midterm

March 19, 2020

1 ASTR 596: FDS - The Midterm

1.0.1 Dun, dun dun.

(suspenseful music here)

1.1 Start with the Cepheid data you have already seen in lecture 03 and 04.

1.2 There are measurements of several Cepheid variable stars in each of 9 nearby galaxies.

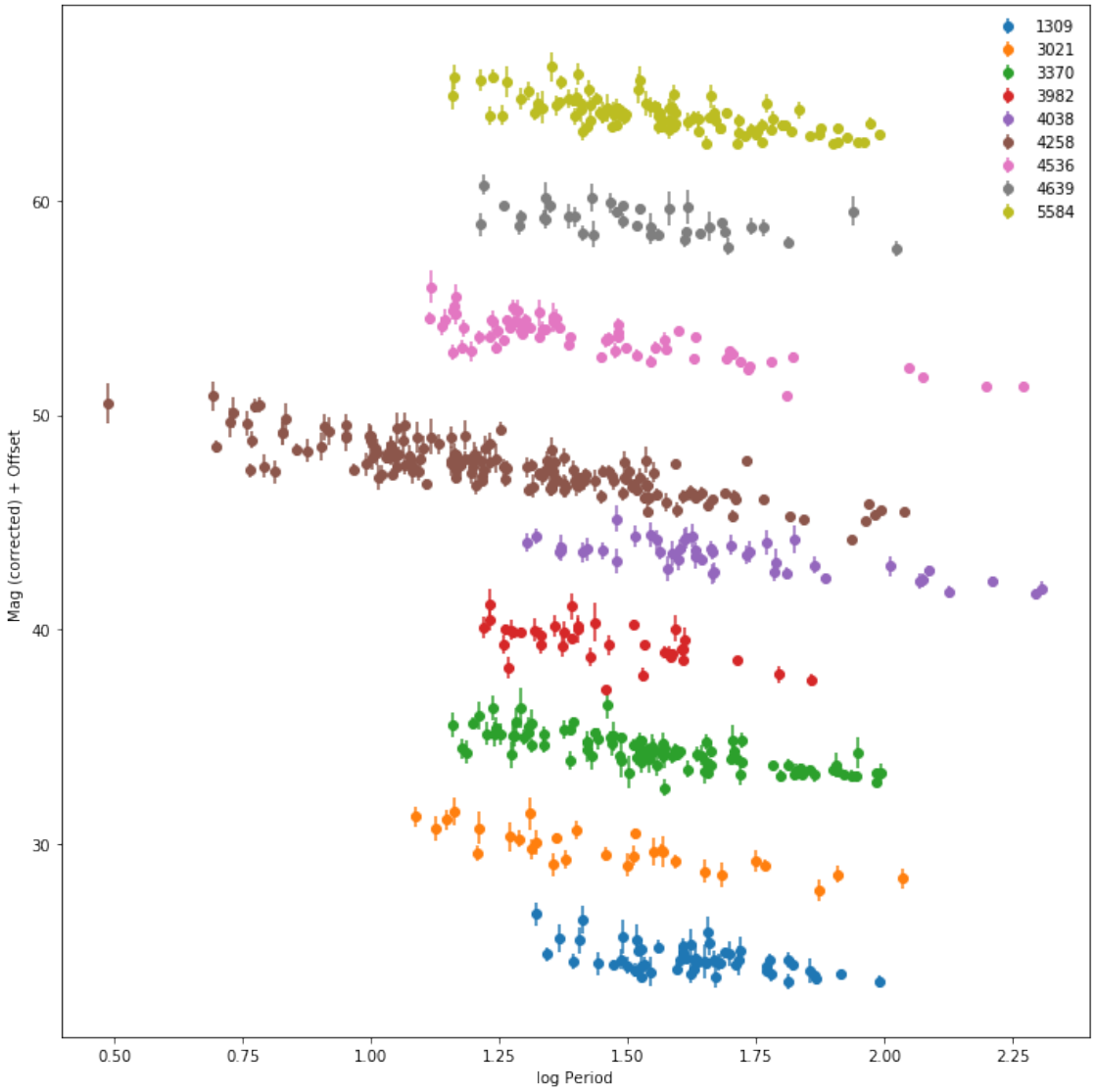
1.3 Begin by reading this code and familiarizing yourself with what attributes and functions are available to you. In particular, it will help to read comments at the top of the data file.

[2]:
```
exec(open('cepheids.py').read())
ceph = Cepheids('../../data/03/R11ceph.dat')
hosts = ceph.list_hosts()
print(hosts)
```

```
['1309' '3021' '3370' '3982' '4038' '4258' '4536' '4639' '5584']
```

[3]:
```
%matplotlib inline
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111)

for i, ID in enumerate(hosts):
    ceph.select(ID)
    mobs = ceph.mobs
    logP = ceph.logP
    sigma_obs = ceph.sigma
    ax.errorbar(logP, mobs+(5*i), yerr=sigma_obs, linestyle='None', marker='o',␣
 ↪label=ID)

ax.legend(frameon=False)
ax.set_xlabel('log Period')
```

```
ax.set_ylabel('Mag (corrected) + Offset ')
fig.tight_layout()
```



2 Q1.

2.1 This data was taken from Riess et al., 2011

2.2 The global model for the magnitudes of these Cepheids is given in Equation 1. You may ignore the effect of metalicity

2.3 Some parameters (such as distance modulus) are specific to each galaxy.

2.4 Others are common to all galaxies.

2.5 Assuming Gaussian magnitude errors and no period error (and you may ignore metallicity entirely), but that Cepheids scatter off the period-luminosity relation because of some Gaussian intrinsic dispersion σ_{int}, write down your expression for the likelihood being careful to use the same indices that Riess et. al used.

The originial equation (1) from Riess et al (2011), ignoring metallicity is:

$$m_{W,i,j} = (\mu_{0,i} - \mu_{0,4258}) + zp_{W,4258} + b_W \times log(P_{i,j})$$

where $m_{W,i,j}$ and P are observables and $zp_{W,4258}$ and b_W are nuisance parameters.

$\mu_{0,i} - \mu_{0,4258}$ is the different for each galaxy i, but same for each Cephids j in the same galaxy,

so we can view $\mu_{0,i} - \mu_{0,4258}$ as additional nuisance parameters,

and there are as many of them as the numbers of individual galaxies in the data.

The likelihood is

$$L \equiv \prod_{i=1}^{n} \prod_{j=1}^{m} p(\{m_{W,i,j}, P_{i,j}\} | M(zp_{W,4258}, b_W, \mu_{0,i} - \mu_{0,4258}))$$

which can be written to be

$$L \equiv \prod_{i=1}^{n} \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_{int}} exp(\frac{-(m_{W,i,j} - Model(P_{i,j}, zp_{W,4258}, b_W, \mu_{0,i} - \mu_{0,4258}))^2}{2\sigma_{int}})$$

and the log likelihood is

$$lnL = \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} \frac{-(m_{W,i,j} - Model(P_{i,j}, zp_{W,4258}, b_W, \mu_{0,i} - \mu_{0,4258}))^2}{2\sigma_{int}^2}$$

3 Q2.

3.1 Given what you know for each of the parameters in this model, write down physically motivated (i.e. not just top hats on everything) priors, along with your explanation.

3.2 Think particularly about the priors for distances to each galaxy.

$zp_{W,4258}$ is the intercept in y-axis and b_W is the slope of the data of NGC4258, so the priors for them can be determined by looking at the NGC4258 data by eye. $zp_{W,4258}$ is around 30, and b_W is negative. I use normal distributions for priors for $zp_{W,4258}$ and b_W, based on my inital fitting results with scipy minimize (see below for plots).

We have n different $\mu_{0,i} - \mu_{0,4258}$ paramters for galaxy distances, where n is the number of unique galaxies in the dataset. Since NGC4258 acts as the baseline, $\mu_{0,i} - \mu_{0,4258}$ is 0 for this galaxy and we reduce to $n-1$ paramters for galaxy distances. μ is the distance modulus and is related to the physical distance d by $\mu = 5log_{10}(d) + 5$. We expect the fitted distances to follow normal distributions, and therefore fitted μ and their priors should also follow normal distributions. The initial guess are also based on the scipy minimize fitting results.

4 Q3.

4.1 Implement this likelihood and prior in python and sample the posterior distribution with emcee.

4.2 Construct traceplots for each parameter, as well as corner plots for each pair of parameters.

4.3 If your Markov Chains are reasonable, verify that your model fits are reasonable by reconstructing Fig. 7 of Riess et al. 2011

4.4 Compare the intercept you find for the intercepts $\mu_{0,i} - \mu_{0,4258}$ vs Table 3 of Riess et al. 2011

[4]:
```
import emcee
import numpy as np
import matplotlib.pyplot as plt
import corner

def PLmodel(logP, zp, bw, ui0):
    return bw*logP + zp + ui0;

def chisq(params, host, mag, mag_err, logP):
    zp = params[0]
    bw = params[1]
    ui = params[2:]
```

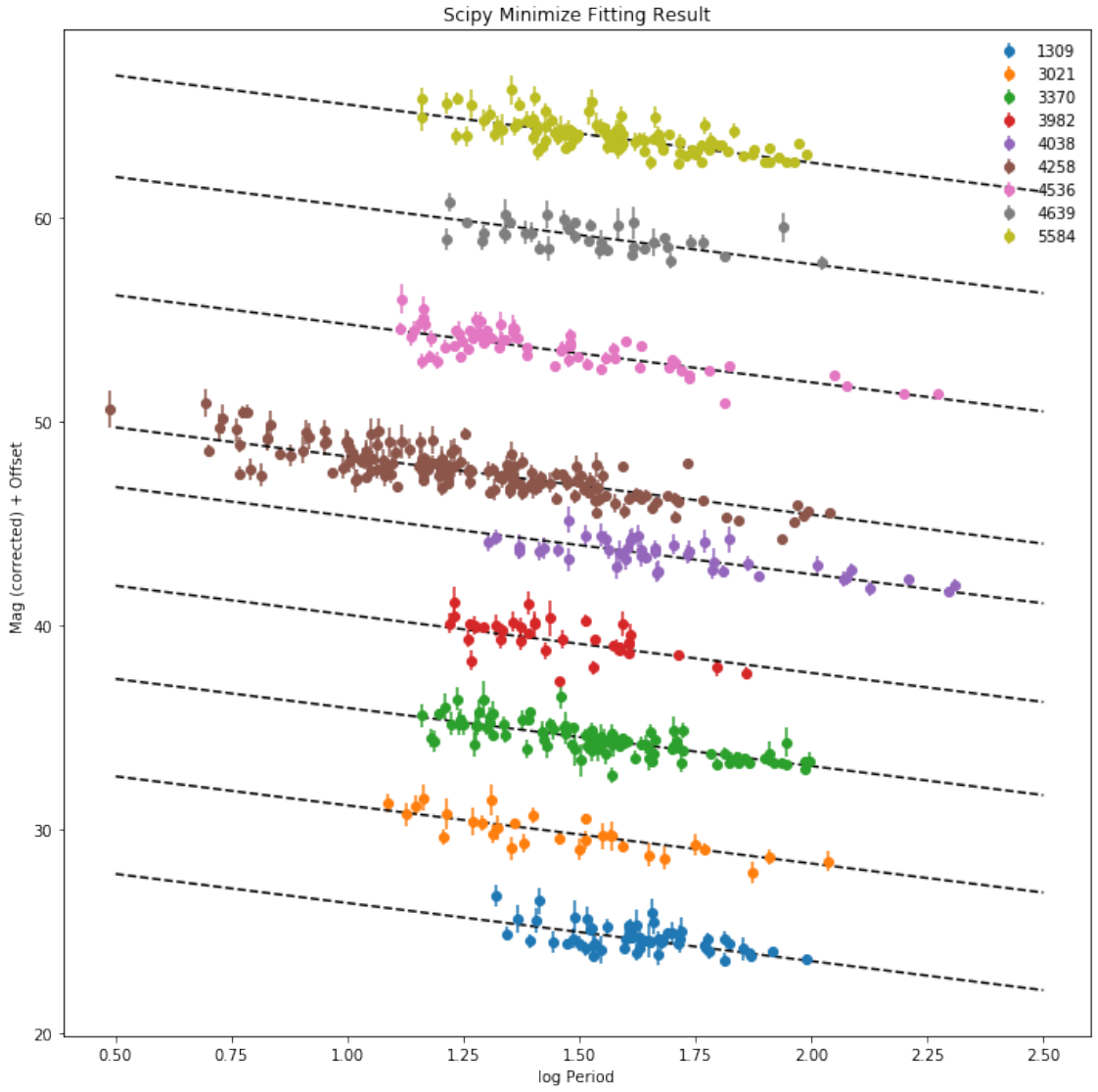
```
    #print('data shape',len(args[0]))
    #host, mag, mag_err, logP = args

    host0 = np.unique(host)[np.where(np.unique(host)!=4258)]
    uij = np.zeros(len(host))
    for i in range(len(host)):
        if host[i]!=4258:
            ## for NGC4258, the distance is fixed to 0, if host is not 4258,␣
 ↪fit for the reddening-free distance.
            uij[i] = ui[np.where(host[i]==host0)[0][0]]

    chisq = np.sum((mag-PLmodel(logP, zp, bw, uij))**2/mag_err**2)
    return chisq;

def logLikelihood(params, host, mag, mag_err, logP):
    return -0.5*chisq(params, host, mag, mag_err, logP);
```

[5]:
```
## read in data, try using the scipy optimization for initial guess
import scipy.optimize as so
hostid, mag, mag_err, period = np.loadtxt('../../data/03/R11ceph.
 ↪dat',comments='#',usecols=(1,2,3,4)).T
logP0 = np.log10(period)

ngal = len(np.unique(hosts))
u0 = np.random.normal(loc=2.,size=ngal-1)
zp0 = 30.
bw0 = -1.
p0 = [zp0,bw0]+list(u0)

res = so.minimize(chisq, p0, args=(hostid, mag, mag_err, logP0))
print(res.x)
```

```
[26.11796463 -2.8507764   3.09898642  2.8863536   2.66584117  2.23493854
  2.07466598  1.48356471  2.28567951  2.26186255]
```

[6]:
```
## plot the results from scipy minimize
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111)

plot_logP = np.linspace(0.5,2.5,num=100)
host0 = np.unique(hostid)[np.where(np.unique(hostid)!=4258)]

for i, ID in enumerate(hosts):
    ceph.select(ID)
    mobs = ceph.mobs
    logP = ceph.logP
    sigma_obs = ceph.sigma
```

```
    ax.errorbar(logP, mobs+(5*i), yerr=sigma_obs, linestyle='None', marker='o',␣
↪label=ID)
    if ID !='4258':
        u0 = res.x[np.where(float(ID)==host0)[0]+2]
    else:
        u0 = 0.
    ax.plot(plot_logP, (5*i)+PLmodel(plot_logP, res.x[0], res.x[1],␣
↪u0),linestyle='--',color='k')

ax.legend(frameon=False)
ax.set_xlabel('log Period')
ax.set_ylabel('Mag (corrected) + Offset ')
plt.title('Scipy Minimize Fitting Result')
fig.tight_layout()
```



[7]:
```
## set up the priors and probability
import scipy.stats as st
def logPrior(params):
    zp = params[0]
    bw = params[1]
    ui = params[2:]

    if  zp>0 and zp<100 and bw>-10 and bw<0 and np.max(ui)<6 and np.min(ui)>-1.:

        ## for the gaussian priors, we don't want a very constraining prior,
        ## so we set the width of the prior to 1/10 of the parameter values
        p_zp = np.log(st.norm.pdf(zp, loc=res.x[0], scale=res.x[0]/10.))
        p_bw = np.log(st.norm.pdf(bw, loc=res.x[1], scale=res.x[0]/10.))

        prior = p_zp+p_bw

        for i in range(len(ui)):
            p_ui = np.log(st.norm.pdf(ui[i], loc=res.x[i+2], scale=res.x[i+2]/
 ↪10.))
            prior = prior+p_ui

        return prior;
    else:
        return -np.inf;

def logProb(params, host, mag, mag_err, logP):
    #print (logPrior(params)+logLikelihood(params,*data))
    if logPrior(params)>-100.:
        return logPrior(params)+logLikelihood(params, host, mag, mag_err, logP);
    else:
        return -np.inf;
```

[8]:
```
## run MCMC, using the scipy minimize results as prior

nwalkers = 100
ndim = len(p0)

p0 = res.x
pos = [p0 + np.random.randn(ndim) for i in range(nwalkers)]

sampler = emcee.EnsembleSampler(nwalkers, ndim, logProb, args=(hostid, mag,␣
 ↪mag_err, logP0))
sampler.run_mcmc(pos, 500)
```

```
/Users/jell0727/anaconda/envs/fds/lib/python3.7/site-
packages/emcee/moves/red_blue.py:99: RuntimeWarning: invalid value encountered
```

```
    in double_scalars
      lnpdiff = f + nlp - state.log_prob[j]
```

[8]:
```
State([[26.06268363 -2.82040482  3.09091215  2.96161687  2.72556804  2.234043
    2.049766    1.46190789  2.30975198  2.26731172]
  [26.14048558 -2.84147381  3.06377004  2.78758537  2.65860674  2.163792
    2.07926707  1.45122383  2.22866098  2.1899563 ]
  [26.11719466 -2.83412044  3.04746259  2.78126293  2.669154    2.21691203
    2.13796076  1.44492334  2.198052    2.205247  ]
  [26.22665857 -2.92271772  3.07906817  2.87863345  2.72295906  2.19535372
    2.10542024  1.49657195  2.33969346  2.25616876]
  [26.2593412  -2.91721329  3.03284414  2.83129962  2.67242365  2.17959161
    2.00732193  1.39183143  2.29128933  2.22411027]
  [26.14810259 -2.86613503  3.15259883  2.93066287  2.62996871  2.20413852
    1.97851234  1.41802904  2.28411082  2.21756031]
  [26.25443936 -2.9361075   3.12818249  2.89778472  2.66375933  2.22071036
    2.02018358  1.50399216  2.22236359  2.27263059]
  [26.09604254 -2.81005316  3.0103285   2.96050761  2.60546471  2.15382738
    2.04510791  1.43816394  2.31686984  2.18474057]
  [26.27654766 -2.94000677  3.10911908  2.74715706  2.68639649  2.27845977
    2.0331544   1.46508669  2.33393201  2.25649909]
  [26.06067075 -2.82562479  3.12919692  2.92062096  2.64778063  2.30788484
    2.07722781  1.48549163  2.36915164  2.27232532]
  [26.09582197 -2.83120329  3.09133992  2.82882371  2.65143597  2.20128091
    2.07187745  1.50575883  2.29304444  2.28191336]
  [26.10573847 -2.83524121  3.08555793  2.92408124  2.63137906  2.2110872
    2.07065672  1.53231588  2.29399626  2.20742249]
  [26.13034266 -2.84223743  3.0123413   2.92052833  2.65856576  2.23532137
    2.16006862  1.46144769  2.27705717  2.17515336]
  [26.03393592 -2.79107249  3.04884896  2.92692746  2.65075426  2.23114397
    2.0282698   1.46666621  2.3694791   2.26255661]
  [26.19446476 -2.88016453  3.04295341  2.96121297  2.57807552  2.17675537
    1.98261931  1.47798616  2.29527165  2.25485781]
  [26.17934686 -2.87302304  3.11057844  2.80094017  2.63658788  2.24422379
    2.14811037  1.42877989  2.27038563  2.24283767]
  [26.13246449 -2.85057972  3.07942116  2.89998548  2.6627572   2.19526212
    2.04026282  1.49676423  2.27162617  2.27570155]
  [26.08663625 -2.85058811  3.19289358  2.95081365  2.68506383  2.25329565
    2.08055162  1.48817466  2.24105323  2.24860556]
  [26.12080211 -2.85683622  3.10983241  2.82559303  2.63459093  2.21926276
    2.05971374  1.47959917  2.25248482  2.24360551]
  [26.00650096 -2.78615261  3.10372093  2.96207016  2.64791071  2.26455021
    2.05485519  1.48057105  2.22328784  2.27718204]
  [26.21009978 -2.92897916  3.10511787  2.88912002  2.71821282  2.257477
    2.17682345  1.50461285  2.32813558  2.27548745]
  [26.10897411 -2.83175019  3.10817783  2.98547031  2.65020123  2.19561456
    1.94594841  1.49405086  2.28950085  2.2479803 ]
```

```
[26.20108766 -2.89685986  3.14652143  2.91421435  2.69583163  2.22658654
  2.09248948  1.47222169  2.27560435  2.22277369]
[26.16681502 -2.86491374  3.11922633  2.72532598  2.65252893  2.25686352
  2.07916996  1.46700469  2.23379793  2.28677784]
[26.19160218 -2.88291286  3.04240463  2.95144612  2.65006733  2.254029
  2.14257762  1.48745742  2.28539586  2.23198392]
[26.13721801 -2.83993151  3.09221061  2.79102408  2.59402747  2.22058123
  1.93350541  1.40840738  2.22140252  2.21167027]
[26.12428538 -2.84255441  3.17653634  2.79961786  2.64511672  2.19425102
  1.9948224   1.47892586  2.34677286  2.17690292]
[26.01830906 -2.81123687  3.15820896  2.91598839  2.67683272  2.27707835
  2.15690411  1.54141346  2.33553214  2.28644243]
[26.11132773 -2.84830402  3.12712054  2.96001736  2.64921654  2.27385614
  2.12105235  1.49898111  2.25033149  2.26996879]
[26.02822579 -2.77786583  3.06425368  2.80397823  2.61805928  2.21932706
  2.05988268  1.44387788  2.16512201  2.22235983]
[26.03714072 -2.80751808  3.10765307  2.93215414  2.68688862  2.24447834
  2.01886739  1.50947839  2.193997    2.32248125]
[26.13081865 -2.84952561  3.12497016  2.82840515  2.72456998  2.14924397
  2.02680664  1.47763491  2.22983249  2.24715277]
[26.05227806 -2.81988129  3.09092136  2.96004522  2.63986949  2.30984062
  2.06820766  1.50964764  2.33942708  2.2418181 ]
[26.03344935 -2.78921424  3.08402562  2.95062206  2.63344694  2.25216592
  1.97190577  1.48024284  2.30626643  2.24695723]
[26.07824946 -2.81540911  3.09245788  2.92206053  2.630336    2.15937442
  2.08015047  1.46234597  2.23500715  2.23722027]
[26.24828471 -2.94110748  3.05782979  2.85722451  2.65935715  2.17136981
  2.12953454  1.48871642  2.29617439  2.27203371]
[26.08365132 -2.82865577  3.10564304  2.73939574  2.67513813  2.26336902
  2.00014649  1.47363021  2.29934992  2.29398998]
[26.08010723 -2.81325846  3.05088987  2.84749095  2.66606895  2.23769304
  1.99227966  1.47601933  2.13498259  2.21219527]
[26.14791428 -2.86822951  3.10427995  2.88574824  2.6787496   2.23079058
  2.07798861  1.46030994  2.20082794  2.24358606]
[26.18161163 -2.89336758  3.07446917  2.86671904  2.70688505  2.23940387
  2.14232734  1.51855391  2.19655518  2.30209219]
[26.08621766 -2.82503138  3.05791265  2.75060171  2.65569512  2.24000832
  2.03915148  1.47968232  2.22146082  2.24134401]
[26.1914687  -2.87095236  3.06488817  2.86587823  2.66013978  2.17438352
  2.01316565  1.39378499  2.2487214   2.24523514]
[26.0851634  -2.83230159  3.11194963  2.81917406  2.66208402  2.21265506
  2.02985505  1.49144933  2.32616181  2.28049727]
[26.18655511 -2.89168793  3.09896921  2.83194145  2.69210394  2.18594689
  2.05517873  1.49268903  2.2531437   2.28382133]
[26.25913968 -2.93191976  3.08921801  2.83161149  2.6701278   2.2417698
  2.26075717  1.46981621  2.21191074  2.22316694]
[26.18547227 -2.87920404  3.0594088   2.76005163  2.63845894  2.19151266
```

```
  2.02260691  1.42722332  2.24415507  2.24934764]
[26.19051384 -2.90393529  3.17106422  2.85644162  2.72149722  2.23294178
  2.10305769  1.48204453  2.21213586  2.26715168]
[26.07037015 -2.82726795  3.13351657  2.91663968  2.63548601  2.25189577
  2.05244637  1.50503291  2.36261215  2.22569346]
[26.24925679 -2.93095625  3.12253948  2.95474966  2.6778054   2.20583771
  2.04417362  1.49442169  2.2926837   2.2818365 ]
[26.02097208 -2.79708857  3.06350412  2.88953668  2.73308445  2.27847764
  2.04063537  1.47927442  2.36014124  2.24123575]
[26.13062826 -2.86158342  3.11942944  2.94783923  2.7264634   2.21133796
  2.06068421  1.51073726  2.28142151  2.25189681]
[26.15877151 -2.86448369  3.03968575  2.80521258  2.71220919  2.23499543
  2.09049415  1.49473505  2.27478596  2.22656883]
[26.20565864 -2.89366939  3.06345186  2.92240628  2.67000597  2.24304577
  1.98071101  1.48430805  2.23149387  2.2890457 ]
[26.11579163 -2.84299828  3.11129036  2.92347822  2.66524974  2.23972712
  1.99871446  1.47776828  2.31086152  2.26079048]
[26.13328803 -2.8642797   3.05845265  2.96117911  2.67657662  2.15016507
  2.16531485  1.48740268  2.33697553  2.25445941]
[26.15346707 -2.85888223  3.12767704  2.79115873  2.64739781  2.24016274
  2.03144373  1.45717744  2.2178355   2.19232295]
[26.0272749  -2.80018192  3.11790958  2.80244368  2.67134079  2.27937386
  2.14927576  1.51481675  2.25642992  2.27295284]
[26.01763911 -2.77945564  3.07174412  2.90788845  2.61595705  2.22998708
  2.08355254  1.41188736  2.23269821  2.23753045]
[26.00919717 -2.79532558  3.08139603  2.84160912  2.67128611  2.33500422
  2.07190776  1.4918443   2.38888002  2.30366317]
[26.16890709 -2.88663055  3.17570128  2.82432133  2.64225889  2.26547166
  2.10631906  1.49019915  2.37172727  2.25112991]
[26.20504091 -2.92869503  3.1507406   2.94033956  2.73259704  2.29511052
  2.16825742  1.50780493  2.37940238  2.33852232]
[26.04736477 -2.80531248  3.09934627  2.83744763  2.65656422  2.2881799
  2.02900199  1.47955948  2.28006022  2.27013817]
[26.02100296 -2.79783356  3.15614939  2.74749659  2.69321923  2.22429674
  2.07451178  1.50567298  2.32944447  2.28298781]
[26.0863381  -2.8088229   2.98413041  2.89242866  2.66466224  2.17325386
  2.09135034  1.49971758  2.25094135  2.17696895]
[26.07236524 -2.84330104  3.06086979  2.89823363  2.67807582  2.24459132
  2.07727215  1.51743737  2.32295613  2.25921281]
[26.14929284 -2.86094681  3.05758507  2.87718174  2.63742955  2.26138184
  2.02121029  1.4639805   2.26854429  2.22839141]
[26.16454802 -2.85978907  3.11179536  2.91970815  2.6236959   2.17370812
  2.12743622  1.52798679  2.23781347  2.2489148 ]
[26.10222455 -2.84153631  3.11376319  2.81655172  2.62097575  2.21811033
  2.0569123   1.49582771  2.15460619  2.28931706]
[26.08816044 -2.84678735  3.10390544  2.82424602  2.73803004  2.25291732
  2.14333885  1.50843342  2.27573534  2.27161108]
```

```
[26.1512383  -2.86381876  3.12717404  2.841339    2.64665313  2.17763428
  2.02787161  1.4836973   2.27885527  2.29982703]
[26.24788661 -2.91373905  3.04830574  2.85919965  2.61408094  2.20704889
  2.07225135  1.45352681  2.16698643  2.1836588 ]
[26.26765026 -2.94648833  3.15341391  2.93490084  2.66596244  2.21679418
  2.05485491  1.46727777  2.29081422  2.23337623]
[26.18624083 -2.8903333   3.13093656  2.79145757  2.68975664  2.20643257
  2.09779854  1.53630488  2.2299528   2.27792164]
[26.06118511 -2.81728132  3.08580253  2.83214087  2.63118236  2.25423756
  2.09764307  1.49628293  2.23599853  2.21814352]
[26.21315501 -2.90306821  3.13196764  2.90622242  2.68477804  2.22446505
  2.06592404  1.44355906  2.30658622  2.21128587]
[26.19738755 -2.88610475  3.1729714   2.96487243  2.6626307   2.2551748
  2.07195656  1.49708358  2.25428346  2.25919779]
[26.17534965 -2.86803197  3.08440407  2.90648416  2.6373342   2.16168033
  1.98667768  1.47044249  2.2213969   2.16643091]
[26.02576654 -2.78077227  3.04681805  2.88996175  2.6145711   2.28254425
  2.05468504  1.43772361  2.27773959  2.22027604]
[26.12501645 -2.83441356  3.08431915  2.92296807  2.66634188  2.16211642
  2.07121521  1.43392212  2.24933709  2.23602337]
[26.01665919 -2.78244481  3.07892607  2.88024614  2.64457691  2.23654749
  2.14842833  1.44275876  2.28833634  2.26088861]
[26.14210508 -2.85155837  3.05162733  2.84132901  2.62226756  2.17001412
  2.07147164  1.49811301  2.25091391  2.23288035]
[26.10605879 -2.82668207  3.0948463   2.94878442  2.62099819  2.20872348
  1.98217628  1.40363447  2.2517142   2.24795644]
[26.13491151 -2.84831789  3.02992526  2.80723289  2.62864173  2.14250897
  2.04671287  1.49103696  2.21604794  2.2237646 ]
[26.19868519 -2.91641705  3.15605742  2.78567889  2.67189043  2.25148406
  2.06362507  1.45469126  2.20183897  2.31355381]
[26.06142197 -2.81910484  3.18389404  2.80162631  2.63114178  2.2819139
  2.0109449   1.48650017  2.28395527  2.24507135]
[26.0715851  -2.83861476  3.11067886  2.90853243  2.72446387  2.28203131
  2.14572325  1.49320259  2.38933075  2.289051  ]
[26.08709518 -2.81728453  3.03123147  2.85122415  2.60418925  2.26773392
  2.04060541  1.44931134  2.29607431  2.21753821]
[26.06221746 -2.81821299  3.08528828  2.85449458  2.70377981  2.17076214
  2.10183344  1.53021445  2.29649633  2.21237961]
[25.97393461 -2.73833417  3.06649647  2.75676387  2.62919249  2.24497178
  2.00702537  1.4514051   2.23176806  2.18491849]
[26.08717403 -2.81877401  3.03576972  2.75713614  2.67051712  2.19841894
  2.05357571  1.46984183  2.29905365  2.21089498]
[26.0616275  -2.82176849  3.08822249  2.88463054  2.69852786  2.24896246
  2.17060577  1.41251     2.34647982  2.29939427]
[26.07130392 -2.84517465  3.14940445  2.9406833   2.67416158  2.24810875
  2.05570869  1.53074573  2.33058395  2.32532685]
[26.06855144 -2.80298876  3.08520373  2.92117488  2.62111158  2.2202473
```

```
   1.97539936  1.47614096  2.33160439  2.24658496]
 [26.19222317 -2.88670902  3.04105087  2.83569368  2.65138212  2.15766438
   2.14126701  1.51850457  2.27948753  2.24619777]
 [26.10778232 -2.8338564   3.07236725  2.84884833  2.63218106  2.23920684
   2.09079432  1.46027975  2.21887473  2.2212264 ]
 [26.10042769 -2.84648869  3.17313411  2.7976957   2.65558481  2.23307957
   2.15711868  1.47859597  2.33122797  2.23023502]
 [26.16038091 -2.85430458  3.11375675  2.83366267  2.6476126   2.26492645
   2.01546978  1.42653223  2.27071902  2.19518293]
 [26.17491016 -2.87523712  3.08222468  2.89117319  2.59579339  2.26032934
   2.07334579  1.43197508  2.20918827  2.26905518]
 [26.16878266 -2.8818421   3.12418829  2.92949786  2.61904237  2.20698239
   2.07410541  1.47364677  2.27946477  2.24132657]
 [26.03846836 -2.81347564  3.09540527  2.95009396  2.58585532  2.25430492
   2.10766759  1.49393651  2.28466901  2.3143748 ]], log_prob=[-1229.50458914
-1230.3539562  -1231.44775837 -1230.50930361
 -1234.10096179 -1233.46810654 -1230.47316746 -1232.94167108
 -1233.76090254 -1229.50654771 -1228.00849126 -1230.56457133
 -1233.58285308 -1229.94562778 -1233.39863951 -1230.45502193
 -1227.76316072 -1232.03400769 -1228.5394738  -1229.64916144
 -1230.27356806 -1231.20035712 -1229.60561405 -1232.29104405
 -1230.69276437 -1232.55977799 -1235.55455442 -1230.08347952
 -1228.15119368 -1231.24273856 -1232.24645221 -1232.64020528
 -1230.35826048 -1229.3298551  -1228.67093249 -1232.0585549
 -1230.35700202 -1232.79638284 -1228.32933081 -1231.44826581
 -1229.1489665  -1231.56856181 -1227.75004669 -1228.43825831
 -1235.86393973 -1229.5406043  -1230.93838812 -1230.15866117
 -1229.63839552 -1232.07644215 -1229.19527168 -1231.22761681
 -1231.21901026 -1227.72107329 -1231.5604973  -1230.51439637
 -1230.07581155 -1231.1850292  -1232.4132838  -1231.16636271
 -1231.84395194 -1227.99889909 -1231.23341684 -1235.03566037
 -1230.72571063 -1227.82899897 -1234.11169688 -1232.77461066
 -1229.84970337 -1229.54665406 -1233.25839065 -1231.44909688
 -1231.25047685 -1229.88813661 -1230.40922688 -1233.20028909
 -1232.44519882 -1230.18349137 -1229.63625826 -1229.83074754
 -1228.79210048 -1230.81590966 -1230.74589023 -1234.86788821
 -1231.91210714 -1229.91803666 -1229.72363249 -1232.12208101
 -1233.15006188 -1229.92273282 -1234.03242433 -1229.73720539
 -1230.11739919 -1231.39655638 -1227.95368364 -1231.43246659
 -1231.35568309 -1231.19049211 -1228.7355225  -1234.4524153 ], blobs=None,
random_state=('MT19937', array([ 821036140,  142430049, 3050514764,  718617570,
200405673,
       2173571599, 3507221201,  582315377, 1193031990, 2771464327,
       2657706946, 3942695621, 1445182873,   12061494, 1589247656,
       1420134667, 1125911641, 2056923133,  460238790, 2610657582,
       1826841191,  358942456,  411841361, 3316023546, 2373314018,
       3314182780, 1195455893, 3612674678,  236052453, 3341201312,
```

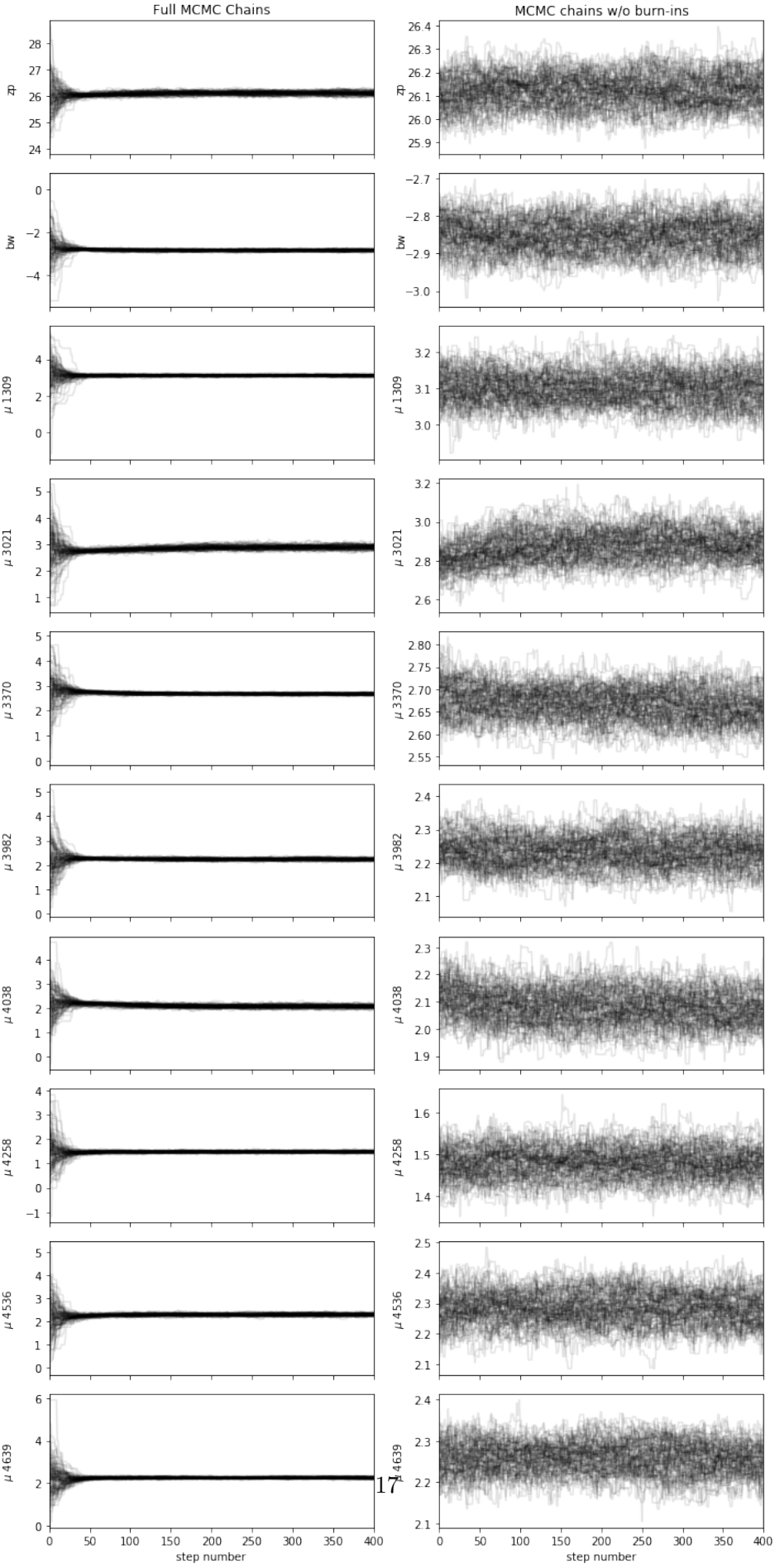
1318248229, 1954888975, 3379705681, 1557424781, 763010744,
3044852397, 3709294909, 774261078, 1171821098, 1920262461,
2981116570, 457070454, 3285141596, 1344887031, 1381967905,
2879856749, 164116729, 3056226702, 4004275104, 147260640,
1764499708, 2023724116,   3438442, 1413384669, 1401912611,
1701716331, 3971653062, 454045607, 3522393925, 208243453,
3146622164, 3353465681, 2172292423, 4036158363, 1068597026,
3089360344, 2614934126, 1132503807, 2827381679, 4132573613,
1793754579, 3959784962, 280089075, 3853787984, 3858436723,
2262963254, 1102181197, 3998787452, 3120100492, 2906026686,
3594429538, 3709810637, 1887269828, 1820906469, 1447464641,
3615930793, 916766450, 368943224, 3898694615, 2831362493,
4181920536, 1363762294, 933500429, 2583632713, 4205372372,
 228289638, 948611674, 3266746912, 3523391229, 1685693117,
2901483956, 230388818, 3042109727, 2104236112, 1663875142,
2080854019, 607880963, 4200350880, 1022062176, 3948593593,
3928324240, 2646640557, 2724896508, 3358680238, 1672294857,
4184616598, 1302607251, 2322115495,   25822899, 1620192218,
2484431367, 2111960831, 2421314352, 440578577,   49356378,
1323629931, 1126424974, 2490756180, 1706989312, 2924431112,
3893711840, 2688263303, 579638892, 671586411, 3515472999,
1781982398, 3862230425, 3359125041, 3060066773, 1195005836,
1975692964, 1311368328, 1149139647, 2553277974, 2194704020,
 415637976, 2899034434,   59228325, 3450054244, 1046296707,
2740489565, 3218621570, 1997557344, 2737767601, 2761556269,
2505539846, 2945761320, 2231420663, 188913271, 2236143611,
3106494671, 1666811153, 4227682032, 491371397, 3882160684,
2085514065, 3538794314, 3779458192, 1623995940, 390464483,
2019494738, 1898508323, 2049086915, 791335966, 3823244639,
1422822288, 2532821002, 2460795844, 3944723708, 1010386792,
3902681341, 3664078808, 432787790, 407270472, 500338217,
 386928281, 1834843689, 225176102, 3156059856, 1188829735,
2068379459, 631762419, 1590379370,   83639726, 2202700055,
4198458976, 3833584542, 2415774354, 1083740107, 1940857693,
3385881621, 4073940174, 2588769097, 2541824290, 2667316237,
 786148517, 3885763927, 1224215302, 3769127468, 741428536,
3229465096, 859655194, 1807153798, 777502344, 1037414240,
1342516867, 1582001547, 1396362956, 1279978091, 2101581828,
2035835938, 1751129694, 4138260101, 2649821573, 4160192096,
3802690535,   16942685, 4024675765, 3892352564, 3920839955,
1564723412, 3093079197, 730037304, 2007156344, 2109901173,
3624016274, 3820713914, 355271787, 1065360311, 1254337731,
 880289421, 3662323429, 4038043191, 4040755862, 4148359283,
 812900913, 1282955415, 3609752050, 3995677555, 469106959,
 419749326, 865258333, 4191507289, 2628283696, 3308371645,
3413889585, 970275380, 1313804430,   37590108, 385759886,
3239914735, 897323380, 1630368606, 797646186, 2525995408,

 710634241, 974859449, 740833934, 3640510594, 3724260338,
2440157239,   25320801, 1255967510, 2992351925, 688467336,
1822296467, 3532535318, 3372661019, 1639389403, 3546924291,
1754243436, 2958073973, 101124989, 224719536, 948262902,
4205680777, 2761266256, 1311752393, 314902235, 3697804555,
3533958696, 665059818, 3094688417, 1886727558, 3422153687,
 904515264, 1585733556, 931151547, 1016533256, 934614785,
1614902495, 2308078102, 4113528815, 3210583781, 3872964449,
 552906283, 346365599,   37050462, 3575078244, 2727136555,
 432372791, 464960514, 2169544186, 3016216447, 269884618,
1267272201, 554151852, 269440584, 2917051436, 165302482,
 958578136,   89607671, 143074788, 2748297736, 1833893278,
1364143382, 1711706006, 1652889277, 2221881617, 2700212691,
2819030129, 2487301697, 2991500372, 3183382445, 4286502551,
2626928958, 3322995646, 2202995845, 3333889186, 4208779433,
2209603984, 3561103504, 1937310282, 2355202187, 3680613075,
3854986695, 3223078261, 4092324436, 1953893576, 247640921,
 566104701, 2051970426, 1121110971, 3771380273, 2444919442,
2087399792, 3110729923, 4152817352, 3219823046, 2601906227,
4166249596, 1714042611, 1593774526, 1318309517, 1644956491,
 750076206, 3137020362, 1759723564, 2883823496, 3047952710,
2553958337, 913413567, 3992168277, 1372978611, 2047056591,
2544796998, 1064577691, 4173090397, 1985876063, 2466803950,
 524258472, 1756910934, 557088553, 2955740090, 219050227,
3252568915, 870224408, 864407321,   57979364, 572625417,
 410384469, 938861827, 2100663431, 3420527318, 523181071,
2017494085, 1779665394, 539949302, 2944959305, 950174799,
4041082714, 2515673314, 380169515, 4165711468, 1105192951,
3621606460, 1976978705, 3701747091, 3393028316, 3422523746,
3334682554, 2518319060, 1542207343, 4142841355, 3550021406,
 445902402, 1933397597, 2003336387, 1044818092,   18607487,
3798919384, 1412655062, 1103875649, 3639877953, 4229560563,
3637173094, 1879863366, 1680645189, 177960046, 1630980446,
 690292915, 2489416418, 2515695844, 2196508143, 3306955058,
1373445853, 3346496008, 3953564430, 2665494125, 4212881538,
3832302048, 131751928, 2880939096, 3960280484, 2686414631,
2903215373, 463365671, 1227063360, 1632819619, 3347792287,
2650747204, 3416140393, 2919770014, 3029712707, 1050128510,
2777687020, 485819212, 1084853300, 4221205045, 1218568978,
1464037571, 2627154606, 3139240068, 760200030, 938679078,
3112881610, 1744136787, 779106046, 1105009745, 4152826640,
3672150956, 1222818012, 2425233317, 3534194662, 2062292436,
3863395253, 558418043, 1099950038, 1798873208, 1389567883,
1277619240, 1050792116, 3860380087, 4258582530, 2555305211,
1724233565, 4267104541, 3927279591, 1260661553, 1782084890,
2384831706, 592027662, 3245181294, 2673616555, 521503813,
2958221836, 2989370145, 3594987655, 2496556581, 758368744,

```
        116442465,  645196560,  446616592, 1308656578, 2493974857,
       2722028496, 2538218164, 1405189072, 2906177145, 2390420953,
       3294834707, 3760137603, 2017518772, 4061056972, 3594818578,
       3286324358, 1088669139, 2957024419, 3818430211, 1098199027,
         56946117,  667080473, 3950924068,  729431162, 1660861859,
       4202036523, 3298252066, 3193847971, 3827592171, 2777898816,
       2789892495, 3842723798, 3812135041, 3148761941, 1623151967,
       1325711822, 2347240869, 1722179677, 2328832606, 3209355533,
       4144146477, 2636201336, 3661388050, 2962149176,  448421709,
       3137261257, 1056831874, 1483433360, 2654204070, 1350821459,
       2770350925, 3942484969, 2986891462, 1021089838, 2189041496,
       1086350457, 1909660651, 2019862627, 3009734560, 2842638005,
       3987683469, 3320142111,  121253897, 3430706241, 1860426754,
       1518110060, 1829763874, 1332936070, 3486929804, 2747980154,
       2631407053, 2750155332, 4216633946,  675288717,   36577271,
       1974439801, 1706844121, 2000466670,  198490961, 1142979566,
       1049088664, 3044806298, 3889486497, 4289141917, 2675339860,
       3388638944,  377487384, 1107643198,  758083512, 1722006823,
        666870831, 2126543859, 1775368063, 1601036791,  973151320,
        853293090,  197550266, 1675305822, 3987585092, 4045934903,
       1021132487, 3986541072, 2527917512, 1145498720, 2484572243,
       3371529493, 2592874435, 1834609177, 2543505370, 4018316816,
       2535130332, 2099456374,  588777032,  417338778, 4108298931,
       2477810780, 1019748250, 1484504699,  710592241, 2673681744,
       1968841288, 3956206693, 2708651997, 1919650756], dtype=uint32), 350, 0,
0.0))
```

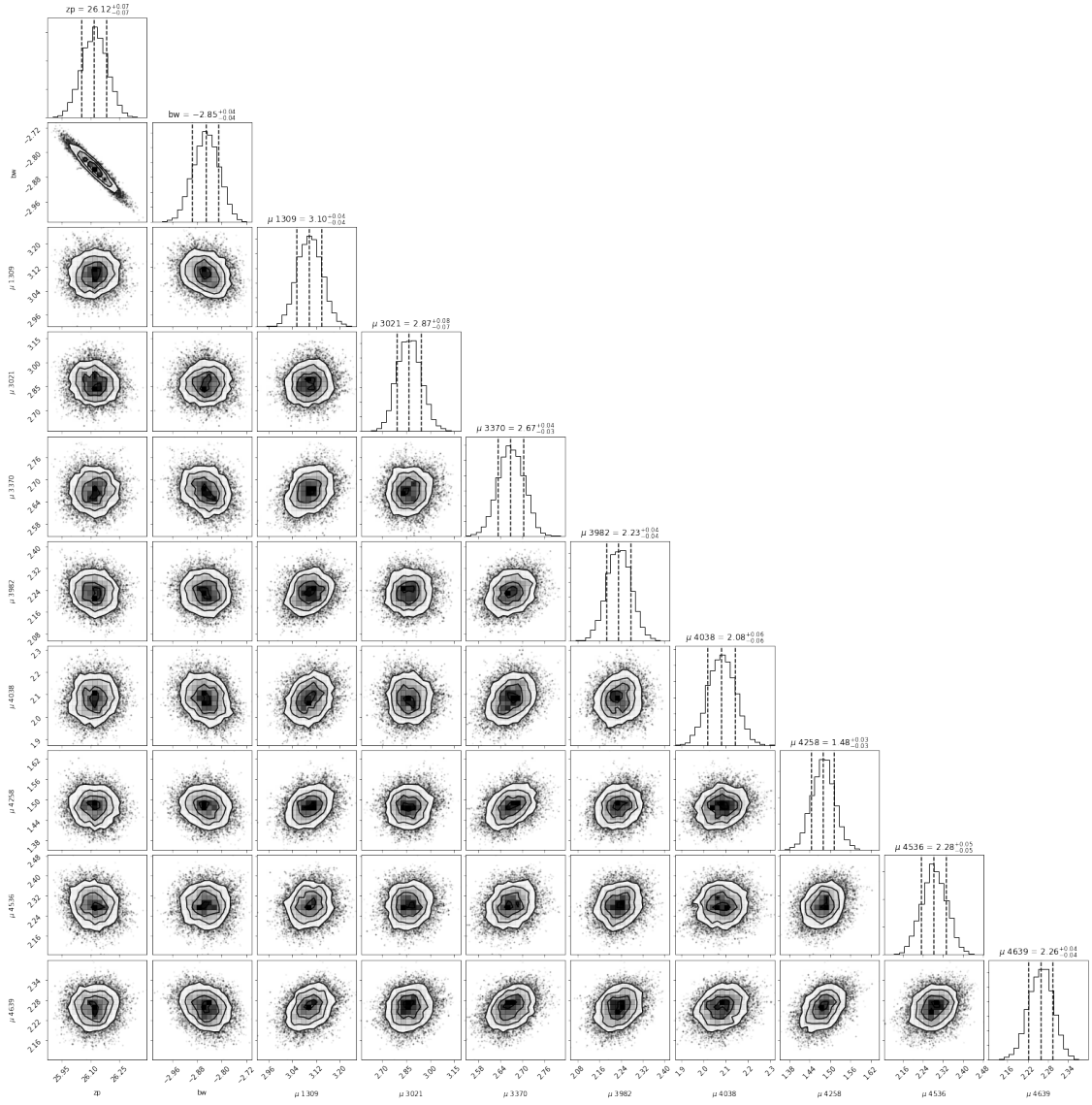
[22]:
```
fig, axes = plt.subplots(nrows=ndim,ncols=2, figsize=(10,20), sharex=True)
samples = sampler.get_chain()
labels = ['zp', 'bw', r'$\mu$ 1309', r'$\mu$ 3021', r'$\mu$ 3370' ,r'$\mu$␣
↪3982', r'$\mu$ 4038', r'$\mu$ 4258' ,r'$\mu$ 4536', r'$\mu$ 4639', r'$\mu$␣
↪5584']

for i in range(ndim):
    ax = axes[i,0]
    ax.plot(samples[:, :, i], "k", alpha=0.1)
    ax.set_xlim(0, len(samples))
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)
    if i ==0: ax.set_title('Full MCMC Chains')

    bx = axes[i,1]
    bx.plot(samples[100:, :, i], "k", alpha=0.1)
    bx.set_xlim(0, len(samples)-100)
    bx.set_ylabel(labels[i])
    bx.yaxis.set_label_coords(-0.1, 0.5)
    if i ==0: bx.set_title('MCMC chains w/o burn-ins')
```

```
axes[ndim-1,0].set_xlabel("step number")
axes[ndim-1,1].set_xlabel("step number")
plt.tight_layout()
```


Full MCMC Chains
MCMC chains w/o burn-ins
zp
bw
μ 1309
μ 3021
μ 3370
μ 3982
μ 4038
μ 4258
μ 4536
μ 4639
step number
step number

[19]:
```
## decide by eye that burn-in is roughly within 100 steps, and discard the␣
 ↪burn-ins
flat_samples = sampler.get_chain(discard=100, flat=True)

fig = corner.corner(flat_samples,labels=labels,quantiles=[0.16, 0.5, 0.84],␣
 ↪show_titles=True, title_kwargs={"fontsize": 12})

for i in range(ndim):
    mcmc = np.percentile(flat_samples[:, i], [16, 50, 84])
    q = np.diff(mcmc)
    print(labels[i], 'Best fit: {:.2f}'.format(mcmc[1]), '{:.2f}'.
 ↪format(-q[0]), '+{:.2f}'.format(q[1]))
```

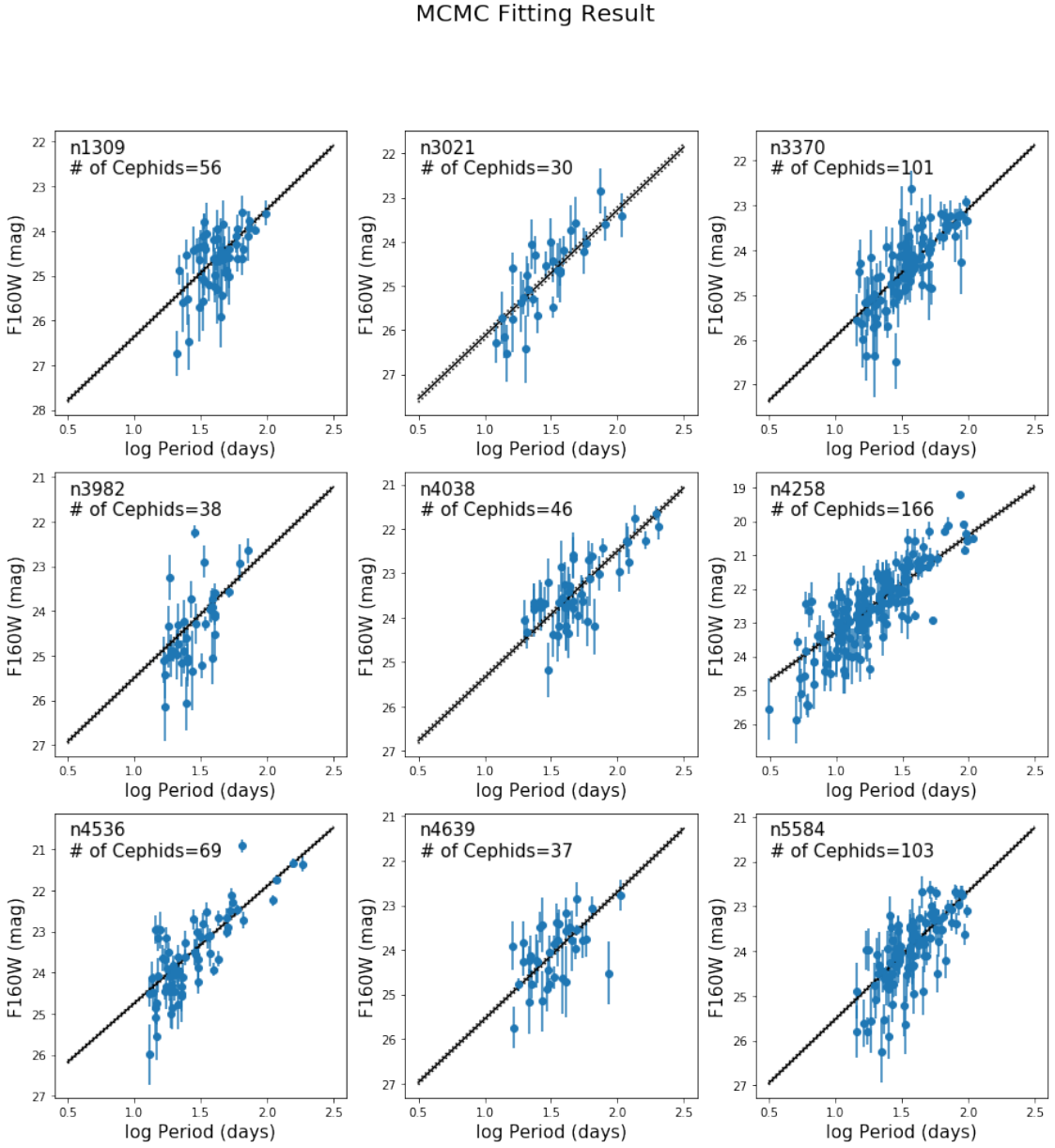
```
zp Best fit: 26.12 -0.07 +0.07
bw Best fit: -2.85 -0.04 +0.04
$\mu$ 1309 Best fit: 3.10 -0.04 +0.04
$\mu$ 3021 Best fit: 2.87 -0.07 +0.08
$\mu$ 3370 Best fit: 2.67 -0.03 +0.04
$\mu$ 3982 Best fit: 2.23 -0.04 +0.04
$\mu$ 4038 Best fit: 2.08 -0.06 +0.06
$\mu$ 4258 Best fit: 1.48 -0.03 +0.03
$\mu$ 4536 Best fit: 2.28 -0.05 +0.05
$\mu$ 4639 Best fit: 2.26 -0.04 +0.04
```



[48]:
```
## plot the results from MCMC

best_fit = np.percentile(flat_samples, 50, axis=0)

fig = plt.figure(figsize=(15, 15))
#ax = fig.add_subplot(111)

plot_logP = np.linspace(0.5,2.5,num=100)
host0 = np.unique(hostid)[np.where(np.unique(hostid)!=4258)]

nplot = 0
for i, ID in enumerate(hosts):
```

```
    ax = fig.add_subplot(331+i)
    ceph.select(ID)
    mobs = ceph.mobs
    logP = ceph.logP
    sigma_obs = ceph.sigma
    ax.errorbar(logP, mobs, yerr=sigma_obs, linestyle='None', marker='o')

    if ID !='4258':
        u0 = best_fit[np.where(float(ID)==host0)[0]+2]
        mcmc = np.percentile(flat_samples[:, np.where(float(ID)==host0)[0]+2],␣
↪[16, 50, 84])
        q = np.diff(mcmc)
    else:
        u0 =0
    ax.plot(plot_logP, PLmodel(plot_logP, best_fit[0], best_fit[1],␣
↪u0),linestyle='-',color='k')
    ax.plot(plot_logP, PLmodel(plot_logP, best_fit[0], best_fit[1],␣
↪u0-q[0]),linestyle=':',color='k')
    ax.plot(plot_logP, PLmodel(plot_logP, best_fit[0], best_fit[1],␣
↪u0+q[1]),linestyle=':',color='k')

    #ax.legend(frameon=False)
    ax.text(0.05,0.85,'n'+ID+'\n# of Cephids='+str(len(mobs)),transform = ax.
↪transAxes,fontsize=15)
    ax.set_xlabel('log Period (days)',fontsize=15)
    ax.set_ylabel('F160W (mag)',fontsize=15)
    plt.gca().invert_yaxis()


plt.suptitle('MCMC Fitting Result',fontsize=20)
```

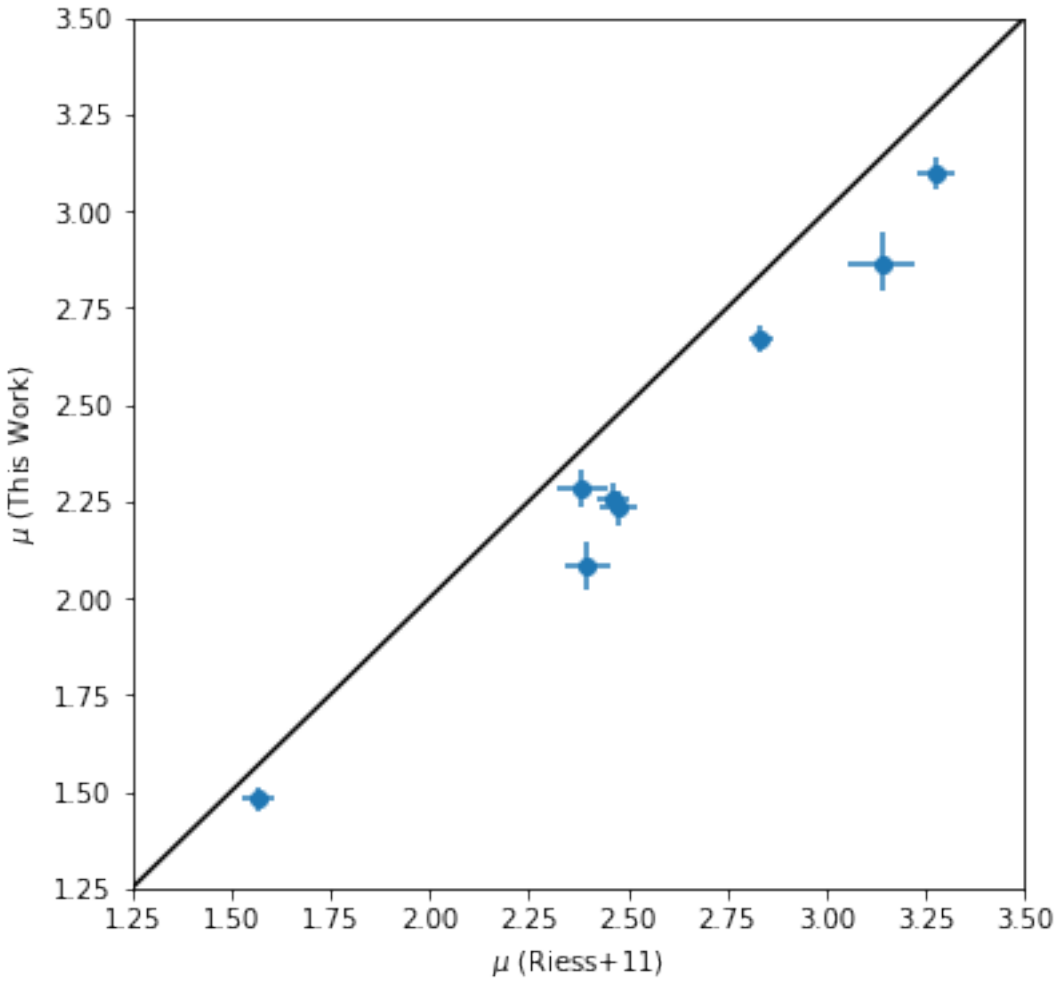
[48]: Text(0.5, 0.98, 'MCMC Fitting Result')



[85]:
```
## compare with Riess Table 3
r11= np.loadtxt('apj383673t3_ascii.txt',usecols=(6),skiprows=4,max_rows=8).T
r11_err= np.loadtxt('apj383673t3_ascii.
 ↪txt',usecols=(7),skiprows=4,max_rows=8,dtype=str).T
r11_err = np.array([float(x[1:-1]) for x in r11_err])

hostname = np.loadtxt('apj383673t3_ascii.
 ↪txt',usecols=(0),skiprows=4,max_rows=8,dtype=str).T
hostname = np.array([x.split('n')[1] for x in hostname],dtype=int)
```

```
r11 = r11[np.argsort(hostname)]
r11_err = r11_err[np.argsort(hostname)]

u0_err = np.zeros((2,8))
for i in range(2,ndim):
    mcmc = np.percentile(flat_samples[:, i], [16, 50, 84])
    q = np.diff(mcmc)
    u0_err[0,i-2] = q[0]
    u0_err[1,i-2] = q[1]

plt.figure(figsize=(6,6))
plt.errorbar(r11,best_fit[2:],xerr=r11_err,yerr=u0_err, fmt='o')
plt.plot([1.25,3.5],[1.25,3.5],'k-')
plt.xlim([1.25,3.5])
plt.ylim([1.25,3.5])
plt.xlabel('$\mu$ (Riess+11)')
plt.ylabel('$\mu$ (This Work)')
```

[85]: Text(0, 0.5, '$\\mu$ (This Work)')



Compared to the Riess+11 results, our fitted distances to each galaxy are smaller, this might be because we ignored the metallicity term.

5 Q4.

5.1 Riess et al. do a *simultaneous* fit to the Cepheids and SNe data (Table 3 of their paper).

5.2 I'm slightly less cruel, so you can take your intercepts $\mu_{0,i} - \mu_{0,4258}$ and the re-express equation 4 using a substitution of equation 3.

5.3 Write that expression down.

Equation (3) in Riess et al (2011):

$$m^0_{v,i} = (\mu_{0,i} - \mu_{0,4258}) + m^0_{v,4258}$$

Equation (4) in Riess et al (2011):

$$log(H_0) = \frac{(m^0_{v,4258} - \mu_{0,4258}) + 5a_v + 25}{5}$$

From Equation (3), we have:

$$m^0_{v,4258} = m^0_{v,i} - (\mu_{0,i} - \mu_{0,4258})$$

And Equation (4) becomes

$$log(H_0) = \frac{(m^0_{v,i} - (\mu_{0,i} - \mu_{0,4258}) - \mu_{0,4258}) + 5a_v + 25}{5}$$

where we can populate $m^0_{v,i} + 5a_v$ from Riess et al (2011) Table 3,

use the best-fit results for $(\mu_{0,i} - \mu_{0,4258})$,

and $\mu_{0,4258}$ from Humphreys et al (2013), to estimate H_0.

6 Q5

6.1 Using the supernova data in Table 3 (enter it into whatever data structure you deem best) and the distance modulus to NGC 4258 from Humphreys et al., 2013 and using the simple sample statistics you learned in Week 1, estimate the Hubble constant.

[23]:
```
hostname = np.loadtxt('apj383673t3_ascii.
 ↪txt',usecols=(0),skiprows=4,max_rows=8,dtype=str).T
m_vi = np.loadtxt('apj383673t3_ascii.txt',usecols=(4),skiprows=4,max_rows=8).T

hostname = np.array([x.split('n')[1] for x in hostname],dtype=int)
m_vi = m_vi[np.argsort(hostname)]

ui = best_fit[2:]

u4258 = 5.*np.log10(7.6e6)+5.

H0 = 10**((m_vi-ui-u4258+25.)/5.)

## print out H0 results from each galaxy
print(H0)
```

```
[0.71337227 0.76897408 0.78443512 0.73010141 0.76412028 0.71204572
 0.74221315 0.83596521]
```

[24]:
```
## simple mean and median calculations

print('Simple Mean','{:.3f}'.format(np.mean(H0)),', and standard deviation','{:.
↪3f}'.format(np.std(H0)))
print('Simple Median','{:.3f}'.format(np.median(H0)),', 16 percentile','{:.3f}␣
↪(-{:.3f})'.format(np.percentile(H0,16),np.median(H0)-np.percentile(H0,16))␣
↪,',84 percentile','{:.3f} (+{:.3f})'.format(np.percentile(H0,84),np.
↪percentile(H0,84)-np.median(H0)))
```

```
Simple Mean 0.756 , and standard deviation 0.039
Simple Median 0.753 , 16 percentile 0.715 (-0.038) ,84 percentile 0.783 (+0.029)
```

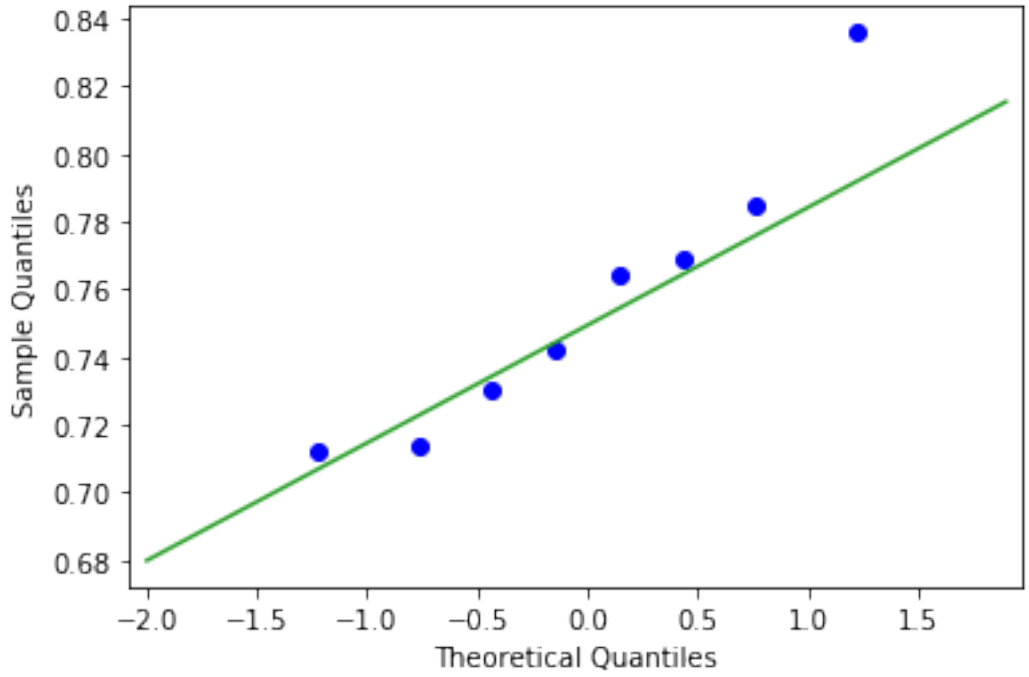
[25]:
```
## mean and median after sigma clipping
## this is not good for this dataset because there are not a lot of data points
## and all points are within 2 sigma

H0_clip, c1, c2 = st.sigmaclip(H0, low=2.0, high=2.0)

print('After 2-sigma clipping: Mean','{:.3f}'.format(np.mean(H0_clip)),', and␣
↪standard deviation','{:.3f}'.format(np.std(H0_clip)))
print('After 2-sigma clipping: Median','{:.3f}'.format(np.median(H0_clip)),',␣
↪16 percentile','{:.3f} (-{:.3f})'.format(np.percentile(H0_clip,16),np.
↪median(H0_clip)-np.percentile(H0_clip,16)) ,',84 percentile','{:.3f} (+{:.
↪3f})'.format(np.percentile(H0_clip,84),np.percentile(H0_clip,84)-np.
↪median(H0_clip)))
```

```
After 2-sigma clipping: Mean 0.745 , and standard deviation 0.026
After 2-sigma clipping: Median 0.742 , 16 percentile 0.713 (-0.029) ,84
percentile 0.770 (+0.027)
```

[26]:
```
## using the quantile-quantile plot
## fit a straight line in the qq-plot

import statsmodels.api as sm

qq_25_75 = np.percentile(H0, [25, 75])
x_25 = st.norm.ppf(0.25)
x_75 = st.norm.ppf(0.75)
xvals = np.arange(-2, 2, 0.1)
def iqr_line(xvals):
    return ((xvals - x_25)/(x_75-x_25)) * (qq_25_75[1] - qq_25_75[0] ) +␣
↪qq_25_75[0]

sm.qqplot(H0)
```

```
plt.plot(xvals, iqr_line(xvals), color='C2', label='IQR line')
print('From QQ plot fitting: Median','{:.3f}'.format(iqr_line(0.)),', 1st␣
↪quantile','{:.3f} (-{:.3f})'.format(iqr_line(-1.),iqr_line(0.)-iqr_line(-1.
↪)) ,',3rd quantile','{:.3f} (+{:.3f})'.format(iqr_line(1.),iqr_line(1.
↪)-iqr_line(0.)))
```

```
From QQ plot fitting: Median 0.749 , 1st quantile 0.715 (-0.035) ,3rd quantile
0.784 (+0.035)
```



[ ]:

[ ]: