Institute for
Health Metrics
and Evaluation

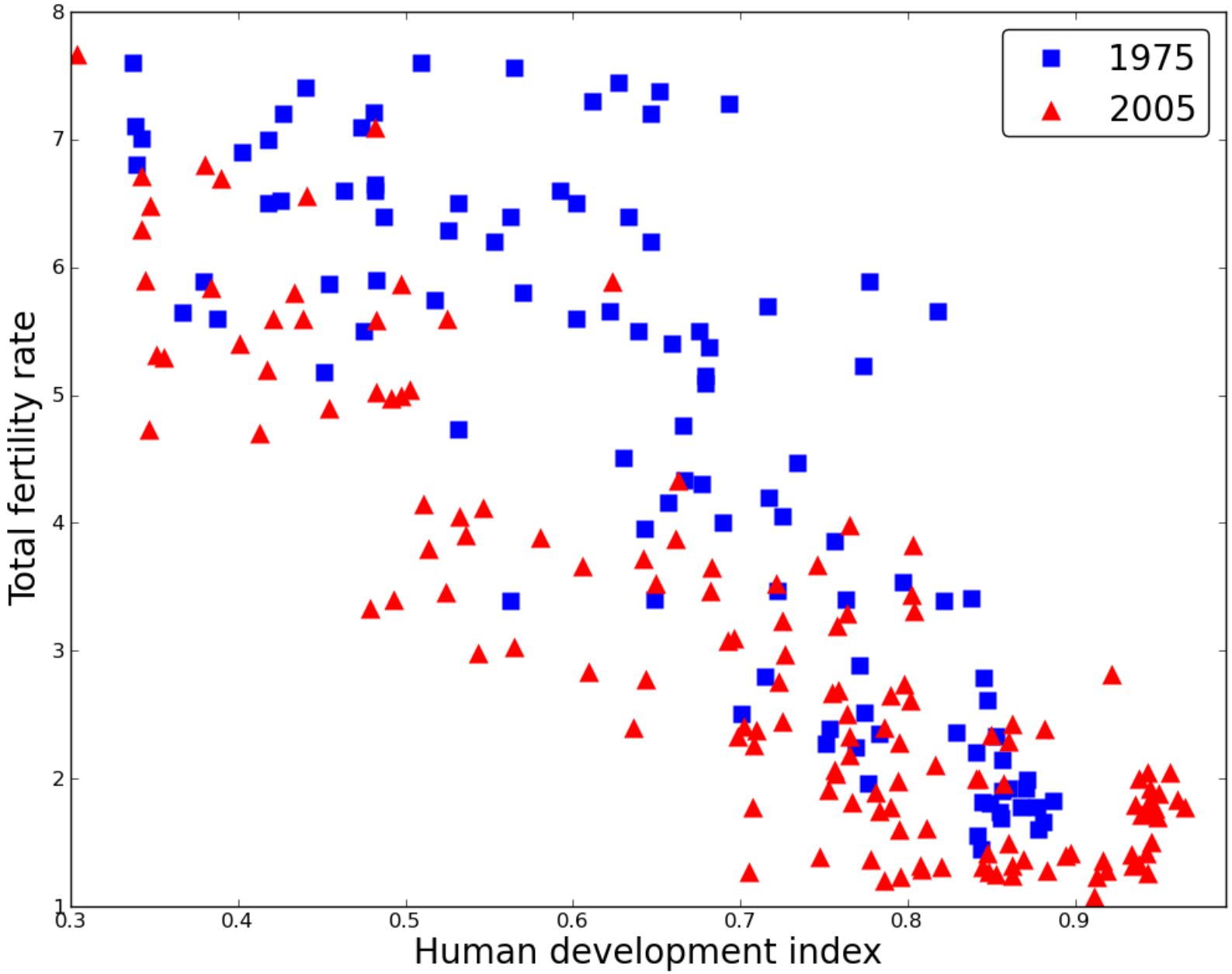
PyMC by Example: Human Fertility

SciPy 2011

Abraham D Flaxman

Institute for Health Metrics and Evaluation

UNIVERSITY OF WASHINGTON


1975
2005
Total fertility rate
Human development index
8
7
6
5
4
3
2
1
0.3
0.4
0.5
0.6
0.7
0.8
0.9

nature International weekly journal of science

Search this

Access

To read this story in full you will need to login or make a payment (see right).

nature.com > Journal home > Table of Contents

Letter

Nature **460**, 741-743 (6 August 2009) | doi:10.1038/nature08230; Received 1 April 2009; Accepted 17 June 2009

Advances in development reverse fertility declines

Mikko Myrskylä[1], Hans-Peter Kohler[1] & Francesco C. Billari[2]

1. Population Studies Center, University of Pennsylvania, 3718 Locust Walk, Philadelphia, Pennsylvania 19104, USA
2. DONDENA "Carlo F. Dondena" Centre for Research on Social Dynamics, Department of Decision Sciences and IGIER, Università Bocconi, via Röntgen 1, 20136 Milan, Italy

Correspondence to: Hans-Peter Kohler[1] Correspondence and requests for materials should be addressed to H.-P.K. (Email: hpkohler@pop.upenn.edu).

Top

During the twentieth century, the global population has gone through unprecedented increases in economic and social development that coincided with substantial declines in human fertility and population growth rates[1, 2]. The negative

ARTICLE LINKS

- Figures and tables
- Supplementary info

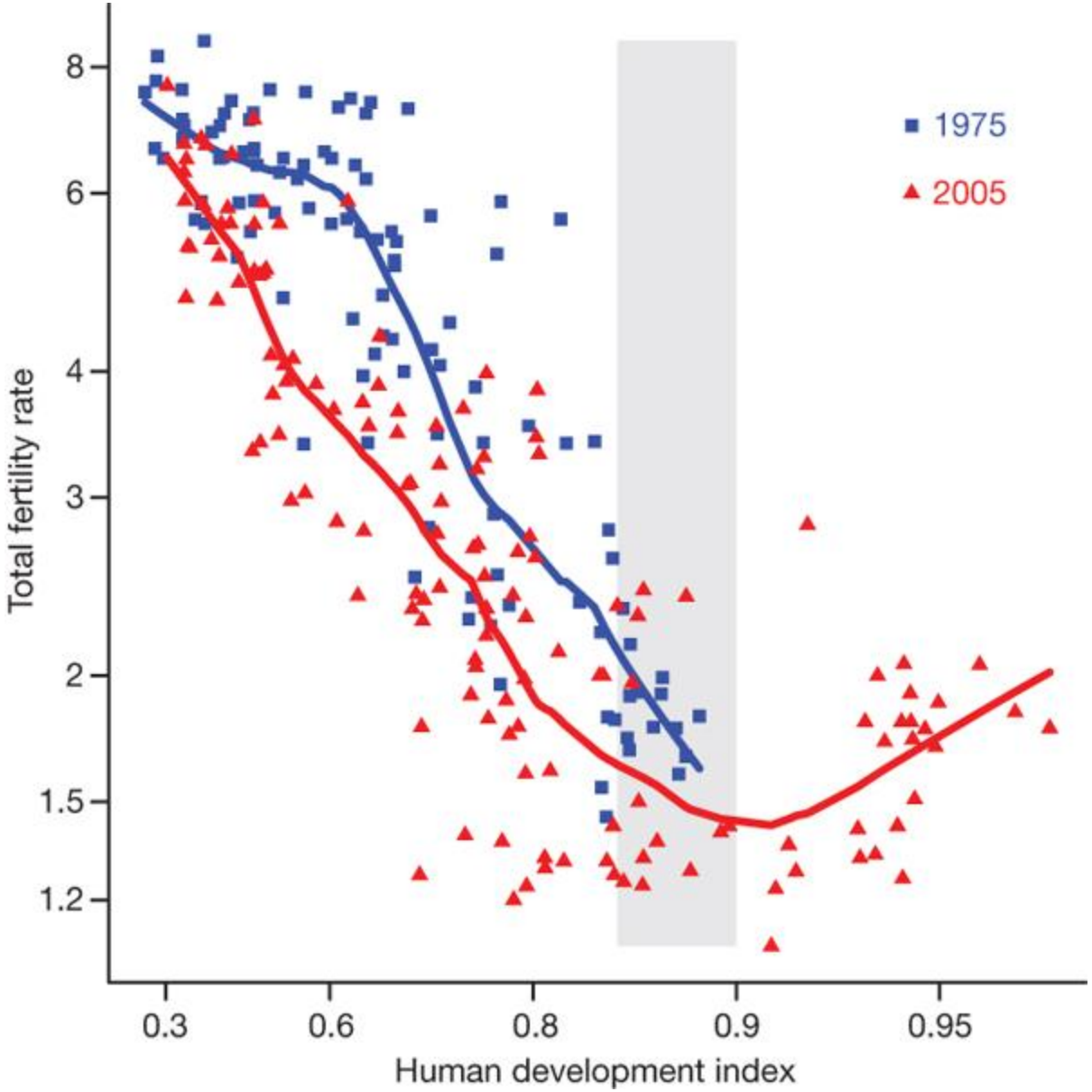
SEE ALSO

- News and Views by Tuljapurkar
- Editor's Summary

ARTICLE TOOLS

- Send to a friend
- Export citation
- Export references
- Rights and permissions
- Order commercial reprints
- Bookmark in Connotea

Cross-sectional relationship between TFR and HDI in 1975 and 2005.



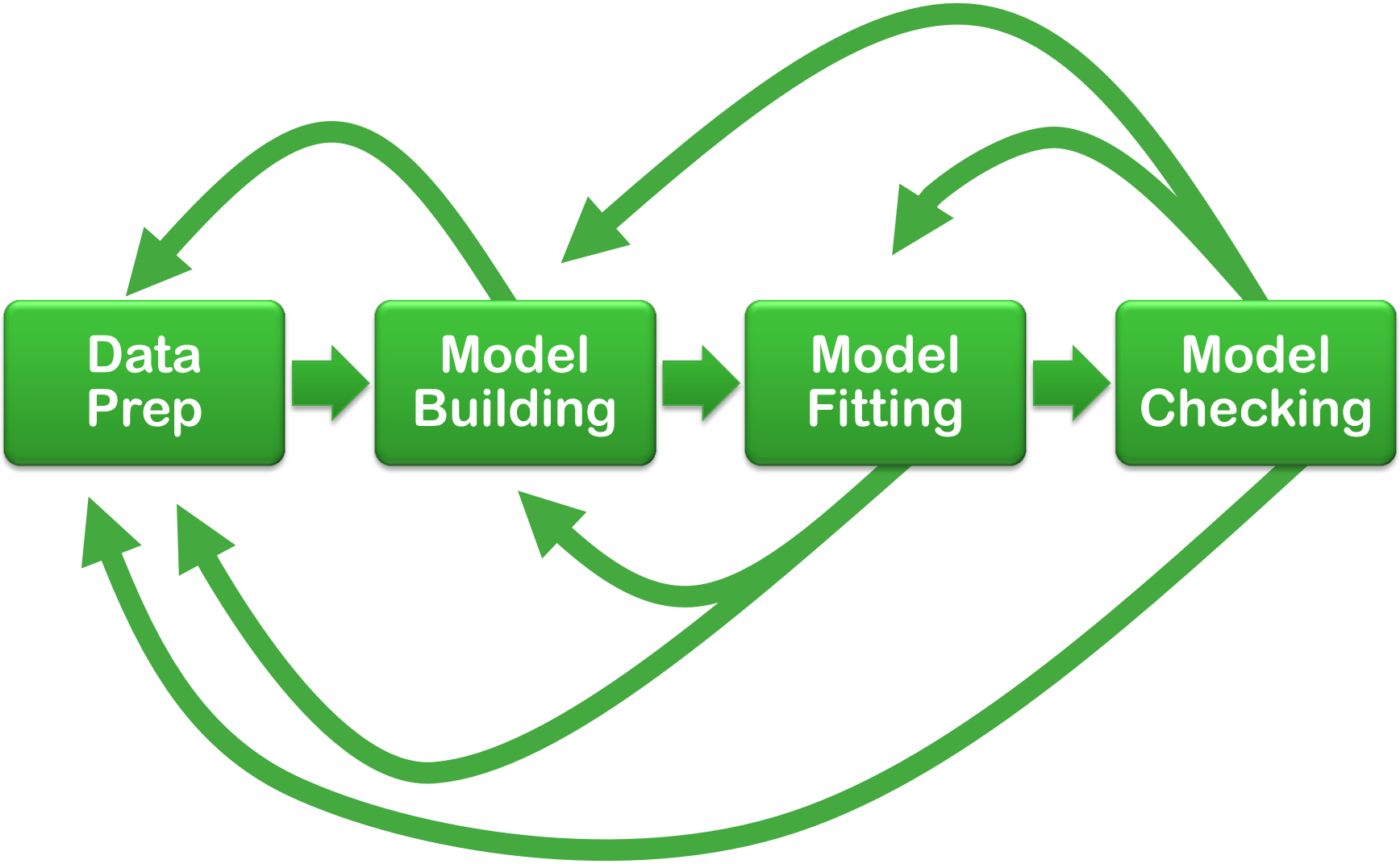
M Myrskylä *et al. Nature* **460**, 741-743 (2009) doi:10.1038/nature08230

nature

Learning Objective

Know enough PyMC to check this out

A Generic Template for Bayesian data analysis



pymc-project-template / src

name	age	mess
..		
data.py	May 24, 2011	init
graphics.py	May 24, 2011	init
models.py	May 24, 2011	init
tests.py	May 24, 2011	init

The Data

100755 | 145 lines (144 sloc) | 94.991 kb

```
country,HDI.1975,HDI.1976,HDI.1977,HDI.1978,HDI.1979,HDI.1980,
Albania,,,,,,0.731273056,0.734932246,0.737145554,0.738451574,0
Algeria,0.565066871,0.57096611,0.575103358,0.581994626,0.58797
Angola,0.427032369,0.427252059,0.4274633,0.428155305,0.428839
Argentina,0.796895676,0.796294282,0.800532019,0.798749194,0.80
Armenia,0.700787712,0.702149028,0.703451552,0.705567048,0.7096
Australia,0.856418407,0.857742049,0.859507393,0.86295765,0.865
Austria,0.844760239,0.849874426,0.854348125,0.854713814,0.8594
Azerbaijan,,,,,,,,,,,,,,,0.770722523,0.76666317,0.745701042,0
Bahrain,0.773655346,0.774279264,0.774879186,0.779222144,0.7835
Bangladesh,0.387413871,0.389184115,0.389373185,0.395334908,0.4
Belarus,,,,,,,,,,,,,,,0.7830714,0.780204413,0.773003586,0.763
Belgium,0.854899715,0.857868123,0.858153017,0.861384937,0.8643
Belize,0.646159734,0.644969023,0.647105514,0.651537112,0.65573
Benin,0.338790582,0.337774288,0.338946694,0.339810674,0.343432
Bolivia,0.570293598,0.571454562,0.572813625,0.576896988,0.5799
```

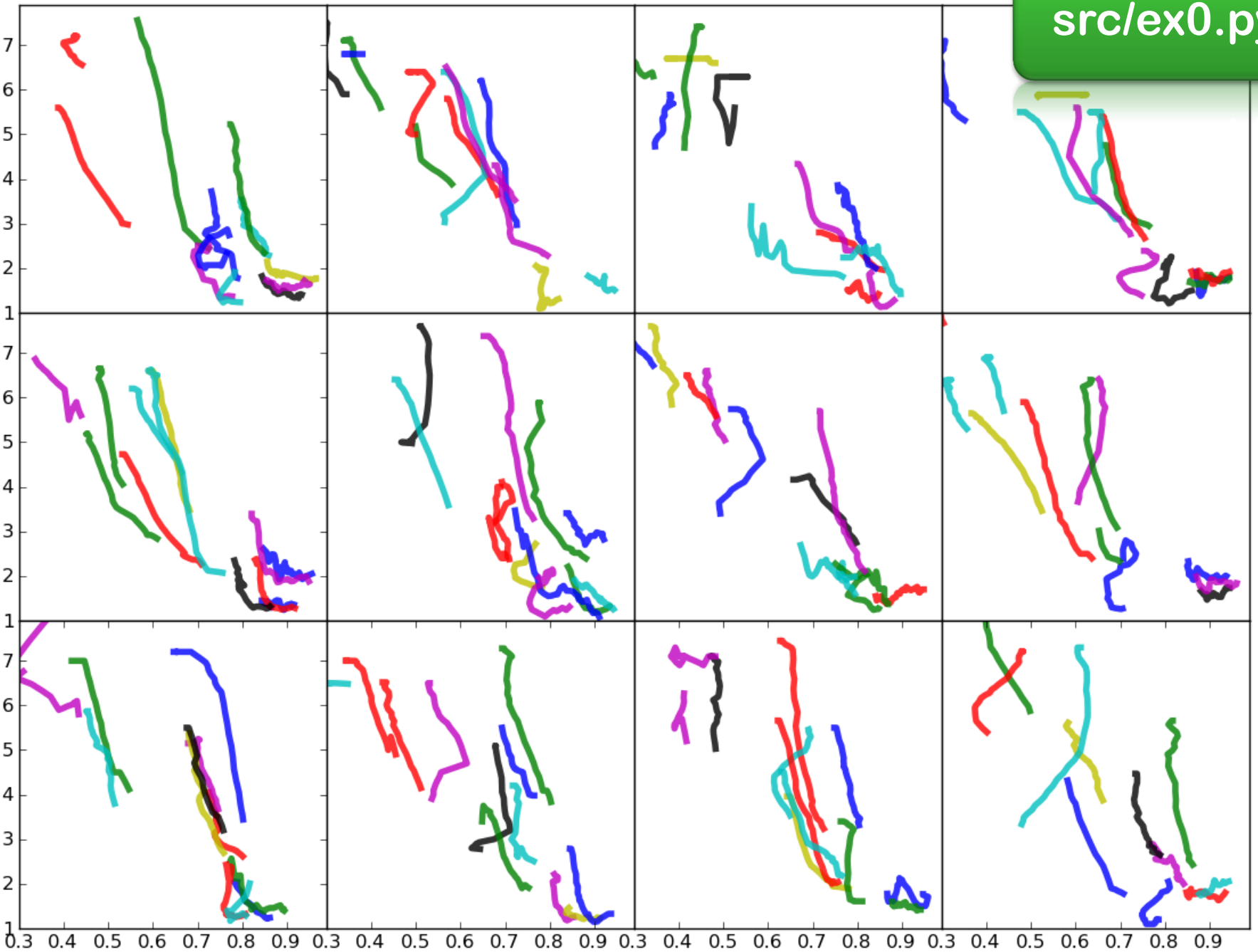
src/data.py

```
import pylab as pl

orig = pl.csv2rec('nature08230-s2.csv')

country = []; year = []; hdi = []; tfr = []
for row in orig:
    for y in range(1975, 2006):
        if pl.isnan(row['hdi%d'%y]) \
                or pl.isnan(row['tfr%d'%y]):
            continue
        country.append(row['country'])
        year.append(y)
        hdi.append(row['hdi%d'%y])
        tfr.append(row['tfr%d'%y])

all = pl.np.core.rec.fromarrays([country, year, hdi, tfr],
                                names=['country’,‘year’,‘hdi’,‘tfr‘])

hdi2005 = all.hdi[all.year==2005]
tfr2005 = all.tfr[all.year==2005]
```

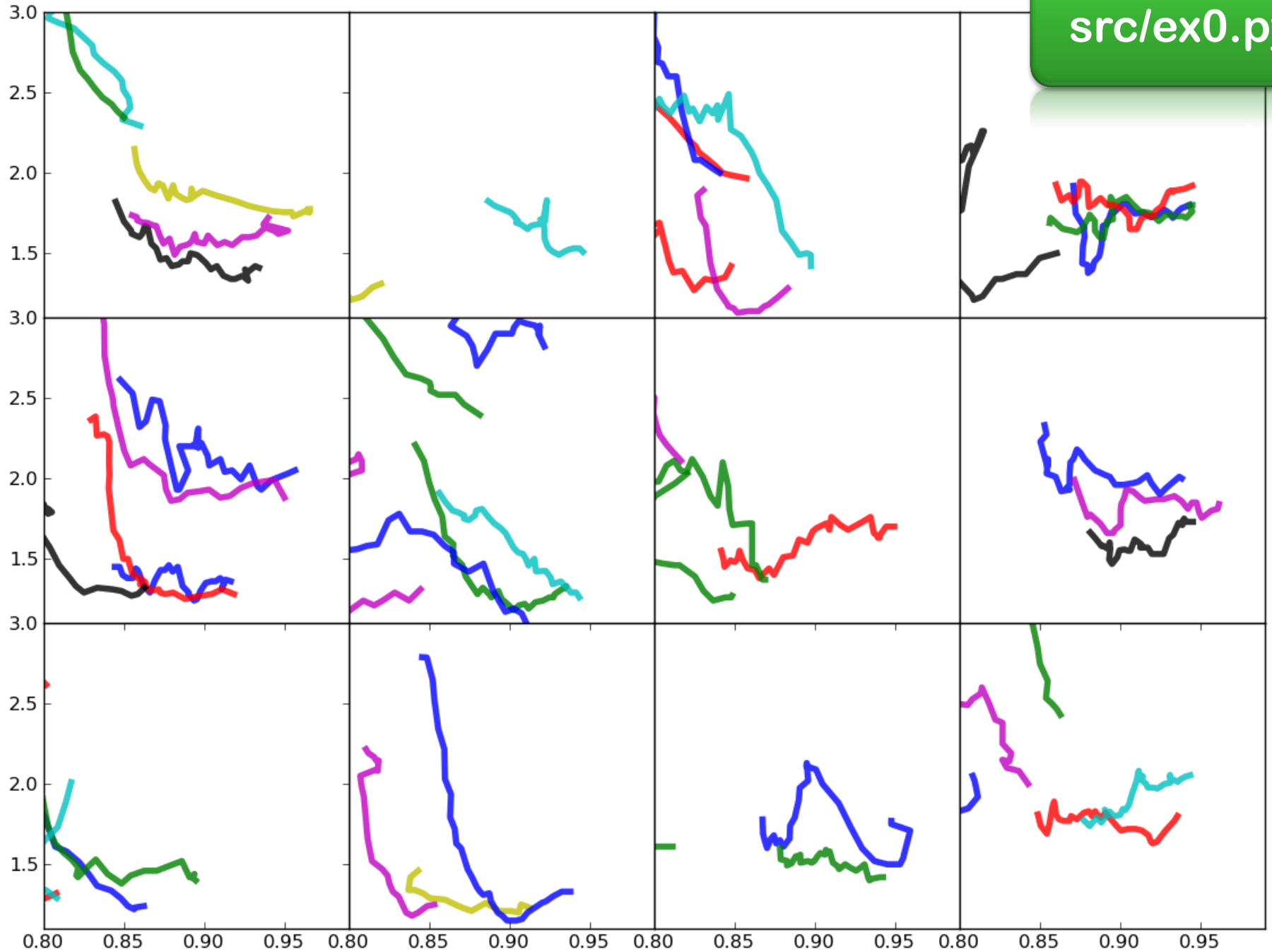
src/graphics.py

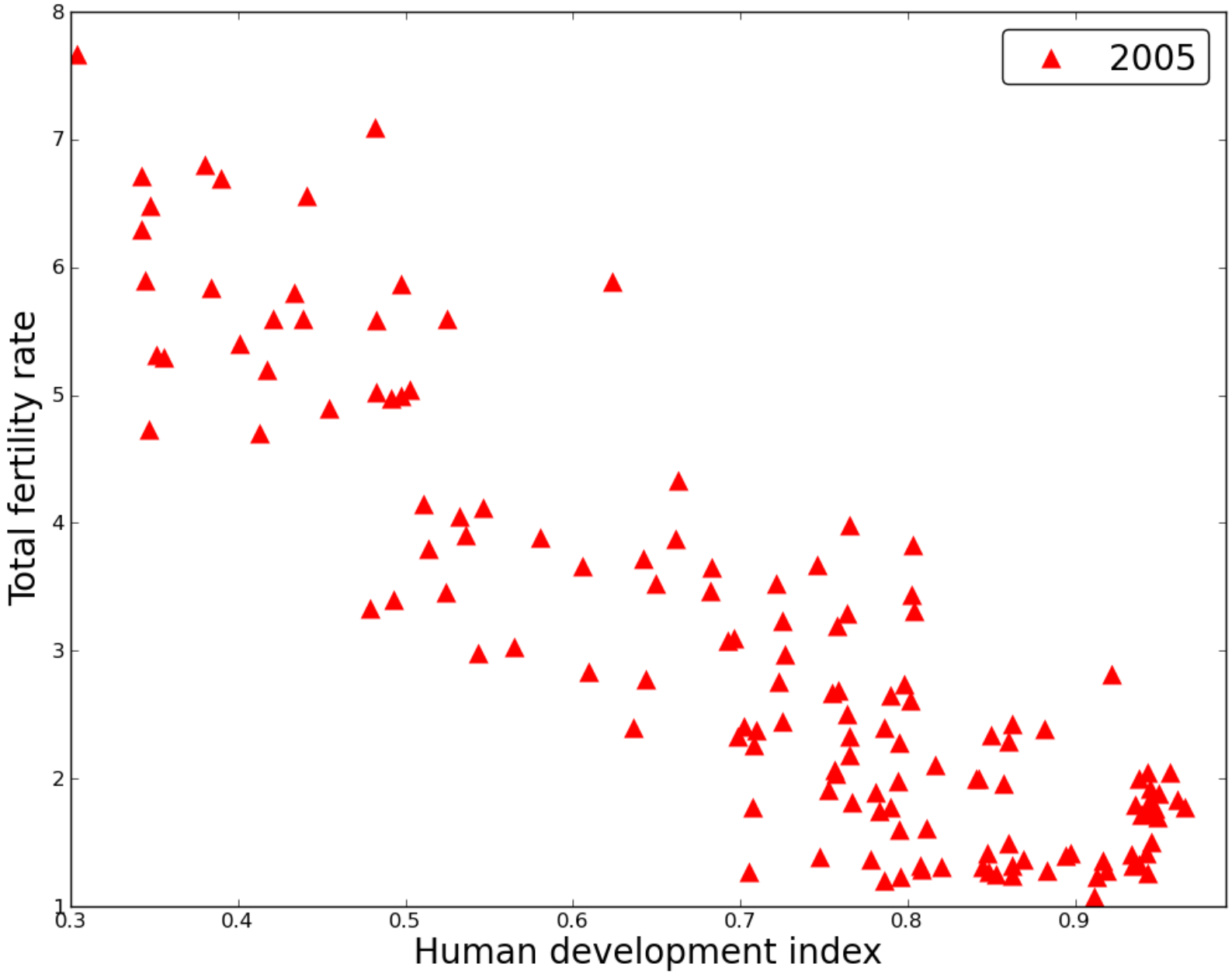
```
def plot_each_country(axis_bounds):
    years = range(1975, 2006)
    for i, c in enumerate(pl.unique(data.all.country)):
        pl.subplot(3, 4, i/12+1)
        pl.plot(data.all.hdi[data.all.country==c],
                data.all.tfr[data.all.country==c],
                linewidth=4, alpha=.8)
        pl.axis(axis_bounds)

    for r in range(3):
        for c in range(4):
            subplot(3, 4, r*4+c+1)
            if r != 2:
                pl.xticks([])
            if c != 0:
                pl.yticks([])

    pl.subplots_adjust(.05, .05, .95, .95, 0, 0)
```

src/ex0.py



src/ex0.py




2005
Total fertility rate
Human development index
8
7
6
5
4
3
2
1
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Simple model

$$\mathrm{TFR}_i = \beta_0 + \beta_1 \, \mathrm{HDI}_i + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma^2)$$

src/models.py

```
def linear():
    beta = mc.Uninformative('beta', value=[0., 0.])
    sigma = mc.Uniform('sigma', lower=0, upper=100, value=1.)

    @mc.deterministic
    def y_mean(beta=beta, X=data.hdi2005):
        return beta[0] + beta[1]*X

    y_obs = mc.Normal('y_obs', value=data.tfr2005,
                      mu=y_mean, tau=sigma**-2,
                      observed=True)

    return vars()
```

```
In [1]: import models

In [2]: vars = models.linear()

In [3]: vars
Out[3]:
{'beta': <pymc.distributions.Uninformative 'beta' at 0xa
 'sigma': <pymc.distributions.Uniform 'sigma' at 0xae9b
 'y_mean': <pymc.PyMCObjects.Deterministic 'y_mean' at 
 'y_obs': <pymc.distributions.Normal 'y_obs' at 0xae9b6
```

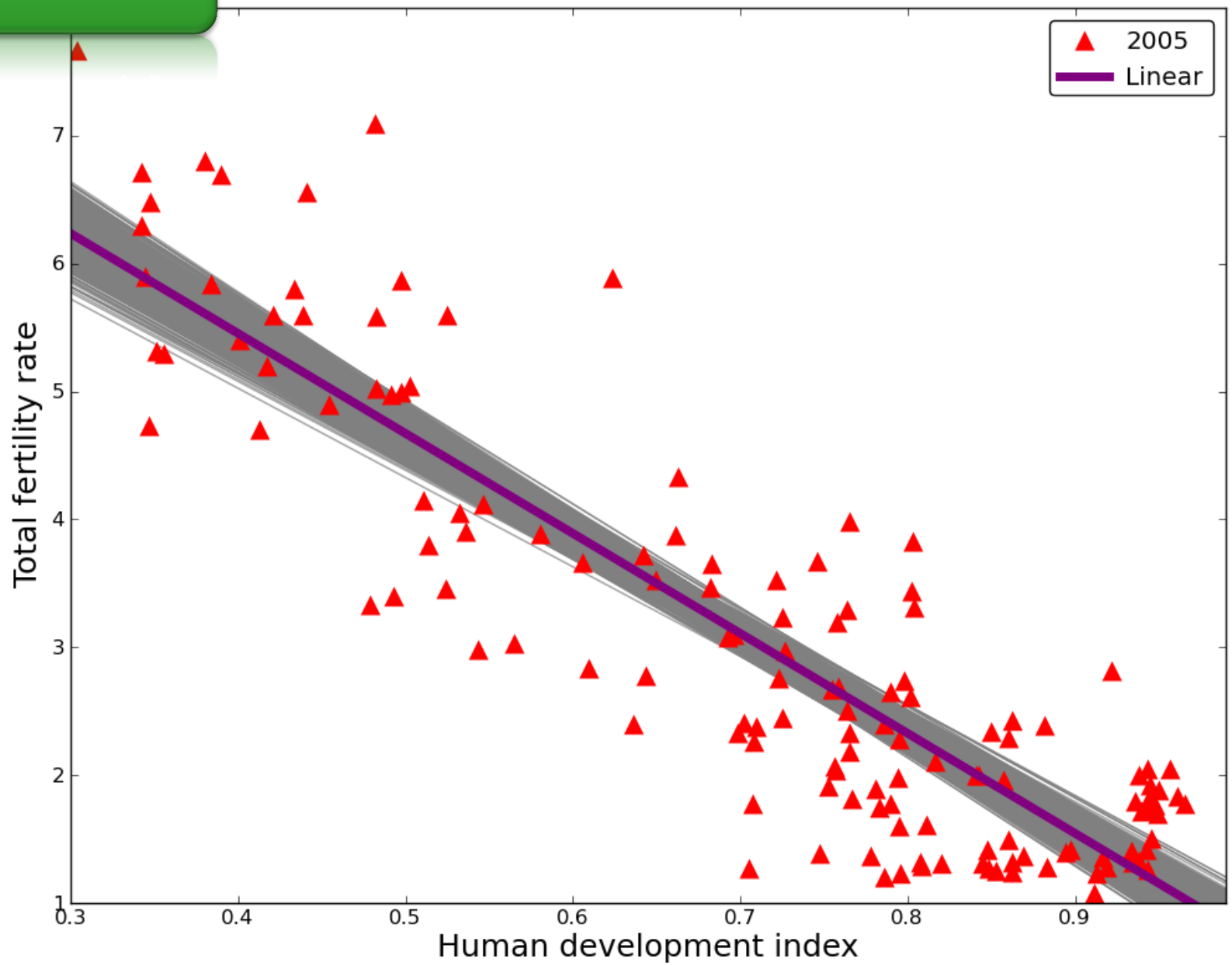
```
In [10]: vars['beta'].value
Out[10]: array([ 0.,  0.])

In [11]: vars['y_obs'].logp
Out[11]: -947.69862197605573
```

src/models.py

```
def fit_linear():
    vars = linear()

    mc.MAP(vars).fit(method='fmin_powell')

    m = mc.MCMC(vars)
    m.sample(iter=10000, burn=5000, thin=5)
    return m
```

```
In [1]: import models

In [2]: m = models.fit_linear()
Sampling: 100% [000] Iterations: 10000

In [3]: import graphics

In [4]: graphics.plot_all_data()

In [5]: graphics.plot_linear_model(m)
```

src/ex1.py

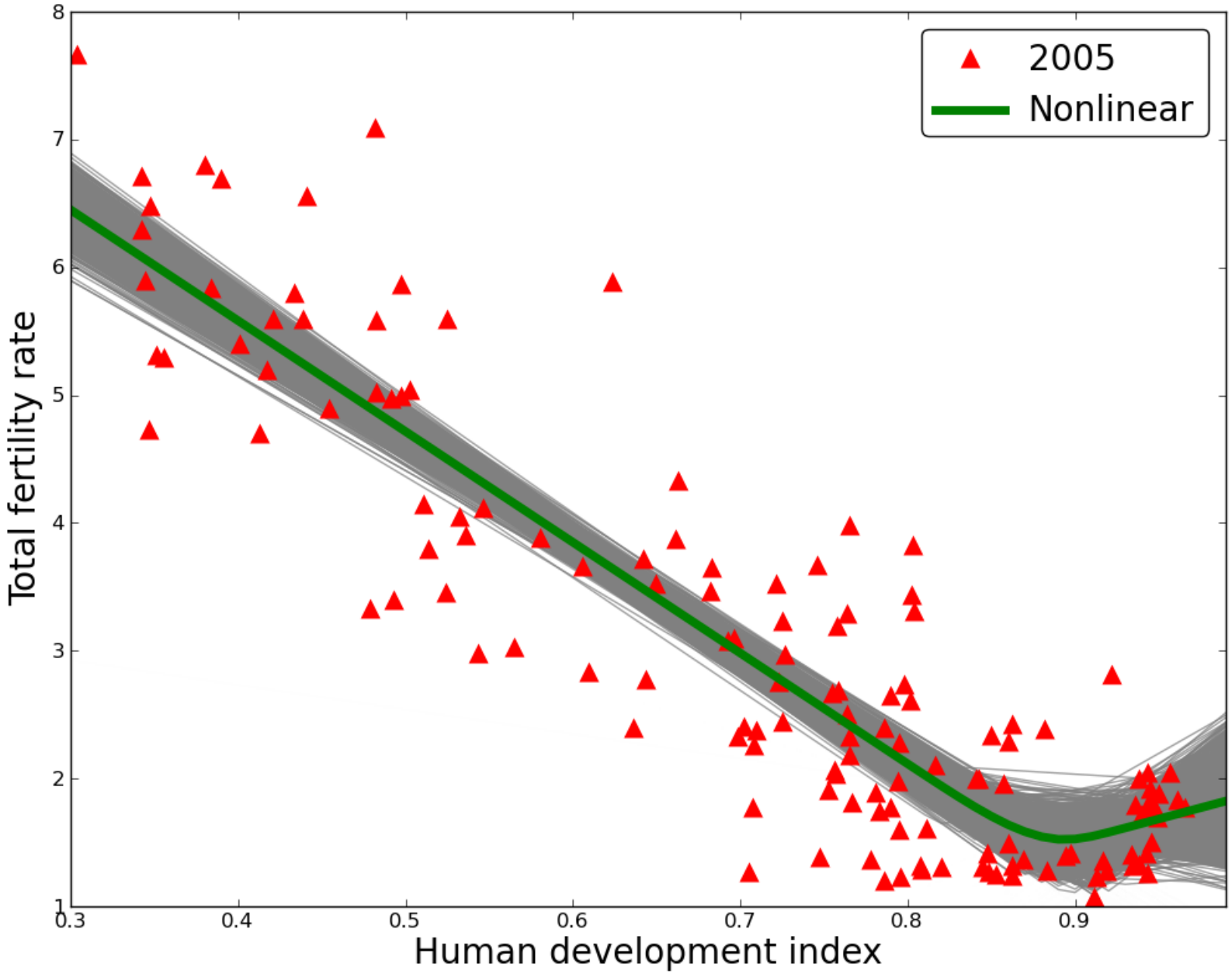

2005
Linear
Total fertility rate
Human development index
1
2
3
4
5
6
7
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Less simple model

$$\mathrm{TFR}_i = \beta_0 + \beta_1 \,\mathrm{HDI}_i + \beta_2 (\mathrm{HDI}_i - \gamma)^+ + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma^2)$$

```python
def nonlinear():
    beta = mc.Uninformative('beta', value=[0., 0., 0.])
    gamma = mc.Normal('gamma', mu=.9, tau=.05**-2)
    sigma = mc.Uniform('sigma', lower=0, upper=100, value=1.)

    @mc.deterministic
    def y_mean(beta=beta, gamma=gamma, X=data.hdi2005):
        return beta[0] + beta[1]*X \
            + beta[2]*pl.maximum(0., X-gamma)

    y_obs = mc.Normal('y_obs', value=data.tfr2005,
                      mu=y_mean, tau=sigma**-2,
                      observed=True)

    return vars()
```


2005
Nonlinear
Total fertility rate
Human development index
8
7
6
5
4
3
2
1
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Your turn to do an extension
Suggestions:

- **Alternative Data**
  - logit(HDI) vs log(TFR)
  - Different time period than 2005
  - Just one country over time

- **Alternative Priors**
  - More informative betas
  - Less informative gamma

- **Alternative Models**
  - Quadratic
  - “Country-level Random Effects”
  - Others?

Institute for
Health Metrics
and Evaluation

PyMC by Example: Human Fertility II

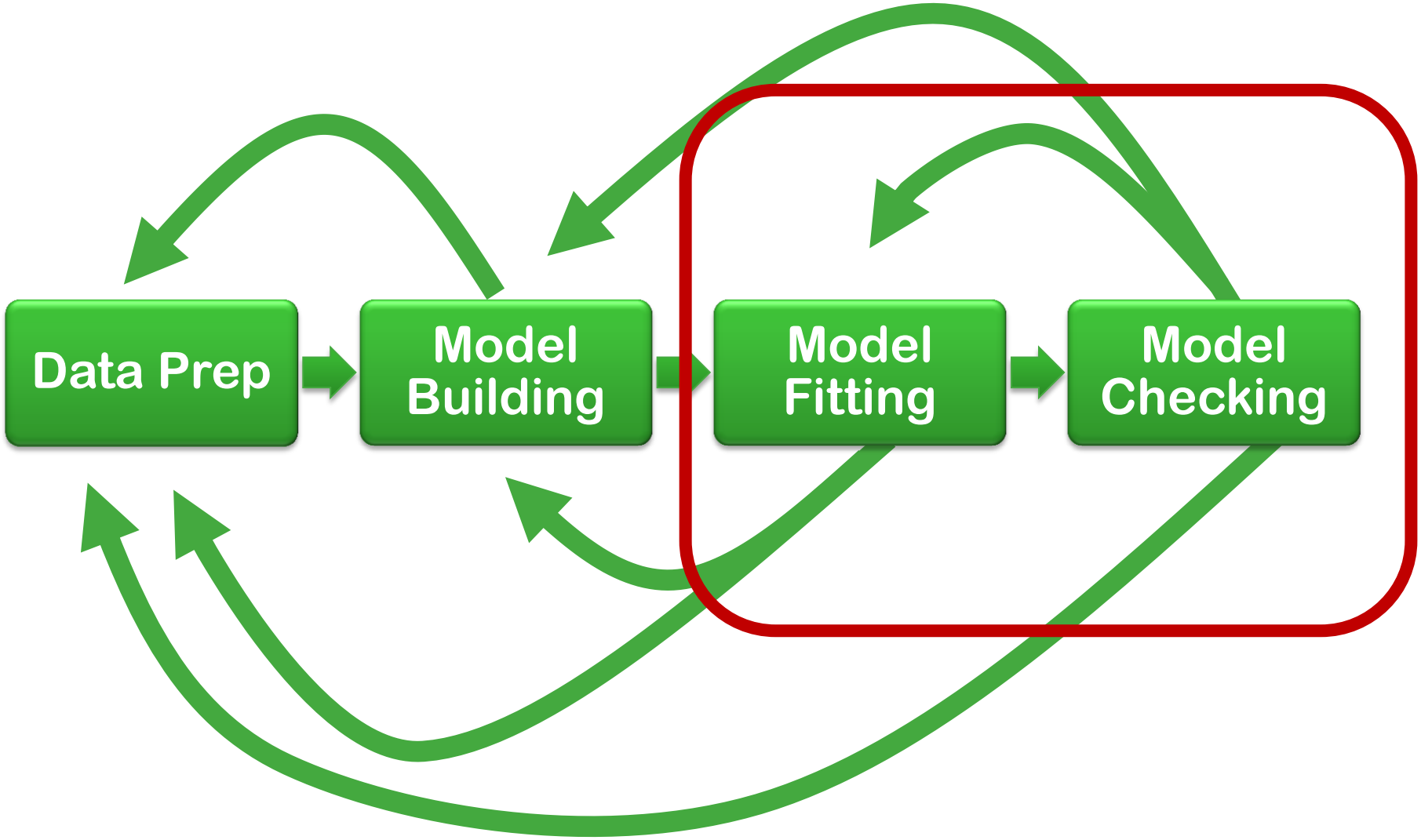
SciPy 2011

Abraham D Flaxman

Institute for Health Metrics and Evaluation

UNIVERSITY OF WASHINGTON

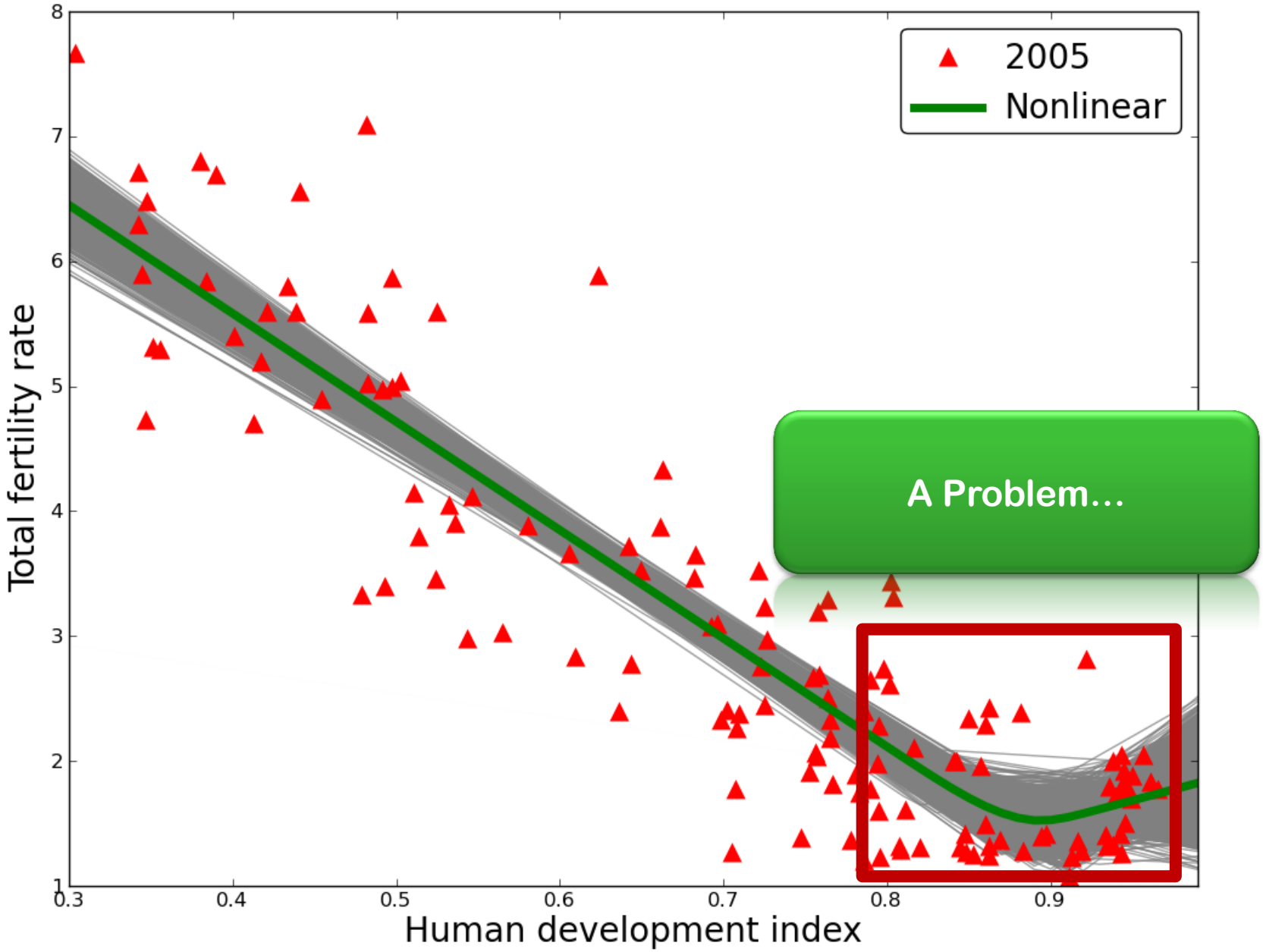
A Generic Template for Bayesian data analysis



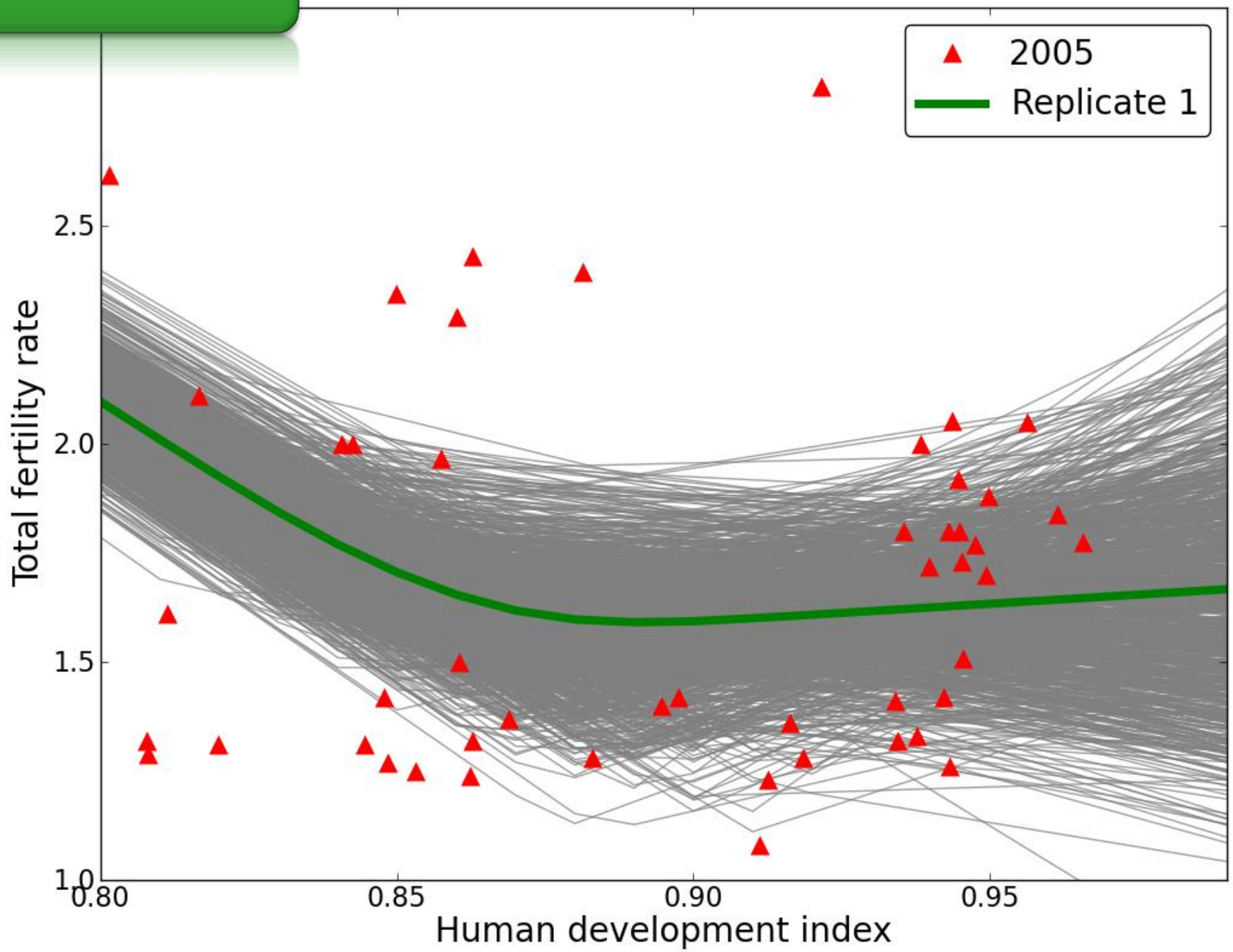
$$\mathrm{TFR}_i = \beta_0 + \beta_1 \,\mathrm{HDI}_i + \beta_2 (\mathrm{HDI}_i - \gamma)^+ + \epsilon_i$$

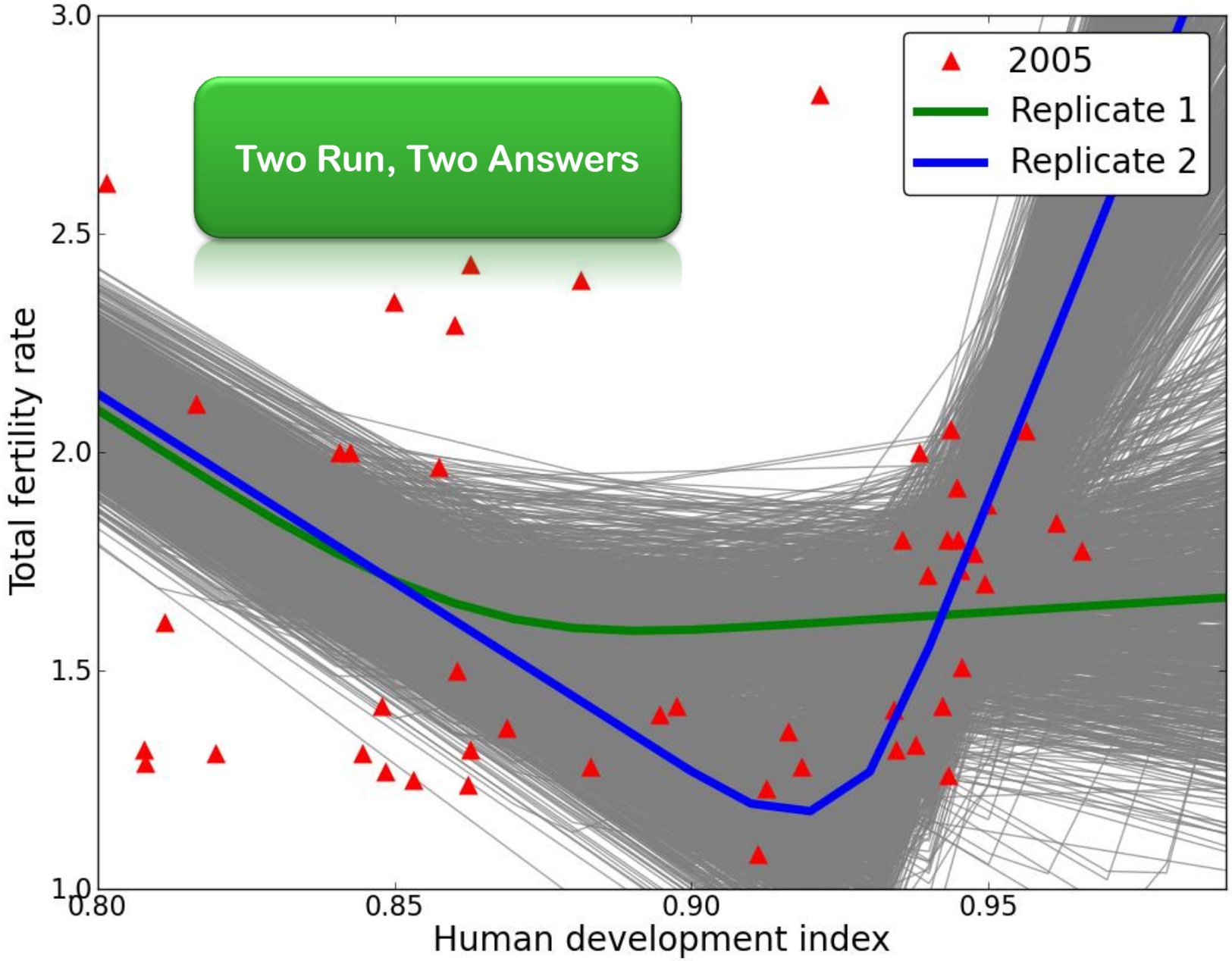
$$\epsilon_i \sim N(0, \sigma^2)$$

```
@mc.deterministic
    def y_mean(beta=beta, gamma=gamma, X=data.hdi):
        return beta[0] + beta[1]*X \
            + beta[2]*pl.maximum(0., X-gamma)

y_obs = mc.Normal('y_obs', value=data.tfr,
                  mu=y_mean, tau=sigma**-2,
                  observed=True)
```


2005
Nonlinear
A Problem…
Total fertility rate
Human development index
8
7
6
5
4
3
2
1
0.3
0.4
0.5
0.6
0.7
0.8
0.9

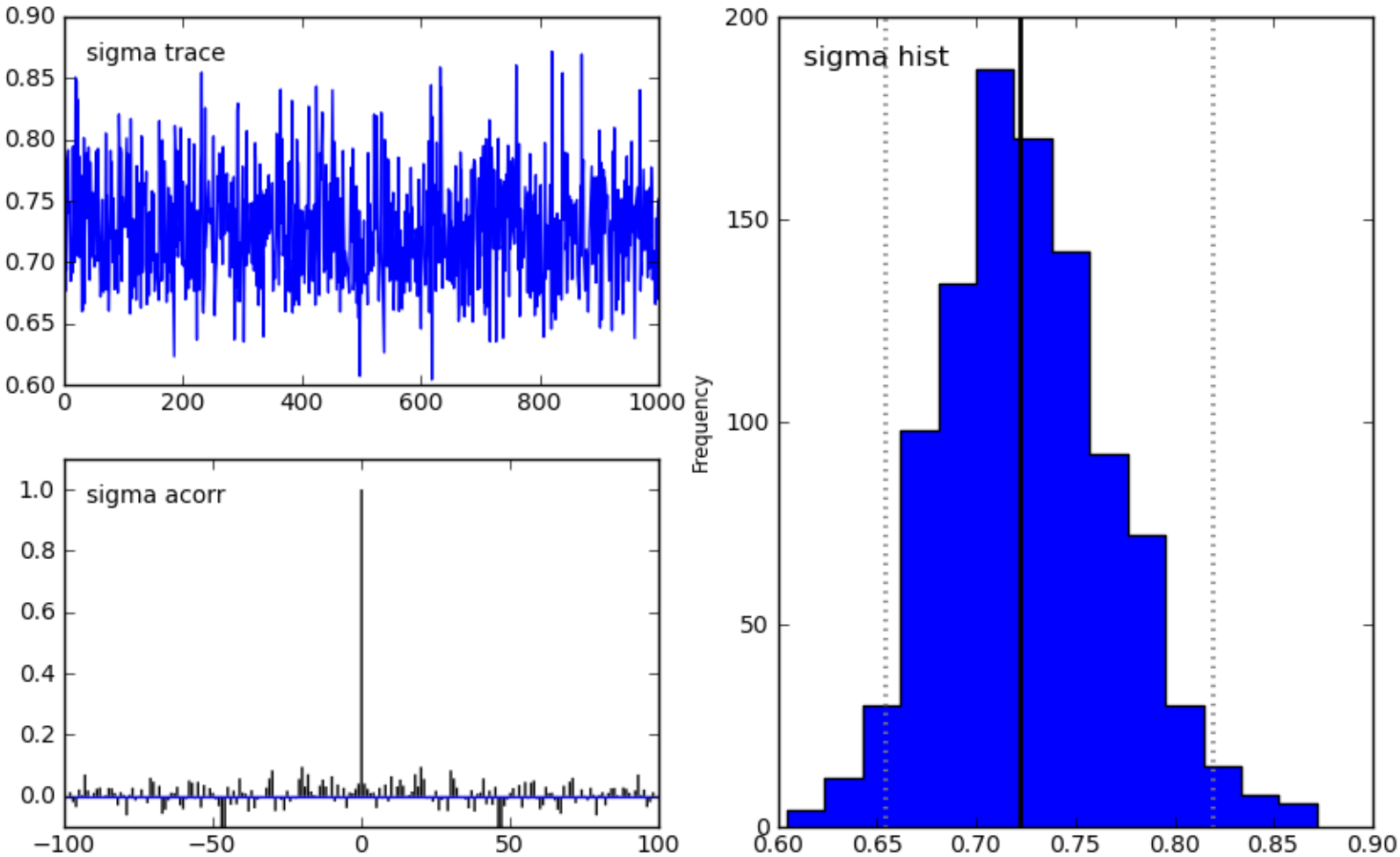
src/ex3.py

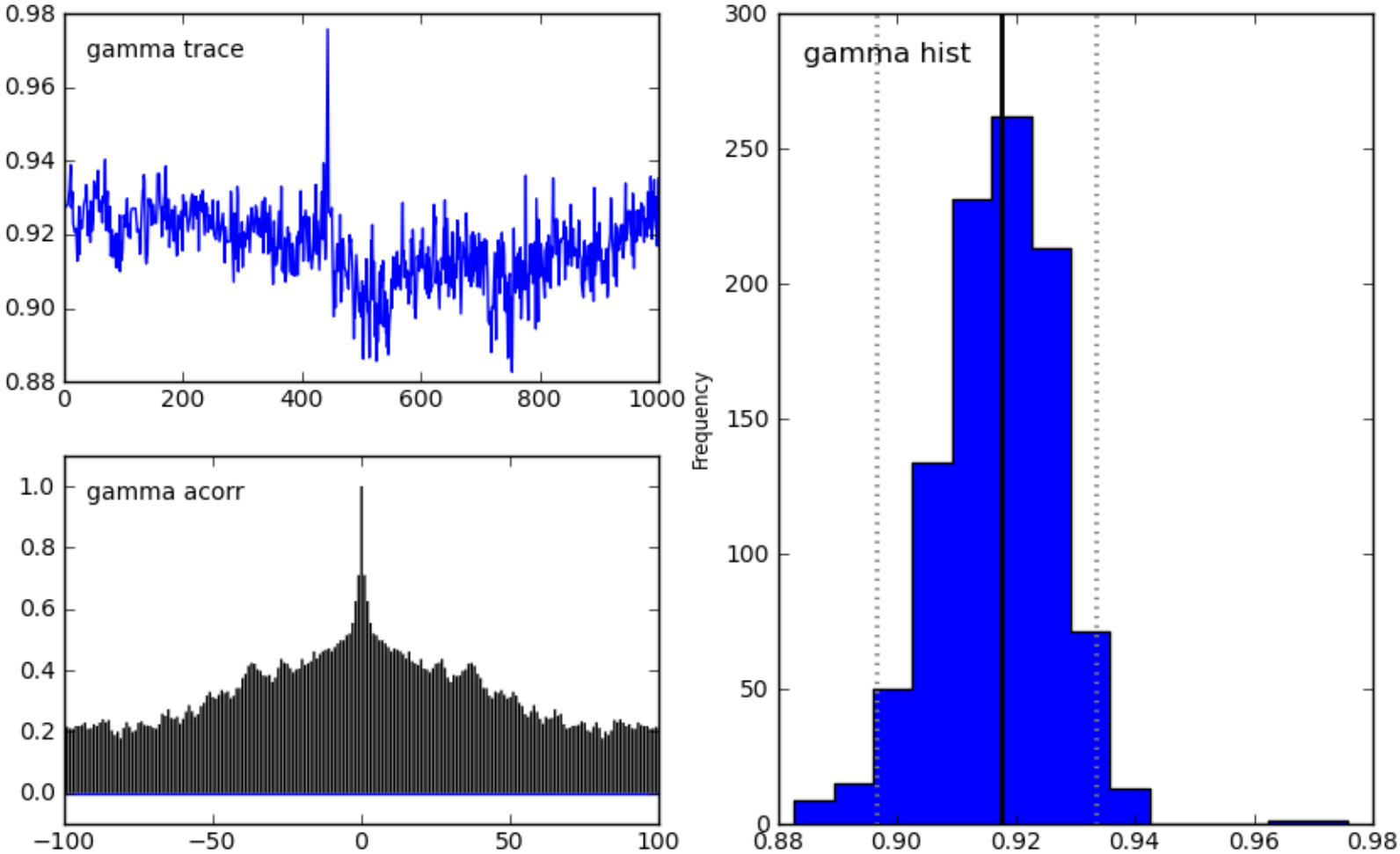

2005
Replicate 1
Total fertility rate
2.5
2.0
1.5
1.0
0.80
0.85
0.90
0.95
Human development index

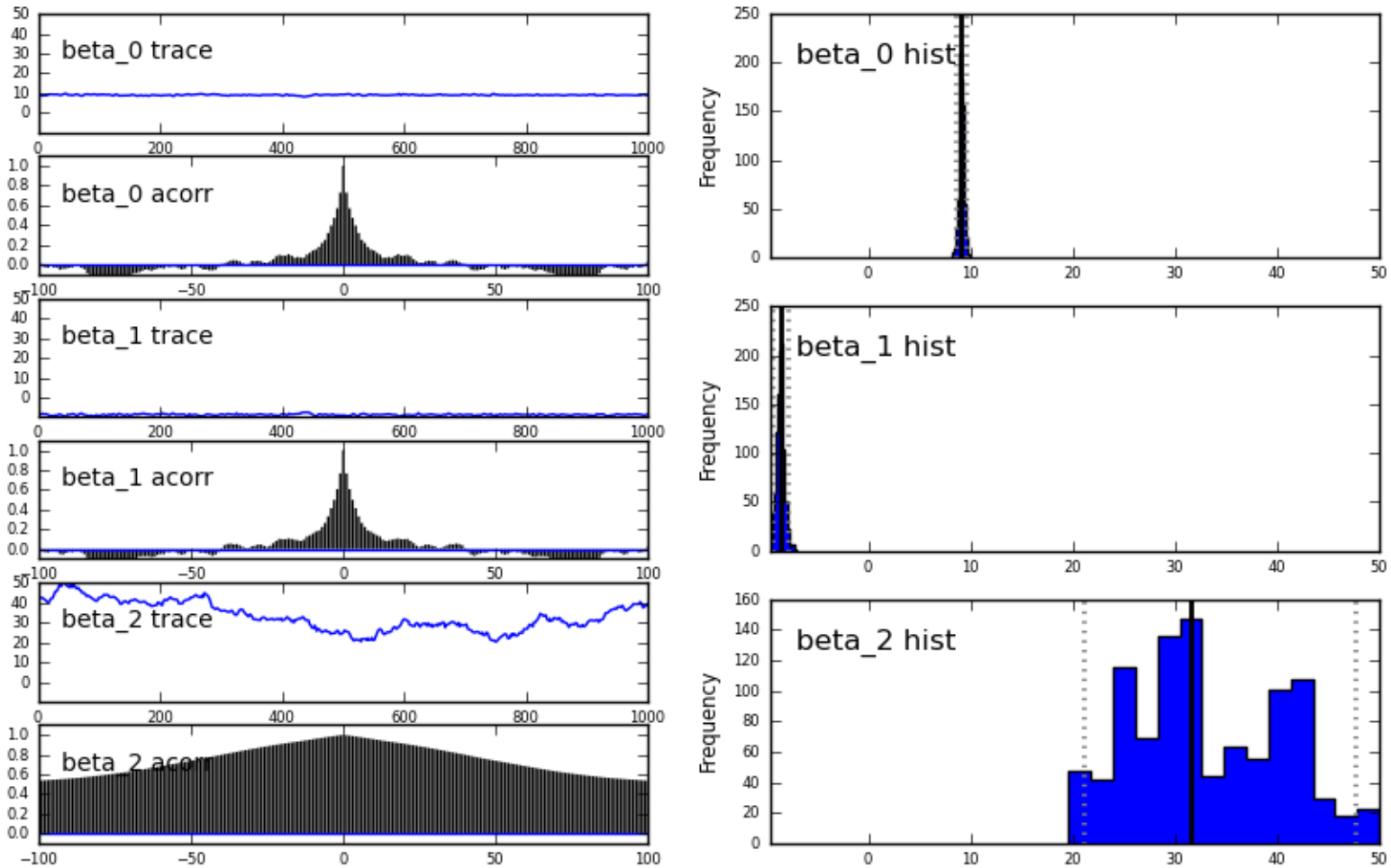

Two Run, Two Answers
2005
Replicate 1
Replicate 2
Total fertility rate
Human development index
3.0
2.5
2.0
1.5
1.0
0.80
0.85
0.90
0.95

Model Fitting in PyMC

```
map = mc.MAP(vars)
map.fit(method='fmin_powell')

mcmc = mc.MCMC(vars)
mcmc.sample(iter=10000, burn=5000, thin=5)

# see results of mcmc with this
mc.Matplot.plot(mcmc)
```


sigma trace
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
200
400
600
800
1000
sigma acorr
1.0
0.8
0.6
0.4
0.2
0.0
−100
−50
0
50
100
sigma hist
Frequency
200
150
100
50
0
0.60
0.65
0.70
0.75
0.80
0.85
0.90


gamma trace
0.98
0.96
0.94
0.92
0.90
0.88
0
200
400
600
800
1000
gamma acorr
1.0
0.8
0.6
0.4
0.2
0.0
−100
−50
0
50
100
gamma hist
Frequency
300
250
200
150
100
50
0
0.88
0.90
0.92
0.94
0.96
0.98


beta_0 trace
beta_0 acorr
beta_1 trace
beta_1 acorr
beta_2 trace
beta_2 acorr
beta_0 hist
beta_1 hist
beta_2 hist
Frequency

How to cope when MCMC don't converge

```
# run chain longer
m.sample(50000, 25000, 50)

# change initial values
mc.MAP(vars).fit(method='fmin_powell')

# change step methods
m = mc.MCMC(vars)
m.use_step_method(mc.AdaptiveMetropolis,
                  m.beta)
```

Step Method Movies

Convergence diagnostics in PyMC

Graphics:

```
mc.Matplot.plot(m)
```

Stats:

```
mc.diagnostics
   .discrepancy
   .gelman_rubin
   .geweke
   .iat
   .raftery_lewis
```

Other:

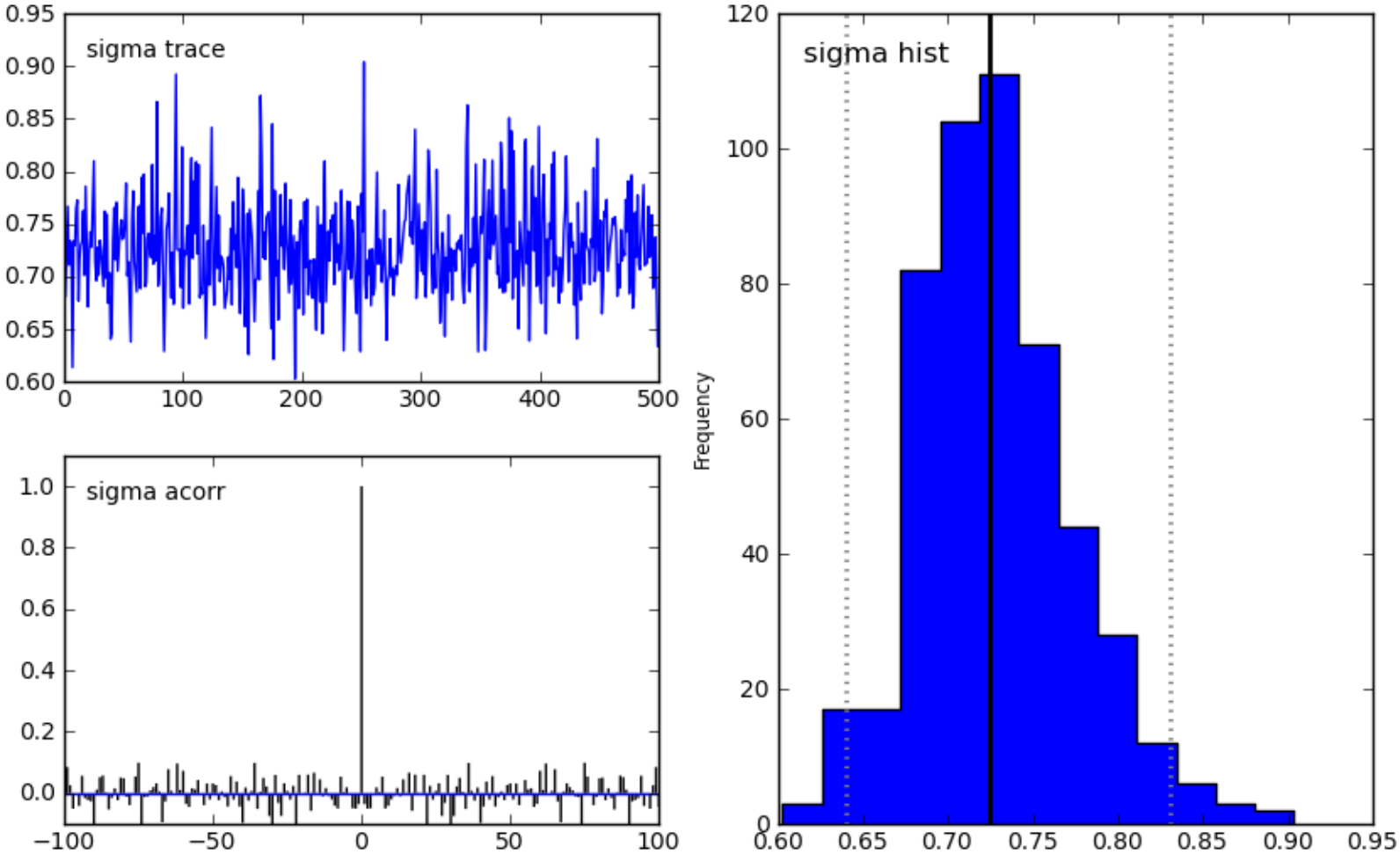
start from a bunch of different initial values and confirm that they all yield the same results.

In our case, Adaptive Metropolis gets the job done

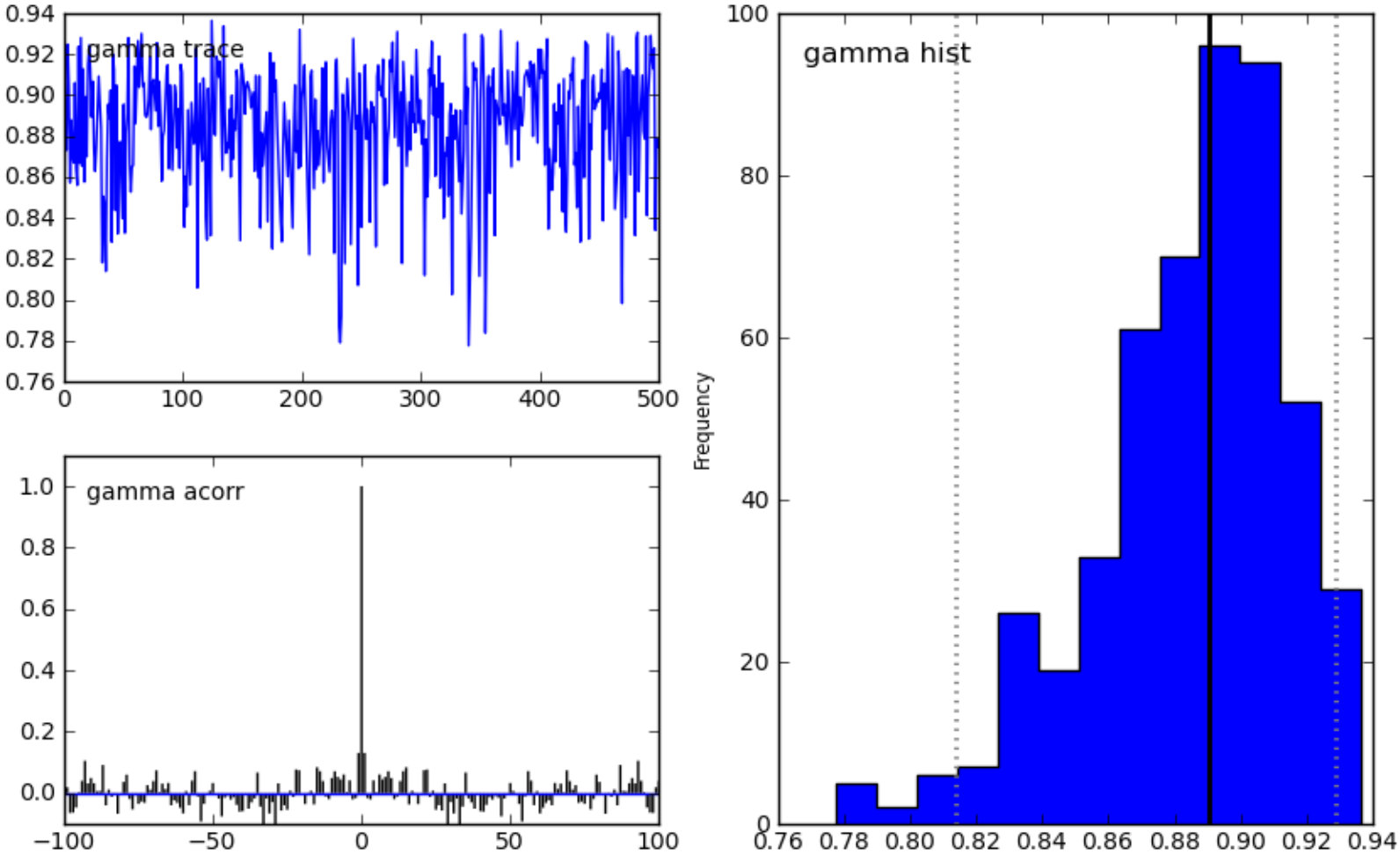
```
vars = models.nonlinear()

# fit with MCMC, but not with
# default step methods
m = mc.MCMC(vars)
m.use_step_method(mc.AdaptiveMetropolis,
                  [m.beta, m.gamma])
m.sample(50000, 25000, 50)

# explore model convergence
mc.Matplot.plot(m)
```

src/ex4.py

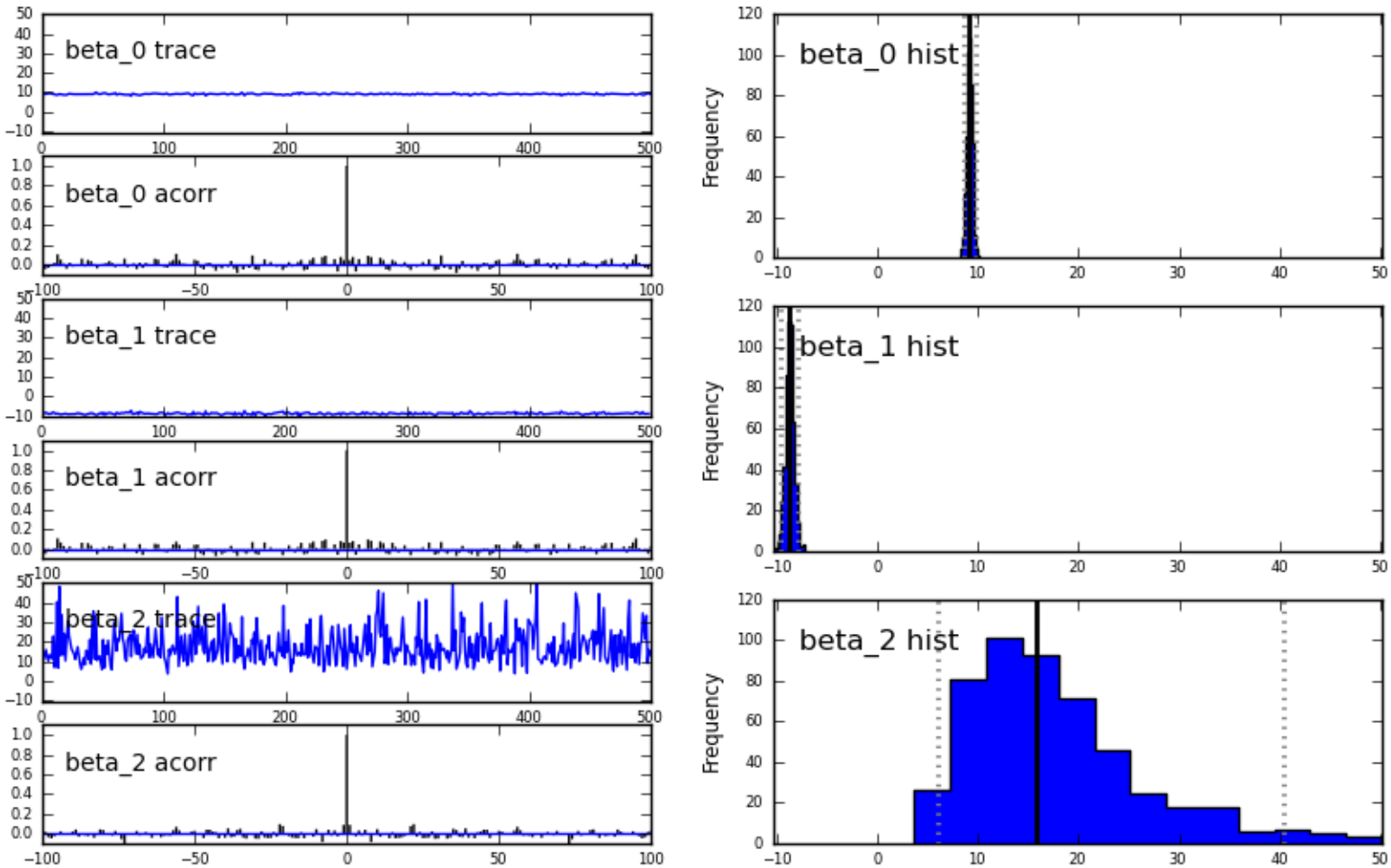
src/ex4.py


sigma trace
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
100
200
300
400
500
sigma acorr
1.0
0.8
0.6
0.4
0.2
0.0
−100
−50
0
50
100
sigma hist
Frequency
120
100
80
60
40
20
0
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95

src/ex4.py

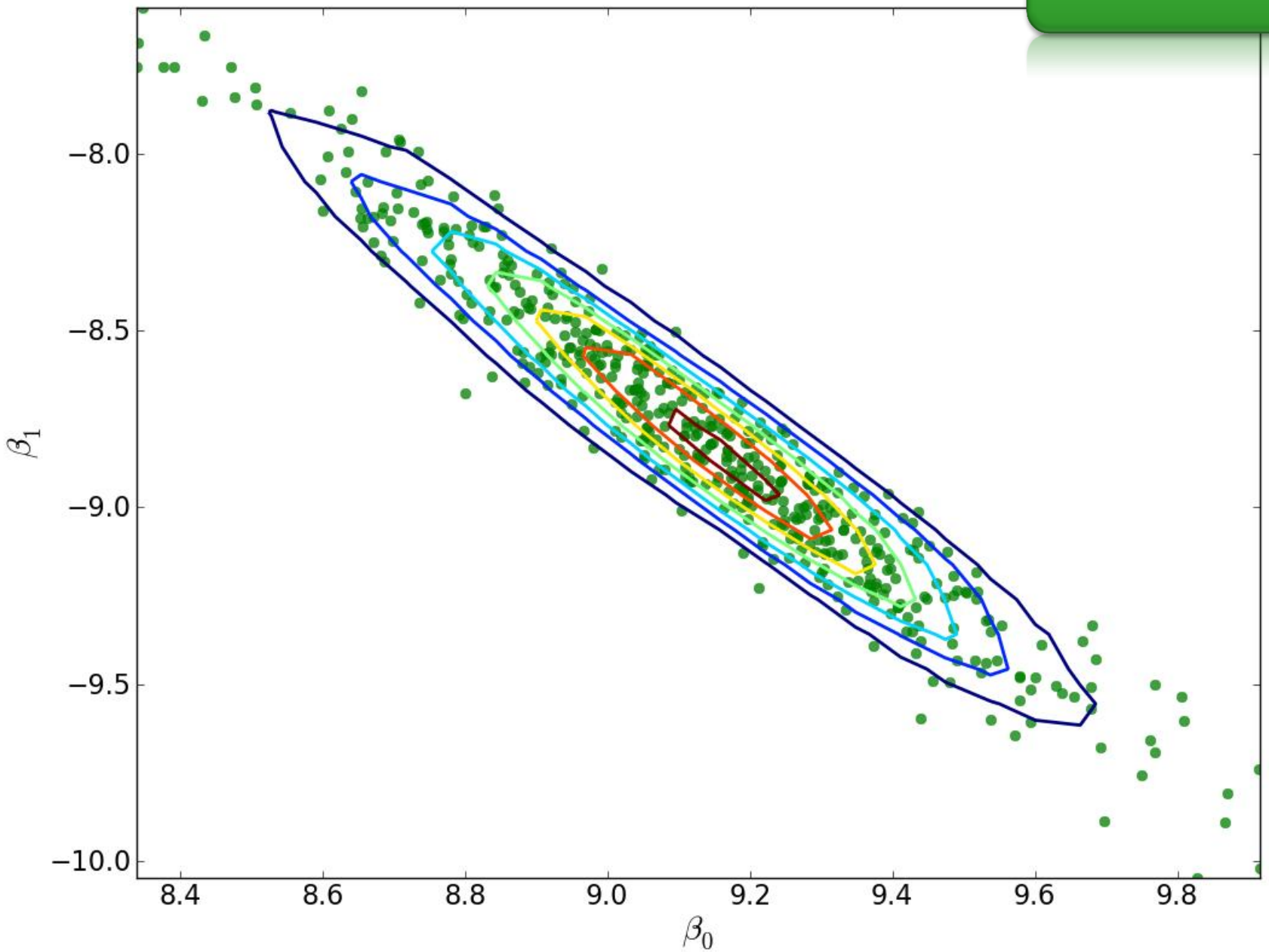

gamma trace
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0
100
200
300
400
500
gamma acorr
1.0
0.8
0.6
0.4
0.2
0.0
−100
−50
0
50
100
gamma hist
Frequency
100
80
60
40
20
0
0.76
0.78
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94

src/ex4.py

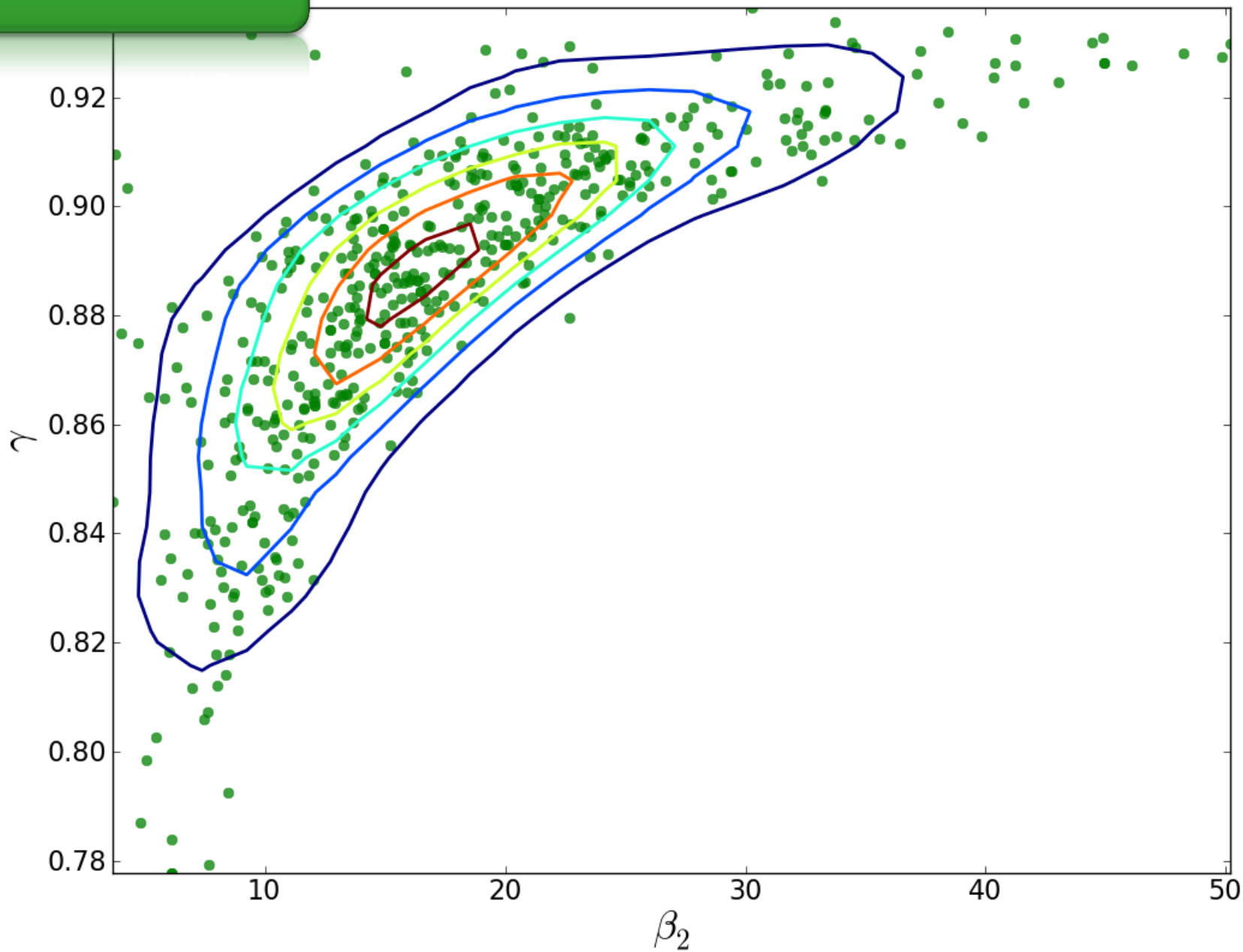

beta_0 trace
beta_0 acorr
beta_1 trace
beta_1 acorr
beta_2 trace
beta_2 acorr
beta_0 hist
beta_1 hist
beta_2 hist
Frequency

In hindsight,
why was that model tough to fit?

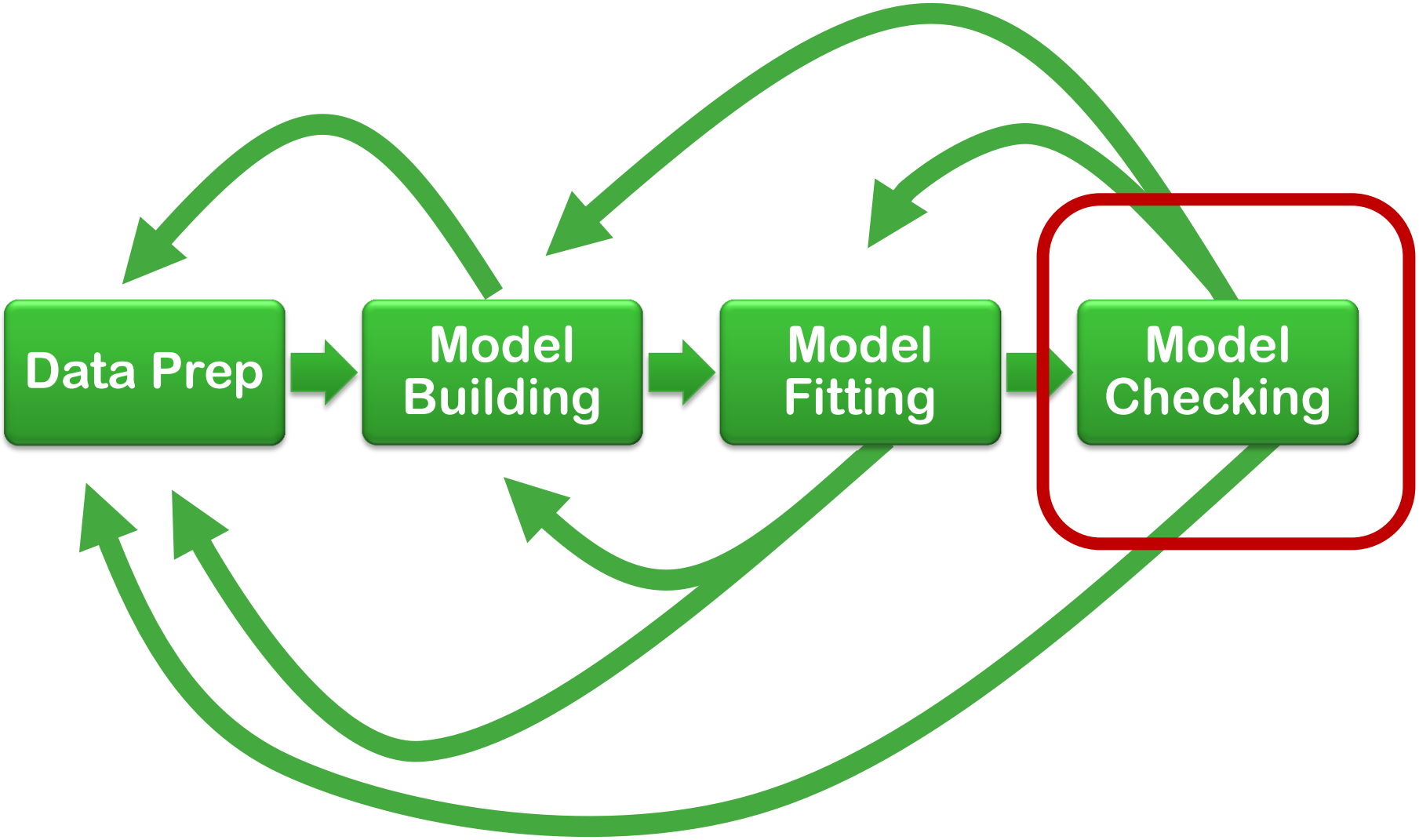
src/ex4.py



src/ex4.py



A Generic Template for Bayesian data analysis

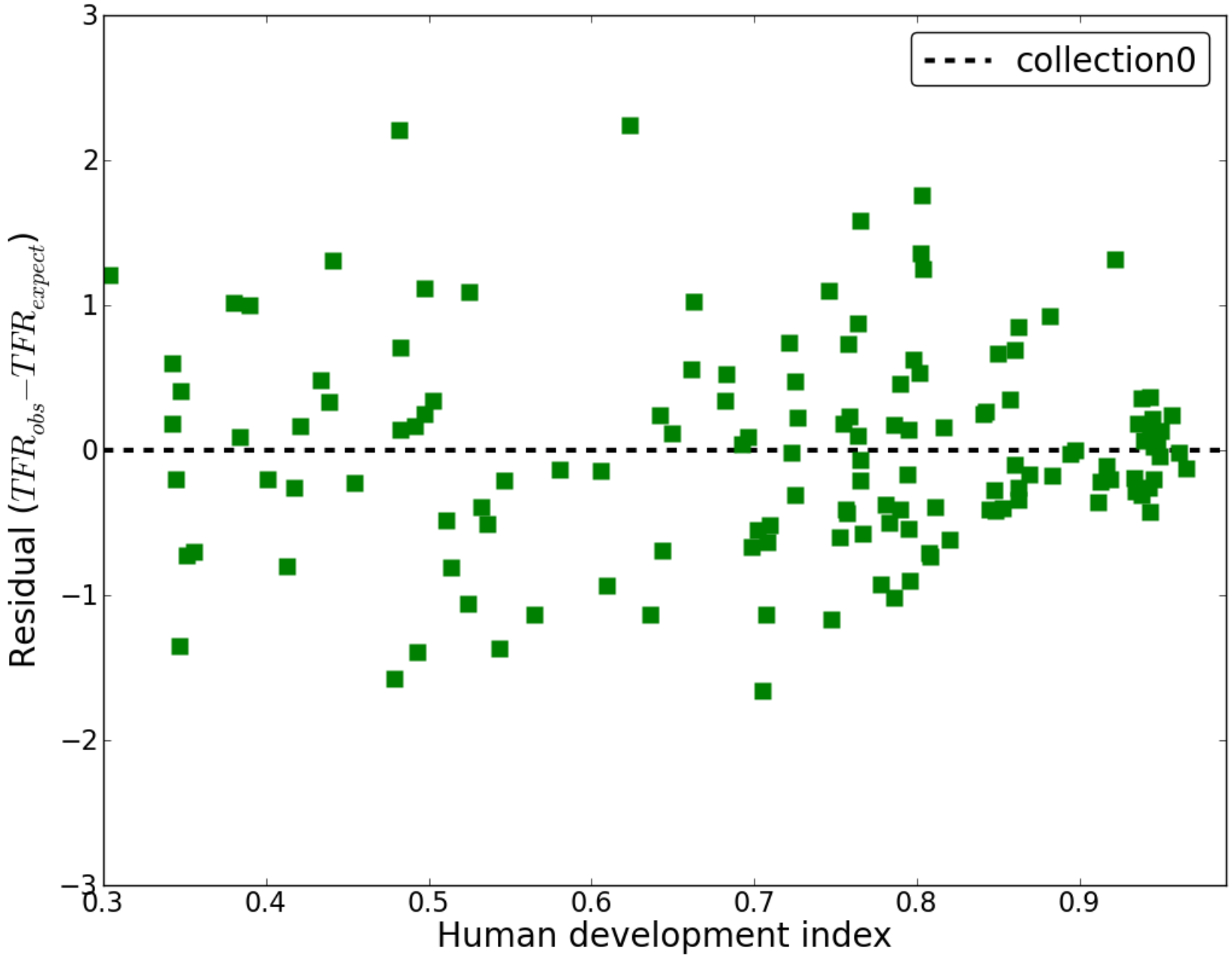


Information Criteron

```
map = mc.MAP(vars)
map.fit(method='fmin_powell')
map.AIC, map.BIC

mcmc = mc.MCMC(vars)
mcmc.sample(iter=10000, burn=5000, thin=5)
mcmc.dic

# or plot residuals

# or posterior predictive checks
```

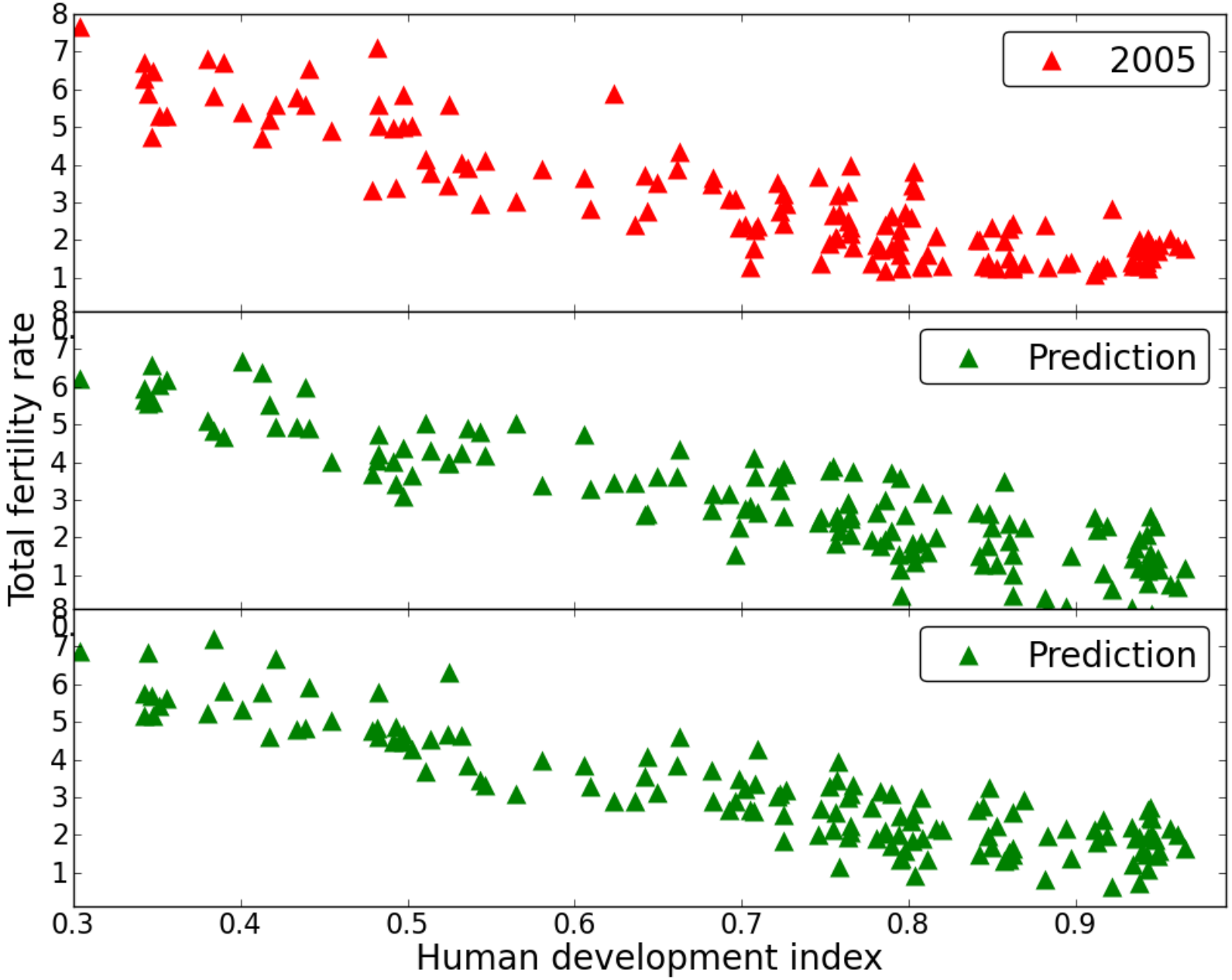
src/ex5.py

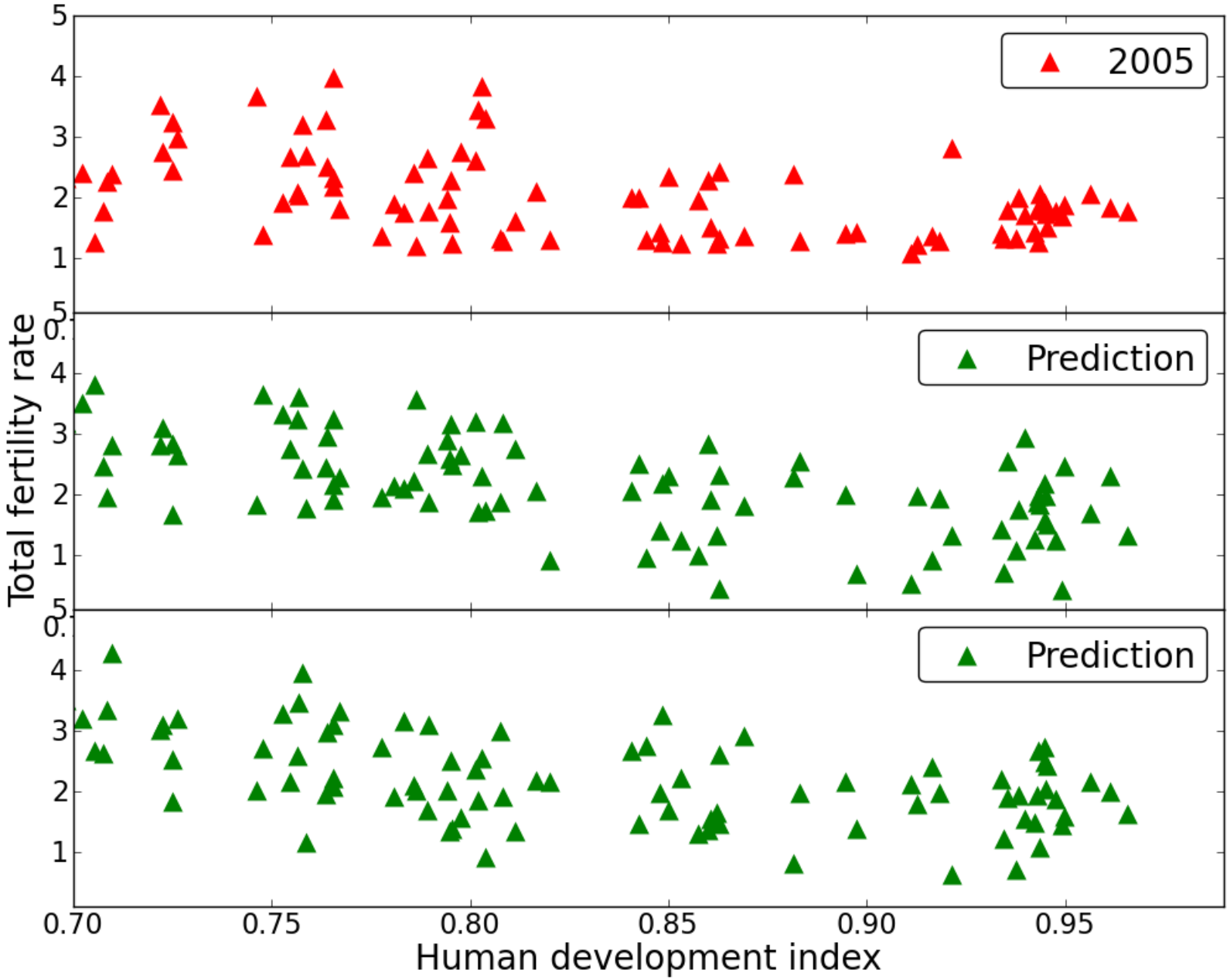
```
y_err = mcmc.y_obs.value - mcmc.y_mean.stats()['mean']
pl.hlines([0], 0, 1, linewidth=3, linestyle='dashed')
pl.plot(data.hdi2005, y_err, 'gs', mew=0, ms=10)
```


collection0
Residual ($TFR_{obs} - TFR_{expect}$)
Human development index
3
2
1
0
−1
−2
−3
0.3
0.4
0.5
0.6
0.7
0.8
0.9

src/ex5.py

```
# add a data posterior prediction deterministic
@mc.deterministic
def y_pred(mu=vars['y_mean'], sigma=vars['sigma']):
    return mc.rnormal(mu, sigma**-2)
vars['y_pred'] = y_pred
```


2005
Prediction
Prediction
Total fertility rate
Human development index
0.3
0.4
0.5
0.6
0.7
0.8
0.9


2005
Prediction
Prediction
Total fertility rate
Human development index
0.70
0.75
0.80
0.85
0.90
0.95

Saving Output

src/ex6.py

```
import pymc as mc
import models
vars = models.nonlinear()

# fit with MCMC, save results
# databases: ram, pickle, txt, sqlite, mysql, hdf5
mcmc = mc.MCMC(vars, db='txt',
               dbname='nonlinear', dbmode='w')
mcmc.sample(5000, 2500, 5)

# load the database from disk
db = mc.database.txt.load('nonlinear')
print db.beta.stats()
```

Brainstorm – what do you want to do with this?

- More complicated models
- Alternative step methods
- Alternative graphics

- Go through a similar exercise with a different application

- Do something with MCMC that has nothing to do with statistics, e.g. making mazes or jigsaw puzzles, or randomly coloring graphs

- Something else?