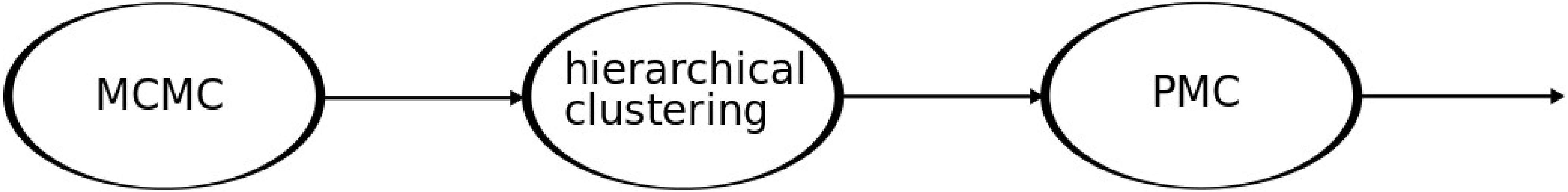
# ~~Monty Python's~~ Stephan's search for the holy grail

1. Starting point: Fred's algorithm
2. Improvements, where and why
3. Hierarchical clustering → Variational Bayes
4. PMC → Variational Bayes

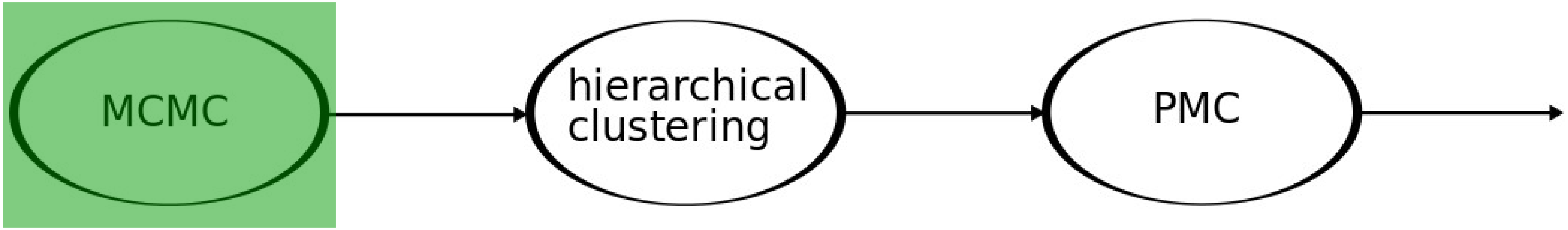
# Fred's algorithm



Overall goal: Find “good” proposal for Importance Sampling

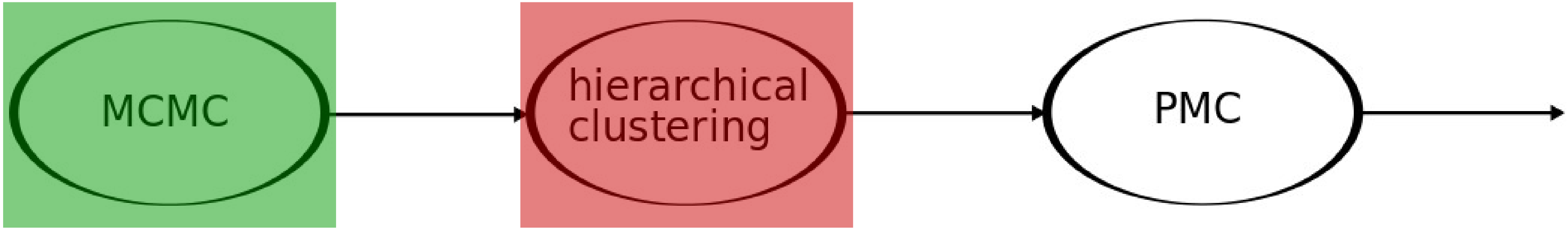
$$\int P(x) = \int \frac{P(x)}{q(x)} q(x) \approx \sum_{x_i \in samples} \frac{P(x_i)}{q(x_i)} \quad where \quad x_i \sim q(x)$$

# Fred's algorithm



does its job

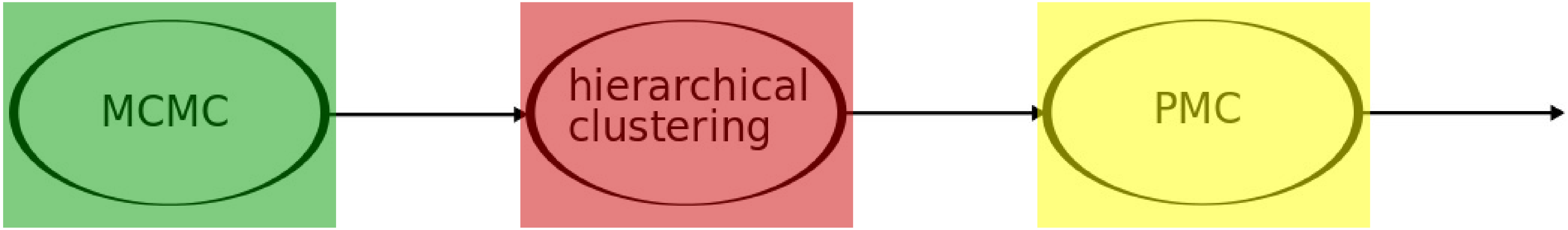
# Fred's algorithm



does its job

does not reduce redundant information

# Fred's algorithm

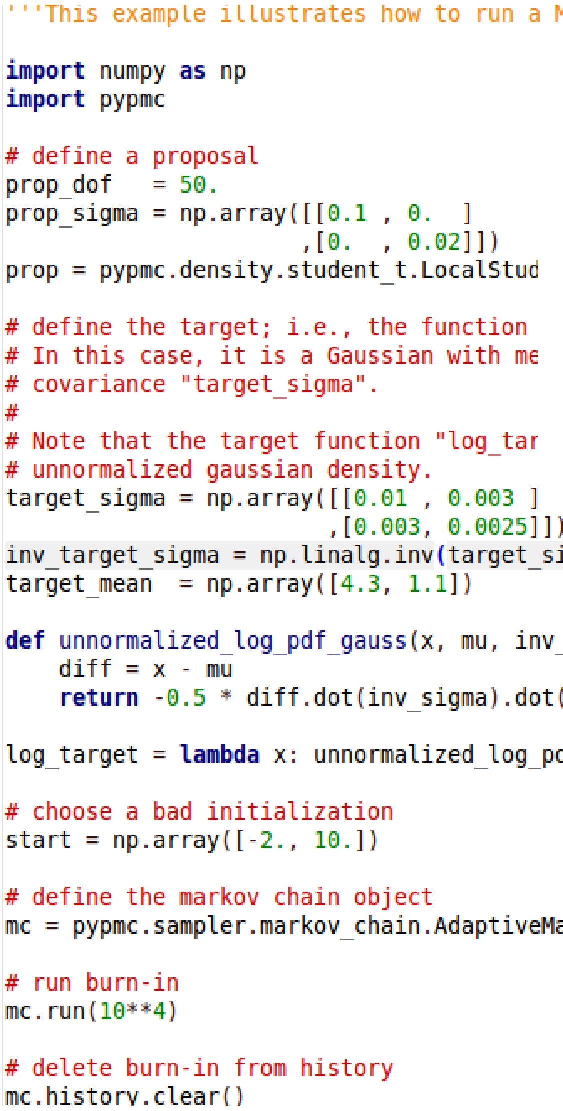


does its job

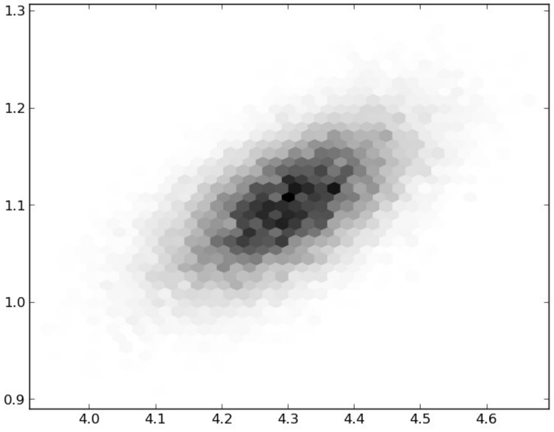
does not reduce redundant information

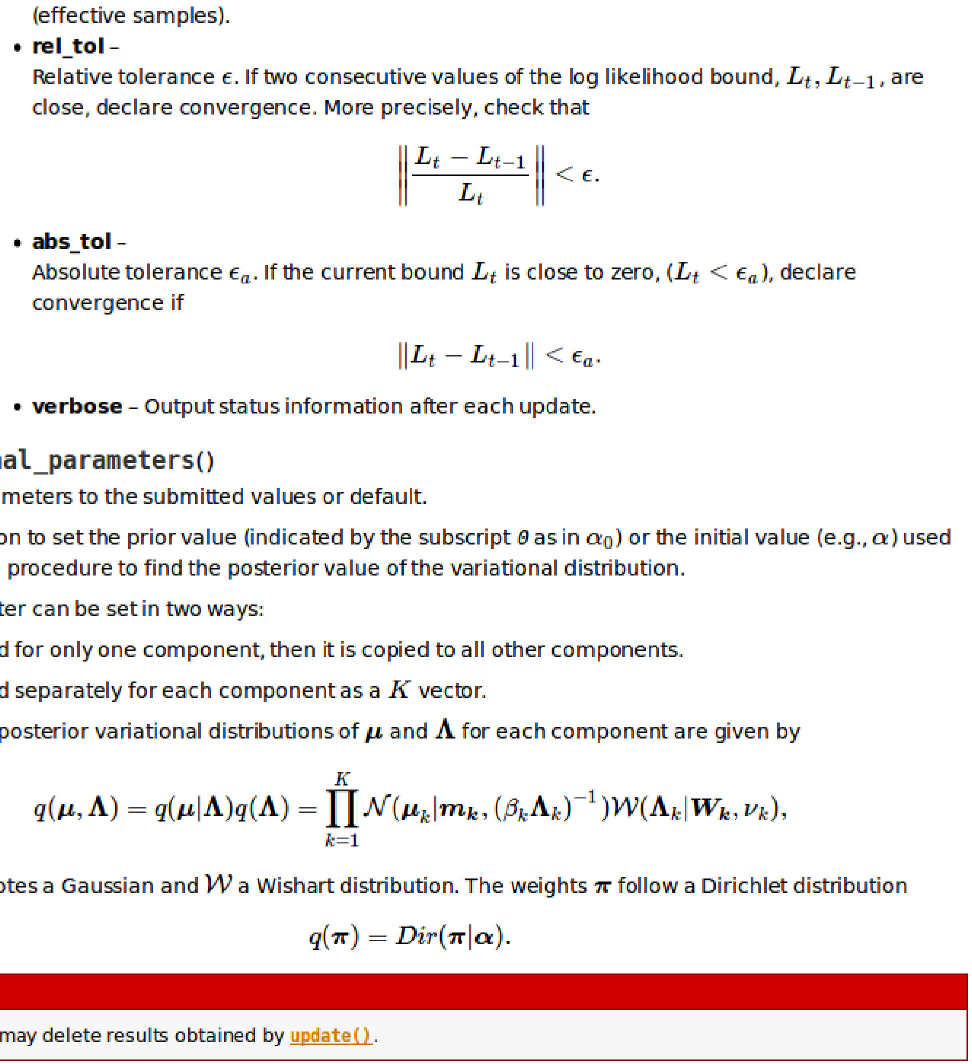
no re-use of samples in subsequent update steps

# https://github.com/fredRos/pypmc



```
'''This example illustrates how to run a Markov Chain using pypmc'''

import numpy as np
import pypmc

# define a proposal
prop_dof   = 50.
prop_sigma = np.array([[0.1 , 0.   ]
                      ,[0.   , 0.02]])
prop = pypmc.density.student_t.LocalStud

# define the target; i.e., the function
# In this case, it is a Gaussian with me
# covariance "target_sigma".
#
# Note that the target function "log_tar
# unnormalized gaussian density.
target_sigma = np.array([[0.01 , 0.003 ]
                        ,[0.003, 0.0025]])
inv_target_sigma = np.linalg.inv(target_sigma)
target_mean  = np.array([4.3, 1.1])

def unnormalized_log_pdf_gauss(x, mu, inv_sigma):
    diff = x - mu
    return -0.5 * diff.dot(inv_sigma).dot(diff)

log_target = lambda x: unnormalized_log_pdf_gauss(x, target_mean, i

# choose a bad initialization
start = np.array([-2., 10.])

# define the markov chain object
mc = pypmc.sampler.markov_chain.AdaptiveMarkovChain(log_target, prc

# run burn-in
mc.run(10**4)

# delete burn-in from history
mc.history.clear()
```





(effective samples).

- **rel_tol** – Relative tolerance $\epsilon$. If two consecutive values of the log likelihood bound, $L_t, L_{t-1}$, are close, declare convergence. More precisely, check that

$$\left\| \frac{L_t - L_{t-1}}{L_t} \right\| < \epsilon.$$

- **abs_tol** – Absolute tolerance $\epsilon_a$. If the current bound $L_t$ is close to zero, ($L_t < \epsilon_a$), declare convergence if

$$\| L_t - L_{t-1} \| < \epsilon_a.$$

- **verbose** – Output status information after each update.

`set_variational_parameters()`

Reset the parameters to the submitted values or default.

Use this function to set the prior value (indicated by the subscript $0$ as in $\alpha_0$) or the initial value (e.g., $\alpha$) used in the iterative procedure to find the posterior value of the variational distribution.

Every parameter can be set in two ways:

1. It is specified for only one component, then it is copied to all other components.
2. It is specified separately for each component as a $K$ vector.

The prior and posterior variational distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ for each component are given by

$$q(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\boldsymbol{\mu}|\boldsymbol{\Lambda}) q(\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k | \boldsymbol{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k | \boldsymbol{W}_k, \nu_k),$$

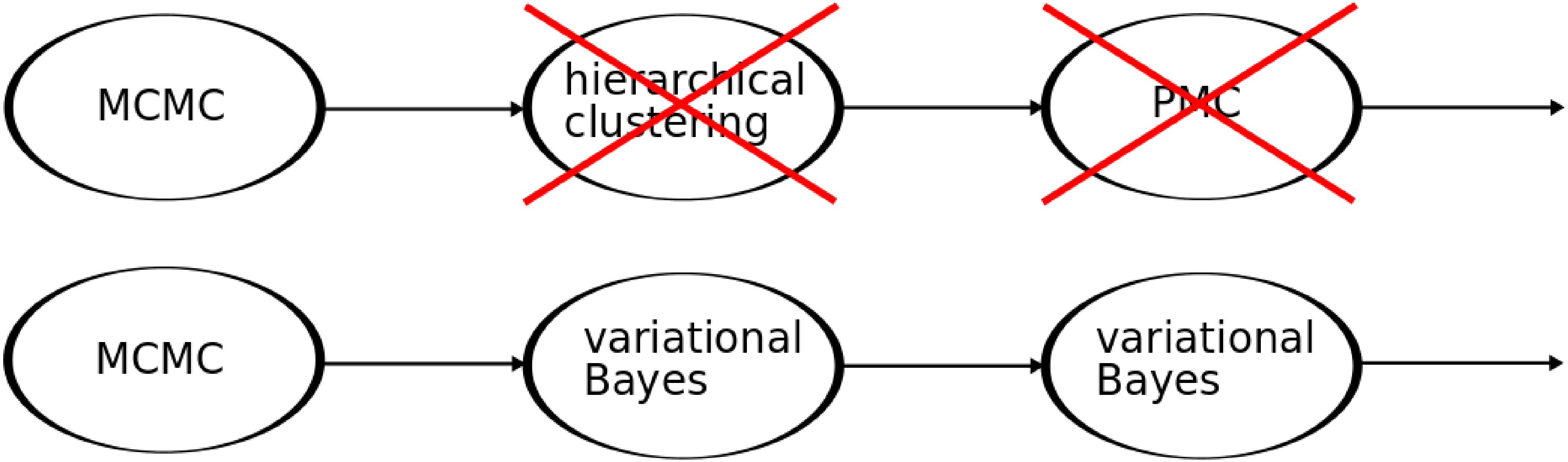
where $\mathcal{N}$ denotes a Gaussian and $\mathcal{W}$ a Wishart distribution. The weights $\boldsymbol{\pi}$ follow a Dirichlet distribution

$$q(\boldsymbol{\pi}) = Dir(\boldsymbol{\pi}|\boldsymbol{\alpha}).$$

**Warning**

This function may delete results obtained by `update()`.

# Suggested new algorithm



# Variational Bayes

- adapt Gaussian mixture to samples
- Expectation Maximization (EM) with non trivial prior

- Very detailed derivation in “Pattern Recognition and Machine learning” (Christopher M. Bishop)

# Hierarchical clustering (hc) → VB

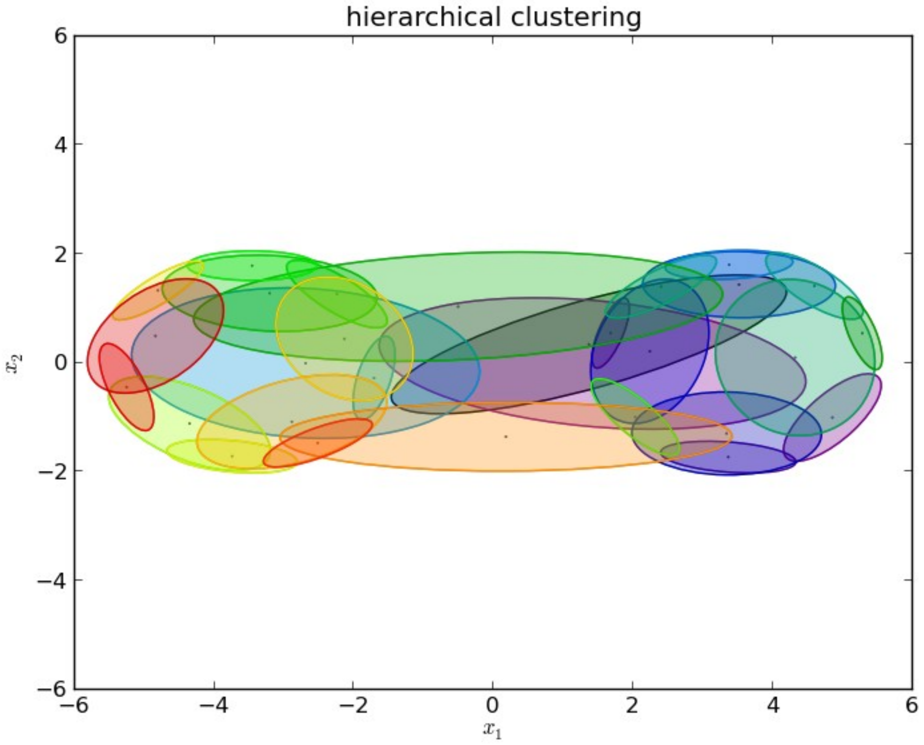
Advantages:

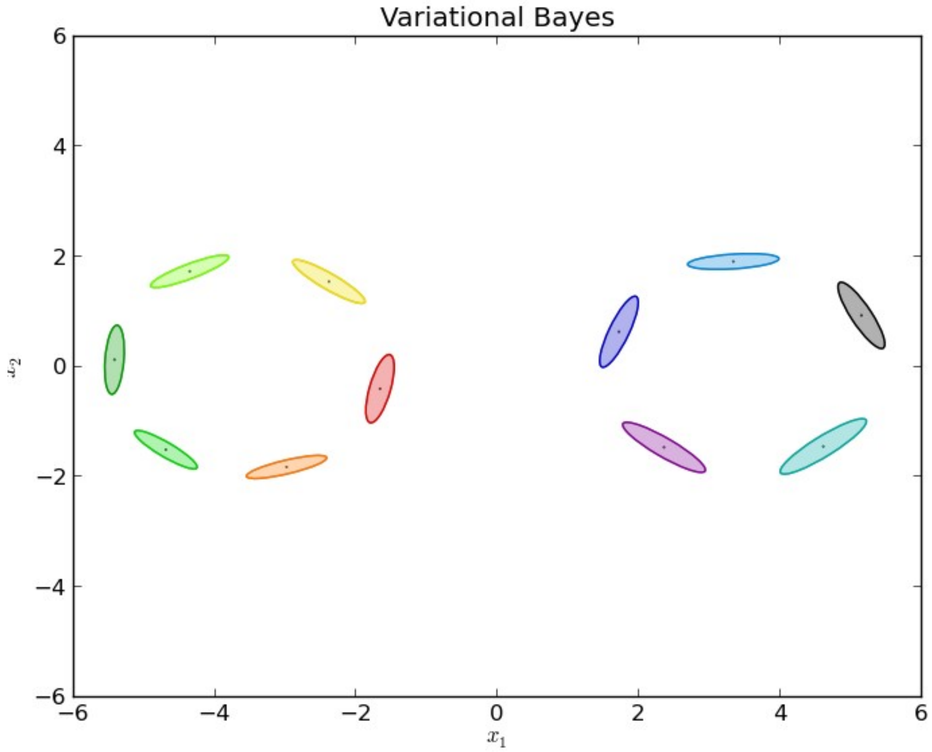
- Can at least reproduce the hc results, most runs bring better perplexity and effective sample size
- Automatically removes components with redundant information

Disadvantage:

- Computationally much more expensive

# Example: Gaussian shells, 2 dim



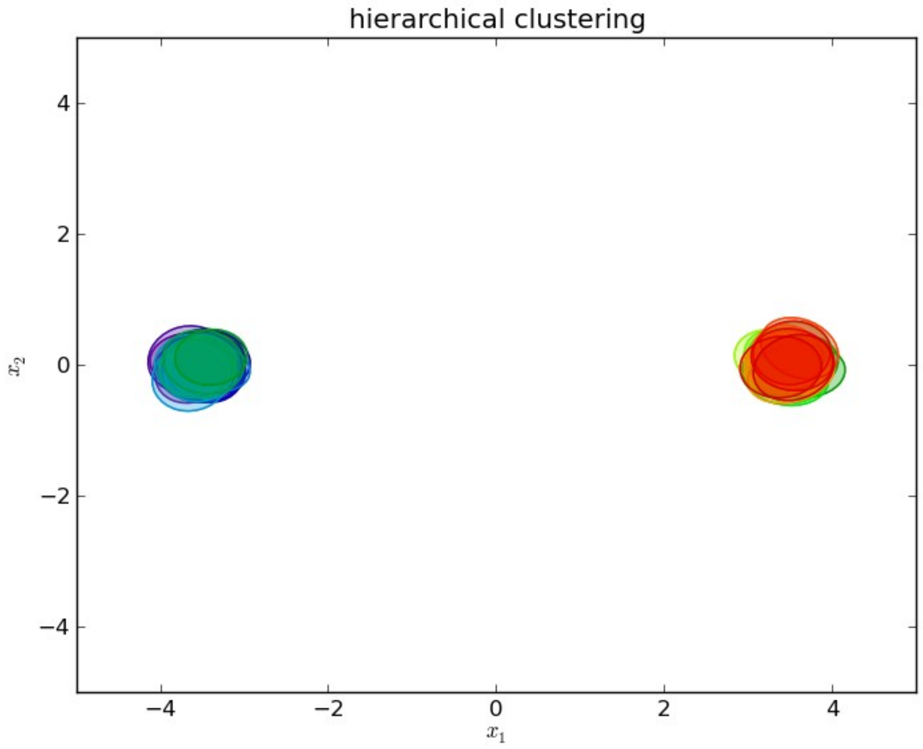


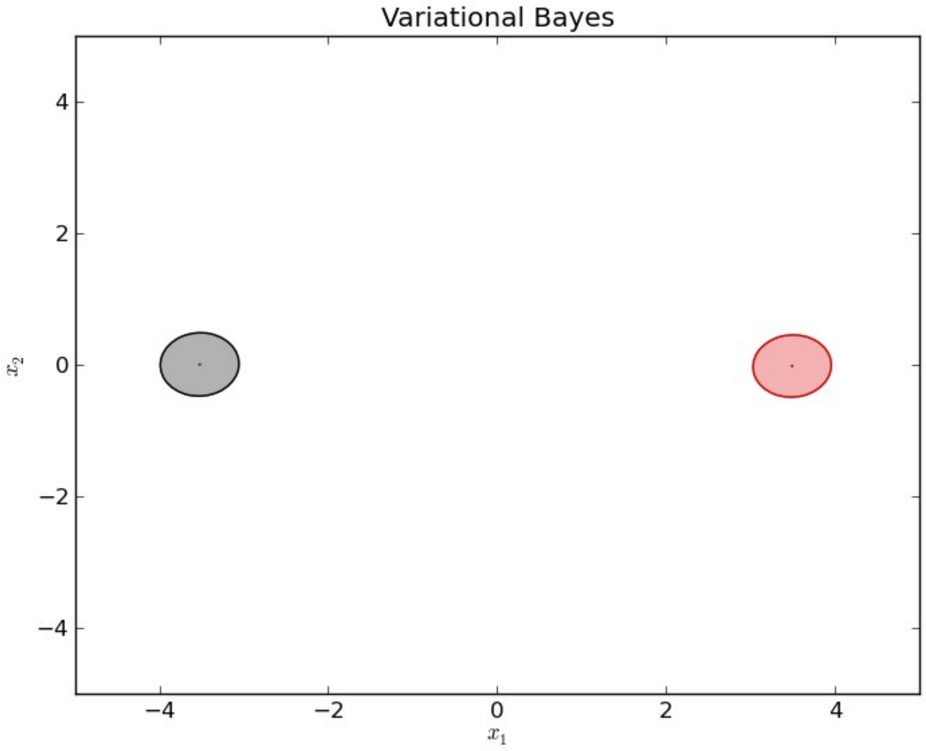
- Start clustering with 15 components per shell

- Converged with 29 components after 21 steps
- Perplexity: ~32%
- ESS: ~27%

- Converged with 11 components after 304 steps
- Perplexity: ~65%
- ESS: ~46%

# Example: Gaussian shells, 20 dim





- Start clustering with 25 components per shell

- Converged with 50 components after 5 steps
- Perplexity: ~33%
- ESS: ~23%

- Converged with 2 components after 304 steps
- Perplexity: ~39%
- ESS: ~28%

# PMC → VB (to come ...)

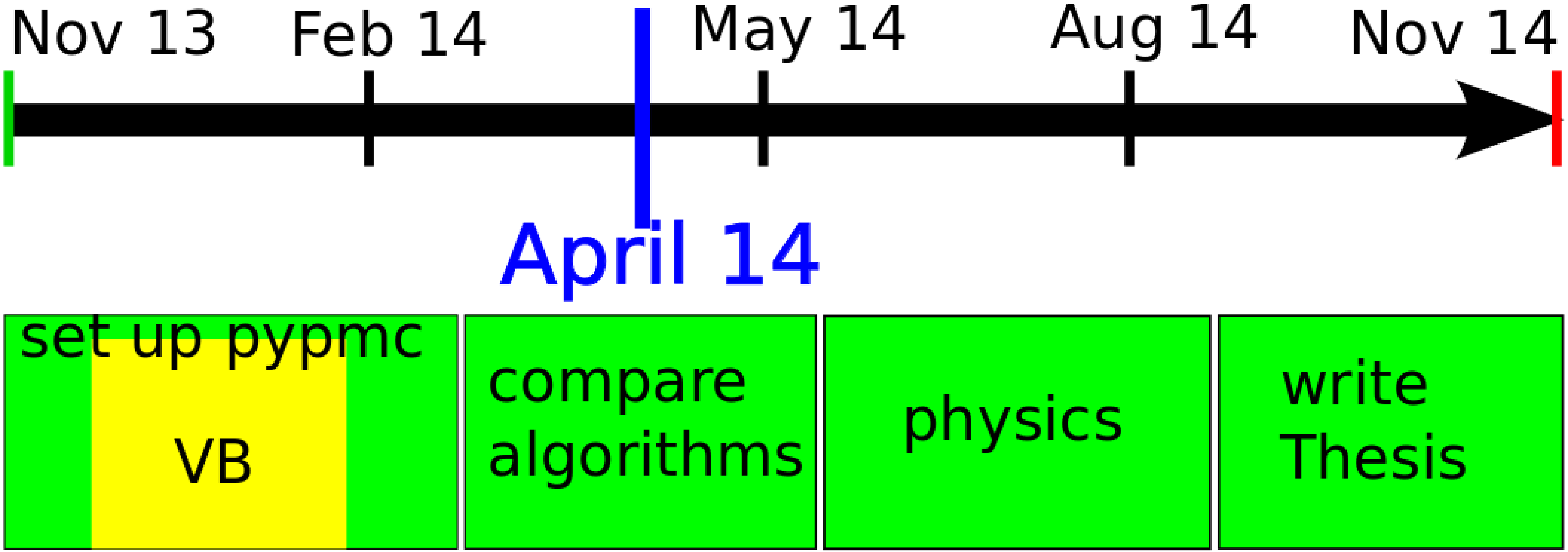
Hopes:

- VB allows (forces) you to define priors for the model parameters
  - ➔ We can set the posterior from the previous step as new prior. → Use all information properly.

Fears:

- Unjustifiable computational effort

# Timeline



Nov 13

Feb 14

May 14

Aug 14

Nov 14

April 14

set up pypmc

VB

compare algorithms

physics

write Thesis

# Variational Bayes

Given data **X**, we can write for the evidence of any model:

$$\ln p(\boldsymbol{X}) = \mathcal{L}(q) + KL(q \| p)$$

where we define:

$$\mathcal{L}(q) \equiv \int q(\boldsymbol{Z}, \boldsymbol{\theta}) \ln \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta})}{q(\boldsymbol{Z}, \boldsymbol{\theta})} \right\} d\boldsymbol{Z}\, d\boldsymbol{\theta}$$

$$KL(q \| p) \equiv -\int q(\boldsymbol{Z}, \boldsymbol{\theta}) \ln \left\{ \frac{p(\boldsymbol{Z}, \boldsymbol{\theta} | \boldsymbol{X})}{q(\boldsymbol{Z}, \boldsymbol{\theta})} \right\} d\boldsymbol{Z}\, d\boldsymbol{\theta}$$

**θ** are model parameters

**Z** are “latent variables” (next slide)

# Variational Bayes

Definition: Latent (hidden) variables

Be **X** = {$x_1$, … ,$x_N$} your data

**Z** = {$z_1$, … ,$z_N$} is called latent if **Z** contains information which can not uniquely be determined from **X** but inferred in a probabilistic model.

Example (Gaussian mixtures):

$$g(x)=\sum_i \omega_i \mathcal{G}_i(x|\mu_i,\Sigma_i)$$

$$\boldsymbol{X}\sim g(x)$$

Then the index i is a latent variable

# Variational Bayes

Given data **X**, we can write for the evidence of any model:

$$\ln p(\boldsymbol{X}) = \mathcal{L}(q) + KL(q\|p)$$

Holds for any probability distribution q

Main idea: Restrict q to factorize as below and minimize the Kullback-Leibler divergence

$$q(\boldsymbol{Z}, \boldsymbol{\theta}) = q(\boldsymbol{Z})\, q(\boldsymbol{\theta})$$

I.e. we look for the closest factorizing q to p in the sense of KL(q||p)

# Variational Bayes

Given data **X**, we can write for the evidence of any model:

$$\ln p(\boldsymbol{X}) = \mathcal{L}(q) + KL(q \| p)$$

Minimizing KL is equivalent to maximizing $\mathcal{L}$.

Maximizing $\mathcal{L}$, in the case of a Gaussian mixture model, leads to an EM-like algorithm.