

Présentation du contexte scientifique

Context and Approach

Method

Application

Conclusion

A modelling framework for assessing lethal and sublethal effects of GM maize pollen on non target Lepidoptera

**Emily Walker, Antoine Messéan, Samuel Soubeyrand, Melen Leclerc,
Jean-Francois Rey**

INRA (France)
BioSP - EcoInnov

April 2016 - Future for Butterflies
Genetically modified crops and Lepidoptera session



Context and Approach	Method	Application	Conclusion
Context			

To provide tools for risk assessors and managers: to **assess risk** of **GM pollen** dispersal (from Bt maize) on **non target organisms** (NTO, butterflies) at the **landscape scale**.

- AMIGA project (FP7, European project): Assessing and Monitoring the Impacts of Genetically modified (GM) plants on Agro-ecosystems
- Towards a more quantitative risk assessment of Bt maize cultivation

Objectives:

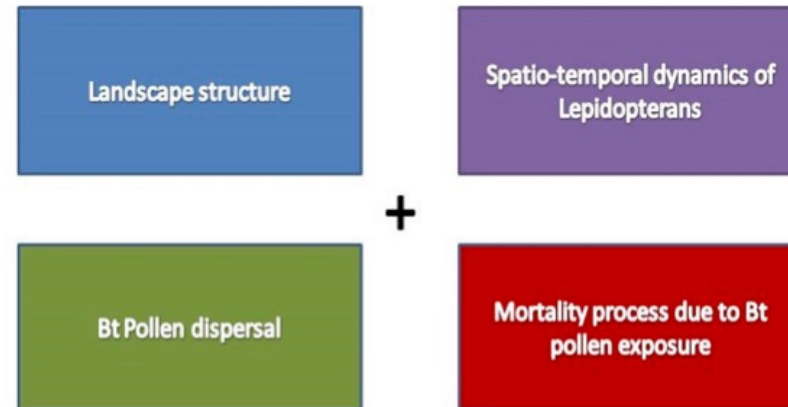
- Design generic **spatially explicit models** for assessing the impacts of GM crops on non-target organisms within agricultural landscapes
- Application to impacts of Bt maize on non-target Lepidoptera larvae (*Inachis io*)



Approach: model structure

Develop a **generic landscape model for quantifying and analysing the impacts** of Bt maize cultivation on non-target organisms (butterflies).

To assess the impacts of the introduction of Bt maize cultivation depending on receiving environments and management scenarios.



Approach: modelling choices

Spatial scale = regional (landscape)

Temporal scale = day

One larval stage = 1st larval stage

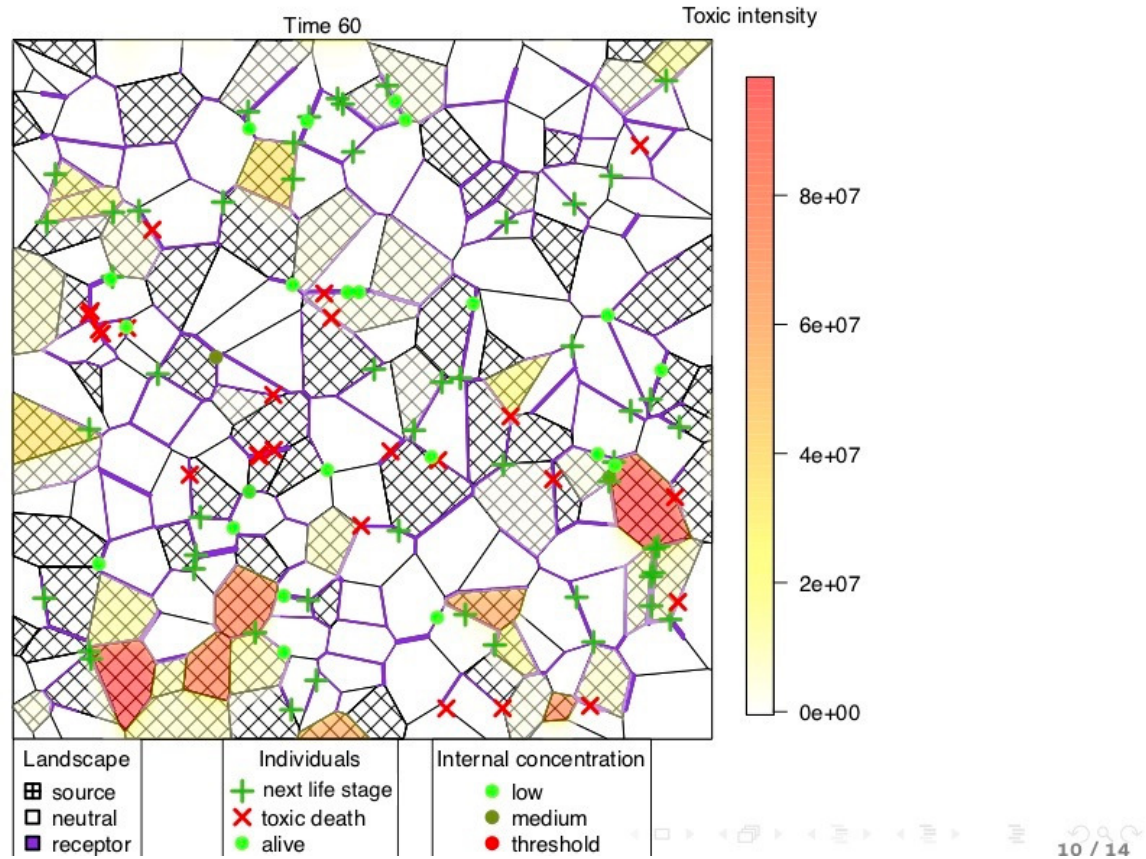
Steps forward compared to existing models (e.g. Perry et al. 2012; Holst et al., 2013):

- both spatially and temporally explicit
- different dispersal kernels can be considered
- temporal effects taken into account for both fields (flowering) and butterflies \Rightarrow phenology
- ecotoxicological model to assess influence of this process on the results

\Rightarrow R package named **briskaR (biological risk assessment with R)**

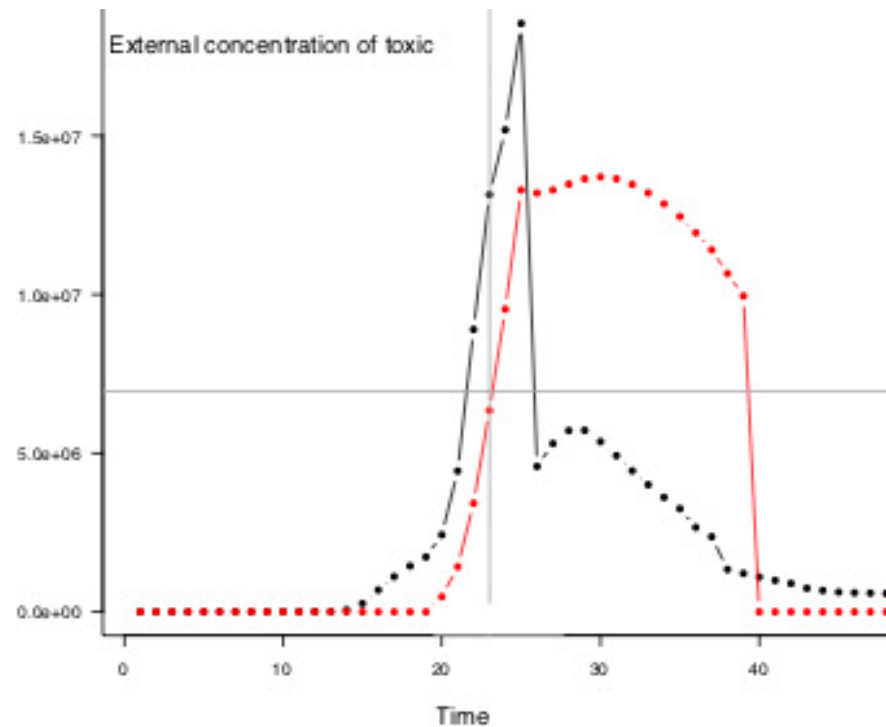
- Calculating the individual risk, conditional on the temporal evolution of the external concentration of toxic

End of simulation



Question : Comment modéliser la survenue de la mortalité d'un individu au cours d'une séquence temporelle d'exposition à un toxique ?

Exemples d'expositions au cours du temps subies par 2 individus localisées en 2 endroits différents :



Quand la mort survient, si elle survient, au cours de telles séquences d'exposition ?

Difficulté : pas de données pour des courbes telles que celles présentées ci-dessus mais des données expérimentales pour des processus d'exposition simplifiés

Felke et al. (2010) : larves de paons du jour (*Inachis io*) exposées à du pollen de maïs Bt-176



Données

```
In [ ]: ## (nombre de grains de pollen / cm2) x (2 cm2 ; surface de la portion de feuille) x (40 répétitions)
tox.dose=c(0,23,51,104,205,288)
tox=tox.dose*2*40

## proportion de mortalité
mortality=c(42, 70, 45, 101, 269, 340)/400
names(mortality)=paste("dose",tox.dose,sep="_")

## display data
plot(tox.dose,mortality,ylim=c(0,1),
      xlab="Toxic dose (# toxic pollen grains / cm2)",
      ylab="Death proportion")
```


Modèle de mortalité pour un groupe de n individus

X : dose toxique administrée au groupe de n individus

p_{ij} : proportion de la dose de pollen toxique ingérée par la larve i le jour $j \in \{1, 2\}$ (on partage le *gâteau* en $2n$ parts éventuellement inégales)

$$(p_{11}, \dots, p_{n1}, p_{12}, \dots, p_{n2}) \sim \text{Dirichlet} \left(\frac{1}{\beta}, \dots, \frac{1}{\beta} \right)$$

$$E(p_{ij}) = \frac{1}{2n} \quad V(p_{ij}) = \frac{n-1}{n^2(1+n/\beta)} \quad \text{Cov}(p_{ij}, p_{i'j'}) = \frac{-1}{n^2(1+n/\beta)} \quad (i, j) \neq (i', j')$$

β : paramètre gouvernant la variance des proportions

ρ_{ij} : concentration interne de "toxiques" dans la larve i le jour $j \in \{1, 2, 3, \dots\}$ (les 2 premiers jours, la larve ingère une dose toxique; à partir du jour 2, la larve élimine une proportion k_{out} de la concentration interne du jour passé)

$$\begin{aligned} \rho_{i1} &= Xp_{i1} \\ \rho_{i2} &= Xp_{i1}(1 - k_{\text{out}}) + Xp_{i2} \\ \rho_{ij} &= Xp_{i1}(1 - k_{\text{out}})^{j-1} + Xp_{i2}(1 - k_{\text{out}})^{j-2} \quad j \geq 2 \end{aligned}$$

Y_{ij} : variable binaire indiquant si la larve i est vivante ($Y_{ij} = 1$) ou morte ($Y_{ij} = 0$) le jour j .

$$\begin{aligned} Y_{i0} &= 1 \\ Y_{ij} &\underset{\text{indép.}}{\sim} \text{Bernoulli} \left(\alpha_1 \exp \left(- (\alpha_2 \rho_{ij})^{\alpha_3} \right) Y_{i(j-1)} \right) \quad j \geq 2 \end{aligned}$$

$\alpha_1 \in [0, 1]$: probabilité de mortalité journalière "naturelle"

$\alpha_2, \alpha_3 > 0$: paramètres gouvernant l'effet de la concentration toxique interne sur la mortalité journalière

Estimation

Paramètres : $(\beta, k_{\text{out}}, \alpha_1, \alpha_2, \alpha_3)$

Données : taux de mortalité global (i.e. sur 400 individus) au 7ème jour pour chaque dose

Estimation bayésienne classique : nombreuses variables latentes dépendantes ($400 \times 2 \times 6$ p_{ij} et $400 \times 6 \times 6$ Y_{ij})

Recours à l'ABC, en utilisant les taux de mortalité globaux au 7ème jour et des simulations du modèle de mortalité

Chargement des packages R utilisés

```
In [ ]: library(MCMCpack) ## load this package for drawing data from the Dirichlet distribution
library(abc) ## load this package for running functions related to ABC
library(psych) ## load this package for generating plots with functions pairs.panels() and violinBy()
```

Exploration de la distribution Dirichlet utilisée pour partager le "gâteau"

```
In [ ]: par(mfrow=c(2,2))
for(b in c(0.01,0.1,1,4)){
  hist(rdirichlet(n=40,alpha=rep(1/b,20)),xlab="Proportion",xlim=c(0,1),main=paste("beta =",b))
  abline(v=1/20,col=2)
}
```

Exploration de la fonction gouvernant la probabilité de mortalité journalière relative (i.e. relative à la mortalité "naturelle")

```
In [ ]: a2=0.1## try both values 0.1 and 0.02
tox.seq=0:100
par(mfrow=c(2,2))
for(a3 in c(0.5,1,2,10)){
  plot(tox.seq,exp(-(a2*tox.seq)^a3),type="l",ylim=c(0,1),
       xlab="Individual internal toxic level",
       ylab="Relative daily survival probability",
       main=paste("alpha2 =",a2,"; alpha3 =",a3))
  points(tox/400,exp(-(a2*tox/400)^a3),pch=19,col=2)
}
```

Simulateur du modèle et exemples de sorties

Arguments du simulateur :

n: nombre total d'individus

nj: nombre de groupes

X: vecteur de doses

kout: proportion d'élimination journalière de la concentration toxique interne

alpha1,alpha2,alpha3: paramètres de mortalité,

beta: paramètre gouvernant la variance des proportions dans le partage de la dose toxique

ngroup: nombre de groupes dans lesquels les individus sont répartis

```

In [ ]: ## simulator
mortality.pop=function(n,nj,X,kout,alpha1,alpha2,alpha3,beta, ngroup=1){
  death.prop=rep(0,length(X))
  size=n/ngroup
  for(i in 1:length(X)){
    prop.feed=rdirichlet(n=ngroup,alpha=rep(1/beta,size*2))
    Q=cbind(as.numeric(prop.feed[,1:size]),as.numeric(prop.feed[,size+1:size]))
    rho=cbind(X[i]*Q[,1])%*%(1-kout)^((1:nj)-1) +
      cbind(X[i]*Q[,2])%*%c(0,(1-kout)^((2:nj)-2))
    event=ceiling(alpha1*exp(-(alpha2*rho)^alpha3)-matrix(runif(nj*n),n,nj))
    surv=apply(event,1,min)
    death.prop[i]=mean(1-surv)
  }
  return(death.prop)
}

## run and show 1 simulation ; play with alpha2 (0.1; 0.01; 0.001)
mortality.sim=mortality.pop(n=400,nj=7,tox,kout=0.5,alpha1=0.99,alpha2=0.01,
  alpha3=4,beta=0.25,ngroup=40)
cat("Doses toxiques:\n")
print(tox.dose)
cat("Taux de mortalité observés aux différentes doses toxiques:\n")
print(mortality.sim)
plot(tox.dose,mortality.sim,ylim=c(0,1),
  xlab="Toxic dose (# toxic pollen grains / cm2)",
  ylab="Death proportion")
points(tox.dose,mortality,col=2) ## death proportions observed for Inachis io

```

Rappel sur l'ABC acceptation-rejet

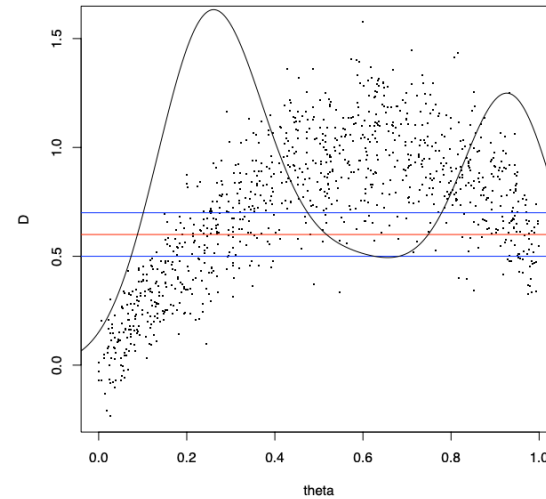
Principe de l'ABC

Notations :

- ▶ \mathcal{D} : jeu de données observé
- ▶ θ : paramètres du modèle
- ▶ π : loi a priori des paramètres
- ▶ \mathcal{M} : Modèle paramétré par θ sensé générer les données \mathcal{D}

Procédure :

- ▶ Simuler des jeux de paramètres θ_i , $i = 1, \dots, l$, sous la loi a priori π
- ▶ Pour chaque jeu de paramètres θ_i , simuler un jeu de données \mathcal{D}_i
- ▶ Retenir les \mathcal{D}_i "ressemblant" à \mathcal{D} pour construire la loi a posteriori de θ



Le rôle des statistiques résumées

Dernière étape de la procédure ABC :

- ▶ Retenir les \mathcal{D}_i “ressemblant” à \mathcal{D} pour construire la loi a posteriori de θ

Pb quand les \mathcal{D}_i et \mathcal{D} sont de grande dimension et dans des espaces continus

Dans l'ABC, on se sert de statistiques résumées pour comparer les jeux de données \mathcal{D}_i et \mathcal{D}

Notations :

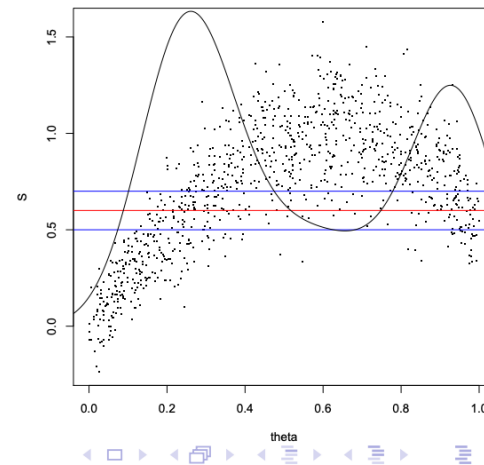
- ▶ $S = s(\mathcal{D})$: statistiques résumées observées
- ▶ $S_i = s(\mathcal{D}_i)$: statistiques résumées pour les paramètres simulés θ_i

ABC basique : acceptation/rejet avec seuil de tolérance

- ▶ S_0 : statistiques résumées observées
- ▶ $\theta_i \sim \pi(\cdot)$, $i = 1, \dots, l$, l très grand
- ▶ $S_i = s(\mathcal{M}_{\theta_i}(\omega_i))$ où \mathcal{M}_{θ_i} est un modèle stochastique implicite ($\mathcal{D}_i = \mathcal{M}_{\theta_i}(\omega_i)$)
- ▶ Ensemble des points acceptés pour estimer la loi a posteriori $p(\theta \mid d(S, S_0) \leq \epsilon)$:

$$\{\theta_i : i = 1, \dots, l \text{ et } d(S_i, S) \leq \epsilon\}$$

$$\begin{aligned} \hat{p}(\theta \mid d(S, S_0) \leq \epsilon) \\ = \frac{\sum_{i=1}^l \delta_{B(S_0, \epsilon)}(S_i) \delta_{\theta}(\theta_i)}{\sum_{i=1}^l \delta_{B(S_0, \epsilon)}(S_i)} \end{aligned}$$



Lois a priori des paramètres

A priori, les paramètres sont supposés indépendants

β : paramètre gouvernant la variance des proportions relatives au partage du "gâteau"

$$\beta \sim \text{Uniforme}(0, 4)$$

k_{out} : taux journalier d'élimination de la toxicité interne

$$k_{\text{out}} \sim \text{Uniforme}(0, 1)$$

α_1 : taux journalier de mortalité "naturelle"

$$\alpha_1 \sim \text{Beta}(50, 1)$$

α_2 : premier paramètre gouvernant l'effet de la concentration toxique interne sur la mortalité journalière

$$\alpha_2 \sim \text{Exponential}(1000)$$

α_3 : deuxième paramètre gouvernant l'effet de la concentration toxique interne sur la mortalité journalière

α_3 fixé :

$$\alpha_3 \sim \text{Dirac}(1)$$

α_3 variables :

$$\alpha_3 \sim \text{Uniforme}(0, 10)$$

```
In [ ]: ## plot priors
par(mfrow=c(2,3))
plot(seq(-5,10,0.01), dunif(seq(-5,10,0.01),0,4), type="l", xlab=expression(beta), ylab="Density")
plot(seq(-2,3,0.01), dunif(seq(-2,3,0.01),0,1), type="l", xlab=expression(k[out]), ylab="Density")
plot(seq(0,1.1,0.01), dbeta(seq(0,1.1,0.01), shape1=50, shape2=1), type="l", xlab=expression(alpha[1]), ylab="Density")
plot(seq(0,0.01,0.0001), dexp(seq(0,0.01,0.0001), rate=1000), type="l", xlab=expression(alpha[2]), ylab="Density")
plot(seq(-5,15,0.01), seq(-5,15,0.01)==1, type="l", xlab=expression(alpha[3]), ylab="Density",
      main="Fixed alpha3")
plot(seq(-5,15,0.01), dunif(seq(-5,15,0.01),0,10), type="l", xlab=expression(alpha[3]), ylab="Density",
      main="Variable alpha3")
```


Série de simulations avec $\alpha_3 = 1$

```
In [ ]: ## run simulations for ABC (try first with alpha3=1)
time0=proc.time()
nsimul=10^4
param=cbind(runif(nsimul,0,1), ## kout
            rbeta(nsimul,shape1=50,shape2=1), ## alpha1
            rexp(nsimul,rate=1000), ## alpha2
            rep(1,nsimul), ## alpha3=1
            runif(nsimul,0,4)) ## beta
colnames(param)=c("kout", "alpha1", "alpha2", "alpha3", "beta")
stats=t(apply(param,1,function(u) mortality.pop(n=400, nj=7, tox, u[1],u[2],u[3],u[4],u[5],ngroup=40)))
colnames(stats)=names(mortality)
proc.time()-time0
```

```
In [ ]: param[1:10,]
stats[1:10,]
```

```
In [ ]: ## show simulations
nsimul0=min(10^4,nsimul)
pairs.panels(param[1:nsimul0,],pch=".")
pairs.panels(rbind(stats[1:nsimul0,],mortality),pch=c(rep(19,nsimul0),21),bg="green")
```

Le vecteur des statistiques résumées (mortalités en fct des doses) est clairement "en dehors" de la distribution a priori des statistiques avec $\alpha_3 = 1$

→ Inutile d'appliquer l'ABC quand α_3 est fixé à 1

Série de simulations avec α_3 distribué a priori dans une loi uniforme

```
In [ ]: ## run simulations for ABC (try with alpha3 a priori uniformly distributed in [0,10])
time0=proc.time()
nsimul=10^5
param=cbind(runif(nsimul,0,1), ## kout
            rbeta(nsimul,shapel=50,shape2=1), ## alpha1
            rexp(nsimul,rate=1000), ## alpha2
            runif(nsimul,0,10), ## alpha3
            runif(nsimul,0,4)) ## beta
colnames(param)=c("kout", "alpha1", "alpha2", "alpha3", "beta")
stats=t(apply(param,1,function(u) mortality.pop(n=400, nj=7, tox, u[1],u[2],u[3],u[4],u[5],ngroup=40)))
colnames(stats)=names(mortality)
proc.time()-time0
```

```
In [ ]: ## save param and stats as rds files
# saveRDS("save/param.rds")
# saveRDS("save/stats.rds")

## load param and stats from existing files
# param=readRDS("save/param.rds")
# stats=readRDS("save/stats.rds")
# nsimul=nrow(stats)
```

```
In [ ]: param[1:10,]
stats[1:10,]
```

```
In [ ]: ## show simulations
nsimul0=min(10^4,nsimul)
pairs.panels(param[1:nsimul0,],pch=".")
pairs.panels(rbind(stats[1:nsimul0,],mortality),pch=c(rep(19,nsimul0),21),bg="green")
```

La fonction *abc()* du package *abc* de Csilléry et al.

abc: Tools for Approximate Bayesian Computation (ABC)

Lien sur le CRAN

<https://cran.r-project.org/web/packages/abc/index.html> (<https://cran.r-project.org/web/packages/abc/index.html>)

Vignette

<https://cran.r-project.org/web/packages/abc/vignettes/abcvignette.pdf> (<https://cran.r-project.org/web/packages/abc/vignettes/abcvignette.pdf>)

Description

This function performs multivariate parameter estimation based on summary statistics using an ABC algorithm. The algorithms implemented are rejection sampling, and local linear or non-linear (neural network) regression. A conditional heteroscedastic model is available for the latter two algorithms.

Usage

```
abc(target, param, sumstat, tol, method, hcorr = TRUE, transf = "none", logit.bounds, subset = NULL, kernel = "epanechnikov", numnet = 10, sizenet = 5, lambda = c(0.0001,0.001,0.01), trace = FALSE, maxit = 500, ...)
```

Main arguments

target: a vector of the observed summary statistics.

param: a vector, matrix or data frame of the simulated parameter values, i.e. the dependent variable(s) when method is "loclinear", "neuralnet" or "ridge".

sumstat: a vector, matrix or data frame of the simulated summary statistics, i.e. the independent variables when method is "loclinear", "neuralnet" or "ridge".

tol: tolerance, the required proportion of points accepted nearest the target values.

method: a character string indicating the type of ABC algorithm to be applied. Possible values are "rejection", "loclinear", "neuralnet" and "ridge". See also Details.

Application de l'ABC acceptance-rejet avec une tolérance arbitraire

```
In [ ]: ## apply ABC (only print output)
rej0=abc(target=mortality, param=param, sumstat=stats, tol=0.001, method="rejection")
rej0
summary(rej0)
## param[rej0$region,] ## table with all retained parameters
```

Sélection de la tolérance par validation croisée

```
In [ ]: ## but what tolerance value?
time0=proc.time()
tol.seq=c(0.0001,0.0005,0.001,0.005,0.01)
rej.cv=cv4abc(param=param, sumstat=stats, abc.out=rej0, nval=200, tols=tol.seq)
proc.time()-time0
summary(rej.cv)
```

```
In [ ]: ## save rej.cv as rds files
# saveRDS(tol.seq, "save/tol.seq.rds")
# saveRDS(rej.cv, "save/rej.cv.rds")

## load param and stats from existing files
# tol.seq=readRDS("save/tol.seq.rds")
# tol.seq
# rej.cv0=readRDS("save/rej.cv.rds")
# summary(rej.cv0)
```

```
In [ ]: ## display mean error assessed by cross-validation over each parameter
summ.rej.cv=summary(rej.cv)
par(mfrow=c(2,3))
apply(summ.rej.cv,2,function(u) plot(tol.seq,u,type="b",log="x"))
mean.error=rowMeans(summ.rej.cv)
plot(tol.seq,mean.error,type="b",log="x")
tol.opt=tol.seq[mean.error==min(mean.error)]

cat("Tolérance sélectionnée par validation croisée:\n")
as.numeric(tol.opt)

plot(rej.cv)
```

Application de l'ABC acceptance-rejet avec le seuil de tolérance sélectionné par validation croisée

```
In [ ]: ## apply ABC
rej=abc(target=mortality, param=param, sumstat=stats, tol=tol.opt, method="rejection")
rej
summary(rej)
par(mfrow=c(2,3))
hist(rej)
```

Echantillons a priori et a posteriori des paramètres

```
In [ ]: pairs.panels(rbind(param[1:nsimul0,],param[rej$region,]),pch=c(rep(19,nsimul0),rep(21,sum(rej$region))),
                    bg="green")

pairs.panels(param[rej$region,],pch=21,bg="green")
```

Echantillons a priori et a posteriori des statistiques

```
In [ ]: pairs.panels(rbind(stats[1:nsimul0,],mortality),pch=c(rep(19,nsimul0),21),
                    bg="green")

pairs.panels(rbind(stats[rej$region,],mortality),pch=c(rep(19,sum(rej$region)),21),
            bg="green")
```

Distributions a posteriori des prédictions des proportions de mortalité aux différentes doses toxiques

```
In [ ]: ## Marginal posterior distributions of mortality proportions
median.mortality=apply(stats,2,median)
median.mortality
cols.index=round((max(median.mortality)-median.mortality)/(max(median.mortality)-min(median.mortality))*100)
violinBy(stats,col=heat.colors(100)[pmax(1,cols.index)],
          xlab="Toxic dose (# toxic pollen grains / cm2)",
          ylab="Death proportion at the 7th day")
points(mortality,pch=19)
```

Pour aller plus loin

Dans le package abc

Nous avons utilisé l'ABC acceptance-rejet : `abc(target=mortality, param=param, sumstat=stats, tol=tol.opt, method="rejection")`

Autres méthodes implémentées dans le package abc :

`method="loclinear"` : a local linear regression method corrects for the imperfect match between $S(y)$ and $S(y_0)$

`method="ridge"` : method that performs local-linear ridge regression and deals with the collinearity issue

`method="neuralnet"` : non-linear regression to minimize the departure from non-linearity using the function `nnet`

Analyses complémentaires disponibles dans le package abc :

Model selection, Goodness-of-fit, Posterior predictive checks.

Autres packages R

abctools: Tuning ABC analyses <https://journal.r-project.org/archive/2015-2/nunes-prangle.pdf> (<https://journal.r-project.org/archive/2015-2/nunes-prangle.pdf>)

abcrf: ABC via random forests

EasyABC: Several algorithms for performing efficient ABC sampling schemes, including 4 sequential sampling schemes and 3 MCMC schemes

Logiciels

DIY-ABC: to perform parameter estimation and model selection for population genetics models

ABC-SysBio python package: parameter inference and model selection for dynamical systems

ABCtoolbox programs: various ABC algorithms including rejection sampling, MCMC without likelihood, a particle-based sampler, and ABC-GLM

PopABC: software package for inference of the pattern of demographic divergence, coalescent simulation, Bayesian model choice

In []: