Introduction

Modèle recordschool de la plateforme RECORD : L'école-chercheurs « Approches interdisciplinaires de la modélisation des agroécosystèmes » organisée par l'INRA en mars 2017 à Sainte Foy Lès Lyon s'appuie sur un modèle de la plateforme RECORD. Ce modèle correspond au dépôt de modèles **recordschool** des services web erecord.

Programme R appelant le modèle recordschool en utilisant RVLE : Cette école-chercheurs comporte en « Séquence 8.3 » un « Hackaton 3 » d'exploration de modèles, écrit en langage de programmation R. Pour appeler le modèle recordschool, ce hackaton utilise le paquet RVLE qui permet le lien entre R et la plateforme RECORD. Dans ce cas, la suite logicielle sur laquelle est basée la plateforme RECORD (VLE et RVLE) ainsi que le modèle recordschool lui-même doivent avoir été installés sur le poste de travail où est lancée la session R.

Programme R appelant le modèle recordschool en utilisant erecord : Etant donné que le modèle recordschool est disponible sous les services web erecord, il est possible d'appeler le modèle recordschool sous forme de services web erecord, sans avoir dans ce cas à installer quoi que ce soit sur le poste de travail où est lancée la session R, si ce n'est les librairies R qui permettront d'écrire sous R les appels aux services web erecord (librairies 'RCurl' et 'rjson').

Programmes R d'exploration de modèles dans le cas du modèle recordschool

Ce document reprend le code intégral de certaines sessions R du « Hackaton 3 », dans lequel les traitements utilisant RVLE ont été colorés en orange et mis en commentaires pour être remplacés par des traitements utilisant erecord colorés en vert.

Généralités

Identification du simulateur sous erecord

Le simulateur traité dans ce qui suit est le fichier **record_school.vpz** du paquet **Hackaton_3**. Dans les services web erecord il a pour identifiant **Id=832**. C'est cet identifiant qui sert à le désigner dans les requêtes erecord à son sujet. Pour savoir comment le trouver : voir « Connaître l'identifiant d'un simulateur ».

Valeurs par défaut d'un simulateur

Les services web erecord proposent une requête permettant d'obtenir les conditions de simulation d'un simulateur : valeurs par défaut de paramétrage, début et durée de simulation. Il s'agit de la ressource *vpz/input* . Exemple : voir « Connaître les valeurs par défaut d'un simulateur ».

Modification et simulation d'un simulateur

Les services web erecord proposent une requête permettant de modifier un simulateur (son paramétrage…) avant d'en lancer la simulation et d'en obtenir les résultats de simulation. Il s'agit de la ressource *vpz/output*. Elle est mobilisée ci-dessous : voir « Simulation d'un LHS », « Optimisation multi-critère avec nsga2 ».

Simulation d'un LHS

simus_LHS.R

code orange : en utilisant rvle (modèle local) – code vert : en utilisant erecord (modèle distant)

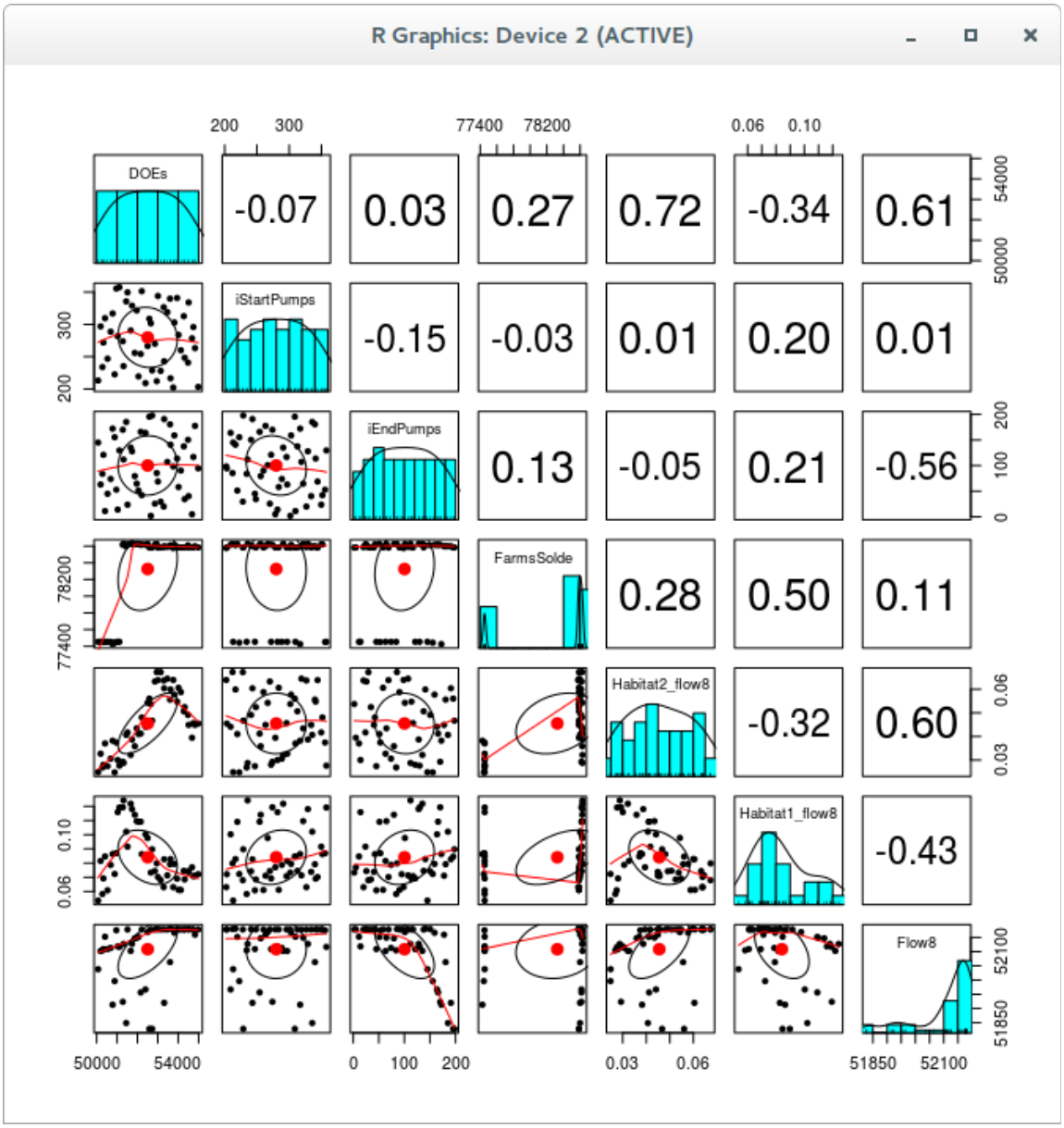
```
##
## chargement des libraries nécessaires
##

#library(rvle) #simulation du modèle fil rouge
library('RCurl')
library('rjson')
library(lhs) #pour générer un plan LHS

##
## définition des paramètres
##
factors = c("cond_DE_gestionnaire.DOE", "cond_DE_gestionnaire.DCR",
            "cond_DE_gestionnaire.PumpRate", "cond_DE_gestionnaire.iStartPump",
            "cond_DE_gestionnaire.iEndPump")
bounds = data.frame(min=c(50000,30000,4000,200,1), max=c(55000,40000,6000,360,200));
rownames(bounds) <- factors;

##
## définition des variables de sorties du modèle
##
variables =  c("Farm1Solde","Farm2Solde","Farm3Solde","Farm4Solde","Farm5Solde",
               "Flow8", "Habitat2_flow8", "Farm2_NbBirds");

##
## Lecture du modèle et configuration pour la simulation
##
#f = new("Rvle", pkg="Hackaton_3", file="record_school.vpz");
#f = setDefault(f, outputplugin = c(Eco="vle.output/storage"));
#f = setDefault(f, outputplugin = c(optim="vle.output/storage"));
#f = setDefault(f, plan="linear", proc="thread", thread=3);
#f = setDefault(f, duration=1000);

##
## Génération d'un plan LHS pour 3 facteurs (DOE, iStartPump, iEndPump)
##
nsimus = 50;
LHS = optimumLHS(nsimus, 3)

##
```

```
## Recentrer les valeurs du LHS (entre 0 et 1) dans le domaine de variation des 3 facteurs
##
DOEs = LHS[,1]*(bounds["cond_DE_gestionnaire.DOE", "max"] -
bounds["cond_DE_gestionnaire.DOE", "min"]) +  bounds["cond_DE_gestionnaire.DOE", "min"];
iStartPumps = LHS[,2]*(bounds["cond_DE_gestionnaire.iStartPump", "max"] -
bounds["cond_DE_gestionnaire.iStartPump", "min"]) +
bounds["cond_DE_gestionnaire.iStartPump", "min"];
iEndPumps = LHS[,3]*(bounds["cond_DE_gestionnaire.iEndPump", "max"] -
bounds["cond_DE_gestionnaire.iEndPump", "min"]) +
bounds["cond_DE_gestionnaire.iEndPump", "min"];

##
## Configuration pour la simulation du plan entier en une fois
##
#f = setDefault(f, cond_DE_gestionnaire.DOE=DOEs);
#f = setDefault(f, cond_DE_gestionnaire.iStartPump=iStartPumps);
#f = setDefault(f, cond_DE_gestionnaire.iEndPump=iEndPumps);

#r = results(run(f))

header = c('Content-Type'='application/json', Accept='application/json')
vpz_output_url="http://erecord.toulouse.inra.fr:8000/vpz/output/"
vpz_id = 832 # pkg="Hackaton_3" file="record_school.vpz"
postfields = toJSON(list(vpz=vpz_id, duration=1000,
                         cond_DE_gestionnaire.DOE=DOEs,
                         cond_DE_gestionnaire.iStartPump=iStartPumps,
                         cond_DE_gestionnaire.iEndPump=iEndPumps,
                         outselect=c("Eco","optim"),
                         plan="linear", restype="dataframe", style="tree", format="json"))
res = postForm(uri=vpz_output_url, .opts=list(postfields=postfields, httpheader=header))
responsedata = fromJSON(res)
r = fromJSON(responsedata$res)

##
## Récupération de la moyenne temporelle pour chacune des simulations
## du solde de la ferme 1
##
Farm1Solde = unlist(lapply(r, function(simu) {
 mean(simu$Eco$"Top model,Farms,Farm1:EcoIndicatorsObserver.Solde")
}))

##
## Récupération de la moyenne temporelle pour chacune des simulations
## du solde de la ferme 2
##
Farm2Solde = unlist(lapply(r, function(simu) {
  mean(simu$Eco$"Top model,Farms,Farm2:EcoIndicatorsObserver.Solde")
```

```
}))

##
## Récupération de la moyenne temporelle pour chacune des simulations
## du solde de la ferme 5
##
Farm5Solde = unlist(lapply(r, function(simu) {
  mean(simu$Eco$"Top model,Farms,Farm5:EcoIndicatorsObserver.Solde")
}))

FarmsSolde = Farm1Solde+Farm2Solde+Farm5Solde;

##
## Récupération de la moyenne temporelle pour chacune des simulations
## du débit à l'exutoire
##
Flow8 = unlist(lapply(r, function(simu) {
  mean(simu$optim$"Top model:River.Flow8")
}))

##
## Récupération de la moyenne temporelle pour chacune des simulations
## de la qualité de l'habitat des poissons (espèce 1)
##
Habitat1_flow8 = unlist(lapply(r, function(simu) {
  mean(simu$optim$"Top model,FishHabitatQuality:FishHabitatQuality_8.Habitat1")
}))

##
## Récupération de la moyenne temporelle pour chacune des simulations
## de la qualité de l'habitat des poissons (espèce 2)
##
Habitat2_flow8 = unlist(lapply(r, function(simu) {
  mean(simu$optim$"Top model,FishHabitatQuality:FishHabitatQuality_8.Habitat2")
}))

save.image(file="rdata/simus_LHS.rdata")

#plot
load(file="rdata/simus_LHS.rdata")
library(psych)
pairs.panels(cbind(DOEs, iStartPumps, iEndPumps, FarmsSolde,
                   Habitat2_flow8, Habitat1_flow8, Flow8), method="spearman")
```

Résultat en utilisant erecord (modèle distant)

simus_LHS.rdata



Optimisation multi-critère avec nsga2

nsga2.R

code orange : en utilisant rvle (modèle local) – code vert : en utilisant erecord (modèle distant)

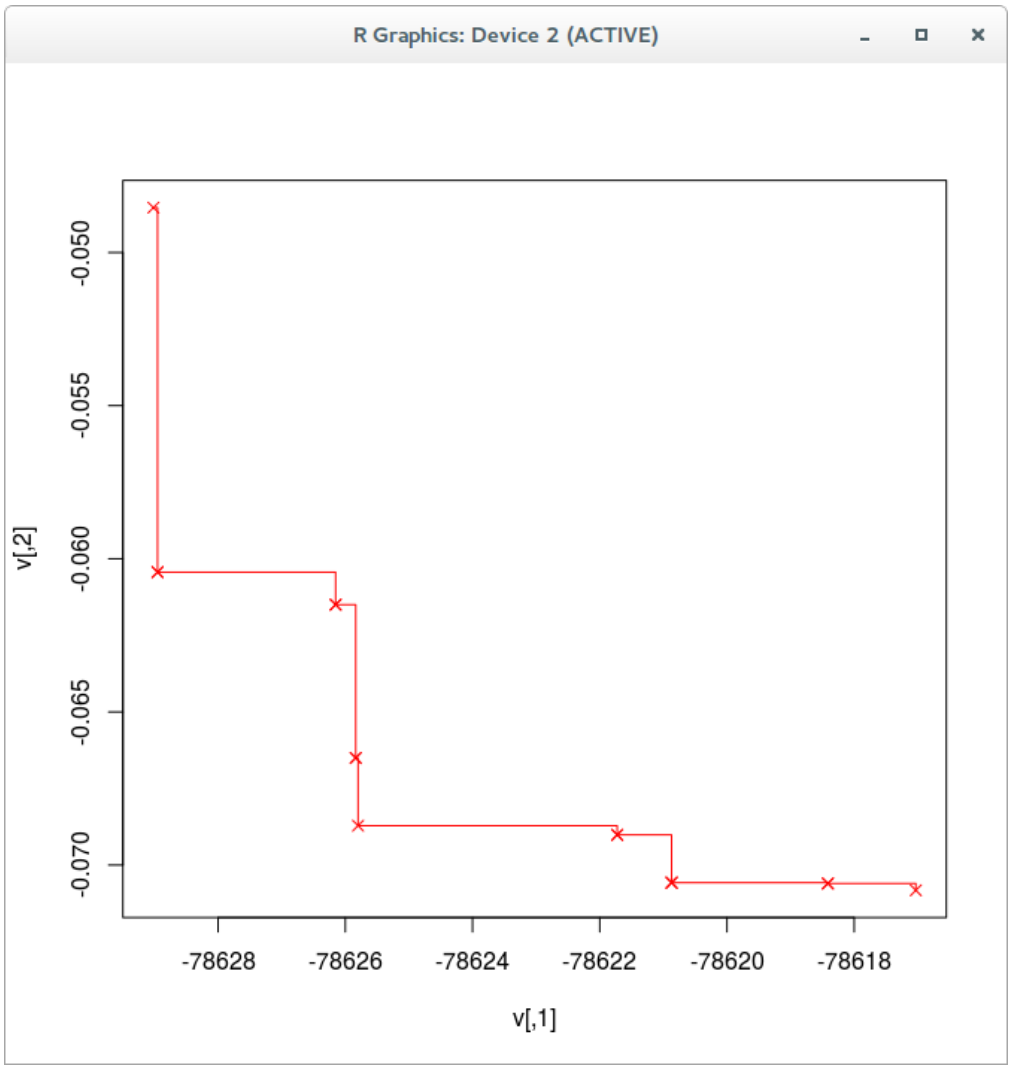
```
##
## chargement des libraries nécessaires
##
#library(rvle) #simulation du modèle fil rouge
library('RCurl')
library('rjson')
library(mco)  #pour utiliser l'optimiseur nsga2

##
## définition des paramètres
##
factors = c("cond_DE_gestionnaire.DOE", "cond_DE_gestionnaire.DCR",
            "cond_DE_gestionnaire.PumpRate", "cond_DE_gestionnaire.iStartPump",
            "cond_DE_gestionnaire.iEndPump")
bounds = data.frame(min=c(50000,30000,4000,200,1), max=c(55000,40000,6000,360,200));
rownames(bounds) <- factors;

##
## définition des variables de sorties du modèle
##
variables =  c("Farm1Solde","Farm2Solde","Farm3Solde","Farm4Solde","Farm5Solde",
               "Flow8", "Habitat2_flow8", "Farm2_NbBirds");

##
## chargement du modèle et configuration pour la simulation
##
#f = new("Rvle", pkg="Hackaton_3", file="record_school.vpz")
#f = setDefault(f, outputplugin = c(Eco="vle.output/storage"));
#f = setDefault(f, outputplugin = c(optim="vle.output/storage"));
#f = setDefault(f, plan="linear", proc="thread", thread=6);
#f = setDefault(f, duration=1000);

header = c('Content-Type'='application/json', Accept='application/json')
vpz_output_url="http://erecord.toulouse.inra.fr:8000/vpz/output/"
vpz_id = 832 # pkg="Hackaton_3" file="record_school.vpz"

##
## définition d'une fonction qui prend en entrées un plan d'expérience
## en entrée: X matrice de n lignes et 3 colonnes : DOE, iStartPump, iEndPump
## en sortie: une matrice de n simus des variables FarmsSolde et Habitat2_flow8
## Note: comme nsga2 fait de la minimisation et que l'on souhaite maximiser
```

```
## on met les critères en négatif.
##
fnsga2 = function(X)
{
   str(X)
   #f = setDefault(f,  cond_DE_gestionnaire.DOE=X[,1]);
   #f = setDefault(f, cond_DE_gestionnaire.iStartPump=X[,2]);
   #f = setDefault(f, cond_DE_gestionnaire.iEndPump=X[,3]);
   #r = results(run(f));
  postfields = toJSON(list(vpz=vpz_id, duration=1000,
                    cond_DE_gestionnaire.DOE=X[,1],
                    cond_DE_gestionnaire.iStartPump=X[,2],
                    cond_DE_gestionnaire.iEndPump=X[,3],
                    outselect=c("Eco","optim"),
                    plan="linear", restype="dataframe", style="tree", format="json"))
   res = postForm(uri=vpz_output_url, .opts=list(postfields=postfields, httpheader=header))
   responsedata = fromJSON(res)
   r = fromJSON(responsedata$res)

  Farm1Solde = unlist(lapply(r, function(simu) {
    mean(simu$Eco$"Top model,Farms,Farm1:EcoIndicatorsObserver.Solde")
  }))

  Farm2Solde = unlist(lapply(r, function(simu) {
    mean(simu$Eco$"Top model,Farms,Farm2:EcoIndicatorsObserver.Solde")
  }))

  Farm5Solde = unlist(lapply(r, function(simu) {
    mean(simu$Eco$"Top model,Farms,Farm5:EcoIndicatorsObserver.Solde")
  }))

  FarmsSolde = Farm1Solde+Farm2Solde+Farm5Solde;

  Habitat2_flow8 = unlist(lapply(r, function(simu) {
    mean(simu$optim$"Top model,FishHabitatQuality:FishHabitatQuality_8.Habitat2")
  }))
  str(rbind(-FarmsSolde, -Habitat2_flow8))
  return(rbind(-FarmsSolde, -Habitat2_flow8));
}

#
#Appel de l'optimiseur
#
resnsga2 =  nsga2(fnsga2, idim=3, odim=2,
           lower.bounds = bounds[c("cond_DE_gestionnaire.DOE",
"cond_DE_gestionnaire.iStartPump", "cond_DE_gestionnaire.iEndPump"),]$min,
           upper.bounds = bounds[c("cond_DE_gestionnaire.DOE",
```

```
"cond_DE_gestionnaire.iStartPump", "cond_DE_gestionnaire.iEndPump"),]$max,
          popsize = 16, generations = 50,
          vectorized=TRUE)

save.image(file="rdata/nsga2.rdata")

#plot
load(file="rdata/nsga2.rdata")
library(mco)
plot(resnsga2)
```

Résultat en utilisant erecord (modèle distant)

nsga2.rdata



Connaître l'identifiant d'un simulateur

Organisation des modèles sous erecord

Les modèles contiennent chacun un certain nombre de simulateurs et sont rangés sous erecord dans des dépôts de modèles. Les services web erecord proposent des requêtes permettant de connaître pour chaque simulateur, modèle, dépôt de modèles, son identifiant qui permet de le désigner dans les requêtes erecord à son sujet. Il s'agit des ressources db/vpz, db/pkg, db/rep…

Le modèle recordschool sous erecord

Sous erecord, deux dépôts de modèles concernent l'école-chercheurs « Approches interdisciplinaires de la modélisation des agroécosystèmes » :

- Le dépôt de modèles **recordschool** contient le modèle recordschool lui-même.

- Le dépôt de modèles **recordschool_tp** contient des suppléments utilisés dans les exercices de l'école-chercheurs.

Le simulateur record_school.vpz du paquet Hackaton_3

Le simulateur auquel on s'intéresse est le simulateur **record_school.vpz** du paquet **Hackaton_3**. Il fait partie du dépôt de modèles **recordschool_tp**. Voici différents moyens de trouver son Id.

Trouver l'Id du simulateur au moyen de l'interface utilisateur erecord

En préalable à l'écriture d'un programme R, on peut chercher sous l'interface web utilisateur erecord http://erecord.toulouse.inra.fr:8000/home l'Id du simulateur qui nous intéresse. Celle-ci donne en page d'accueil la liste des dépôts de modèles, celle des modèles et celle des simulateurs.

En sélectionnant le « models repository » **recordschool_tp** (**Id 10**), puis son « model » **Hackaton_3** (**Id 207**), puis son « simulator » **record_school.vpz** (**Id 832**), on trouve que le simulateur **record_school.vpz** du paquet **Hackaton_3** a pour **Id=832**.

Programme R de recherche de l'Id du simulateur

Le simulateur auquel on s'intéresse est le simulateur **record_school.vpz** du paquet **Hackaton_3**. En supposant que l'on sache qu'il fait partie du dépôt de modèles **recordschool_tp**, on peut :

- rechercher l'Id **rep_id** du « models repository » dont name="**recordschool_tp**"
- puis dans celui-ci rechercher l'Id **pkg_id** du « model » dont name="**Hackaton_3**"
- puis dans celui-ci rechercher l'Id **vpz_id** du « simulator » dont name="**record_school.vpz**"

code.R

```
# chargement des libraries nécessaires aux requetes erecord
library('RCurl')
library('rjson')

# default values
rep_id = NULL; pkg_id = NULL; vpz_id = NULL;

# get all the « models repositories » in 'tree' style
res = getForm("http://erecord.toulouse.inra.fr:8000/db/rep/", mode='tree')
rep_list = fromJSON(res)

for (rep in rep_list){
   if (rep$name == "recordschool_tp" ){

      rep_id= rep$id

      pkg_list = rep$vlepkg_list
      for (pkg in pkg_list){
         if (pkg$name == "Hackaton_3" ){

            pkg_id = pkg$id

            # get all the « simulators » of the « model » with id=pkg_id in 'tree' style
            res = getForm("http://erecord.toulouse.inra.fr:8000/db/vpz/", mode='tree', pkg=pkg_id)
            vpz_list = fromJSON(res)

            for (vpz in vpz_list){
               if (vpz$name == "record_school.vpz"){
                  vpz_id = vpz$id
               }
            }
         }
      }
   }
}

rep_id; pkg_id; vpz_id;
```

Autre programme R de recherche de l'Id du simulateur

Le simulateur auquel on s'intéresse est le simulateur **record_school.vpz** du paquet **Hackaton_3**. En supposant que l'on ne sache pas à quel dépôt de modèles il appartient, on peut rechercher l'Id **pkg_id** du « model » dont name="**Hackaton_3**" et ayant un « simulator » dont name="**record_school.vpz**". On en déduit l'Id **vpz_id** de ce « simulator », ainsi que l'Id **rep_id** et le nom **repository_name** du « models repository » auquel ils appartiennent.

code.R

```
# chargement des libraries nécessaires
library('RCurl')
library('rjson')

# default
rep_id = NULL; pkg_id = NULL; vpz_id = NULL; repository_name = NULL;

# get all the « models » in 'tree' style
res = getForm("http://erecord.toulouse.inra.fr:8000/db/pkg/", mode='tree')
pkg_list = fromJSON(res)

for (pkg in pkg_list){
   if (pkg$name == "Hackaton_3" ){
      pkg_id_candidat = pkg$id
      for (vpz in pkg$vlevpz_list){
         if (vpz$name == "record_school.vpz"){
            vpz_id = vpz$id
            pkg_id = pkg_id_candidat
            rep_id = pkg$vlerep$id
            repository_name = pkg$vlerep$name
         }
      }
   }
}

vpz_id; pkg_id;  rep_id; repository_name;
```

Remarque

Cette recherche rapide identifie un simulateur **record_school.vpz** contenu dans un paquet **Hackaton_3**, mais ne garantit pas que ce soit le seul qui soit présent sous erecord et donc que ce soit bien celui recherché. En effet différentes versions d'un modèle, ou d'un simulateur peuvent coexister sous erecord dans différents dépôts de modèles. Par exemple, d'une part un simulateur record_school.vpz existe dans le modèle RS_FilRouge du dépôt de modèles recordschool et d'autre part il en existe un autre dans le modèle Hackaton_3 du dépôt de modèles recordschool_tp. La seule façon de trancher est de se renseigner auprès des personnes de erecord ou propriétaires du modèle.

Connaître les valeurs par défaut d'un simulateur

code.R

```
# chargement des libraries nécessaires
library('RCurl')
library('rjson')

header = c('Content-Type'='application/json', Accept='application/json');

# ou bien sans filtrage (retourne tous les paramètres)
postfields = toJSON(list(vpz=832, # pkg="Hackaton_3" file="record_school.vpz"
                         style="compactlist", format="json"))

# ou bien avec option 'parselect' de filtrage de paramètres ou de conditions
postfields = toJSON(list(vpz=832, # pkg="Hackaton_3" file="record_school.vpz"
                         parselect=c("cond_CropIDs.CropIDs", "cond_custom_gestionnaire",
                                     "cond_DE_LeafIndexArea.fTsum2",
                                     "cond_DE_LeafIndexArea.fTsum1",
                                     "cond_DE_LeafIndexArea.fLAI1",
                                     "cond_DE_LeafIndexArea.fLAI2"),
                         style="compactlist", format="json"))

rep = postForm(uri="http://erecord.toulouse.inra.fr:8000/vpz/input/",
               .opts=list(postfields=postfields, httpheader=header))
responsedata = fromJSON(rep)
responsedata

fTsum1= fromJSON(responsedata$cond_DE_LeafIndexArea.fTsum1)
fTsum1

CropIDs = fromJSON(responsedata$cond_CropIDs.CropIDs)
CropIDs[[1]]

xban1 = fromJSON(responsedata$ cond_custom_gestionnaire.xban1)
xban1[[1]]
```