Introducción a la Inferencia Bayesiana

La práctica

Operacionales

- Trabajo en grupos de $N < 3$ ($N<=2$)
- Modalidad *“Hágalo Usted Mismo”*
- Mayormente en computadora
- Seguimos apunte de *Wagenmakers & Lee*
- Herramienta a elección

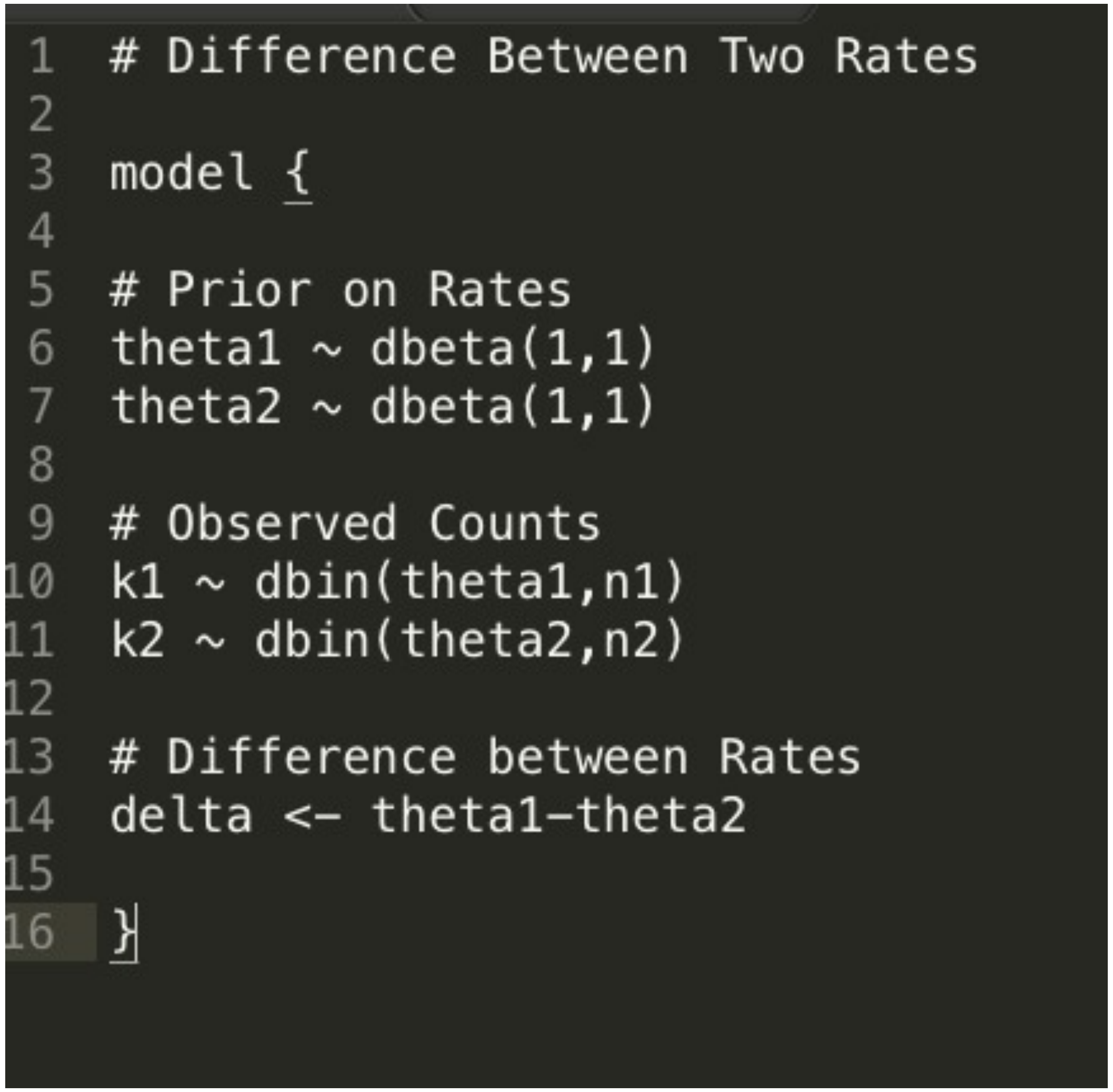
Herramientas

- WinBUGS/jags (+MATLAB)
- pymc
- church
- WebPPL

WinBUGS/jags

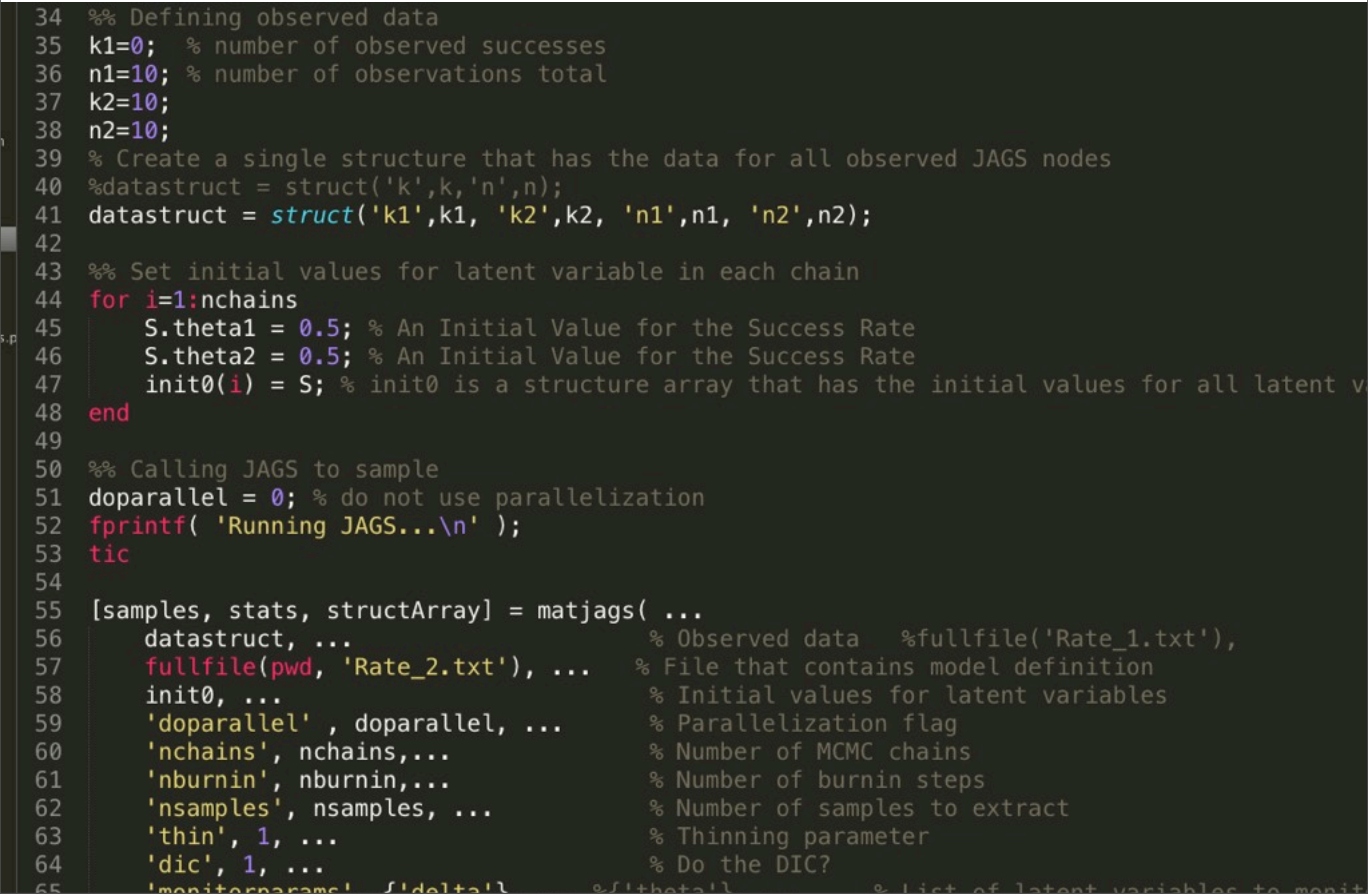
- Enlatado/open source
- Interfaz gráfica, pero se puede usar sin
- Sintaxis cercana a R
- Scripts de MATLAB y R para correr
- Usado en el apunte
- Herramienta más *standard,* no muy flexible
- http://mcmc-jags.sourceforge.net/

WinBUGS/jags



```
# Difference Between Two Rates

model {

# Prior on Rates
theta1 ~ dbeta(1,1)
theta2 ~ dbeta(1,1)

# Observed Counts
k1 ~ dbin(theta1,n1)
k2 ~ dbin(theta2,n2)

# Difference between Rates
delta <- theta1-theta2

}
```

Calling matjags.m

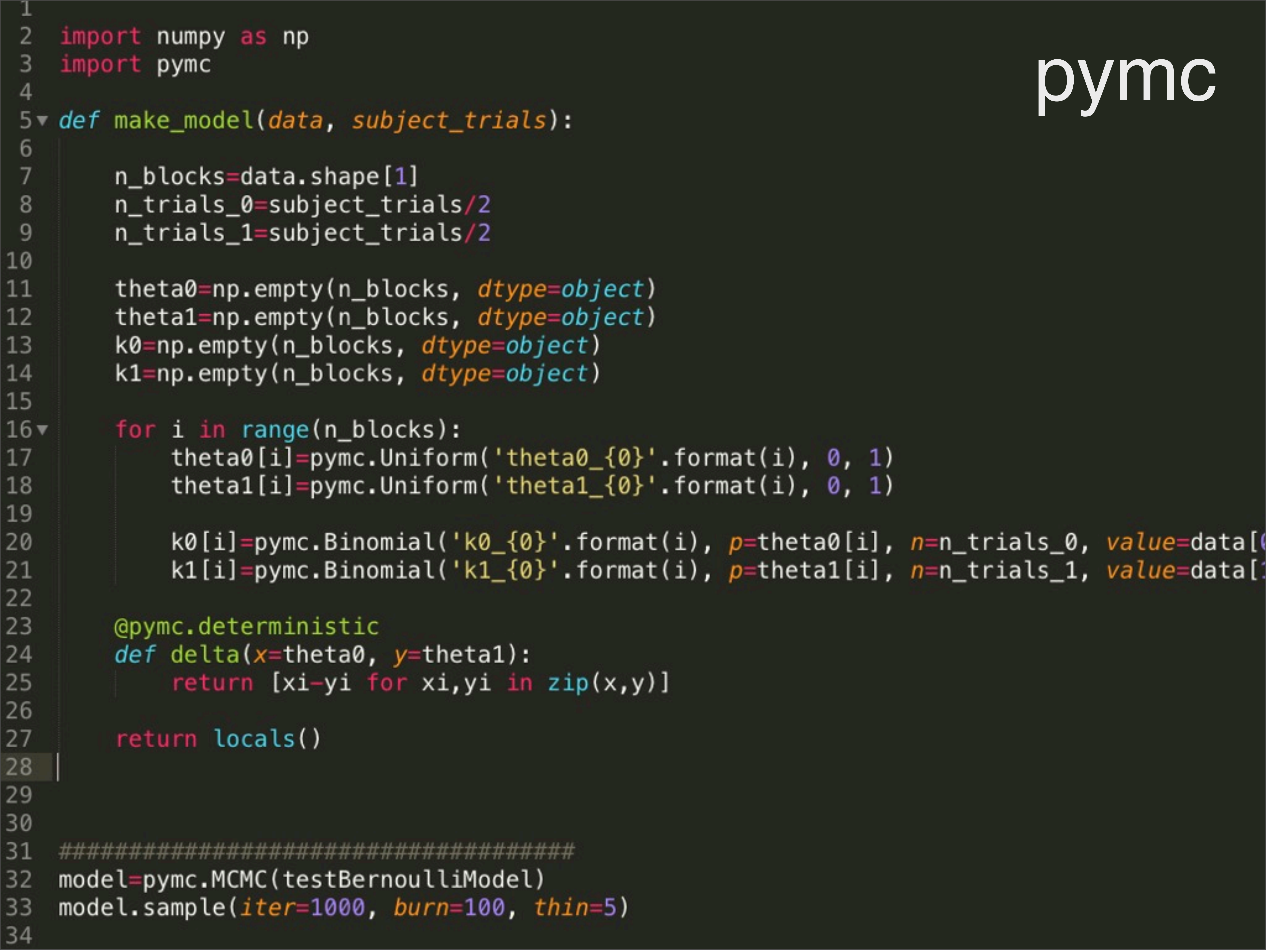


```
%% Defining observed data
k1=0;  % number of observed successes
n1=10; % number of observations total
k2=10;
n2=10;
% Create a single structure that has the data for all observed JAGS nodes
%datastruct = struct('k',k,'n',n);
datastruct = struct('k1',k1, 'k2',k2, 'n1',n1, 'n2',n2);

%% Set initial values for latent variable in each chain
for i=1:nchains
    S.theta1 = 0.5; % An Initial Value for the Success Rate
    S.theta2 = 0.5; % An Initial Value for the Success Rate
    init0(i) = S; % init0 is a structure array that has the initial values for all latent va
end

%% Calling JAGS to sample
doparallel = 0; % do not use parallelization
fprintf( 'Running JAGS...\n' );
tic

[samples, stats, structArray] = matjags( ...
    datastruct, ...                         % Observed data   %fullfile('Rate_1.txt'),
    fullfile(pwd, 'Rate_2.txt'), ...       % File that contains model definition
    init0, ...                              % Initial values for latent variables
    'doparallel' , doparallel, ...          % Parallelization flag
    'nchains', nchains,...                  % Number of MCMC chains
    'nburnin', nburnin,...                  % Number of burnin steps
    'nsamples', nsamples, ...               % Number of samples to extract
    'thin', 1, ...                          % Thinning parameter
    'dic', 1, ...                           % Do the DIC?
    'monitorparams', {'delta'},      %{'theta'}          % List of latent variables to monit
```

pymc

- Librería de python
- Más flexible, menos robusta?
- python
- https://pypi.python.org/pypi/pymc

pymc

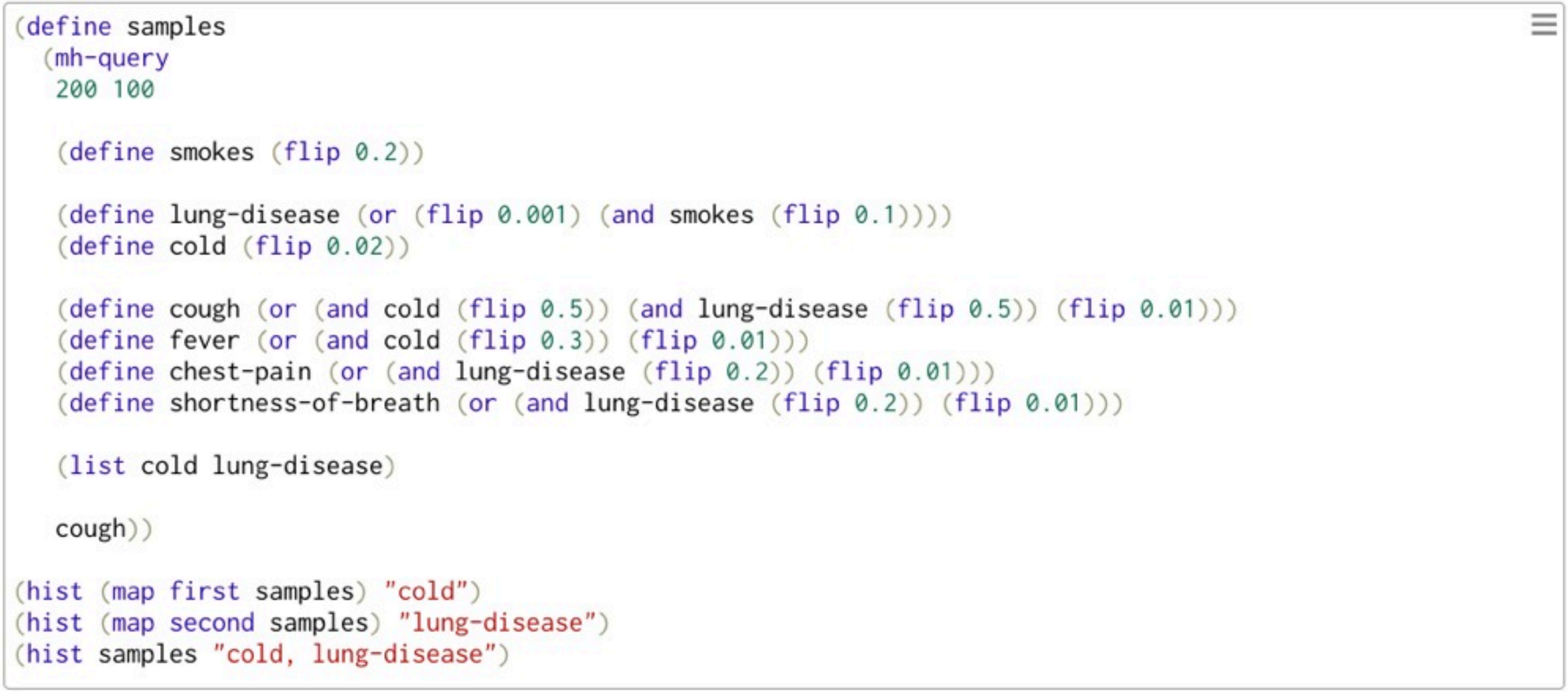


```python

import numpy as np
import pymc

def make_model(data, subject_trials):

    n_blocks=data.shape[1]
    n_trials_0=subject_trials/2
    n_trials_1=subject_trials/2

    theta0=np.empty(n_blocks, dtype=object)
    theta1=np.empty(n_blocks, dtype=object)
    k0=np.empty(n_blocks, dtype=object)
    k1=np.empty(n_blocks, dtype=object)

    for i in range(n_blocks):
        theta0[i]=pymc.Uniform('theta0_{0}'.format(i), 0, 1)
        theta1[i]=pymc.Uniform('theta1_{0}'.format(i), 0, 1)

        k0[i]=pymc.Binomial('k0_{0}'.format(i), p=theta0[i], n=n_trials_0, value=data[
        k1[i]=pymc.Binomial('k1_{0}'.format(i), p=theta1[i], n=n_trials_1, value=data[

    @pymc.deterministic
    def delta(x=theta0, y=theta1):
        return [xi-yi for xi,yi in zip(x,y)]

    return locals()



#####################################
model=pymc.MCMC(testBernoulliModel)
model.sample(iter=1000, burn=100, thin=5)

```

church

- Lenguaje funcional probabilístico, basado en *scheme*
- Muestras <-> Ejecuciones
- Diseñado para ciencias cognitivas
- “Lenguaje de la mente”
- Varios compiladores, complejo hacerlos andar, resultados diversos: mejor en web
- https://probmods.org (lenguaje + libro)

church



```
(define samples
  (mh-query
   200 100

   (define smokes (flip 0.2))

   (define lung-disease (or (flip 0.001) (and smokes (flip 0.1))))
   (define cold (flip 0.02))

   (define cough (or (and cold (flip 0.5)) (and lung-disease (flip 0.5)) (flip 0.01)))
   (define fever (or (and cold (flip 0.3)) (flip 0.01)))
   (define chest-pain (or (and lung-disease (flip 0.2)) (flip 0.01)))
   (define shortness-of-breath (or (and lung-disease (flip 0.2)) (flip 0.01)))

   (list cold lung-disease)

   cough))

(hist (map first samples) "cold")
(hist (map second samples) "lung-disease")
(hist samples "cold, lung-disease")
```

WebPPL

- Lenguaje funcional probabilístico, *subset* de JavaScript
- Misma gente que church
- Mismas ideas que church
- Varios compiladores, complejo hacerlos andar, resultados diversos: mejor en web
- https://github.com/probmods/webppl
- Libro asociado: http://dippl.org/