In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import piccard as pic
```

Assume we have a pulsar (J0030 from the first IPTA Mock Data Challenge, open1, here), and read it in an HDF5 file

In [8]:
```
# Creating hdf5 files goes through the DataFile class
t2df = pic.DataFile('J0030.h5')
t2df.addpulsar('mdc1-open1/J0030+0451.par', 'mdc1-open1/J0030+0451.tim')
```

In [12]:
```
# With a datafile, we can construct a 'likelihood' object, which is the central class of the package
likob = pic.ptaLikelihood('J0030.h5')
```

In [13]:
```
# A model for the data (all data in the hdf5, so can be multi-pulsar) can be made with the initModel member func
# Likelihood function 'mark3' is the standard for red noise analysis, and will default to that if not given
likob.initModel(nfreqmodes=30, varyEfac=True, incRedNoise=True, noiseModel='powerlaw', likfunc='mark3')
```

In [19]:
```
# The model has the following variable dimensions.
print "Number of dimensions: ", likob.dimensions
# A total dictionary of all parameters is kept in likob.pardes (I'll make the names more descriptive). Do
# not edit the pardes dictionary yourself. The 'real' model is saved in sub-classes and must be edited through
# the initModel function
for i in range(len(likob.pardes)): print likob.pardes[i]
```

```
Number of dimensions:  3
{'index': 0, 'name': 'pulsarname', 'sigtype': 'efac', 'sigindex': 0, 'correlation': 'single', 'pulsar': 0,
'id': u'efacJ0030+0451'}
{'index': 1, 'name': 'powerlaw', 'sigtype': 'powerlaw', 'sigindex': 1, 'correlation': 'single', 'pulsar': 0,
'id': 'RN-Amplitude'}
{'index': 2, 'name': 'powerlaw', 'sigtype': 'powerlaw', 'sigindex': 1, 'correlation': 'single', 'pulsar': 0,
'id': 'RN-spectral-index'}
{'index': -1, 'name': 'powerlaw', 'sigtype': 'powerlaw', 'sigindex': 1, 'correlation': 'single', 'pulsar':
0, 'id': 'low-frequency-cutoff'}
```

In [20]:
```
# The default start parameters, minimum value, maximum value, and stepsize
# Parameter indices belong to the 'index' key of the likob.pardes dictionary.
# A '-1' in the pardes dictionary means that the parameter is kept fixed in a
# subsequent analysis.
print likob.pstart, likob.pmin, likob.pmax, likob.pwidth
```

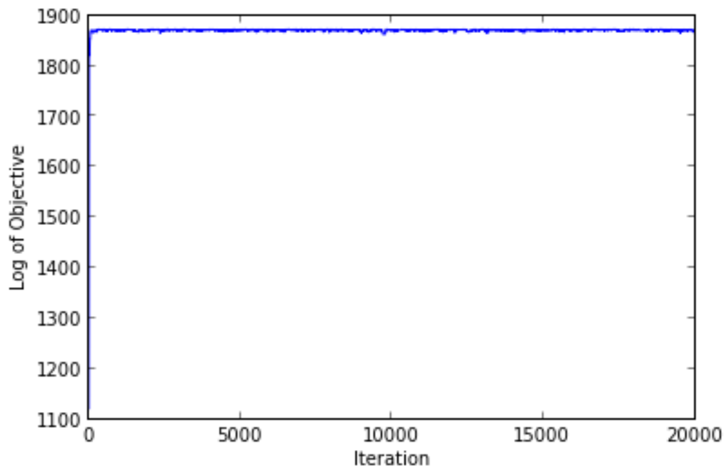
```
[   1.    -14.      2.01] [  0.001 -16.       1.02 ] [ 1000.       -5.        6.98] [ 0.1  0.1  0.1]
```

In [23]:
```
# Calculate the value of the posterior, likelihood, or prior:
print likob.logprior(likob.pstart), likob.loglikelihood(likob.pstart), likob.logposterior(likob.pstart)
```

```
-11.0907200329 1132.24124175 1121.15052171
```

In [22]:
```
# Run a t-walk algorithm on the posterior for 20000 steps
pic.Runtwalk(likob, 20000, 'J0030-twalk-burnin.txt', thin=5, analyse=True)
```

```
pytwalk: Running the twalk with 20000 iterations. Wed, 07 Aug 2013, 12:39.
       Finish in approx. 17 seconds.
pytwalk: finished, Wed, 07 Aug 2013, 12:39:30.
Acceptance rates for the Walk, Traverse, Blow and Hop kernels:[ 0.45564063  0.16266099  0.20359281
0.01129944]
Global acceptance rate: 0.30515
AutoMaxlag: maxlag= 57.
Integrated Autocorrelation Time:    12.2, IAT/n:     4.1
```



In [24]:

```
# Removing the burn-in of the MCMC chains is easy on the command line
!sed '1,300d' J0030-twalk-burnin.txt > J0030-twalk.txt
```

In [28]:

```
# We can run a more optimal Metropolis algorithm by using the t-walk as a tuning chain
pic.RunMetropolis(likob, 20000, 'J0030-metro.txt', initfile='J0030-twalk.txt', resize=1.0)
```

```
Obtaining initial positions from 'J0030-twalk.txt'
Running Metropolis-Hastings sampler
Sample: 19900 = 99.5%   acc. fr. = 0.432028   pos = -1.322027e+01 4.794301e+00  lnprob = 1.871749e+03
('Mean acceptance fraction:', 0.43202839858007097)
('Autocorrelation time:', 23.371900797369577)
```

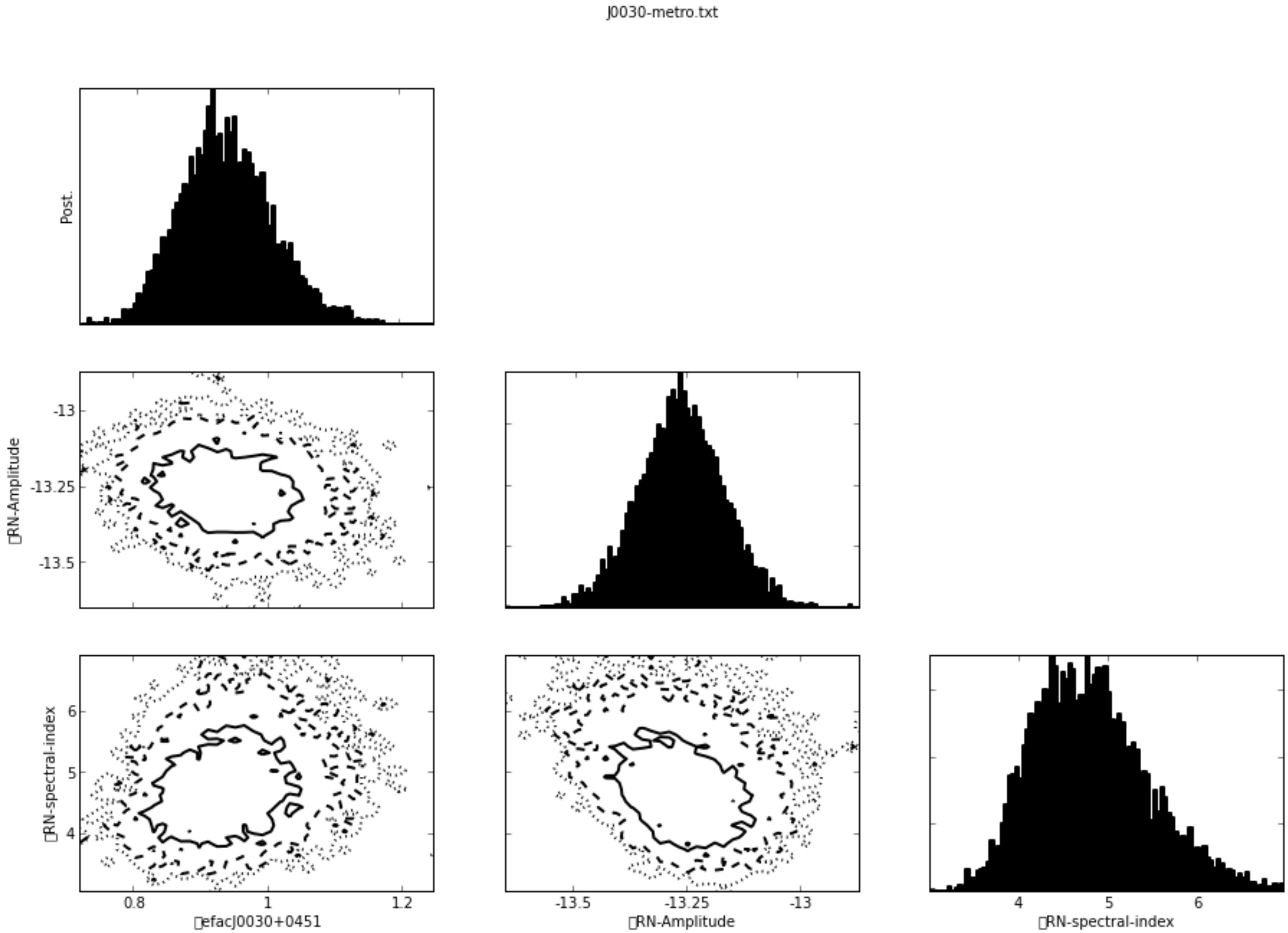
In [29]:

```
# The parameters in the chain are stored in 'J0030-metro.txt.parameters.txt'.
# All sampler routines use this convention
!cat J0030-metro.txt.parameters.txt
```

```
0       0       efac    single  pulsarname      efacJ0030+0451
1       0       powerlaw        single  powerlaw        RN-Amplitude
2       0       powerlaw        single  powerlaw        RN-spectral-index
-1      0       powerlaw        single  powerlaw        low-frequency-cutoff
```

In [31]:
```
# Plot the results of the MCMC (which uses the .parameters.txt file)
pic.triplot('J0030-metro.txt')
```

```
parametersfilename =  J0030-metro.txt.parameters.txt
figurefilename =  J0030-metro.txt.fig.eps
chainfilename =  J0030-metro.txt
```



In [ ]:

In [ ]:

In [ ]: