Eguratseko energia potentzial eskuragarria

June 15, 2021

[ ]:
```
import netCDF4 as nc
import numpy as np
from numpy import linspace, zeros, random
import scipy
from scipy import integrate
from scipy.special import erf
import math
import matplotlib
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression #Karratu txikienen␣
 ↪metodoarekin erregresio lineala egiteko
import metpy.calc as mpcalc
import scipy.signal as ss
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
import statsmodels
from statsmodels.stats import stattools
import pymannkendall as mk
from statsmodels.graphics import tsaplots
import matplotlib.pyplot as plt

#NetCDF artxiboa irakurtzeko:
ds=nc.Dataset('datuak.nc') #Tenperaturaren datuak 1979-2019 (2019 barne).␣
 ↪Copernicus-etik aterata.

temp=ds['t']
time=ds['time']
lon=ds['longitude']
lat=ds['latitude']
lev=ds['level']

x1=len(temp[0,:,0,0])
y1=len(temp[0,0,:,0])
z1=len(temp[0,0,0,:])
t1=len(temp[:,0,0,0])

x=range(x1)
```

```
y=range(y1)
z=range(z1)
t=range(t1)

landa=lon[:]*2*math.pi/360 #lambda, longitudearen angelua
phi=lat[:]*2*math.pi/360 #latitudearen angelua
cos=np.cos(phi)

a=6371000 #lurraren erradioa
c=1005.9  #c_p
R=287.058 #aire lehorrerako R konstantea
g=9.806   #grabitatea
kappa=R/c
po=1000   #Lurrazaleko presioa hPa-etan
lev2=lev
t2=41 #urte kopurua

ds=nc.Dataset('surfacepressure.nc') #gainazaleko presioa, Pascal-etan.
 ↪Copernicus-etik aterata. Niño-rentzat.
sp=ds['sp']

ds=nc.Dataset('dP.nc') #Presio tarteak (Kodigoan aurrerago ageri da nola dauden
 ↪kalkulatuta).
dp=ds['dP']
dp=dp['dP'][:]
ds=nc.Dataset('gamesdp.nc') #Boer gabeko gamma (Kodigoan aurrerago ageri da nola
 ↪dauden kalkulatuta).
gamma=ds['gamesdp']
gamma=gamma['gamesdp'][:]
ds=nc.Dataset('gamesb.nc') #Gamma Boer-ekin (Kodigoan aurrerago ageri da nola
 ↪dauden kalkulatuta).
gammab=ds['gamesb']
gammab=gammab['gamesb'][:]
ds=nc.Dataset('gammaIH.nc') #Gamma Boer-ekin (Kodigoan aurrerago ageri da nola
 ↪dauden kalkulatuta).
gammaIH=ds['dP']
gammaIH=gammaIH['dP'][:]
ds=nc.Dataset('gammaHH.nc') #Gamma Boer-ekin (Kodigoan aurrerago ageri da nola
 ↪dauden kalkulatuta).
gammaHH=ds['dP']
gammaHH=gammaHH['dP'][:]
# Azken gamma horiek ez dira berdinak banaketa espazialerako eta denbora
 ↪analisirako, batez bestekoak aldatzen baitira.
```

1 Boer-en beta

```
[ ]: presioa=np.zeros((t1,x1,y1,z1))
     for i in range(x1):
         presioa[:,i,:,:]=lev[i]
     b=np.ones((t1,x1,y1,z1), dtype=bool)
     for i in range(x1):
         b[:,i,:,:]=presioa[:,i,:,:]<sp[:]/100 #True false matrize bat sortu
     beta = b.astype(int) # True diren guztiak =1 eta false guztiak =0
```

2 Batez bestekoak eta integralak

```
[ ]: def timeavg(X,t2,x1,y1,z1): #Batez bestekoa denboran. Batez bestekoa urteka␣
      ↪banaketa espazialarentzat.
         Xta=np.zeros((t2,x1,y1,z1))
         for i in range(t2):
             Xta[i,:,:,:]=np.mean(X[12*i:12*(i+1),:,:,:],axis=0)
         return Xta
```

```
[ ]: def zonalavg(X,z1): #Batez bestekoa longitudean zehar Boer-en metodorik gabe.
         za=np.mean(X,3)
         return za
```

```
[ ]: def zonalavgta(X,beta,t2,x1,y1,z1): #Batez bestekoa longitudean zehar Boer-en␣
      ↪metodoarekin.
         Xbeta=X*beta
         Xbetata=np.zeros((t2,x1,y1,z1))
         betata=np.zeros((t2,x1,y1,z1))
         Xta=np.zeros((t2,x1,y1,z1))

         for i in range(t2):
             Xbetata[i,:,:,:]=np.mean(Xbeta[12*i:12*(i+1),:,:],axis=0)
             betata[i,:]=np.mean(beta[12*i:12*(i+1),:],axis=0)
             Xta[i,:]=np.mean(X[12*i:12*(i+1),:],axis=0)

         Xbetataza=np.mean(Xbetata,3)
         betataza=np.mean(betata,3)
         Xtaza=np.mean(Xta,3)

         zata=np.zeros((t2,x1,y1))
         for i in range(t2):
             for j in range(x1):
                 for k in range(y1):
                     if betataza[i,j,k]!=0:
                         zata[i,j,k]=Xbetataza[i,j,k]/betataza[i,j,k]
                     else:
```

```
                    zata[i,j,k]=Xtaza[i,j,k]

    return zata
```

[ ]:
```
def areaavg(X,t1,x1,y1,z1): #Azalerako batez bestekoa Boer-en metodorik gabe.
    Xza=np.mean(X,3)
    Xza2=np.zeros((t1,x1,y1))
    for i in y:
        Xza2[:,:,i]=Xza[:,:,i]*cos[i]
    aa=-scipy.integrate.simps(Xza2,phi,axis=2)/2
    return aa
```

[ ]:
```
def areaavgIH(X,t1,x1,y1,z1):
    Xza=np.mean(X,3)
    Xza2=np.zeros((t1,x1,91))
    for i in range(91):
        Xza2[:,:,i]=Xza[:,:,i]*cos[i]
    aa=-scipy.integrate.simps(Xza2,phi[:91],axis=2)
    return aa
```

[ ]:
```
def areaavgHH(X,t1,x1,y1,z1):
    Xza=np.mean(X,3)
    Xza2=np.zeros((t1,x1,91))
    for i in range(91):
        Xza2[:,:,i]=Xza[:,:,90+i]*cos[90+i]
    aa=-scipy.integrate.simps(Xza2,phi[90:],axis=2)
    return aa
```

[ ]:
```
def areaavgta(X,beta,t1,x1,y1,z1): #Azalerako batez bestekoa Boer-en␣
 ↪metodoarekin.
    Xbeta=X*beta
    Xbetata=np.zeros((t1,x1,y1,z1))
    betata=np.zeros((t1,x1,y1,z1))
    Xta=np.zeros((t1,x1,y1,z1))

    Xbetata=timeavg(Xbeta,t1,x1,y1,z1)
    betata=timeavg(beta,t1,x1,y1,z1)
    Xta=timeavg(X,t1,x1,y1,z1)

    betataza=np.mean(betata,3)
    Xbetataza=np.mean(Xbetata,3)
    Xbetataza2=np.zeros((t1,x1,y1))
    betataza2=np.zeros((t1,x1,y1))
    for i in y:
        Xbetataza2[:,:,i]=Xbetataza[:,:,i]*cos[i]
        betataza2[:,:,i]=betataza[:,:,i]*cos[i]
    Xbetataaa=-scipy.integrate.simps(Xbetataza2,phi,axis=2)/2
```

```
    betataaa=-scipy.integrate.simps(betataza2,phi,axis=2)/2

    aa=Xbetataaa/betataaa

    return aa
```

[ ]:
```
def areaavgtaIH(X,beta,t1,x1,y1,z1): #Azalerako batez bestekoa Boer-en␣
↪metodoarekin.
    Xbeta=X[:,:,:91,:]*beta[:,:,:91,:]
    Xbetata=np.zeros((t1,x1,91,z1))
    betata=np.zeros((t1,x1,91,z1))
    Xta=np.zeros((t1,x1,91,z1))

    Xbetata=timeavg(Xbeta,t1,x1,91,z1)
    betata=timeavg(beta,t1,x1,91,z1)
    Xta=timeavg(X,t1,x1,y1,z1)

    betataza=np.mean(betata,3)
    Xbetataza=np.mean(Xbetata,3)
    Xbetataza2=np.zeros((t1,x1,91))
    betataza2=np.zeros((t1,x1,91))
    for i in range(91):
        Xbetataza2[:,:,i]=Xbetataza[:,:,i]*cos[i]
        betataza2[:,:,i]=betataza[:,:,i]*cos[i]
    Xbetataaa=-scipy.integrate.simps(Xbetataza2,phi[:91],axis=2)/2
    betataaa=-scipy.integrate.simps(betataza2,phi[:91],axis=2)/2

    aa=Xbetataaa/betataaa

    return aa
```

[ ]:
```
def areaavgtaHH(X,beta,t1,x1,y1,z1): #Azalerako batez bestekoa Boer-en␣
↪metodoarekin.
    Xbeta=X[:,:,90:181,:]*beta[:,:,90:181,:]
    Xbetata=np.zeros((t1,x1,y1,z1))
    betata=np.zeros((t1,x1,y1,z1))
    Xta=np.zeros((t1,x1,y1,z1))

    Xbetata=timeavg(Xbeta,t1,x1,y1,z1)
    betata=timeavg(beta,t1,x1,y1,z1)
    Xta=timeavg(X,t1,x1,y1,z1)

    betataza=np.mean(betata,3)
    Xbetataza=np.mean(Xbetata,3)
    Xbetataza2=np.zeros((t1,x1,91))
    betataza2=np.zeros((t1,x1,91))
    for i in range(91):
```

```
        Xbetataza2[:,:,i]=Xbetataza[:,:,i]*cos[90+i]
        betataza2[:,:,i]=betataza[:,:,i]*cos[90+i]
    Xbetataaa=-scipy.integrate.simps(Xbetataza2,phi[90:181],axis=2)/2
    betataaa=-scipy.integrate.simps(betataza2,phi[90:181],axis=2)/2

    aa=Xbetataaa/betataaa

    return aa
```

[ ]:
```
def int3D(X,t2,x1,y1,z1,dp):
    dp2=timeavg(dp,t2,x1,y1,z1) #Hau erabili Boer gabe
    integrakizuna=X*a**2/g
    integrakizunacos=np.zeros((t2,x1,y1,z1))
    for i in y:
        integrakizunacos[:,:,i,:]=integrakizuna[:,:,i,:]*cos[i]
    vertint=np.sum(integrakizunacos*dp2,axis=1) #Hau erabili Boer gabe
    vertint=scipy.integrate.simps(integrakizunacos,lev,axis=1) #Hau erabili␣
 ↪Boer-ekin
    vertintlon=np.mean(vertint,axis=2)*2*math.pi
    integral=-scipy.integrate.simps(vertintlon,phi,axis=1)

    return integral
```

3 dP presio tarteak

[ ]:
```
dP=np.zeros((t1,x1,y1,z1))
presioa2=np.zeros((t1,x1+1,y1,z1))

presioa2[:,21,:,:]=1045 #20 dP sortzeko sartu behar da maila hau, balioak ez du␣
 ↪axola sp-ko maximoa baino handiagoa den bitartean.
for i in x:
    presioa2[:,i,:,:]=lev[i]

b=np.ones((t1,x1+1,y1,z1), dtype=bool)
for i in range(x1+1):
    b[:,i,:,:]=presioa2[:,i,:,:]<sp[:]/100
beta2 = b.astype(int)

betarollbehera=np.roll(beta2,1,axis=1)
betarollbehera[:,0,:,:]=1
betarollgora=np.roll(beta2,-1,axis=1)
betarollgora[:,21,:,:]=0
pbeta=presioa2*beta2

dP[:,0,:,:]=lev[1]-lev[0]
for i in x[1:]:
```

```
      dP[:,i,:,:]=(presioa2[:,i+1,:,:]-presioa2[:,i-1,:,:])/2
pbeta=presioa2*beta2

dP[:,0,:,:]=lev[1]-lev[0]
for i in x[1:]:
      dP[:,i,:,:]=(presioa2[:,i+1,:,:]-presioa2[:,i-1,:,:])/2
muga=beta2-betarollgora
dPgainazal=sp[:]/100-(np.sum(muga*presioa2,axis=1)+np.sum(np.
 ↪roll(muga,-1,axis=1)*presioa2,axis=1))/2
dPgainazal2=np.zeros((t1,x1,y1,z1))
for i in x:
    dPgainazal2[:,i,:,:]=muga[:,i,:,:]*dPgainazal
dP=dP*beta-dP*muga[:,:21,:,:]+dPgainazal2
```

[ ]:
```
#netCDF artxiboa sortzeko. Ondoren Panoply programarekin irudikatuko dira␣
 ↪artxiboko datuak.
denbora=t
sf=(np.max(dP)-np.min(dP))/(2**16-1)  #Enpaketatzeko eta desenpaketatzeko
ao=np.min(dP)+2**(16-1)*sf  #Enpaketatzeko eta desenpaketatzeko
g = nc.Dataset('dP.nc','w', format='NETCDF4')
tempdevgrp = g.createGroup('dP')

tempdevgrp.createDimension('lon', 360)
tempdevgrp.createDimension('lat', 181)
tempdevgrp.createDimension('lev', 21)
tempdevgrp.createDimension('t', 492)

lon = tempdevgrp.createVariable('lon', 'f4', 'lon')
lat = tempdevgrp.createVariable('lat', 'f4', 'lat')
lev = tempdevgrp.createVariable('lev', 'i4', 'lev')
time = tempdevgrp.createVariable('t','i4','t')
temp_bbdeviation = tempdevgrp.createVariable('dP', 'f4', ('t','lev','lat','lon'))

tempdevgrp.scale_factor=sf
tempdevgrp.add_offset=ao

lon[:] = linspace(0,359,360, endpoint=True)
lat[:] = linspace(90,-90,181, endpoint=True)
lev[:] = np.array(lev2,dtype=float)
time[:] = denbora[:]

temp_bbdeviation[:] = dP[:]

g.close()

#Irakurlearentzat luzeegia ez izateko, netCDF artxiboak sortzeko kodea hemen␣
 ↪bakarrik erakutsiko da.
```

4 Gamma Boer gabe

```
[ ]: #Denborako batez bestekoaren arabera aldatu egiten da
tempta=timeavg(temp,t2,x1,y1,z1)
temptaaa=areaavg(tempta,t2,x1,y1,z1)
dtdp=np.gradient(temp,lev,axis=1) #Tenperaturaren presioarekiko deribatua puntu␣
 ↪eta garai bakoitzerako.
gam=np.zeros((t1,x1,y1,z1))
for i in x:
    gam[:,i,:,:]=dtdp[:,i,:,:]*(-g)*lev[i]/R/temp[:,i]
gamta=timeavg(gam,t2,x1,y1,z1)
gamtaaa=areaavg(gamta,t2,x1,y1,z1)
gamma=g/c/temptaaa/(g/c+gamtaaa)
```

```
[ ]: #GammaIH
tempta=timeavg(temp,t1,x1,y1,z1)
temptaaa=areaavgIH(tempta,480,x1,y1,z1)
dtdp=np.gradient(temp,lev,axis=1) #Tenperaturaren presioarekiko deribatua puntu␣
 ↪eta garai bakoitzerako.
gam=np.zeros((492,x1,y1,z1))
for i in x:
    gam[:,i,:,:]=dtdp[:,i,:,:]*(-g)*lev[i]/R/temp[:,i]
gamta=timeavg(gam,t1,x1,y1,z1)
gamtaaa=areaavgIH(gamta,480,x1,y1,z1)
gamma=g/c/temptaaa/(g/c+gamtaaa)
```

```
[ ]: #GammaHH
temptaaa=areaavgHH(tempta,480,x1,y1,z1)
gamtaaa=areaavgHH(gamta,480,x1,y1,z1)
gamma=g/c/temptaaa/(g/c+gamtaaa)
```

5 Gamma Boer-ekin

```
[ ]: #Denborako batez bestekoaren arabera aldatu egiten da
#IH eta HH lortzeko goian bezala
temptaaa=areaavgta(temp,beta,t2,x1,y1,z1)
dtdp=np.gradient(temp,lev,axis=1) #Tenperaturaren presioarekiko deribatua puntu␣
 ↪eta garai bakoitzerako.
gam=np.zeros((t1,x1,y1,z1))
for i in x:
    gam[:,i,:,:]=dtdp[:,i,:,:]*(-g)*lev[i]/R/temp[:,i]
gamtaaa=areaavgta(gam,beta,t2,x1,y1,z1)
gamma=g/c/temptaaa/(g/c+gamtaaa)
gammab=gamma
```

6 Banaketa espaziala

[ ]:
```
#Banaketak kalkulatzeko beharko diren batez bestekoak
betata=timeavg(beta,t2,x1,y1,z1) #Boer-ekin
betataza=np.mean(betata,3) #Boer-ekin
tempta=timeavg(temp,t2,x1,y1,z1)
temptaaa=areaavg(tempta,t2,x1,y1,z1)
temptaza=np.mean(tempta,3)
```

7 EPEZ

[ ]:
```
#Boer gabeko kalkulua. Pm=EPEZ bezala ulertu.
temptaaa2=np.zeros((t2,x1,y1))
for j in range(t2):
    for i in y:
        temptaaa2[j,:,i]=temptaaa[j,:]

biderketarako=(temptaza-temptaaa2)**2
Pm_JperKg=np.zeros((t2,x1,y1))
for i in x:
    for j in range(t2):
        Pm_JperKg[j,i,:]=gamma[j,i]*biderketarako[j,i,:]*0.5*c*100 #latitudea vs␣
 ↪presioa denboran zehar (41 urteetarako)
Pm_JperKgta=np.mean(Pm_JperKg,0) #41 urtetako batez bestekoa
Pm_JperKg2=np.zeros((t2,x1,y1,z1))
for i in z:
    Pm_JperKg2[:,:,:,i]=Pm_JperKg # Integralaren funtzioan sartzeko

Pmt=int3D(Pm_JperKg2,t2,x1,y1,z1,dp)/(4*math.pi*a**2) # Azalera unitateko EPEZen␣
 ↪balioa (41 urteetarako)
np.mean(Pmt,0) #Batez bestekoa 41 urteetan
```

[ ]:
```
#Boer-ekin
temptaaa2=np.zeros((t2,x1,y1))
for j in range(t2):
    for i in y:
        temptaaa2[j,:,i]=temptaaa[j,:]

biderketa=betataza*(temptaza-temptaaa2)**2
Pm_JperKg=np.zeros((t2,x1,y1))
for i in x:
    for j in range(t2):
        Pm_JperKg[j,i,:]=gammab[j,i]*biderketa[j,i,:]*0.5*c*100 #latitudea vs␣
 ↪presioa denboran zehar (41 urteetarako)
Pm_JperKgta=np.mean(Pm_JperKg,0) #41 urtetako batez bestekoa
Pm_JperKg2=np.zeros((t2,x1,y1,z1))
```

```
for i in z:
    Pm_JperKg2[:,:,:,i]=Pm_JperKg # Integralaren funtzioan sartzeko

Pmt=int3D(Pm_JperKg2,t2,x1,y1,z1,dp)/(4*math.pi*a**2) # Azalera unitateko EPEZen balioa (41 urteetarako)
np.mean(Pmt,0) #Batez bestekoa 41 urteetan
```

8 EEPE iraunkorra

[ ]:
```
#Boer gabe
#Pestatt=EEPE iragankorra

dp2=timeavg(dp,t2,x1,y1,z1)
temptaza2=np.zeros((t2,x1,y1,z1))
for i in z:
    temptaza2[:,:,:,i]=temptaza

biderketarako=(tempta-temptaza2)**2

Pestat_JperKg=np.zeros((t2,x1,y1,z1))
for i in range(x1):
    for j in range(t2):
        Pestat_JperKg[j,i,:,:]=gamma[j,i]*biderketarako[j,i,:,:]*0.5*c*100 #latitudea vs presioa vs longitudea denboran zehar (41 urteetarako)

Pe_statJperKg_longlat=np.sum(Pestat_JperKg*dp2,axis=1)/g #Presio guztien ekarpena batu latitudea vs longitudea lortzeko
Pe_statJperKg_longlatta=np.mean(Pe_statJperKg_longlat,0) #Aurrekoaren batez bestekoa 41 urteetan
Pestat_JperKgta=np.mean(np.mean(Pestat_JperKg,3),0) # latitudea vs presioa lortzeko denboran batez bestekoa eginda
Pestatt=int3D(Pestat_JperKg,t2,x1,y1,z1,dp)/(4*math.pi*a**2) #EEPE iraunkorraren azalerako balioak

np.mean(Pestatt,axis=0) #Batez bestekoa 41 urteetan
```

[ ]:
```
#Boer-ekin
#Pestatt=EEPE iragankorra

temptaza2=np.zeros((t2,x1,y1,z1))
for i in z:
    temptaza2[:,:,:,i]=temptaza

biderketa=betata*(tempta-temptaza2)**2
```

```
Pestat_JperKg=np.zeros((t2,x1,y1,z1))
for i in range(x1):
    for j in range(t2):
        Pestat_JperKg[j,i,:,:]=gammab[j,i]*biderketa[j,i,:,:]*0.5*c*100
 ↪#latitudea vs presioa vs longitudea denboran zehar (41 urteetarako)

Pe_statJperKg_longlat=scipy.integrate.simps(Pestat_JperKg,lev[:],axis=1)/g
 ↪#Presio guztien ekarpena batu latitudea vs longitudea lortzeko
Pe_statJperKg_longlatta=np.mean(Pe_statJperKg_longlat,0) #Aurrekoaren batez
 ↪bestekoa 41 urteetan
Pestat_JperKgta=np.mean(np.mean(Pestat_JperKg,3),0) # latitudea vs presioa
 ↪lortzeko denboran batez bestekoa eginda
Pestatt=int3D(Pestat_JperKg,t2,x1,y1,z1,dp)/(4*math.pi*a**2) #EEPE iraunkorraren
 ↪azalerako balioak


np.mean(Pestatt,axis=0) #Batez bestekoa 41 urteetan
```

9 EEPE iragankorra

[ ]:
```
#Boer gabe
tempta=timeavg(temp,t2,x1,y1,z1)
dp2=timeavg(dp,t2,x1,y1,z1)
tempta2=np.zeros((t1,x1,y1,z1))
for k in range(t2):
    for i in range(12):
        tempta2[i+k*12,:,:,:]=tempta[k,:,:,:]
biderketarako=(temp[:]-tempta2)**2
biderketarako2=timeavg(biderketarako,t2,x1,y1,z1)
Petrans_JperKg=np.zeros((t2,x1,y1,z1))
for i in x:
    for j in range(t2):
        Petrans_JperKg[j,i,:,:]=gamma[j,i]*biderketarako2[j,i,:,:]*0.5*c*100

Pe_transJperKg_longlat=np.sum(Petrans_JperKg*dp2,axis=1)/g
Petranst=int3D(Petrans_JperKg,t2,x1,y1,z1,dp)/(4*math.pi*a**2)
Petrans_JperKgta=np.mean(np.mean(Petrans_JperKg,3),0)
Pe_transJperKg_longlatta=np.mean(Pe_transJperKg_longlat,0)
np.mean(Petranst,axis=0)
```

[ ]:
```
#Boer-ekin
tempta=timeavg(temp,t2,x1,y1,z1)
betata=timeavg(beta,t2,x1,y1,z1)
tempta2=np.zeros((t1,x1,y1,z1))
for k in range(t2):
```

```
    for i in range(12):
        tempta2[i+k*12,:,:,:]=tempta[k,:,:,:]
biderketarako=beta*(temp[:]-tempta2)**2
biderketarako2=timeavg(biderketarako,t2,x1,y1,z1)
Petrans_JperKg=np.zeros((t2,x1,y1,z1))
for i in x:
    for j in range(t2):
        Petrans_JperKg[j,i,:,:]=gammab[j,i]*biderketarako2[j,i,:,:]*0.5*c*100
Pe_transJperKg_longlat=scipy.integrate.simps(Petrans_JperKg,lev[:],axis=1)/g
Petranst=int3D(Petrans_JperKg,t2,x1,y1,z1,dp)/(4*math.pi*a**2)
Petrans_JperKgta=np.mean(np.mean(Petrans_JperKg,3),0)
Pe_transJperKg_longlatta=np.mean(Pe_transJperKg_longlat,0)
np.mean(Petranst,axis=0)
```

10 ———————— (banaketa espazialarena hemen amaitzen da)

11 Banaketak denboran zehar

[ ]:
```
# 41 datu gutxiegi direnez moving average erabiliko da. Horregatik, timeavg␣
 ↪aldatu egin behar da.
# Honekin gamma eta energien balioak ere aldatu egingo dira apur bat eta␣
 ↪denboraren mugetako balio batzuk desagertuko dira.
# Banaketak denboran zehar lortzeko aldaketa bakarra ondorengoa da:
# Boerren banaketak erabiliko dira.

def timeavg(X,t1,x1,y1,z1):
    Xta=np.zeros((t1-12,x1,y1,z1)) #EEPE iragankorra kalkulatzean -24 erabili
    for i in t[6:486]: # eta hemen 6:474
        Xta[i-6,:,:,:]=np.mean(X[i-6:i+6,:,:,:],axis=0)
    return Xta
```

12 Denbora analisia

[ ]:
```
# Azken timeavg horrekin, denbora analisirako gehien behar diren funtzioak:
# Boer gabeko balioekin egingo da denbora analisia.
tempta=timeavg(temp,t1,x1,y1,z1)
temptaaa=areaavg(tempta,t1-12,x1,y1,z1)
temptaza=np.mean(tempta,3)
```

13 EPEZ

[ ]:
```
temptaaa2=np.zeros((t1-12,x1,y1))
for j in range(t1-12):
    for i in y:
        temptaaa2[j,:,i]=temptaaa[j,:]

biderketarako=(temptaza-temptaaa2)**2
Pm_JperKg=np.zeros((t1-12,x1,y1))
for i in x:
    for j in range(t1-12):
        Pm_JperKg[j,i,:]=gamma[j,i]*biderketarako[j,i,:]*0.5*c*100
Pm_JperKg2=np.zeros((t1-12,x1,y1,z1))
for i in z:
    Pm_JperKg2[:,:,:,i]=Pm_JperKg[:,:,:]

Pmt=int3D(Pm_JperKg2,t1,x1,y1,z1,dp)/(4*math.pi*a**2) #Pmt=EPEZ azalera␣
↪unitateko balioa 480 denbora puntutarako.
```

14 EEPE iraunkorra

[ ]:
```
temptaza2=np.zeros((t1-12,x1,y1,z1))
for i in z:
    temptaza2[:,:,:,i]=temptaza

biderketarako=(tempta-temptaza2)**2

Pestat_JperKg=np.zeros((t1-12,x1,y1,z1))
for i in range(x1):
    for j in range(t1-12):
        Pestat_JperKg[j,i,:,:]=gamma[j,i]*biderketarako[j,i,:,:]*0.5*c*100
Pestatt=int3D(Pestat_JperKg,t1,x1,y1,z1,dp)/(4*math.pi*a**2) #Pestatt=EEPE␣
↪iraunkorraren azalera unitateko balioa 480 denbora puntutarako.
```

15 EEPE iragankorra

[ ]:
```
biderketa=(temp[6:486]-tempta)**2
biderketa2=timeavg(biderketa,t1,x1,y1,z1)
Petrans_JperKg=np.zeros((t1-24,x1,y1,z1))
for i in x:
    for j in range(t1-24):
        Petrans_JperKg[j,i,:,:]=gamma[6+j,i]*biderketa2[j,i,:,:]*0.5*c*100
Petranst=int3D(Petrans_JperKg,t1,x1,y1,z1,dp)/(4*math.pi*a**2) #Pestatt=EEPE␣
↪iragankorraren azalera unitateko balioa 480 denbora puntutarako.
```

16 Hemisferioetarako

17 EPEZ

[ ]:
```
#IH dP (Boer erabiltzeko beste integrala erabili)
temptaaa=areaavgIH(tempta,t1-12,x1,y1,z1)
temptaaa2=np.zeros((t1-12,x1,91))
for j in range(t1-12):
    for i in range(91):
        temptaaa2[j,:,i]=temptaaa[j,:]

biderketarako=(temptaza[:,:,:91]-temptaaa2)**2
Pm_JperKg=np.zeros((t1-12,x1,91))
for i in x:
    for j in range(t1-12):
        Pm_JperKg[j,i,:]=gammaIH[j,i]*biderketarako[j,i,:]*0.5*c*100
dp2=np.mean(timeavg(dp,492,x1,y1,z1),3)#Hau erabili Boer gabe
integrakizuna=Pm_JperKg*a**2/g
integrakizunacos=np.zeros((480,x1,91))
for i in range(91):
    integrakizunacos[:,:,i]=integrakizuna[:,:,i]*cos[i]
vertint=np.sum(integrakizunacos*dp2[:,:,:91],axis=1)*2*math.pi #Hau erabili Boer␣
 ↪gabe
#vertint=scipy.integrate.simps(integrakizunacos,lev,axis=1) #Hau erabili␣
 ↪Boer-ekin
integral=-scipy.integrate.simps(vertint,phi[:91],axis=1)/(2*math.pi*a**2)
```

[ ]:
```
#HH dP (Boer erabiltzeko beste integrala erabili)
temptaaa=areaavgHH(tempta,t1-12,x1,y1,z1)
temptaaa2=np.zeros((t1-12,x1,91))
for j in range(t1-12):
    for i in range(91):
        temptaaa2[j,:,i]=temptaaa[j,:]

biderketarako=(temptaza[:,:,90:181]-temptaaa2)**2
Pm_JperKg=np.zeros((t1-12,x1,91))
for i in x:
    for j in range(t1-12):
        Pm_JperKg[j,i,:]=gammaHH[j,i]*biderketarako[j,i,:]*0.5*c*100
dp2=np.mean(timeavg(dp,492,x1,y1,z1),3)#Hau erabili Boer gabe
integrakizuna=Pm_JperKg*a**2/g
integrakizunacos=np.zeros((480,x1,91))
for i in range(91):
    integrakizunacos[:,:,i]=integrakizuna[:,:,i]*cos[90+i]
vertint=np.sum(integrakizunacos*dp2[:,:,90:181],axis=1)*2*math.pi #Hau erabili␣
 ↪Boer gabe
```

```
#vertint=scipy.integrate.simps(integrakizunacos,lev,axis=1) #Hau erabili␣
 ↪Boer-ekin
integral=-scipy.integrate.simps(vertint,phi[90:181],axis=1)/(2*math.pi*a**2)
```

[ ]:
```
np.mean(integral)
```

18 EEPE iraunkorra

[ ]:
```
#IH (Boer erabiltzeko beste integrala erabili)
temptaza2=np.zeros((t1-12,x1,91,z1))
for i in z:
    temptaza2[:,:,:,i]=temptaza[:,:,:91]

biderketa=(tempta[:,:,:91,:]-temptaza2)**2

Pestat_JperKg=np.zeros((t1-12,x1,91,z1))
for i in range(x1):
    for j in range(t1-12):
        Pestat_JperKg[j,i,:,:]=gammaIH[j,i]*biderketa[j,i,:,:]*0.5*c*100
dp2=timeavg(dp,492,x1,y1,z1)#Hau erabili Boer gabe
integrakizuna=Pestat_JperKg*a**2/g
integrakizunacos=np.zeros((480,x1,91,z1))
for i in range(91):
    integrakizunacos[:,:,i,:]=integrakizuna[:,:,i,:]*cos[i]
vertint=np.sum(integrakizunacos*dp2[:,:,:91,:],axis=1)
vertintlon=np.mean(vertint,2)*2*math.pi
integral=-scipy.integrate.simps(vertintlon,phi[:91],axis=1)/(2*math.pi*a**2)
```

[ ]:
```
np.mean(integral)
```

[ ]:
```
#HH (Boer erabiltzeko beste integrala erabili)
temptaza2=np.zeros((t1-12,x1,91,z1))
for i in z:
    temptaza2[:,:,:,i]=temptaza[:,:,90:181]

biderketa=(tempta[:,:,90:181,:]-temptaza2)**2

Pestat_JperKg=np.zeros((t1-12,x1,91,z1))
for i in range(x1):
    for j in range(t1-12):
        Pestat_JperKg[j,i,:,:]=gammaHH[j,i]*biderketa[j,i,:,:]*0.5*c*100
dp2=timeavg(dp,492,x1,y1,z1)#Hau erabili Boer gabe
integrakizuna=Pestat_JperKg*a**2/g
integrakizunacos=np.zeros((480,x1,91,z1))
for i in range(91):
    integrakizunacos[:,:,i,:]=integrakizuna[:,:,i,:]*cos[90+i]
```

```
vertint=np.sum(integrakizunacos*dp2[:,:,90:181,:],axis=1)
vertintlon=np.mean(vertint,2)*2*math.pi
integral=-scipy.integrate.simps(vertintlon,phi[90:181],axis=1)/(2*math.pi*a**2)
np.mean(integral)
```

19 EEPE iragankorra

[ ]:
```
#IH (Boer erabiltzeko beste integrala erabili)
biderketa=(temp[6:486,:,:91,:]-tempta[:,:,:91,:])**2
biderketa2=timeavg(biderketa,t1,x1,91,z1)
Petrans_JperKg=np.zeros((t1-24,x1,91,z1))
for i in x:
    for j in range(t1-24):
        Petrans_JperKg[j,i,:,:]=gammaIH[6+j,i]*biderketa2[j,i,:,:]*0.5*c*100
```

[ ]:
```
dp2=timeavg(dp,492,x1,y1,z1)#Hau erabili Boer gabe
integrakizuna=Petrans_JperKg*a**2/g
integrakizunacos=np.zeros((468,x1,91,z1))
for i in range(91):
    integrakizunacos[:,:,i,:]=integrakizuna[:,:,i,:]*cos[i]
vertint=np.sum(integrakizunacos*dp2[6:474,:,:91,:],axis=1)
vertintlon=np.mean(vertint,2)*2*math.pi
integral=-scipy.integrate.simps(vertintlon,phi[:91],axis=1)/(2*math.pi*a**2)
```

[ ]:
```
np.mean(integral)
```

[ ]:
```
#HH (Boer erabiltzeko beste integrala erabili)
biderketa=(temp[6:486,:,90:181,:]-tempta[:,:,90:181,:])**2
biderketa2=timeavg(biderketa,t1,x1,91,z1)
Petrans_JperKg=np.zeros((t1-24,x1,91,z1))
for i in x:
    for j in range(t1-24):
        Petrans_JperKg[j,i,:,:]=gammaHH[6+j,i]*biderketa2[j,i,:,:]*0.5*c*100
```

[ ]:
```
dp2=timeavg(dp,492,x1,y1,z1)#Hau erabili Boer gabe
integrakizuna=Petrans_JperKg*a**2/g
integrakizunacos=np.zeros((468,x1,91,z1))
for i in range(91):
    integrakizunacos[:,:,i,:]=integrakizuna[:,:,i,:]*cos[90+i]
vertint=np.sum(integrakizunacos*dp2[6:474,:,90:181,:],axis=1)
vertintlon=np.mean(vertint,2)*2*math.pi
integral=-scipy.integrate.simps(vertintlon,phi[90:181],axis=1)/(2*math.pi*a**2)
```

[ ]:
```
np.mean(integral)
```

20 Zikloak kendu denbora serieei

[ ]:
```
# Denbora serieak. Pmt=EPEZ, Petranst= EEPE iragankorra eta Pestatt= EEPE␣
 ↪iraunkorra
ds=nc.Dataset('Pmt.nc')
Pmt=ds['Pmt']
Pmt=Pmt['Pmt'][:]
ds=nc.Dataset('Petranst.nc')
Petranst=ds['Petranst']
Petranst=Petranst['Petranst'][:]
ds=nc.Dataset('Pestatt.nc')
Pestatt=ds['Pestatt']
Pestatt=Pestatt['Pestatt'][:]
```

[ ]:
```
# Egindako moving average horrek maiztasun altuko zikloak ezabatu ditu, orain␣
 ↪denborarekiko joera lineala kenduko da
# Mann-Kendall testa joerak lortzeko

Pmt_ta=np.zeros((12))
batura=0
for k in range(12):
    for i in range(40):
        batura=batura+Pmt[k+i*12]
    Pmt_ta[k]=batura/40
    batura=0
Pmt_anomaly=np.zeros((480))
for k in range(12):
    for i in range(40):
        Pmt_anomaly[k+12*i]=Pmt[k+12*i]-Pmt_ta[k]
Pmt_detrend=ss.detrend(Pmt_anomaly)# Anomalien joera kentzeko balio du.

Pestatt_ta=np.zeros((12))
batura=0
for k in range(12):
    for i in range(40):
        batura=batura+Pestatt[k+i*12]
    Pestatt_ta[k]=batura/40
    batura=0
Pestatt_anomaly=np.zeros((480))
for k in range(12):
    for i in range(40):
        Pestatt_anomaly[k+12*i]=Pestatt[k+12*i]-Pestatt_ta[k]
Pestatt_detrend=ss.detrend(Pestatt_anomaly)

Petranst_ta=np.zeros((12))
batura=0
for k in range(12):
```

```
    for i in range(39):
        batura=batura+Petranst[k+i*12]
    Petranst_ta[k]=batura/39
    batura=0
Petranst_anomaly=np.zeros((468))
for k in range(12):
    for i in range(39):
        Petranst_anomaly[k+12*i]=Petranst[k+12*i]-Petranst_ta[k]
Petranst_detrend=ss.detrend(Petranst_anomaly)
```

21 El Niño

[ ]:
```
#Denbora serieei kendu beharreko hurrengo zikloak ENSO zikloak dira, beraz,␣
 ↪azter dezagun lehenik El Niño.
ds=nc.Dataset('sst.nc')
sst=ds['sst'][:,85:96,190:241]
timesst=ds['time'][:]
lonsst=ds['longitude'][190:241]
latsst=ds['latitude'][85:96]

y1sst=len(sst[0,:,0])
z1sst=len(sst[0,0,:])
t1sst=len(sst[:,0,0])

ysst=range(y1sst)
zsst=range(z1sst)
tsst=range(t1sst)

landasst=lonsst[:]*2*math.pi/360
phisst=latsst[:]*2*math.pi/360
cossst=np.cos(phisst)
landa=lon[:]*2*math.pi/360
phi=lat[:]*2*math.pi/360
cos=np.cos(phi)
dlanda=landa[1]-landa[0]
dphi=phi[1]-phi[0]

#Niño 3.4 tenperatura
temp_pacif=temp[:,:,85:96,190:241]
lon_pacif=lonsst
lat_pacif=latsst
lev_pacif=lev

#Niño indize ofizialak:
from numpy import genfromtxt
my_data = genfromtxt('Niño_indices.csv', delimiter=',')
```

```
aux=0
indices=np.zeros((t1))
for i in range(len(my_data[:,0])):
    for j in range(len(my_data[0,:])):
        indices[aux]=my_data[i,j]
        aux=aux+1
```

[ ]:
```
#Nik kalkulatutako SST indizeak
sst1=np.mean(sst,axis=2)
sst2=np.zeros((t1sst,y1sst))
for i in ysst:
    sst2[:,i]=sst1[:,i]*cossst[i]
sst_mean=scipy.integrate.simps(sst2,phisst,axis=1)/(np.sin(phisst[y1sst-1])-np.
 ↪sin(phisst[0]))
sst_mean_ta=np.zeros((12))
batura=0
for k in range(12):
    for i in range(41):
        batura=batura+sst_mean[k+i*12]
    sst_mean_ta[k]=batura/41
    batura=0

sst_anomaly=np.zeros((t1))
for k in range(12):
    for i in range(41):
        sst_anomaly[k+12*i]=sst_mean[k+12*i]-sst_mean_ta[k] #Hilabete bakoitzari␣
 ↪hilabete horri dagokion batez bestekoa kendu.
#indizeak lortzeko anomaliak 5 hilabeteko moving averagekin pasa eta gero␣
 ↪desbiazio estandarra zatitu normalizatzeko.
sstanom=np.zeros((t1-2))
for i in t[1:491]:
    sstanom[i-1]=np.mean(sst_anomaly[i-1:i+1],axis=0)
```

[ ]:
```
fig, ax = plt.subplots(figsize=(50,10))
ax.plot(range(490),sstanom)
ax.plot(range(490),indices[1:491])
ax.plot(range(490),0.5*np.ones((490)))
ax.plot(range(490),-0.5*np.ones((490)))

ax.grid()
#plt.ca().invert_yaxis()
#plt.xticks(np.arange(0,492,24), np.arange(1979, 2020,2),fontsize=25)
plt.yticks(fontsize=25)
plt.ylabel(r'Anomaliak (ºC)', fontsize=45)
#plt.savefig('indices.png')
plt.show()
```

[ ]:
```
# Eguratseko tenperaturaren korrelazioak El Niño 3.4 eremuan

maxcorr=np.zeros((21)) #korrelazio maximoa gordetzeko presio bakoitzerako
lagg=np.zeros((21)) #atzerapena korrelazio maximoarentzat

for i in range(21):
    #tenperaturaren anomaliak presio bakoitzerako El Niño 3.4 eremuan:
    temp_pacif1=np.mean(temp_pacif[:,i,:,:],axis=2)
    temp_pacif2=np.zeros((t1sst,y1sst))
    for k in range(y1sst):
        temp_pacif2[:,k]=temp_pacif1[:,k]*cossst[k]
    temp_pacif_mean=scipy.integrate.simps(temp_pacif2,phisst,axis=1)/(np.
 ↪sin(phisst[y1sst-1])-np.sin(phisst[0]))
    temp_pacif_mean_ta=np.zeros((12))
    batura=0
    for k in range(12):
        for j in range(41):
            batura=batura+temp_pacif_mean[k+j*12]
        temp_pacif_mean_ta[k]=batura/41
        batura=0

    temp_pacif_anomaly=np.zeros((t1))
    for k in range(12):
        for j in range(41):
            ␣
 ↪temp_pacif_anomaly[k+12*j]=temp_pacif_mean[k+12*j]-temp_pacif_mean_ta[k]


    #sst-rekin bezala, 3 hilabeteko moving average
    decompose_result_temp_pacif = seasonal_decompose(temp_pacif_anomaly,period=3)
    trend = decompose_result_temp_pacif.trend

    #sst eta tenperaturaren anomaliei denborako joera kenduko zaie elkarren␣
 ↪arteko korrelazioan eraginik izan ez dezan
    temp_pacif_anomaly_detrend=ss.detrend(trend[1:491])
    sstanomdetrend=ss.detrend(sstanom)

    correlation = scipy.signal.correlate(temp_pacif_anomaly_detrend,␣
 ↪sstanomdetrend[:], mode="full")
    lags = scipy.signal.correlation_lags(temp_pacif_anomaly_detrend.size,␣
 ↪sstanomdetrend[:].size, mode="full")
    lag = lags[np.argmax(correlation)] #Atzerapena

    #Balio ez esanguratsuak baztertzeko
    if lag<0:
        maxcorr[i]=1000
        lagg[i]=1000
```

```
        continue

    maxcorr[i]=scipy.stats.pearsonr(temp_pacif_anomaly_detrend[lag:],np.
 ↪roll(sstanomdetrend,lag)[lag:])[0] #Pearsonen korrelazioa atzerapen␣
 ↪horretarako
    lagg[i]=lag
```

[ ]:
```
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(lagg[4:],lev[4:]) # Balio ez esanguratsuak baztertuz

ax.grid()
plt.gca().invert_yaxis()
plt.yticks(fontsize=20)
plt.xticks(fontsize=20)
plt.ylabel(r'Presioa (hPa)', fontsize=30)
plt.xlabel(r'Atzerapena (hilabete)', fontsize=30)
#plt.savefig('lagg.png')
plt.show()
```

[ ]:
```
#700 hPa-etan gertatzen dena ikusiko da, ideia bat egiteko

temp_pacif1=np.mean(temp_pacif[:,20,:,:],axis=2)
temp_pacif2=np.zeros((t1sst,y1sst))
for i in range(y1sst):
    temp_pacif2[:,i]=temp_pacif1[:,i]*cossst[i]
temp_pacif_mean=scipy.integrate.simps(temp_pacif2,phisst,axis=1)/(np.
 ↪sin(phisst[y1sst-1])-np.sin(phisst[0]))
temp_pacif_mean_ta=np.zeros((12))
batura=0
for k in range(12):
    for i in range(41):
        batura=batura+temp_pacif_mean[k+i*12]
    temp_pacif_mean_ta[k]=batura/41
    batura=0

temp_pacif_anomaly=np.zeros((t1))
for k in range(12):
    for i in range(41):
        temp_pacif_anomaly[k+12*i]=temp_pacif_mean[k+12*i]-temp_pacif_mean_ta[k]

#sst-rekin bezala, 3 hilabeteko moving average
decompose_result_temp_pacif = seasonal_decompose(temp_pacif_anomaly,period=3)
trend = decompose_result_temp_pacif.trend

#sst eta tenperaturaren anomaliei denborako joera kenduko zaie elkarren arteko␣
 ↪korrelazioan eraginik izan ez dezan
temp_pacif_anomaly_detrend=ss.detrend(trend[1:491])
```

```
sstanomdetrend=ss.detrend(sstanom)

fig, ax = plt.subplots(figsize=(50,10))
ax.plot(range(490),temp_pacif_anomaly_detrend)
ax.plot(range(490),sstanomdetrend)

ax.grid()
#plt.ca().invert_yaxis()
plt.xticks(np.arange(11,491,24), np.arange(1980, 2020,2),fontsize=25)
plt.yticks(fontsize=25)
plt.ylabel(r'Anomaliak (ºC)', fontsize=45)
#plt.savefig('1000hPa.png')
plt.show()
```

[ ]:
```
# energien korrelazioak
#Pmt_detrend EPEZi dagokio
#Pmt_detrend=Pestatt_detrend+Petranst_detrend jarri EEPE aztertzeko eta␣
 ↪sstanomdetrend-en mugak aldatu [11:479]ra


correlation = scipy.signal.correlate(Pmt_detrend[:], sstanomdetrend[5:485],␣
 ↪mode="full")
lags = scipy.signal.correlation_lags(Pmt_detrend[:].size, sstanomdetrend[5:485].
 ↪size, mode="full")
lag = lags[np.argmax(correlation)]
maxcorr1=scipy.stats.pearsonr(Pmt_detrend[lag:],np.roll(sstanomdetrend[5:
 ↪485],lag)[lag:])[0]
p1=scipy.stats.pearsonr(Pmt_detrend[:],np.roll(sstanomdetrend[5:485],lag))[1]
lag1=lag
correlation[np.argmax(correlation)]=0
lag2=lags[np.argmax(correlation)]
maxcorr2=scipy.stats.pearsonr(Pmt_detrend[lag2:],np.roll(sstanomdetrend[5:
 ↪485],lag2)[lag2:])[0]
p2=scipy.stats.pearsonr(Pmt_detrend[:],np.roll(sstanomdetrend[5:485],lag2))[1]
correlation[np.argmax(correlation)]=0
lag3=lags[np.argmax(correlation)]
maxcorr3=scipy.stats.pearsonr(Pmt_detrend[lag3:],np.roll(sstanomdetrend[5:
 ↪485],lag3)[lag3:])[0]
p3=scipy.stats.pearsonr(Pmt_detrend[:],np.roll(sstanomdetrend[5:485],lag3))[1]
```

[ ]:
```
#sst anomalien eta energien artean pearson korrelazioa aplikatu daitekeen edo ez␣
 ↪ikusteko

plt.figure(figsize=(10, 5))
matplotlib.pyplot.scatter(sstanomdetrend[11:479],Pmt_detrend, s=8)
plt.ylabel('EEPE anomaliak denborarekiko joera gabe ($J/m^2$)',fontsize=10)
```

```
plt.xlabel('SST anomaliak denborarekiko joera gabe (ºC)',fontsize=10)
plt.savefig('lagepe1.png')
```

[ ]:
```
#Niño urtetako energien eta beste urteen arteko ezberdintasunak.

#Niño maximoak hartu eta inguruko 12 hilabeteak erabili batez besetekuantzat
#bost maximoak, kontutan izan pmt eta sstanom-ren hasierak ezberdinak direla
print(24+np.argmax(sstanom[24:72]))
print(84+np.argmax(sstanom[84:120]))
print(120+np.argmax(sstanom[120:200]))
print(84+120+np.argmax(sstanom[84+120:240]))
print(350+np.argmax(sstanom[350:490]))
```

[ ]:
```
Pmt_niño=(np.mean(Pmt[35:47])+np.mean(Pmt[91:103])+np.mean(Pmt[144:156])+np.
 ↪mean(Pmt[214:226])+np.mean(Pmt[430:442]))/5
Pmt_normal=(np.sum(Pmt)-(np.sum(Pmt[35:47])+np.sum(Pmt[91:103])+np.sum(Pmt[144:
 ↪156])+np.sum(Pmt[214:226])+np.sum(Pmt[430:442])))/(480-12*5)
(Pmt_niño-Pmt_normal)/Pmt_normal*100
```

[ ]:
```
Pet_niño=(np.mean((Pestatt[6:474]+Petranst[:])[29:41])+np.mean((Pestatt[6:
 ↪474]+Petranst[:])[85:97])+np.mean((Pestatt[6:474]+Petranst[:])[138:150])+np.
 ↪mean((Pestatt[6:474]+Petranst[:])[208:220])+np.mean((Pestatt[6:474]+Petranst[:
 ↪])[424:436]))/5
Pet_normal=(np.sum((Pestatt[6:474]+Petranst[:]))-(np.sum((Pestatt[6:
 ↪474]+Petranst[:])[29:41])+np.sum((Pestatt[6:474]+Petranst[:])[85:97])+np.
 ↪sum((Pestatt[6:474]+Petranst[:])[138:150])+np.sum((Pestatt[6:474]+Petranst[:
 ↪])[208:220])+np.sum((Pestatt[6:474]+Petranst[:])[424:436])))/(468-12*5)
(Pet_niño-Pet_normal)/Pet_normal*100
```

[ ]:
```
#Niño garaian eta garai normaletan energien banaketen ezberdintasunak

ds=nc.Dataset('Pm_JperKg_Boer_denboran.nc')
Pmt=ds['Pm_JperKg']
Pmt=Pmt['Pm_JperKg'][:]

ds=nc.Dataset('Pestat_JperKg_Boer_denboran.nc')
Pestatt=ds['PestatJperKgbetagabeta']
Pestatt=Pestatt['PmestatJperKgbetagabeta'][:]

ds=nc.Dataset('Petrans_JperKg_Boer_denboran.nc')
Petranst=ds['PestatJperKgbetagabeta']
Petranst=Petranst['PmestatJperKgbetagabeta'][:]

ds=nc.Dataset('Pe_statJperKg_longlat_Boer_denboran.nc')
Pestattlonlat=ds['PestatJperKglonglatbetagabeta']
Pestattlonlat=Pestattlonlat['PmestatJperKglonglatbetagabeta'][:]
```

```
ds=nc.Dataset('Pe_transJperKg_longlat_Boer_denboran.nc')
Petranstlonlat=ds['PestatJperKglonglatbetagabeta']
Petranstlonlat=Petranstlonlat['PmestatJperKglonglatbetagabeta'][:]

Pmt_niño=(np.mean(Pmt[35:47,:],0)+np.mean(Pmt[91:103,:],0)+np.mean(Pmt[144:156,:
 ↪],0)+np.mean(Pmt[214:226,:],0)+np.mean(Pmt[430:442,:],0))/5
Pmt_normal=(np.sum(Pmt[:],0)-(np.sum(Pmt[35:47,:],0)+np.sum(Pmt[91:103,:],0)+np.
 ↪sum(Pmt[144:156,:],0)+np.sum(Pmt[214:226,:],0)+np.sum(Pmt[430:442,:],0)))/
 ↪(480-12*5)
emaitza=Pmt_niño-Pmt_normal
```

22 Denbora serieak ENSO kenduta

[ ]:
```
ds=nc.Dataset('Pmt.nc')
Pmt=ds['Pmt']
Pmt=Pmt['Pmt'][:]
ds=nc.Dataset('Petranst.nc')
Petranst=ds['Petranst']
Petranst=Petranst['Petranst'][:]
ds=nc.Dataset('Pestatt.nc')
Pestatt=ds['Pestatt']
Pestatt=Pestatt['Pestatt'][:]
ds=nc.Dataset('PmtIH.nc')
PmtIH=ds['Pmt']
PmtIH=PmtIH['Pmt'][:]
ds=nc.Dataset('PmtHH.nc')
PmtHH=ds['Pmt']
PmtHH=PmtHH['Pmt'][:]
ds=nc.Dataset('PestattIH.nc')
PestattIH=ds['Pmt']
PestattIH=PestattIH['Pmt'][:]
ds=nc.Dataset('PestattHH.nc')
PestattHH=ds['Pmt']
PestattHH=PestattHH['Pmt'][:]
ds=nc.Dataset('PetranstIH.nc')
PetranstIH=ds['Pmt']
PetranstIH=PetranstIH['Pmt'][:]
ds=nc.Dataset('PetranstHH.nc')
PetranstHH=ds['Pmt']
PetranstHH=PetranstHH['Pmt'][:]
```

[ ]:
```
#Pmt bezala goiko edozein definitu denbora muga egokiekin
Pmt=Petranst
Pmt_mean_ta=np.zeros((12))
batura=0
for k in range(12):
```

```
    for i in range(39):
        batura=batura+Pmt[k+i*12]
    Pmt_mean_ta[k]=batura/39
    batura=0

Pmt_anomaly=np.zeros((468))
for k in range(12):
    for i in range(39):
        Pmt_anomaly[k+12*i]=Pmt[k+12*i]-Pmt_mean_ta[k]

slope, intercept, r_value, p_value, std_err = stats.linregress(t[12:
 ↪480],Pmt_anomaly)
Pmt_joerakenduta=Pmt_anomaly[:]-slope*t[12:480]#karratu minimoekin lortutako␣
 ↪joeraren malda kendu joera ezabatzeko
slope1=slope

slope, intercept, r_value, p_value, std_err = stats.linregress(t[:],sst_anomaly[:
 ↪])
sst_joerakenduta=sst_anomaly[:]-slope*t[:]#sst anomaliei joera kendu

slope, intercept, r_value, p_value, std_err = stats.
 ↪linregress(sst_joerakenduta[9:477],Pmt_joerakenduta[:])
Pmt_ensokenduta=Pmt_joerakenduta-slope*sst_joerakenduta[9:477]#sst-rekiko␣
 ↪erlazio lineala kendu
Pmt_ensokendutajoerakin=Pmt_ensokenduta+slope1*t[12:480]
Pmt_ensogabehh=np.zeros((468))
for k in range(12):
    for i in range(39):
        Pmt_ensogabehh[12*i+k]=Pmt_ensokendutajoerakin[12*i+k]+Pmt_mean_ta[k]
slopehh, intercepthh, r_value, p_value, std_err = stats.linregress(t[12:
 ↪480],Pmt_ensogabehh)
```

[ ]:
```
#autokorrelazio funtzioa
fig = tsaplots.plot_acf(ss.detrend(Pmt), lags=48, alpha=0.05)
plt.ylabel('Korrelazio koefizientea',fontsize=10)
plt.xlabel('Atzerapenak',fontsize=10)
plt.show()
plt.savefig('epezautokor.png')
```