Tutorial of RCaN

Hilaire Drouineau - INRAE Bordeaux, UR EABX
Benjamin Planque - IMR
Christian Mullon - IRD

Contents

```r
library(RCaN)
```

This package aims at fitting a chance and necessity ecosystem model. A full description of this kind of model is provided in

Planque, B., & Mullon, C. (n.d.). Modelling chance and necessity in natural systems. ICES Journal of Marine Science. https://doi.org/10.1093/icesjms/fsz173

Data requirements

Inputs should be provided in an xlxs data file. This file can be generated using an independent java graphical user interface, but it can be created independently. An artificial file is provided in the package as a template and as an example for this vignette. You can find the path to the file by entering

```r
system.file("extdata", "CaN_template_mini.xlsx", package = "RCaN")
#> [1] "/media/Data/Documents/Bordeaux/equipe estuaire/CaN/can/RCaN/inst/extdata/CaN_template_mini.xlsx
```

There are a few important things to remember: * the names of the sheets should match the names of the sheets in the template (i.e. "Components & input parameter", "Fluxes", "Input time-series", "Constraints") * the names of the columns in sheets "Components & input parameter", "Fluxes" and "Constraints" and there should be a column "Year" in the sheet "Input time-series" * it is worthwile giving as much information as possible in the "Info" sheet * you can add other sheets and other columns in your file is needed, as the minimal sets of sheets and columns are provided

Sheet "Components & input parameter"

```
#>           Component in_out InitialBiomass AssimilationE Digestibility
#> 1 PhytoAndBacteria    Out             NA            NA          0.65
#> 2  HerbZooplankton     In          16608             1          0.90
#> 3  OmniZooplankton     In          16864             1          0.90
#> 4          Fishery    Out             NA            NA            NA
#>   OtherLosses Inertia Satiation RefugeBiomass
#> 1          NA      NA        NA            NA
#> 2         8.4     7.6       128        166.08
#> 3         5.5     3.1        42        168.64
#> 4          NA      NA        NA            NA
```

Components correspond to the boxes of the trophic food web. Most components are internal to the ecosystem (denoted "In"), but other can be external, for example to deal with fishery or migrations, and are denoted "Out". The definition of the parameters are provided in Planque and Mullon (in press). You can leave some cells empty: * if you do not want to use a specific constraint (for example, if you do not want to use a satiety constraint for a component, leave the cell empty) * for "Out" component, you can leave all cells empty since the dynamics of external components are not modelled, except digestibility of components that are eated by internal components (for example in the template, we have a digestibility for PhytoAndBacteria).

Sheet "Fluxes"

```
#>   Flux             From              To Trophic
#> 1  F01 PhytoAndBacteria HerbZooplankton       1
#> 2  F02 PhytoAndBacteria OmniZooplankton       1
#> 3  F04  HerbZooplankton OmniZooplankton       1
#> 4  F08  OmniZooplankton         Fishery       0
```

Column "Flux" aims at providing name to identify the differents fluxes. Then, "From" and "To" define the different (oriented) flow in the food web. The names in "From" and "To" should match the names in Components (case sensitive)

Sheet "Input time-series"

```
#>   Year HerbZooplankton_Biomass OmniZooplankton_Biomass Benthos_Biomass
#> 1 1988                   16608                   16864          105000
#> 2 1989                   27872                   13616          105000
#> 3 1990                   23504                    7696          105000
#> 4 1991                   21776                   14640          105000
#> 5 1992                   26816                    7008          105000
#>   PelagicFish_Biomass DemersalFish_Biomass Mammals_Biomass Birds_Biomass
#> 1                 576                 1472             560          11.2
#> 2                1200                 1376             560          11.2
#> 3                6304                 1392             560          11.2
#> 4                8304                 2112             560          11.2
#> 5                7232                 2976             560          11.2
#>   OmniZooplankton_Landings PelagicFish_Landings DemersalFish_Landings
#> 1                   48.689              114.242               545.886
#> 2                   62.748              123.017               414.012
#> 3                   81.164               95.948               274.270
#> 4                   74.862             1036.327               403.927
#> 5                   68.568             1271.204               588.490
#>   Mammals_Landings PrimaryProduction
#> 1         10.79485             1e+06
#> 2          6.51655             1e+06
#> 3          6.32185             1e+06
#> 4          6.29700             1e+06
#> 5          5.97415             1e+06
```

Here you provide all the time series you have. The first column correspond to years and define the time range of the model. Then following time series correspond to the different time series (observations) that you will use to define constraints.

sheet "Constraints"

```
#>    Id                          Constraint Time.range
#> 1 C01     F01+F02<=PrimaryProduction*1.3  1988:1991
#> 2 C02 -(F01+F02)<=-PrimaryProduction*0.7  1988:1991
#> 3 C03       F08=OmniZooplankton_Landings  1988:1991
```

The "Id" correspond to identifiers of the constraints. Then, the constraint correspond to a mathematical linear equation of constraint. You can potentially use "sum" and "mean" to computes sum or means over the time range. Both inequality ("<=" or ">=") and equality ("=") constraints can be used. Time.Range defines the time range of the constraint, they can be denoted either either with "firstyear:lastyear" or as a vector of years "c(year1,year2…)"

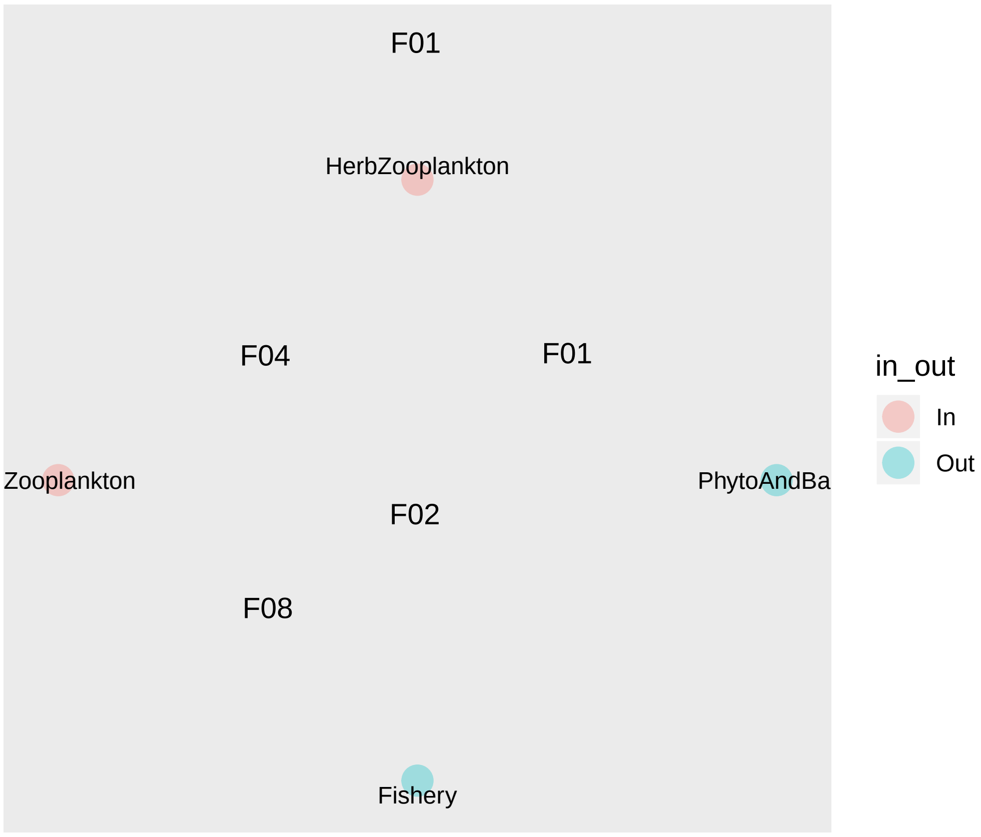
Building the model

The first step of the workflow is to build the model. The function build_CaNmod does this job. It reads the template file and provides a CaNmod object that includes all necessary data to fit the model. You see here the different elements included in the object.

```
myCaNmod <- build_CaNmod(system.file("extdata", "CaN_template_mini.xlsx", package = "RCaN"))
names(myCaNmod)
#>  [1] "components_param" "species"          "fluxes_def"
#>  [4] "flow"             "series"           "ntstep"
#>  [7] "data_series_name" "constraints"      "H"
#> [10] "N"                "A"                "C"
#> [13] "v"                "L"                "M"
#> [16] "b"                "symbolic_enviro"
```

Everything should be all right if your template was well created. The function retrieves meaningful error message if a problem is detected. Two functions can be used to check that the model is correct. The function ggCaNmod draws a graph to show you the trophic food web

```
g<-ggCaNmod(myCaNmod)
print(g)
```



The function pdfCaNmod builds an even more detailed summary of the model as a pdffile

Checking the model configuration

The set of constraints define a convex polytope, i.e. a conspace subspace of a n-dimensional space (n corresponding to the number of the parameter) with flat sides. Each point inside the polytope is a possible solution of the model. The idea of Chance and Necessity model is to uniformly sample within those polytopes in order to explore the whole space of possible solutions. Nevertheless, the polytope should not be empty (otherwise, there is no solution) and preferentially should be bounded (otherwise a parameter can range to infinity). To check that the polytope is ok, you can use checkPolytopeStatusCaNmod which will tell you the status of your polytope.

```
checkPolytopeStatusCaNmod(myCaNmod)
#> [1] "polytope ok"
```

If your polytope is not bounded, you should probably add additional constraints on not bounded parameter. To see which parameter is not bounded, you can use the function getAllBoundsParamCaNmod. An Inf in the upperbound column would correspond to a non bounded parameter, otherwise you have lower and upperbounds of each parameter

```
getAllBoundsParamCaNmod(myCaNmod)
#>
  |
  |                                                                      |   0%
  |
  |===                                                                   |   5%
  |
  |======                                                           |  10%
  |
  |==========                                                       |  15%
  |
  |=============                                                    |  20%
  |
  |================                                                 |  25%
  |
  |====================                                             |  30%
  |
  |=======================                                          |  35%
  |
  |==========================                                       |  40%
  |
  |=============================                                    |  45%
  |
  |================================                                 |  50%
  |
  |====================================                             |  55%
  |
  |=======================================                          |  60%
  |
  |==========================================                       |  65%
  |
  |==============================================                   |  70%
  |
  |=================================================                |  75%
  |
  |====================================================             |  80%
  |
  |=======================================================          |  85%
  |
  |==========================================================       |  90%
  |
  |==============================================================   |  95%
  |
  |=================================================================| 100%
#>         param lowerbound   upperbound
#> 1  F01[1988]   2098.474  1300000.000
#> 2  F02[1988]      0.000   708288.000
#> 3  F04[1988]      0.000   708288.000
#> 4  F08[1988]     48.689       48.689
#> 5  F01[1989]   1886.051  1300000.000
#> 6  F02[1989]      0.000  1298086.657
#> 7  F04[1989]      0.000   843769.760
#> 8  F08[1989]     62.748       62.748
#> 9  F01[1990]   1885.286  1300000.000
#> 10 F02[1990]      0.000  1298087.422
#> 11 F04[1990]      0.000   843770.258
#> 12 F08[1990]     81.164       81.164
#> 13 F01[1991]   1885.108  1300000.000
#> 14 F02[1991]      0.000  1298087.610
#> 15 F04[1991]      0.000   843770.375
#> 16 F08[1991]     74.862       74.862
#> 17 F01[1992]      0.000 12862626.425
#> 18 F02[1992]      0.000  6441502.023
#> 19 F04[1992]      0.000  6441502.023
#> 20 F08[1992]      0.000        0.000
```

Since this function can be time consuming, we provide a version that focus on a parameter of interest, here for example the first parameter:

```
getBoundParamCaNmod(myCaNmod, p="F01[1988]")
#> [1]    2098.474 1300000.000
```

If your polytope is empty, it means that there is no possible solutions and that some constraints should be relaxed. A function can help you to find which constraint raise problem: findingIncompatibleConstraintsCaNmod. For example, here we artificially add a constraint corresponding to a negative flow and look at the result:

```
myCaNmod2 <- myCaNmod
#we artificially add incompatible constraints (first flow is negative)
myCaNmod2$A <- rbind(myCaNmod2$A,c(1,rep(0,ncol(myCaNmod2$A)-1)))
rownames(myCaNmod2$A)[nrow(myCaNmod2$A)]<-"neg_flow"
myCaNmod2$b <- c(myCaNmod2$b,-1)
incomp<-findingIncompatibleConstraintsCaNmod(myCaNmod2)
#> [1] "###polytope is ok when following constraints are relaxed:"
#> [1] " 1989 : Biomass positiveness_refuge_HerbZooplankton,  1990 : Biomass positiveness_refuge_HerbZo
#> [1] "####Those constraints seem incompatible with:"
#> [[1]]
#> [1] " 1989 : Biomass positiveness_refuge_HerbZooplankton:  neg_flow"
#>
#> [[2]]
#> [1] " 1990 : Biomass positiveness_refuge_HerbZooplankton:  1989 : Biomass positiveness_refuge_HerbZo
#>
#> [[3]]
#> [1] " 1988 : inertia_sup_HerbZooplankton:  1989 : Biomass positiveness_refuge_HerbZooplankton,  1990
#>
#> [[4]]
#> [1] " neg_flow: "
```

Here the first line tells you that the algorithm successfully fit the model by relaxing 4 constraints: “neg_flow” is the one we added, two constraints of biomass positiveness and one constraint of inertia. Then the algorithm tell you that “1989 : Biomass positiveness_refuge_HerbZooplankton” appears to be incompatible with “neg_flow”, “1990 : Biomass positiveness_refuge_HerbZooplankton” with both “1989 : Biomass positiveness_refuge_HerbZooplankton” and “neg_flow”… Since neg_flow is always listed in the incompabible list, and that neg_flow can not be subtituted by another constraint (neg_flow:), it would be the first one to be removed.

After this step, you should be able to fit the model.

Fitting the model

The aim of this step is to achieve a uniform sampling within the convex poltyope. To do that, we use the package cpgsR available on github. Since the sampling is based on a mcmc algorithm, it can be a good idea to run several independent chains; the library allows you to run several chain in parallel using multicore facility provided by package parallel and doParallel. For example, here is the solution to run 2 chains in parrallel with 1000 samples in each chain:

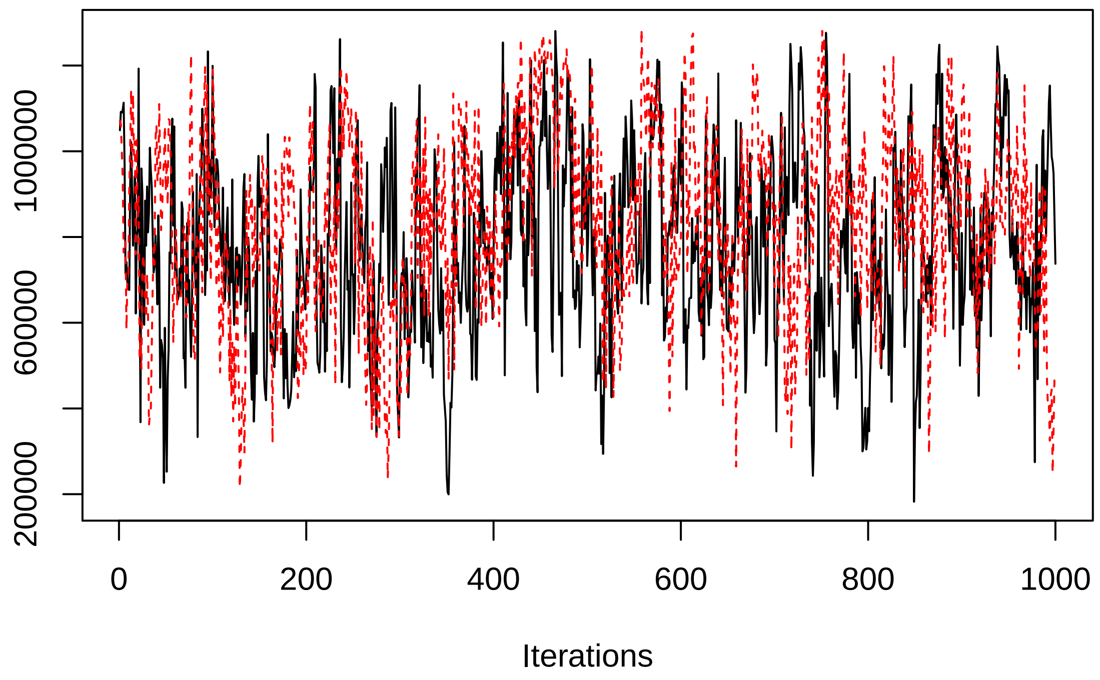
```
res <- fitmyCaNmod(myCaNmod, 1000,nchain=2,ncore=2)
```

The argument ncore allows to use several cores in parallel, however the library ensure that the computation leave one core unused and that the number of cores remain smaller than the number of chains

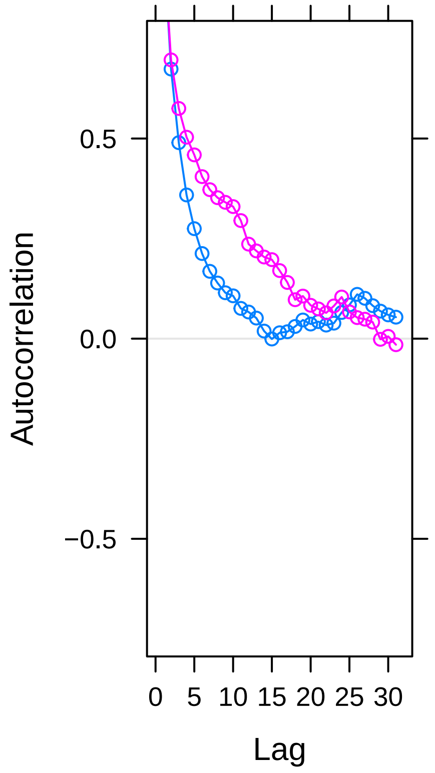
Since the result is a mcmc.list object, we can use all the diagnostics tools provided in the package coda, such traceplot, summary or gelman and rubin tests or autocorrelograms.

```
library(coda)
nchain(res)
#> [1] 2
summary(res)
#>
#> Iterations = 1:1000
#> Thinning interval = 1
#> Number of chains = 2
#> Sample size per chain = 1000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>                             Mean        SD  Naive SE Time-series SE
#> F01[1988]             8.043e+05 2.156e+05 4.820e+03      1.409e+04
#> F02[1988]             2.278e+05 1.542e+05 3.449e+03      1.156e+04
#> F04[1988]             2.201e+05 1.480e+05 3.310e+03      8.166e+03
#> F08[1988]             4.869e+01 0.000e+00 0.000e+00      0.000e+00
#> F01[1989]             6.805e+05 2.493e+05 5.575e+03      2.289e+04
#> F02[1989]             3.571e+05 2.362e+05 5.281e+03      2.060e+04
#> F04[1989]             1.960e+05 1.466e+05 3.278e+03      1.178e+04
#> F08[1989]             6.275e+01 0.000e+00 0.000e+00      0.000e+00
#> F01[1990]             7.379e+05 2.575e+05 5.757e+03      2.130e+04
#> F02[1990]             3.418e+05 2.489e+05 5.565e+03      2.206e+04
#> F04[1990]             2.305e+05 1.579e+05 3.532e+03      1.081e+04
#> F08[1990]             8.116e+01 0.000e+00 0.000e+00      0.000e+00
#> F01[1991]             7.833e+05 2.515e+05 5.623e+03      2.256e+04
#> F02[1991]             3.495e+05 2.457e+05 5.494e+03      2.314e+04
#> F04[1991]             2.757e+05 1.654e+05 3.699e+03      1.530e+04
#> F08[1991]             7.486e+01 0.000e+00 0.000e+00      0.000e+00
#> F01[1992]             1.731e+06 1.428e+06 3.192e+04      9.499e+04
#> F02[1992]             1.267e+06 1.005e+06 2.248e+04      7.604e+04
#> F04[1992]             1.209e+06 9.571e+05 2.140e+04      7.848e+04
#> F08[1992]             4.986e+10 2.859e+10 6.393e+08      1.012e+10
#> HerbZooplankton[1988] 1.661e+04 0.000e+00 0.000e+00      0.000e+00
#> HerbZooplankton[1989] 3.603e+04 1.803e+04 4.032e+02      6.423e+02
#> HerbZooplankton[1990] 2.932e+04 1.701e+04 3.803e+02      9.118e+02
#> HerbZooplankton[1991] 2.966e+04 1.768e+04 3.954e+02      1.063e+03
#> HerbZooplankton[1992] 2.779e+04 1.558e+04 3.485e+02      1.316e+03
#> OmniZooplankton[1988] 1.686e+04 0.000e+00 0.000e+00      0.000e+00
#> OmniZooplankton[1989] 6.274e+04 2.238e+04 5.005e+02      8.236e+02
#> OmniZooplankton[1990] 7.422e+04 2.727e+04 6.098e+02      1.491e+03
#> OmniZooplankton[1991] 7.808e+04 2.807e+04 6.276e+02      1.781e+03
#> OmniZooplankton[1992] 8.637e+04 2.399e+04 5.365e+02      1.620e+03
#>
```

```
#> 2. Quantiles for each variable:
#> 
#>                             2.5%       25%       50%       75%     97.5%
#> F01[1988]              3.846e+05 6.601e+05 8.027e+05 9.615e+05 1.203e+06
#> F02[1988]              1.276e+04 9.887e+04 2.053e+05 3.350e+05 5.544e+05
#> F04[1988]              7.364e+03 9.984e+04 1.962e+05 3.236e+05 5.440e+05
#> F08[1988]              4.869e+01 4.869e+01 4.869e+01 4.869e+01 4.869e+01
#> F01[1989]              2.252e+05 4.936e+05 6.808e+05 8.624e+05 1.155e+06
#> F02[1989]              1.781e+04 1.618e+05 3.278e+05 5.259e+05 8.930e+05
#> F04[1989]              7.006e+03 7.619e+04 1.710e+05 2.835e+05 5.333e+05
#> F08[1989]              6.275e+01 6.275e+01 6.275e+01 6.275e+01 6.275e+01
#> F01[1990]              2.132e+05 5.526e+05 7.510e+05 9.304e+05 1.190e+06
#> F02[1990]              1.402e+04 1.385e+05 2.943e+05 4.981e+05 9.345e+05
#> F04[1990]              1.018e+04 1.001e+05 2.052e+05 3.401e+05 5.612e+05
#> F08[1990]              8.116e+01 8.116e+01 8.116e+01 8.116e+01 8.116e+01
#> F01[1991]              2.670e+05 6.024e+05 8.138e+05 9.825e+05 1.193e+06
#> F02[1991]              1.494e+04 1.509e+05 3.039e+05 5.053e+05 9.194e+05
#> F04[1991]              1.631e+04 1.417e+05 2.654e+05 3.945e+05 5.996e+05
#> F08[1991]              7.486e+01 7.486e+01 7.486e+01 7.486e+01 7.486e+01
#> F01[1992]              5.187e+04 5.791e+05 1.367e+06 2.525e+06 5.374e+06
#> F02[1992]              5.272e+04 4.559e+05 1.035e+06 1.874e+06 3.616e+06
#> F04[1992]              3.549e+04 4.471e+05 9.862e+05 1.770e+06 3.529e+06
#> F08[1992]              3.023e+09 2.589e+10 4.910e+10 6.829e+10 1.073e+11
#> HerbZooplankton[1988] 1.661e+04 1.661e+04 1.661e+04 1.661e+04 1.661e+04
#> HerbZooplankton[1989] 7.837e+03 2.142e+04 3.468e+04 4.840e+04 7.378e+04
#> HerbZooplankton[1990] 6.071e+03 1.554e+04 2.603e+04 4.020e+04 6.908e+04
#> HerbZooplankton[1991] 6.119e+03 1.529e+04 2.611e+04 4.136e+04 6.796e+04
#> HerbZooplankton[1992] 4.401e+03 1.579e+04 2.510e+04 3.768e+04 6.248e+04
#> OmniZooplankton[1988] 1.686e+04 1.686e+04 1.686e+04 1.686e+04 1.686e+04
#> OmniZooplankton[1989] 1.898e+04 4.598e+04 6.421e+04 8.075e+04 1.008e+05
#> OmniZooplankton[1990] 2.276e+04 5.419e+04 7.439e+04 9.411e+04 1.243e+05
#> OmniZooplankton[1991] 2.605e+04 5.748e+04 7.837e+04 9.905e+04 1.278e+05
#> OmniZooplankton[1992] 3.807e+04 6.949e+04 8.782e+04 1.037e+05 1.298e+05
traceplot(res[,"F01[1988]"])
```

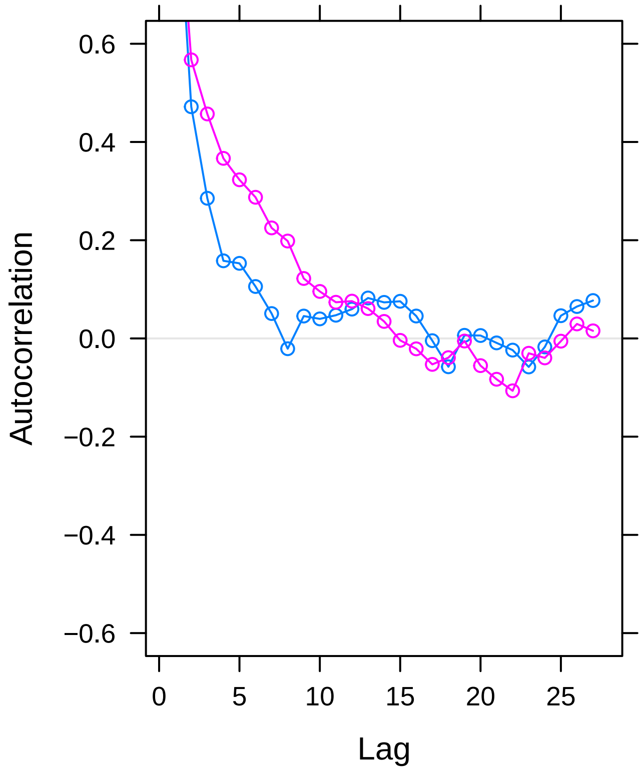


```
gelman.diag(res[,"F01[1988]"])
#> Potential scale reduction factors:
#>
#>      Point est. Upper C.I.
#> [1,]       1.03       1.14
acfplot(res[,"F01[1988]"])
```



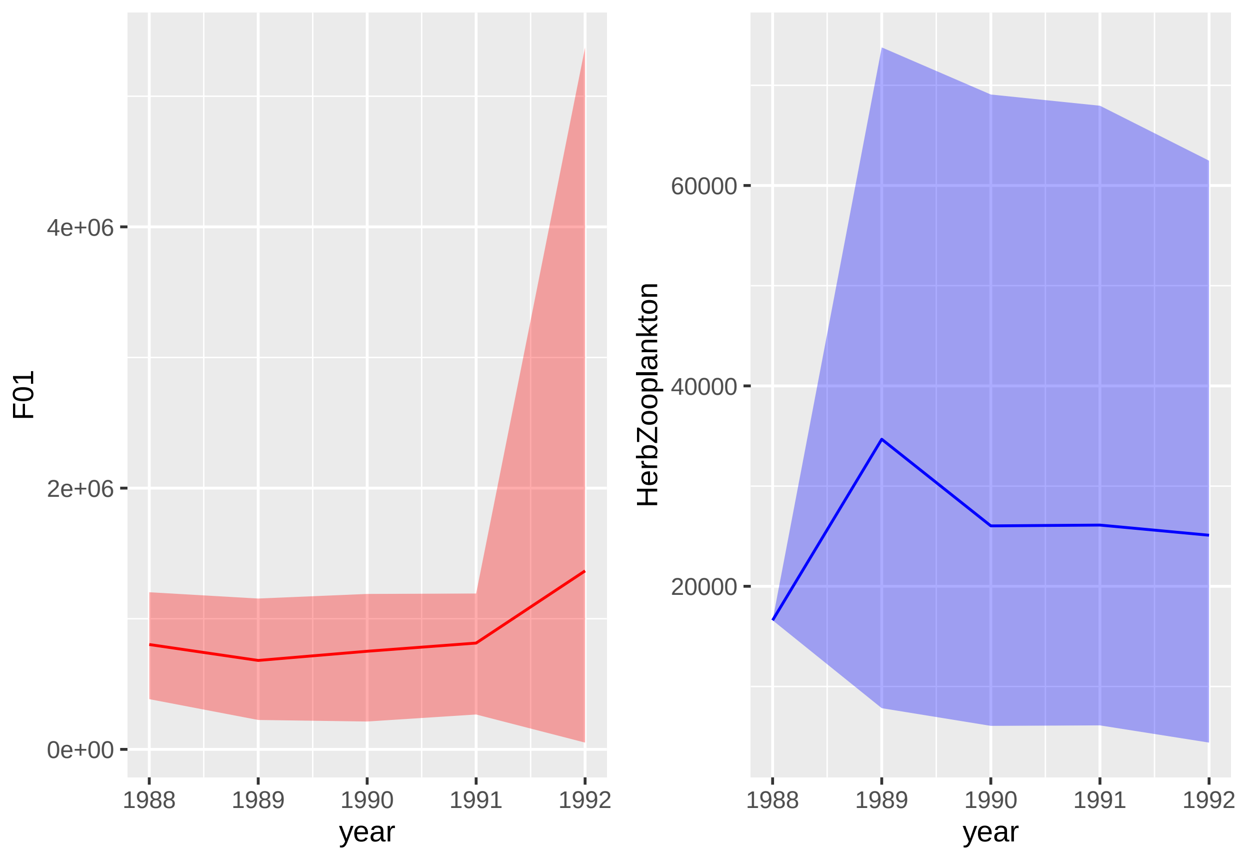
We can also do a posteriori thinning if required, for example if we chose to keep only one iteration on two:

```
thin(res)
#> [1] 1
thinned_res <- window(res,thin=2)
thin(thinned_res)
#> [1] 2
acfplot(thinned_res[,"F01[1988]"])
```



The package provides a function that draw a ggplot of time series of biomass or flux. For example, here we draw a combination of time trends of “HerbZooPlankton” and “F01”:

```
herbzoo<-ggResult(res,myCaNmod,"HerbZooplankton","blue")
f01<-ggResult(res,myCaNmod,"F01","red")
library(gridExtra)
grid.arrange(f01,herbzoo,nrow=1)
```


F01
4e+06
2e+06
0e+00
1988
1989
1990
1991
1992
year
HerbZooplankton
60000
40000
20000
1988
1989
1990
1991
1992
year