RCAN : A SOFTWARE FOR CHANCE AND NECESSITY MODELLING

HILAIRE DROUINEAU, BENJAMIN PLANQUE, AND CHRISTIAN MULLON

Abstract. Uncertainty is a key challenge in modelling marine systems and a source of misunderstandings between scientists and other approach. Recently, a Chance and Necessity approach has been proposed to address this issue in trophic network modelling. Instead of seeking for a precise description of the system functioning and trajectory, CaN approach (a) focuses on the few certain physical and physiological laws and (b) samples all trajectories that can satisfy available observations that have been collected about the system. This type of approach is thought to facilitate discussion among actors since (i) it does not require complex skills and knowledge and (ii) it is based on information shared by most actors. However, to be a good discussion support, user-friendly software are required to easily implement such models.

In this paper, we present a software allowing to quickly develop this approach about a marine ecosystem that has been observed for several years. RCaN is a library of R commands. It implements efficient algorithms allowing to sample polytopes in spaces with a very large number of dimensions. RCaNGUI is a graphical interface to RCaN.

We recall the main principles of CaN modelling, we present the software that has been developed and we give all the details of its use to the analyze of the trophic marine system of the Barents sea.

1. Introduction

Rather than attempting to model multiple ecological processes in details, CaN modelling focuses on exploring possible ecological dynamics given a set of constraints. The principles of CaN modelling and the corresponding mathematical formulation have been outlined in [1]. The general idea is that it is easier for various actors (managers, scientists, stakeholders) to agree about ecological constraints, which separate what is possible from what isn't, than to agree on the mechanistic formulation of detailed ecological processes. This is particularly true when modelling complex ecological systems for which available information/observations are limited. This approach is called: "Chance and Necessity" modelling (CaN): chance refer to the indeterminacy in ecological processes while necessity reflects the constraints that delineate what is possible from what isn't.

While the CaN principles can be applied to a wide range of ecological problems, the first application was specifically designed to explore the temporal dynamics of marine systems [1]. This has been motivated by the shift from conventional single-species fisheries management towards ecosytem-based fishery management (EBFM,

see for example [2, 3]). This shift implies upscaling of the traditional population scale management into a complex set of biotic (e.g. trophic food interactions) and abiotic sets of interactions (e.g. interactions with physical habitats). This has led to the development of complex ecological models, but which ability to help management is questioned by many actors who may have limited trust in such models [4]. In this context, there is a call for transparency in the science that supports management, and participatory modelling and assessment have been proposed a a way to rebuild mutual trust. CaN is proposed as one tool to support transparency and participatory modelling by providing a simplified food-web modelling framework, in which the ideas of uncertainty and ecological limits are central. Assumptions about knowns and unknowns and about possible and impossible processes are presented explicitly and all model constituents can be described in plain language (no modelling jargon).

By contributing to the transfer of biomass, energy and polluants in ecosystems, trophic interactions play a major role in ecosystem functioning. Trophic food web models have proved to be key tools for the quantification of these fluxes, they provide a holistic view of ecosystem functioning, and are increasingly used to assess ecosystem ecological status and to explore the effect of anthropogenic pressures such as fisheries [5]. Food web models, mostly Ecopath with Ecosim (EwE) [6, 7], have played an important role in the implementation of the EBFM. CaN modelling further contributes to this effort by providing a modelling framework for marine food-webs that primarily relies on ecological constraints which can be agreed upon by diverse actors.

The principles of CaN modelling have been outlined in [1]. The main technical difficulty: sampling a polytope in a highly dimensional space have been tackled (inside the R framework). Practical results have been obtained [1].

Until now, there was no dedicated software to easily perform CaN modelling. Initial way of doing was collecting and structuring data, then running and improving R-scripts to obtain results. In a second time, a standardization occurs with the definition of a common format for a RCaN-file (a template) and the R-scripts have been organized as a R-package.

Then appeared the necessity of going further, of allowing the user to manage the RCaN file and the simulations in a more interactive way, to replace all processes in the common context of a scientific project.

(1) For CaN modelling to be a useful support to discussion, there is a need for a user-friendly software that can handle the complex numerical and mathematical parts of the modelling. In such way, users can then focus on the discussion about their knowledge of the ecological system and how to best

structure a model that fits their needs. Ideally, the software should include a graphical user interface (GUI) to facilitate joint model construction through a visual representation of the trophic network, and direct entry of model parameter values, observational data and system constraints.

(2) Moreover, file management can become difficult : coherency inside data can be lost. One has to avoid links without corresponding nodes, constraints related to a non-existing components.

(3) Importation of observation data-files often result in the repetition of many procedures.

(4) Easily produced graphical outputs of the model results are a desired feature to promote discussions among modellers and other actors and to ease interpretation of model results.

(5) Registration of meta-information about the model is also an essential element to ensure transparency and reproducibility of the modelling exercise.

Thus, there is a need for managing output of simulations, for managing meta information, for keeping track of the workflow, about the modelling experience. Therefore an user friendly has been developed to help the user who is confronted to these issues.

In this contribution, we present an implementation of CaN modelling in R, called RCaN, and an associated GUI in Java, called RCaNconstructor. RCaN and RCaNconstructor can be used jointly or separately. Their combined use allow users to easily construct, document, sample and interprete the results of CaN food-web models.

The relationship between RCaN and RCaNconstructor is shown in figure 1.

In section 2 we recall the elements of the RCaN modelling approach. In section 3, we present the format of an RCan file. In section 4 we present the R library RCaN which is designed to construct, sample and produce graphical outputs of CaN food-web models. In section 5 we then present RCaNconstructor, the graphical user interface that facilitates the co-construction of CaN models and the inclusion of meta-information. In section 6, we briefly describe the Barents Sea trophic network which we use as a case study example. Finally, we discuss...

2. Principles of the CaN approach

In CaN, food-web dynamics is defined by time-trajectories of biomass and fluxes between trophic groups. The biomass of all trophic groups and the fluxes between them can usually not be precisely observed. Therefore, the food-web dynamics is indeterminate, and a range of dynamics is possible. However, not everything is possible because some ecological or physiological constraints are known to bound the temporal evolution of ecological system, and/or because some observations can

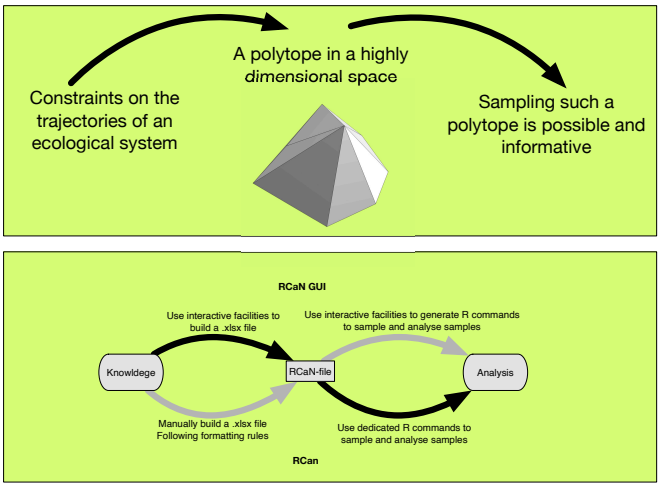

FIGURE 1. Top: Principles of CaN modelling. Bottom: RCaN and RCaNconstructor: The usual ways of using RCAN, with a common element: a RCaN-file.

inform about the past dynamics of some components of the food-web. These eco-physiological constraints and observations can be translated into mathematical constraints that need to be satisfied before a food-web trajectory can be considered possible. The ensemble of these mathematical constraints constitutes the "necessity" in a CaN model. It is by focusing on constraints that can be easily expressed and communicated between actors and by explicitly acknowledging indeterminacy in natural systems that CaN modelling can facilitate exchanges among actors to support resource management.

2.1. **Network structure.** In CaN modelling, the food-web is defined as a network. In this network, each vertex/node corresponds to a species, or more generally a trophospecies, i.e. a group or sub-group of species that share common prey and predators. Trophospecies are usually within the model domain and their biomass trajectories are explicitly considered. It is however possible to add species that are outside the model domain, and for which the biomass trajectories are not considered. Examples of species outside the model domain are provided in the case study in section 6. Each edge/link corresponds to a flux of biomass from/to trophospecies. Most links are trophic links (from a prey to a predator). It is also possible to add non-trophic links which can represent import/export of biomass from adjacent regions or removal of biomass through fishing.

2.2. **Components characteristics.** Then, each species or component is characterized by the following biological properties. **Satiation** reflects the maximum

consumption rate per unit biomass of the predator. **Inertia** expresses that variations in biomass from one year to the next are bounded between a maximum growth rate and a maximum mortality rate. **Digestibility** is a correction factor that accounts for variations in energy content between prey. **Assimilation efficiency** expresses the proportion of the biomass ingested by a predator that is effectively assimilated; the product of the potential assimilation efficiency by the digestibility correction factor is the absorption efficiency (the proportion of prey biomass digested and absorbed). **Refuge biomass** is the absolute minimum biomass that a species group can reach. **Other losses** is a mortality coefficient that account for losses, i.e. metabolic losses and other mortality, not explicitly accounted for in the model.

Components outside the model domain do not require input parameter except when they constitute a prey, in which case digestibility should be provided.

2.3. **Observations.** The CaN modelling approach emphasizes the past observations of the dynamics of the food-web. These observations can be, for example: catch or landings data, biological samples of specific populations, stomach samples or other sources of diet information, survey abundance estimates, outputs from stock assessment models, estimated ranges of biomass, and so on.

2.4. **Constraints.** Constraints that limit the dynamics of the food-web are at the heart of CaN modelling and they are the way by which most of the knowledge and data enter a CaN model. Some constraints are implicitly incorporated into CaN while others can be explicitly formulated.

The few implicit constraints are standard for all CaN models. They are related to the above components characteristics. These are as follows: (a) biomass of compartments are positive and above refuge biomass, (b) trophic fluxes between compartments are positive, (c) the sum of fluxes into a compartment is limited according to its biomass (satiation), (d) the proportional changes in biomass of a compartment are bounded (inertia).

Additional explicit constraints can be provided. These are written in the form of inequalities or equalities. The left side of the (in)equality must contains a reference to one or several compartments or fluxes. The right side of the (in)equality can contain fixed values, compartments, fluxes, data time series.

Roughly speaking, there are two kinds of additional explicit constraints. Some constraints are the expression of a knowledge about system functioning and relate the biomass of a components to its incoming or outgoing fluxes. Some other constraints relate the possible values of a biomass or a flux to observations; here is the way to deal with uncertainties.

2.5. **Trajectories.** A food-web trajectory is defined by the ensemble of biomass fluxes at each time-step and by the initial food-web configuration, i.e. the initial biomass in each compartment. Each trajectory can be expressed as a single point in a multi-dimensional space of dimension $P = nF \times nt + nB_0$, with nF the number of fluxes, nt the number of time steps and nB_0 the number of initial biomasses. This dimension can be very high: a simple ecosystem has about 10 components and 20 types of fluxes; following it during 15 time steps results in $P = 20 \times 15 + 10 = 310$.

All constraints are linear equalities or inequalities. The can be expressed as bearing on food-web trajectories. As constraints are linear, the set of the trajectories that satisfy all constraints has the shape of a convex polytope in a high dimensional space.

2.6. **Sampling trajectories.** The main tool of a CaN modelling approach is the sampling of possible trajectories given (1) the food-web structure, (2) the species biological properties, (3) available observations and (4) constraints. The collection of sampled trajectories expresses the indeterminacy in the system.

Sampling food-web trajectories is achieved by sampling this high-dimension polytope using dedicated sampling algorithms.

3. RCaN file

In RCaN, all input information is provided in a simple excel spreadsheet.

In sheet **Components**, first column corresponds to the name of a component. The second column indicates whether the component is inside our outside the trophic network. Other columns correspond to components characteristics that have been presented above.

In sheet **Trophic fluxes**, each flux is characterized by an Id (first column) and links a source component (column From) to a sink component (column To) and by its type, trophic or non trophic (last column); in a trophic flux, digestibility and assimilation should be applied to the flux.

In sheet **Observations**, first column gives the identifier of time steps (most of the time years). Other columns correspond to the extracted time series of observations that can be used to constrain the system (see 2.3).

In sheet **Constraints**, rows correspond to the constraints on biomasses, fluxes and time series. Each constraint has an Id (first column), a formula defining the linear constraints involving fluxes, time series and biomass in components (second column), a time period of validity (third column) and a flag indicating whether the constraint should be used or not (fourth column, a constraint can be latter activated or deactivated using the *toggle_constraint* function provided in the package).

What is important to notice is that the contents of these sheets are interrelated and that this results in some coherency rules.

4. RcaN

4.1. **Introduction.** RCaN is an R package aiming at building and running CaN trophic food web model. As explained in previous parts, the mathematical structure to analyse is a set of analytical linear equalities and inequalities. See [1] for a complete mathematical exposition.

The rationale is that: (a) constraints related to elementary biological processes or to past observations result in linear equalities or inequalities involving the dynamics of the system (its trajectories), (b) these constraints define a set in a highly dimensional space, this set has the structure of a polytope, (c) a lot can learnt about the system by sampling this polytope.

Implementing the whole process implies (a) to specify entities of the system, (b) to express constraints involving entities and their trajectories, (c) transform set of constraints in a polytope of trajectories, (d) analyse elementary properties of the polytope (non-emptiness, bounds), (e) sampling the polytope, (f) analyse sample.

Let us give her some technical features of the implementation of these different steps in the R language.

4.2. The different steps to use RCaN.

4.2.1. *Build a polytope from RCaN-file.* Once the RCaN-file is ready, the first step is to organize all the information and to build the polytope describing the considered model. More precisely, the mass conservation of biomasses and constraints can be summarized in matrix notation, and all these matrices should be automatically generated whatever are the constraints and parameters used by the modeller.

Constraints can be summarized in a matrix form as $A \cdot x <= b$ (inequality constraints) and $C \cdot x = v$ (equality constraints) with x a solution (e.g. a trajectory defined by initial biomasses per component and fluxes at each time step), A and C two matrices with as many columns as parameters in the model (fluxes per time step and initial biomass per component) and a row per constraint, and vector v and b the bounds of the inequalities.

Using the constraints formula and the symbolic links among variables, RCaN automatically build matrices C and A and vectors b and v for each active constraint. This is done by a simple call to the function **build_CaNmod** which reads all the information from the RCaN-file and returns a single object that represents all the trophic network representation and especially, all the corresponding polytope

matrices. For example, in the Barents Sea case study (seed (6), the function automatically constructs the matrix A which has 2593 rows and 775 columns, and the matrix C which has 54 rows (and by definition, the same number of columns).

4.2.2. *Checking polytope characteristics.* At first try, the specification of model often leads to empty polytope (i.e. a model with no solution that satisfy all the constraints) or unbounded polytope (i.e. a model in which the specified constraints are not sufficient to bound a given parameter that can vary subsequently to infinty). Before running the model, it is wise to check the status of the polytope (the model won't run with an empty polytope while results are generally not relevant with unbounded polytope). The package provides different functions to check the properties of the polytope, of which the two most important are **checkPolytopeStatusCaNmod**, which checks whether the polytope is closed and non empty, and **getAllBoundsParamCaNmod**, which estimates the bounds for all parameters. If the model is empty, the package provides a function **findingIncompatibleConstrCaNmod**, which indicates to the user which constraints raise problems.

4.2.3. *Sampling polytope.* Once all the previous steps are completed, the aim of this step is to achieve a uniform sampling within the convex polytope (if it is not empty). Achieving a uniform sampling is not straightfoward since the a CaN polytope has a very high number of dimensions: the number of time steps times the number of internal components (corresponding to initial biomasses), plus the number of fluxes. This is the role of the function **fitmyCaNmod**, which carry out a uniform sampling within the polytope and returns results in the standard **coda** format of a **mcmc.list** [8]. Based on a Monte-Carlo sampler, the functions allows running several chains in parallel.

4.2.4. *Diagnostics about sampling.* One sampling is done, it is necessary to check whether the uniform sampling was successful. Since the result of **fitCaNmod)**is a mcmc.list object, we can use all the diagnostics tools provided in the package coda, such **traceplot**, **summary** or **gelman.diag** and rubin tests or autocorrelograms.

4.2.5. *Graphical analysis of sampling.* Each iteration of the MCMC sampling provides a possible set of time series of biomasses and fluxes. RCaN provides several function to explore these results. Among others, **ggResult** and **ggViolin** can be used to plot the temporal distributions of any biomass and/or flux. It is also possible to explore the diet of the species using the function **ggDiet**, to look at the relationships among species with functions **ggPairsBiomass** or **ggTrophicRelation**, or to focus on the biomass variations of specific species with functions **ggGrowth** and **ggSatiation**.

4.3. **Technical considerations.** The development of the packages raises several challenges. The first challenge was to improve computation performance. For this purpose, RCaN is interfaced with **C++** using two R packages **Rcpp** [9] and **RcppEigen** [10]. The second challenge was check the characteristics of the polytope. This was achieved by using standard linear programming library: RCaN uses **lp_solve** through the R package **lpSolveAPI** [11]). A challenge was also to translate the content of a RCaN-file into matrices, and especially to allow user to specify constraints in a simple language. This was made possible by using the symengine R package [12], an interface to the symengine, a fast symbolic manipulation library (the constraint syntax is presented in Supplementary Material).

Finally, the most challenging issue was to achieve the uniform sampling within the polytope given the dimensions of the polytope. To adress this problem, we implement two algorithms. Hit-and-run sampling have been proposed for a long-time to achieve uniform sampling of high-dimensional convex space [13][14]. Basically, hit-and-run sampling starts from a point in the polytope. Then at each iteration, a direction of the high dimensional space is randomly drawn and a new point is uniformly sampled between the two extreme bounds defined by the intersection with the envelop of the polytope. As such, at each iteration, the point is only moved in a random, but single direction, and consequently, when the sampler is "trapped" in a narrow zone of the polytope, it may take many simulation to pick the appropriate direction to exit the narrow zone resulting in autocorrelation. A variation of the hit-and-run sampler is referred as the Gibbs sampler or Coordinate Hit-and-Run [14][15]: instead of picking a direction and updating all coordinates in this direction, the Gibbs sampling picks randomly a dimension (i.e. a parameter of the model) and generate a new parameter value that satisfies the constraints. By doing so, it guarantees an efficient convergence [15]. By default, RCaN uses a Gibbs sample (but hit-and-run can be used), and its tuning is transparent for the user.

5. RCaNconstructor

Main goal of the RCaN constructor is to help the user to manage the RCaN-file. It insures the coherency of the RCaN-file (changing the characteristics of an object somewhere triggers the necessary changes elsewhere).

It allows to run RCaN commands. RCaNconstructor allows to keep a trace of the meta information related to the scientific project of which the modelling is part and of the data files that have provides the observation that constrain the model.

RCaNconstructor is a Java program using Javafx for building the graphical interface and allowing the code to be imported and run on external computers; it uses a Java library named **RCaller** for interfacing R and Java.

5.1. **Meta information.**

5.2. **Views.** RCaNconstructor allows the user to define the objets that are used as inputs of the models. This is done with some dedicated views on the system, its network structure, its components, observations and constraints. According to the principles of RCaN modelling, views on the system are about: (a) the network structure of the trophic system, (b) its components, (c) its fluxes, (d) observations that have been selected on the system, (e) constraints that have been identified about its dynamics.

From all of these the user can add, edit or remove objects. This is saved in a xlsx file which fulfill the requirements of the RCaN RCaN-file.

5.3. **R interface.** The RCaNconstructor is interfaced to R with the RCaller library [16]. From the RCaNconstructor menu, the user can: (a) start a R session and load RCaN library, (b) Build polytope (corresponds to the RCaNMod command of RCaN), (c) sample polytope (corresponds to the fitMyMod command of RCaN), (d) get sampling diagnostics, (e) analyze sample.

6. Example: the Barents Sea food-web

To illustrate CaN modelling using RCaN and RCaNGUI, we use a simplified food-web of the Barents Sea as an example .

In what we present below we have chosen to use the RCaN constructor to build the RCaN-file. And then to run RCaN commands to build the polytope, sample it and analyse the properties of the sampling.

6.1. **Introduction.** Some words about the system. And why to model it with a RCaN approach.

Monitoring of the Barents Ecosystem has been conducted for several decades and a collection of data series are reported regularly and compiled by the ICES working group on the integrated assessment of the Barents Sea (WGIBAR, [17]). These time series cover the period 1988-2013 and include data on:

(1) landings for pelagic and demersal fish, krill, shrimps and marine mammals,
(2) survey-based biomass estimates of zooplankton, pelagic and demersal fish,
(3) satellite based estimate of Net Primary Production (NPP, from 1998),
(4) consumption estimates by Atlantic cod (*Gadus morhua*) of krill, shrimps, capelin, herring, polar cod and Atlantic cod.

In addition, there are estimates of the minimum and maximum plausible biomasses of benthos, marine mammals and birds which are used as limits for the whole time-period in the model.

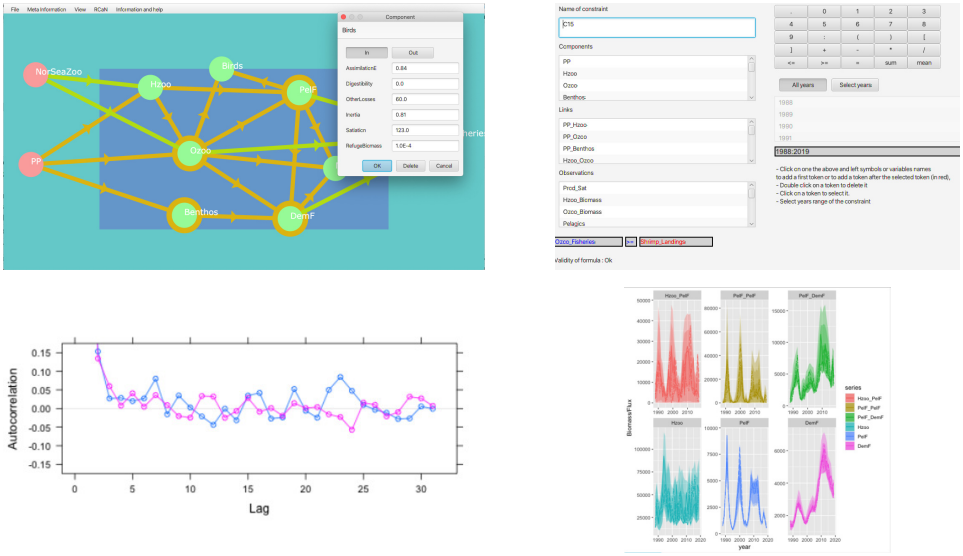

FIGURE 2. Using RCaN and RCaN-constructor to learn about the Barents sea ecosystem. Top left: Using RCaNconstructor to interactively build the trophic network. Top right: using the constructor to interactively define constraints. Bottom left: plotting analysis of sampling : the auto-correlation of a flow. Bottom right: plotting the variations of biomass and flows.

6.2. **Meta Information.** With RCaN constructor (menu Meta Information), give details about the model project (submenu Model project), give the data provenance (submenu Observations).

6.3. **Identifying Network Structure.** After analysis, discussion and synthesis, it appears that the Barents Sea food web is composed of 11 components. Four of them are external (two plankton standing stocks from the adjacent Norwegian Sea, primary producers and Fisheries), and eight are within the model domain, ranging from plankton (Hzoo: herbivorous zooplankton, Ozoo: omnivorous zooplankton), benthos (bent), fishes (PelF: pelagic fishes, DemF: demersal fishes) and top predators (MM: marine mammals, Birds). Compartments are interconnected with 24 biomass fluxes among which 18 correspond to trophic relationships and 6 to imports (plankton) or export (fisheries catches) of biomass.

With RCaN constructor, we graphically design this trophic network (menu Views, submenu Network). The interface (fig. **??**, right) allows to interactiveley add, edit, delete components and fluxes, and to graphically design the network by moving the components.

6.4. **Setting components characteristics.** The input parameters for individual components of the food web are taken from [18]. These were derived from the

Metabolic Theory of Ecology, Life-history theory or previous modelling exercises using for example Ecopath. These are summarised in Table 1, in the supplementary material.

6.5. **Importing observations.** Menu Views. Submenu Observations. Button add an observation.

6.6. **Identifying constraints.** Menu views. Submenu Constraints. Add a constraint (fig. **??**).

The model contains implicit constraints about satiation, inertia, positive flows and refuge biomass. Several additional constraints are defined based on knowledge about the system and available observations. In the Barents Sea case study, these additional constraints are as follows:

(1) The consumption of primary production by zooplankton and benthos is constrained by net primary production estimates (± 30%) for the period when these estimates are available (1998 to 2013). For other years, this sum is bounded by estimates of absolute maximum and minimum primary production that are set to 1/2 million and 2 million tonnes respectively.
(2) Time-series of landings provide accurate information about the fluxes from herbivorous and omnivorous zooplankton, pelagic and demersal fish, and marine mammals to corresponding fisheries. Therefore, these fluxes can be set to the values of the estimated landings.
(3) The annual estimates of biomass for herbivorous and omnivorous zooplankton, and pelagic and demersal fish, derived from research surveys, can be used to constrain the biomass of these groups.
(4) The estimates of minimum and maximum plausible biomasses for benthos, marine mammals and birds can be used as limits for these groups for whole time-period in the model.
(5) The food consumption by demersal fish can be bounded based on consumption estimates by cod in the Barents Sea. These bounds are as follows:
    (a) the consumption of demersal fish by demersal fish is greater than the consumption of cod by cod,
    (b) the consumption of pelagic fish by demersal fish is greater than the consumption of capelin, herring and polar cod by cod;
    (c) the consumption of omnivorous zooplankton by demersal fish is greater than the consumption of krill and shrimp by cod
    (d) the total consumption by demersal fish is not greater than twice the the total consumption estimates by cod.

These above additional constraints are used to couple the dynamics of the food-web to observations/assumptions about the system.

6.7. **Building polytope.** From now we use RCaN. We could do the same with RCaNconstructor.

We build the polytope with following R command :

```
library(RCaN)
myCaNmod <- build\_CaNmod("FILENAME.xlsx")
```

Lets keep in mind that this is a polytope in a 320 dimension space. Getting the solution takes 23 minutes.

We check the non-emptiness and the the boundedness of the polytope, we get the bounds of the polytope in all dimension, we plot the polytope in one dimension with:

```
checkPolytopeStatusCaNmod(myCaNmod) \\
getAllBoundsParamCaNmod(myCaNmod) \\
plotPolytope2DCaNmod(myCaNmod,c("Ozoo[1988]", "HZoo2OZoo[1988]"))
```

6.8. **Building polytope.** With command:

```
begin = Sys.time() \\
res <- fitmyCaNmod(myCaNmod, N=100,thin=100, nchain=2,ncore=2) \\
end=Sys.time()\\
end-begin
```

Getting the solution takes 23 minutes.

6.9. **Checking sampling.** With command:

```
summary(res\$mcmc) \\
gelman.diag(res\$mcmc[,"PP2Bent[2011]"]) \\
acfplot(thinned\_res[,"PP2Bent[2011]"])
```

We get plot in figure 1, bottom left.

6.10. **Graphically analysing sample.** With command:

```
g <- ggResult(res, c("DemF","DemF2Fishery","0Zoo2PelF"), TRUE) \\
library(ggplot2) \\
g + scale\_y\_log10() +guides(color = FALSE, fill = FALSE)
```

1, bottom right.

6.11. **Concluding.** Review initial objectives (RCaNmod, menu Meta Information). Communicate in terms of uncertainty.

7. Discussion

What's next?

References

[1] Benjamin Planque and Christian Mullon. Modelling chance and necessity in natural systems. *ICES Journal of Marine Science*, 2019.

[2] Stephen J Hall and B Mainprize. Towards ecosystem-based fisheries management. *Fish and Fisheries*, 5(1):1–20, 2004.

[3] Elizabeth A Fulton, Anthony DM Smith, David C Smith, and Penelope Johnson. An integrated approach is needed for ecosystem based fisheries management: insights from ecosystem-level management strategy evaluation. *PloS one*, 9(1):e84242, 2014.

[4] Sigrid Lehuta, Raphaël Girardin, Stéphanie Mahévas, Morgane Travers-Trolet, and Youen Vermard. Reconciling complex system models and fisheries advice: Practical examples and leads. *Aquatic Living Resources*, 29(2):208, April 2016. ISSN 0990-7440, 1765-2952. doi: 10.1051/alr/2016022.

[5] É.E. Plagányi. Models for an ecosystem approach to fisheries. Report, Food and Agriculture Organisation of the United Nations, 2007.

[6] J. J. Polovina. An overview of the ecopath model. *Fishbyte*, 2:5–7, 1984.

[7] Johanna Jacomina Heymans, Marta Coll, Jason S. Link, Steven Mackinson, Jeroen Steenbeek, Carl Walters, and Villy Christensen. Best practice in ecopath with ecosim food-web models for ecosystem-based management. *Ecological Modelling*, 331:173–184, 2016. ISSN 0304-3800. doi: http://dx.doi.org/10.1016/j.ecolmodel.2015.12.007. URL `http://www.sciencedirect.com/science/article/pii/S030438001500575X`.

[8] M Plummer, N Best, K Cowles, and K Vines. *Coda: Output Analysis and Diagnostics for MCMC*. 2010.

[9] Dirk Eddelbuettel and James Joseph Balamuta. Extending *R* with *C++*: A Brief Introduction to *Rcpp*. *PeerJ Preprints*, 5:e3188v1, aug 2017. ISSN 2167-9843. doi: 10.7287/peerj.preprints.3188v1. URL `https://doi.org/10.7287/peerj.preprints.3188v1`.

[10] Douglas Bates and Dirk Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. URL `http://www.jstatsoft.org/v52/i05/`.

[11] Kjell Konis and Florian Schwendinger. *lpSolveAPI: R Interface to 'lp$_s$olve' Version 5.5.2.0*, 2020. URL `https://CRAN.R-project.org/package=lpSolveAPI`. R package version 5.5.2.0-17.7.

[12] Jialin Ma, Isuru Fernando, and Xin Chen. *symengine: Interface to the 'SymEngine' Library*, 2020. URL `https://github.com/symengine/symengine.R`. R package version 0.1.0.

[13] Robert L. Smith. Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions. *Operations Research*, 32(6): 1296–1308, December 1984. ISSN 0030-364X. doi: 10.1287/opre.32.6.1296.

[14] Hans C. Andersen and Persi Diaconis. Hit and run as a unifying device. *Journal de la société française de statistique*, 148(4):5–28, 2007.

[15] Aditi Laddha and Santosh Vempala. Convergence of Gibbs Sampling: Coordinate Hit-and-Run Mixes Fast. *arXiv:2009.11338 [cs]*, September 2020.

[16] M Hakan Satman. Rcaller: A software library for calling r from java. *Journal of Advances in Mathematics and Computer Science*, pages 2188–2196, 2014.

[17] ICES. Working group on the integrated assessments of the barents sea (wgibar). *ICES Scientific Reports*, 2(30):212pp, 2020. URL `https://doi.org/10.17895/ices.pub.5998`.

[18] Ulf Lindstrøm, Benjamin Planque, and Sam Subbey. Multiple patterns of food web dynamics revealed by a minimal non-deterministic model. *Ecosystems*, 20 (1):163–182, 2017.

Highlights.

(1) An other way for taking account of uncertainty through the ideas of chance and necessity
(2) Sampling the possible trajectories of trophic system according to (a) first principles and (b) to observations
(3) A dedicated software tool

Keywords. Uncertainty ; R ; Java ; Trophic systems

Software availability

(1) Program title: RCaN
Developer: Hilaire Drouineau
Contact address: hilaire.drouineau@irstea.fr
Software Access: https://www.github.com/...
Year first available: 2020
Hardware required: Windows 7, 10 (tested), Linux, MacOSX
Program language: R
Program size: ...k
Availability and cost: open source
License: LGPL ....

(2) Program title: RCaNGui
Developer: Christian Mullon
Contact address: Christian.Mullon@ird.fr
Software Access: https://.../...
Year first available: 2020
Hardware required: Windows 7, 10 (tested), Linux, MacOSX
Program language: Java
Program size: ...k
Availability and cost: open source
License: LGPL ....

INRAE, Bordeaux
Email address: hilaire.drouineau@inrae.fr

Email address: Benjamin.Planque@hi.no
URL: www.hi.no

Tromsoe, Norway

Paris, France
Email address: christian.mullon@ird.fr
URL: www.ird.fr