Hierarchical Bayesian Modeling with Approximate Bayesian Computation

Jessi Cisewski
Carnegie Mellon University

June 2014

Goal of lecture

- Bring the ideas of Hierarchical Bayesian Modeling (HBM), Approximate Bayesian Computation (ABC), importance sampling, and sequential ABC together in an astronomy-motivated application to exoplanets.

Collaborators

Eric Ford

Megan Shabram

Chad Schafer

samsi

NSF • Duke • NCSU • UNC • NISS

SAMSI ExoStat Group

Planet detection

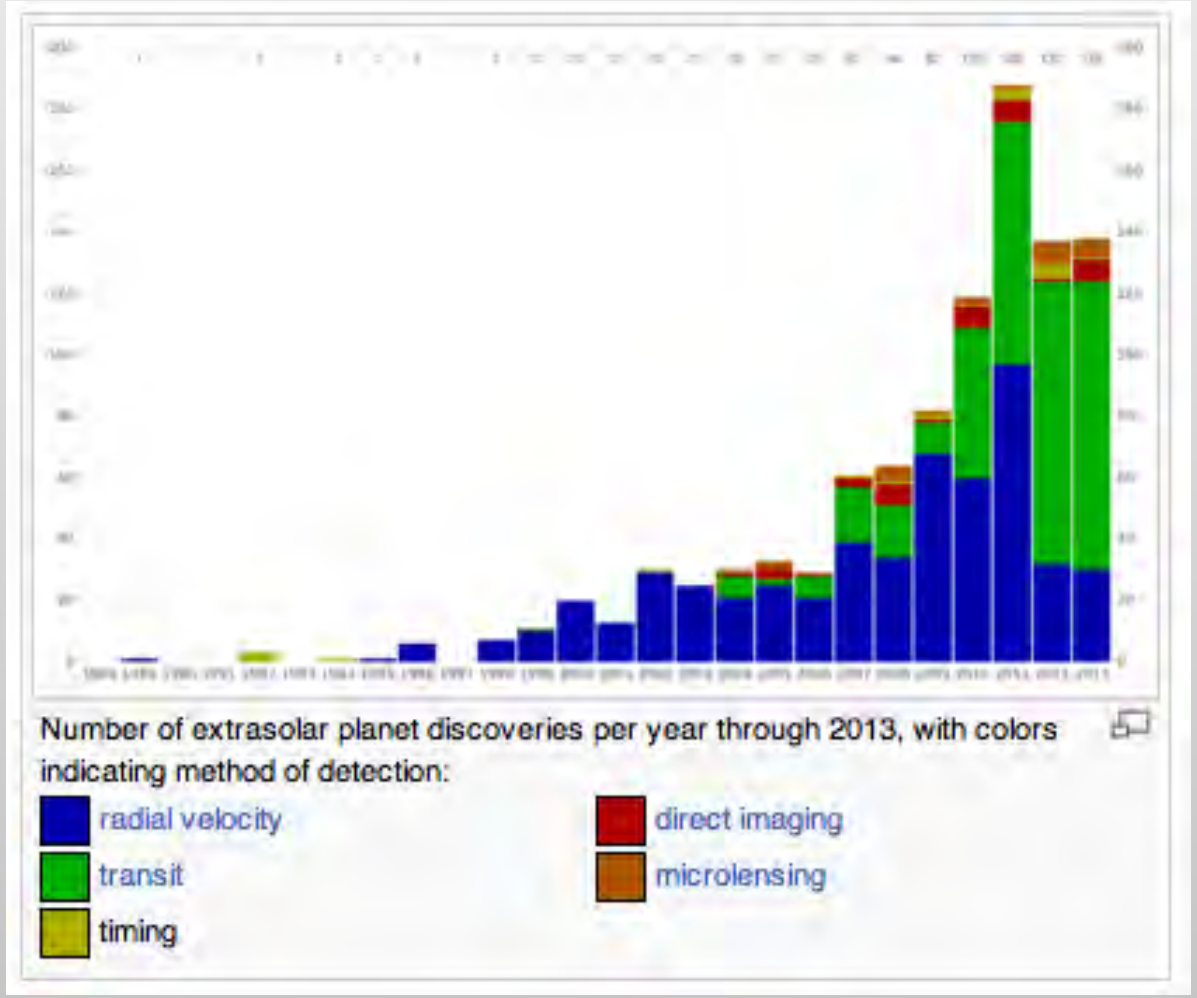
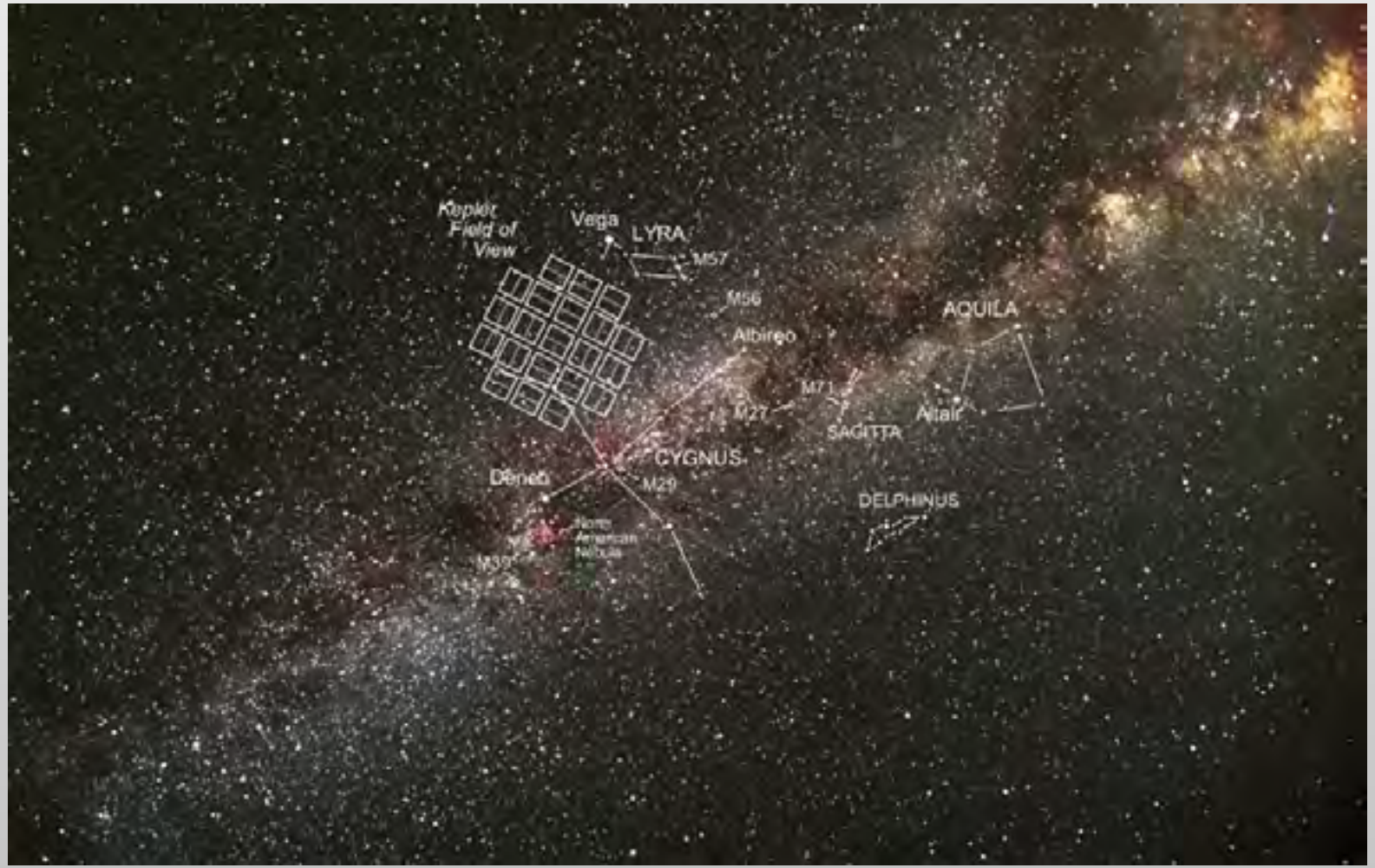


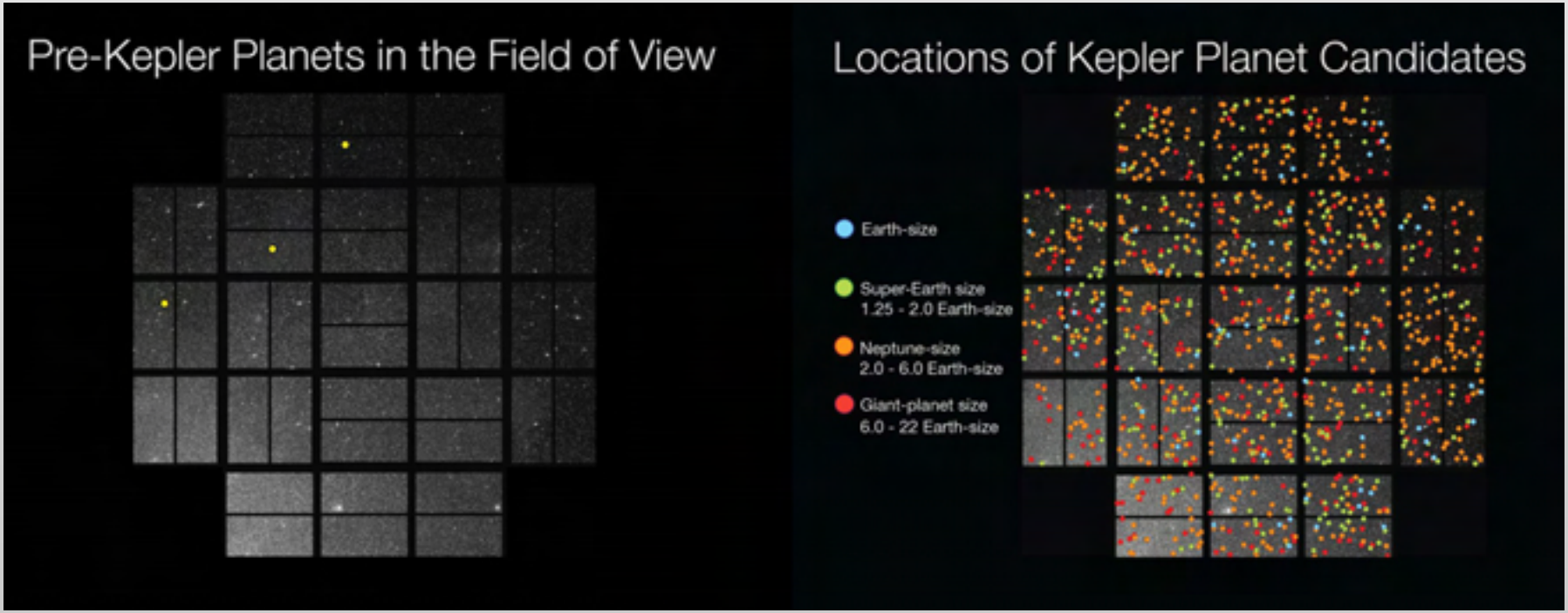
Figure: http://en.wikipedia.org/wiki/File:Exoplanet_Discovery_Methods_Bar.png

Kepler field of view



http://kepler.nasa.gov/images/

Kepler field of view



(pre - 2011)

`http://kepler.nasa.gov/images/`

Each exoplanet, i, has the following parameters:

$$w_i \equiv (R_{s_A}, R_{p_A}, \kappa_i, \Pi_i, \phi_i, e_i, \bar{\omega}_i)$$

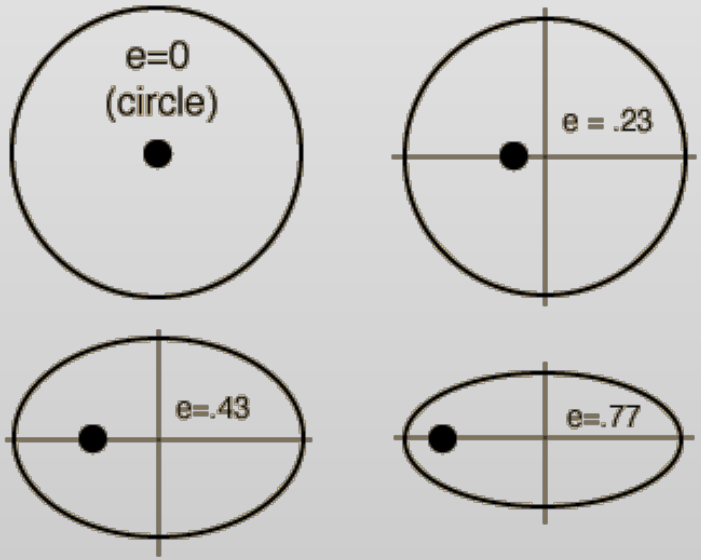
- R_{s_i} = star radius
- R_{p_i} = planet radius
- κ_i = amplitude of velocity model
- Π_i = orbital period
- ϕ_i = orbital phase at fiducial time
- e_i = orbital eccentricity
- $\bar{\omega}_i$ = longitude of perihelion (orientation of ellipse relative to the sky)

Each exoplanet, i, has the following parameters:

$$w_i \equiv (R_{s_A}, R_{p_A}, \kappa_i, \Pi_i, \phi_i, e_i, \bar{\omega}_i)$$

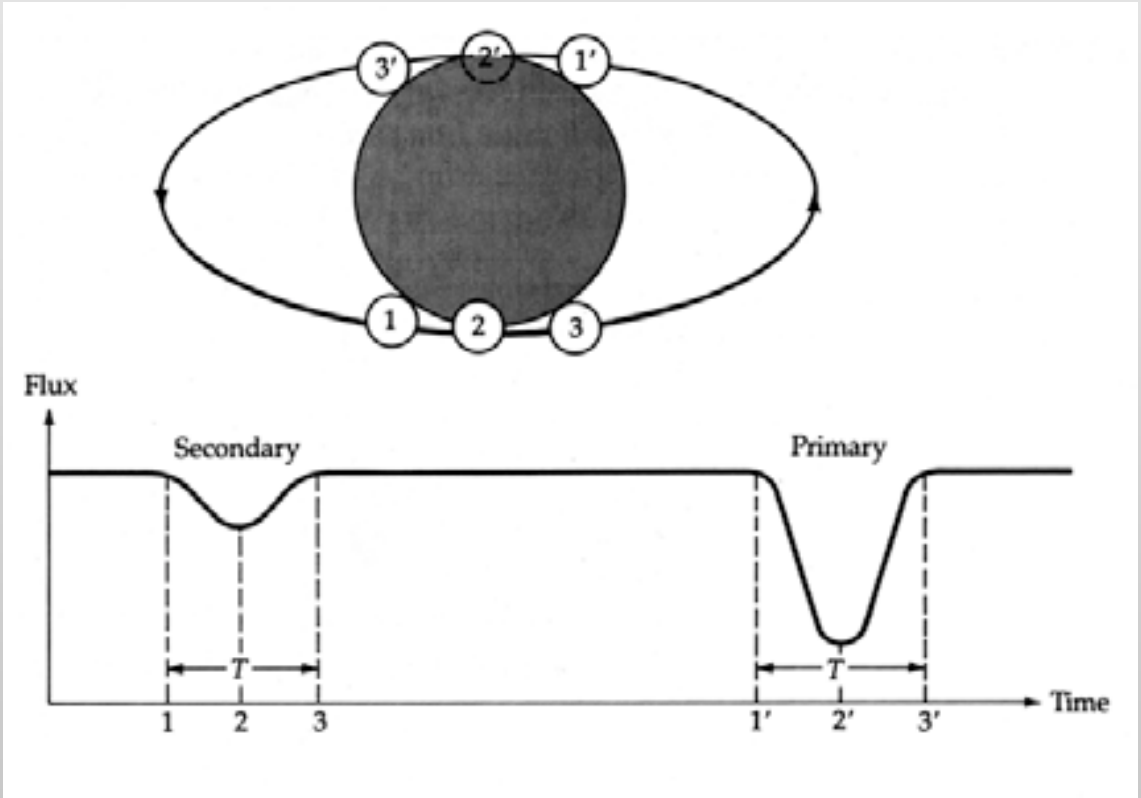
- R_{s_i} = star radius
- R_{p_i} = planet radius
- κ_i = amplitude of velocity model
- Π_i = orbital period
- ϕ_i = orbital phase at fiducial time
- e_i = orbital eccentricity
- $\bar{\omega}_i$ = longitude of perihelion (orientation of ellipse relative to the sky)

Orbital eccentricity



Planet	Eccentricity
Mercury	0.2056
Venus	0.0068
Earth	0.0167
Mars	0.0934
Jupiter	0.0483
Saturn	0.0560
Uranus	0.0461
Neptune	0.0097

Orbital eccentricity



Eccentricity: observables

There is some true (h_i, k_i) for planet i such that

$$\begin{pmatrix} h_i \\ k_i \end{pmatrix} | \text{[some parameters]} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{c_i}^2 & 0 \\ 0 & \sigma_{c_i}^2 \end{pmatrix}\right)$$

- where $h_i = e_i \cos(\bar{\omega}_i)$, $k_i = e_i \sin(\bar{\omega}_i) \implies h_i^2 + k_i^2 = e_i^2$
- $\bar{\omega}_i \sim U[0, 2\pi)$
- But we observe (h_i, k_i) with measurement error

$$\begin{pmatrix} \hat{h}_i \\ \hat{k}_i \end{pmatrix} | \text{[some parameters]} \sim N\left(\begin{pmatrix} h_i \\ k_i \end{pmatrix}, \begin{pmatrix} \sigma_h^2 & 0 \\ 0 & \sigma_k^2 \end{pmatrix}\right)$$

Eccentricity: [some parameters]

Given a sample of exoplanets $p_1, \ldots, p_n$, we assume the following

1. N_m "populations" (assume $N_m = 1, 2,$ or 3)
2. Classify each population as $c_1, \ldots, c_{N_m}$
3. Each population has a certain true proportion f_j, $j = 1, \ldots, N_m$ and $\sum_{j=1}^{N_m} f_j = 1$

Bayesian hierarchical model for eccentricity

$$N_m = \#\text{ "populations" (fixed and "known")}$$

$$f \sim \text{Dirichlet}(1_{N_m})$$

$$c_i \mid f \overset{iid}{\sim} \text{multinomial}(f)$$

$$\sigma_{c_i} \overset{iid}{\sim} U(0,1)$$

$$\begin{pmatrix} h_i \\ k_i \end{pmatrix} | c_i, \sigma_{c_i}^2 \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{c_i}^2 & 0 \\ 0 & \sigma_{c_i}^2 \end{pmatrix}\right)$$

$$\begin{pmatrix} \hat{h}_i \\ \hat{k}_i \end{pmatrix} | c_i, \sigma_{c_i}^2, h_i, k_i \sim N\left(\begin{pmatrix} h_i \\ k_i \end{pmatrix}, \begin{pmatrix} \hat{\sigma}_h^2 & 0 \\ 0 & \hat{\sigma}_k^2 \end{pmatrix}\right)$$

3D Dirichlet Distribution

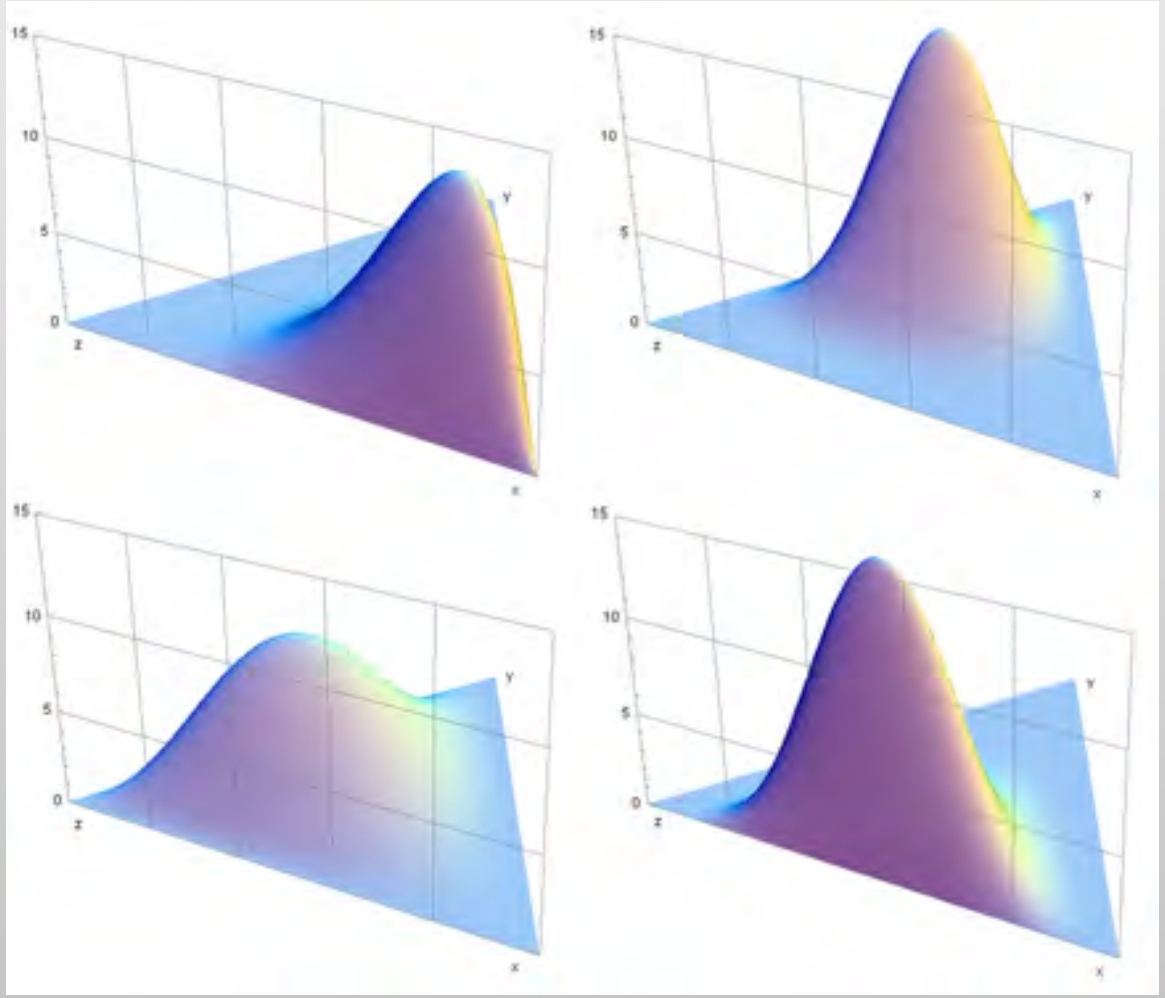


Image: `http://upload.wikimedia.org`

Beta Distribution

$$f(y \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot y^{\alpha-1}(1-y)^{\beta-1}$$

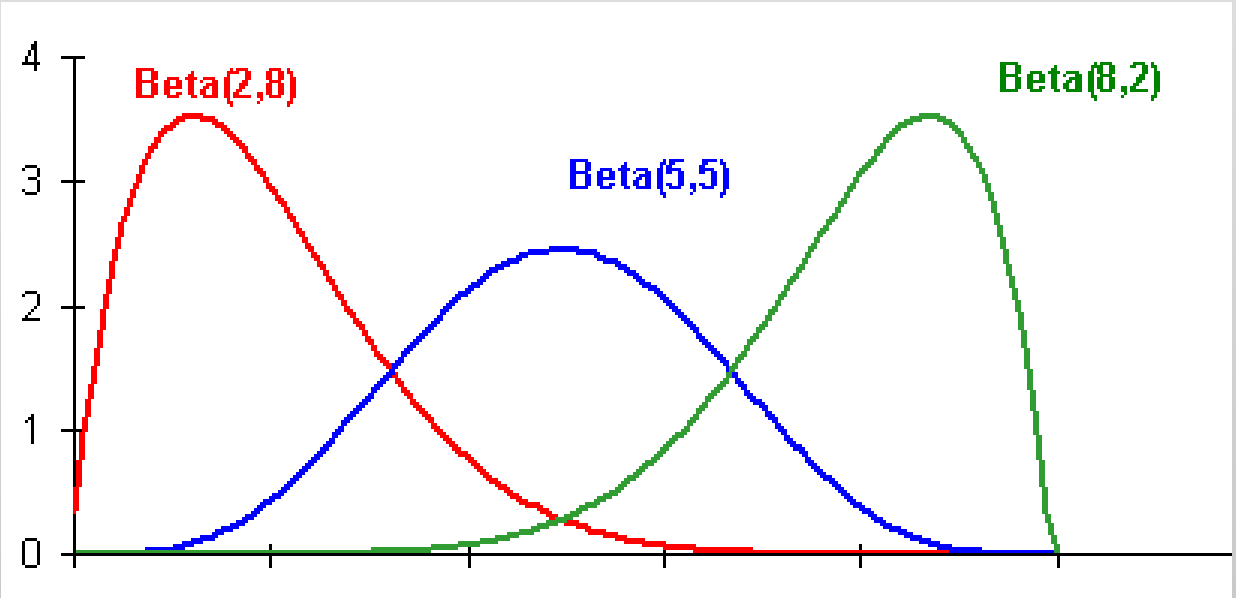


Image: `http://www.vosesoftware.com`

Recall the Basic ABC algorithm

For the observed data $y_{1:n}$, prior $\pi(\theta)$ and distance function ρ:

Algorithm

1. Sample θ^* from prior $\pi(\theta)$
2. Generate $x_{1:n}$ from forward process $f(y \mid \theta^*)$
3. Accept θ^* if $\rho(y_{1:n}, x_{1:n}) < \epsilon$
4. Return to step 1

Generates a sample from an approximation of the posterior:

$$f(x_{1:n} \mid \rho(y_{1:n}, x_{1:n}, \theta) < \epsilon) \cdot \pi(\theta) \approx f(y_{1:n} \mid \theta)\pi(\theta) \propto \pi(\theta \mid y_{1:n})$$

Summary of basic ABC

- Decisions that need to be made:
  1. Distance function (ρ)
  2. Summary statistic(s)
  3. Tolerance (ϵ)
- Finding the "right" ϵ can be inefficient
  $\longrightarrow$ we end up throwing away many of the theories proposed from the selected priors
  $\longrightarrow$ use sequential sampling to improve efficiency

Each exoplanet, i, has the following parameters:

$$w_i \equiv (R_{s_A}, R_{p_A}, \kappa_i, \Pi_i, \phi_i, e_i, \bar{\omega}_i)$$

- R_{s_i} = star radius
- R_{p_i} = planet radius
- κ_i = amplitude of velocity model
- Π_i = orbital period
- ϕ_i = orbital phase at fiducial time
- e_i = orbital eccentricity
- $\bar{\omega}_i$ = longitude of perihelion (orientation of ellipse relative to the sky)

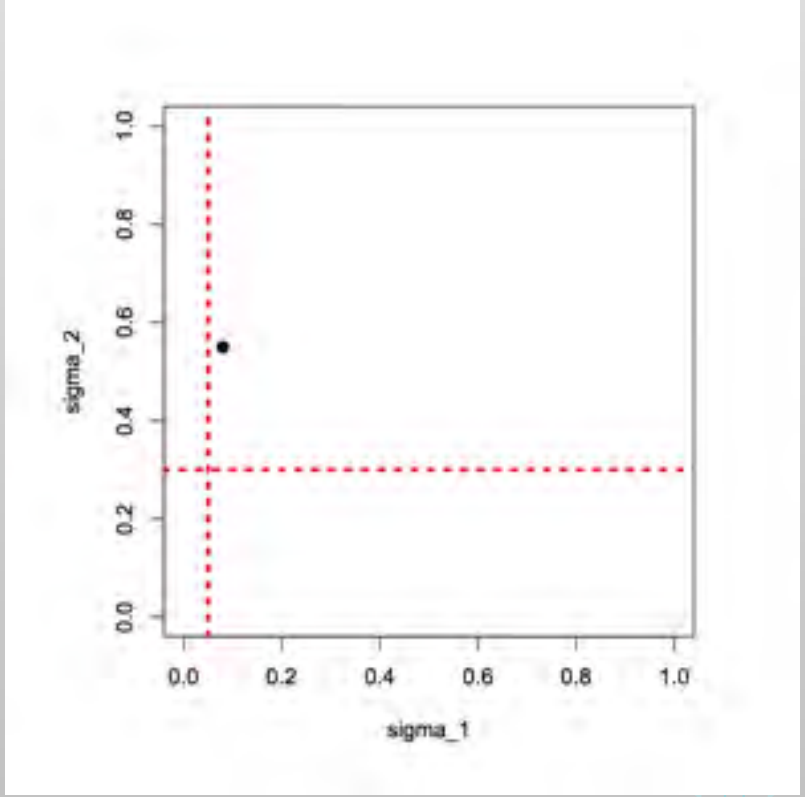
The "observations"

- The observations are simulated from the forward process (i.e. the hierarchical model)
- Two populations of exoplanets ($N_m = 2$)
- True parameter values:
  1. Standard deviations of h's and k's for each population: $(\sigma_1, \sigma_2) = (0.05, 0.30)$
  2. Weights of each population: $(f_1, f_2) = (.7, .3)$
- $n = 500$ observations

```
for(i in 1:N){
 while(d>epsilon0[t]) {
 #------------------------------------------------------ prior on (f1,f2) ~ Dirichlet(1)
  gamma0 = rgamma(Nm,shape = alpha0 ,scale = 1)
  f.proposed0 = gamma0/sum(gamma0)

 #------------------------------------------------------ prior on (sig1, sig2) ~ Uniform(0,1)
  sigma.proposed0 = runif(Nm)
  index0 = sample.int(Nm, size = n, replace = TRUE, prob = f.proposed0)
  sigma.proposed = sigma.proposed0[index0]

 #------------------------------------------------------ generate the h's and k's
  theta1.h = sigma.proposed*rnorm(n)
  theta1.k = sigma.proposed*rnorm(n)

 #------------------------------------------------------ add measurement error
  theta1.hhat = theta1.h + sigma.hhat*rnorm(n)
  theta1.khat = theta1.k + sigma.khat*rnorm(n)

  #------------------------------------------------------ check the distance
  x[,1] = theta1.hhat
  x[,2] = theta1.khat
  d = distance.function(x, data0)
   }

  #------------------------------------------------------ if distance <= epsilon, then keep
  d0[i,t] = d
  sigma.final[i, ,t]= sigma.proposed0
  f.final[i,,t]= f.proposed0
  gamma.sum[i,t] = sum(gamma0)
    }
```

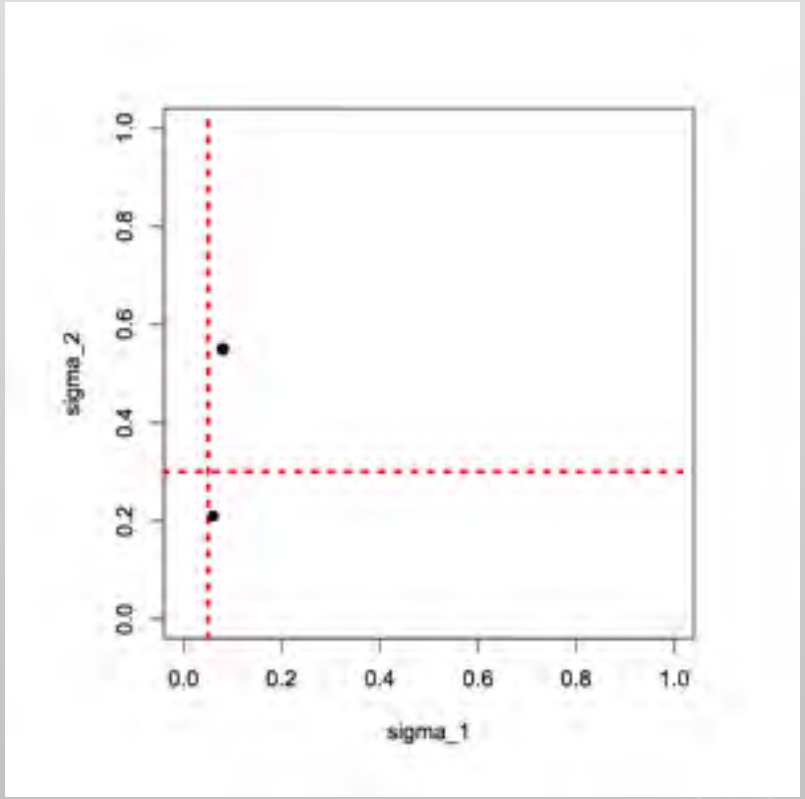
Draw from the prior

```
#------------------------------------------------------ prior on (f1,f2) ~ Dirichlet(1)
 gamma0 = rgamma(Nm,shape = alpha0 ,scale = 1)
 f.proposed0 = gamma0/sum(gamma0)
#------------------------------------------------------ prior on (sig1, sig2) ~ Uniform(0,1)
 sigma.proposed0 = runif(Nm)
 index0 = sample.int(Nm, size = n, replace = TRUE, prob = f.proposed0)
 sigma.proposed = sigma.proposed0[index0]
```



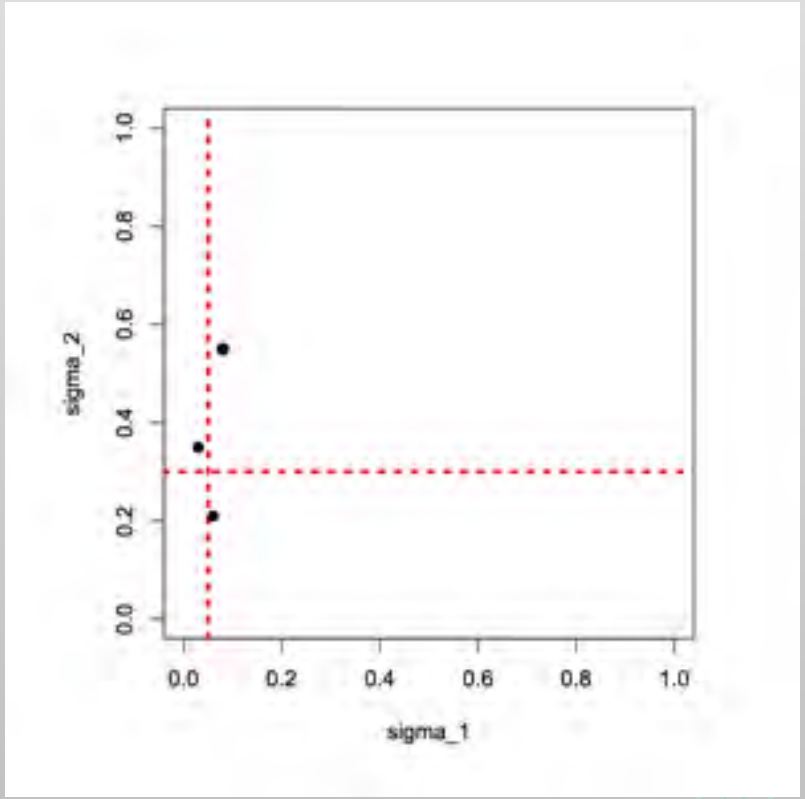
Draw from the prior

```
#------------------------------------------------------ prior on (f1,f2) ~ Dirichlet(1)
 gamma0 = rgamma(Nm,shape = alpha0 ,scale = 1)
 f.proposed0 = gamma0/sum(gamma0)
#------------------------------------------------------ prior on (sig1, sig2) ~ Uniform(0,1)
 sigma.proposed0 = runif(Nm)
 index0 = sample.int(Nm, size = n, replace = TRUE, prob = f.proposed0)
 sigma.proposed = sigma.proposed0[index0]
```



Draw from the prior

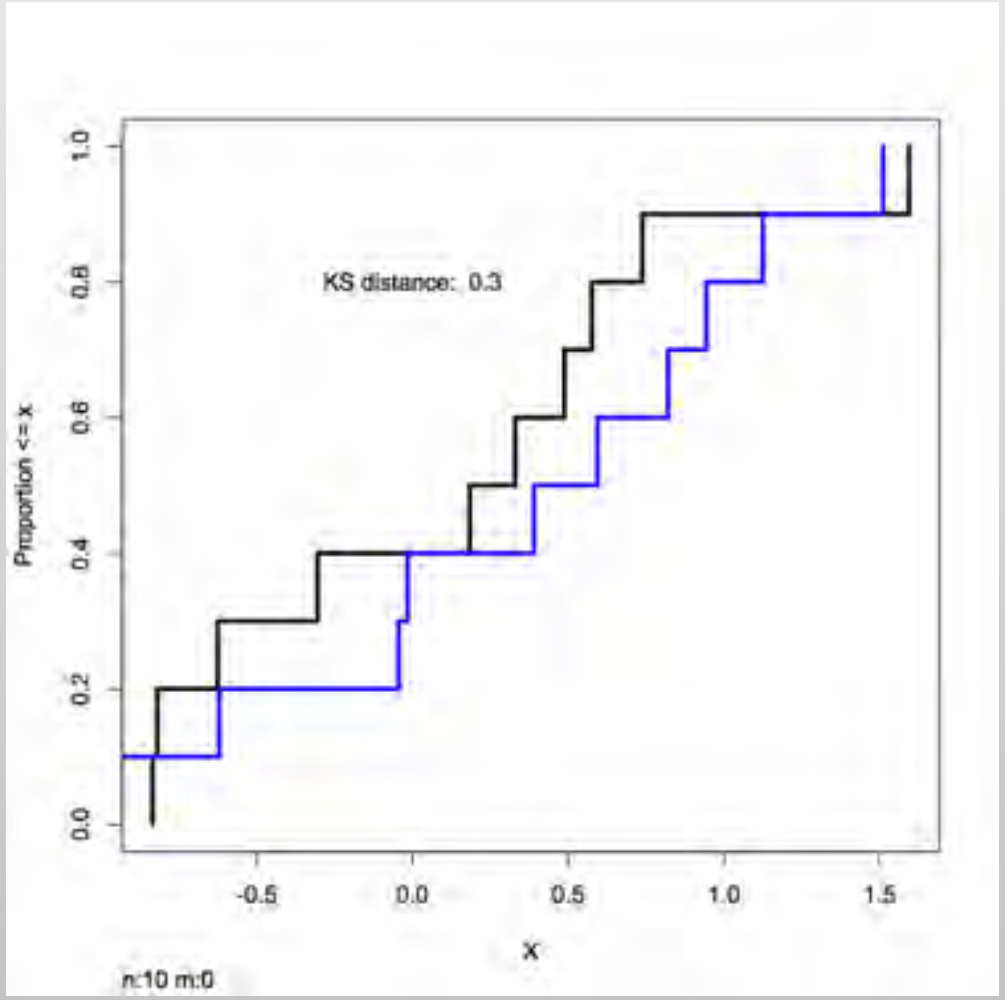
```
#------------------------------------------------------ prior on (f1,f2) ~ Dirichlet(1)
 gamma0 = rgamma(Nm,shape = alpha0 ,scale = 1)
 f.proposed0 = gamma0/sum(gamma0)
#------------------------------------------------------ prior on (sig1, sig2) ~ Uniform(0,1)
 sigma.proposed0 = runif(Nm)
 index0 = sample.int(Nm, size = n, replace = TRUE, prob = f.proposed0)
 sigma.proposed = sigma.proposed0[index0]
```



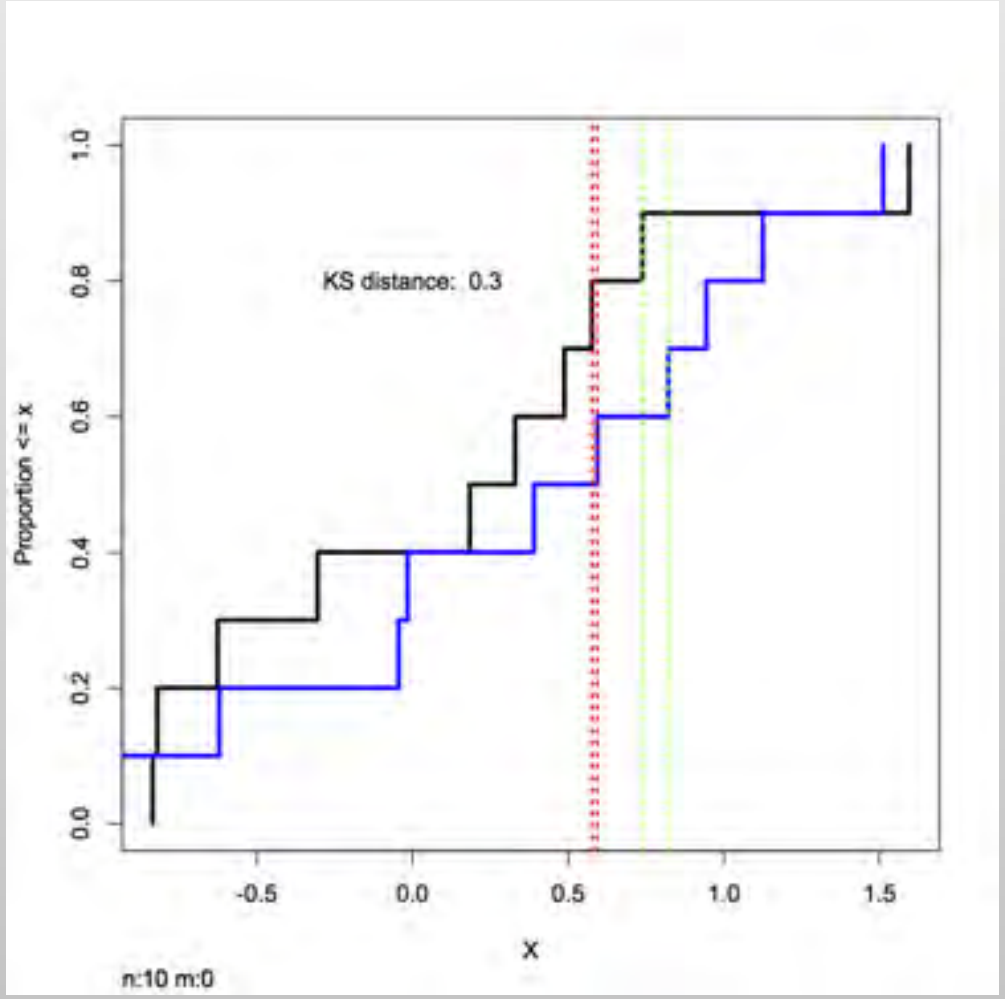
Forward model

```
#------------------------------ generate the h's and k's
 theta1.h = sigma.proposed*rnorm(n)
 theta1.k = sigma.proposed*rnorm(n)


#------------------------------  add measurement error
 theta1.hhat = theta1.h + sigma.hhat*rnorm(n)
 theta1.khat = theta1.k + sigma.khat*rnorm(n)


#------------------------------ check the distance
 x[,1] = theta1.hhat
 x[,2] = theta1.khat
 d = distance.function(x, data0)
   }
```

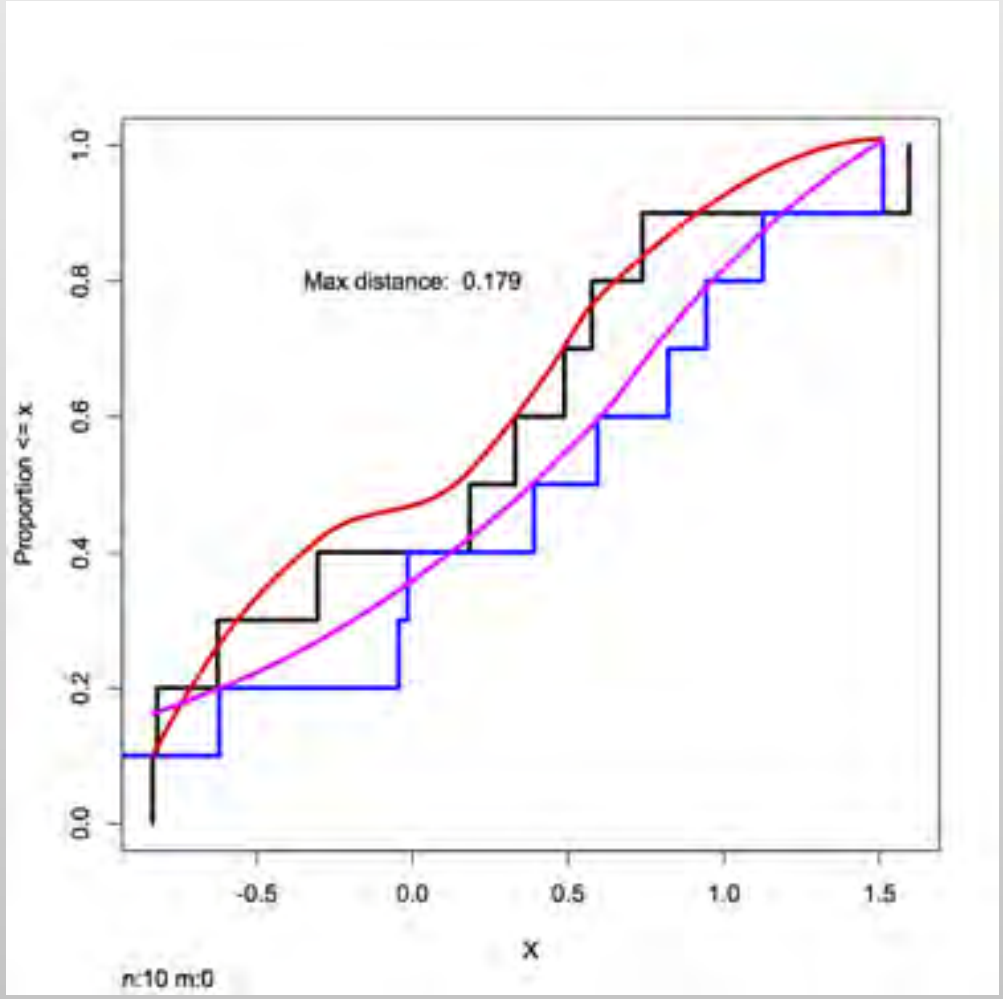
Distance function: Kolmogorov-Smirnov distance



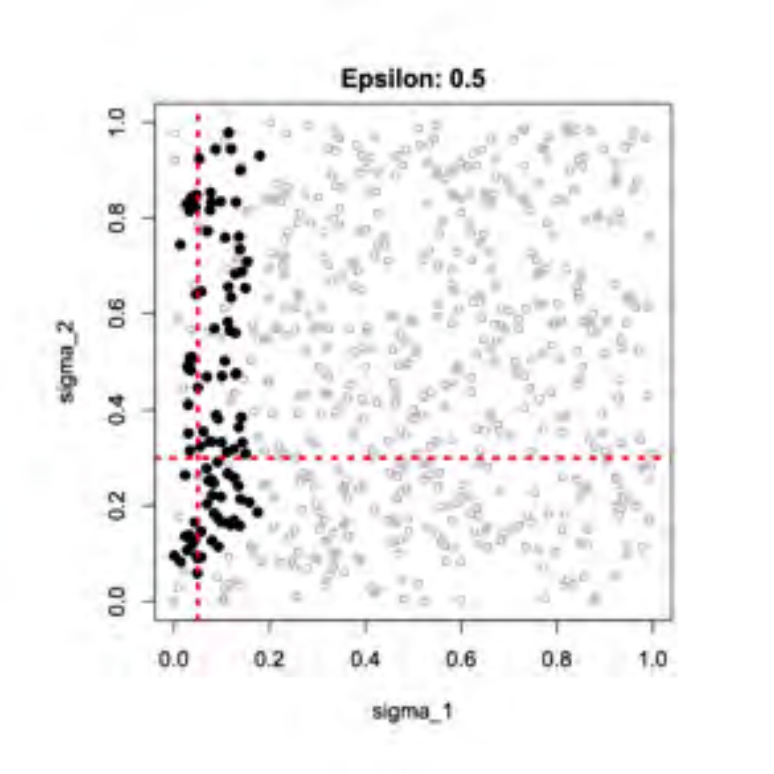
Distance function: Kolmogorov-Smirnov distance

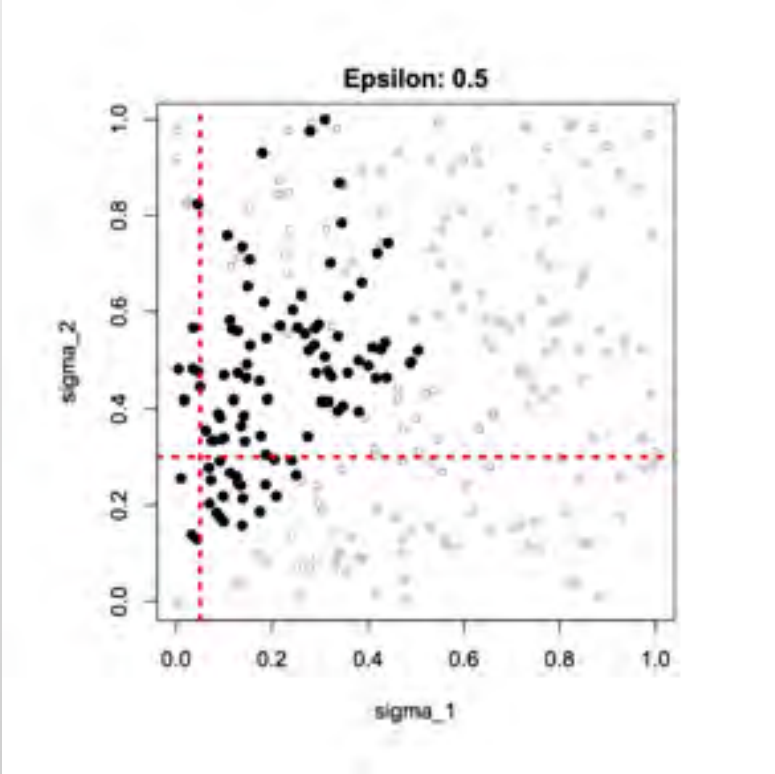


Distance function: Kolmogorov-Smirnov distance

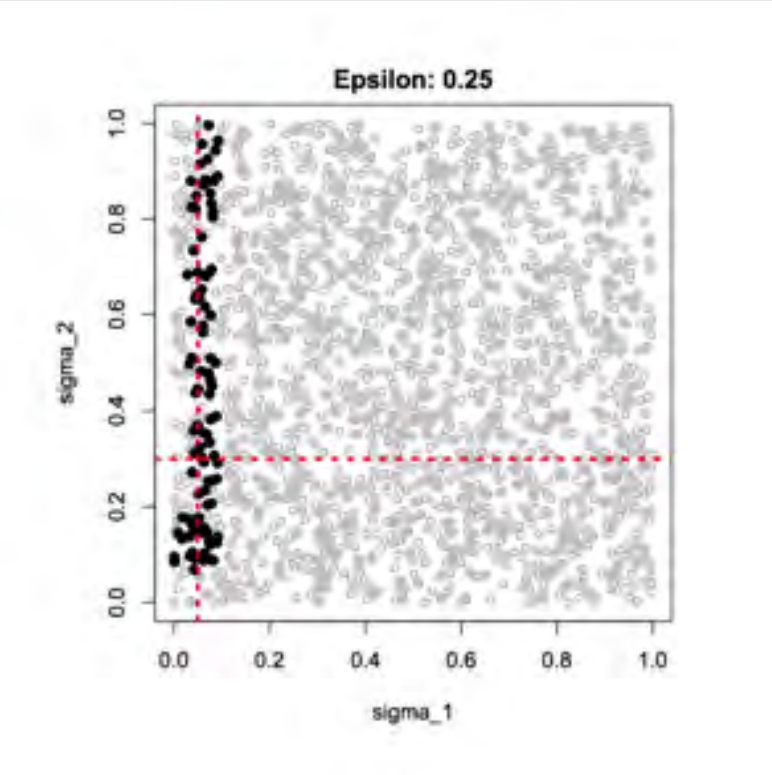


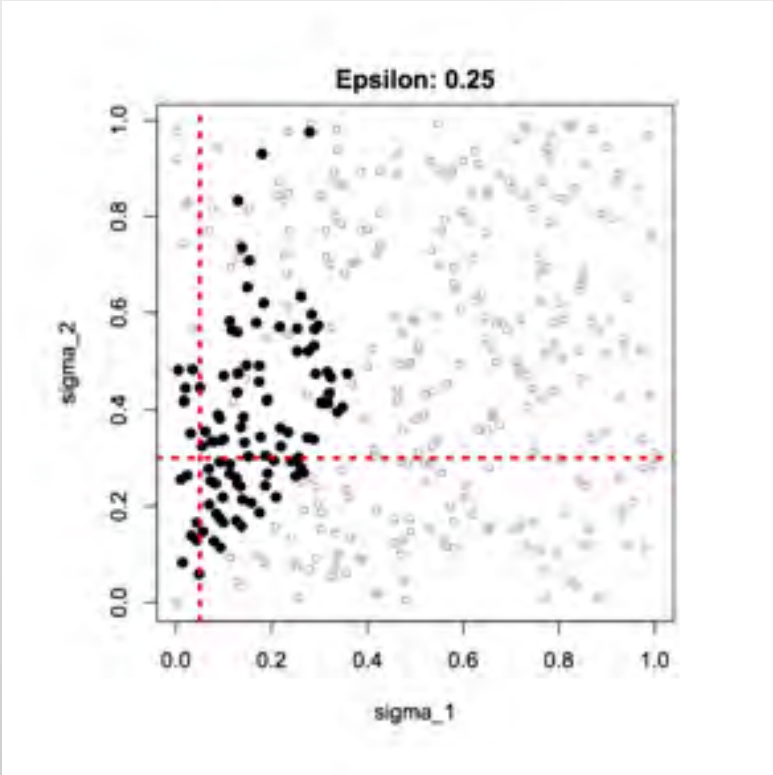
Does the distance function really matter?



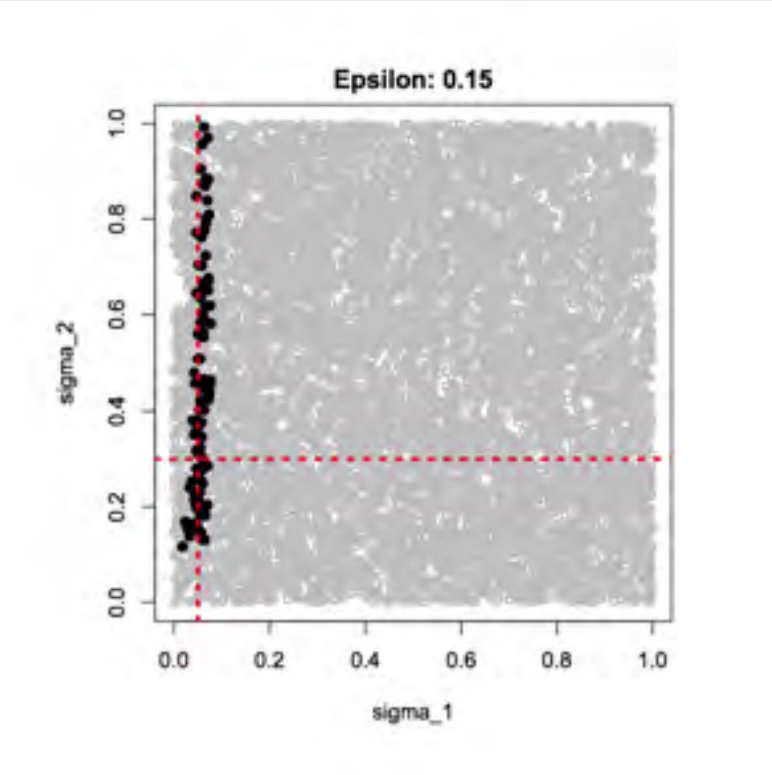


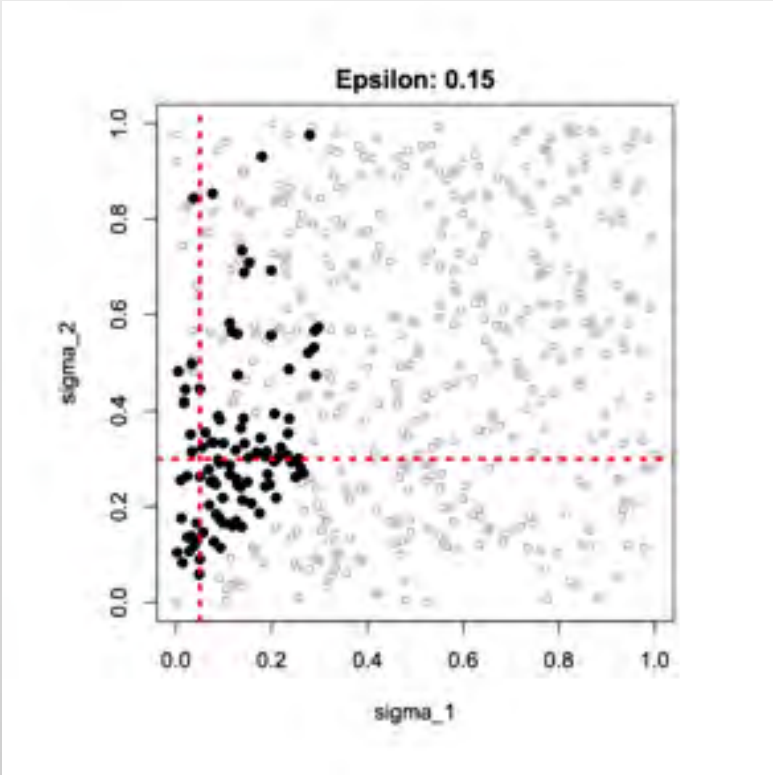
- KS distance on e^2 (left); mean difference of e^2 (right)
- Drawing 100 particles



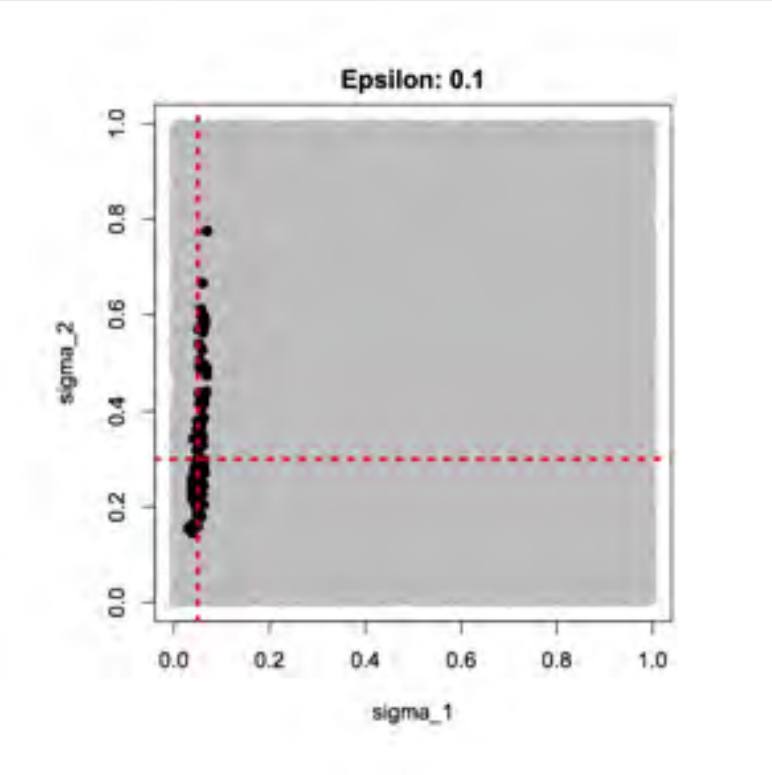


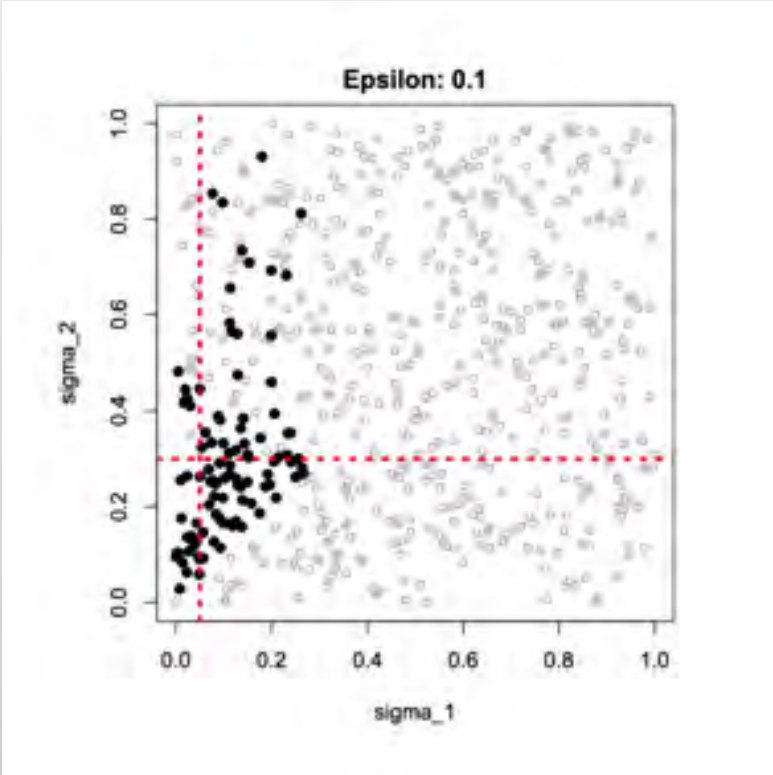
- KS distance on e^2 (left); mean difference of e^2 (right)
- Drawing 100 particles





- KS distance on e^2 (left); mean difference of e^2 (right)
- Drawing 100 particles

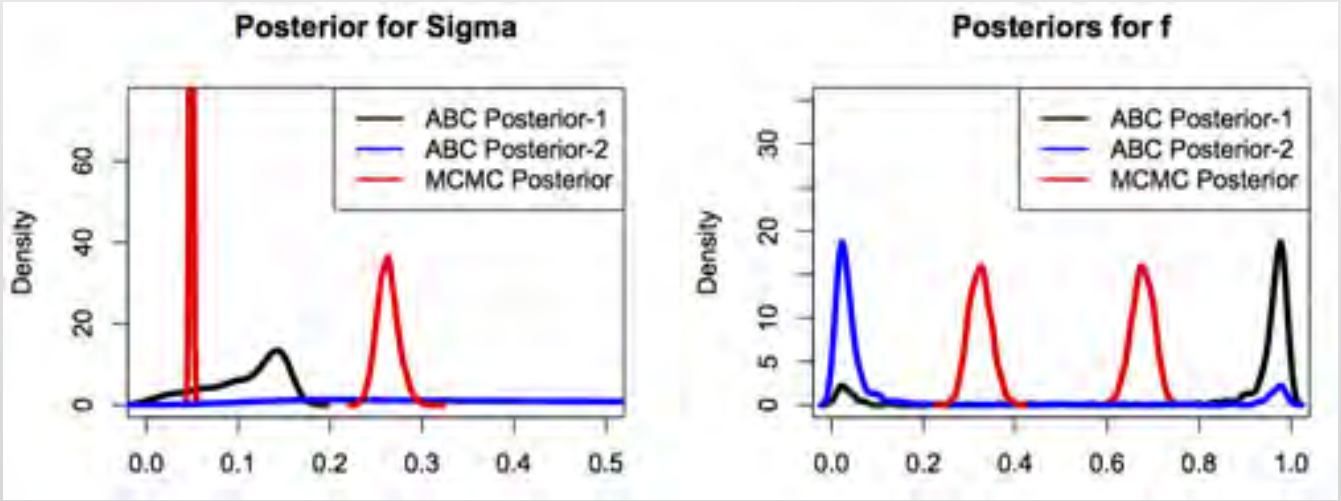




- KS distance on e^2 (left); mean difference of e^2 (right)
- Drawing 100 particles

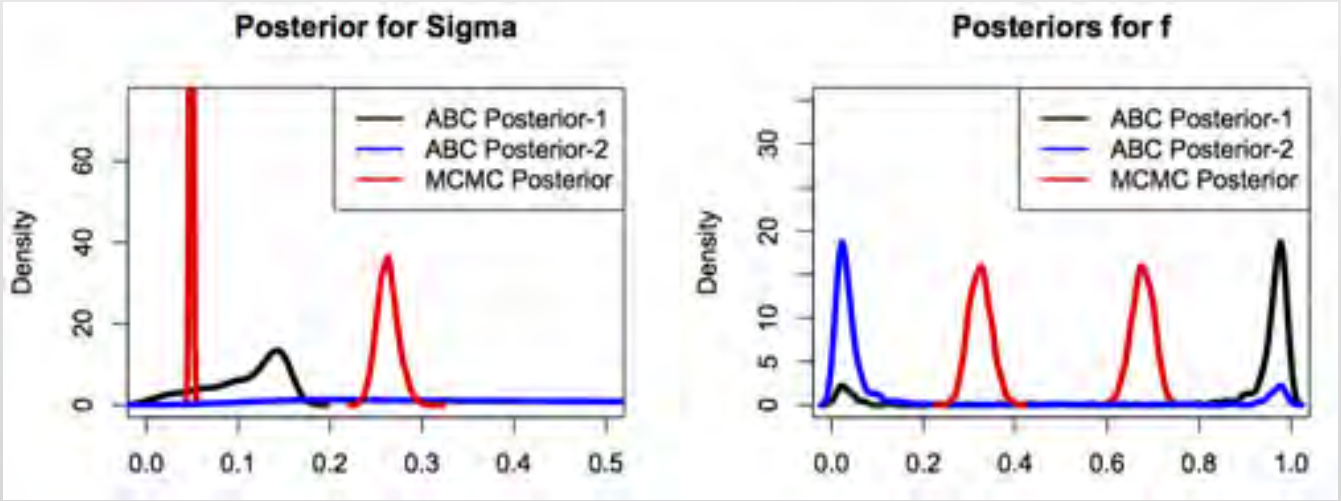
$N = 500$ particles

Mean difference of e^2

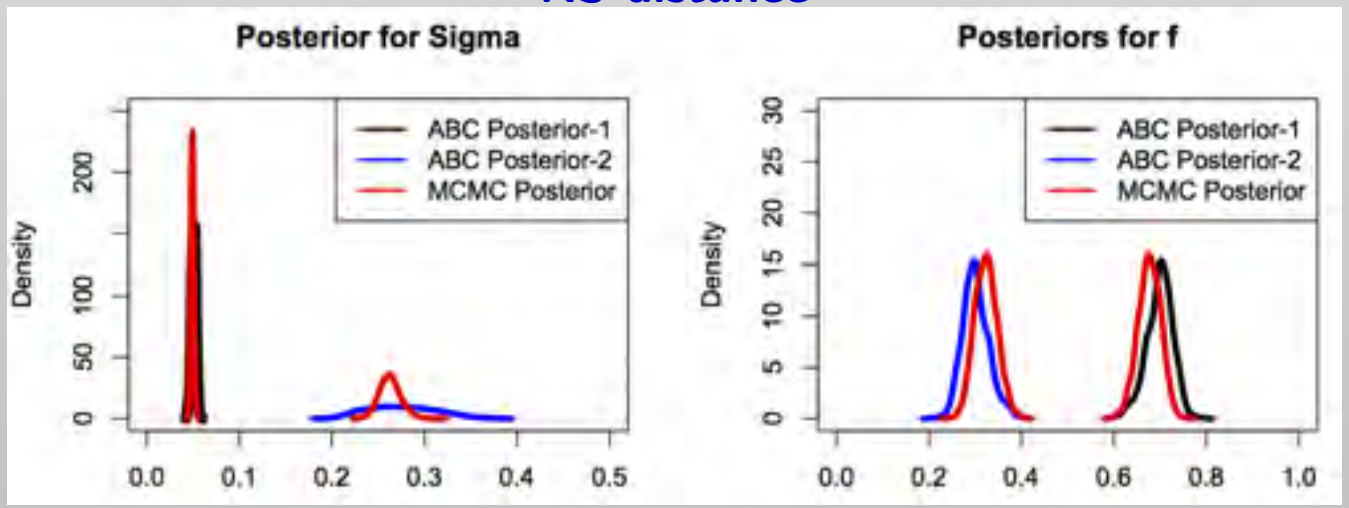


$N = 500$ particles

Mean difference of e^2



KS distance



Decreasing tolerances $\epsilon_1 \geq \cdots \geq \epsilon_T$

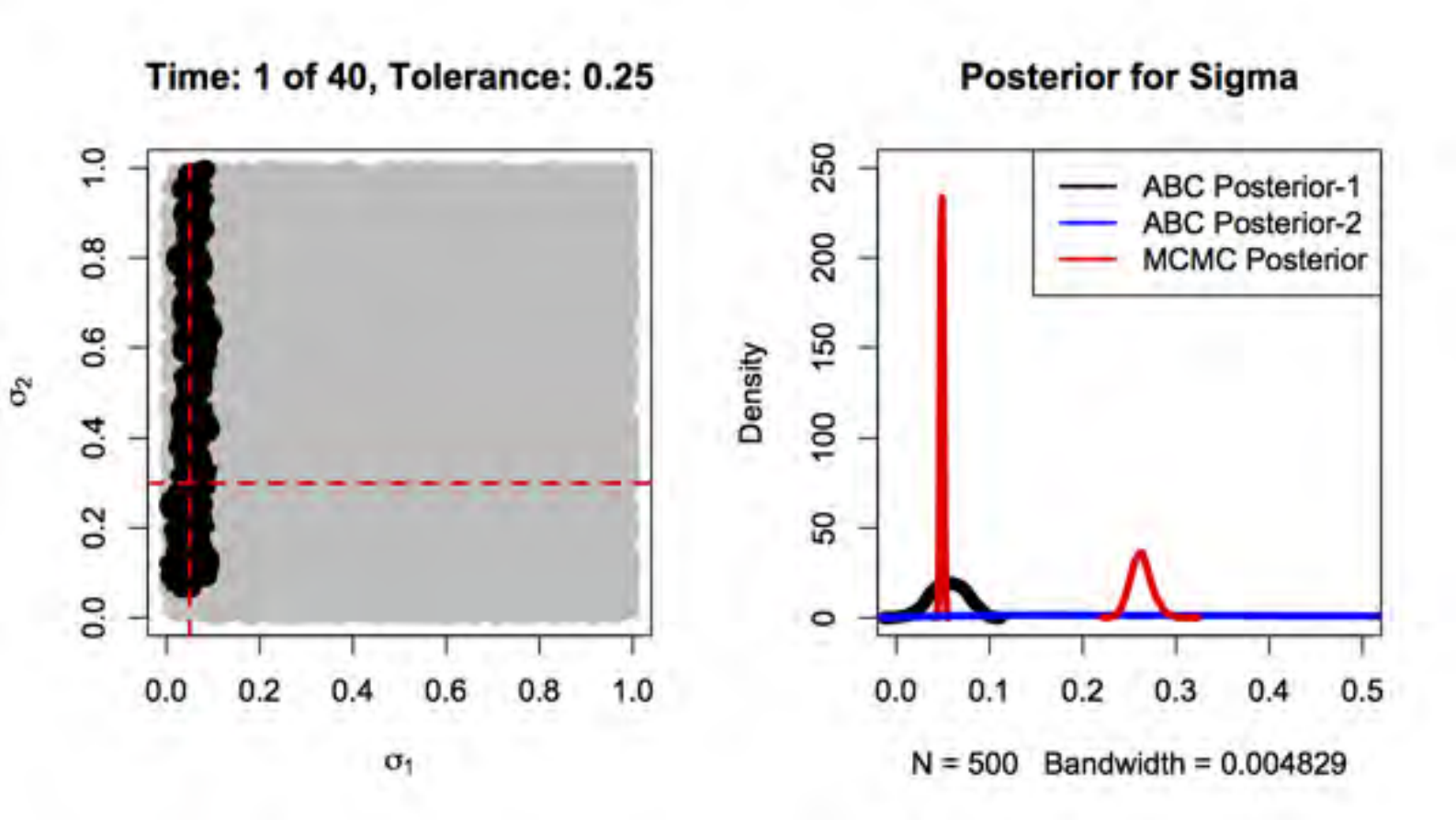
ABC - Population Monte Carlo algorithm* (ABC - PMC)

1. At $t = 1$
   For $i = 1, \ldots, N$ particles
   Generate $\theta_i^{(1)} \sim \pi(\theta)$ and $x \sim f(y \mid \theta_i^1)$ until $\rho(y, x) < \epsilon_1$
   Set $w_i^{(1)} = N^{-1}$
2. At $t = 2, \ldots, T$
   Set $\tau_t^2 = 2 \cdot \text{var}\left(\theta_{1:N}^{(t-1)}\right)$
   For $i = 1, \ldots, N$ particles
   Draw $\theta_i^* \sim \text{multinomial}\left(\theta_{1:N}^{(t-1)}, w_{1:N}^{(t-1)}\right)$
   Generate $\theta_i^{(t)} \mid \theta_i^* \sim N(\theta_i^*, \tau_t^2)$ and $x \sim f(y \mid \theta_i^{(t)})$ until $\rho(y, x) < \epsilon_t$
   Set $w_i^{(t)} \propto \pi(\theta_i^{(t)}) / \sum_{j=1}^N w_j^{(t-1)} \phi[\tau_t^{-1}(\theta_i^{(t)} - \theta_j^{(t-1)})]$

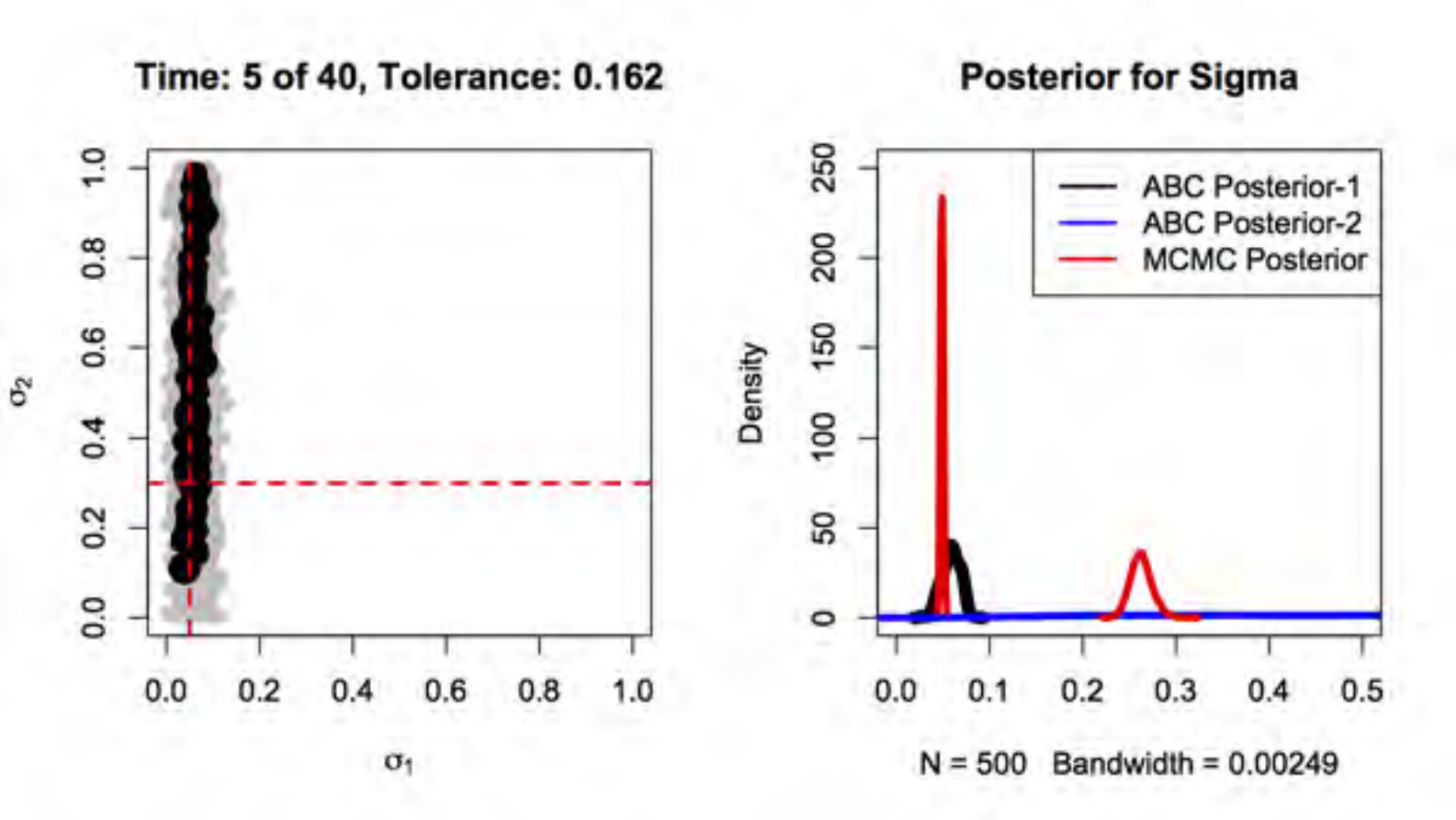
- $\phi(\cdot)$ is the density function of a $N(0, 1)$

*From Beaumont et al. (2009)

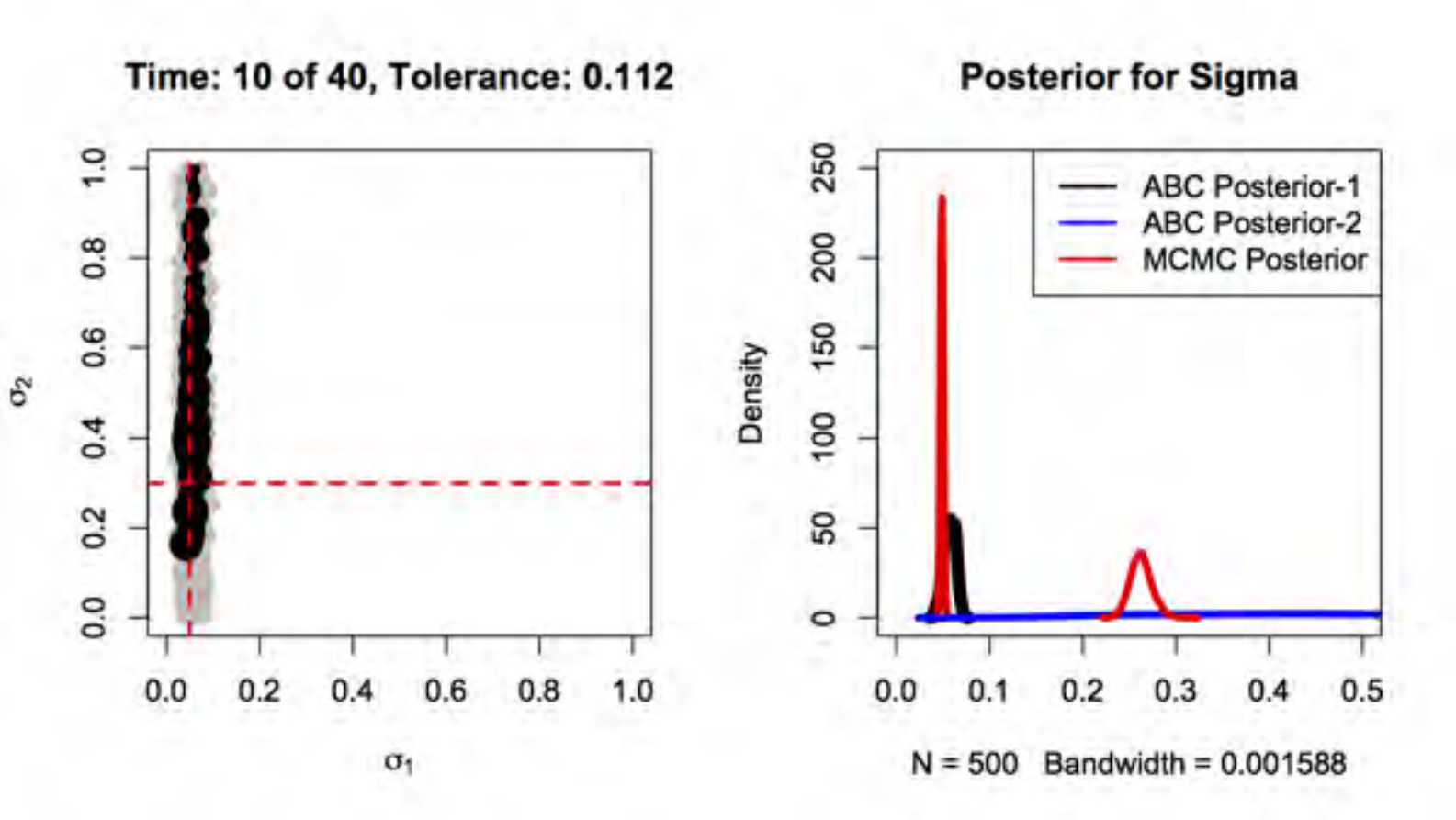
KS -distance with N = 500 particles



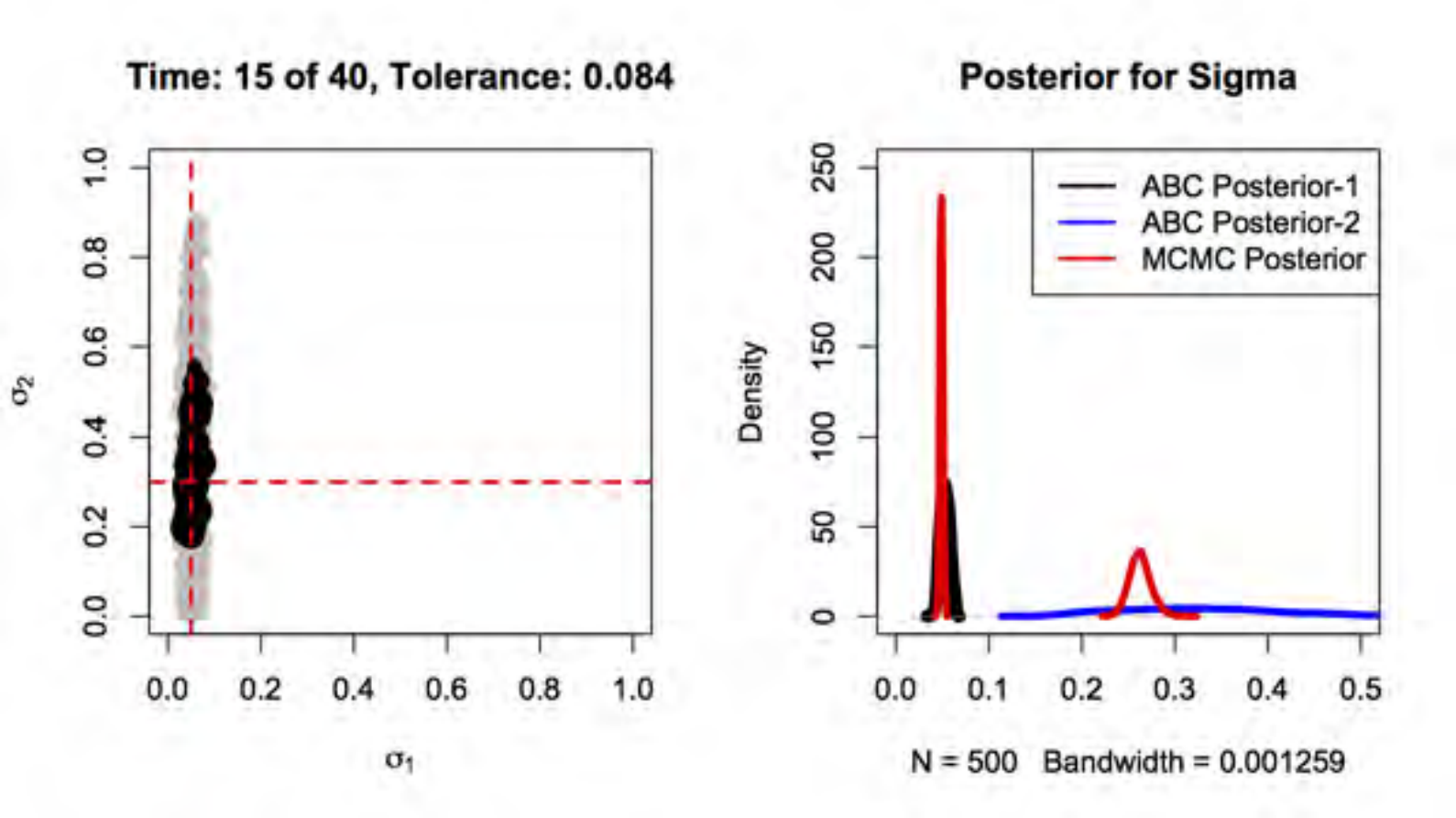
KS -distance with N = 500 particles



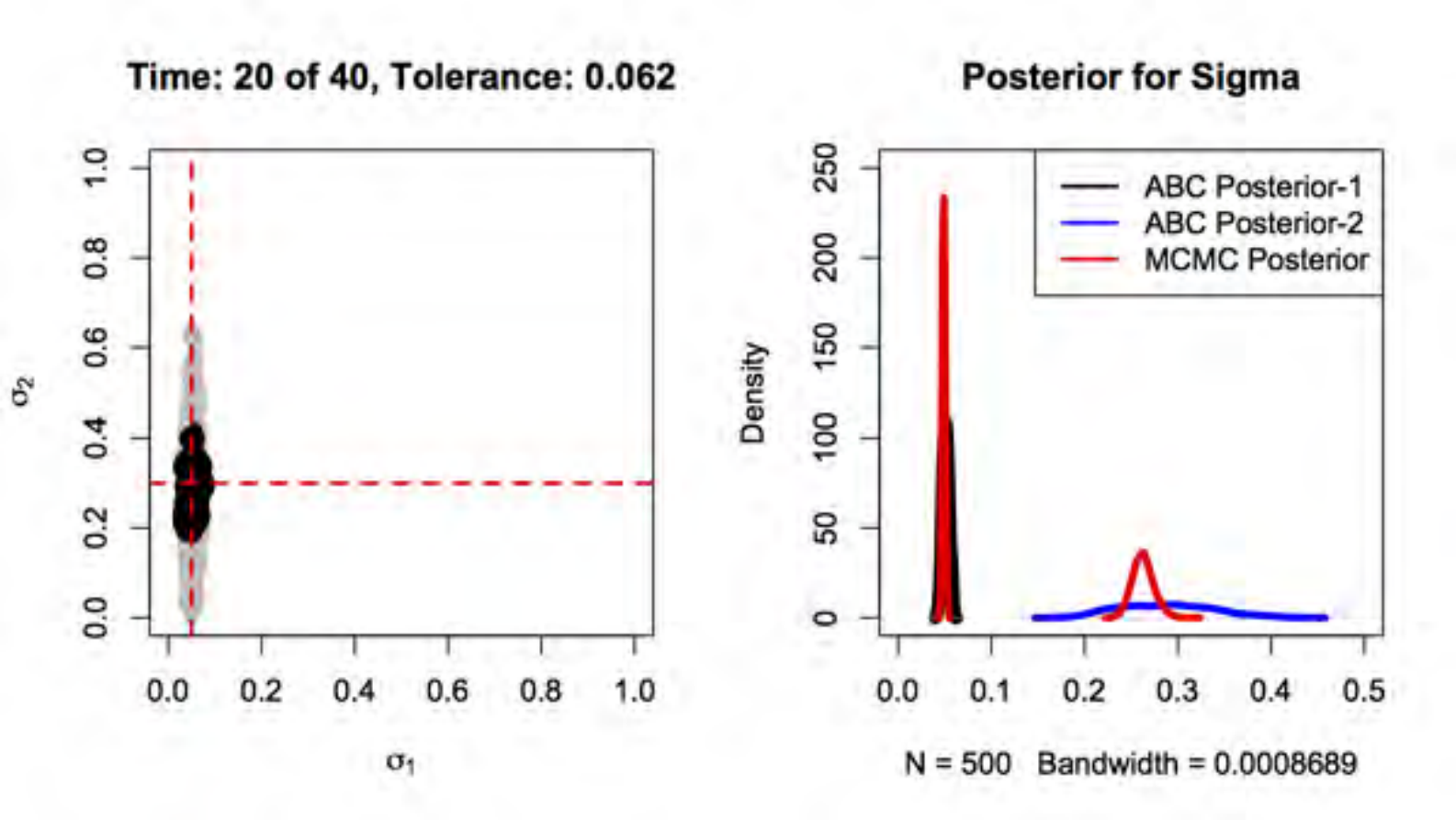
KS -distance with N = 500 particles



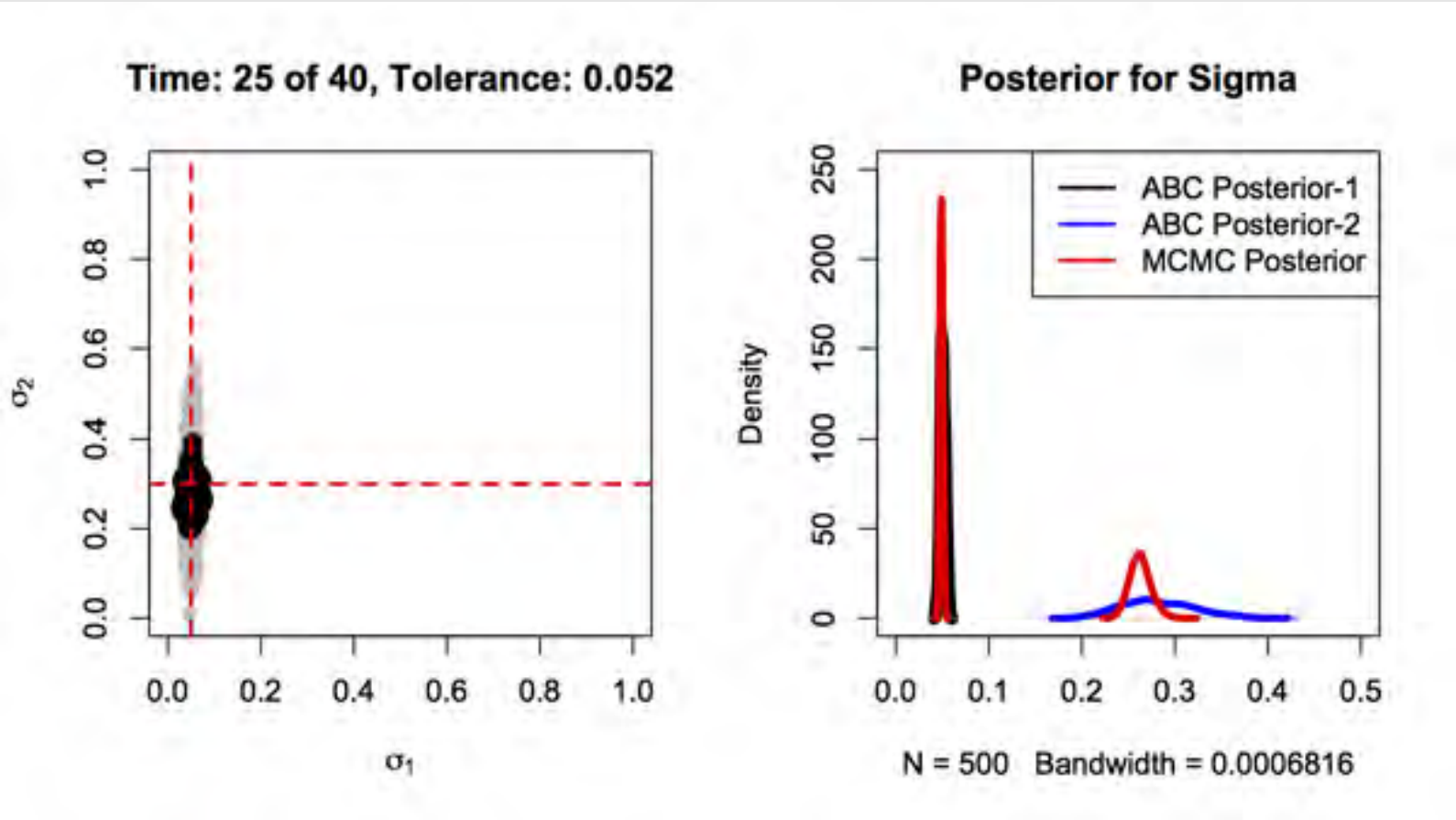
KS -distance with N = 500 particles



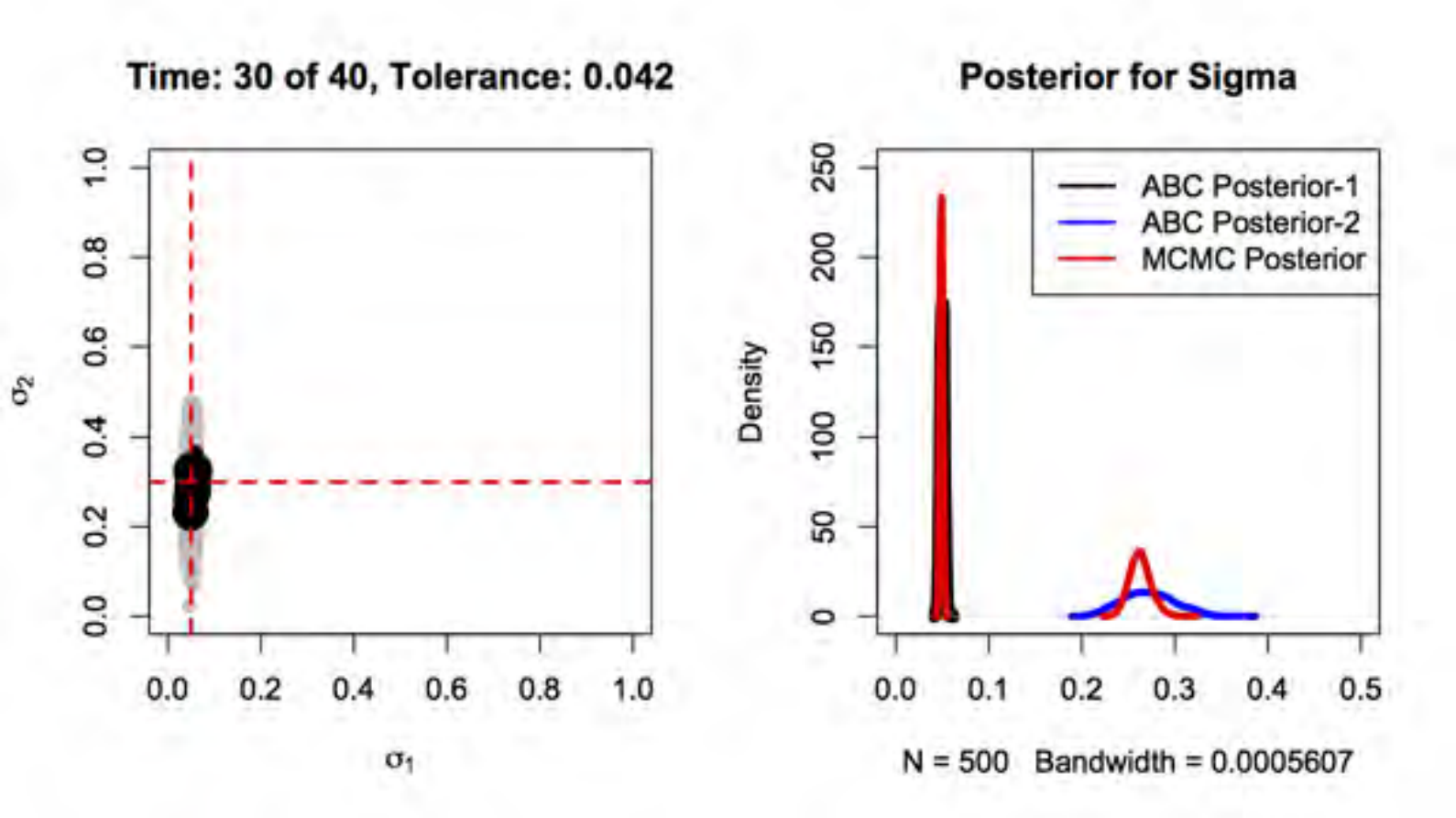
KS -distance with N = 500 particles



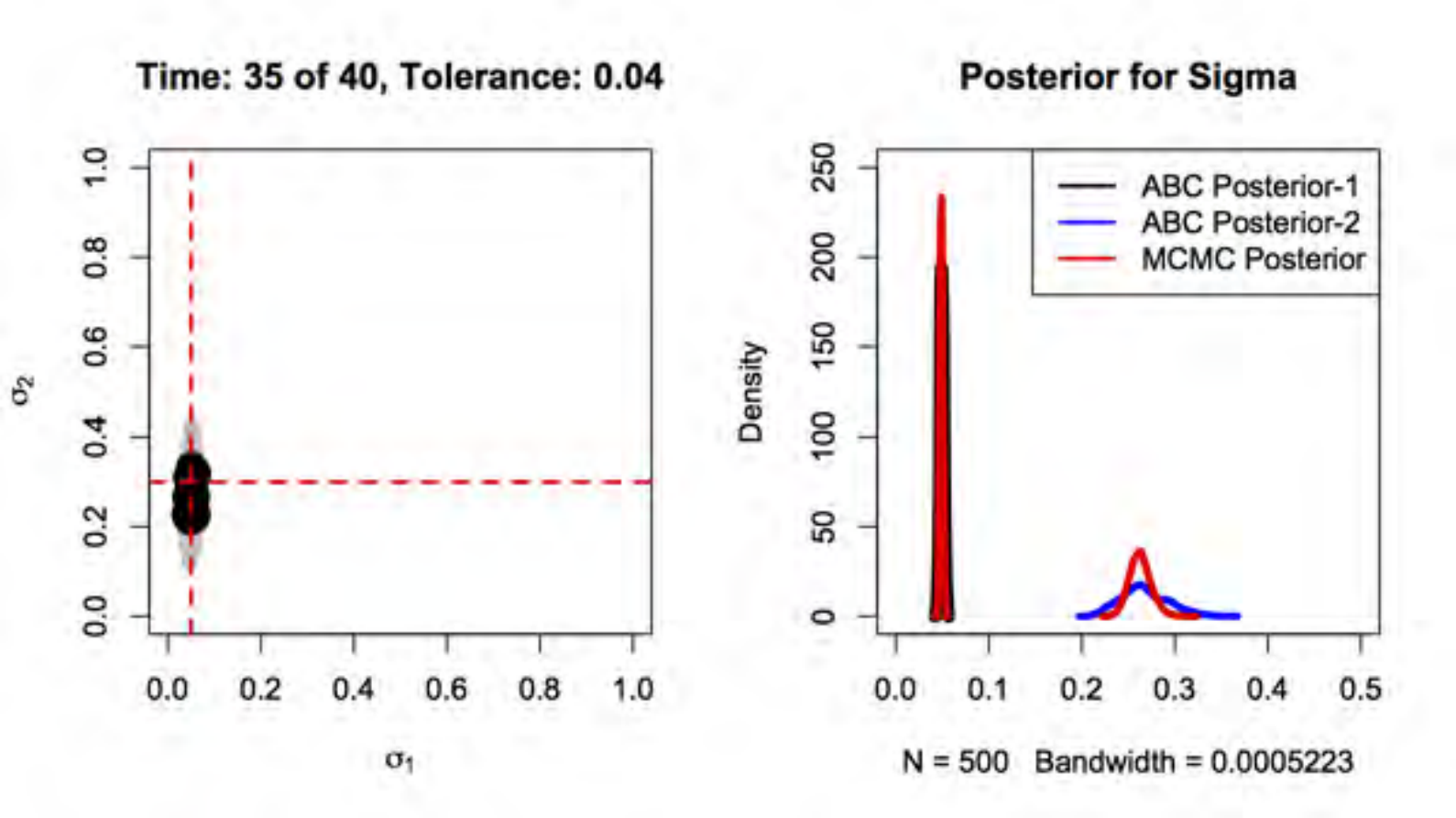
KS -distance with N = 500 particles



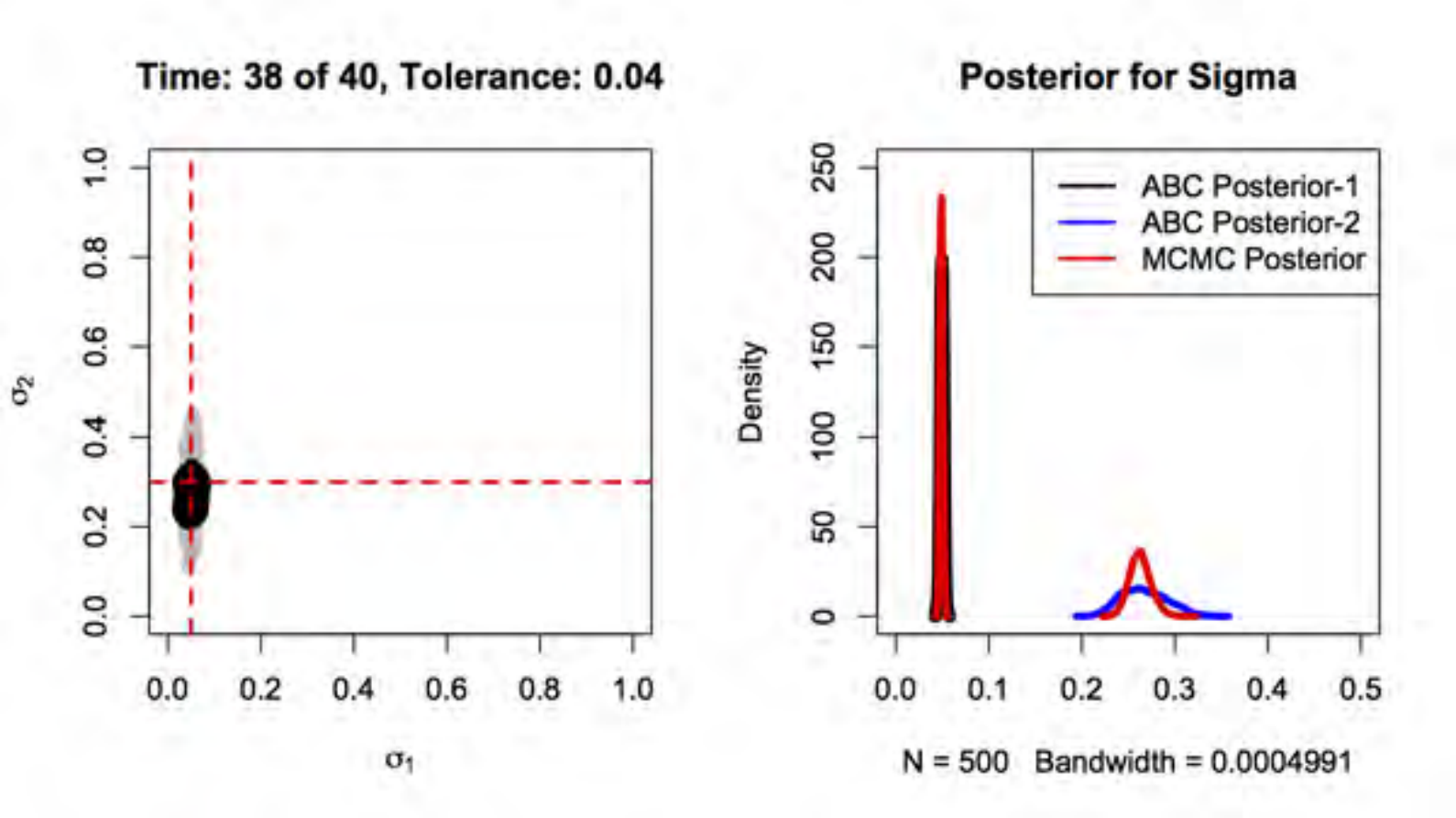
KS -distance with N = 500 particles



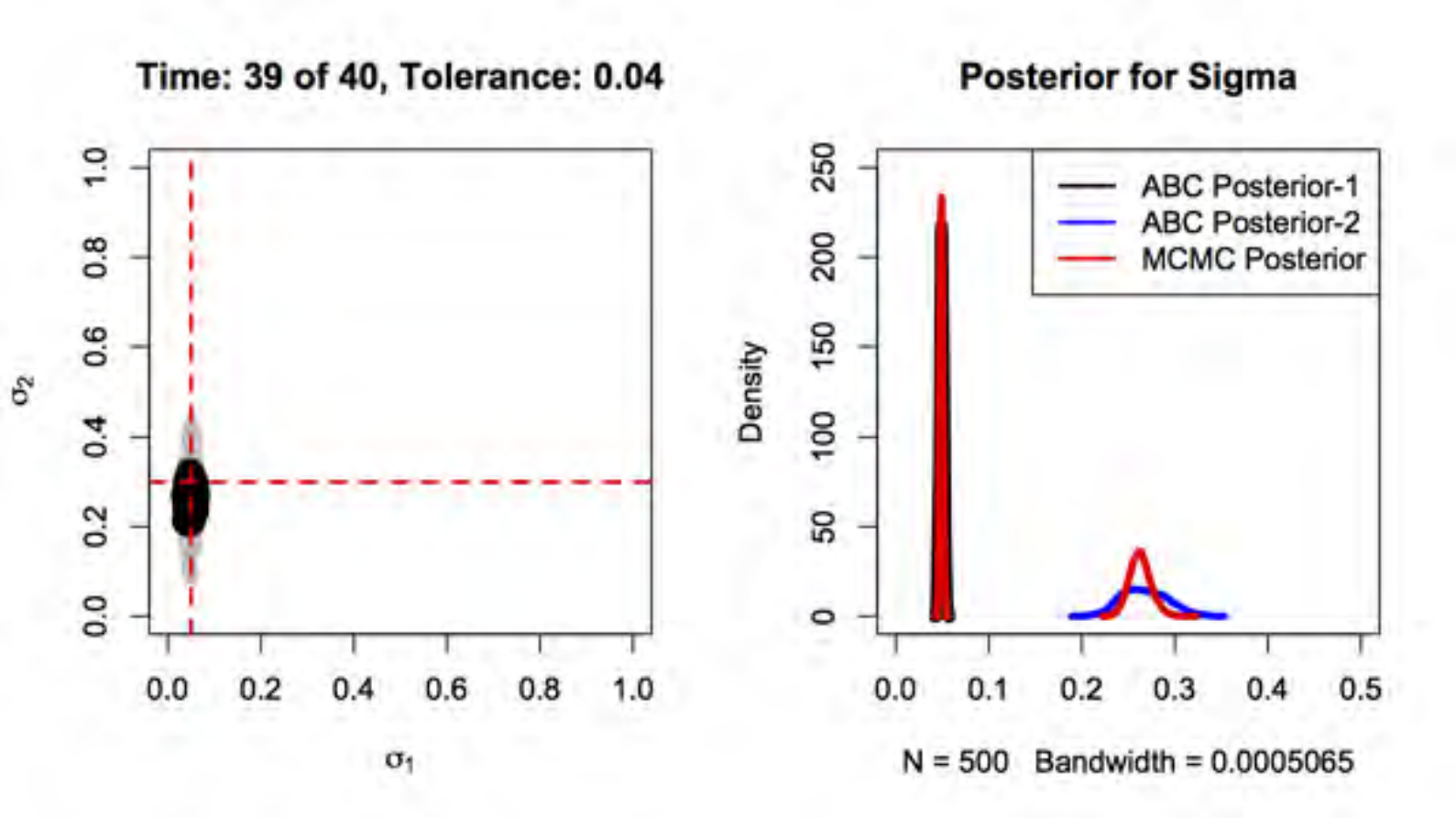
KS -distance with N = 500 particles



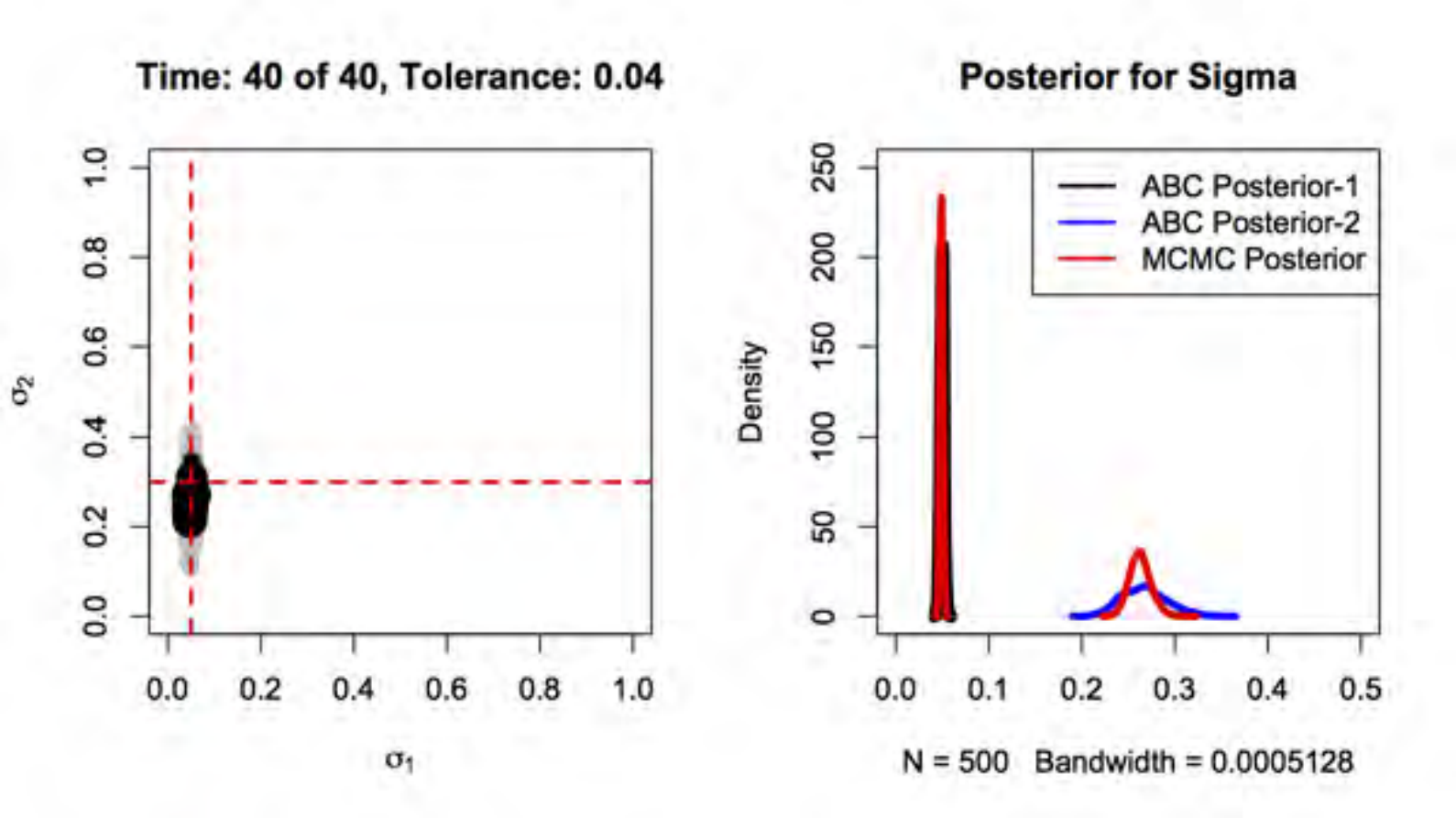
KS -distance with N = 500 particles



KS -distance with N = 500 particles

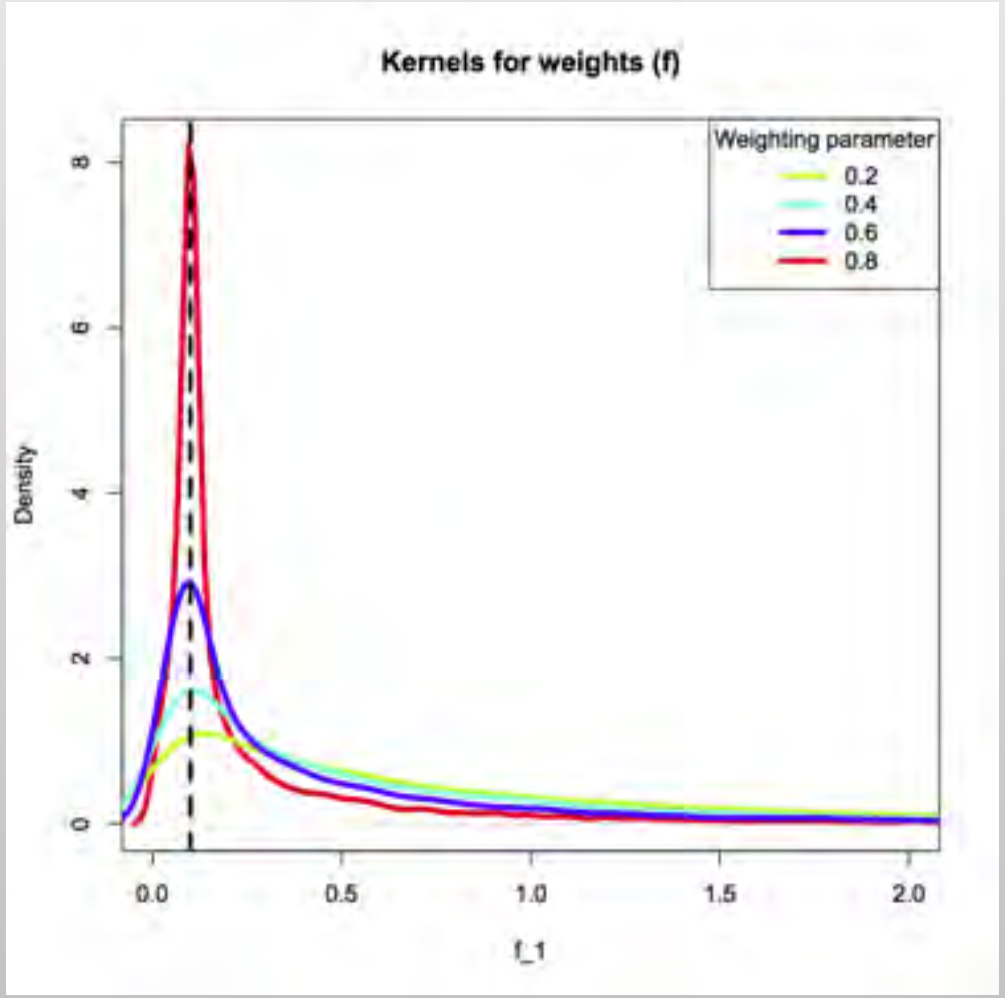


KS -distance with N = 500 particles



Transition kernels.

Moving particles: $\left\{ f_1^{(J)}, \ldots, f_{Nm}^{(J)} \right\}_{J=1}^{N}$



Moving particles: $\left\{ f_1^{(J)}, \ldots, f_{Nm}^{(J)} \right\}_{J=1}^{N}$

$$f_{old} = (f_1, f_2, \ldots, f_{Nm}) \text{ (original mixture weights)}$$

$$f_{old} \sim Dir(\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{Nm}))$$

$$\xi_i \sim Gamma(\alpha_i, 1)$$

$$\implies \xi_+ = \xi_1 + \cdots + \xi_{Nm} \sim Gamma(\alpha_+ = \sum \alpha_i, 1)$$

$$f_{old} = \left(\frac{\xi_1}{\xi_+}, \ldots, \frac{\xi_{Nm}}{\xi_+} \right) \sim Dir(\alpha)$$

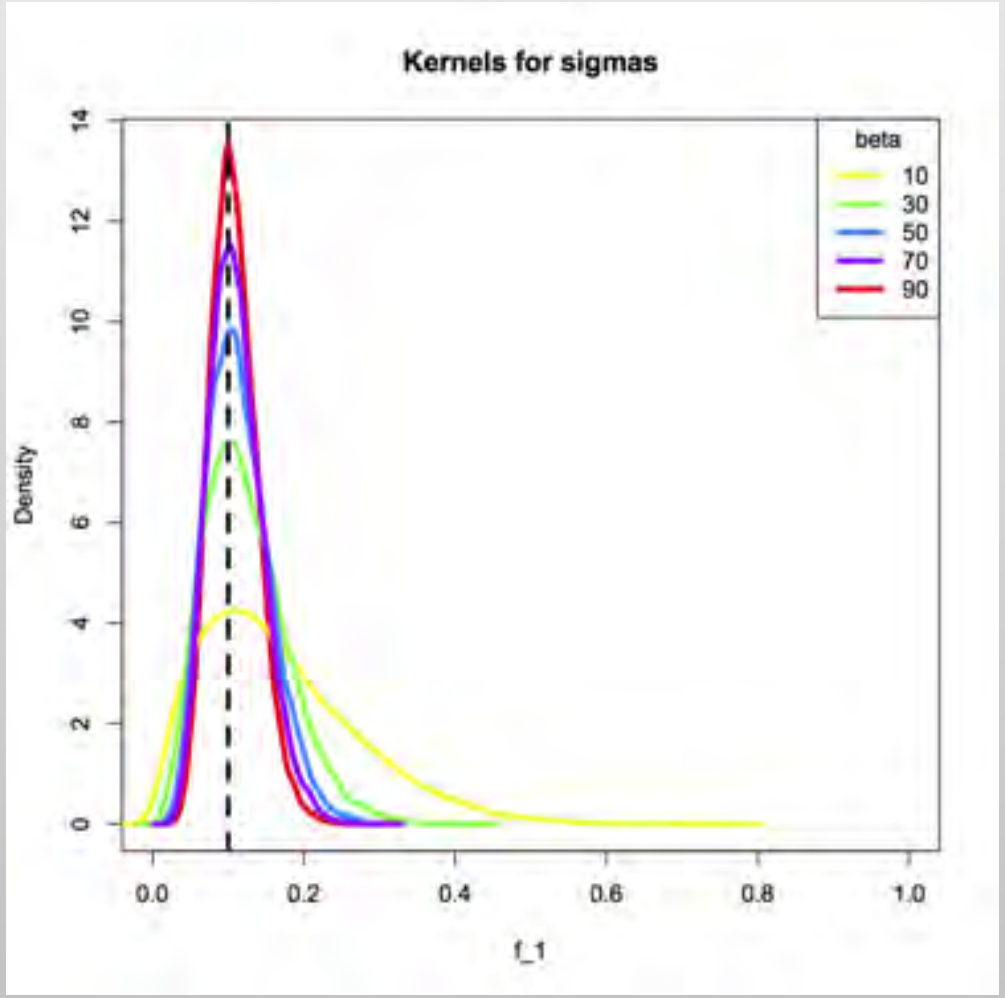
Moving particles: $\left\{ f_1^{(J)}, \ldots, f_{Nm}^{(J)} \right\}_{J=1}^{N}$

Given weighting parameter p

$$
\begin{aligned}
B_i &\sim \text{Beta}(p\alpha_i, (1-p)\alpha_i) \\
\zeta_i &\sim \text{Gamma}((1-p)\alpha_i, 1) \\
\xi_i^* &= \xi_i B_i + \zeta_i, i = 1, \ldots, N_m \\
&\implies \xi_i^* \sim \text{Gamma}(\alpha_i, 1) \\
&\implies \left(\frac{\xi_1^*}{\xi_+^*}, \ldots, \frac{\xi_{Nm}^*}{\xi_+^*} \right) \sim Dir(\alpha)
\end{aligned}
$$

where $\xi_+^* = \sum_{i=1}^{Nm} \xi_i^*$.

Moving particles: $\left\{\sigma_1^{(J)}, \ldots, \sigma_{Nm}^{(J)}\right\}_{J=1}^{N}$



Recall: In a nutshell

"The basic idea behind ABC is that using a representative (enough) summary statistic η coupled with a small (enough) tolerance ϵ should produce a good (enough) approximation to the posterior..."

Marin et al. (2012)

Concluding remarks

- We considered examples related to Type Ia Supernova, Stellar IMF, Exoplanet Eccentricity
- Three main ABC decisions: summary statistic, distance function, tolerance
- Sequential ABC: need to specify a transition kernel and determine importance weights
- Approximate Bayesian Computation could be a useful tool in astronomy, **but** it should be used with care.

Concluding remarks

- We considered examples related to Type Ia Supernova, Stellar IMF, Exoplanet Eccentricity
- Three main ABC decisions: summary statistic, distance function, tolerance
- Sequential ABC: need to specify a transition kernel and determine importance weights
- Approximate Bayesian Computation could be a useful tool in astronomy, **but** it should be used with care.

THANK YOU!!!

Additional useful resources

- `http://approximatebayesiancomputational.wordpress.com/`
- Csilléry et al. (2010): Approximate Bayesian Computation (ABC) in practice
- Csillery et al. (2012): abc: an R package for approximate Bayesian computation (ABC)
- Jabot et al. (2013): EasyABC: performing efficient approximate Bayesian computation sampling schemes (R package)

Bibliography

Beaumont, M. A., Cornuet, J.-M., Marin, J.-M., and Robert, C. P. (2009), "Adaptive approximate Bayesian computation," *Biometrika*, 96, 983 – 990.

Csilléry, K., Blum, M. G., Gaggiotti, O. E., and François, O. (2010), "Approximate Bayesian Computation (ABC) in practice," *Trends in ecology & evolution*, 25, 410 – 418.

Csillery, K., Francois, O., and Blum, M. G. B. (2012), "abc: an R package for approximate Bayesian computation (ABC)," *Methods in Ecology and Evolution*.

Jabot, F., Faure, T., and Dumoullin, N. (2013), *EasyABC: performing efficient approximate Bayesian computation*.

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012), "Approximate Bayesian computational methods," *Statistics and Computing*, 22, 1167 – 1180.