SSM-*tutorial*

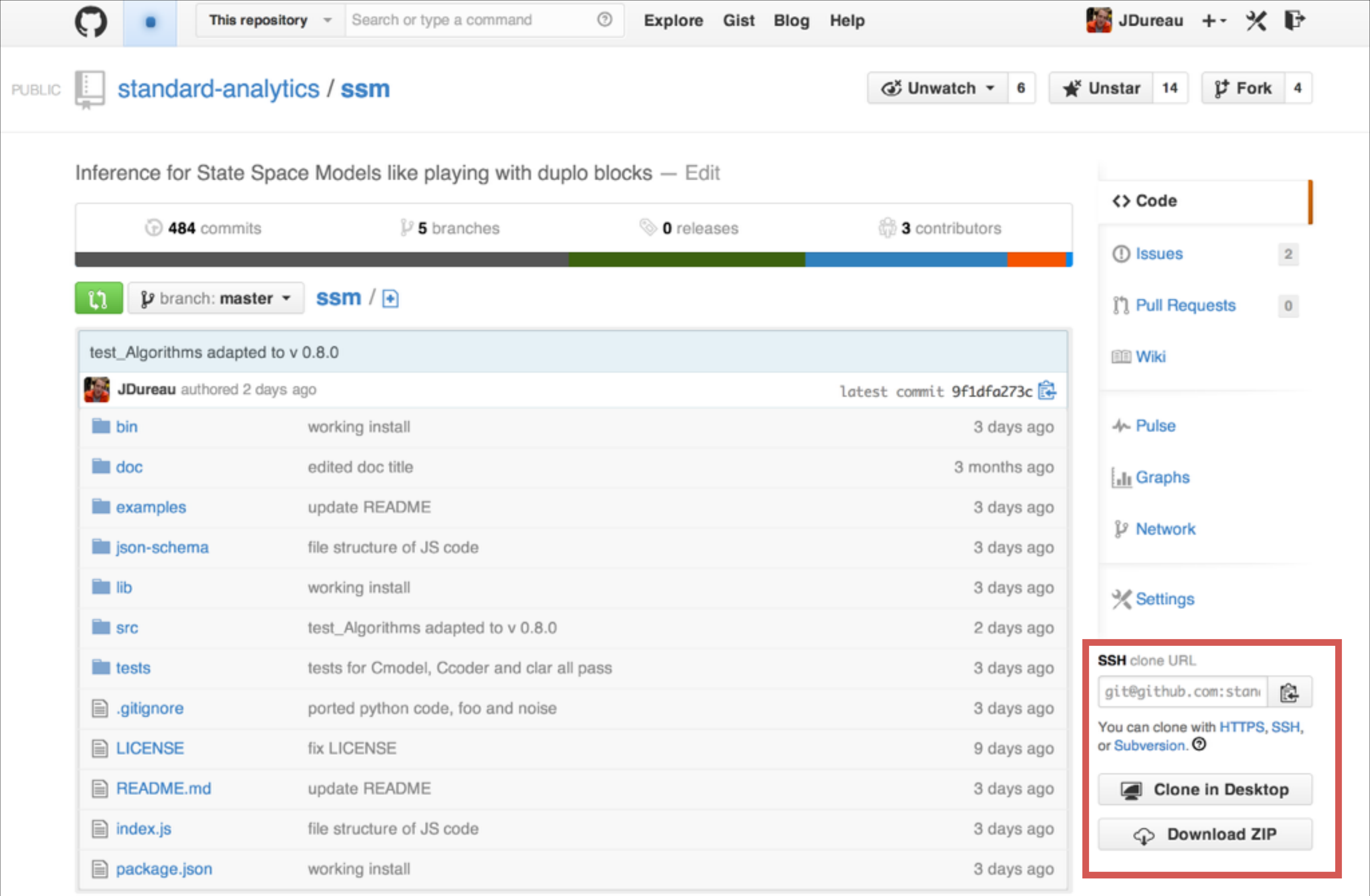
Joseph Dureau

Sébastien Ballesteros

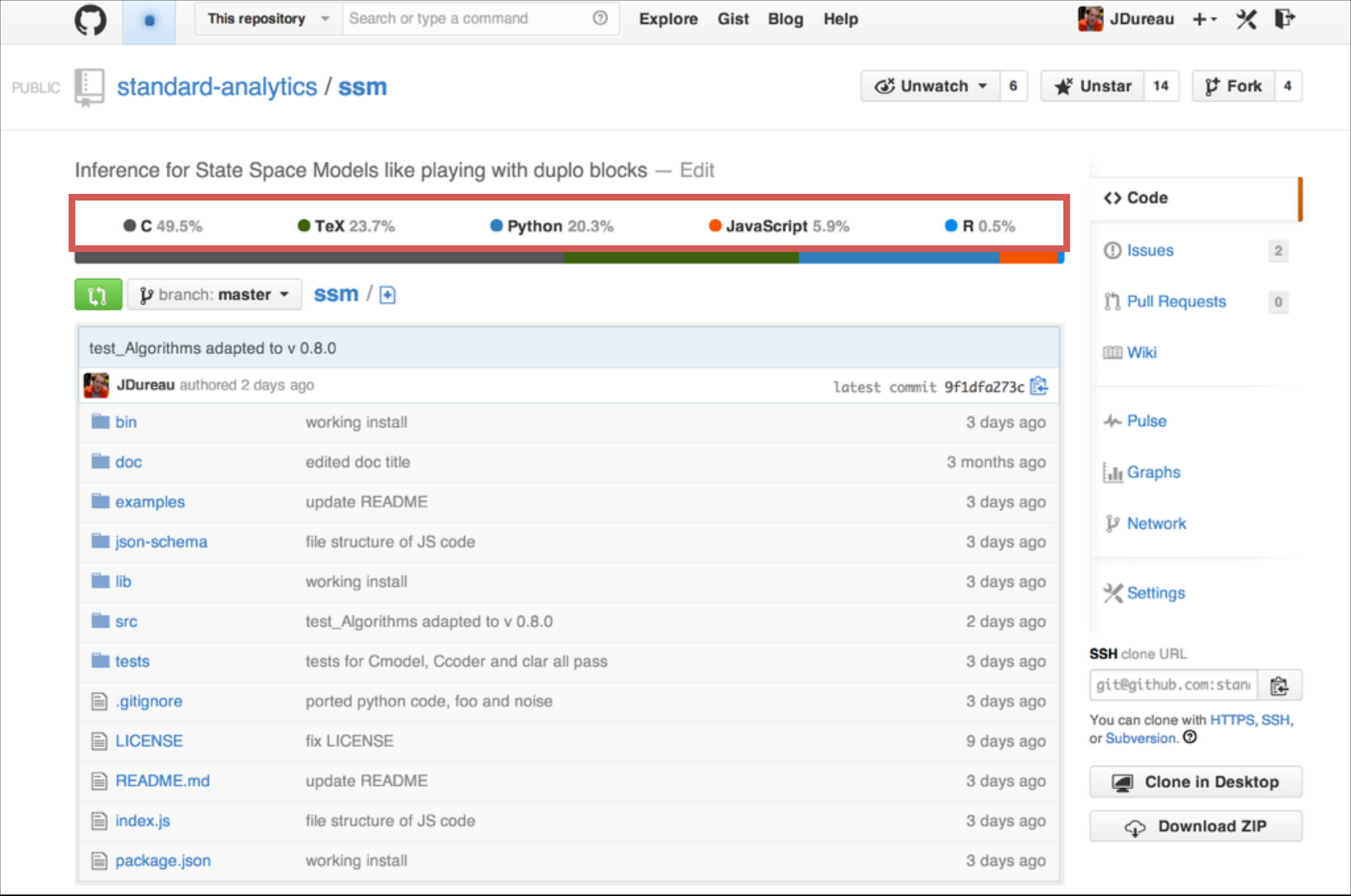
Tiffany Bogich

Organized in collaboration with Institut Pasteur

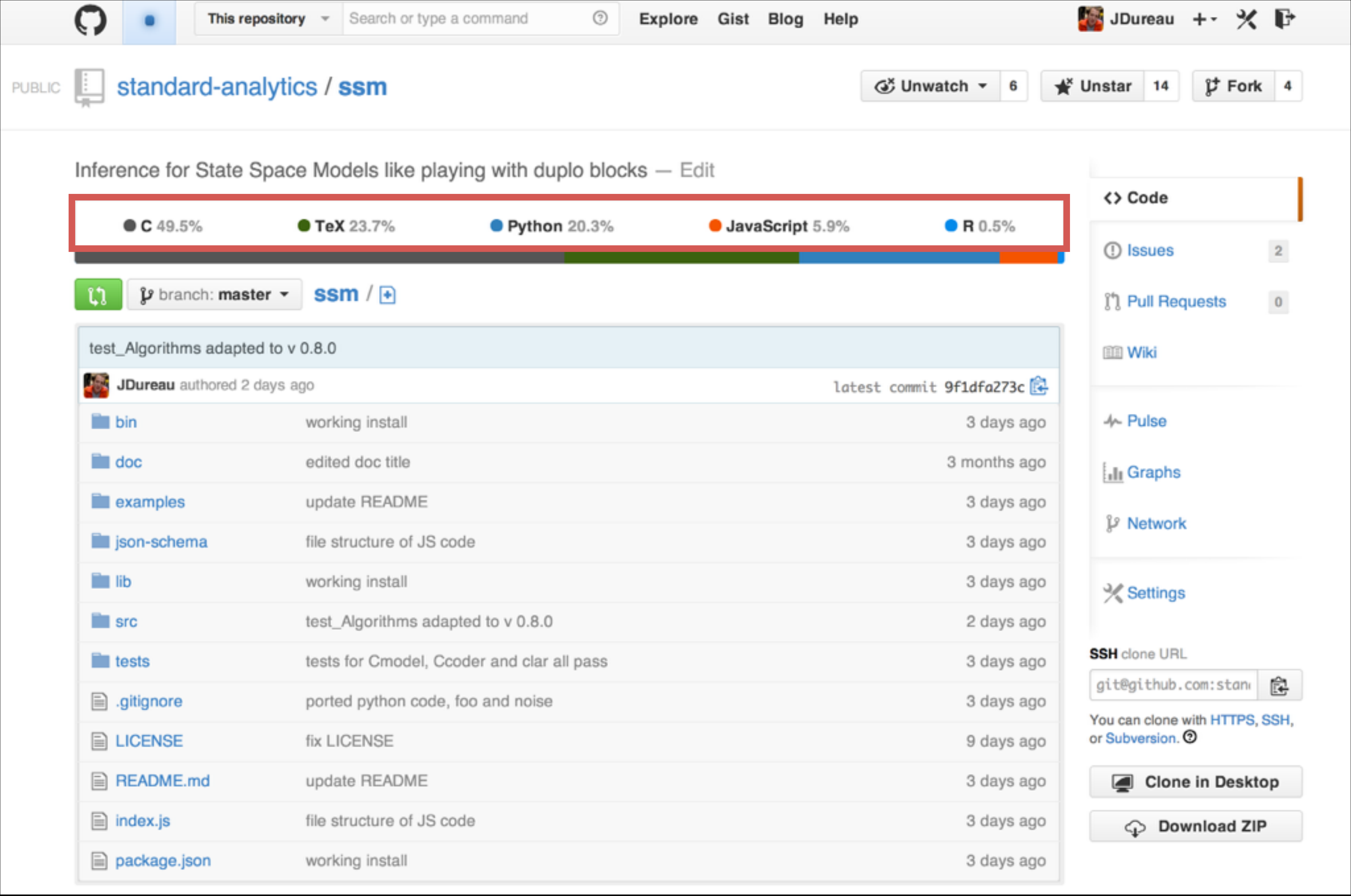
Some context first


This repository
Search or type a command
Explore
Gist
Blog
Help
JDureau
PUBLIC
standard-analytics / ssm
Unwatch 6
Unstar 14
Fork 4
Inference for State Space Models like playing with duplo blocks — Edit
484 commits
5 branches
0 releases
3 contributors
branch: master
ssm /
test_Algorithms adapted to v 0.8.0
JDureau authored 2 days ago
latest commit 9f1dfa273c
bin working install 3 days ago
doc edited doc title 3 months ago
examples update README 3 days ago
json-schema file structure of JS code 3 days ago
lib working install 3 days ago
src test_Algorithms adapted to v 0.8.0 2 days ago
tests tests for Cmodel, Ccoder and clar all pass 3 days ago
.gitignore ported python code, foo and noise 3 days ago
LICENSE fix LICENSE 9 days ago
README.md update README 3 days ago
index.js file structure of JS code 3 days ago
package.json working install 3 days ago
Code
Issues 2
Pull Requests 0
Wiki
Pulse
Graphs
Network
Settings
SSH clone URL
git@github.com:stan
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

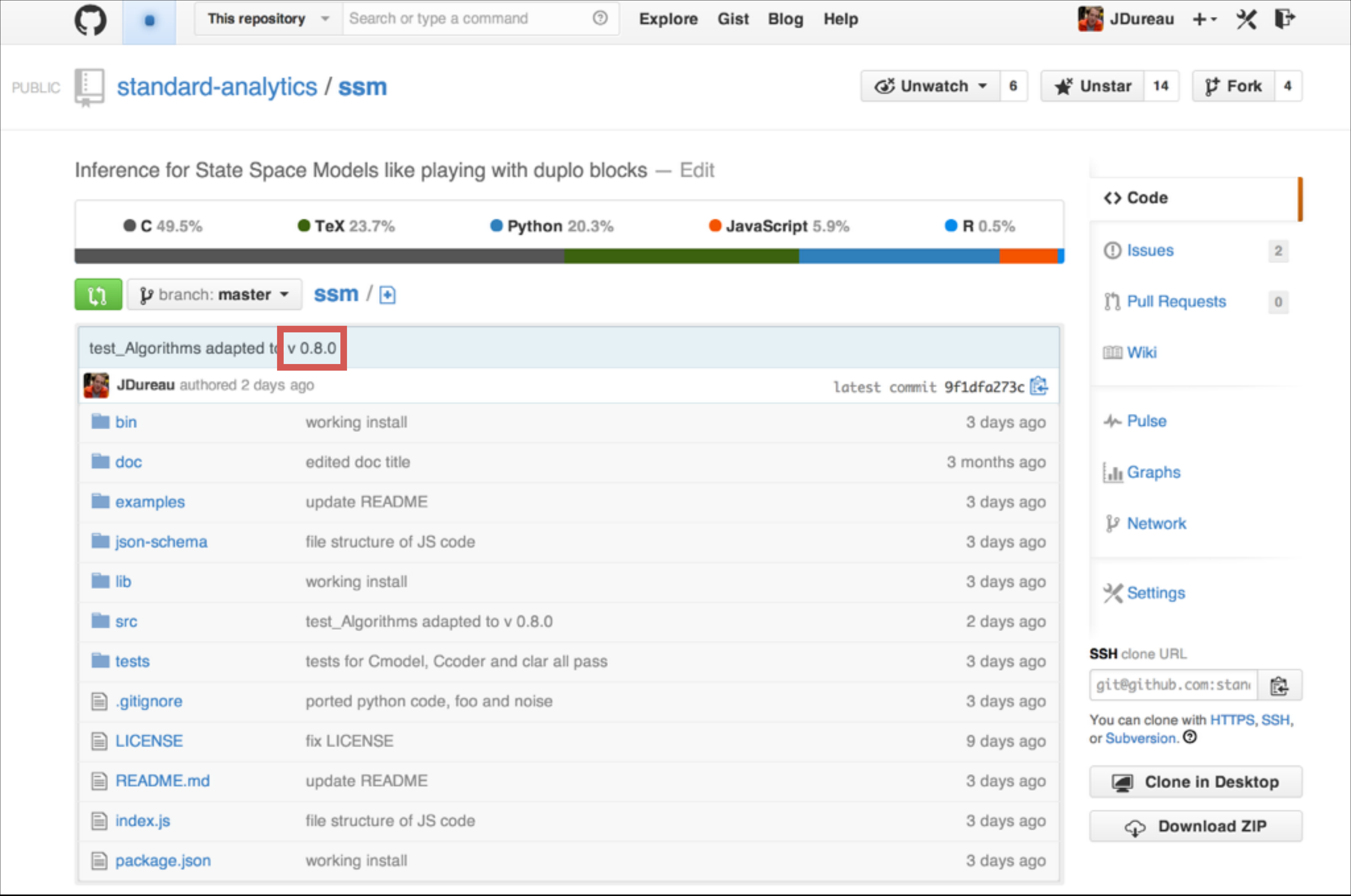
SSM is **free**



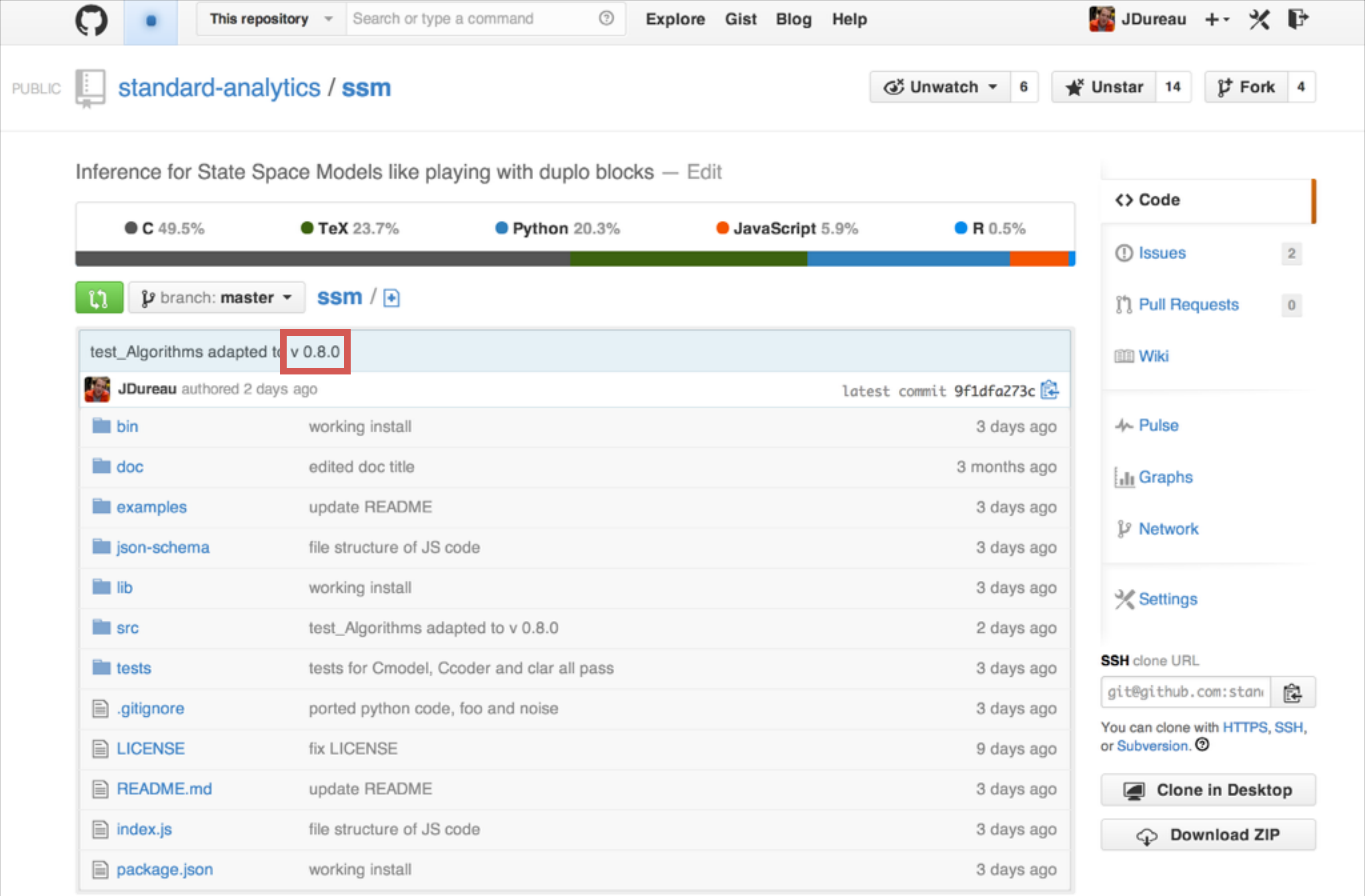
The core of SSM in C…



… wrapped up in so that you don't have to write C



SSM is still **alpha**



The hope is to get **more reliable code** by coding collectively

State
Space
Model

?

$$X_t = \begin{bmatrix} X^{(1)} \\ \vdots \\ X^{(d)} \end{bmatrix}$$

Finite dimensional representation

$$\forall t,\ p(X_{t+dt} | X_{0:t}) = p(X_{t+dt} | X_t)$$

Markovian dynamic

For this tutorial

Bayesian inference for compartmental models in epidemiology

Program

1. Your very first step
2. Random walk in a deterministic world
3. Model creation and model selection
4. Stochasticity and statistical inference
5. Going further

First steps

Terminal terminology

cd: change directory

ls: list files

open <file>: open file

|: pipe

cat: read the content

4 SSM objects:

ssm.json

data.csv

theta.json

prior.json

SSM: data.csv

```
"date","incidence_obs"
"2009-06-01",39
"2009-06-08",113
"2009-06-15",317
"2009-06-22",1233
"2009-06-29",2075
"2009-07-06",3739
"2009-07-13",11520
"2009-07-20",11120
"2009-07-27",6480
"2009-08-03",4000
"2009-08-10",1840
"2009-08-17",960
"2009-08-24",640
```

data.csv

```
"date","incidence_obs"
"2009-06-01",39
"2009-06-08",113
"2009-06-15",317
"2009-06-22",1233
"2009-06-29",2075
"2009-07-06",3739
"2009-07-13",11520
"2009-07-20",11120
"2009-07-27",6480
"2009-08-03",4000
"2009-08-10",1840
"2009-08-17",960
"2009-08-24",640
```

ISO 8601 format

```
"date","incidence_obs"
"2009-06-01",39
"2009-06-08",113
"2009-06-15",317
"2009-06-22",1233
"2009-06-29",2075
"2009-07-06",3739
"2009-07-13",11520
"2009-07-20",11120
"2009-07-27",6480
"2009-08-03",4000
"2009-08-10",1840
"2009-08-17",960
"2009-08-24",640
```

no spaces

prior.json

```
{
  "name": "uniform",
  "distributionParameter" : [
    { "name" : "lower", "value" : 1e-07 },
    { "name" : "upper", "value" : 1e-02 }
  ]
}
```

JSON: JavaScript Object Notation

```
{
  "name": "uniform",
  "distributionParameter" : [
    { "name" : "lower", "value" : 1e-07 },
    { "name" : "upper", "value" : 1e-02 }
  ]
}
```

Lists, between ['s:
ex: ['a',2,[],'jiji']

```
{
  "name": "uniform",
  "distributionParameter" : [
    { "name" : "lower", "value" : 1e-07 },
    { "name" : "upper", "value" : 1e-02 }
  ]
}
```

key-value objects, between {'s:

```
ex: { “key1”: “value1”, “key2”: 2,
      “key3”: [ 3 ],
      “key4”: { “four”: quatre }
    }
```

Name of the
distribution

```
{
  "name": "uniform"
  "distributionParameter" : [
    { "name" : "lower", "value" : 1e-07 },
    { "name" : "upper", "value" : 1e-02 }
  ]
}
```

Supported distributions: uniform, normal, dirac

Parameters of the distribution

```
{
  "name": "uniform",
  "distributionParameter" : [
    { "name" : "lower", "value" : 1e-07 }
    { "name" : "upper", "value" : 1e-02 }
  ]
}
```

Supported distributions: uniform, normal, dirac

Supported parameters: lower, upper, mean, sd, value

```json
{
  "name": "normal",
  "distributionParameter" : [
    { "name" : "mean", "value" : 2 },
    { "name" : "sd", "value" : 1 }
  ]
}
```

Supported distributions: uniform, normal, dirac

Supported parameters: lower, upper, mean, sd, value

```json
{
  "name": "dirac",
  "distributionParameter" : [
    { "value" : 10 }
  ]
}
```

Supported distributions: uniform, normal, dirac
Supported parameters: lower, upper, mean, sd, value

```json
{
  "name": "normal",
  "distributionParameter" : [
    { "name" : "mean", "value" : 2 },
    { "name" : "sd", "value" : 1 },
    { "name" : "lower", "value" : 0 }
  ]
}
```

Supported distributions: uniform, normal, dirac

Supported parameters: lower, upper, mean, sd, value

theta.json

part 1


```
{
  "name": "values",
  "data": {
    "prop_S_london": 0.835,
    "prop_E_london": 0.00057,
    "prop_I_london": 0.000685705,
    "beta_t": 1.113,
    "v_inv": 1.139,
    "k_inv": 1.609
  }
}
```


Parameter vector θ

part 2

```
{
 "name": "covariance",
 "data": {
  "prop_S_london": {"prop_S_london": 0.02} ,
  "prop_E_london": {"prop_E_london": 0.02} ,
  "prop_I_london": {"prop_I_london": 0.02} ,
  "beta_t": {"beta_t": 0.02},
  "v_inv": {"v_inv": 0.02},
  "k_inv": {"k_inv": 0.02}
 }
}
```

Covariance matrix

you shouldn't have to worry about that one ssm.json

5 components

1. Load data
2. Load priors
3. Define population structure
4. Define reactions
5. Define observations

5 components

(y_i) 1. Load data

$p(\boldsymbol{\theta})$ 2. Load priors

$S_t + I_t + R_t = N_t$ 3. Define population structure

$X^{(i)} \xrightarrow{k(X_t,\, \boldsymbol{\theta})^{(j)}} X^{(j)}$ 4. Define reactions

$p(y_i | X_t)$ 5. Define observations

1. Load data

$$(y_i)$$

```
"data": [
  {
    "name": "incidence_obs",
    "require": {
      "path": "../data/data-H1N1-1wave.csv",
      "fields": ["date","incidence_obs"]
    }
  }
],
```

2. Load priors

$$p(\boldsymbol{\theta})$$

Quantity used in the model

```
"inputs": [
  {
    "name": "S_london",
    "transformation": "prop_S_london*N_london",
    "to_resource": "S_london/N_london",
    "description": "Number of susceptible individuals",
    "require": {
      "path": "../priors/prop_S_london.json",
      "name": "prop_S_london"
    }
  },
```

Quantity on which prior is defined

2. Load priors

$$\mathrm{p}(\boldsymbol{\theta})$$

Transformation

```
"inputs": [
    {
        "name": "S_london",
        "transformation": "prop_S_london*N_london",
        "to_resource": "S_london/N_london",
        "description": "Number of susceptible individuals",
        "require": {
            "path": "../priors/prop_S_london.json",
            "name": "prop_S_london"
        }
    },
```

2. Load priors

$$\mathrm{p}(\boldsymbol{\theta})$$

```
"inputs": [
    {
        "name": "S_london",
        "transformation": "prop_S_london*N_london",
        "to_resource": "S_london/N_london",
        "description": "Number of susceptible individuals",
        "require": {
            "path": "../priors/prop_S_london.json",
            "name": "prop_S_london"
        }
    },
```

Relative path

3. Define population structure

```
"populations": [
    {
        "name": "london",
        "composition": ["S_london", "I_london"],
        "remainder": {
            "name": "R_london",
            "pop_size": “N_london"
        }
    }
],
```

Constant population

(R becomes a dummy variable)

3. Define population structure

```
"populations": [
    {
        "name": "london",
        "composition": ["S_london", “I_london”, “R_london”]
    }
],
```

4. Define reactions

$$X^{(i)} \xrightarrow{k(X_t,\, \boldsymbol{\theta})^{(j)}} X^{(j)}$$

```
"reactions": [
    {
        "from": "S_london", "to": "I_london",
        "rate": "beta_t/N_london*I_london",
        "description":"infection", "keywords": ["transmission"],
        "accumulators": [“incidence"]
    },
    {
        "from": "I_london", "to": "R_london",   "rate": "v",
        “description":"recovery"
    }
],
```

$$\text{incidence} = \int k(X_t,\, \boldsymbol{\theta}) X^{(i)} \, dt$$

5. Define observations

$$p(y_i|X_t)=\mathcal{N}\{h(X_t,\boldsymbol{\theta});\mu,\sigma\}$$

```
"observations": [
    {
        "name": "incidence_obs",
        "start": "2009-05-01",
        "distribution": "discretized_normal",
        "mean": "rep * incidence",
        "sd": "sqrt(rep * ( 1.0 - rep ) * incidence )"
    }
]
```

y

$h(X_t)$

You've made it

ssm.json

data.csv

theta.json

prior.json

Your first SSM command

Compile: `ssm`

Enter bin: `cd bin`

Look around: `ls`

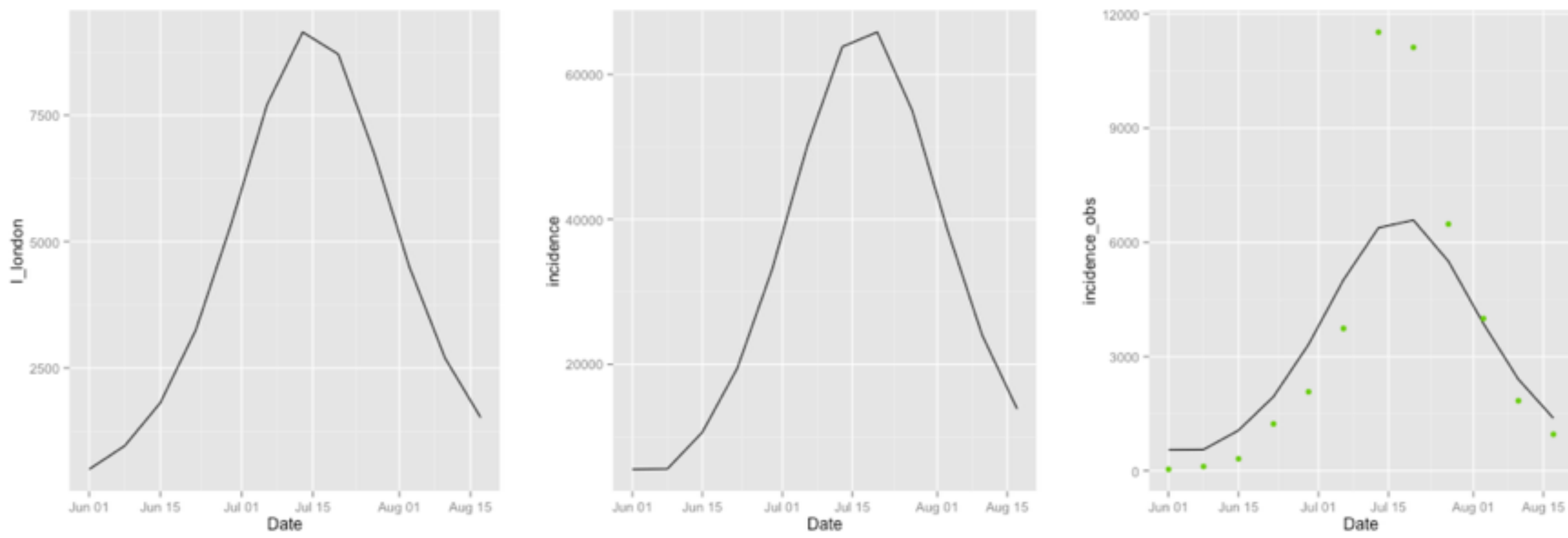
In doubt: `./simul -h`

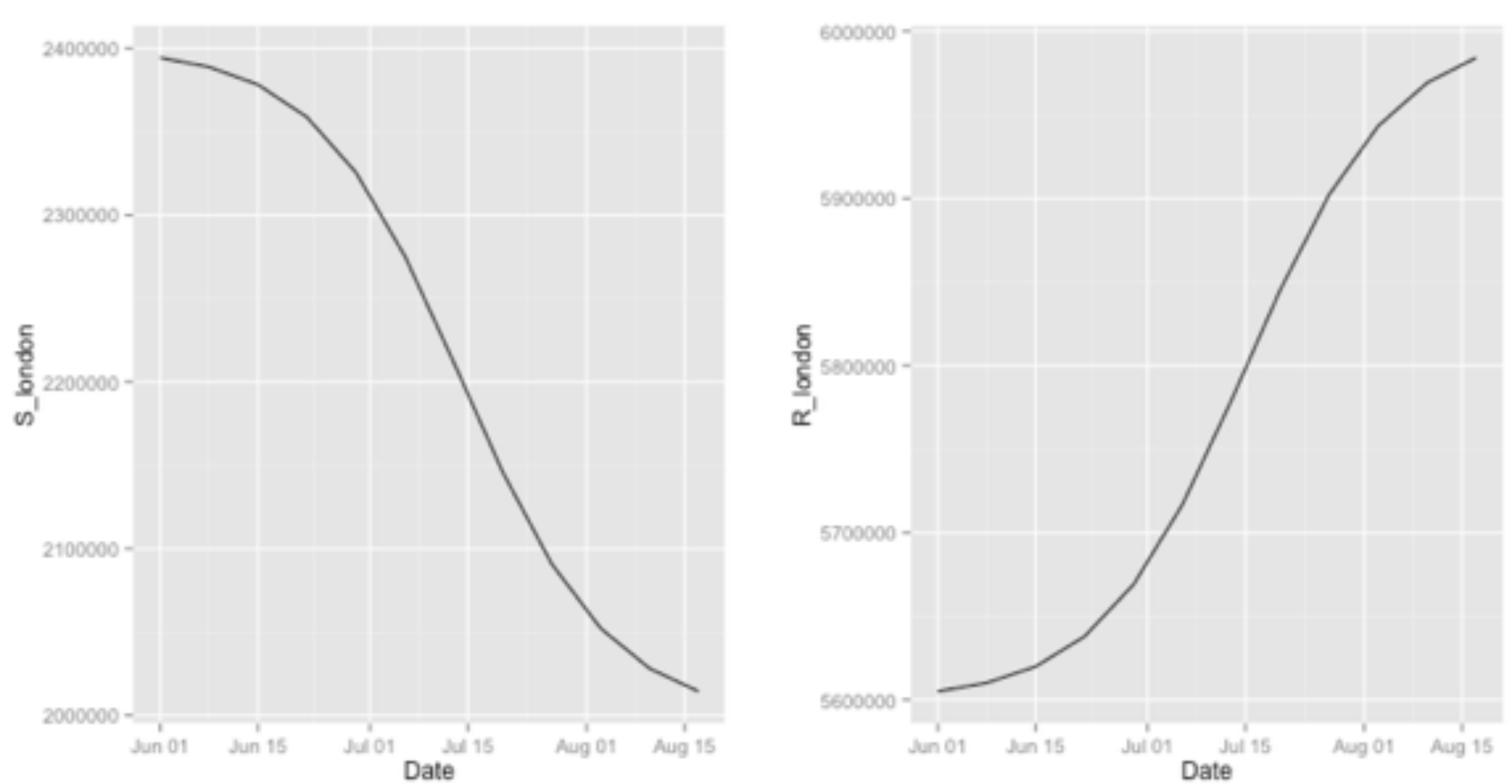
Simulate:

```
cat ../theta.json |./simul --traj
```

Check out the output: `open X_0.csv`

Plot the output: open Tutorial.R with R or RStudio





Let’s talk about inference!

Random walk in a deterministic world

Explore $p(X_t,\boldsymbol{\theta}|y)$

Estimate parameters

$$p(\boldsymbol{\theta}|y)=\int p(X_t,\boldsymbol{\theta}|y)dX_t$$

Reconstruct past scenarios

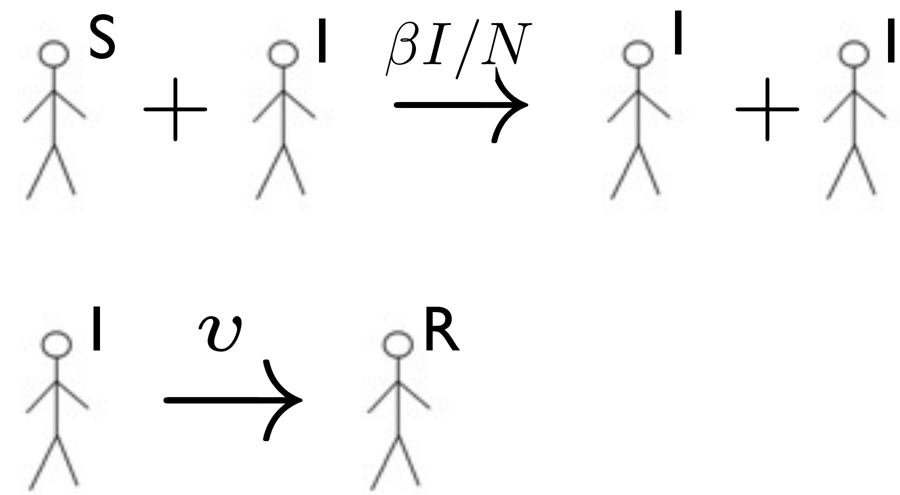
$$p(X_t|y)=\int p(X_t,\boldsymbol{\theta}|y)d\boldsymbol{\theta}$$

Derive model choice indicators: $p(M1|y) > p(M2|y)$

Objectives of Bayesian inference

infection:

recovery:



$$\frac{dS}{dt} = -\beta I/N \times S$$

$$\frac{dI}{dt} = \beta I/N \times S - vI$$

$$\frac{dR}{dt} = vI$$

In a
nice deterministic world....

Good approximation using
Euler or Runge-Kutta methods

Likelihood

✓

$$p(y|\theta) = \prod p(y_i|X_{t_i}, \theta)p(X_{t_i}|\theta)$$

✓

Model

In a
nice deterministic world....

Objective

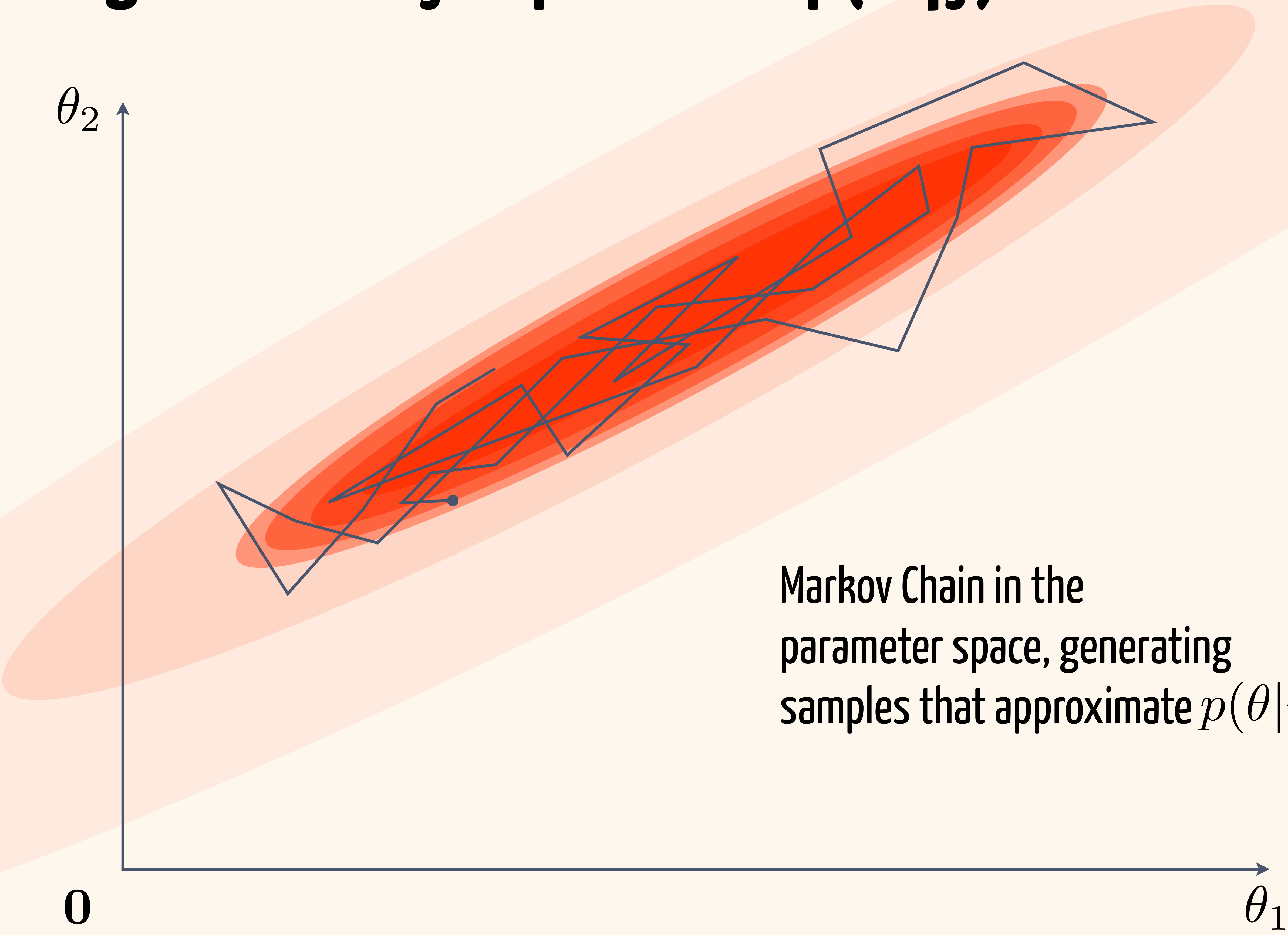
Can be computed *

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

A typical MCMC application problem

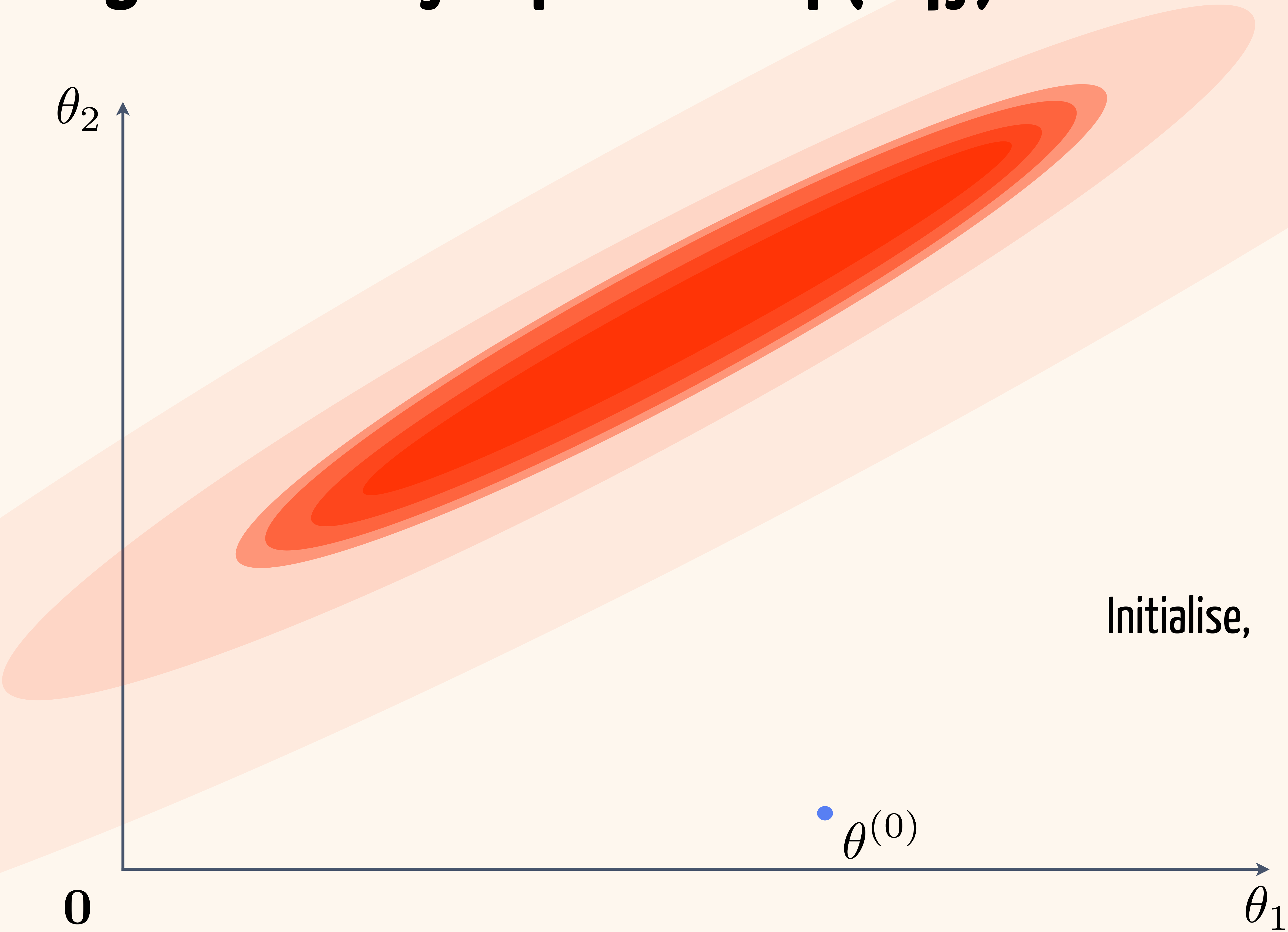
* with arbitrary precision

Marginal density exploration: $p(\theta|y)$



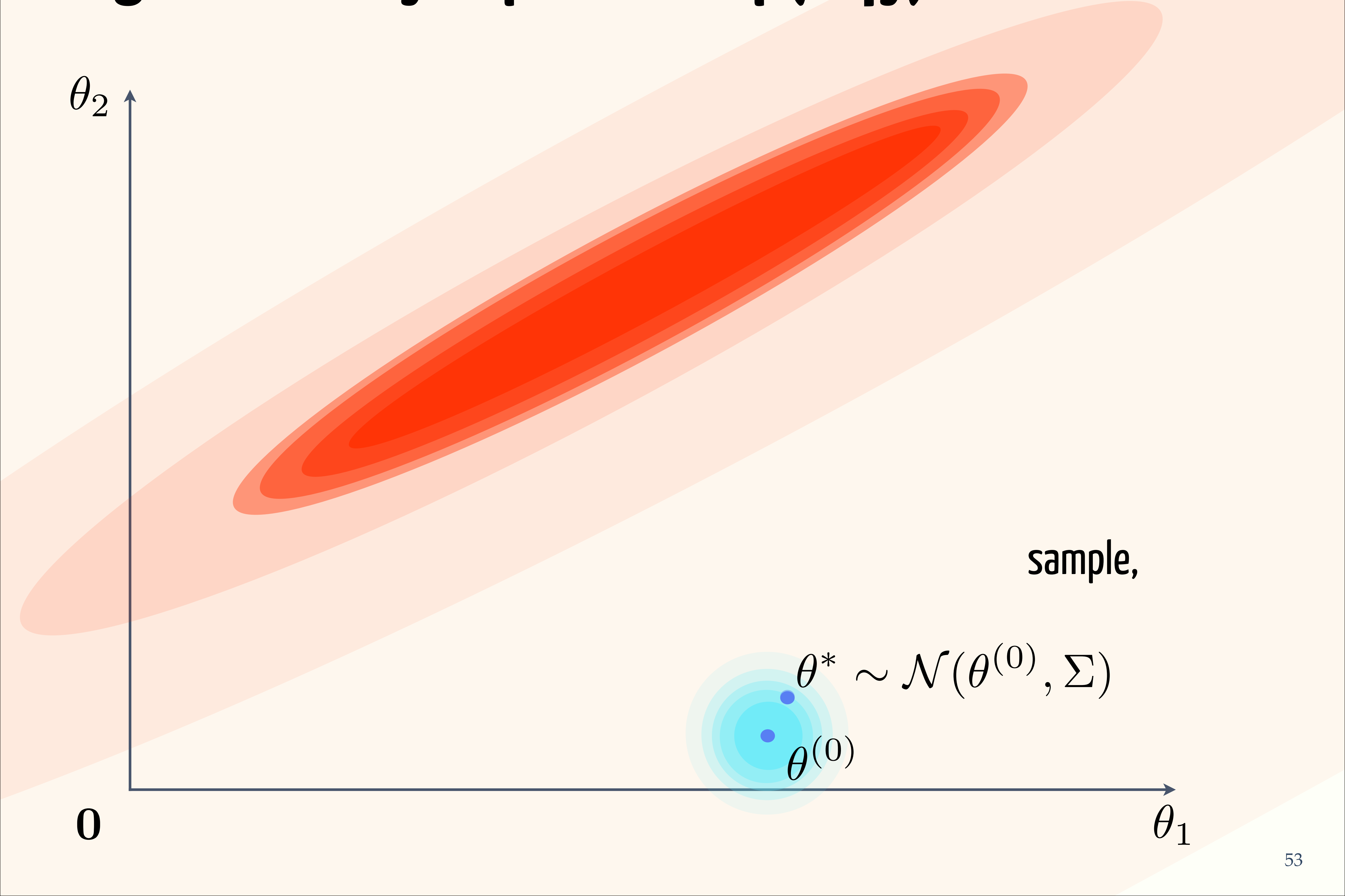
Markov Chain in the parameter space, generating samples that approximate $p(\theta|y)$

Marginal density exploration: $p(\Theta|y)$

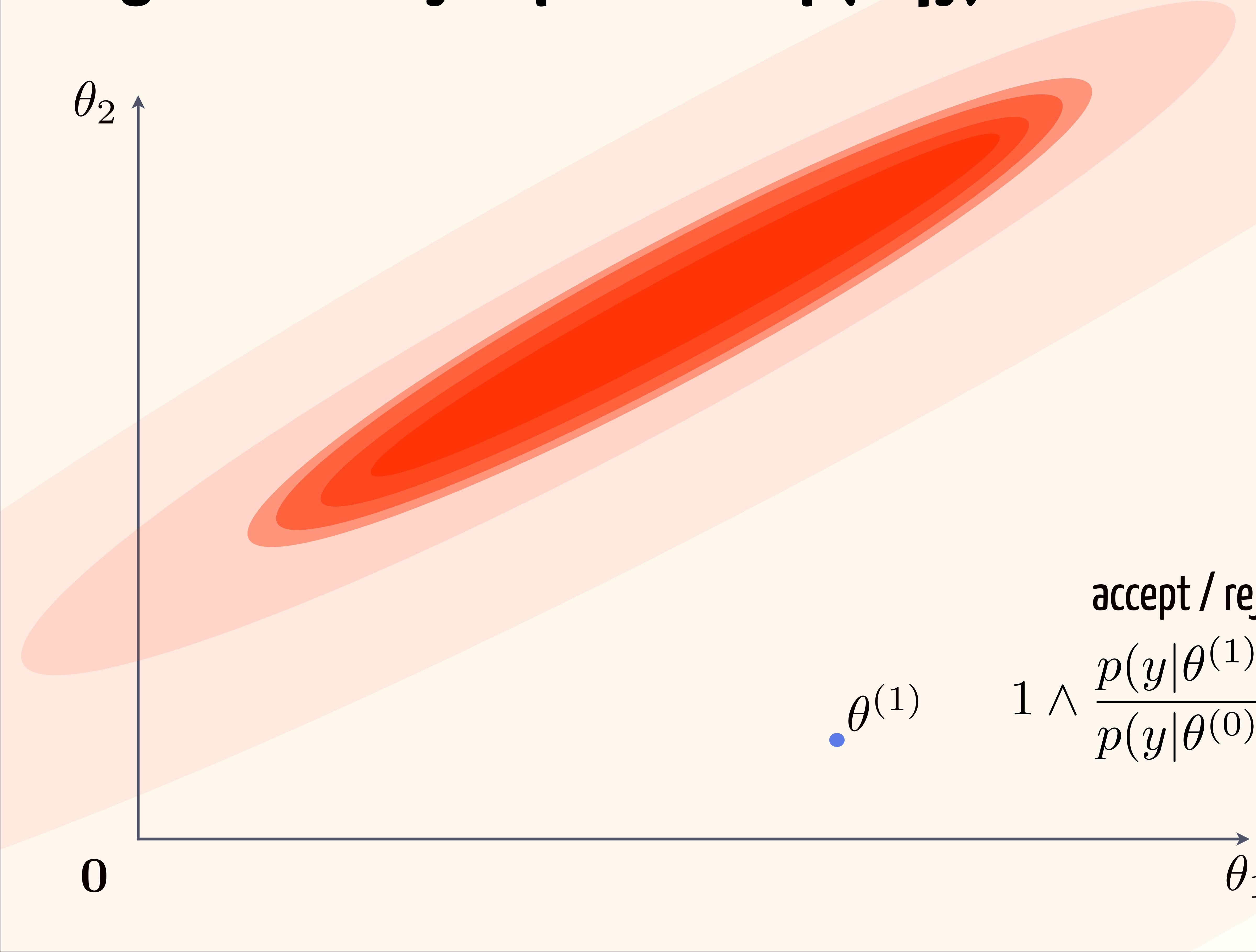


Initialise,

Marginal density exploration: p(Θ|y)



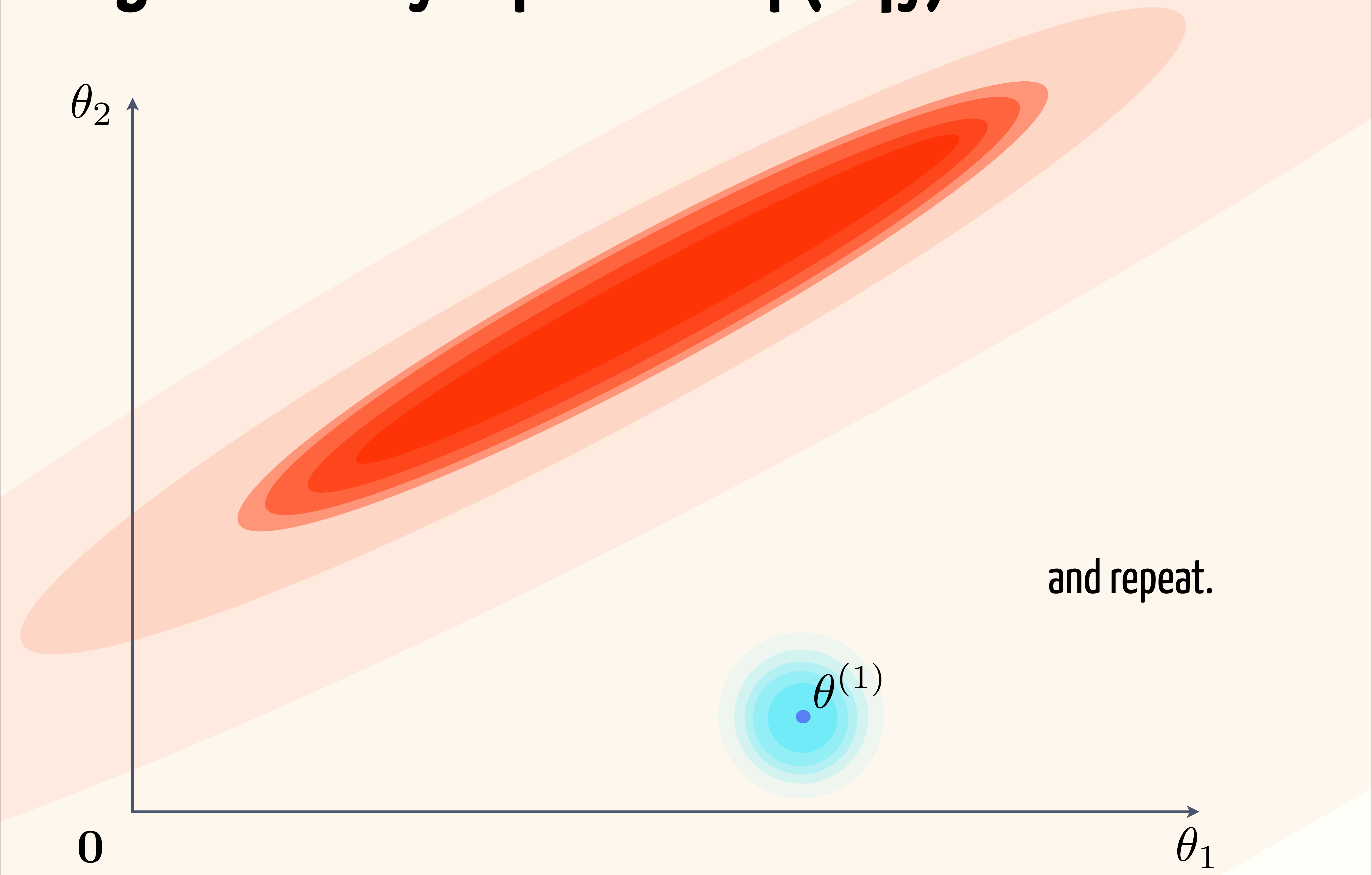
Marginal density exploration: p(Θ|y)



accept / reject,

$$1 \wedge \frac{p(y|\theta^{(1)})p(\theta^{(1)})}{p(y|\theta^{(0)})p(\theta^{(0)})}$$

Marginal density exploration: p(Θ|y)



and repeat.

MCMC algorithm

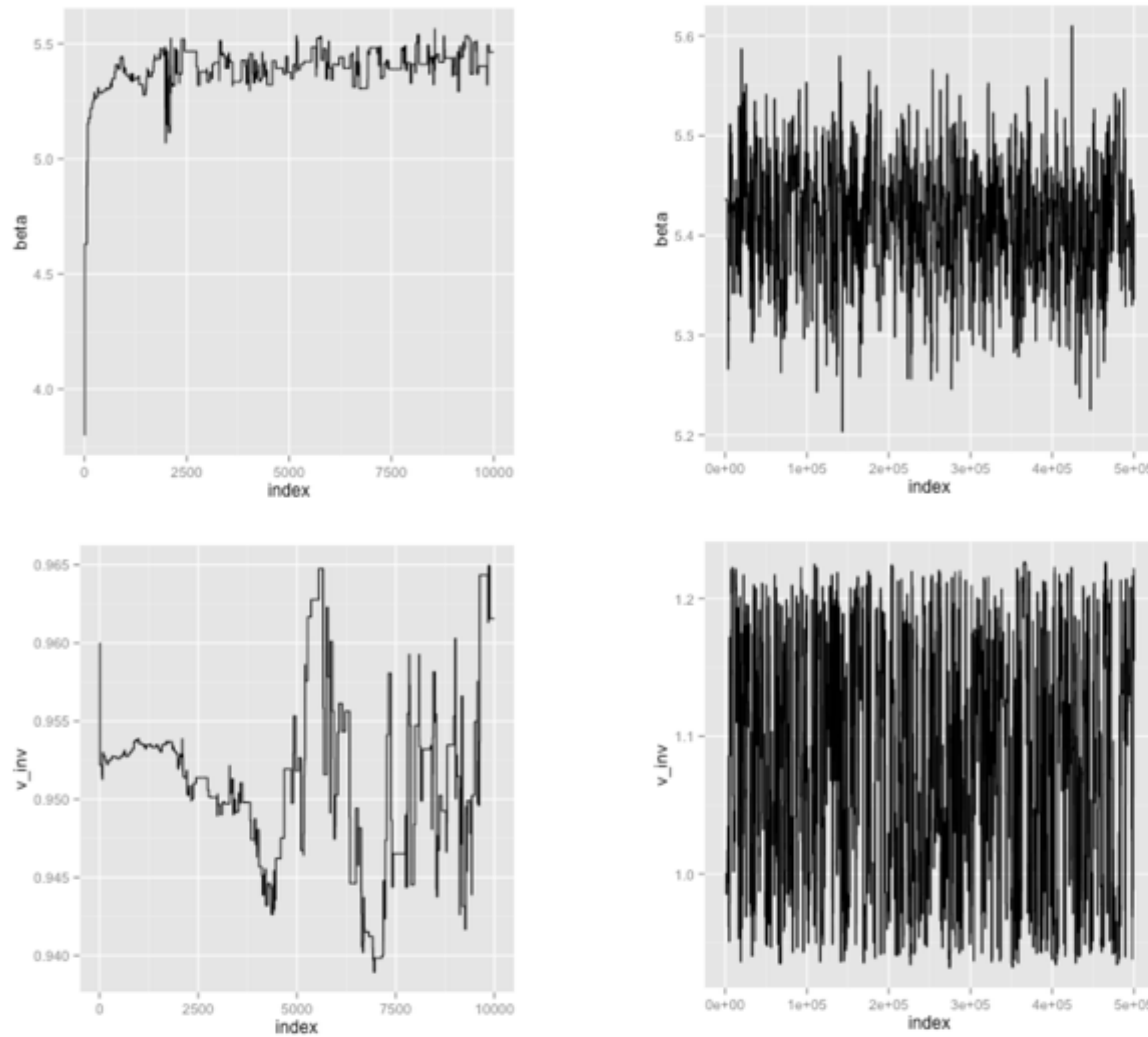
Number of iterations

```
cat ../theta.json | ./pmcmc ode -M 10000 --trace
```

Deterministic version of the algorithm

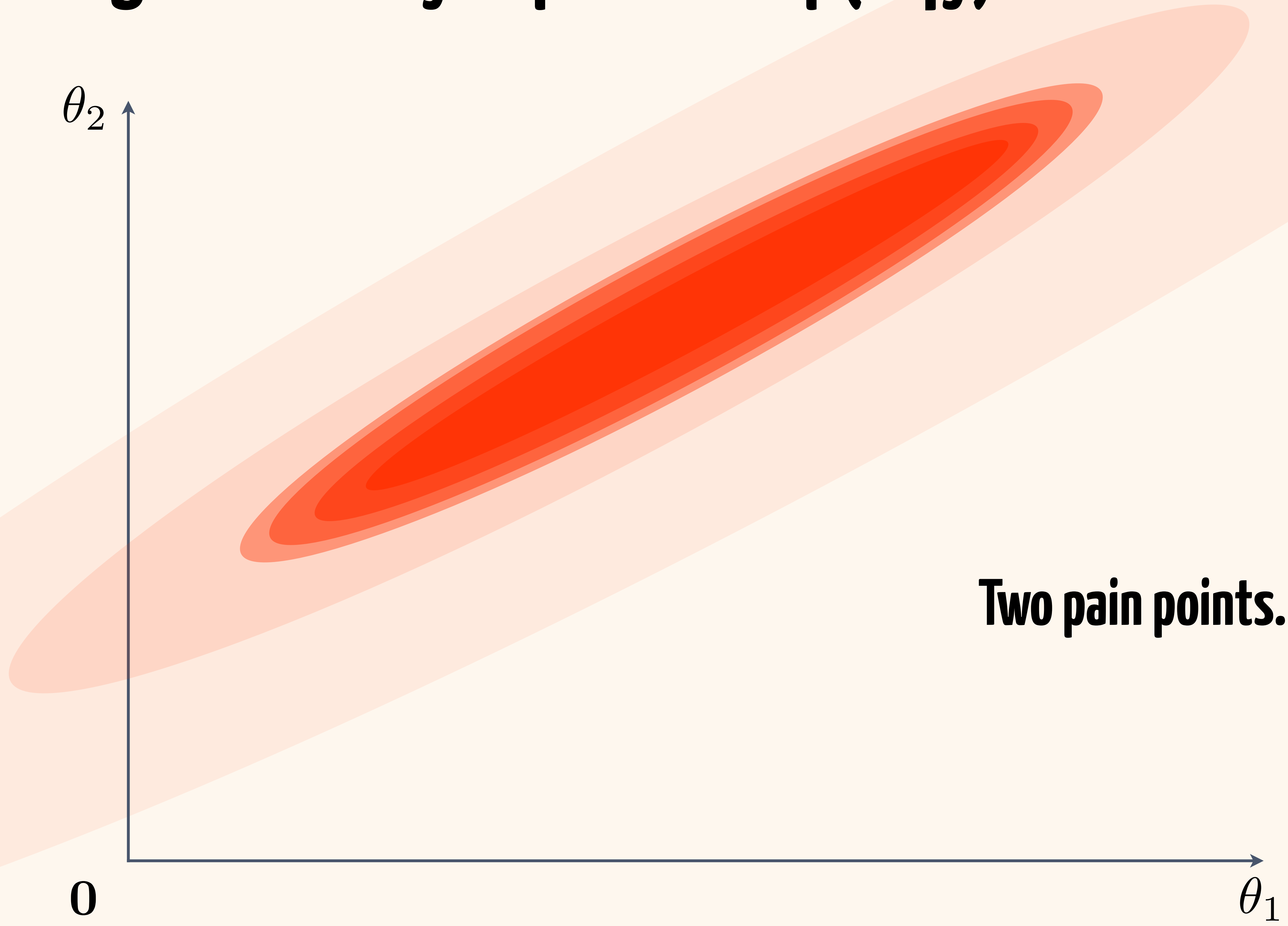
Keep the traces

Run 10000 iterations
And plot the traces of the MCMC



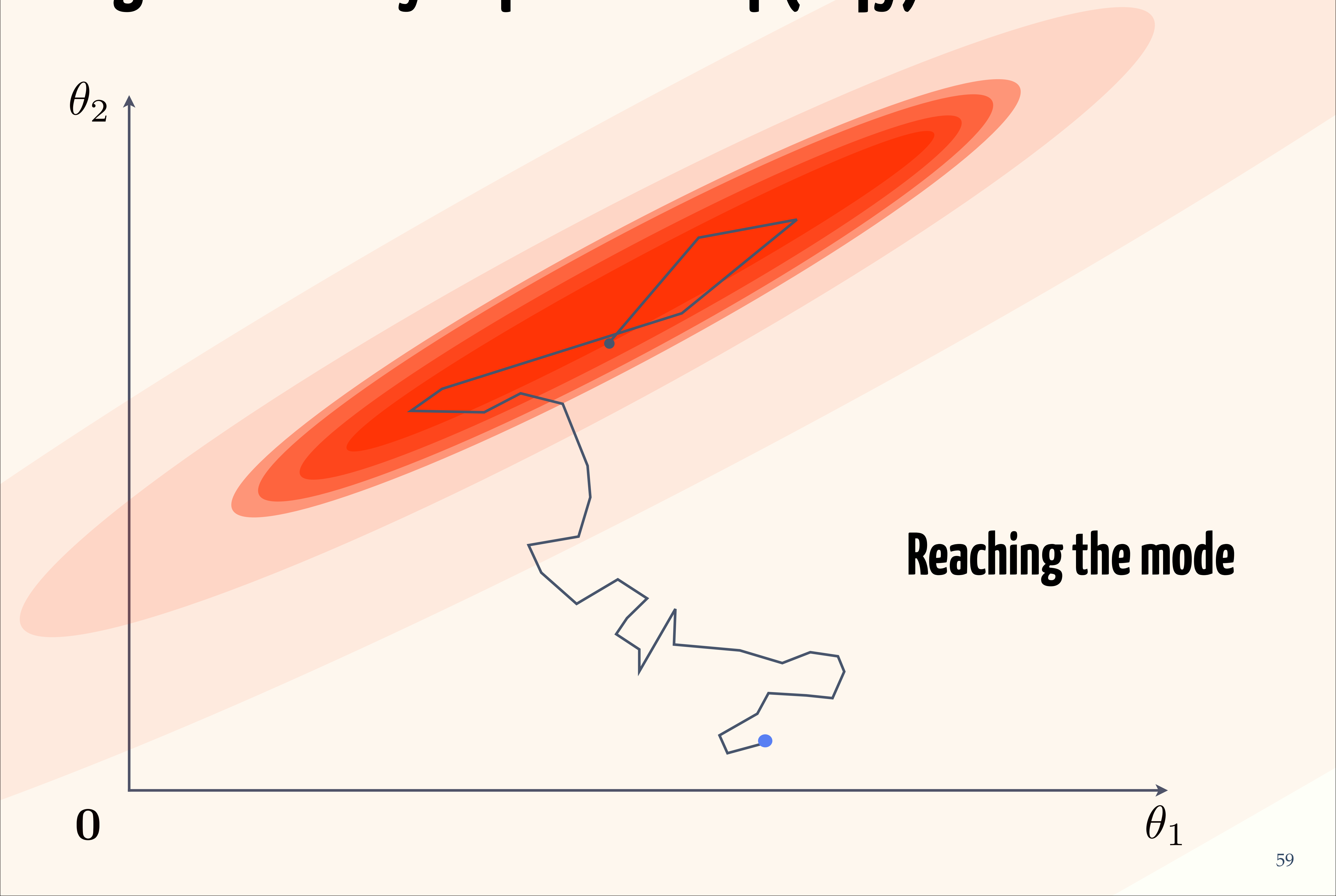
Ideally:
Independent and identically distributed samples

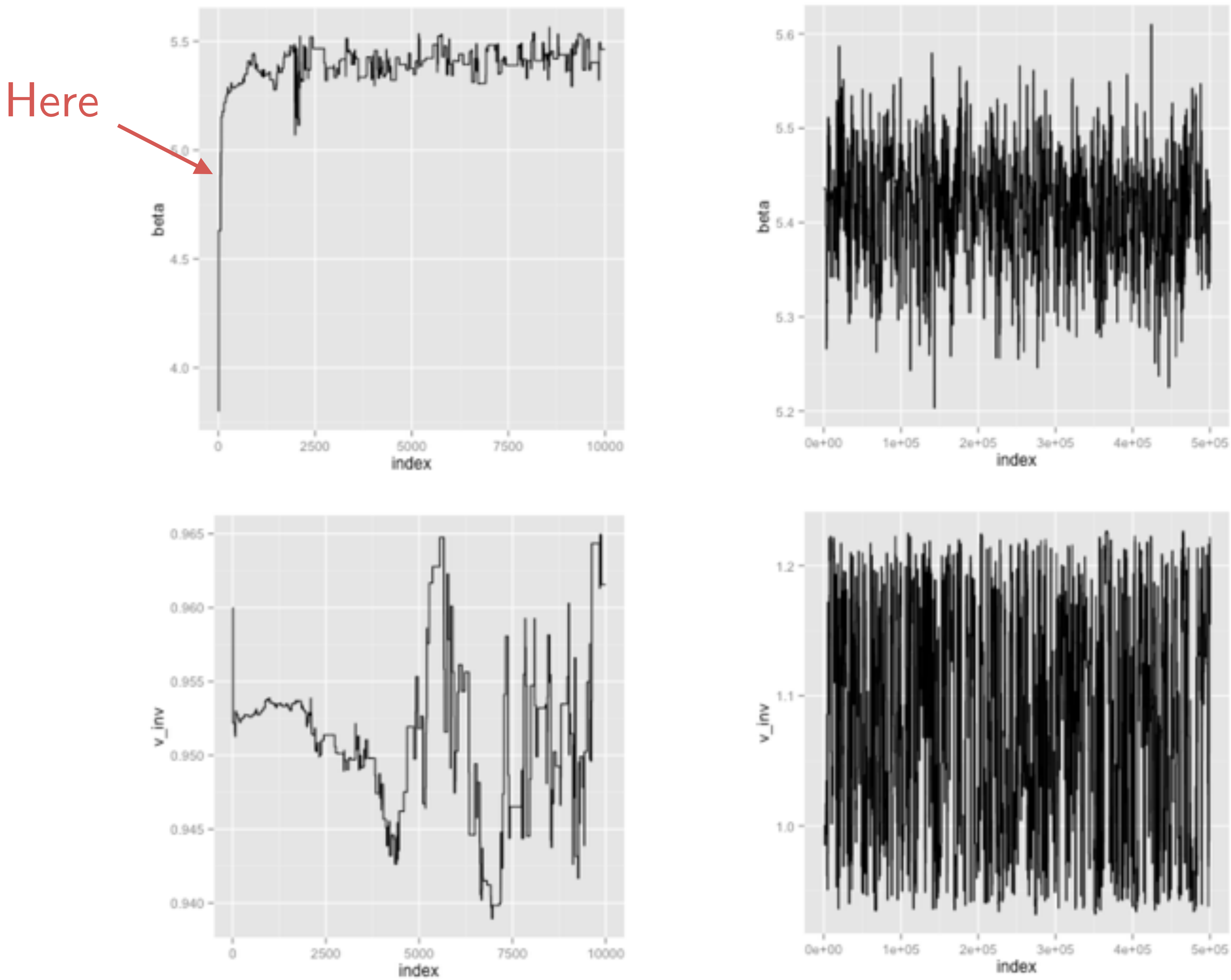
Marginal density exploration: p(θ|y)



Two pain points.

Marginal density exploration: p(Θ|y)

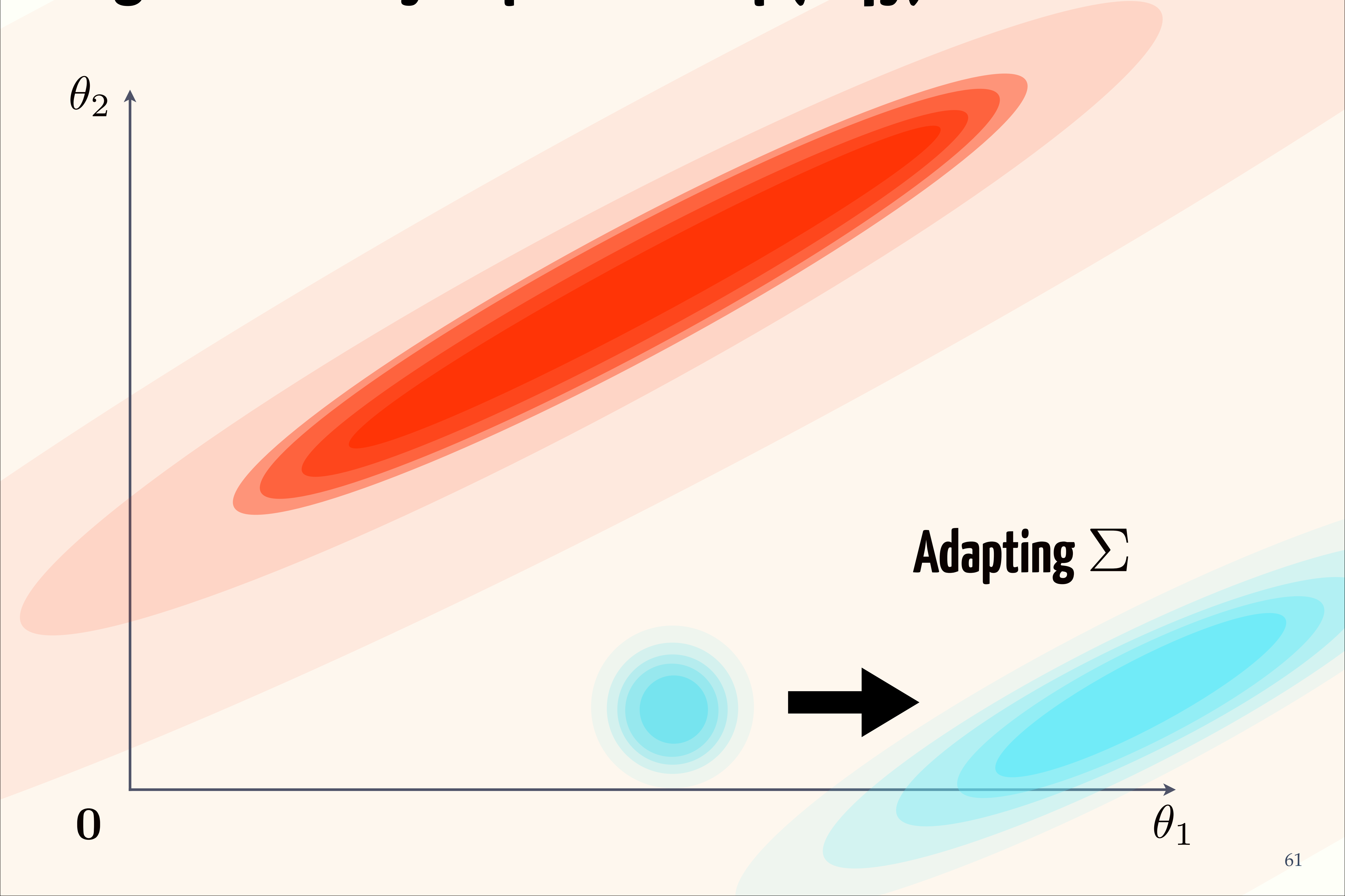




Ideally:

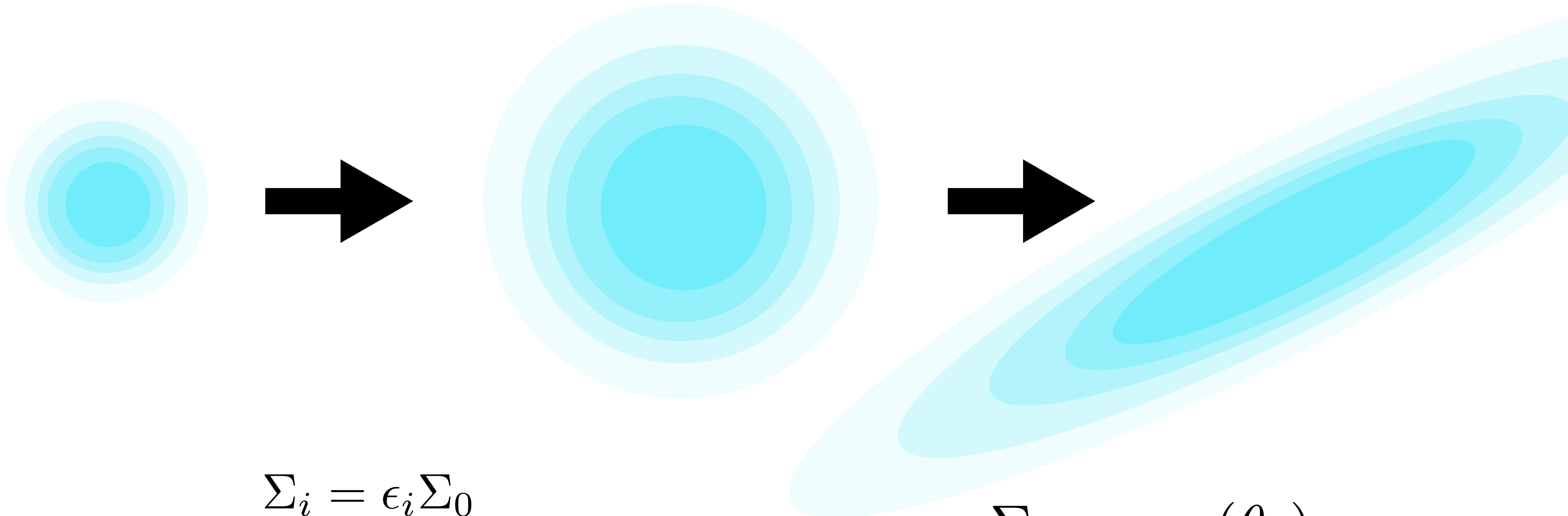
Independent and identically distributed samples

Marginal density exploration: p(Θ|y)



Size

Shape

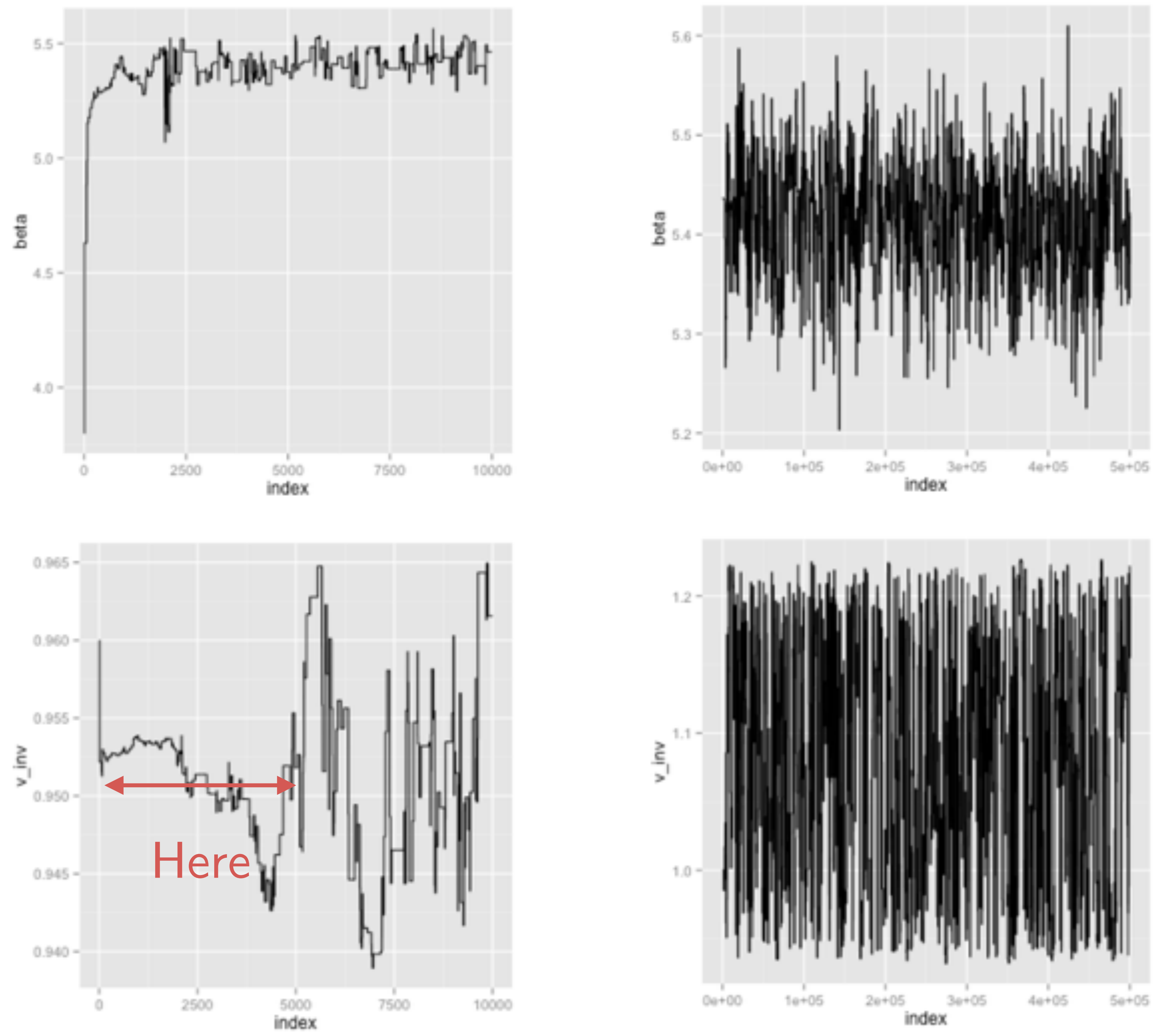


$$\Sigma_i = \epsilon_i \Sigma_0$$

$$log(\epsilon_{i+1}) = log(\epsilon_i) + (AccRate - 0.23)^i$$

$$\Sigma_i = cov(\theta_j)_{j \leq i}$$

Covariance adaptation in 2 phases

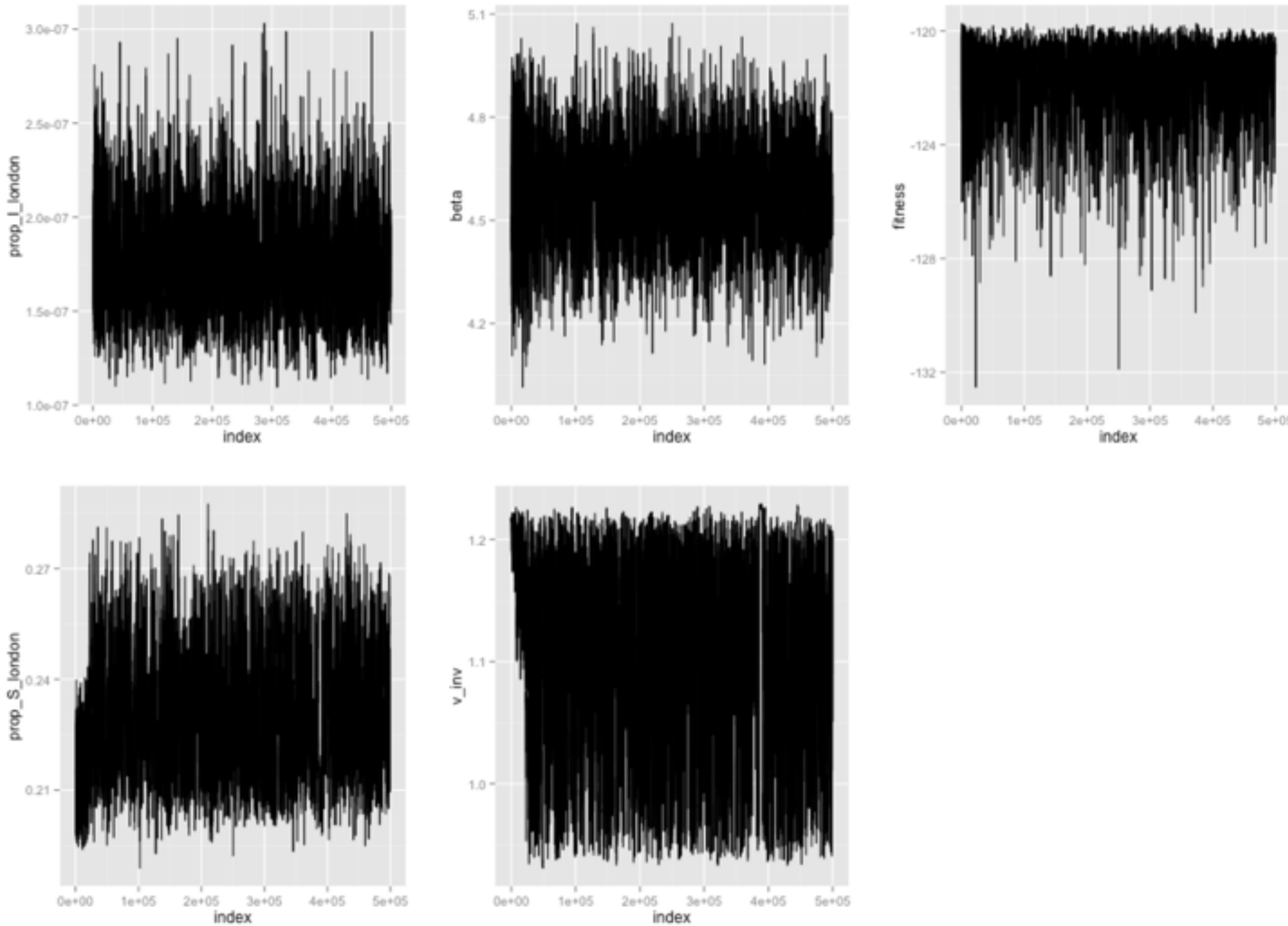


Ideally:
Independent and identically distributed samples

The output
theta is *piped*
into the following
algorithm

```
cat ../theta.json | ./pmcmc ode -M 500000 | ./pmcmc
ode -M 500000 -trace
```

Chaining
algorithms like playing with duplo blocks



Ideally:
Independent and identically distributed samples

Model creation
and
model selection

Estimate parameters

$$p(\boldsymbol{\theta}|y) = \int p(X_t, \boldsymbol{\theta}|y)\, dX_t$$

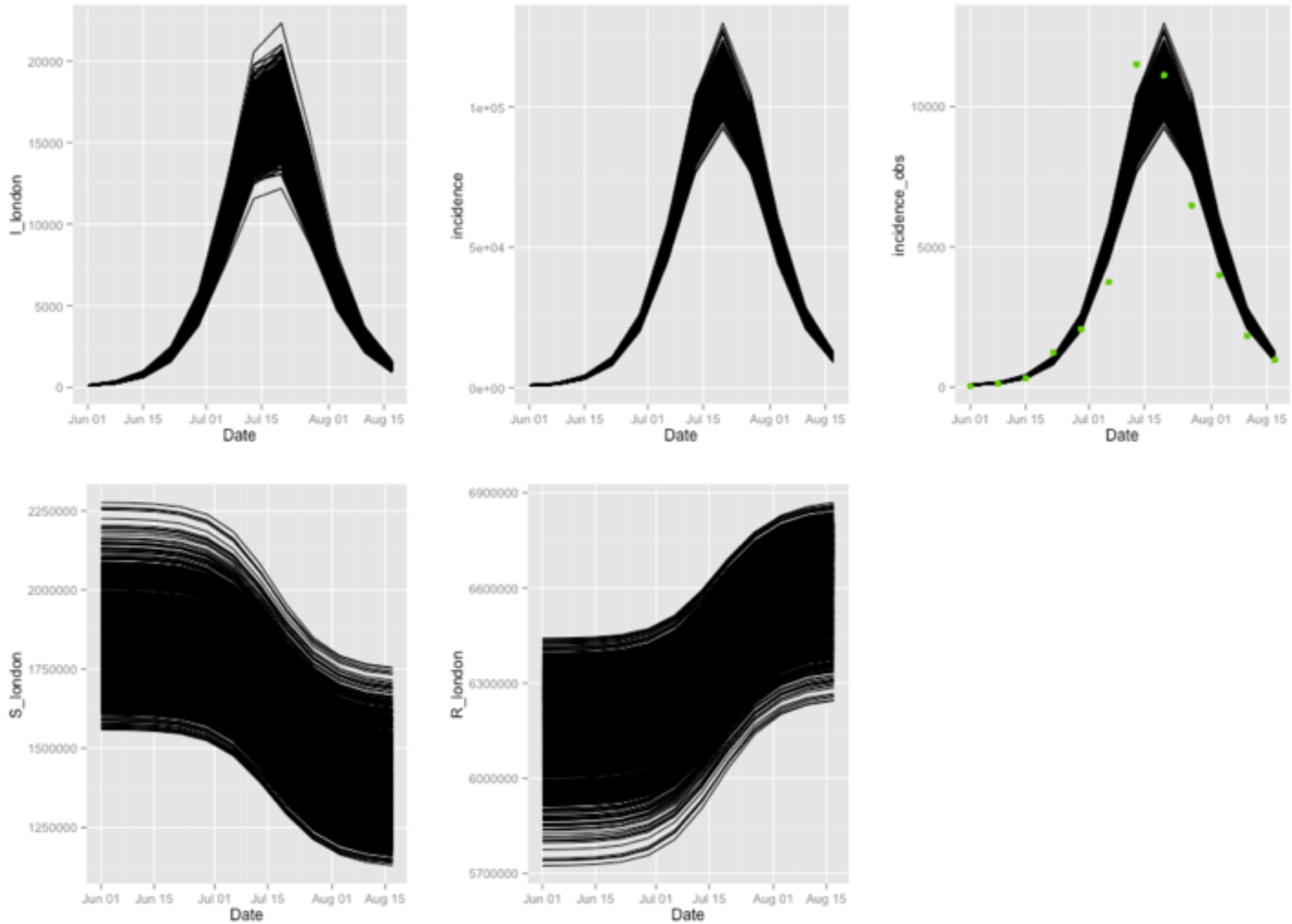
Explore $p(X_t, \boldsymbol{\theta}|y)$

Reconstruct past scenarios

$$p(X_t|y) = \int p(X_t, \boldsymbol{\theta}|y)\, d\boldsymbol{\theta}$$

Derive model choice indicators: $p(M1|y) > p(M2|y)$

Objectives of Bayesian inference



Can we improve the SIR fit?

What about...

...holidays?

...incubation period?

Deviance Information Criterion

Deviance

$$D(\theta) = -2log(p(y|\theta)) + C$$

Proxy of the number
of independent parameters

$$p_D = \frac{1}{2}\widehat{var}(D(\theta))$$

$$DIC = D(\bar{\theta}) + 2p_D$$

A penalized likelihood
criterion

Comparing models

	SIR	SEIR	SIR-holidays
DIC	237		

Build your own SEIR model

`TODO:`

- load prior
- define additional reaction
- modify theta
- compile
- run

```
cat theta.json | ./pmcmc ode -M 500000 | ./pmcmc ode -M 500000 --trace
```

What about...

	SIR	SEIR	SIR-holidays
DIC	237	**241**	

SIR-holidays

TODO:

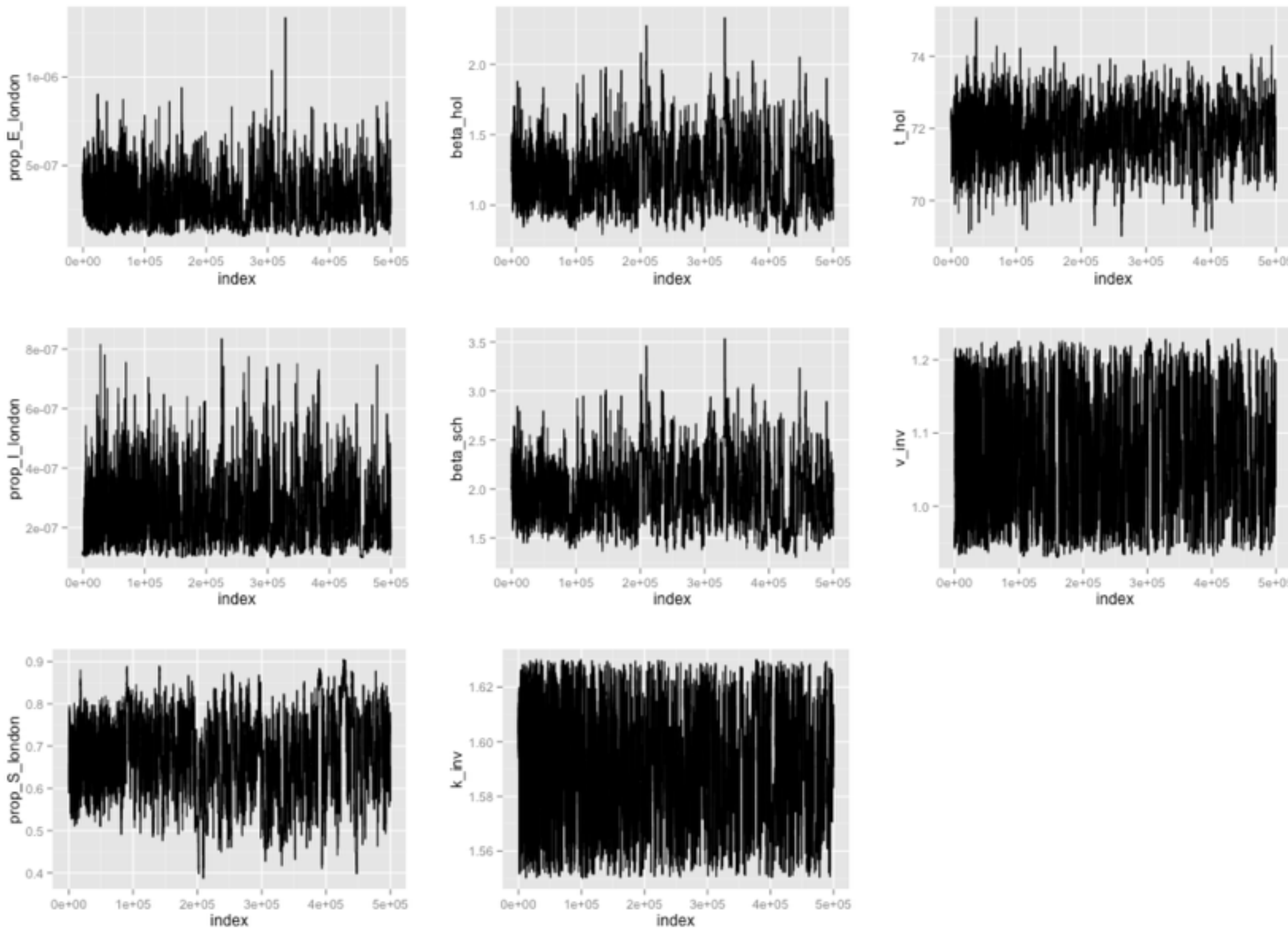
- load prior
- define additional reaction
- modify theta
- compile
- run

```
cat theta.json | ./pmcmc ode -M 500000 | ./pmcmc ode -M 500000 --trace
```

What about...

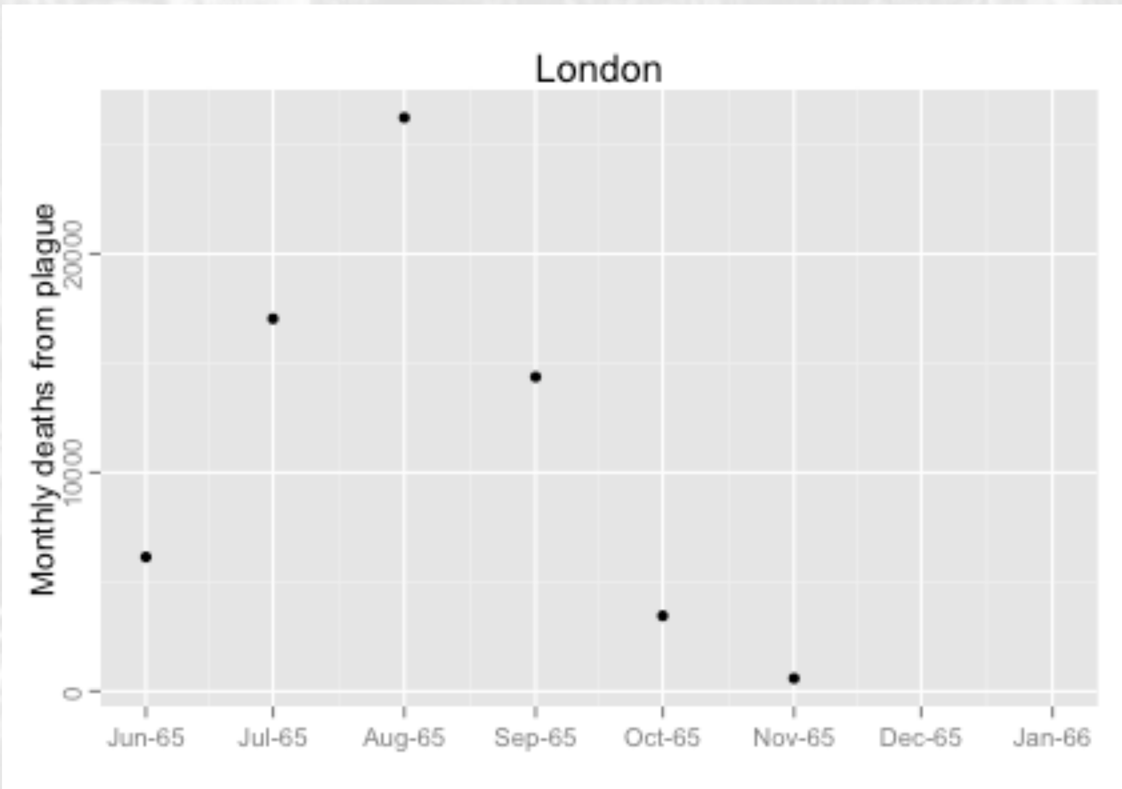
	SIR	SEIR	SIR-holidays
DIC	237	241	**211**

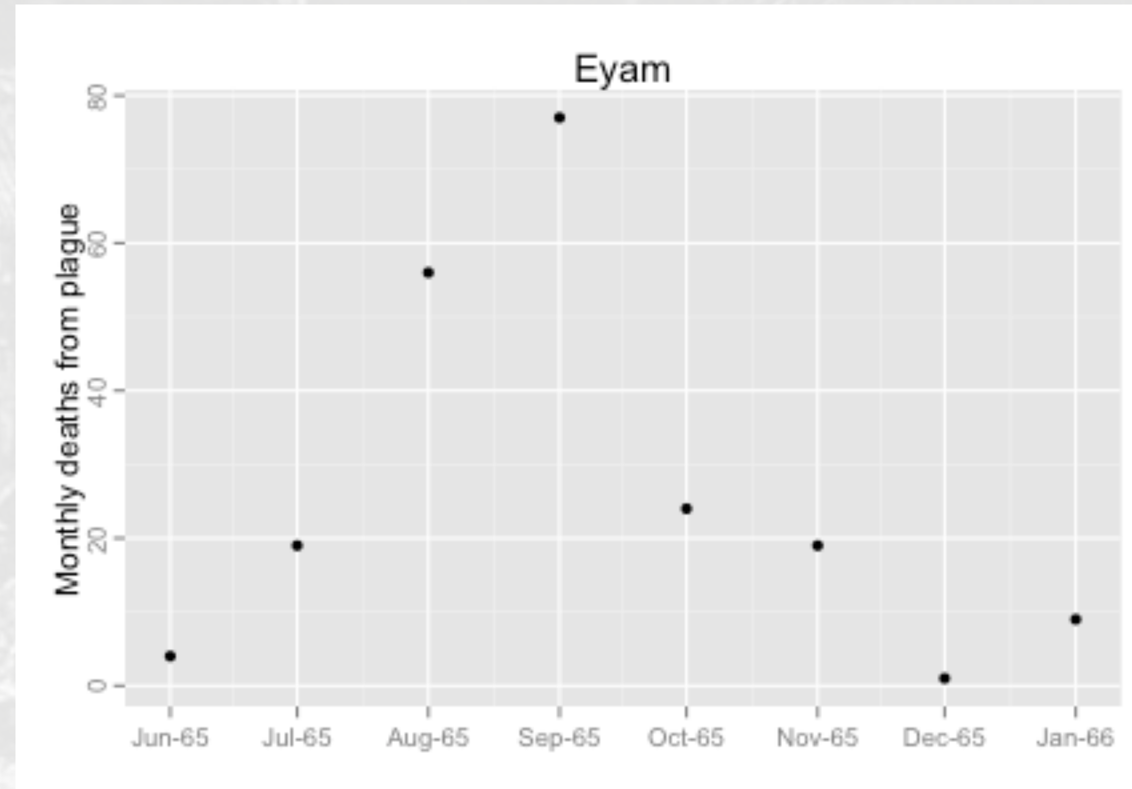
SIR-holidays



Stochasticity and statistical inference

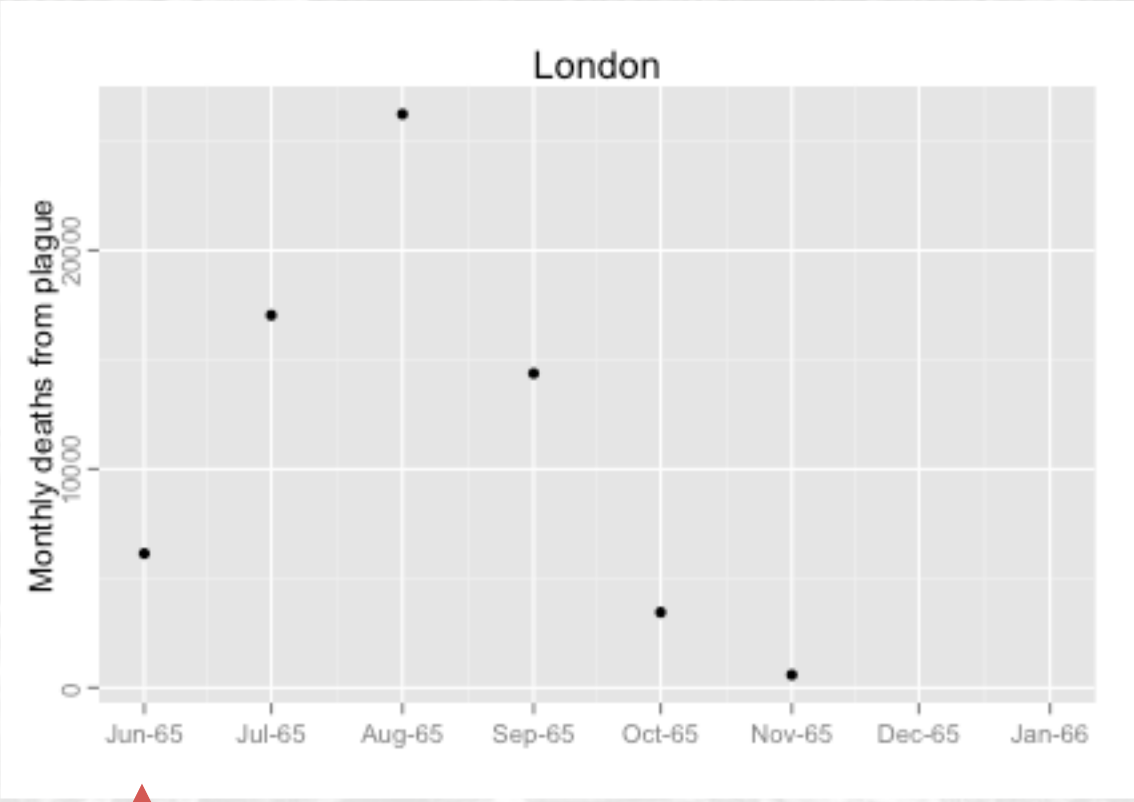
1665 plague in England

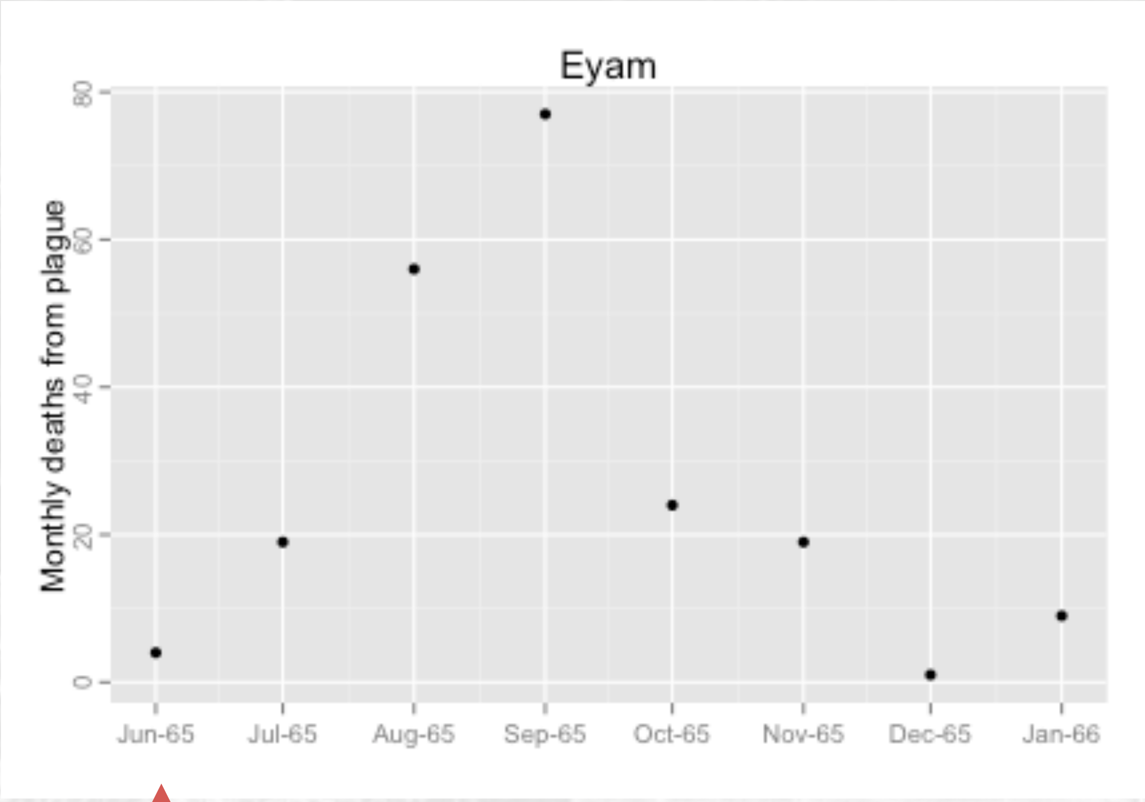




What was life expectancy with plague?
Were infection patterns similar in London and Eyam?

life_exp? r0_London? r0_Eyam?





460 000 inhabitants

350 inhabitants

Not really infinite populations….

ode: ordinary differential equation

sde: stochastic differential equation

psr: poisson with stochastic rates

Different mathematical formalisms

```
cat ../theta.json |./simul --traj --hat

cat ../theta.json |./simul ode --traj --hat

cat ../theta.json |./simul sde --traj --hat

cat ../theta.json |./simul sde -J 2 --traj --hat

cat ../theta.json |./simul sde -J 100 --traj --hat

cat ../theta.json |./simul psr -J 100 --traj --hat
```

Different mathematical formalisms

Compartmental description of the epidemic dynamic

S_t : number of individuals susceptible to be infected
I_t : number of individuals infected

Model:

Independently of the past, S individual transfers to I at rate $\beta_t \dfrac{I_t}{N}$, where $\beta_t = R_0 \mu_D [1 + e \sin(t + \phi)]$

In other words:

$$P([S_{t+dt}, I_{t+dt}] = [S_t - 1, I_t + 1] | [S_t, I_t]) = \beta_t S_t \frac{I_t}{N} + o(dt)$$

$$P([S_{t+dt}, I_{t+dt}] = [S_t, I_t] | [S_t, I_t]) = 1 - \beta_t S_t \frac{I_t}{N} + o(dt)$$

(Markovian jump process)

Compartmental description of the epidemic dynamic

S_t : number of individuals susceptible to be infected
I_t : number of individuals infected

Model:

Independently of the past, S individual transfers to I at rate $\beta_t \frac{I_t}{N}$, where $\beta_t = R_0 \mu_D [1 + e \sin(t + \phi)]$

In other words:

$$P([S_{t+dt}, I_{t+dt}] = [S_t - 1, I_t + 1] | [S_t, I_t]) = \beta_t S_t \frac{I_t}{N} + o(dt)$$

$$P([S_{t+dt}, I_{t+dt}] = [S_t, I_t] | [S_t, I_t]) = 1 - \beta_t S_t \frac{I_t}{N} + o(dt)$$

Intractable for other than small populations

Alternative formulations

Multinomial approximation:

If n is the number of reaction occurrences during $[t, t+dt[$,

$$p = 1 - \exp(-\beta_t S_t \frac{I_t}{N} dt)$$

$$P(n|z_t) = \begin{pmatrix} S_t \\ n \ S_t - n \end{pmatrix} (1-p)^{S_t - n} p^n + o(dt)$$

SDE approximation:

Using the diffusion approximation (large populations),

$$dS_t = -\beta_t S_t \frac{I_t}{N} - \sqrt{\beta_t S_t \frac{I_t}{N}} dB_t$$

$$dI_t = \beta_t S_t \frac{I_t}{N} + \sqrt{\beta_t S_t \frac{I_t}{N}} dB_t$$

Breto et al. (2009). Time series analysis via mechanistic models.
Ethier, S. and Kurtz, T. (1986). Markov processes: characterization and convergence.

Estimate parameters

$$p(\boldsymbol{\theta}|y) = \int p(X_t, \boldsymbol{\theta}|y)\,dX_t$$

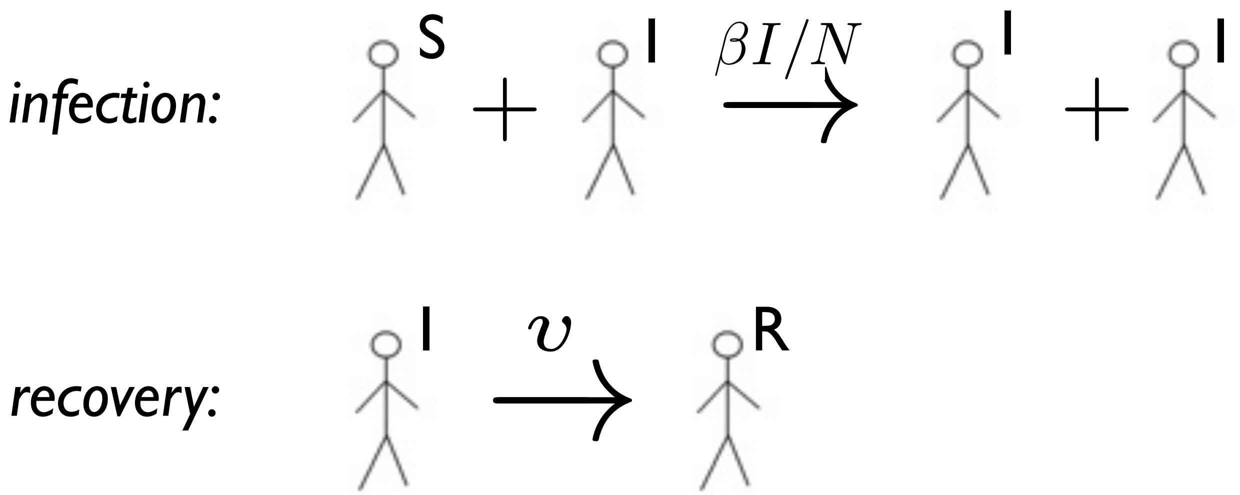
Explore $p(X_t, \boldsymbol{\theta}|y)$

Reconstruct past scenarios

$$p(X_t|y) = \int p(X_t, \boldsymbol{\theta}|y)\,d\boldsymbol{\theta}$$

Derive model choice indicators: $p(M1|y) > p(M2|y)$

Objectives of Bayesian inference



$$\frac{dS}{dt} = -\beta I/N \times S$$

$$\frac{dI}{dt} = \beta I/N \times S - vI$$

$$\frac{dR}{dt} = vI$$

In a
nice deterministic world....

Good approximation using
Euler or Runge-Kutta methods

Likelihood

$$p(y|\theta) = \prod p(y_i|X_{t_i}, \theta)p(X_{t_i}|\theta)$$

✓

✓

Model

In a
nice deterministic world....

Good approximation using
Euler or Runge-Kutta methods

Likelihood

$$p(y|\theta) = \prod p(y_i|X_{t_i}^{euler,\theta}, \theta) \times 1$$

In a
nice deterministic world....

No longer a dirac

Likelihood

$\otimes$

$$p(y|\theta) = \prod p(y_i|X_{t_i},\theta)p(X_{t_i}|\theta)$$

✓

Model

Not that simple in the stochastic world....

Discrete approximation
with *particles*

Likelihood

$$p(y|\theta) = \prod_i \Big(\sum_j p(y_i|X_{t_i}^{(j)}, \theta) p(X_{t_i}^{(j)}|\theta) \Big)$$

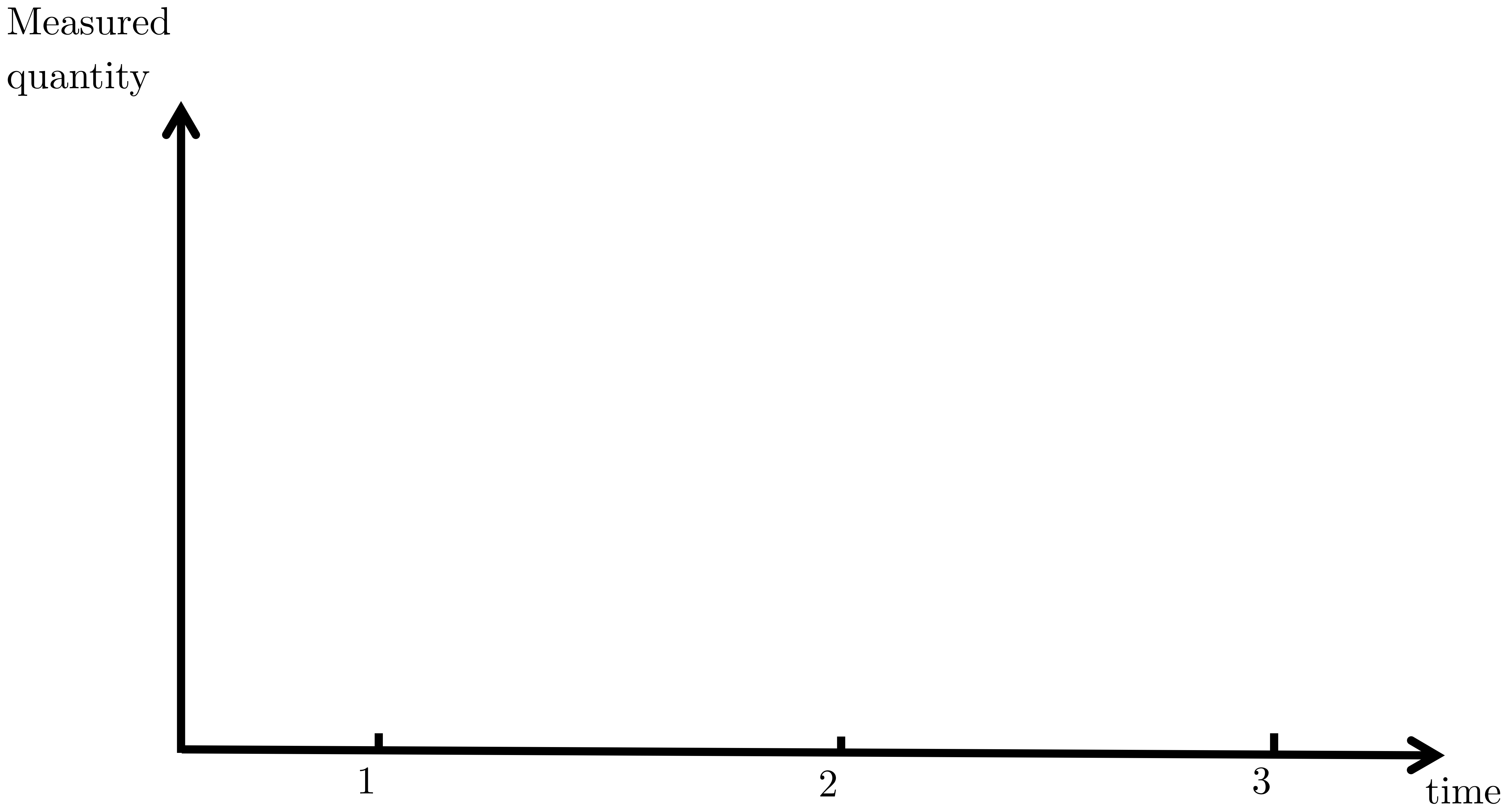
Not that simple
in the stochastic world....

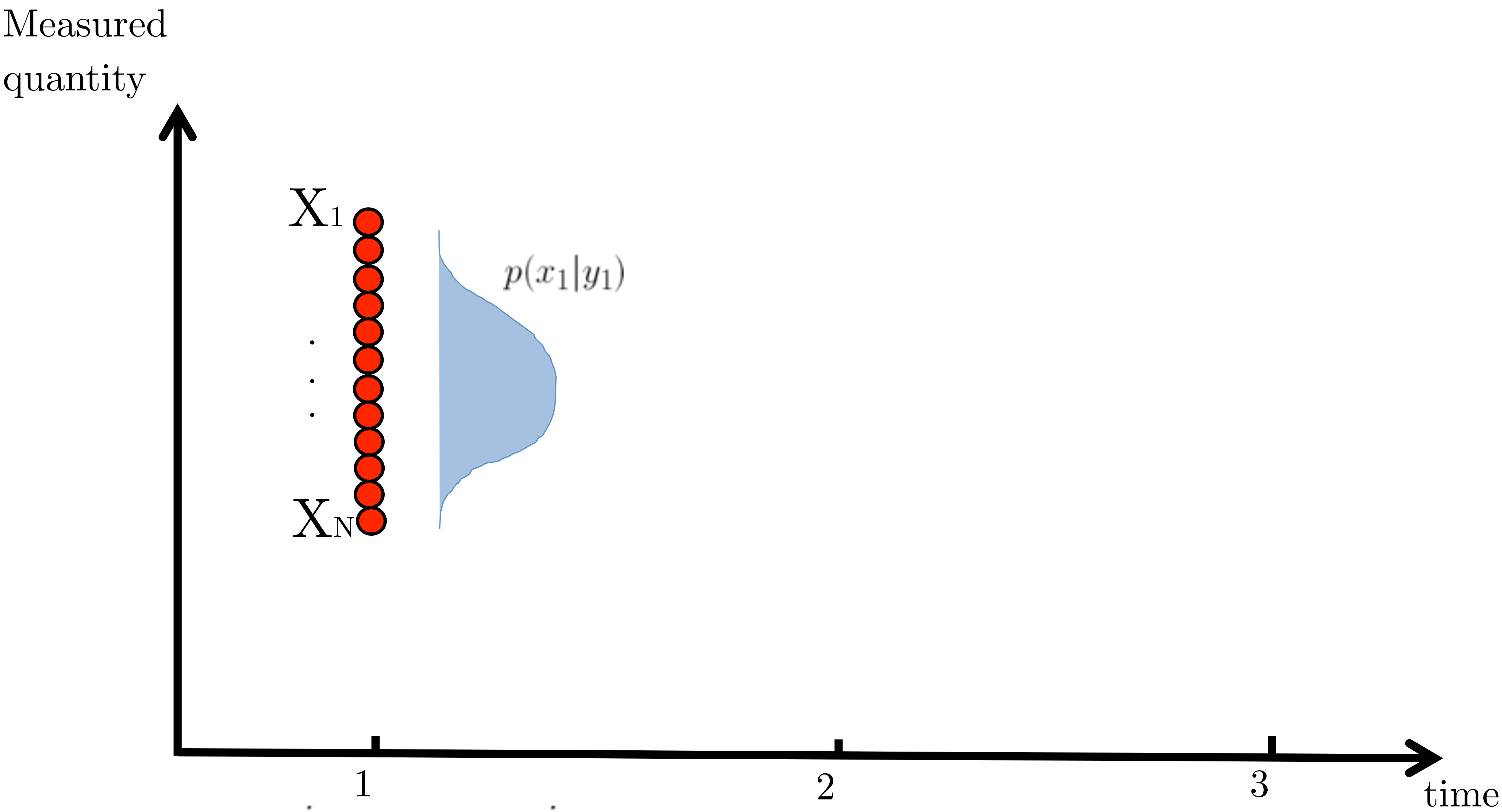
Objectives:

estimation of $p(y|\boldsymbol{\theta})$

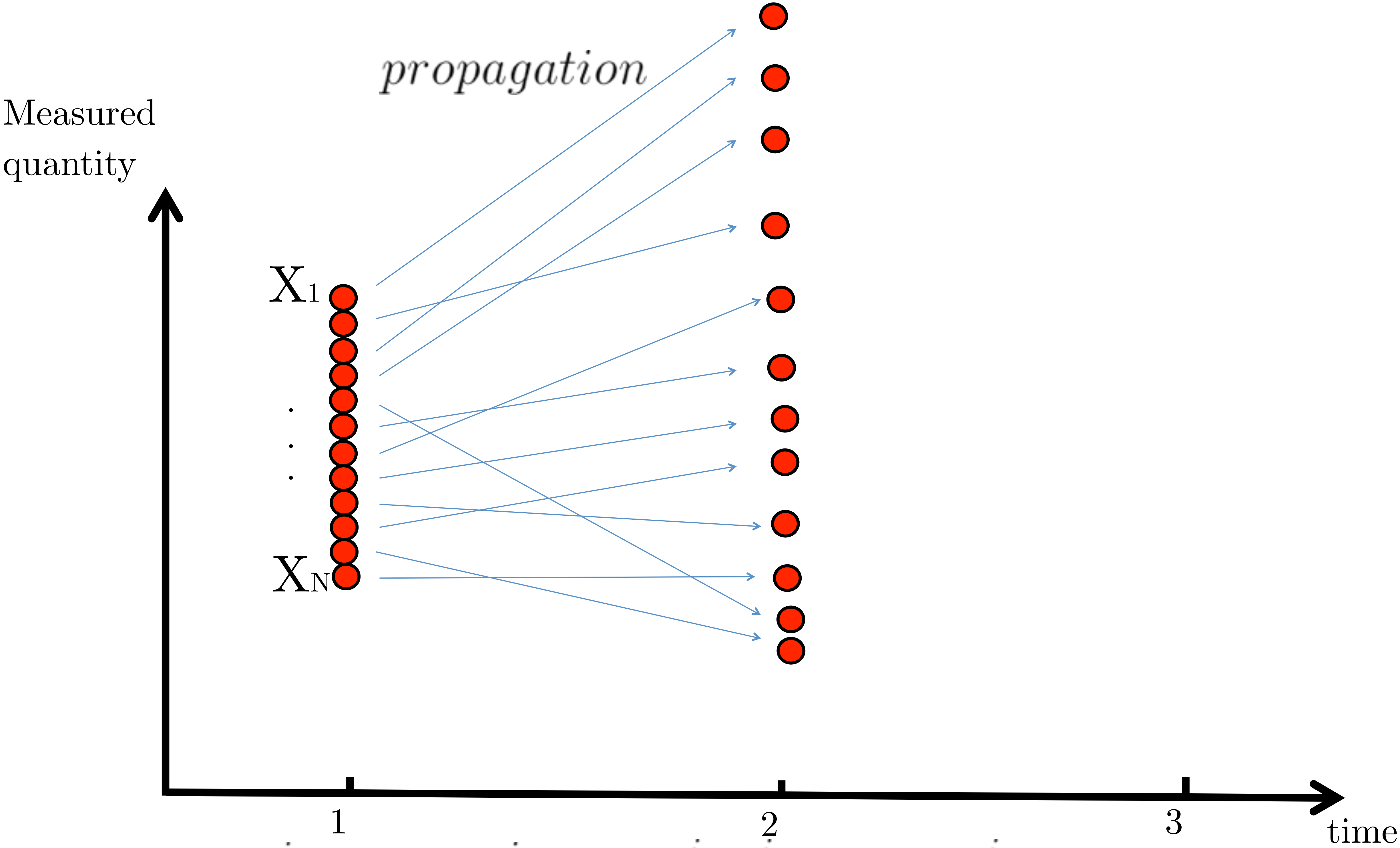
sample from $p(X_{0:n}|y,\boldsymbol{\theta})$

Introduction to particle filtering


Measured
quantity
1
2
3
time

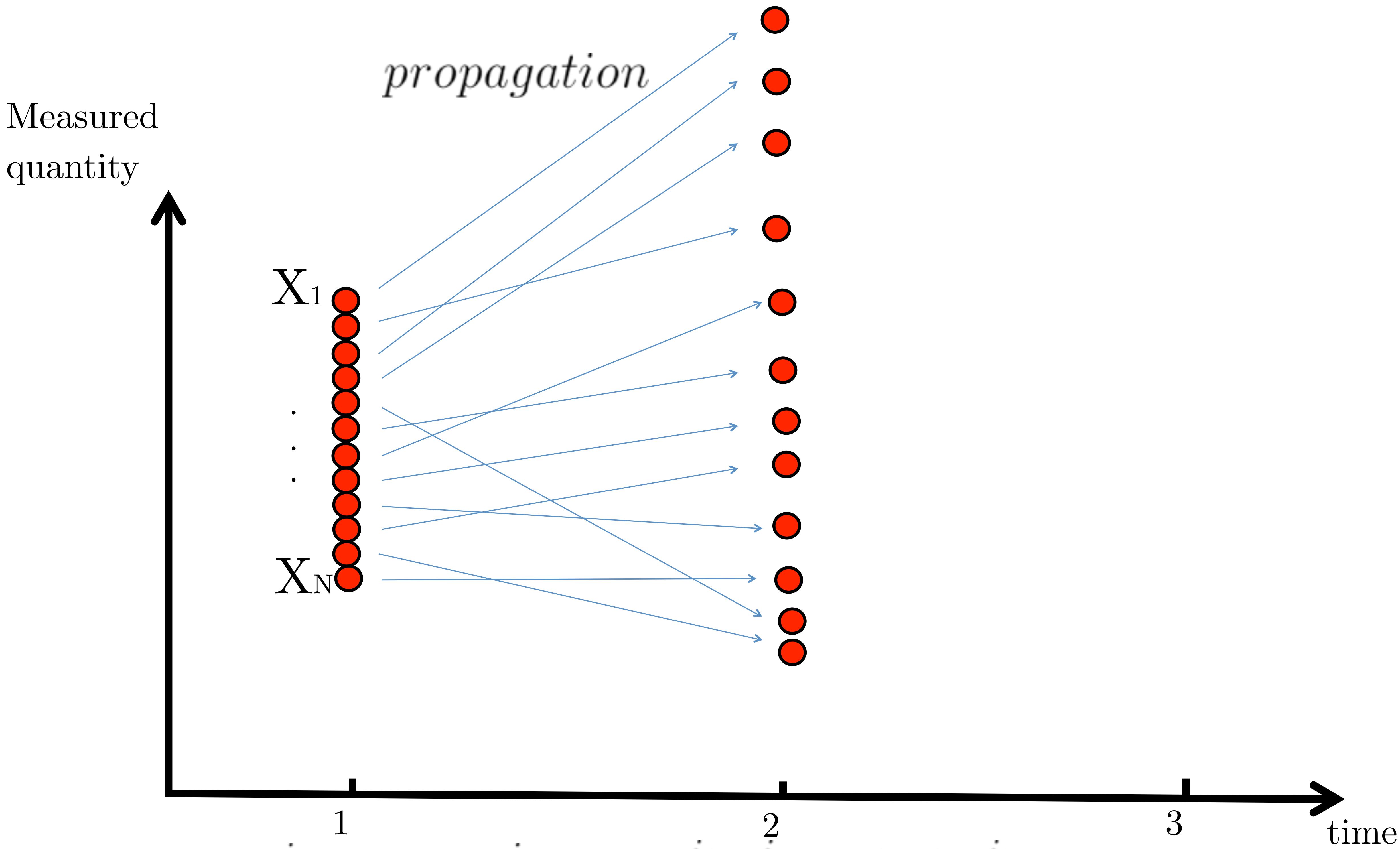


$$W_1^i = p(y_1|X_1^i)$$
$$(X_1^i, W_1^i) \sim p(x_1|y_1)$$



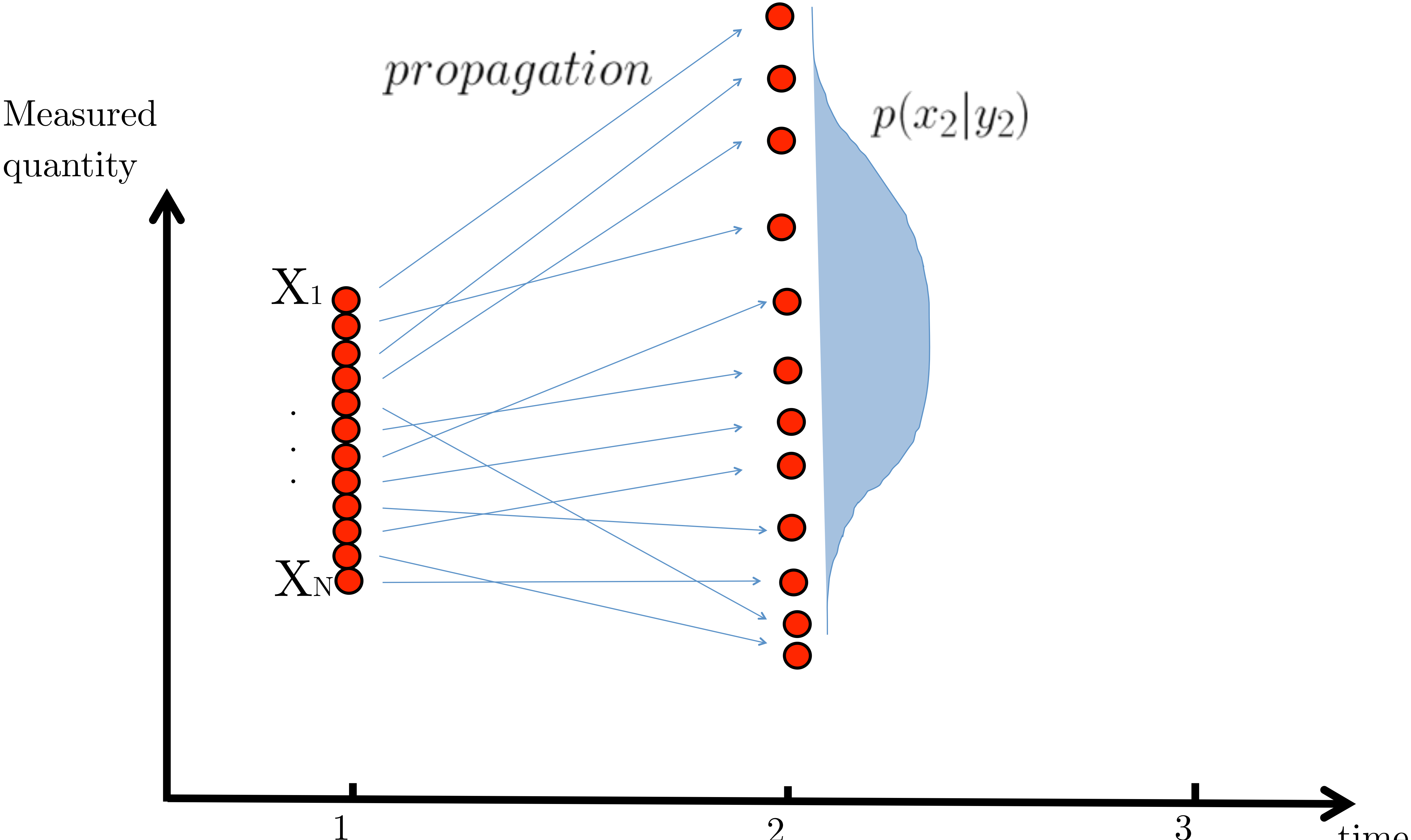
$$W_1^i = p(y_1 | X_1^i) \qquad X_2^i | X_1^i \sim p(x_2 | X_1^i)$$

$$(X_1^i, W_1^i) \sim p(x_1 | y_1)$$



$$W_1^i = p(y_1|X_1^i) \qquad X_2^i|X_1^i \sim p(x_2|X_1^i)$$

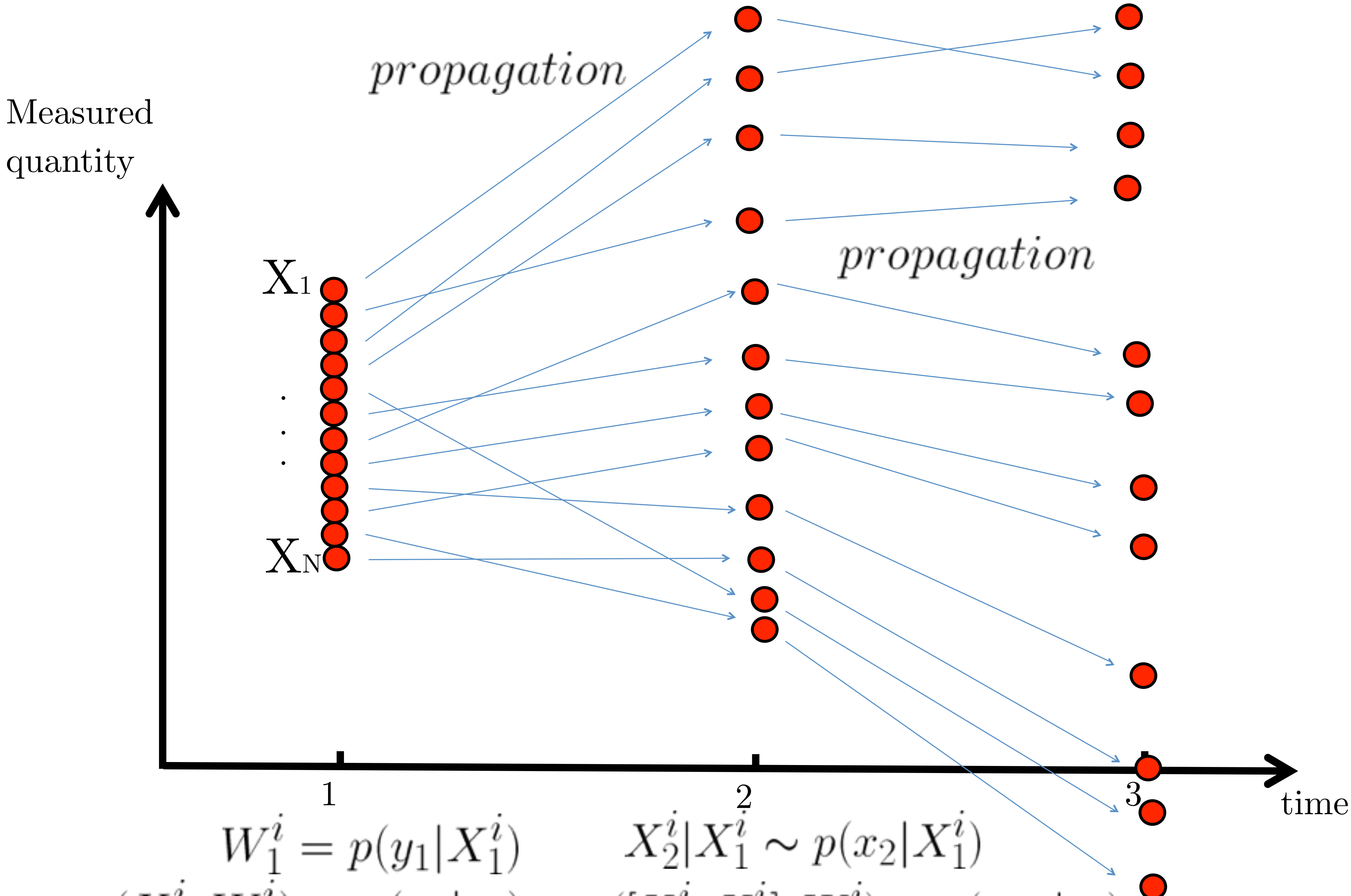
$$(X_1^i, W_1^i) \sim p(x_1|y_1) \qquad ([X_1^i, X_2^i], W_1^i) \sim p(x_{1:2}|y_1)$$



$$W_1^i = p(y_1|X_1^i)$$
$$(X_1^i, W_1^i) \sim p(x_1|y_1)$$

$$X_2^i|X_1^i \sim p(x_2|X_1^i)$$
$$([X_1^i, X_2^i], W_1^i) \sim p(x_{1:2}|y_1)$$

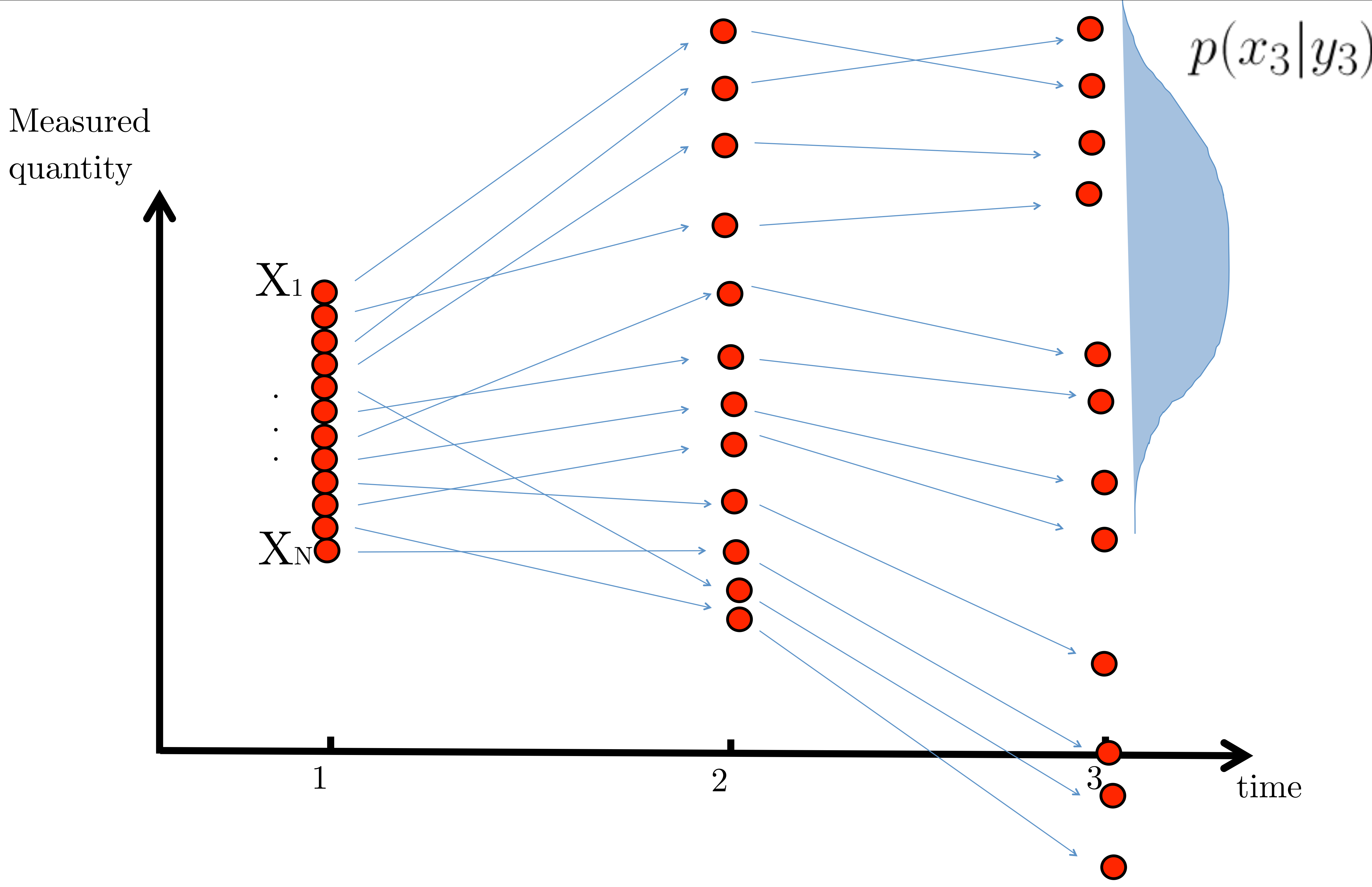
$$W_2^i = W_1^i \times p(y_2|X_2^i)$$
$$([X_1^i, X_2^i], W_2^i) \sim p(x_{1:2}|y_{1:2})$$

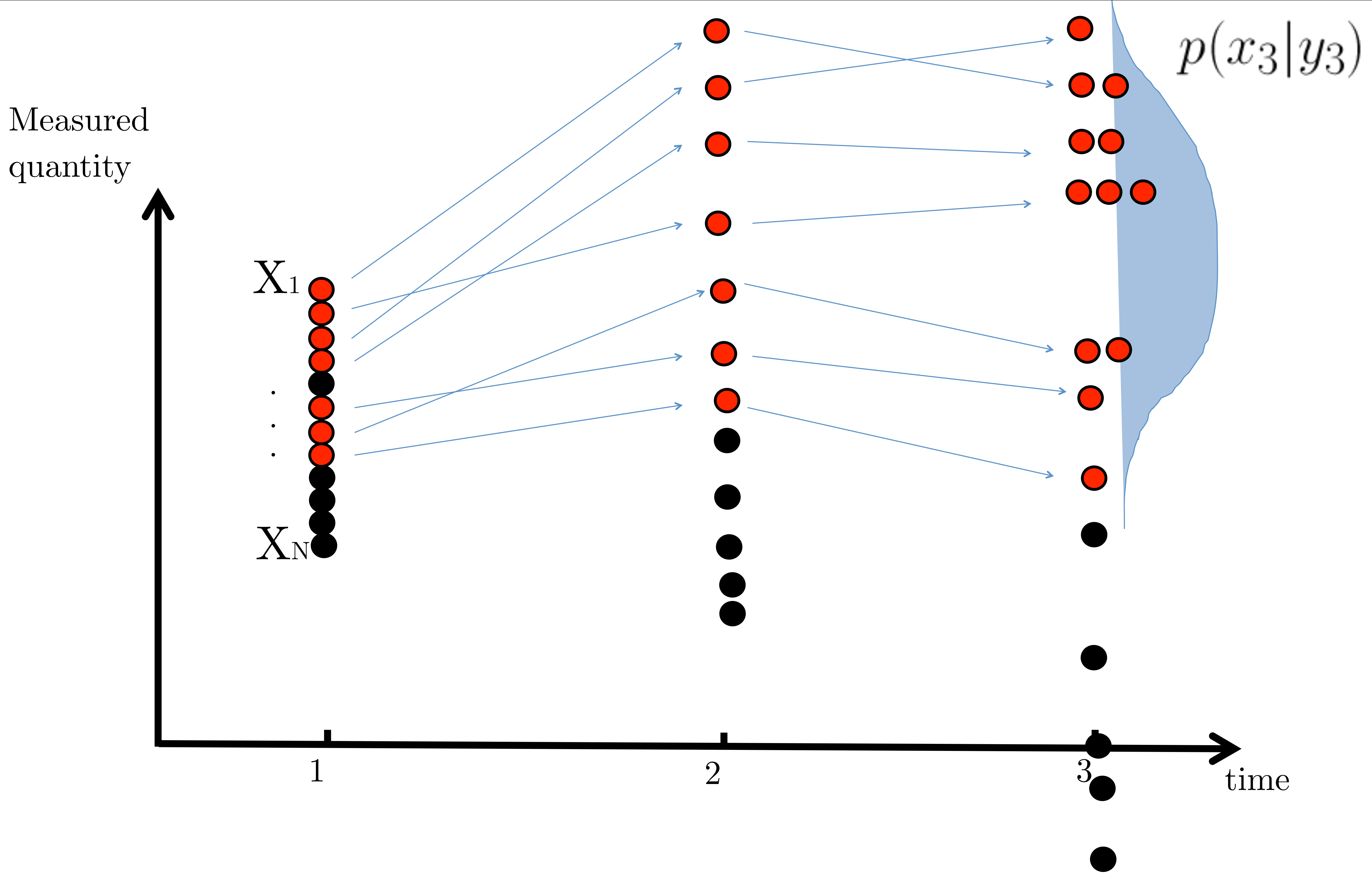


$$W_1^i = p(y_1|X_1^i)$$
$$(X_1^i, W_1^i) \sim p(x_1|y_1)$$

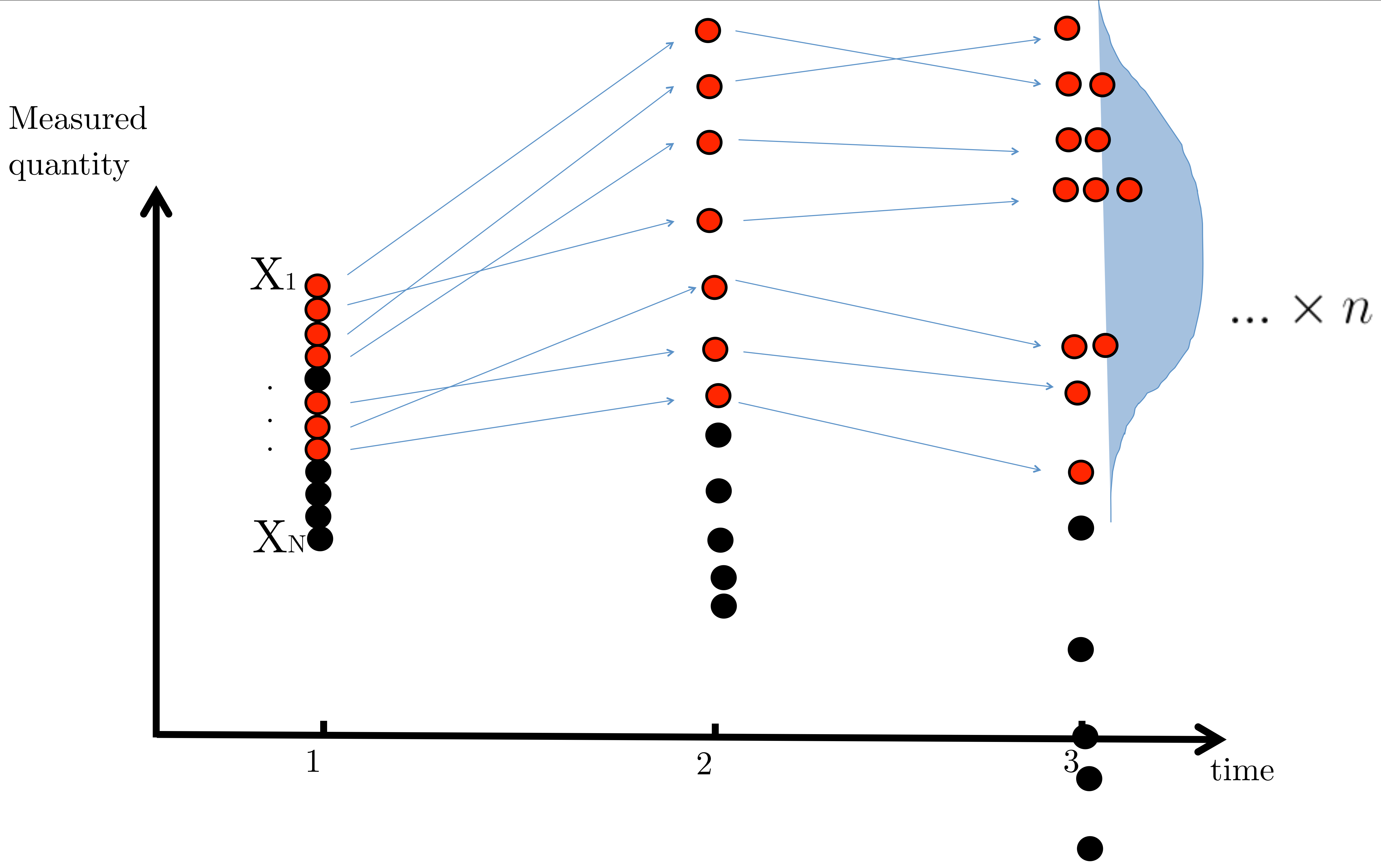
$$X_2^i|X_1^i \sim p(x_2|X_1^i)$$
$$([X_1^i, X_2^i], W_1^i) \sim p(x_{1:2}|y_1)$$

$$W_2^i = W_1^i \times p(y_2|X_2^i)$$
$$([X_1^i, X_2^i], W_2^i) \sim p(x_{1:2}|y_{1:2})$$


$p(x_3|y_3)$
Measured quantity
X_1
X_N
1
2
3
time



$Resample\ [X_1^i, X_2^i, X_3^i]\ according\ to\ W_3^i$



Measured
quantity
X_1
X_N
1
2
3
time
$\ldots \times n$

Unbiased
but noisy
estimator

$$p(y|\theta) = \prod_i \Big(\sum_j p(y_i|X_{t_i}^{(j)}, \theta)p(X_{t_i}^{(j)}|\theta)\Big)$$

Particle filtering, a.k.a. Sequential Monte Carlo

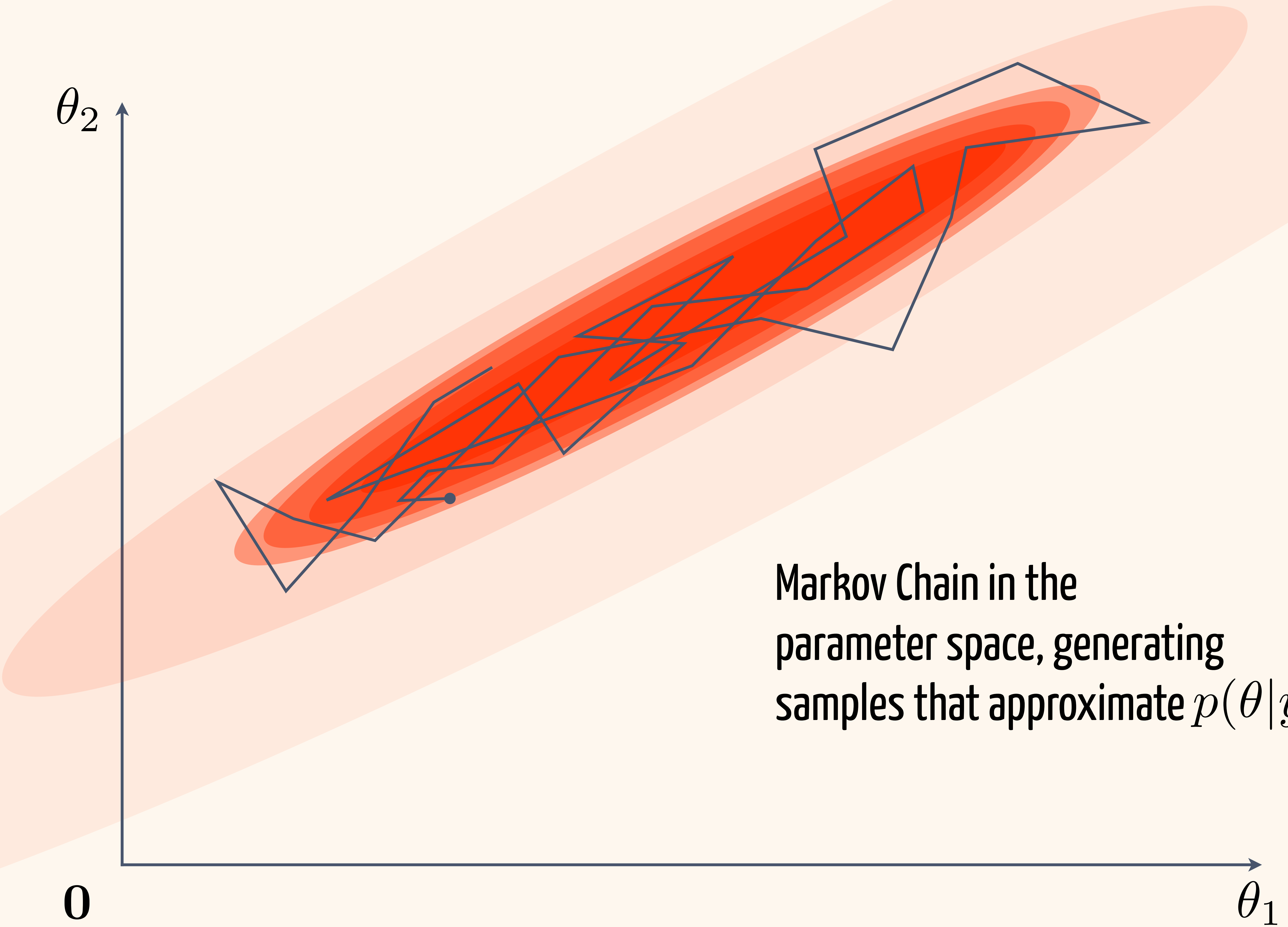
Number of particles

Print out likelihood

```
cat ../theta.json |./smc psr -J 1000 -v -hat
```

Try it

Particle filter + MCMC : PMCMC



Markov Chain in the parameter space, generating samples that approximate $p(\theta|y)$

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle markov chain monte carlo methods.

Number of particles

Print out likelihood

```
cat ../theta.json |./smc psr -J 20000 -v -I 2
```

Seed

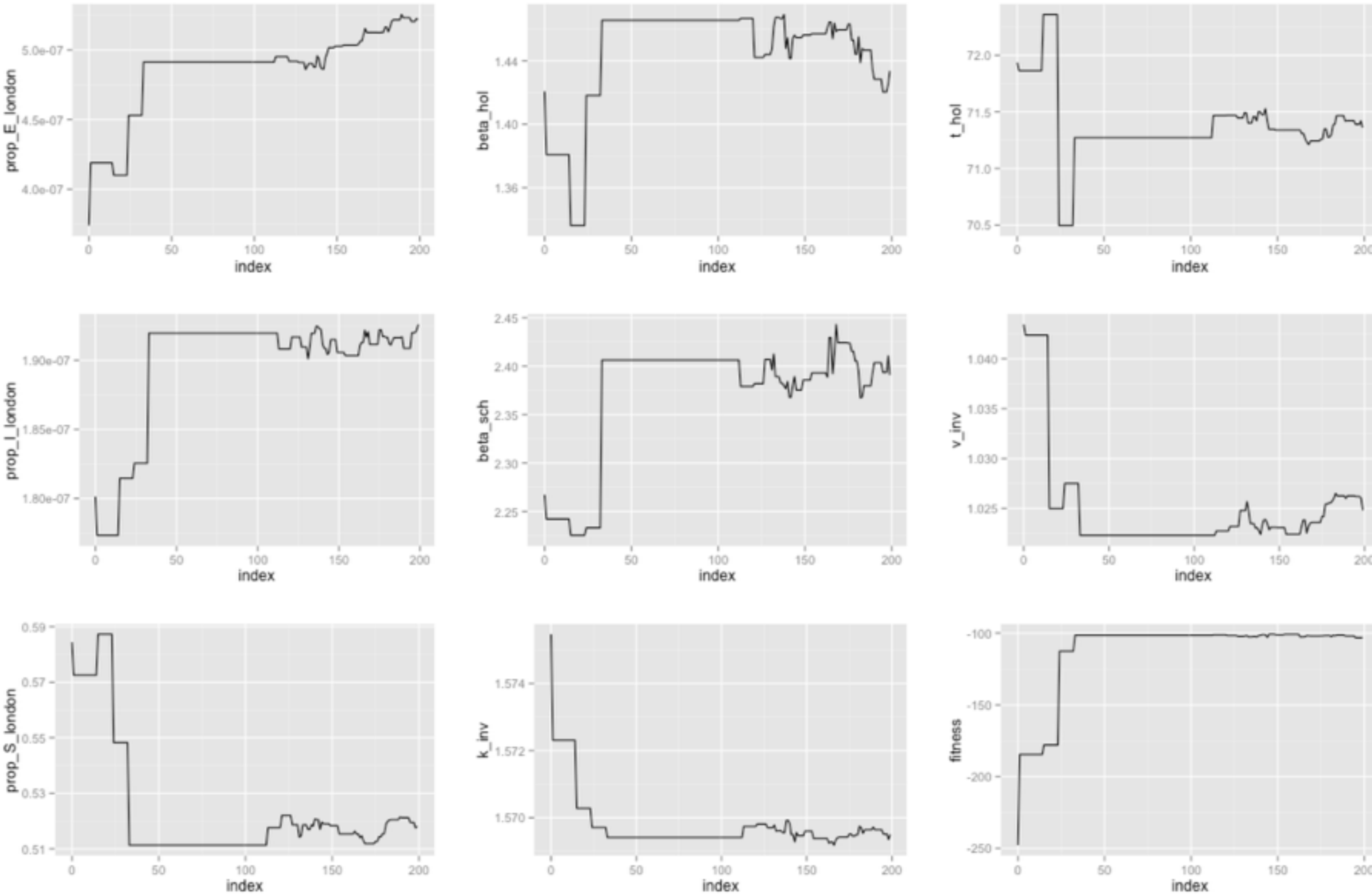
‼ Particle Filters are sensitive to model mis-specification

```
cat ../theta.json |./smc psr -J 1000 -v -I 2
```

Back to H1N1 and sir-hol

```
cat ../theta.json |./pmcmc psr -J 1000 -v -M 100
```

Back to H1N1 and sir-hol



Burn-in is costly. Could we make a first check of the posterior density while capturing stochasticity?

Extended Kalman filter

hypothesis
$$p(x_i|y_{1:i},\theta) = \mathcal{N}(\mu_i^x, \Sigma_i^x)$$
$$p(y_i|x_i,\theta) = \mathcal{N}(h(x_i), \Sigma_i^{obs})$$

hence,
$$\frac{d\mu^x}{dt} = f^{det}(\mu^x)$$
$$\frac{d\Sigma^x}{dt} = Jac(f^{det})\Sigma^x + \Sigma^x Jac(f^{det})^t + Q$$
$$\longrightarrow \quad p^{EKF}(y_{1:n}|\theta)$$

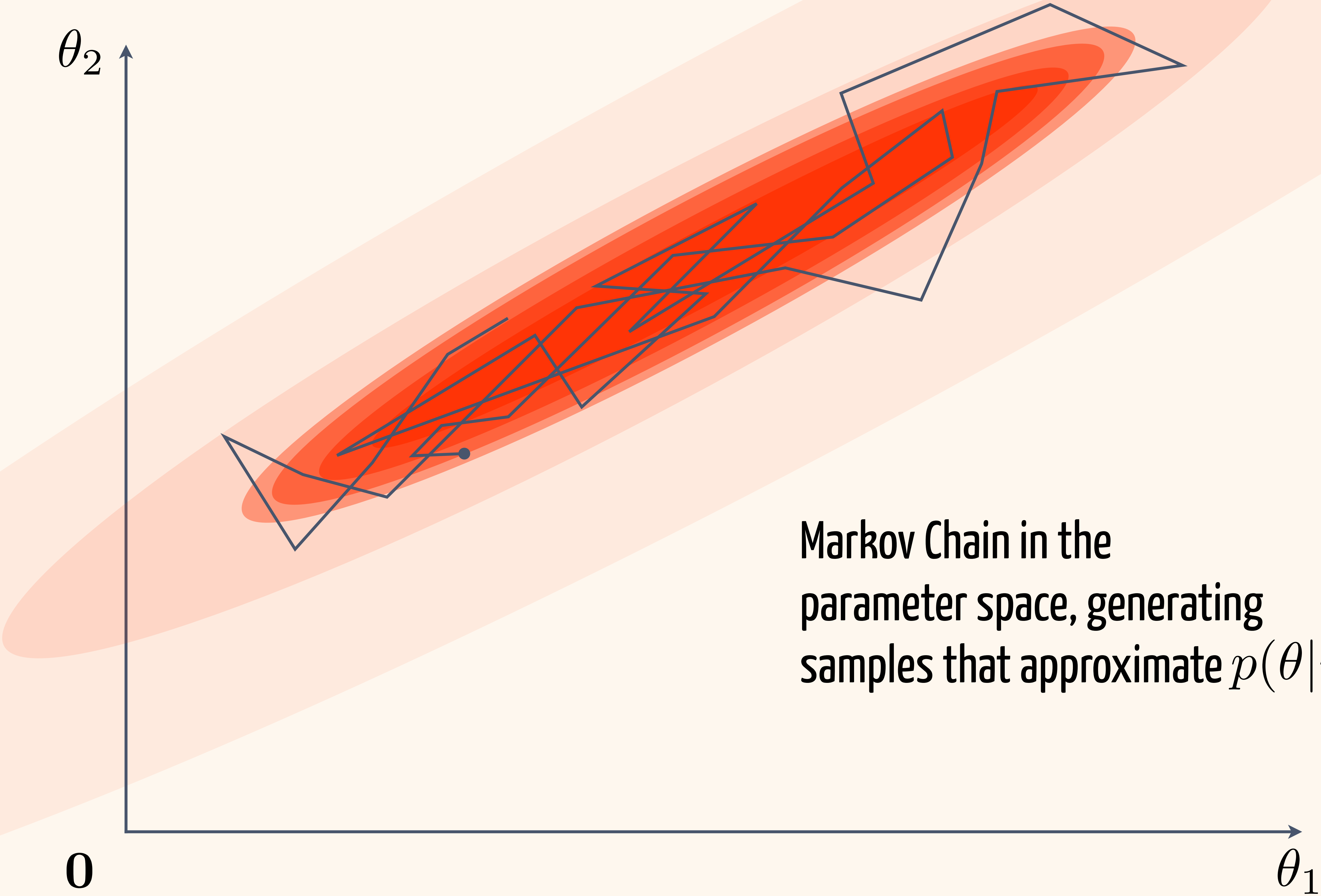
In practice:

Up to 1000 times quicker than particle filter

Uncontrolled approximation

Some numerical unstabilities to be taken care of

Extended Kalmen filter + MCMC : KMCMC



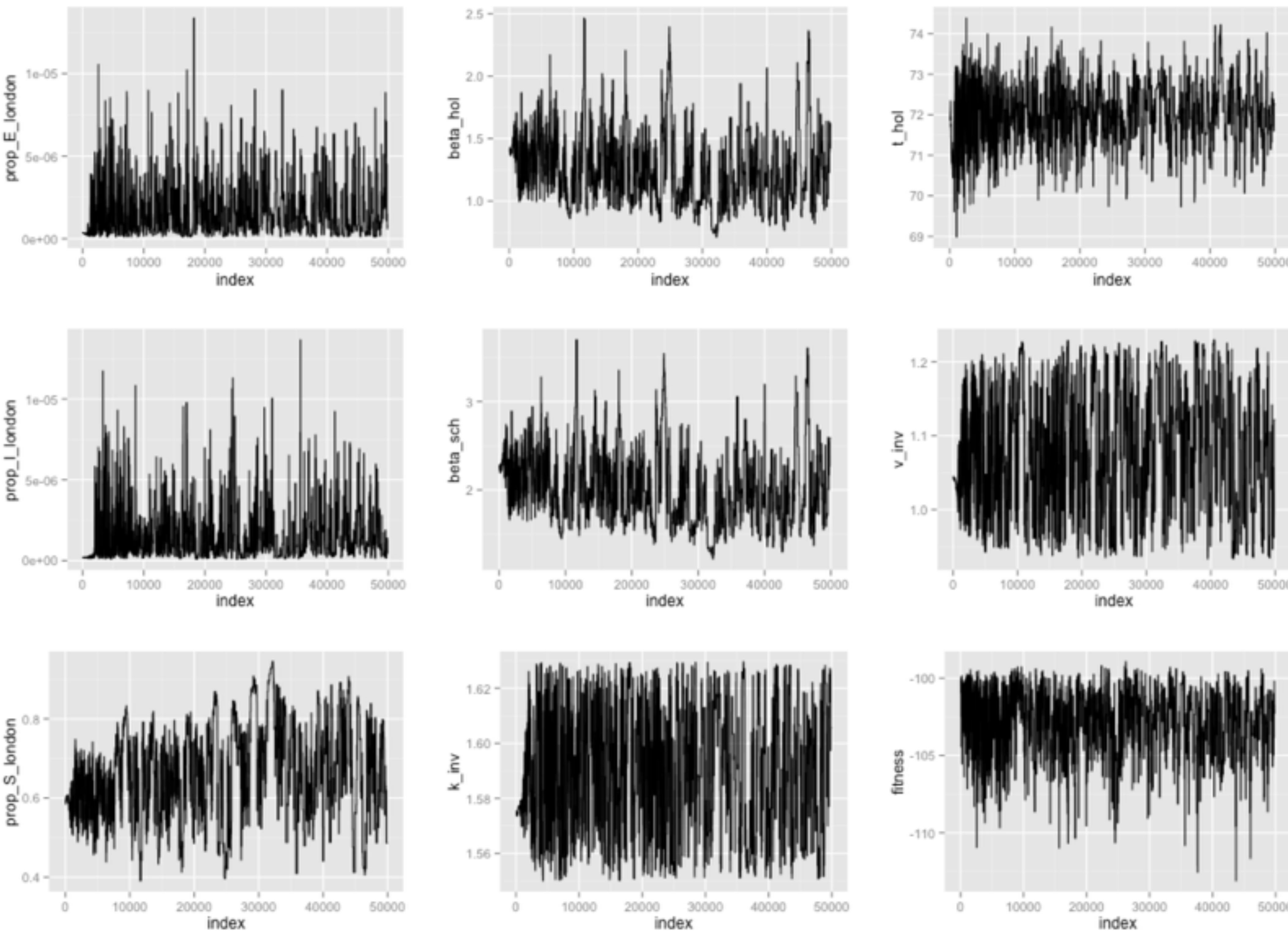
Markov Chain in the parameter space, generating samples that approximate $p(\theta|y)$

Dureau, J., Kalogeropoulos, K., and Baguelin, M. (2013).

```
cat ../theta.json |./kmcmc -v -M 50000 -trace
```

No need for implementation specification
(sde is impled)

Back to H1N1 and sir-hol



10000 iterations of KMCMC

Going Further!

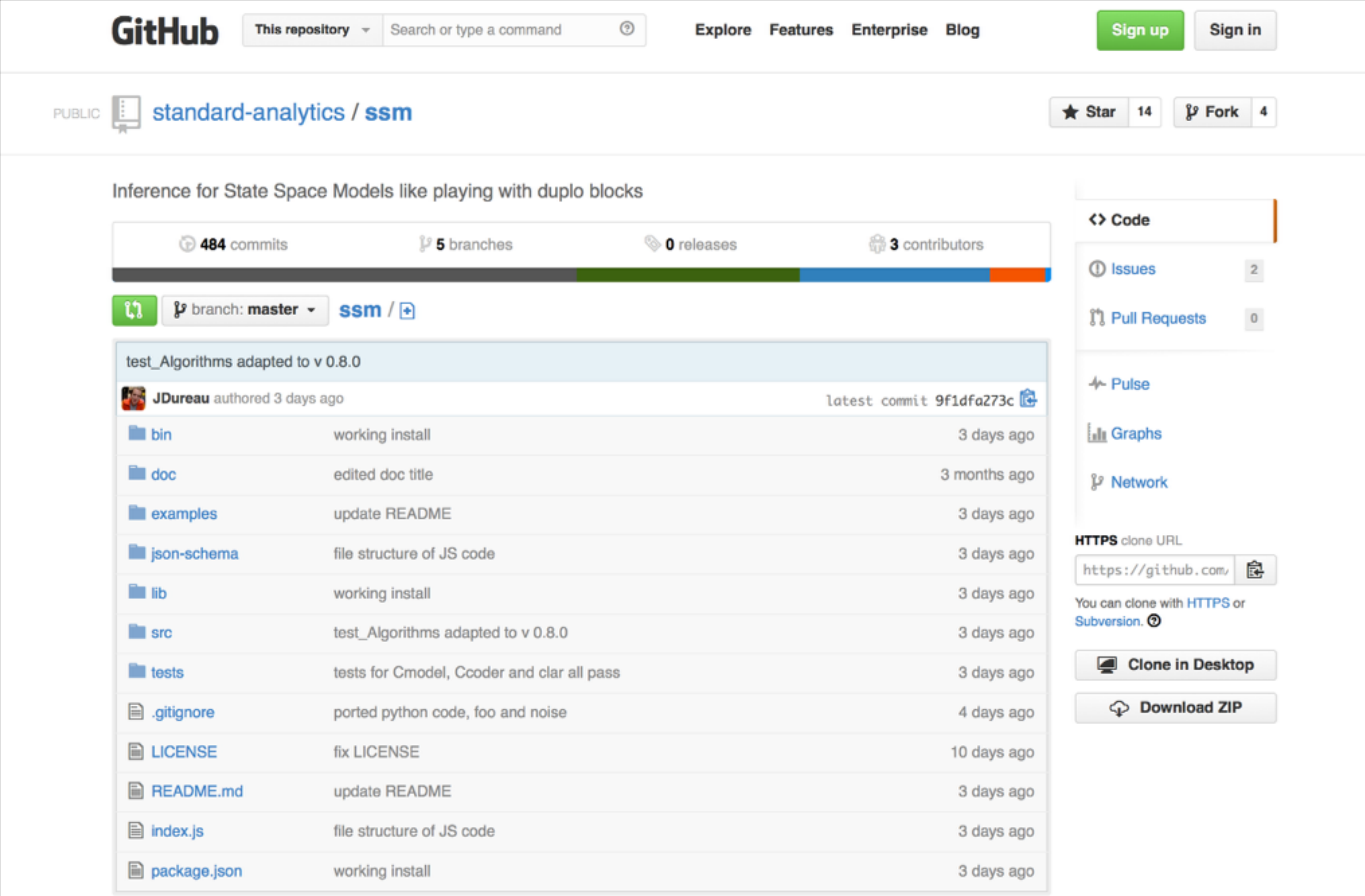
Parallel computing: -N

Environmental stochasticity: stochastic rates and time-varying parameters

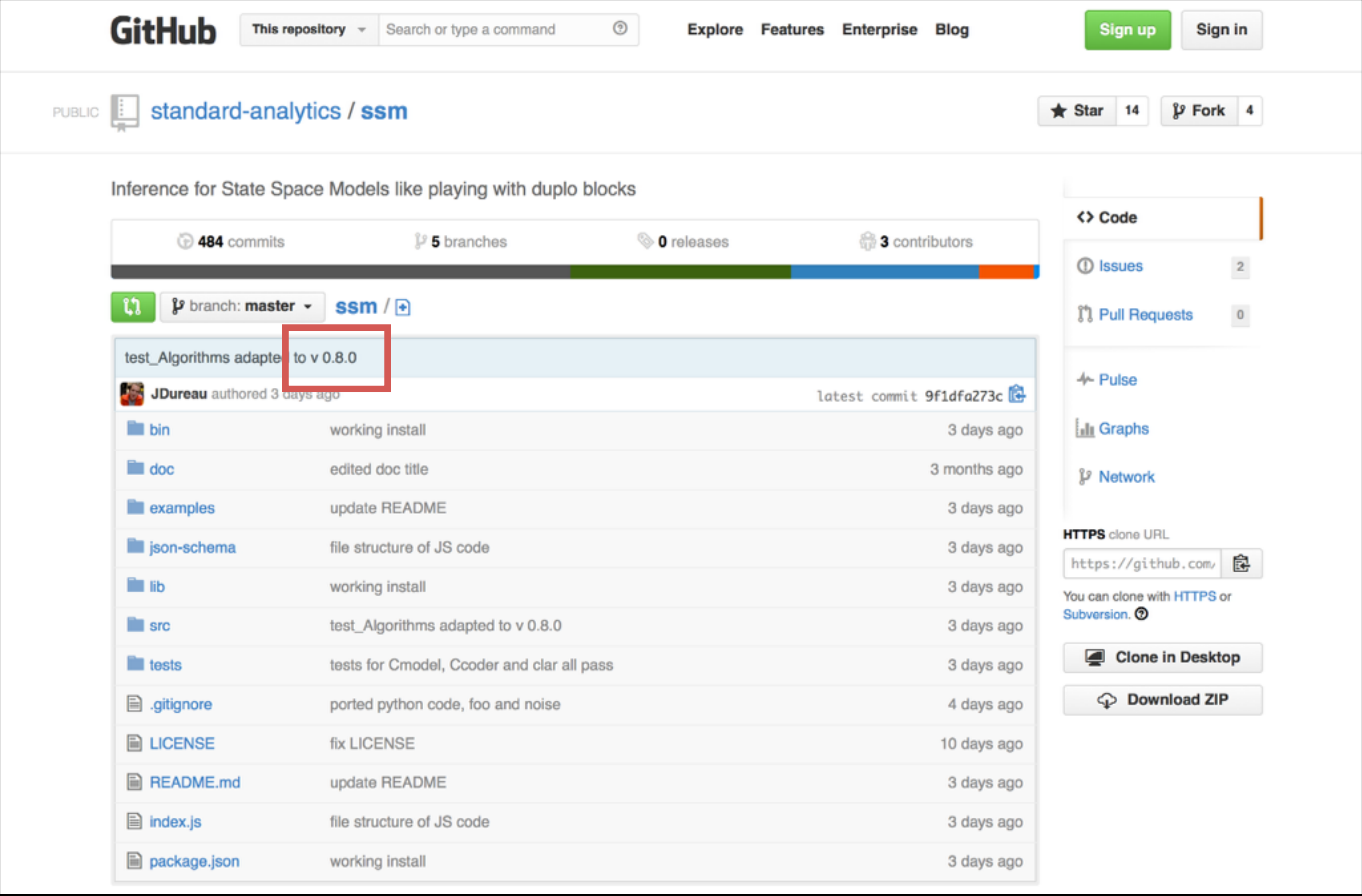
Optimization: simplex, ksimplex and mif algorithms

Program your inference routines

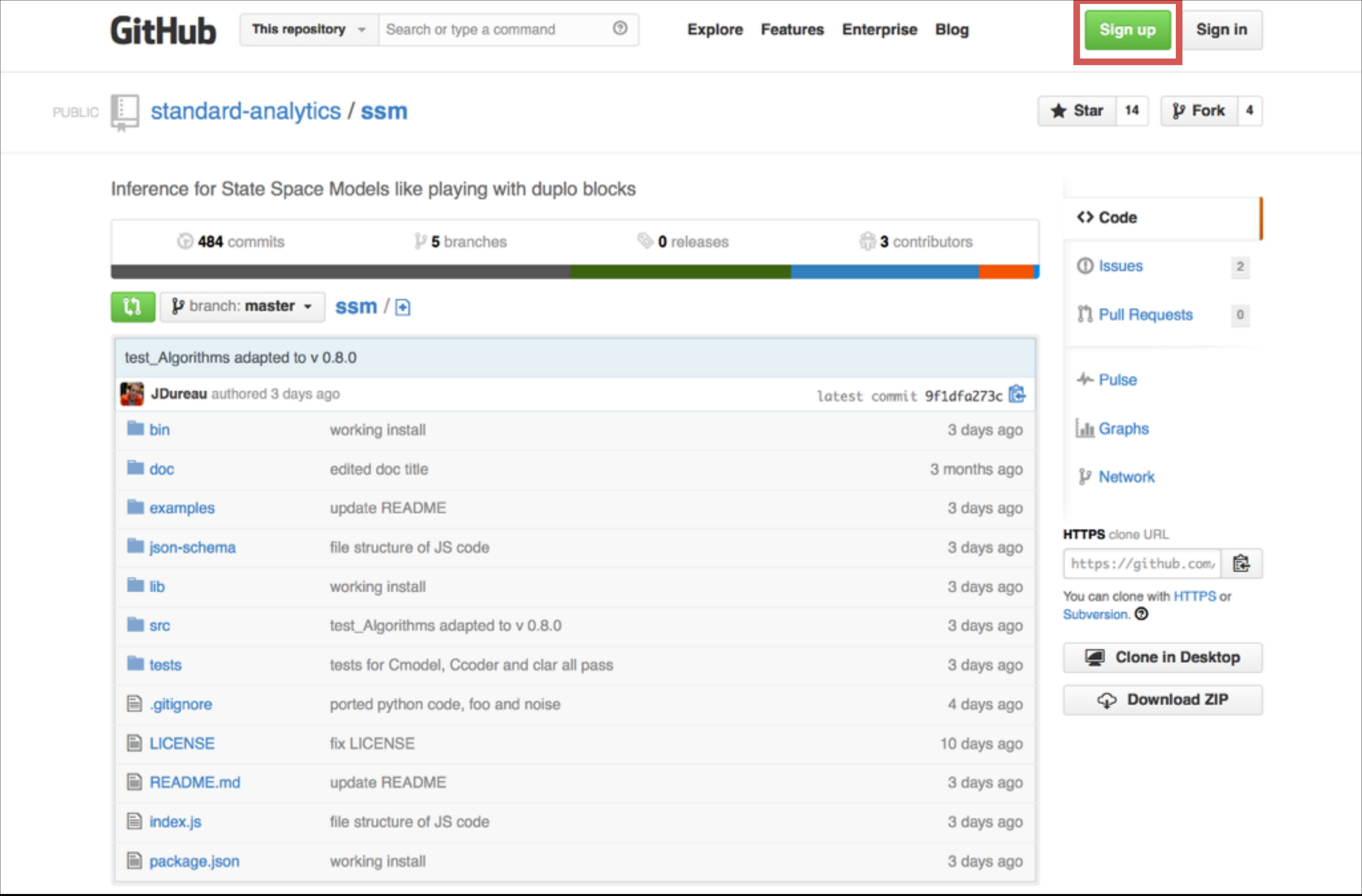
Advanced SSM tools



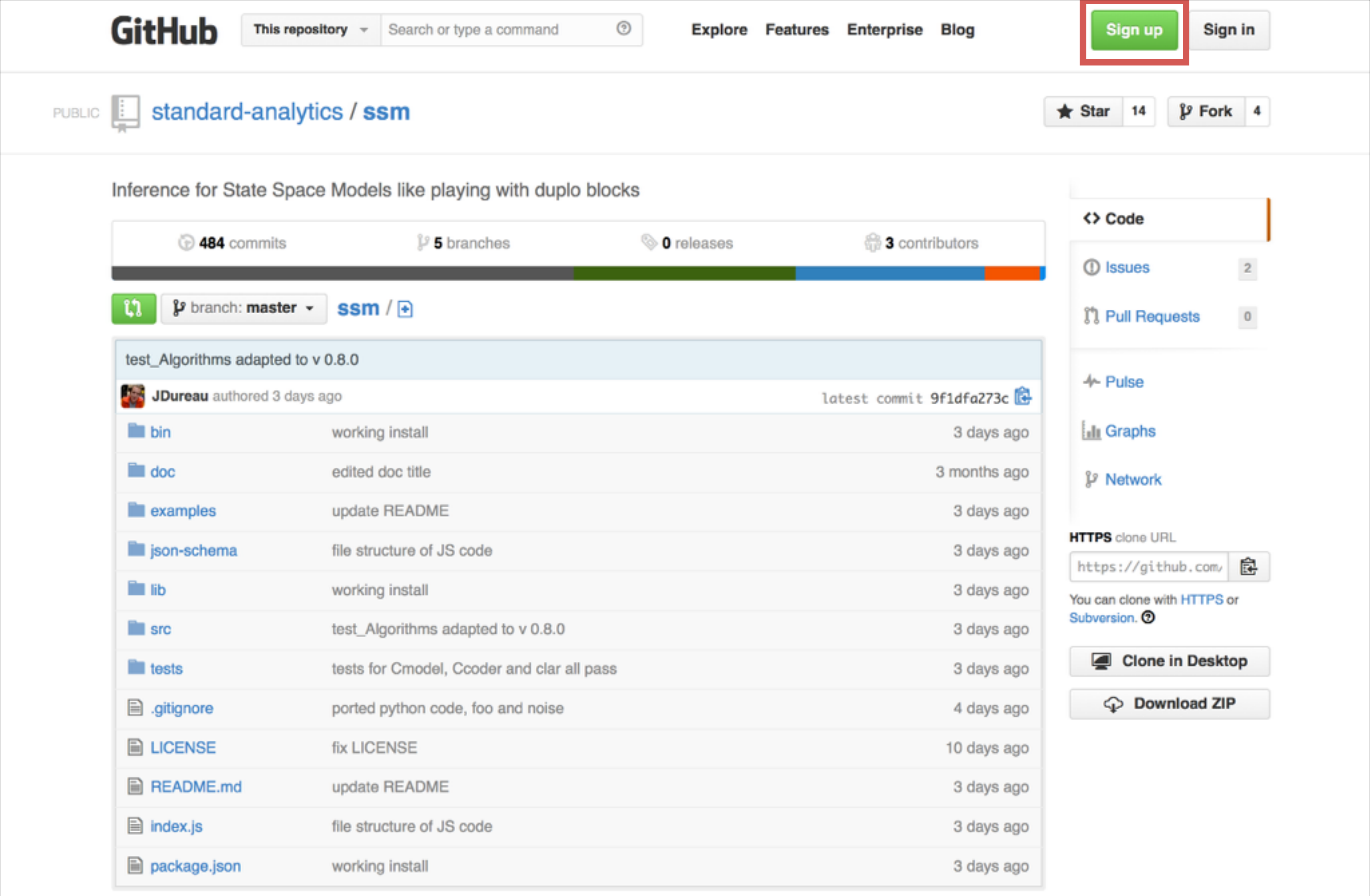
You can start using SSM today.



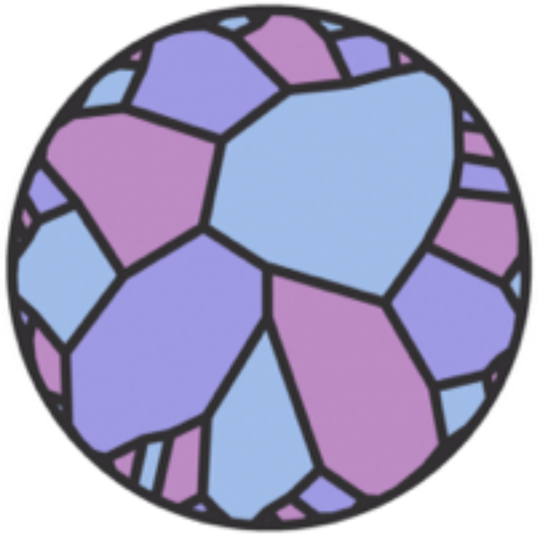
Even if it evolves, it's **version-tracked.**



Sign up on Github. **You won't regret it.**


GitHub
This repository
Search or type a command
Explore
Features
Enterprise
Blog
Sign up
Sign in
PUBLIC
standard-analytics / ssm
Star 14
Fork 4
Inference for State Space Models like playing with duplo blocks
484 commits
5 branches
0 releases
3 contributors
branch: master
ssm /
test_Algorithms adapted to v 0.8.0
JDureau authored 3 days ago
latest commit 9f1dfa273c
bin | working install | 3 days ago
doc | edited doc title | 3 months ago
examples | update README | 3 days ago
json-schema | file structure of JS code | 3 days ago
lib | working install | 3 days ago
src | test_Algorithms adapted to v 0.8.0 | 3 days ago
tests | tests for Cmodel, Ccoder and clar all pass | 3 days ago
.gitignore | ported python code, foo and noise | 4 days ago
LICENSE | fix LICENSE | 10 days ago
README.md | update README | 3 days ago
index.js | file structure of JS code | 3 days ago
package.json | working install | 3 days ago
Code
Issues 2
Pull Requests 0
Pulse
Graphs
Network
HTTPS clone URL
https://github.com/
You can clone with HTTPS or Subversion.
Clone in Desktop
Download ZIP

It’s **free…**

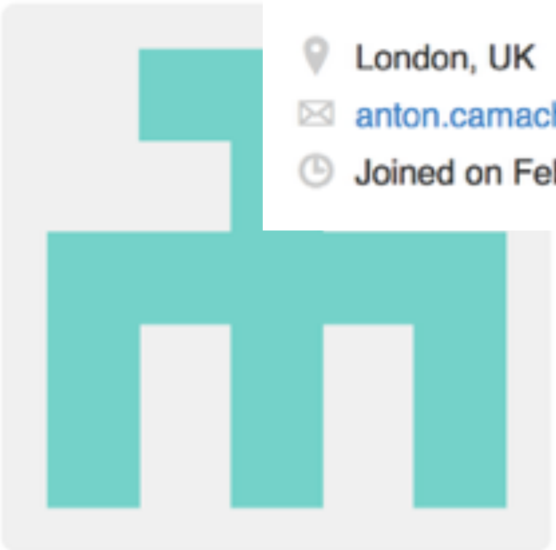


Trevor Bedford
trvrb

Fred Hutchinson Cancer Research Center
Seattle, WA
http://bedford.io
Joined on Nov 06, 2011

Anton
ntncmch

London, UK
anton.camacho@lshtm.ac.uk
Joined on Feb 06, 2012



Oliver
olli0601

London
oliver.ratmann@imperial.ac.uk
Joined on Sep 20, 2012

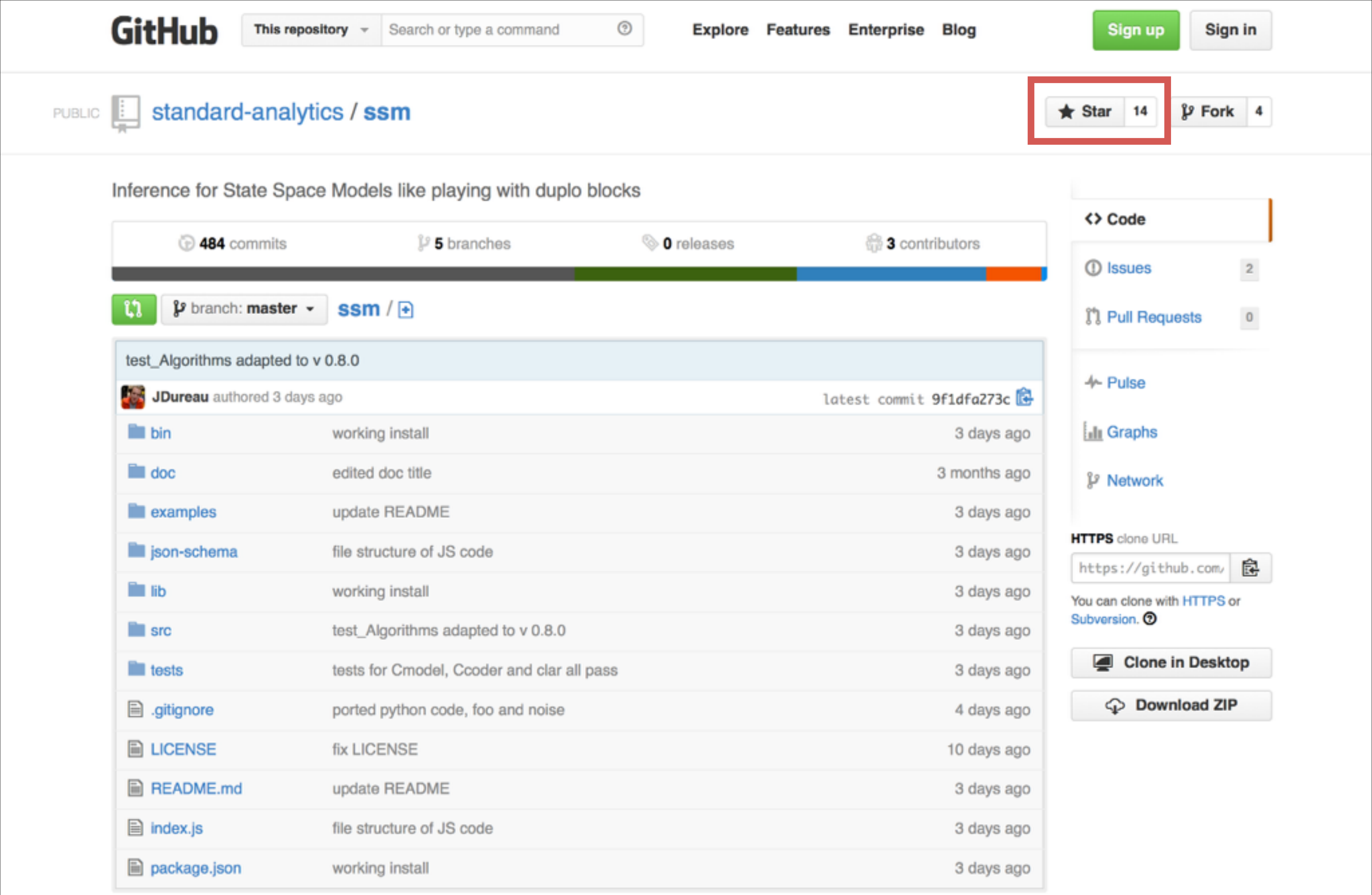
Serge Stinckwich
SergeStinckwich

UCBN & IRD
France
Serge.Stinckwich@gmail.com
http://doesnotunderstand.org/
Joined on Jul 09, 2010

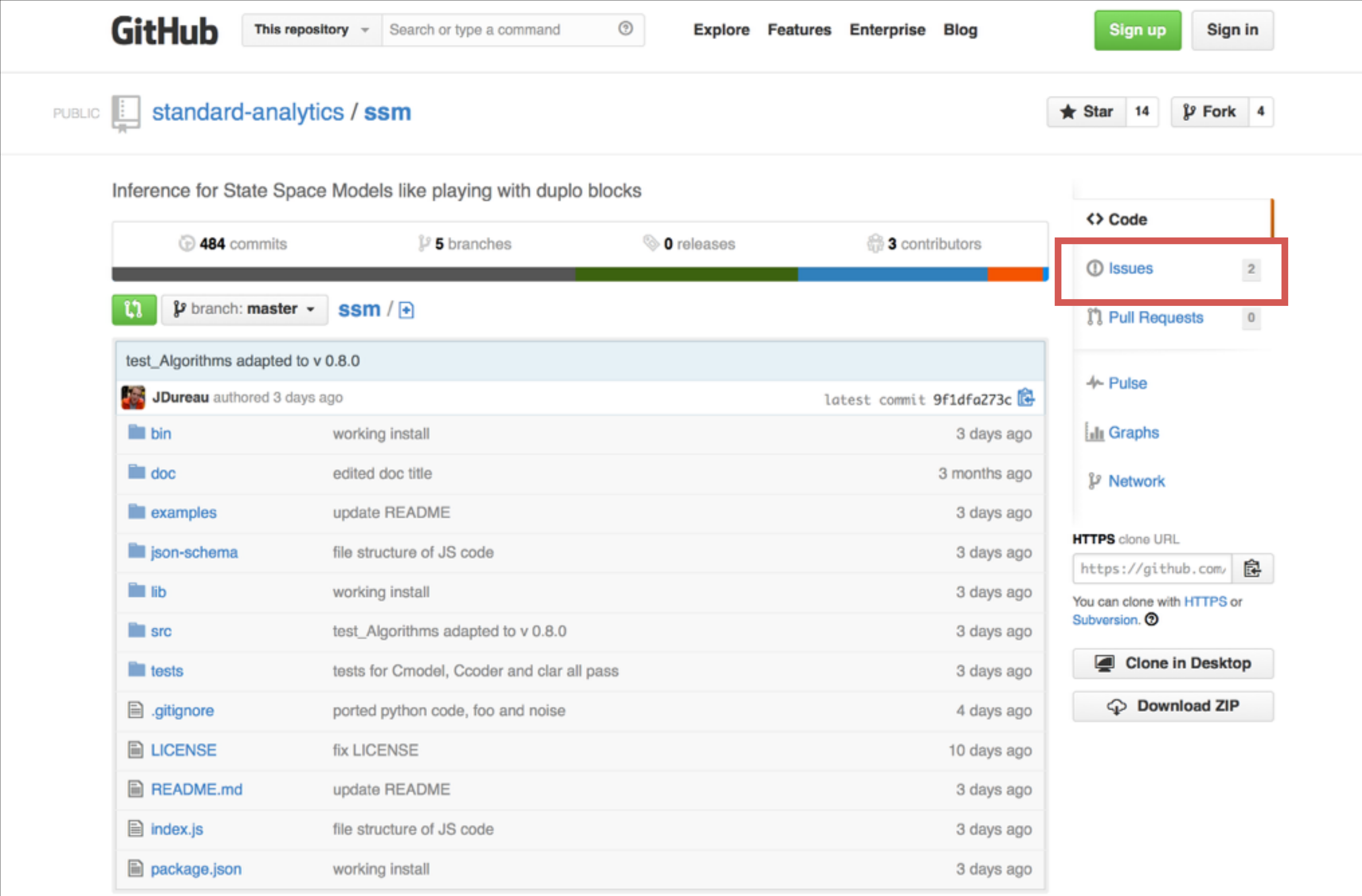
Cazelles Bernard
cazelles

UPMC
Paris
cazelles@biologie.ens.fr
Joined on Nov 25, 2013

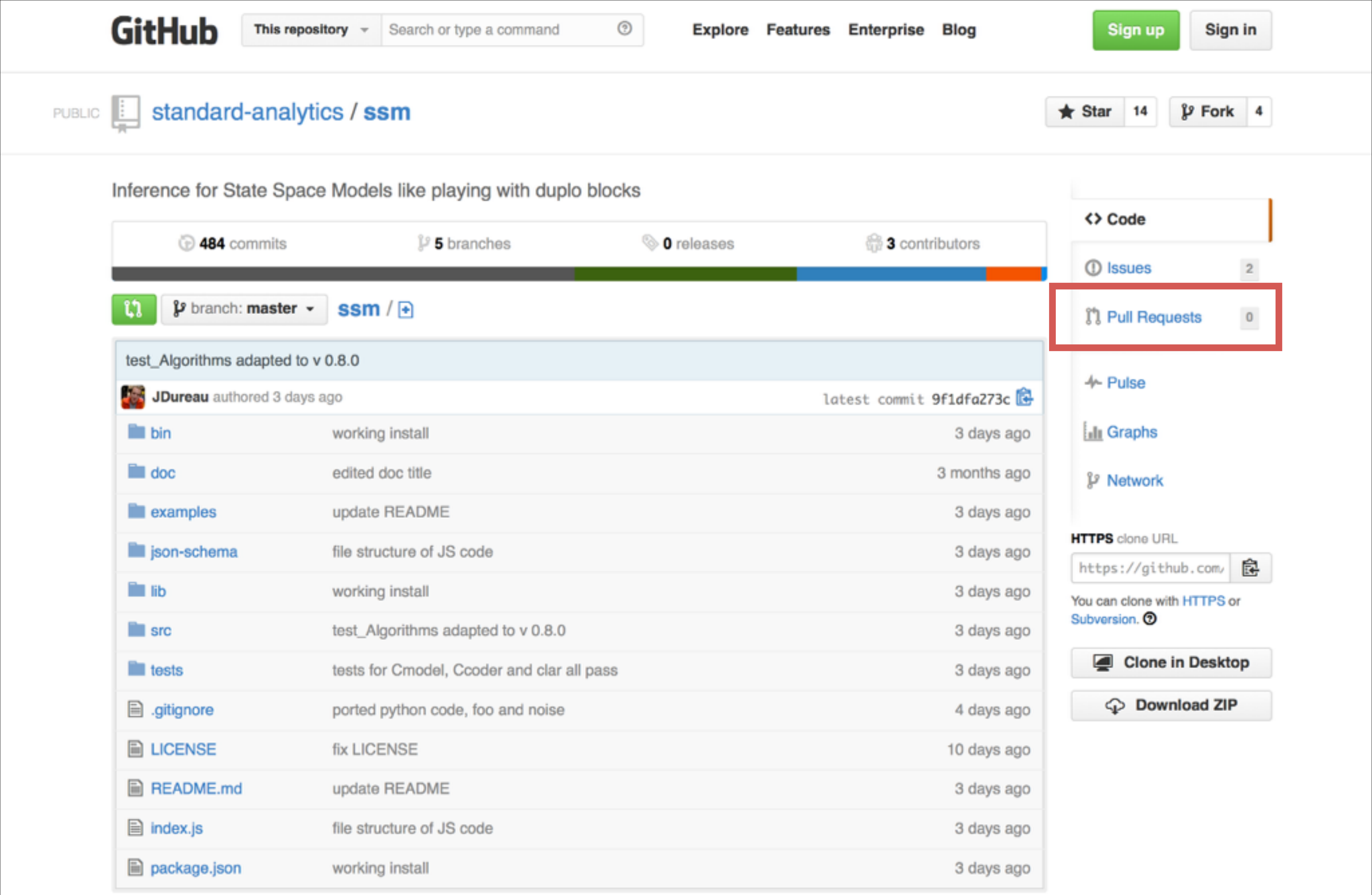
and all the **cool kids** are there !



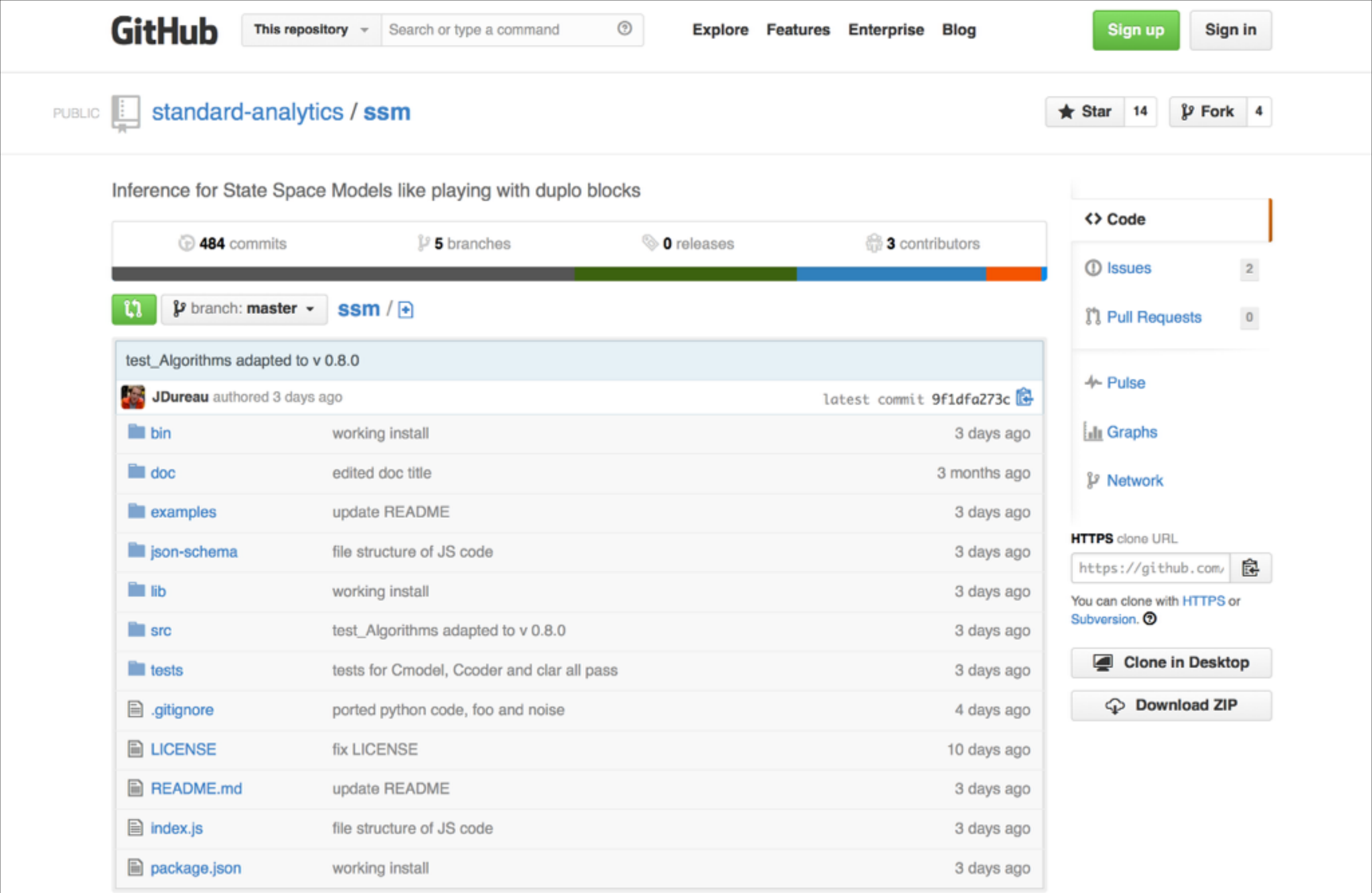
You'll be able to **support SSM.**



Raise issues



and **contribute.**



Let’s make this **together**.