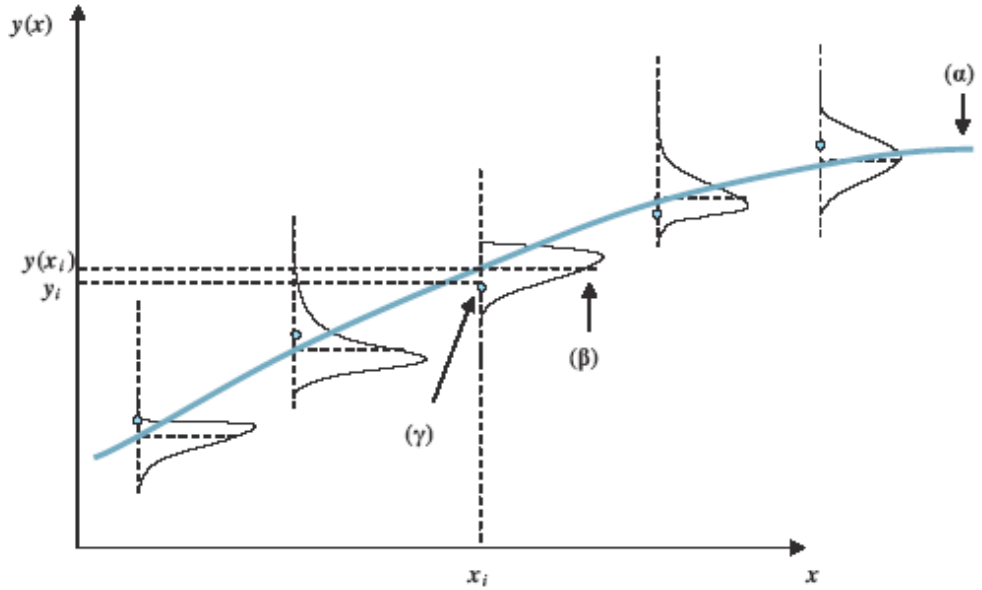
Μέθοδοι μέγιστης πιθανοφάνειας

Οι πιθανοκρατικές μέθοδοι βασίζονται στην παρατήρηση ότι η ελαχιστοποίηση του χ2 αντιστοιχεί σε μεγιστοποίηση πιθανότητας.

Έστω ότι φιτάρουμε σετ δεδομένων, για **το καθένα** από τα οποία η μέση τιμή αλλά και η διακύμανση γύρω από αυτή ακολουθούν **κανονική κατανομή**. Αυτό μπορεί να συμβαίνει πχ, γιατί τα σημεία προέκυψαν μετά από πολλές μετρήσεις για το καθένα (βλ. θεώρημα κεντρικού ορίου).



* See Data Analysis – Siegmund Brandt,
Data analysis recipes: fitting a model to data: Hogg, Bovy, Lang 2010

Μέθοδοι μέγιστης πιθανοφάνειας

Οι πιθανοκρατικές μέθοδοι βασίζονται στην παρατήρηση ότι η ελαχιστοποίηση του χ2 αντιστοιχεί σε μεγιστοποίηση πιθανότητας.

$$P(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(y_i - f(x_i))^2}{2\sigma_i^2}}$$

Conditional probability: πιθανότητα να βρεθεί το y_i δεδομένου του x_i ∀ σημείο και

$$\mathcal{L} \equiv \text{likelihood} = \prod_{i=1}^{N} P(y_i|x_i)$$ για όλο το dataset. Τότε

$$\ln \mathcal{L} = \ln \left[\left(\frac{1}{\sqrt{2\pi}} \right)^N \prod_{i=1}^{N} \frac{1}{\sigma_i} e^{-\frac{(y_i - f(x_i))^2}{2\sigma_i^2}} \right]$$

$$= N \ln \frac{1}{\sqrt{2\pi}} - \ln \prod_{i=1}^{N} \sigma_i + \ln e^{-\sum_{i=1}^{N} \frac{(y_i - f(x_i))^2}{2\sigma_i^2}}$$

$$= C - \sum_{i=1}^{N} \frac{(y_i - f(x_i))^2}{2\sigma^2} = C - \frac{1}{2}\chi^2$$

$\Rightarrow$ likelyhood maximation = χ^2 minimization

Μέθοδοι μέγιστης πιθανοφάνειας

Βασικά πλεονεκτήματα των πιθανοκρατικών μεθόδων σε σχέση με το απλό χ^2

1. Η δυνατότητα χρήσης «prior knowledge» ή «updated knowledge» καθώς εξελίσσεται το πείραμα. Δηλαδή, μπορούμε να δίνουμε **βάρη** (όχι μόνο όρια) βάσει γνωστών ήδη πληροφοριών ώστε να κατευθύνουμε τον αλγόριθμο προς τη σωστή κατεύθυνση.

   Αν δεν γίνει χρήση πρωτύτερης γνώσης, το αποτέλεσμα ίδιο με χ^2

2. Χάρη στις πιθανότητες, μπορούμε όχι μόνο να δούμε ποια είναι η δεύτερη (και τρίτη...) κλπ καλύτερη λύση αλλά και πόσο πιθανό είναι αν αλλάξουμε μια μεταβλητή με καποιο τρόπο ώστε να βρούμε το σωστό αποτέλεσμα.

   Αντιθέτως, οι αλγόριθμοι χ^2 εγγενώς επιστρέφουν μόνο την καλύτερη λύση.

3. Μπορούμε να συγκρίνουμε ακόμα και 2 μοντέλα μεταξύ τους

   → **Bayesian statistics**

' Bayes describes the conditional probability of an event based on data as well as prior information or beliefs about the event or conditions related to the event (1763). Laplace developed the Bayesian interpretation of probability '

Στο πιθανοτικό framework, θέλουμε να μεγιστοποιήσουμε

το $\prod_{i=1}^{N} P(\alpha, \beta \mid y_i)$, όπου α και β οι παράμετροι του μοντέλου,

δηλ την πιθανότητα των α και β δεδομένων των μετρήσεων y_i:

$$\prod_{i=1}^{N} P(\alpha,\beta \mid y_i) = \prod_{i=1}^{N} \frac{P(y_i \mid \alpha,\beta) \cdot P(\alpha,\beta)}{P(y_i)}$$

από θεώρημα Bayes. Γενική μορφή για μοντέλο M, data D:

Likelihood: πιθανότητα συμβατότητας D με κάθε M

prior probability: πιθανότητα μοντέλου, συμπεριλαμβάνει προϋπάρχουσα γνώση

$$P(M \mid D) = \frac{P(D \mid M) \cdot P(M)}{P(D)}$$

posterior probability: πιθανότητα μοντέλου για τις δεδομένες μετρήσεις

Evidence: πιθανότητα μετρήσεων ανεξάρτητη μοντέλου ↔ κοινός παράγοντας κανονικοποίησης (παραλείπεται)

Απόδειξη θεωρήματος για δύο γεγονότα E_1, E_2:

Ορισμός conditional probability*: $P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}$

και $P(E_2|E_1) = \frac{P(E_2 \cap E_1)}{P(E_1)}$

Εφόσον η joint probability

$P(E_1 \cap E_2) = P(E_2 \cap E_1)$ τότε προκύπτει

$P(E_1|E_2) = \frac{P(E_2|E_1) \cdot P(E_1)}{P(E_2)}$

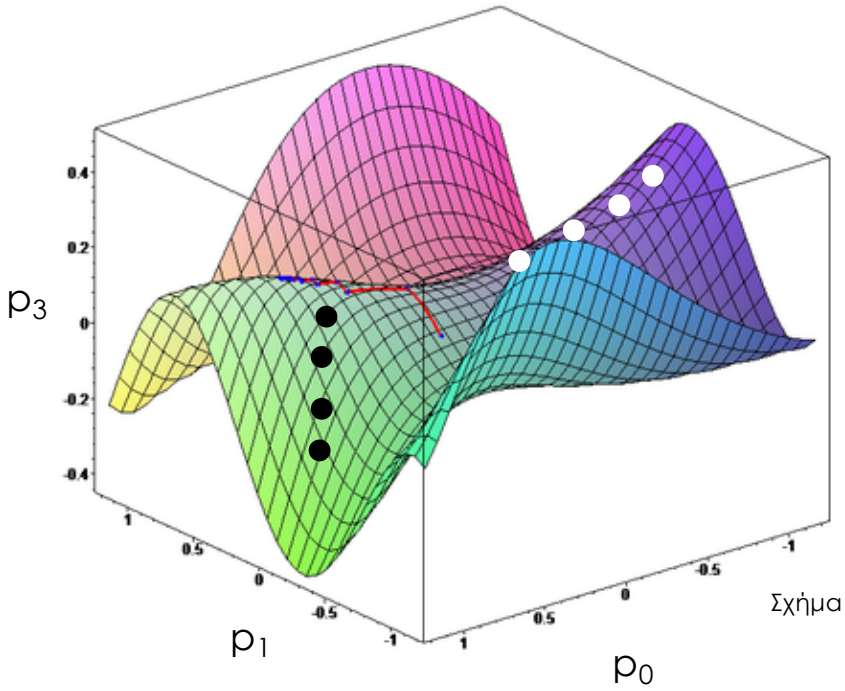
* ποσοστό κοινής πιθανότητας που αναστοιχεί σε ένα γεγονός

Έχοντας τις prior, evidence, και μπορώντας να υπολογίσουμε την likelihood για κάθε μοντέλο, βρίσκουμε την **posterior κι άρα τις βέλτιστες τιμές των παραμέτρων.**

Πακέτο emcee python. Συνδυάζει

- **Θεώρηση Bayesian** (μέσω ελαχιστοποίησης lnP)
- **Monte Carlo** για επιλογή και μεταβολή σημείων στο χώρο παραμέτρων σε κάθε χρόνο, με αλγόριθμο **τύπου Metropolis** για την επιλογή εύρους βήματος.
- Χρήση **Ν αριθμού περιπατητών (walkers)** που ακολουθούν τυχαίας κατεύθυνσης που δείχνουν την κλίση του χώρου
- **αλυσίδα βημάτων** που εξαρτάται μόνο από την τρέχουσα θέση σημείων – όχι τη παρελθοντική. Αλλά δεδομένου ότι τα βήματα σώζονται, η αλυσίδα μπορεί να ακολουθηθεί κι ανάποδα - Markov chain

Π.χ., κίνηση 2 walkers, 4 βημάτων, σε χώρο παραμέτρων



Σχήμα wikpedia

Metropolis (- Hastings) algorithm :

- Θέλουμε να βρούμε σε δείγμα επιδυτικής κατανομής πιθανότητας P με διάφορη PDF $f \sim P$

- Διαλέγουμε συνάρτηση βήματος $g(x)$, επίσης κανονικοποιημένη σαν PDF - flat, random, Gaussian για ψεσανίμηση εντός αυτών ...

- for $i = 1, \ldots,$ number_of_steps :

  α) βρίσκουμε νέο δείγμα x_i ως $g(x_{i-1})$

  β) ελέγχουμε αν $a = \frac{f(x_i)}{f(x_{i-1})}$

  γ) παράγουμε τυχαίο αριθμό $u \in [0,1]$ - από uniform

  δ) Αν $u < a$, κρατάμε νέα δέση. (accept / reject)

Αν $\frac{f(x_i)}{f(x_{i-1})} > 1$, πάντα $u < a \Rightarrow$ λύσεις προς τη σωστή κατεύθυνση επιλέγονται πάντα

Αν $a < 1$ και $u < a$: πάλι κρατάμε τη λύση, ώστε λύσεις προς άλλες κατευθύνσεις να επιτρέπονται τυχαία κι η επίλυση να μην "κολλάει" σε τοπικές λύσεις.

Αν $u > a$: $x_i = x_{i-1}$ (δεν προχωράει σε νέα δέση)

Fit **ευθείας γραμμής με Bayesian** + Monte Carlo (emcee)

Likelihood:

$$\mathcal{L}(y, y_{model}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - (ax_i + b))^2}{2\sigma^2}} = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\sum_{i=1}^{N} \frac{(y_i - (ax_i + b))^2}{2\sigma^2}}$$

Priors:

για να καταλάβουμε την ισχύ τους πρώτα fit χωρίς priors και μετά fit προς τη σωστή κατεύθυνση λόγω αβεβαιοτήτων:

$$P_r(a, b) = P_r(a) \cdot P_r(b) \overset{\text{n.x.}}{=} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(a-a_0)^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma_m^2}} e^{-\frac{(b-b_0)^2}{2\sigma_m^2}}$$

Το τελευταίο βήμα είναι μια παραδοχή γιατί στην προκειμένη δεν έχει κάποιο φυσικό νόημα. Σε άλλες όμως περιπτώσεις είναι μια συνάρτηση που περιγράφει το φυσικό φαινόμενο (π.χ., συνάρτηση μάζας γαλαξιών σε πρόβλημα μέτρησης 1 μάζας από ακτινοβολική εκπομπή).

Χτίζουμε τις πιθανότητες

```python
# emcee requires log-posterior

def log_prior(params):
    alpha, beta=params
    sigma_dev=0.1
    return np.log( 1/2*np.pi*sigma_dev**2) - (alpha-1.53)**2/(2*sigma_dev**2) - (beta-4.86)**2/(2*sigma_dev**2)

# gaussian likelihood for each data point given a model, sum over all data points
def log_likelihood(params, x, y):
    alpha, beta = params
    sigma_typical = 1.5
    y_model = alpha* x + beta
    N=len(x)
    # note that the result should be the same with or without the second term of the sum = constant
    return -0.5 * np.sum((y-y_model)**2 / sigma_typical**2)  - (N/2)*np.log(2*np.pi* sigma_typical** 2)

def log_posterior(params, x, y):
    #return log_likelihood(params, x, y)   # compare to χ2 without errors
    return log_prior(params) + log_likelihood(params, x, y)  # compare to χ2 with errors
```

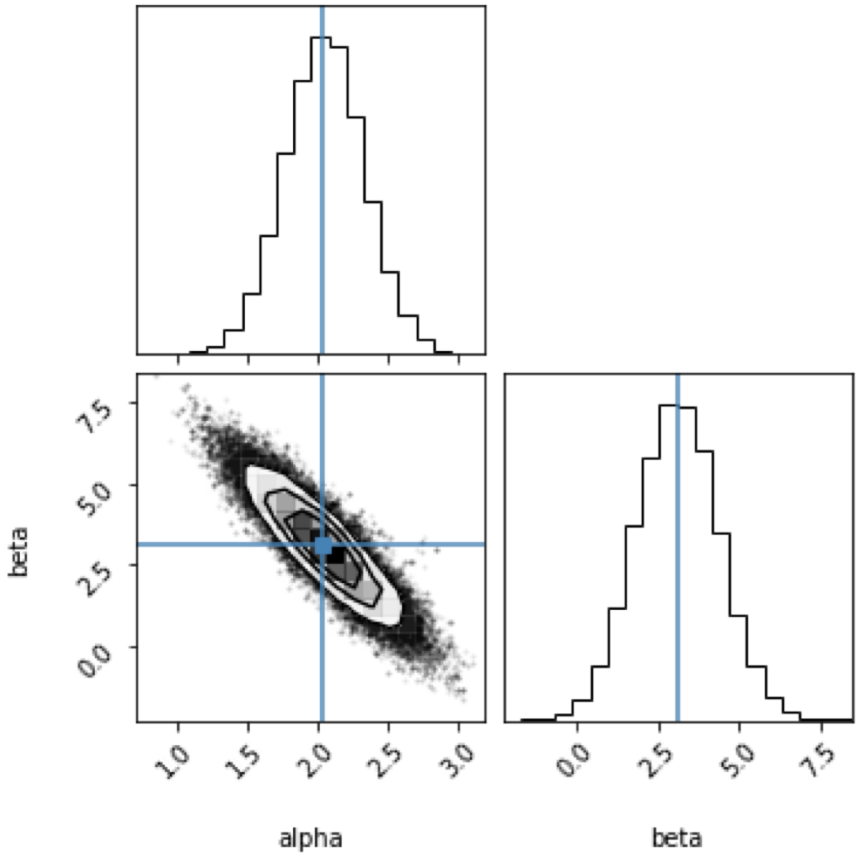
Εκτελούμε τον περίπατο στο χώρο των παραμέτρων

```
# emcee combines MCMC chains (walkers) of a number of steps.

ndim = 2  # number of parameters in the model
nwalkers = 50  # number of MCMC walkers
nburn = 20 # "burn-in" period to stabilise chains
nsteps = 2000  # number of MCMC steps to take


starting_guesses = np.random.random((nwalkers, ndim))*6 # random positioning of 50 walkers in parameter space
#print(starting_guesses)

sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior, args=[x_data, y_data])
sampler.run_mcmc(starting_guesses, nsteps)

#Αφαιρώ τα πρωτα nburn βήματα από καθε walker
#επίσης μπορώ τα βήματα των walkers μαζί για να χαμηλώσω μια διάσταση τον πίνακα
#emcee_trace =  sampler.chain[:, nburn:, :].reshape(-1, ndim)
emcee_trace = sampler.get_chain(discard=nburn,flat=True)
```

```
labels = ['alpha','beta']

#previous_solution=[1.53,4.86] # with errors
previous_solution=[2.04,3.10] # without errors


fig = corner.corner(emcee_trace, labels=labels
    ,truths=[previous_solution[0], previous_solution[1]]);

print( '\n Mean alpha bayesian =  ', emcee_trace[:,0].mean(),
     '\n χ^2 alpha  = ', previous_solution[0],
     '\n Mean beta bayesian =  ', emcee_trace[:,1].mean(),
     '\n χ^2 beta = ', previous_solution[1])
```

Χωρίς βάρη στην prior

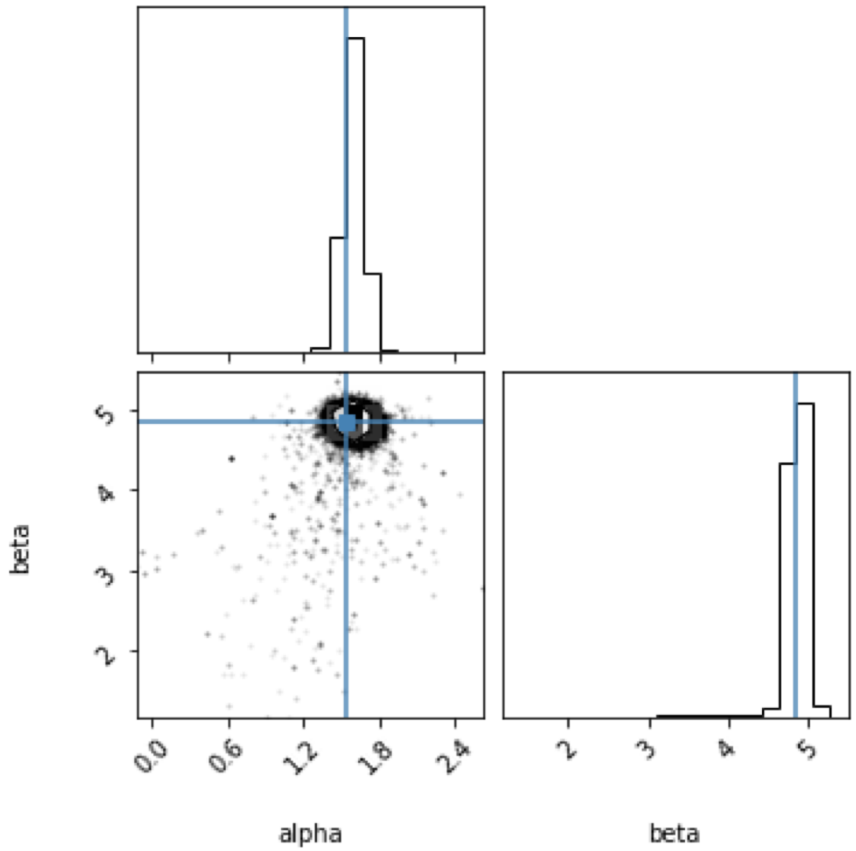
```
Mean alpha bayesian =   2.0436413093307277
χ^2 alpha  =  2.04
Mean beta bayesian =   3.0778930524593844
χ^2 beta =  3.1
```



```
labels = ['alpha','beta']

previous_solution=[1.53,4.86] # with errors
#previous_solution=[2.04,3.10] # without errors


fig = corner.corner(emcee_trace, labels=labels
    ,truths=[previous_solution[0], previous_solution[1]]);

print( '\n Mean alpha bayesian =  ', emcee_trace[:,0].mean(),
     '\n χ^2 alpha  = ', previous_solution[0],
     '\n Mean beta bayesian =  ', emcee_trace[:,1].mean(),
     '\n χ^2 beta = ', previous_solution[1])
```

Με βάρη στην prior

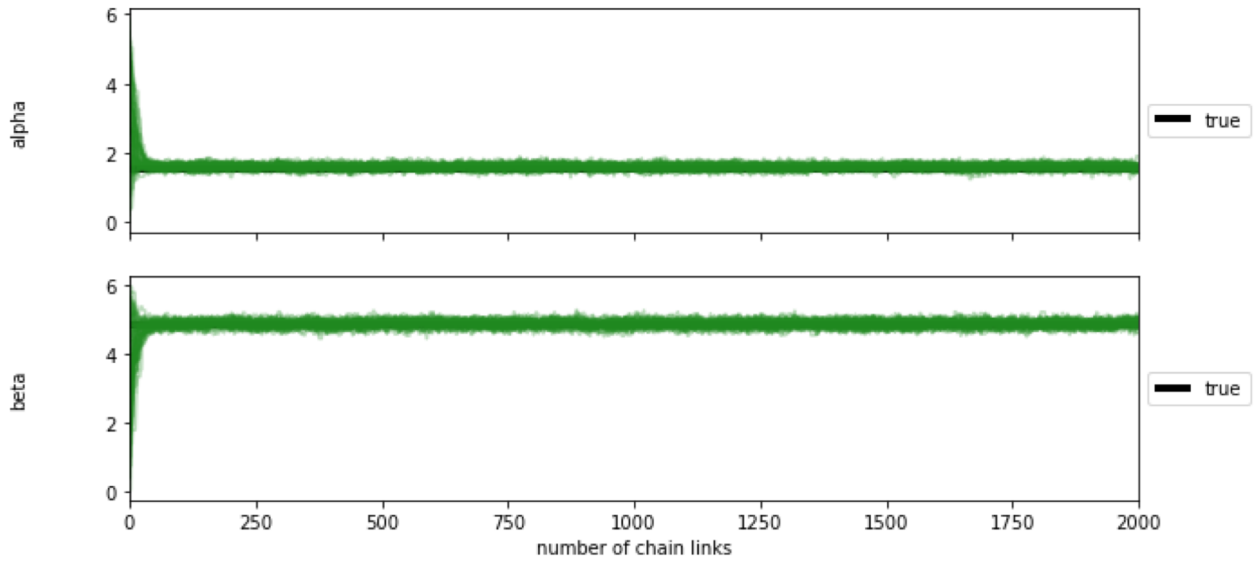
```
Mean alpha bayesian =   1.588738518764105
χ^2 alpha  =  1.53
Mean beta bayesian =   4.851960032274037
χ^2 beta =  4.86
```



Πολύ πιο στενή Κατανομή, όπως αναμενόταν

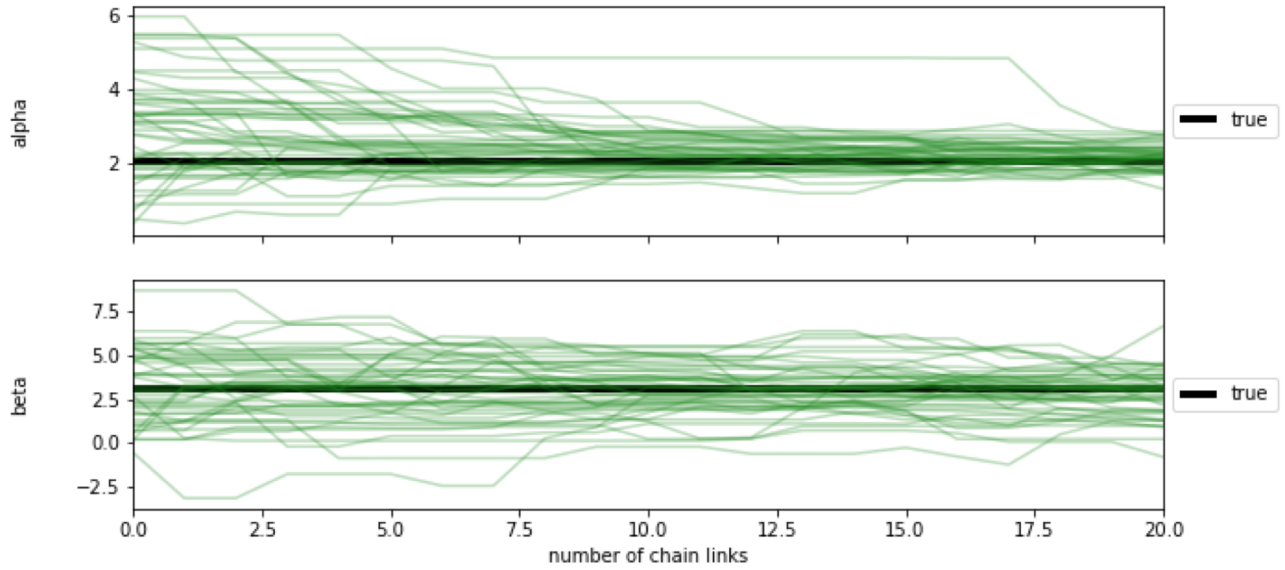
Έλεγχος πως κινούνται οι walkers στο χώρο των παραμέτρων (όλα τα βήματα)

```
fig, axes = plt.subplots(2, figsize=(10, 5), sharex=True)
samples = sampler.get_chain()
labels = [ 'alpha', 'beta']
for i in range(ndim):
    ax = axes[i]
    #ax.plot(samples[:,0, i], 'forestgreen',label='walk') #αν θελω μόνο ένα walker
    ax.plot(samples[:,:, i], 'forestgreen', alpha=0.3)    #βλεπω όλους τους walkers
    ax.set_xlim(0, len(samples))
    #ax.set_xlim(0, nburn)
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)
    ax.hlines(y=previous_solution[i], xmin = 0, xmax = len(samples), linewidth=4, color='k',label='true')
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

axes[-1].set_xlabel('number of chain links');
```



Έλεγχος πως κινούνται οι walkers στο χώρο των παραμέτρων (πρώτα βήματα, including burns, δηλ. πριν τη σύγκλιση)

```
fig, axes = plt.subplots(2, figsize=(10, 5), sharex=True)
samples = sampler.get_chain()
labels = [ 'alpha', 'beta']
for i in range(ndim):
    ax = axes[i]
    #ax.plot(samples[:,0, i], 'forestgreen',label='walk') #αν θελω μόνο ένα walker
    ax.plot(samples[:,:, i], 'forestgreen', alpha=0.3)    #βλεπω όλους τους walkers
    #ax.set_xlim(0, len(samples))
    ax.set_xlim(0, nburn)
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)
    ax.hlines(y=previous_solution[i], xmin = 0, xmax = len(samples), linewidth=4, color='k',label='true')
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

axes[-1].set_xlabel('number of chain links');
```



Έλεγχος πως κινούνται οι walkers στο χώρο των παραμέτρων (πρώτα βήματα, ξεκινώντας από α=1)

```
fig, axes = plt.subplots(2, figsize=(10, 5), sharex=True)
samples = sampler.get_chain()
labels = [ 'alpha', 'beta']
for i in range(ndim):
    ax = axes[i]
    #ax.plot(samples[:,0, i], 'forestgreen',label='walk') #αν θελω μόνο ένα walker
    ax.plot(samples[:,:, i], 'forestgreen', alpha=0.3)    #βλεπω όλους τους walkers
    #ax.set_xlim(0, len(samples))
    ax.set_xlim(0, nburn)
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)
    ax.hlines(y=previous_solution[i], xmin = 0, xmax = len(samples), linewidth=4, color='k',label='true')
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

axes[-1].set_xlabel('number of chain links');
```