Project 1

Markov chain Monte Carlo

This notebook is arranged in cells. Texts are usually written in the markdown cells, and here you can use html tags (make it bold, italic, colored, etc). You can double click on this cell to see the formatting.

The ellipsis (...) are provided where you are expected to write your solution but feel free to change the template (not over much) in case this style is not to your taste.

Hit "Shift-Enter" on a code cell to evaluate it. Double click a Markdown cell to edit.

Write your partner's name here (if you have one).

Link Okpy

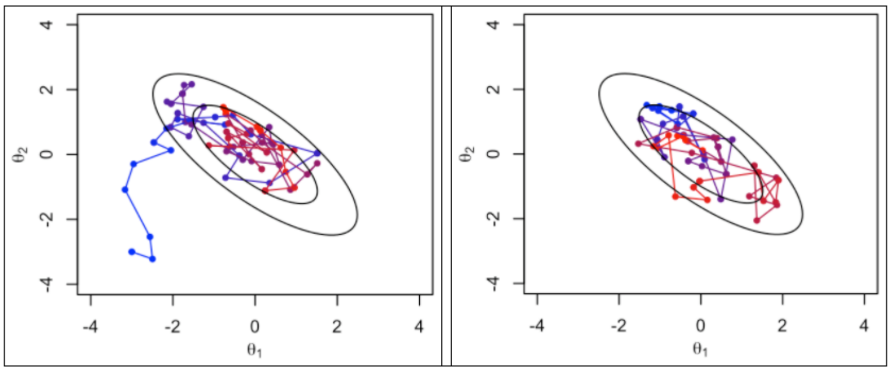
```
In [ ]:
from client.api.notebook import Notebook
ok = Notebook('project1_p3_U.ok')
_ = ok.auth(inline = True)
```

Imports

```
In [ ]:
import numpy as np
from scipy.integrate import quad
#For plotting
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1 - Planck MCMC chain

Markov chain Monte Carlo is a general method based on drawing values of θ from approximate distributions and then correcting those draws to better aproximate the target posterior distribution. The sampling is done sequentially, wtih the distribution of the sampled draws depending on the last value drawn - hence, the draws from a Markov chain. (p. 275, Bayesian Data Analysis, Andrew Gelman et al.) (Remember that a sequence $x_1, x_2, \ldots$ of random events is called a Markov chain if x_{n+1} depends explicitly on x_n only (and not explicitly on previous steps).) Here, we consider six selected cosmologial parameters: $[H_0, \Omega_b h^2, \Omega_c h^2, n_s, A_s, \tau]$, so the "chain" in this case is a random walk through the parameter space.



from https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb (https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb)

As shown in the above figure, chains take time to converge to the target distribution, and you can determine the "burn-in" period, the number of sequences it takes to reach convergence.

In this problem, we provide you MCMC chains (using Planck low and high-l temperature data with lensing reconstruction) from Planck Data Release 1 (http://irsa.ipac.caltech.edu/data/Planck/release_1/ancillary-data/ (http://irsa.ipac.caltech.edu/data/Planck/release_1/ancillary-data/)). You can plot the chains in the parameter space and estimate the posterior distribution.

References:
Bayesian Data Analysis, Andrew Gelman et al.
https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb (https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb)

1. First, we give you one Planck chain without removing the burn-in. In this case, the parameter space is $(H_0, \Omega_b h^2)$. Estimate the burn-in period.

```
In [ ]:
# Load data
data = np.loadtxt("Planck_chain_with_burnin.txt")
# H0
theta1 = data[:,23]
# Omega_b h^2
theta2 = data[:,2]

# Plot chain
plt.plot(theta1, theta2, 'x-')
plt.xlabel('$H_0$')
plt.ylabel('$\Omega_b h^2$')
plt.show()
```

Answer:

2. Now, we provide you with 8 independent Planck MCMC chains. For each chain, we load the data for six cosmological parameters we are considering, $[H_0, \Omega_b h^2, \Omega_c h^2, n_s, A_s, \tau]$. From the chain, estimate the posterior distribution of each of the six parameters. (Plot the 1-d posterior distribution and estimate its mean + standard deviation.)

In [ ]:

```python
# Load data

# H_0
theta1_chain = np.zeros(shape=(8,1981))
# Omega_b h^2
theta2_chain = np.zeros(shape=(8,1981))
# Omega_c h^2
theta3_chain = np.zeros(shape=(8,1981))
# n_s
theta4_chain = np.zeros(shape=(8,1981))
# A_s
theta5_chain = np.zeros(shape=(8,1981))
# tau
theta6_chain = np.zeros(shape=(8,1981))

# 8 Planck chains, each of length 1981 (so theta6_chain[1] contains values of tau in Planck chain 1)
for i in range(8):
    data = np.loadtxt("base_planck_lowl_post_lensing_%d.txt" %(i+1))
    theta1_chain[i] = data[:,27][0:1981]
    theta2_chain[i] = data[:,2][0:1981]
    theta3_chain[i] = data[:,3][0:1981]
    theta4_chain[i] = data[:,6][0:1981]
    theta5_chain[i] = data[:,29][0:1981]*1.e-9
    theta6_chain[i] = data[:,5][0:1981]
```

In [ ]:

```python
...
```

3. For all pairs of the parameters, compute the covariance. Make a 2-d scatterplot of the chains (as in Problem3-Part 1). Then, plot 68% and 95% confidence ellipses on top of the scatterplots, as in Problem1-Part4. Compare your answers with Problem1-Part4 and Problem2-Part2.

In [ ]:

```python
...
```

In MCMC, we need to make sure that chains converge to the posterior distribution. One useful test for convergence is "Gelman-Rubin statistic." For a given parameter, θ, the R statistic compares the variance across chains with the variance within a chain. Intuitively, if the chains are random-walking in very different places, i.e. not sampling the same distribution, R will be large.

In detail, given chains $J = 1, \ldots, m$, each of length n,

Let $B = \frac{n}{m-1}\sum_j \left(\bar{\theta}_j - \bar{\theta}\right)^2$, where $\bar{\theta}_j$ is the average θ for chain j and $\bar{\theta}$ is the global average. This is proportional to the variance of the individual-chain averages for θ.
Let $W = \frac{1}{m}\sum_j s_j^2$, where s_j^2 is the estimated variance of θ within chain j. This is the average of the individual-chain variances for θ.
Let $V = \frac{n-1}{n}W + \frac{1}{n}B$. This is an estimate for the overall variance of θ.

Finally, $R = \sqrt{\frac{V}{W}}$. We'd like to see $R \approx 1$ (e.g. $R < 1.1$ is often used). Note that this calculation can also be used to track convergence of combinations of parameters, or anything else derived from them.

Reference: https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb (https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb)

4. For all six parameters, compute R and determine if the condition $R < 1.1$ is satisfied.

In [ ]:

```python
...
```

The autocorrelation of a sequence, as a function of lag, k, is defined thusly:

$$\rho_k = \frac{\sum_{i=1}^{n-k}\left(\theta_i - \bar{\theta}\right)\left(\theta_{i+k} - \bar{\theta}\right)}{\sum_{i=1}^{n-k}\left(\theta_i - \bar{\theta}\right)^2} = \frac{\mathrm{Cov}_i\left(\theta_i, \theta_{i+k}\right)}{\mathrm{Var}(\theta)}$$

The larger lag one needs to get a small autocorrelation, the less informative individual samples are.

5. Using autocorrelation_plot from pandas (https://pandas.pydata.org/pandas-docs/stable/visualization.html#visualization-autocorrelation (https://pandas.pydata.org/pandas-docs/stable/visualization.html#visualization-autocorrelation)), plot the auto-correlation of six parameters and determine that it gets small for large lag. The given Planck MCMC chains are already heavily thinned, so you will not see much autocorrelation.

In [ ]:

```python
from pandas.tools.plotting import autocorrelation_plot
...
```

Problem 2 - Supernova Cosmology Project

In this homework, we use a compilation of supernovae data to show that the expansion of the universe is accelerating, and hence it contains dark energy. This is the Nobel prize winning research in 2011 (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/ (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/)), and Saul Perlmutter, a professor of physics at Berkeley, shared a prize in 2011 for this discovery.

"The expansion history of the universe can be determined quite easily, using as a “standard candle” any distinguishable class of astronomical objects of known intrinsic brightness that can be identified over a wide distance range. As the light from such beacons travels to Earth through an expanding universe, the cosmic expansion stretches not only the distances between galaxy clusters, but also the very wavelengths of the photons en route. By the time the light reaches us, the spectral wavelength λ has thus been redshifted by precisely the same incremental factor $z = \Delta\lambda/\lambda$ by which the cosmos has been stretched in the time interval since the light left its source. The recorded redshift and brightness of each such object thus provide a measurement of the total integrated expansion of the universe since the time the light was emitted. A collection of such measurements, over a sufficient range of distances, would yield an entire historical record of the universe’s expansion." (Saul Perlmutter, http://supernova.lbl.gov/PhysicsTodayArticle.pdf (http://supernova.lbl.gov/PhysicsTodayArticle.pdf)).

Supernovae emerge as extremely promising candidates for measuring the cosmic expansion. Type I Supernovae arises from the collapse of white dwarf stars when the Chandrasekhar limit is reached. Such nuclear chain reaction occurs in the same way and at the same mass, the brightness of these supernovae are always the same. The relationship between the apparent brightness and distance of supernovae depend on the contents and curvature of the universe.

We can infer the "luminosity distance" D_L from measuring the inferred brightness of a supernova of luminosity L. Assuming a naive Euclidean approach, if the supernova is observed to have flux F, then the area over which the flux is distributed is a sphere radius D_L, and hence

$$F = \frac{L}{4\pi D_L^2}.$$

In Big Bang cosmology, D_L is given by:

$$D_L = \frac{\chi(a)}{a}$$

where a is the scale factor ($\frac{\lambda_0}{\lambda} = 1 + z = \frac{a_0}{a}$, and the quantity with the subscript 0 means the value at present. Note that $a_0 = 1, z_0 = 0.$), and χ is the comoving distance, the distance between two objects as would be measured instantaneously today. For a photon, $cdt = a(t)d\chi$, so $\chi(t) = c\int_t^{t_0} \frac{dt'}{a(t')}$. We can write this in terms of a Hubble factor ($H(t) = \frac{1}{a}\frac{da}{dt}$), which tells you the expansion rate: $\chi(a) = c\int_a^1 \frac{da'}{a'^2 H(a')} = c\int_0^z \frac{dz'}{H(z')}$. (change of variable using $a = \frac{1}{1+z}$.)

Using the Friedmann equation (which basically solves Einstein's equations for a homogenous and isotropic universe), we can write H^2 in terms of the mass density ρ of the components in the universe: $H^2(z) = H_0^2[\Omega_m(1+z)^3 + (1-\Omega_m)(1+z)^2]$.

Ω is the density parameter; it is the ratio of the observed density of matter and energy in the universe (ρ) to the critical density ρ_c at which the universe would halt is expansion. So Ω_0 (again, the subscript 0 means the value at the present) is the total mass and energy density of the universe today, and consequently $\Omega_0 = \Omega_m$ (matter density parameter today; remember we obtained the best-fit value of this parameter in Project 1?) = $\Omega_{\text{baryonoic matter}} + \Omega_{\text{dark matter}}$. If $\Omega_0 < 1$, the universe will continue to expand forever. If $\Omega_0 > 1$, the expansion will stop eventually and the universe will start to recollapse. If $\Omega_0 = 1$, then the universe is flat and contains enough matter to halt the expansion but not enough to recollapse it. So it will continue expanding, but gradually slowing down all the time, finally running out of steam only in the infinite future. Even including dark matter in this calculation, cosmologists found that all the matters in the universe only amounts to about a quarter of the required critical mass, suggesting a continuously expanding universe with deceleration. Then, using all this, we can write the luminosity distance in terms of the density parameters:

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}} \text{ [unit of Mpc]}$$

where $H_0 = 100 \cdot h\ [\text{km} \cdot \text{s}^{-1}\text{Mpc}^{-1}]$.

Fluxes can be expressed in magnitudes m, where $m = -2.5 \cdot \log_{10}F$ + const. The distance modulus is $\mu = m - M$ (M is the absolute magnitude, the value of m if the supernova is at a distance 10pc. Then, we have:

$$\mu = 25 + 5 \cdot \log_{10}\left(D_L \text{ [in the unit of Mpc]}\right)$$

In this assignment, we use the SCP Union2.1 Supernova (SN) Ia compilation. (http://supernova.lbl.gov/union/ (http://supernova.lbl.gov/union/))

First, load the measured data: $z$ (redshift), $\mu$ (distance modulus), $\sigma(\mu)$ (error on distance modulus)

In [ ]:
```
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]
```

1. Plot the measured distance modulus as a function of redshift with errorbars. Then, assume three different scenarios: $\Omega_m = 0, 0.3, 1$.

Remember:

$$D_L = \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}}$$

$$\mu = 25 + 5 \cdot \log_{10}(D_L)$$

Now, plot three curves of $\mu$ as a function of $z$ for $\Omega_m = 0, 0.3, 1$ on top of the measured data (Calculate D_L using quad. For now, assume $h = 0.7$.) How do they fit?

In [ ]:
```
...
```

In [ ]:
```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

You should find that the measured data do not fit well to all three scenarios. "The high-redshift supernovae are fainter than would be expected even for an empty cosmos (corresponding to $\Omega_m = 0$)." So what's wrong?

"If these data are correct, the obvious implication is that the simplest cosmological model must be too simple. The next simplest model might be one that Einstein entertained for a time. Believing the universe to be static, he tentatively introduced into the equations of general relativity an expansionary term he called the “cosmological constant” (Λ) that would compete against gravitational collapse. After Hubble’s discovery of the cosmic expansion, Einstein famously rejected Λ as his “greatest blunder.” In later years, Λ came to be identified with the zero-point vacuum energy of all quantum fields. It turns out that invoking a cosmological constant allows us to fit the supernova data quite well." (Saul Perlmutter, https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/ (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/))

So in short, the data indicates that faint supernovae are further away from the earth than had been theoretically expected. The expansion rate of the universe is increasing indeed. It seems that some mysterious material (which we call "dark energy") is causing such antigravity effects. The cosmological constant, Λ, the value of the energy density of the vacuum of space is widely accepted as a leading candidate of dark energy.

Now let us add a general form of dark energy to our model.

$$H^2(z) = H_0^2[\Omega_m(1+z)^3 + \Omega_{DE}(1+z)^{3(1+w)} + (1-\Omega_m-\Omega_{DE})(1+z)^2].$$

w is the dark energy equation of state, which is the ratio of its pressure to its energy density. $w = -1$ for the cosmological constant Λ.

$\Omega_0 = \Omega_m$ (matter density parameter today) + Ω_{DE} (dark energy density parameter today), and

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + \Omega_{DE}(1+z')^{3(1+w)} + (1-\Omega_m-\Omega_{DE})(1+z')^2]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + \Omega_{DE}(1+z')^{3(1+w)} + (1-\Omega_m-\Omega_{DE})(1+z')^2]^{1/2}} \text{ [unit of Mpc]}$$

where $H_0 = 100 \cdot h$ [km $\cdot$ s^{-1}Mpc^{-1}].

2. Now assume three different scenarios: ($\Omega_m = 0.3, \Omega_{DE} = 0$), ($\Omega_m = 0, \Omega_{DE} = 1, w = -1$), and ($\Omega_m = 0.3, \Omega_{DE} = 0.7, w = -1$). Again, plot three curves of μ as a function of z on top of data (assume $h = 0.7$)

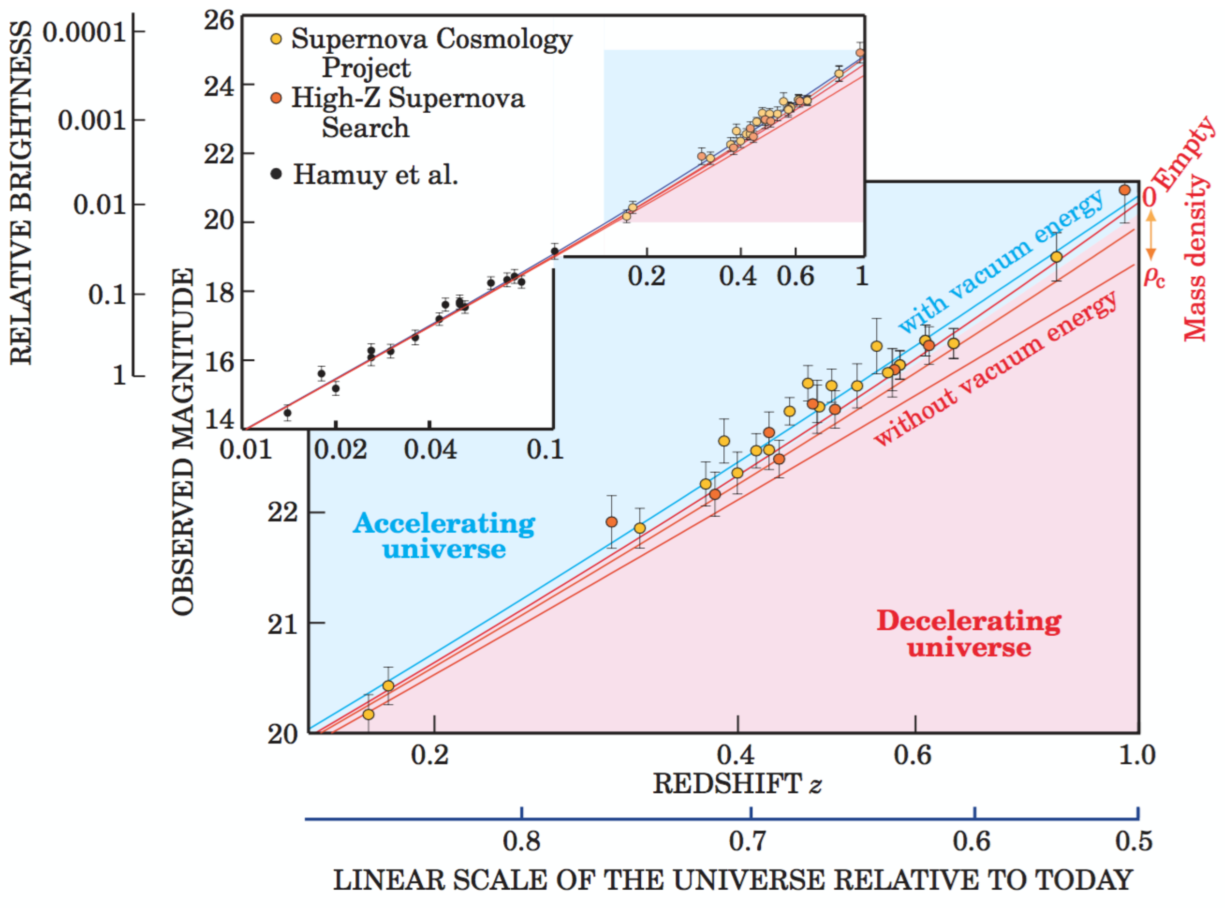
In [ ]:
```
...
```

In [ ]:
```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

You basically reproduced the below figure!



You should see that $\Omega_m = 0.3$ and $\Omega_m = 0.7$ fits the data best. In combination with the CMB data, this shows that about 70% of the total energy density is vacuum energy and 30% is mass.

Now, with measurements of the distance modulus μ, use Bayesian analysis to estimate the cosmological parameters.

let us assume that the universe is flat (which is a fair assumption since the CMB measurements indicate that the universe has no large-scale curvature). $\Omega_0 = \Omega_m + \Omega_{DE} = 1$. Then, we do not need to worry about the curvature term:

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}} \text{ [unit of Mpc]}$$

where $H_0 = 100 \cdot h$ [km $\cdot$ s^{-1}Mpc^{-1}].

Assuming that errors are Gaussian (can be justified by averaging over large numbers of SN; central limit theorem), we calculate the likelihood L as:

$$L \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{N_{SN}} \frac{[\mu_{i,\,\mathrm{data}}(z_i) - \mu_{i,\,\mathrm{model}}(z_i, \Omega_m, w)]^2}{\sigma(\mu_i)^2}\right)$$

where z_i, μ_i, $\sigma(\mu_i)$ are from the measurements, and we compute μ_{model} as a function of z, Ω_m, w.

Next, write an MCMC code using the **Metropolis algorithm**.

Now, assume a more general form of dark energy. (Do not fix w to -1; add w as a parameter.)

In the flat universe,

$$D_L = \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}} \text{ [unit of Mpc]}$$

where $H_0 = 100 \cdot h\ [\mathrm{km} \cdot \mathrm{s}^{-1}\mathrm{Mpc}^{-1}]$. Here, we fix $h = 0.7$.

We calculate the likelihood L as:

$$\ln(L) \approx -\frac{1}{2}\sum_{i=1}^{N_{SN}} \frac{[\mu_{i,\,\mathrm{data}}(z_i) - \mu_{i,\,\mathrm{model}}(z_i, \Omega_m, w)]^2}{\sigma(\mu_i)^2} = -\frac{1}{2}\sum_{i=1}^{N_{SN}} \frac{\Delta\mu_i^2}{\sigma(\mu_i)^2}$$

where

$$\mu_{i,\,\mathrm{model}}(z_i, \Omega_m, w) = 25 + 5 \cdot \log_{10}(D_{L,\,i})$$

$$D_{L,\,i} = \frac{2997.92458}{0.7}(1+z_i)\int_0^{z_i} \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}$$

6. Run the MCMC code to estimate w and Ω_m. Plot 1-d posterior of w and Ω_m as well as 2-d posterior (i.e. plot the chain in two-dimensional parameter space. Make sure that the chain has converged (you can change nsamples, nburn).

Hint:

Set the length of MCMC chains to be 15,000 (or even more if you think that the chain has not yet converged.) In the end, you should throw away the first 20% of the chain as burn-in. (20% is an arbitrary number. You can plot the chain and estimate the burn-in period.)

Then, set the random initial point in the parameter space (w, Ω_m): let w be negative and Ω_m be positive and draw a random number using np.random.uniform(). Set initial likelihood to low value (e.g. -1.e100) so that next point is accepted.

Now, draw a new sample starting from this random initial point. Here we assume that the proposal distribution is Gaussian with arbitrary width: in this problem, we assume that $\sigma = 0.01$ (This determines how far you propose jumps.) for distributions for both w, Ω_m.

For example, say that you start with $(w, \Omega_m) = (-0.3, 0.7)$. Then, draw a new sample of w from a Gaussian with $\mu = -0.3, \sigma = 0.01$ and a new sample of Ω_m from a Gaussian with $\mu = -0.7, \sigma = 0.01$.

Now, evaluate the log likelihood value of this new point.

If the value has gone up, accept the point.

Otherwise, accept it with probability given by ratio of likelihoods: Draw a random number from a uniform distribution between 0 and 1 (α = np.random.uniform()). If the ratio $\ln(\frac{L_{\mathrm{new}}}{L_{\mathrm{old}}})$ is greater than $\ln(\alpha)$ (i.e. $\frac{L_{\mathrm{new}}}{L_{\mathrm{old}}} > \alpha$), then accept it. Otherwise, reject it and stay at your old point.

Repeat this 15,000 times (the length of chain) and plot the distributions of (w, Ω_m).

In [ ]:

```python
# Import data
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]

# length of MCMC chains
nsamples = 15000
# number of parameters: Omega_m and w
npars    = 2

# Define (gaussian) width of the proposal distribution, one for each parameter. This determines how far you propose jumps
Sigma = [0.01, 0.01]

# Number of supernova:
nSN = len(z_data)

# Declare an empty array of the parameter values of each point.
# Theta[:,0] stores a trace of the parameter \Omega_m
# Theta[:,1] stores a trace of the parameter w
# Theta[:,2] stores log-likelihood values at each point
Theta = np.empty([nsamples,npars+1])

# Dmu stores mu(data)-mu(theory), temporarily:
Dmu = np.empty(nSN)

# Random starting point in parameter space
# Set initial likelihood to low value so next point is accepted (could compute it instead):
Theta[0,:] = [np.random.uniform(), -np.random.uniform(), -1.e100]
```

In [ ]:

```python
# Define the likelihood function:

def lnL(Omegam, w):

    # Treat unphysical regions by setting likelihood to (almost) zero:
    if(Omegam<=0 or w>=0):
        lnL = -1.e100
    else:

    # Compute difference with theory mu at redshifts of the SN, for trial Omegam
    # Compute ln(likelihood) assuming gaussian errors
        ...

    return lnL
```

In [ ]:

```python
# Draw new proposed samples from a proposal distribution, centred on old values
# Accept or reject, and colour points according to ln(likelihood):

# Compute initial likelihood value:
Theta[0,npars] = lnL(Theta[0,0], Theta[0,1])

for i in range(1,nsamples):

    # log likelihood value of previous point
    lnL_previous = Theta[i-1,npars]

    # Draw a new sample around the previous point
    Omegam_new = np.random.normal(Theta[i-1,0],Sigma[0])
    w_new = np.random.normal(Theta[i-1,1],Sigma[1])
    # log likelihood value of current point
    lnL_new    = lnL(OmegamProp, wProp)

    # Metroplis-Hastings algorithm:

    if(lnL_new > lnL_previous):
    # Accept point if likelihood has gone up:
        ...
    else:
    # Otherwise accept it with probability given by ratio of likelihoods:
        alpha = np.random.uniform()

        if(lnL_new - lnL_previous > np.log(alpha)):
        # Accept point if the likelihood ratio is greater than alpha:
            ...
        else:
        # Reject; Repeat the previous point in the chain:
            ...

# Remove a burn in period, arbitrarily chosen to be the first 20% of the chain:
nburn = 2*math.floor(nsamples/10)
```

In [ ]:

```python
# Plot the histogram of Omegam after the burn-in phase:
...
# Plot the histogram of w after the burn-in phase:
...

# Scatter plot of the samples (2-d posterior):
...

# Print best-fit values and constraints
print ('Omega_m = ',np.mean(Theta[nburn:nsamples,0]), '+/-' ,np.std(Theta[nburn:nsamples,0]))
print ('w = ',np.mean(Theta[nburn:nsamples,1]), '+/-' ,np.std(Theta[nburn:nsamples,1]))

```

To Submit

Execute the following cell to submit. If you make changes, execute the cell again to resubmit the final copy of the notebook, they do not get updated automatically.
We recommend that all the above cells should be executed (their output visible) in the notebook at the time of submission.
Only the final submission before the deadline will be graded.

In [ ]:

```python
_ = ok.submit()
```