(R>0.1)



step=0.1 mm/h, R>0.1