useR ! 2019 – Thursday, July 11th 2019 – Toulouse

Using Rcpp* packages for easy and fast Gibbs sampling MCMC from within R

Ghislain VIEILLEDENT and Jeanne CLEMENT

Cirad, UMR AMAP, Montpellier, FRANCE

cirad

A M A P botAnique et Modélisation de l'Architecture des Plantes et des végétations

Outline

1. Short presentation of Rcpp* packages
   - Rcpp : extending R with C++
   - RcppGSL for fast random draws
   - RcppArmadillo for high-performance linear algebra
2. Rcpp for Gibbs sampling
   - Gibbs sampling and Bayesian statistics
   - A simple Gibbs sampler
   - Benchmark
3. The `jSDM` R package
   - Joint Species Distribution Models
   - Model specification
   - Comparison with `boral`/`JAGS`

cirad

botAnique et Modélisation de l'Architecture des Plantes et des végétations

Outline

1. Short presentation of Rcpp* packages
   - Rcpp : extending R with C++
   - RcppGSL for fast random draws
   - RcppArmadillo for high-performance linear algebra
2. Rcpp for Gibbs sampling
   - Gibbs sampling and Bayesian statistics
   - A simple Gibbs sampler
   - Benchmark
3. The `jSDM` R package
   - Joint Species Distribution Models
   - Model specification
   - Comparison with `boral`/`JAGS`

cirad

botAnique et Modélisation de l'Architecture des Plantes et des végétations

Rcpp R package

- **Rcpp** is an R package to extend R with C++ code
- Main advantage : C++ is fast, it accelerates R (see next sections)
- Written by **Dirk EDDELBUETTEL** and **Romain FRANCOIS**
- http://www.rcpp.org/

Simple Rcpp example

C++ code (in file `Code/addition.cpp`)

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
int addition(int a, int b) {
  return a + b;
}
```

R code

```
Rcpp::sourceCpp("Code/addition.cpp")
addition(2, 2)
```

```
## [1] 4
```

Rcpp advantages

Thanks to `Rcpp::sourceCpp()`

- Compile the C++ code
- Export the function to the R session
- Direct interchange of R objects (including S3, S4) between R and C++
- ... (many more, see `vignette("Rcpp-package")`)

In an R package

- `Rcpp.package.skeleton()` to generate a new Rcpp package (modifying `DESCRIPTION` and `NAMESPACE`)
- `Rcpp::compileAttributes()` scans the C++ files for `Rcpp::export` attributes and generates the code required to make the functions available in R.

GSL and RcppGSL

GNU Scientific Library

- Numerical library for C and C++ programmers
- Reliable random number generator algorithms
- Thoroughly tested and fast random number distributions
- Linear algebra (matrices and vectors)
- https://www.gnu.org/software/gsl/

RcppGSL

- Interface between R and GSL
- Using Rcpp to interface R and C
- http://dirk.eddelbuettel.com/code/rcpp.gsl.html

GSL random number distributions

- GSL v2.6 includes **38 random number distributions** (see GNU GSL)
- It's easy to implement additional random number distributions from the GSL base distributions (e.g. truncated normal distribution)
- For comparison, R API includes "only" 24 random number distributions (see Writing R Extensions)
- Random draws are faster with GSL than with R (eg. `gsl_ran_gamma()` vs. `R::rgamma()`)

RcppGSL example

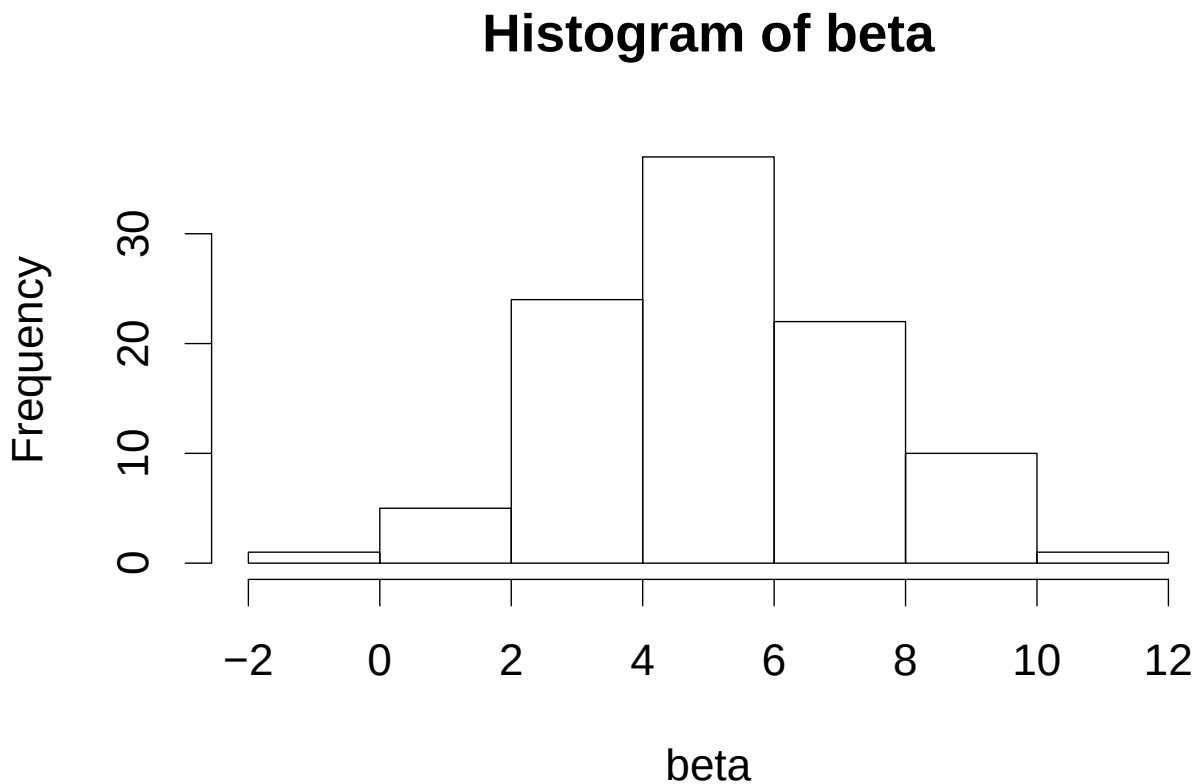
C++ code

```
#include <Rcpp.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>

using namespace Rcpp;

// [[Rcpp::depends(RcppGSL)]]

// [[Rcpp::export]]
Rcpp::NumericVector my_rnorm(int nsamp, double mu,
                             double sigma) {
    gsl_rng *s = gsl_rng_alloc(gsl_rng_mt19937); // Random seed
    Rcpp::NumericVector beta(nsamp);
    for (int i = 0; i < nsamp; i++) {
        beta[i] = mu + gsl_ran_gaussian(s, sigma); // Random draw
    }
    return beta;
}
```

RcppGSL example

R code

```
library(Rcpp)
library(RcppGSL)
beta <- my_rnorm(100, 5, 2)
par(cex=2)
hist(beta)
```



Armadillo and RcppArmadillo

Armadillo

- C++ library for linear algebra and scientific computing
- Provides high-level syntax and functionality : speed and ease of use
- Classes for vectors, matrices and cubes
- Matrix operations, matrix decomposition, linear model solver, etc.
- http://arma.sourceforge.net/

RcppArmadillo

- Interface between R and Armadillo
- Using Rcpp to interface R and C++
- http://dirk.eddelbuettel.com/code/rcpp.armadillo.html

RcppArmadillo example

C++ code

```
#include <RcppArmadillo.h>

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
Rcpp::List fastLm(const arma::mat& X, const arma::colvec& y) {
    int n = X.n_rows, k = X.n_cols;

    arma::colvec coef = arma::solve(X, y);    // fit model y ~ X
    arma::colvec res  = y - X*coef;           // residuals

    // std.errors of coefficients
    double s2 = std::inner_product(res.begin(),
                                   res.end(),
                                   res.begin(), 0.0)/(n - k);

    arma::colvec std_err = arma::sqrt(s2 *
      arma::diagvec(arma::pinv(arma::trans(X)*X)));

    return Rcpp::List::create(Rcpp::Named("coefficients") = coef,
                              Rcpp::Named("stderr")       = std_err,
                              Rcpp::Named("df.residual")  = n - k);
}
```

RcppArmadillo example

R code

```
library(Rcpp)
library(RcppArmadillo)
# Trees data-set
y <- log(trees$Volume)
X <- cbind(1, log(trees$Girth))
# fastLm
mod <- fastLm(X, y)
mod$coef
```

```
##           [,1]
## [1,] -2.353325
## [2,]  2.199970
```

Licenses

- Licenses : GNU General Public License, Apache License 2.0 for Armadillo
- Free software licenses : we can use, modify and redistribute these softwares

Outline

1. Short presentation of Rcpp* packages
   - Rcpp : extending R with C++
   - RcppGSL for fast random draws
   - RcppArmadillo for high-performance linear algebra
2. Rcpp for Gibbs sampling
   - Gibbs sampling and Bayesian statistics
   - A simple Gibbs sampler
   - Benchmark
3. The `jSDM` R package
   - Joint Species Distribution Models
   - Model specification
   - Comparison with `boral`/`JAGS`

cirad

botAnique et Modélisation de l'Architecture des Plantes et des végétations

Gibbs sampling

Gibbs sampling is commonly used for **statistical inference**.

$\Theta = (\theta_1, \ldots, \theta_n)$ from a joint distribution $p(\theta_1, \ldots, \theta_n)$.

1. Begin with some initials values $\Theta^{(i)}$.
2. Next sample $\Theta^{(i+1)} = (\theta_1^{(i+1)}, \ldots, \theta_n^{(i+1)})$?
   We sample iteratively the parameters.
   Update $\theta_j^{(i+1)}$ according to $p(\theta_j^{(i+1)} | \theta_0^{(i+1)}, \ldots, \theta_{j-1}^{(i+1)}, \theta_{j+1}^{(i)}, \ldots, \theta_n^{(i)})$
3. We repeat the above steps k times (Markov chain Monte Carlo).

The samples approximate the joint distribution of all variables.

Can incorporate various algorithms (Metropolis-Hastings, slice sampling, adaptive rejection sampling, Hamiltonian Monte-Carlo) to implement one or more of the sampling steps.

Gibbs sampling

- Involves several loops : MCMC step, parameters
- Random draws
- Matrix computations in case of conjugated priors in Bayesian statistics

Rcpp (C++), RcppGSL (random draws) and RcppArmadillo (matrix computations) are useful for efficient Gibbs sampling.

R code

```r
gibbs_r <- function(N, thin) {
    mat <- matrix(nrow = N, ncol = 2)
    x <- y <- 0

    for (i in 1:N) {
        for (j in 1:thin) {
            x <- rgamma(1, 3, y * y + 4) # Gamma(shape, rate) with R
            y <- rnorm(1, 1 / (x + 1), 1 / sqrt(2 * (x + 1)))
        }
        mat[i, ] <- c(x, y)
    }
    mat
}
```

Rcpp

C++ code with Rccp

```cpp
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericMatrix gibbs_rcpp(int N, int thin) {
    NumericMatrix mat(N, 2);
    double x = 0, y = 0;

    for(int i = 0; i < N; i++) {
        for(int j = 0; j < thin; j++) {
            x = R::rgamma(3, 1 / (y * y + 4)); // R::rgamma(shape, scale)
            y = R::rnorm((1 / (x + 1)), 1 / sqrt(2 * (x + 1)));
        }
        mat(i, 0) = x;
        mat(i, 1) = y;
    }

    return(mat);
}
```

Rcpp + RcppArmadillo + RcppGSL

C++ code with RccpArmadillo and RcppGSL

```
#include <RcppArmadillo.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::depends(RcppGSL)]]

// [[Rcpp::export]]
arma::mat gibbs_rcpp_arma_gsl(int N, int thin) {
    gsl_rng *s = gsl_rng_alloc(gsl_rng_mt19937); // Create RNG seed
    arma::mat mat; mat.zeros(N, 2);
    double x = 0, y = 0;
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < thin; j++) {
            x = gsl_ran_gamma(s, 3,
                      1 / (y * y + 4)); // Gamma(shape, scale)
            y = (1 / (x + 1)) +
                gsl_ran_gaussian_ziggurat(s, 1 / sqrt(2 * (x + 1)));
        }
        mat(i, 0) = x;
        mat(i, 1) = y;
    }
    gsl_rng_free(s); // Free memory
    return(mat);
}
```

Benchmark

```
# Libraries
library(rbenchmark)
# Benchmark
Benchmark <- benchmark(
    "R" = {gibbs_r(100, 10)},
    "rcpp" = {gibbs_rcpp(100, 10)},
    "rcpp_arma_gsl" = {gibbs_rcpp_arma_gsl(100, 10)},
    replications=30,
    columns = c("test", "elapsed", "relative")
)
Benchmark
```

```
##             test elapsed relative
## 1              R   0.186       62
## 2           rcpp   0.009        3
## 3 rcpp_arma_gsl   0.003        1
```

Outline

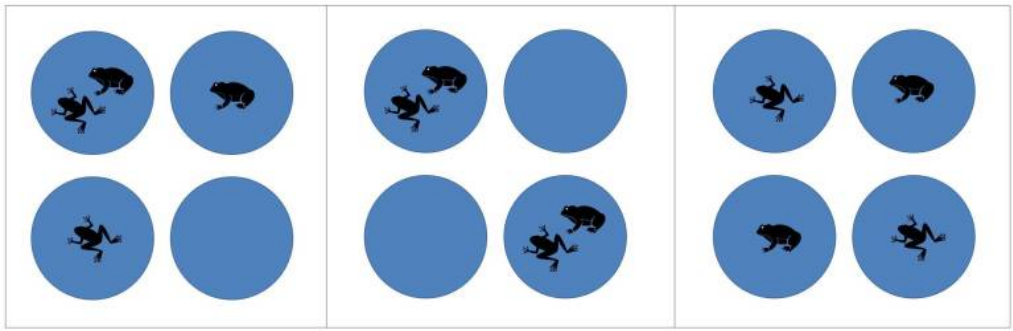
1. Short presentation of Rcpp* packages
   - Rcpp : extending R with C++
   - RcppGSL for fast random draws
   - RcppArmadillo for high-performance linear algebra
2. Rcpp for Gibbs sampling
   - Gibbs sampling and Bayesian statistics
   - A simple Gibbs sampler
   - Benchmark
3. The `jSDM` R package
   - Joint Species Distribution Models
   - Model specification
   - Comparison with `boral`/`JAGS`

cirad

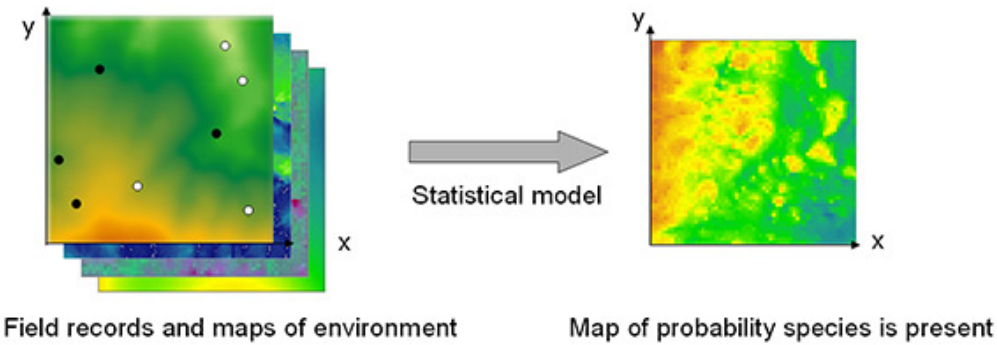
botAnique et Modélisation de l'Architecture des Plantes et des végétations

JSDM utility

- Fit species distribution models
- Accounting for species interaction



- Can be used to explain/predict species range and produce species range map



Field records and maps of environment

Map of probability species is present

Data to fit JSDM

- Species presence/absence on sites
- Environmental variables (climate, lancover) at each site

Sites	Sp1	Sp2	...	Sp_nsp	X1	X2	...	X_nvar
Site1	0	0	...	1	-0.21	-1	...	-1.24
Site2	0	1	...	1	0.25	0	...	-0.53
...	...	...	...	...	...	...	...	-
Site_nsite	1	0	...	1	0.82	1	...	0.34

Statistical model

$Y = \{y_{ij}\}_{j=1,\dots,nsp}^{i=1,\dots,nsite}$, with :

$$y_{ij} = \begin{cases} 0 & \text{if species } j \text{ is absent on site } i \\ 1 & \text{if species } j \text{ is present on site } i. \end{cases}$$

We assume $y_{ij} \sim \mathcal{B}ernouilli(\theta_{ij})$, with :

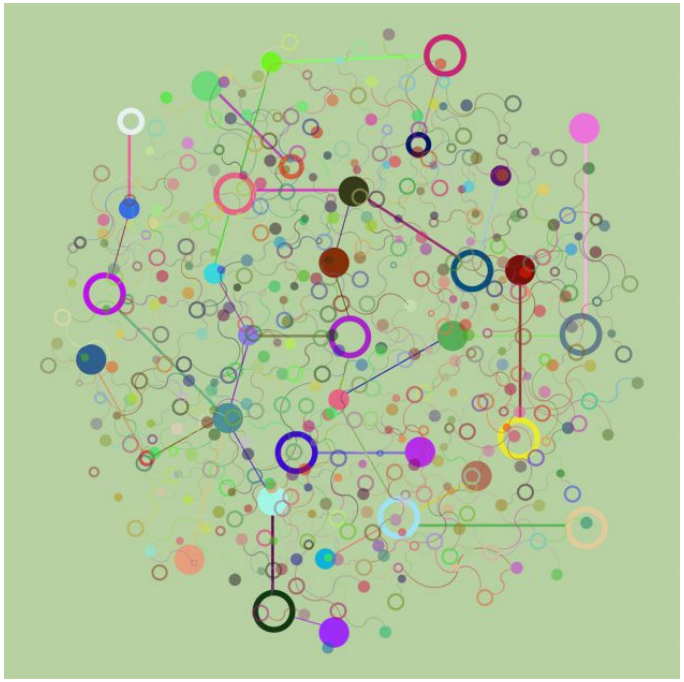
$$\text{probit}(\theta_{ij}) = \alpha_i + \beta_{0j} + X_i\beta_j + W_i\lambda_j$$

α_i : site random effects, with $\alpha_i \sim \mathcal{N}(0, V_\alpha)$
X_i : known environmental variables on site i
W_i : latent variables for site i
β_j, λ_j : species fixed effects

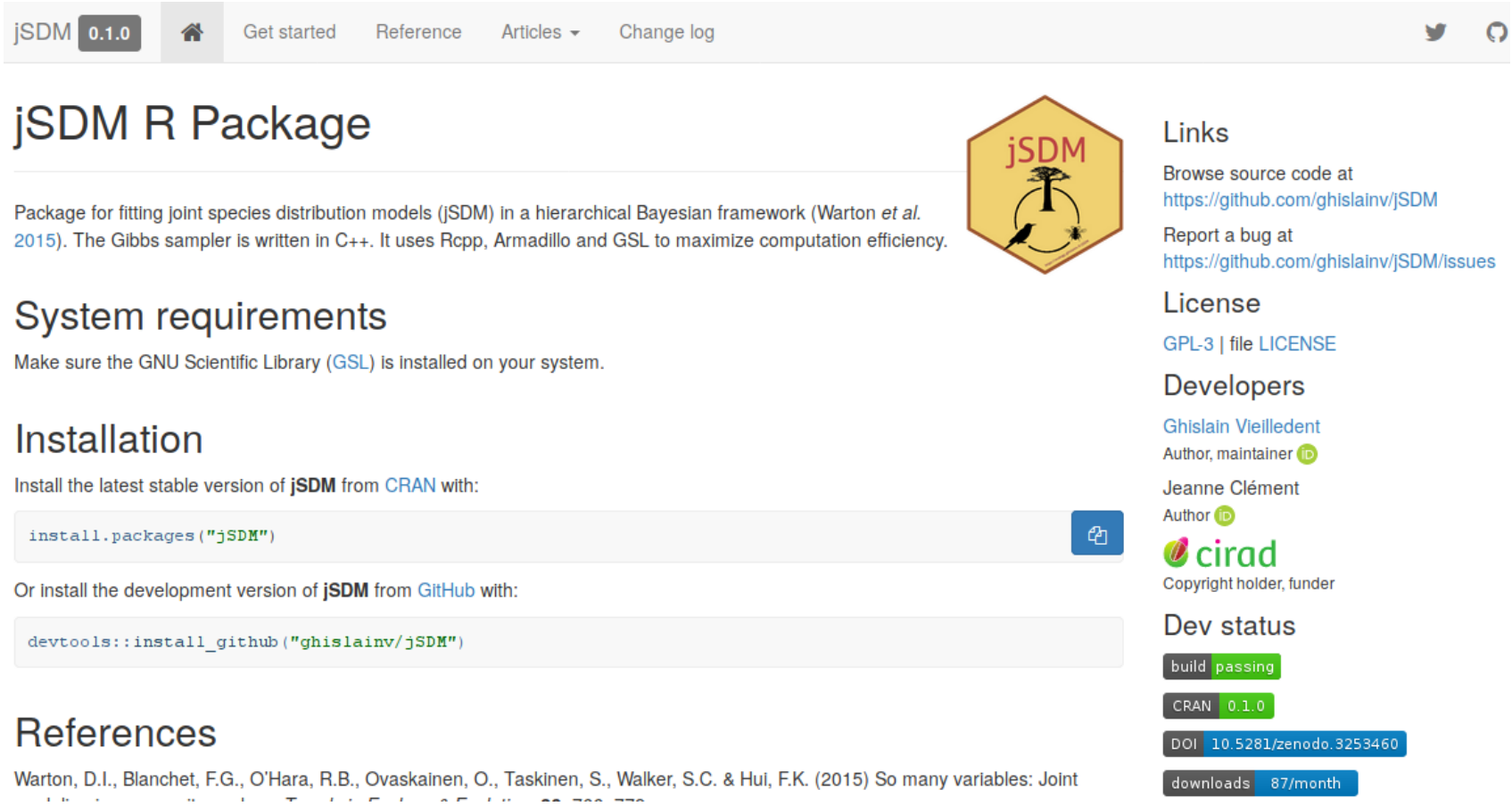
Latent variables W_i : missing predictors + main axes of covariation across taxa (see Warton et al. 2015 <doi : 10.1016/j.tree.2015.09.007>).

Complexity of the model

- Multi-dimensionality : parameters α_i for sites and β_j, λ_j for species
- Non Gaussian process
- Latent-variables : W_i
- Mixed model with site random effects $\alpha_i \sim \mathcal{N}(0, V_\alpha)$



jSDM R package



- https://ecology.ghislainv.fr/jSDM
- Made with Rcpp* packages

boral R package

- R package interfacing R with JAGS for fitting Joint Species Distribution Models
- JAGS is Just Another Gibbs Sampler : http://mcmc-jags.sourceforge.net/
- Approach used in Warton et al. 2015 : <doi : 10.1016/j.tree.2015.09.007>
- boral by Francis K.C. Hui and JAGS by Martyn Plummer

Data-sets

dataset	nsite	nsp	nobs	nX	nlat	npar	nmcmc
Simulated	300	100	30000	2	2	1400	35000
Mosquito	167	16	2672	13	2	757	35000
Eucalypts	458	12	5496	7	2	1494	35000
Frogs	104	9	936	3	2	366	35000
Fungi	800	11	8800	12	2	2565	35000

Mosquitos

Eucalyptus

Frogs

Fungi

Comparison results

Compilation time (in minutes)

	Simulated	Mosquitos	Eucalyptus	Frogs	Fungi
boral	96.9	5.8	17.2	1.2	38.6
jSDM	7.0	1.3	1.8	0.3	4.1

jSDM is **4 to 14** times faster than boral/jags.

Root-mean-square error

Computed for probit(θ_{ij}) with the simulated data-set.

	boral	jSDM
RMSE	1.8	0.6

Deviance

	Simulated	Mosquitos	Eucalyptus	Frogs	Fungi
boral	40486	6936	8779	884	12871
jSDM	15651	1231	1922	150	1982

Conclusion

- Small data-sets **and** simple models : R, *BUGS, JAGS, Stan, INLA, MCMCglmm
- Large data-sets **or** complex hierarchical models : R + Rcpp + RcppGSL + RcppArmadillo

- With Rcpp* packages, the Gibbs sampler can typically be written in about half a day
- Code is reusable and easily packageable
- Tools of incomparable efficiency for statisticians

. . . Thank you for attention . . .

@ghislainv

https://ecology.ghislainv.fr

ghislain.vieilledent@cirad.fr | jeanne.clement16@laposte.net