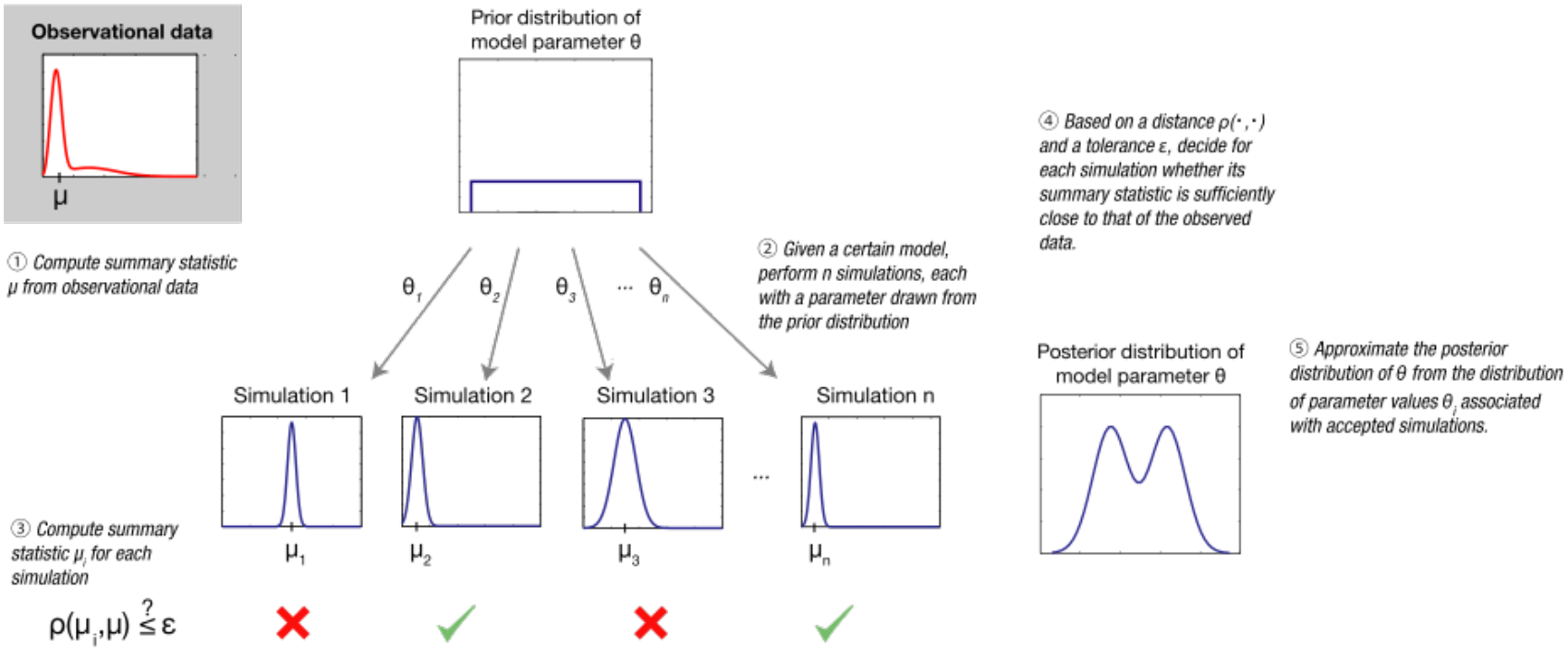
EasyABC : ABC, easy as 123

Nicolas Dumoulin, T. Faure, F. Jabot - Irstea, Lisc

23 Mai 2014

L'ABC en résumé

Approximate Bayesian Computation

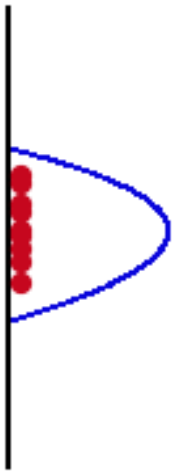


Source : wikipedia

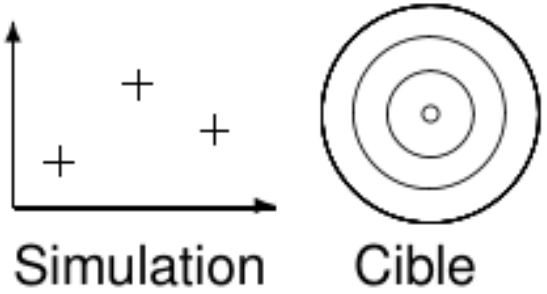
Approximate Bayesian Computation

- Tirer $\theta^* \sim \pi(\theta)$
- Simuler $D' \sim f(\theta^*)$
- Si $\rho(D', D) \leq \epsilon$, accepter θ^*, sinon le rejeter
- Répéter jusqu'à ce qu'un échantillon de la taille désirée soit obtenu (Pritchard et al., 1999)

Distribution *a priori*
$\pi(\theta)$

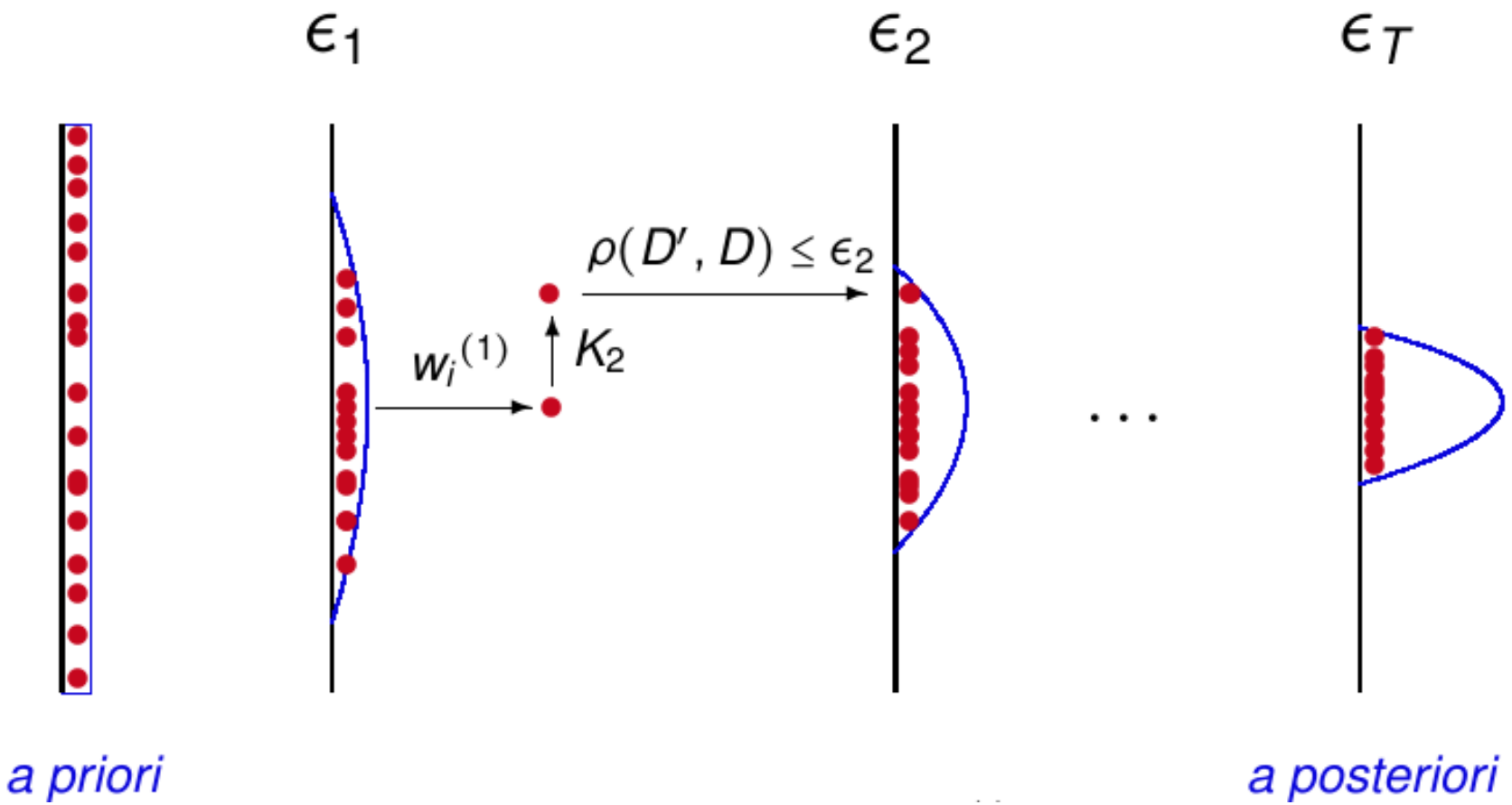


Distribution *a posteriori*
$\mathbb{P}(\theta|\rho(D', D) \leq \epsilon)$



Population Monte Carlo

(Beaumont et al., 2009)



De l'émulation, de la diversité

- Beaucoup d'algorithmes proposés
- Chacun apporte ses améliorations
- Dans ce contexte, pas évident de s'y retrouver et de les tester

De l'émulation, de la diversité

- Beaucoup d'algorithmes proposés
- Chacun apporte ses améliorations
- Dans ce contexte, pas évident de s'y retrouver et de les tester

EasyABC

Une idée, une envie

Une idée, une envie

Une idée, une envie

Une idée, une envie

Fonctionnalités

Cf. la documentation et la vignette

- 7 méthodes implémentées
- Définition de contraintes sur les domaines de définition ("X1<X2+X3")
- Utilisation de fonctions d'échantillonage arbitraire
- Option multicœurs
- Utilisation modèle binaire (exécutable) ou modèle java

Comment ça marche ?

Installation

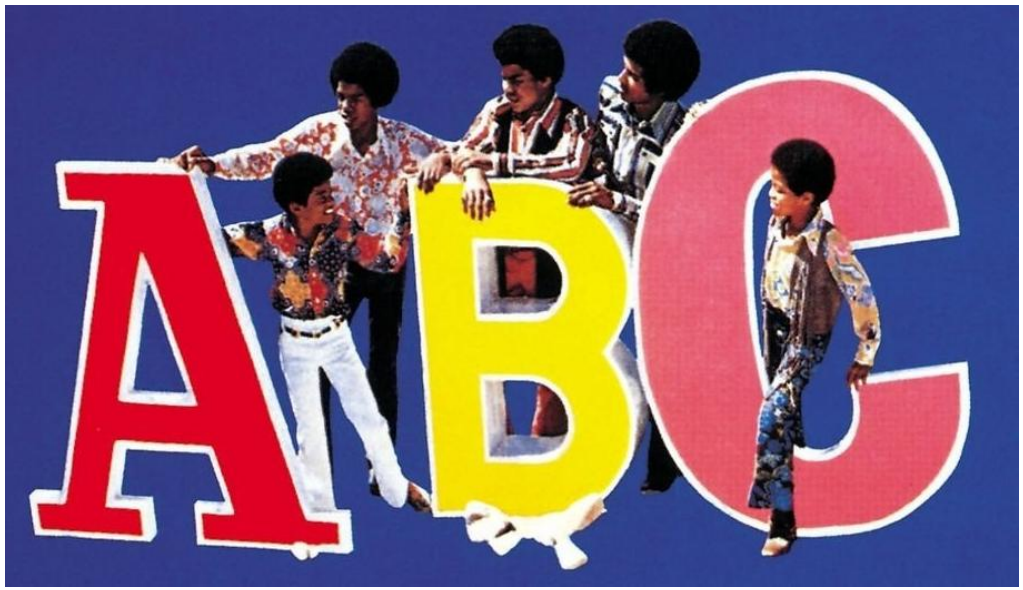
Easy as 123

```
install.packages("EasyABC")
library("EasyABC")
```

Installation

Easy as 123

```
install.packages("EasyABC")
library("EasyABC")
```



Premiers pas : le script

```
# Prenons un modèle comme exemple
> toy_model <- function(x){
+  c(x[1]+x[2] + rnorm(1,0,0.1),
+    x[1]*x[2] + rnorm(1,0,0.1))
+ }
# Définissons notre prior
> toy_prior = list(c("unif",0,1),c("normal",1,2))
# la cible
> sum_stat_obs = c(1.5,0.5)
# top départ !
> ABC <- ABC_rejection(model=toy_model, prior=toy_prior,
+   nb_simul=10, summary_stat_target=sum_stat_obs,
+   tol=0.2)
```

Premiers pas : le résultat

```
> ABC
$param
           [,1]       [,2]
param 0.7341818 0.01361074
param 0.6534961 0.80211864
$stats
          [,1]      [,2]
[1,] 0.8332396 0.1414957
[2,] 1.4453496 0.5851046
$weights
[1] 0.5 0.5
$stats_normalization
[1] 1.820294 1.015811
$nsim
[1] 10
$nrec
[1] 2
```

ABC séquentiel

- (Beaumont et al., 2009)
  - l'utilisateur doit fournir les seuils de tolérance successifs (tolerance_tab)

```
> ABC_Beaumont <- ABC_sequential(method="Beaumont",
+   model=toy_model, prior=toy_prior,
+   nb_simul=10, summary_stat_target=sum_stat_obs,
+   tolerance_tab=c(1.25,0.75))
```

ABC séquentiel

- (Lenormand et al., 2012)
  - l'utilisateur doit fournir le nombre de simulations initial (nb_simul)
  - à chaque itération, le nombre de nouvelles simulations est un ratio de nb_simul (alpha = 0.5 par défaut)
  - seuil d'arrêt p_acc_min par défaut à 0.05
  - obligation d'utiliser un LHS uniforme

```
> toy_prior2=list(c("unif",0,1),c("unif",0.5,1.5))
> ABC_Lenormand<-ABC_sequential(method="Lenormand",
+  model=toy_model, prior=toy_prior2,
+  nb_simul=10, summary_stat_target=sum_stat_obs,
+  p_acc_min=0.4)
```

Multicœurs

- La fonction du modèle doit accepter comme premier argument l'index du flux de nombre pseudo-aléatoires à utiliser

```
> toy_model_parallel <- function(x) {
+  set.seed(x[1])
+  c(x[2]+x[3] + rnorm(1,0,0.1),
+    x[2]*x[3] + rnorm(1,0,0.1))
+ }
```

- Chaque fonction d'EasyABC accepte une option n_cluster (par défaut 1)

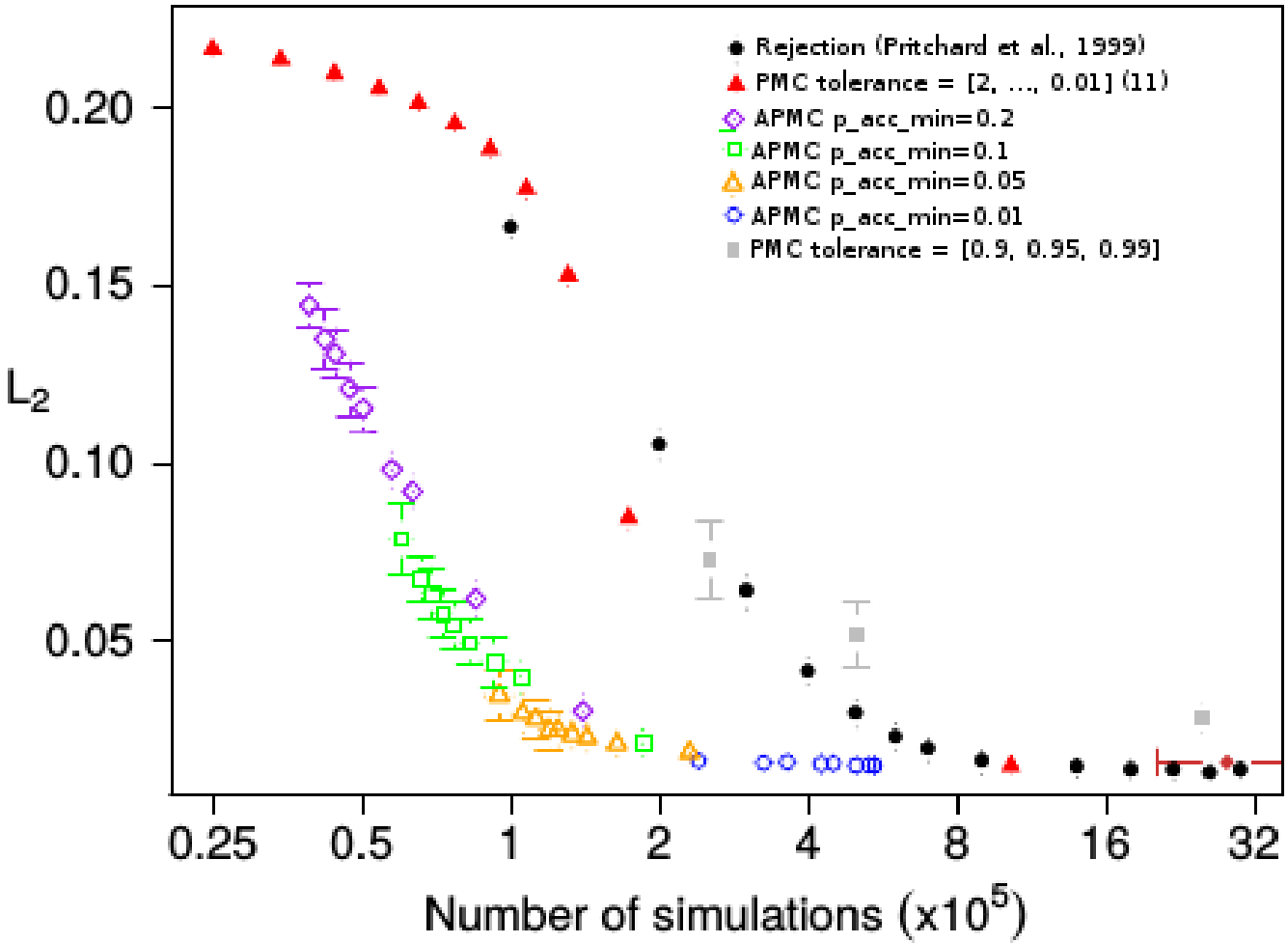
```
> ABC <- ABC_rejection(model=toy_model_parallel,
+  prior=toy_prior, nb_simul=n,  n_cluster=2,
+  summary_stat_target=sum_stat_obs, tol=0.2,
+  use_seed=TRUE)
```

Conclusion

Algorithmes embarqués

- ABC rejection (Pritchard et al., 1999)
- ABC séquentiel
  - (Beaumont et al., 2009)
  - (Drovandi and Pettitt, 2011)
  - (Del Moral et al., 2012)
  - (Lenormand et al., 2012)
- Markov Chain Monte Carlo
  - (Marjoram et al., 2003)
  - (Wegmann et al., 2009)

Une piste pour choisir



(Lenormand et al., 2013)

Le projet

- Démarré en juin 2012
- Déposé sur CRAN en novembre 2012
- 6 mises à jour depuis
  - corrections d'anomalies
  - nouvelles fonctionnalités
- 1000+ visites sur notre site web (pas de stats sur CRAN) dont 90% hors France

Perspectives

- version scala, plugin OpenMole
- Intégration dans mtk
- Collaborations pour intégrer de nouveaux algos

Merci

install.packages("EasyABC")

http ://easyabc.r-forge.r-project.org/