2014
réunion annuelle des utilisateurs
15 décembre,
Nantes

Compte-rendu

Stéphanie Mahévas

Les prochaines rencontres des utilisateurs d'ISIS-Fish auront lieu les
10 et 11 décembre 2015 à Nantes

La réunion s'est tenue sur une journée. L'ordre du jour initial était
1) un tour d'horizon des nouveautés techniques de la dernière version d'ISIS-Fish (4.3)
2) un aperçu des avancées de certaines applications
3) questions techniques et méthodologiques
4) projets et développements à venir.
Seuls les points 1), 3) et 4) ont pu être abordés. La journée a été très tournée sur les aspects techniques.

Nouveautés de la version 4.4 d'ISIS-Fish

Prédiction des temps de simulation

Pourrait-on prédire la taille des objets de resultats? oui si on fait tourner sur un pas de temps - Faut-il ajouter cette facilité?

Temps de simulation peut maintenant être spécifié en mois (plutot qu'en années) : reste à bien tester, attention la dernière année qui apparaitra dans les resultats pourra ne correspondre qu'à quelque mois

Détection de R : installer rjava dans R

Dépendance de script : optimisation de l'embarquement du code

Nouvel objet observations dans la base de Region
pour le stockages des observations : nb ligne = produit des dimensions à partir d'un fichier (équivalent du format dataframe de R).

Amélioration des performances
La plus part des matrices manipulées dans ISIS sont très creuses (« sparse ») (90% des valeurs nulles), donc changement de mode de stockage (on ne stocke que les valeurs non nulles) et de mode de calcul (multiplications de valeurs nulles non faites). Une librairie a été spécifiquement développée à cet effet. Les méthodes qui s'appliquent sur les matrices creuses peuvent être un peu plus longues si la matrice initiale n'était pas vraiment creuse. Il existe donc un seuillage à 1000: si on a plus de 1000 valeurs nulles alors on passe en matrice creuse. Le seuillage est un paramètre de configuration réglable (dans l'interface « simulation.matrix.threshold.use.sparse.class»).

Cache mémoire : amélioration du stockage dans le cache. Il existe un paramètre qui fixe le nombre de pas de temps pendant lequel on conserve les résultats (par défaut 13 mais réglable dans la configuration « store.result.cache.step »).

Simatrix et gravity - fallait-il garder une écriture matricielle ou fonctionnelle des équations? L'avantage des matrices : plus adaptée pour faire des optimisations. Donc avec le passage en matrice creuse, la conclusion a été de conserver le mode matriciel. Toutes les écritures fonctionnelles ont été passées en matricielles (surtout pour les calculs économiques et à la marge dans le reste)

Recherche de code non performant : bien vérifier les temps associés aux règles car les temps longs sont souvent liés aux règles des utilisateurs. Temps de calcul des règles, des scripts et des stockages dans le résumé de la simulation

Structure des résultats : pour tous les résultats sélectionnes création d'un pseudo fichier csv compressé (pseudo : meta informations en entête du fichier - nom de la matrice, dimensions) à la place des fichiers mapped : pour chaque résultat un fichier par pas de temps. Ce changement a été fait car sous windows les fichiers mappés posaient parfois des problèmes. En plus c'est plus simple et plus lisible. Maintenant ce type de fichier est nécessaire pour faire les exports. Attention : il n'existe pas de vérification de la sélection du résultat nécessaire à un export. En revanche ces résultats sont consultables dans l'interface de résultats, même si la database (région) n'existe plus. Un exemple de code R est disponible en annexe pour illustrer l'extraction facile des valeurs de ces fichiers résultats.

Configuration des simulations - précisions sur les variables disponibles:
max.threads = nombre simu à lancer en même temps
in.process = on ne ferme pas le processus (pb si fuite de memoire)
sub.process = nombre de coeurs de lancement de simu (= nb de simus simultanées), par defaut le nb de coeurs (-1)
sub.max.memory : valeur mémoire allouée pour un processus. Attention : max.memory * sub.process < RAM dispo
class java des matrices creuses : on a la possibilité de passer en double ou en float (2 fois moins de place) mais il est recommandé de toujours prendre double pour éviter les erreurs d'arrondi.
Choix du seuil à partir duquel on passe de matrice dense à creuse : 1000
Flag Lazy : pour éviter qu'une copie de matrice soit effectivement copiée (on pointe sur la matrice)

store.result.ondisk = nb de pas de temps de sauvegardes - par défaut tous. Attention les exports de résultats ont lieu en fin de simulation, l'export ne contient donc que les pas de temps sauvegardés.
store.result.cachestep = nb de pas de temps de sauvegarde des résultats dans le cache (par exemple pour les regles, le recrutement)

Optimisation : onglet dans lancement de simulation

selection d'un script d'optimisation : nouveau type de methodes
selection d'un script de fonction d'objectif : qui a besoin du context, des exports et des observations (entrées dans l'interface)
La structure du code d'optimisation (séquence) :
first : creation et lancement des premières simulation
end : caldul sur les premieres simu
next : creation et lancements des simu suivants (pas forcement le meme nombre). Il contient la condition pour la generation suivante
end : idem. Exécuté après chaque génération
next: idem
end :idem
next = ensemble vide - on s'arrête
finish
Exemple de méthodes associées à l'objet context d'optimisation (cf API):
newSimulation(), getGeneration(n), getParam(i = index de la simu ds la generation).
Peut-on ajouter une méthode tableau qui créerait une table de stockage des simus avec la valeur fonction d'objectif, valeurs des paramètres et autres paramètres d'intérêt?
OUI - dans le init, l'utilisateur devra y spécifier les noms de colonnes
Il faut aussi pouvoir sélectionner plusieurs fonctions d'objectifs qui seraient stockées dans la table, utilisée pour faire l'optimisation. La fonction d'objectif peut rendre une table de doubles.
illustration de mise en oeuvre : ex Audric (code, historique,..., code R d'exemples) - présentationNB : une librairie R de scripts d'optimisation (surtout algo.gen.) couplables à tout modèle a été développée. Son potentiel sera à explorer.

Script R d'analyse des résultats : 1 journée de travail sur les scripts R pour travailler sur les sorties d'ISIS (Yves, Sigrid, Loic, Steph, Audric, Mathieu) : fev-mars 2015 - **Sigrid se charge de faire le doodle**

Gestion de l'aléatoire : 2 configurations

- via R en utilisation de les distributions disponibles mais suppose d'avoir une instance R d'ouverte (c'est tres couteux) ou appel à la fonction avant les simulations et sauvegarde dans un fichier en pensant à bien sauvegarder la graine
- en utilisant les librairies java (ssj)

illustration par Sigrid d'une relation stock-recrutement stochastique (dans l'interface reproduction) - présentation

Discussions sur la méthode de création un nouveau simulateur (simulatorPerso)

- creer une methode dans defaultSimulator et surcharger la methode dans simulatorPerso
illustration avec l'exemple avec Marxan (Yves)

actuellement on crée des tagvalues dans le simulatorPerso
Yves a été oblige de modifier des methodes de SiMatrix.
Pour eviter d'avoir des scripts qui ont le meme nom que dans la boite officielle et qui sont differents de l'officiel, il faudrait pouvoir creer un SiMatrixPerso qui heriterait de siMatrix comme pour DefaultSimulator et PersoSimulator.

Changement de DefaultSimulator

modification de la séquence des processus dans simulateur : bascule de boite grise avant la boite rose

GetRecruitment : migrations et changement de groupe s'opéraient sur les oeufs (groupe nul). Il est nécessaire de distinguer deux configurations : description d'une reproduction ou d'une relation stock-recrutement**.**

- **groupe 0 : groupe (age ou longueur) dans lequel apparaisse les individus qui peuvent subir de la mortalité par pêche, des migrations, des changements de classe (c'est à dire normalement le recrutement)**
- **on distingue deux équations : ponte (reproduction) et le recrutement**
- **ponte : équation appelée à chaque mois de la saison de reproduction, elle met à disposition l'abondance à t et retourne le nombre d'oeufs (matrice zone*temps). A chaque pas de temps entre l'instant de ponte et le recrutement, cette matrice subit uniqment la mortalité naturelle du groupe null. Au moment du recrutement elle subira une migration zoneRepro-zoneRecru.**
- **recrutement : équation appelée à chaque pas de temps de la saison recrutement (définie par saison de repro, etalement recru et delta), les recrues sont récupérées dans la matrice de ponte par défaut. Il est possible d'écrire une relation stock-recrutement différente (constante, fonction d'autres parametres ou variables). Dans cette équation on accède aussi à l'abondance aux instants de la reproduction précédente (t-delta).**

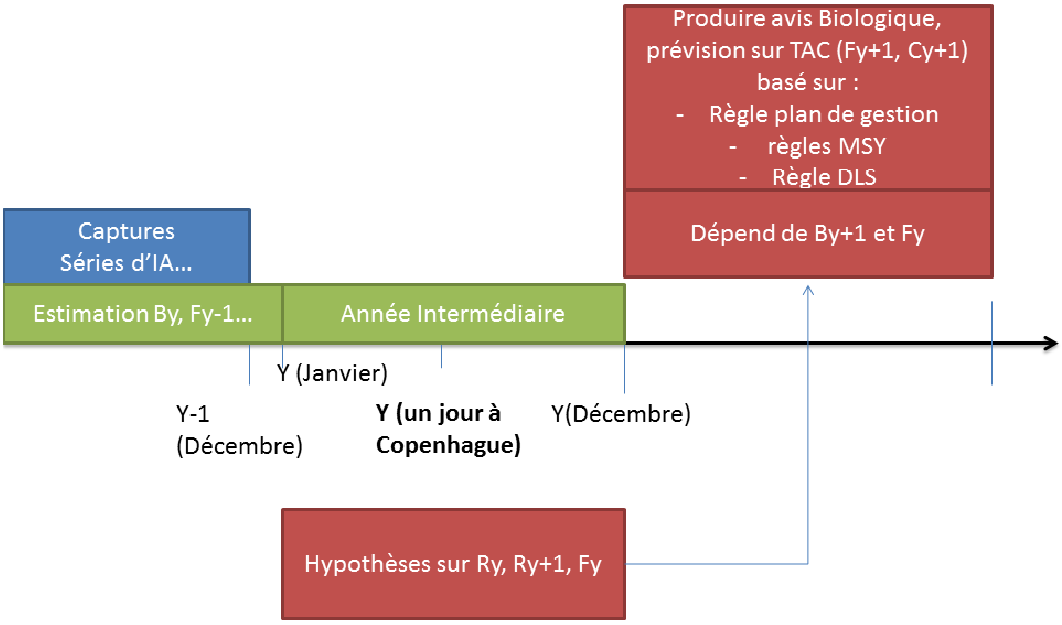
MSE

- ### Couplage ISIS avec XSA via R pour faire une MSE
  1. sorties d'ISIS (captures aux ages, CPUE d'une flottille, mortalité naturelle aux ages,...)
  2. dans R : alimentation des objets FLR
  3. sortie de FLXSA : F, SSB
  4. dans ISIS : calcul de la HCR et definition du TAC

regle ISIS : xsa.java (bientôt disponible commentée)

- ### Approche HCR sans Evaluation de stock

prise en compte de la vraie procedure appliquee au CIEM (decalage de deux ans sur les observations + estimations du recrutement intermediaire et suivante (hypo) et du F intermediaire (a partir du TAC)



- **SS3 - ISIS-fish (thèse Audric)**

SS3 : dyn de pop stochastique (erreur d'observation et de processus) - optimisation d'une fonction de vraisemblance (ADMB) pour estimer les variables et les paramètres - structure en age/taille
coupler les deux outils : utilitaire de lancement de programmes externes dans ISIS - utiliser R ou développer un bout de code Java pour encapsuler les méthodes nécessaires au lancement de l'executable. Il est preferable d'utiliser R - exemple de code fait par Yves pour le couplage ISIS et Marxan

Rapprochement avec les professionnels
cas d'étude golfe de Gascogne : prospective par scenarisation (travail de Laurie, geoprospective, approche participative via les cartes)
cas d'etude Manche : discussion sur une connaissance commune via les cartes - comparaison des connaissances - etude du rouget

Projets :

- **en cours**

Européen : MYFish, Socioec (fev2015), Vectors (Janv2015)
Region : Coselmar, Panache (printemps2015)

- **confirmé à venir**

Discardless : cas d'etude Mediterranée (these de Sandrine), cas d'etude de Manche (0.5 mois/an) - tester l'effet de stratégie de limitation des rejets via des approches spatialisées

- **en cours de demande**

projet methodologique : approche multi-modèles (ANR demande en cours)
interreg : appli cocochannel et prolonger le couplage ISIS-Marxan
BG1 : application Channel (coordination Paul)

Budget :

- fonctionnement minimal annuel requis 20keuros, pouvant aller jusqu'à 50keuros pour du nouveau developpemnt
- pas de budget validé à l'eprd 2015....

Annexe :
Exemple de code R pour lire les fichiers resultats pseudo csv et recoller les pas de temps:

```
resultName = "matrixBiomass Sole"
colnames = c("group", "zone","val")
sk = length(colnames)+1

ssb <- function(fileL,resultName){
        id = as.numeric(strsplit(fileL,"_")[[1]][3])
        path3 <- paste(simudir,fileL,"/results/matrix/",resultName,"/",sep="")
        tabs <- data.frame()
        for(s in seq(0,95,12)){ # boucle sur certains pas de temps
                fileY = paste(path3,s,"-",resultName,".csv.gz",sep="")
                tab <- read.table(file = fileY, skip=sk, sep=";", col.names=colnames)
                tabs <- as.data.frame(rbind(tabs,c(s,sum(tab$val),id)
        }
        names(tabs)=c("step","val","id")
        return(tabs)
}
```

Statut des objets JAVA :
Rien : visi package
Private : visible que ds la class ou elle est def (jamais)
Protected : visible ds la class et les enfants de la class
Public : visible partout
Final : tu peux plus modifier la valeur
Static : accessible meme si tu n as pas une instance de la class (pas utiliser)
Ex rule :
param_pop si pas public pas affichable ds l interface !
pour les methodes utiliser private ou protected

Point **Partage de scripts :**
- Audric : recuit simulé, codes traitement sorties
- Yves : routines R chargement d'export a genericiser, routine R pour créer une grille dont les cellules ont les memes noms que les cellules ISIS
- Loic : Script R cartes a partir des donnees de derive larvaire, (tt dans le officiel), exports d'AS par zones

- Sigrid : calibration ad-hoc, forcage effort, Tac et MLS avec obligation à debarquer, remplissage des parametres de pêche via une règle…