



# OpenFLUID

Software Environment  
for Spatial Modelling in Landscapes

## Package ROpenFLUID pour R Présentation & exemples d'utilisation

Equipe OpenFLUID

*LISAH - Laboratoire d'étude des Interactions Sol-Agrosystème-Hydrosystème*



This document is licensed  
under Creative Commons license



# Plan

- 1 Introduction
- 2 Bases et fonctionnement
- 3 Exemples d'utilisation

# L'environnement et le langage R

R est un logiciel libre de **traitement de données et d'analyse statistique**:

- évolution du langage S (1975)
- **environnement statistique** et **langage de programmation**
- programmation fonctionnelle, procédurale ou orientée objet
- structures de **données évoluées**: vecteurs, matrices, tableaux, data frames, listes



<http://www.r-project.org>

Utilisation en ligne de commande ou interface graphique (RStudio, RKWard, plugins Eclipse ou Emacs,...)

# L'environnement et le langage R

Organisation générale :

- **un environnement unique**, intégrant le langage et la visualisation graphique
- **une base** pour la statistique courante
- **des paquets** (ou **extensions**) permettant d'étendre les fonctionnalités de base

Catégories de paquets (liste non exhaustive)

- statistiques
- fonctions mathématiques
- import/export de données
- **interfaçage avec des outils externes**
- ...

# ROpenFLUID

**ROpenFLUID** est un paquet R permettant le **pilotage de simulations OpenFLUID**:

- la définition et la modification de paramètres
- l'exécution de simulations
- l'utilisation des résultats de simulation sous la forme de données R

ROpenFLUID ajoute dans R des **commandes propres à OpenFLUID**, sous la forme de fonctions R préfixées par "OpenFLUID."

Les commandes ROpenFLUID peuvent **interagir avec d'autres commandes R** ou des commandes fournies par d'autres paquets.

ROpenFLUID utilise les mêmes jeux de données qu'OpenFLUID (format FluidX)



# Installation et chargement du paquet

Le paquet ROpenFLUID est à télécharger puis à installer dans R

## Installation du paquet ROpenFLUID

```
install.packages("ROpenFLUID_2.1.8-20190424.tar.gz", repos=NULL)
```

Le paquet ROpenFLUID doit être chargé dans la session R pour utiliser les commandes OpenFLUID

## Chargement du paquet ROpenFLUID

```
library("ROpenFLUID")
```

# Lancement d'une simulation

Etapes de lancement d'une simulation avec ROpenFLUID:

- Ouverture du jeu de **données d'entrée** avec `OpenFLUID.openDataset`
- Définition du **chemin de sortie des résultats** avec `OpenFLUID.setCurrentOutputDir`
- **Sélection des résultats** à produire avec `OpenFLUID.addVariablesExportAsCSV`
- **Exécution de la simulation** avec `OpenFLUID.runSimulation`

# Lancement d'une simulation

## Lancement et configuration d'une simulation

```
# directories for input dataset and results
OFdataset = "/path/to/project/IN"
OFresults = "/path/to/project/OUT"

# open the input dataset
OFsimu = OpenFLUID.openDataset(OFdataset) # OpenFLUID simulation

# definition and configuration of results
OpenFLUID.setCurrentOutputDir(OFresults) # directory of output files
OpenFLUID.addVariablesExportAsCSV(OFsimu,"RS")
OpenFLUID.addVariablesExportAsCSV(OFsimu,"SU")

# execution of the simulation
OpenFLUID.runSimulation(OFsimu)
```

# Exploitation des résultats

Utilisation de la fonction `OpenFLUID.loadResult` pour rendre les résultats de simulation disponibles sous la forme de données R

## Accès aux résultats en tant que données R

```
# execution of the simulation
OpenFLUID.runSimulation(OFsimu)

# loading of results as R data
OFrain = OpenFLUID.loadResult(OFsimu,"SU",5,"water.atm-surf.H.rain")
OFrunoff = OpenFLUID.loadResult(OFsimu,"SU",5,"water.surf.H.runoff")
OFinfiltration = OpenFLUID.loadResult(OFsimu,"SU",5,"water.surf.H.infiltration")
OFlevel = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.H.level-rs")
```

# Accès et modification des paramètres de simulateurs

Utilisation des fonctions `OpenFLUID.getSimulatorParam` et `OpenFLUID.setSimulatorParam` pour **obtenir** ou **modifier des paramètres** de simulateurs

## Modification de la célérité moyenne du transfert sur les RS

```
# get the meancel value for water.surf.transfer-rs.hayami simulator
mc = OpenFLUID.getSimulatorParam(OFsimu,"water.surf.transfer-rs.hayami",
    "meancel")

# decrease of the meancel value by 0.05
mc = mc - 0.05

# apply of the news meancel value for water.surf.transfer-rs.hayami simulator
OpenFLUID.setSimulatorParam(OFsimu,"water.surf.transfer-rs.hayami","meancel",mc)

# execution of the simulation with the new meancel parameter
OpenFLUID.runSimulation(OFsimu)
```

# Accès et modification des attributs spatiaux

Utilisation des fonctions `OpenFLUID.getAttribute` et `OpenFLUID.setAttribute` pour **obtenir** ou **modifier des attributs** du domaine spatial

## Modification de l'humidité initiale de la SU5

```
# get the initial soil moisture
ti = OpenFLUID.getAttribute(OFsimu,"SU",5,"thetaini")

# increase of the initial soil moisture by 10%
ti = ti * 1.1

# apply the new initial soil moisture
OpenFLUID.setAttribute(OFsimu,"SU",5,"thetaini",ti)

# execution of the simulation with the new soil moisture
OpenFLUID.runSimulation(OFsimu)
```

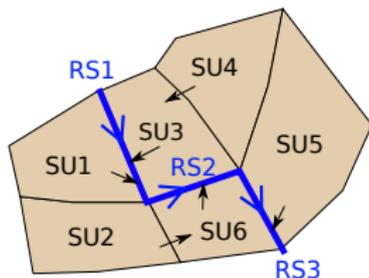
# Plan

- 1 Introduction
- 2 Bases et fonctionnement
- 3 Exemples d'utilisation
  - Simulation exemple
  - Utilisations simples
  - Utilisations avancées

# Simulation exemple

Domaine spatial :

- 6 parcelles (SU),
- 3 tronçons de fossés (RS)



Modèle MHYDAS:

- production de pluie sur SU,
- partage infiltration et ruissellement sur SU (Morel-Seytoux)
- transfert du ruissellement sur SU (Hayami)
- transfert du débit dans le réseau sur RS (Hayami)

Les sorties :

- infiltration / ruissellement (SU)
- débit et hauteur d'eau (RS)

# Visualisation graphique des résultats

Visualisation de la pluie, de l'infiltration et du ruissellement sur la SU5, sous forme graphique

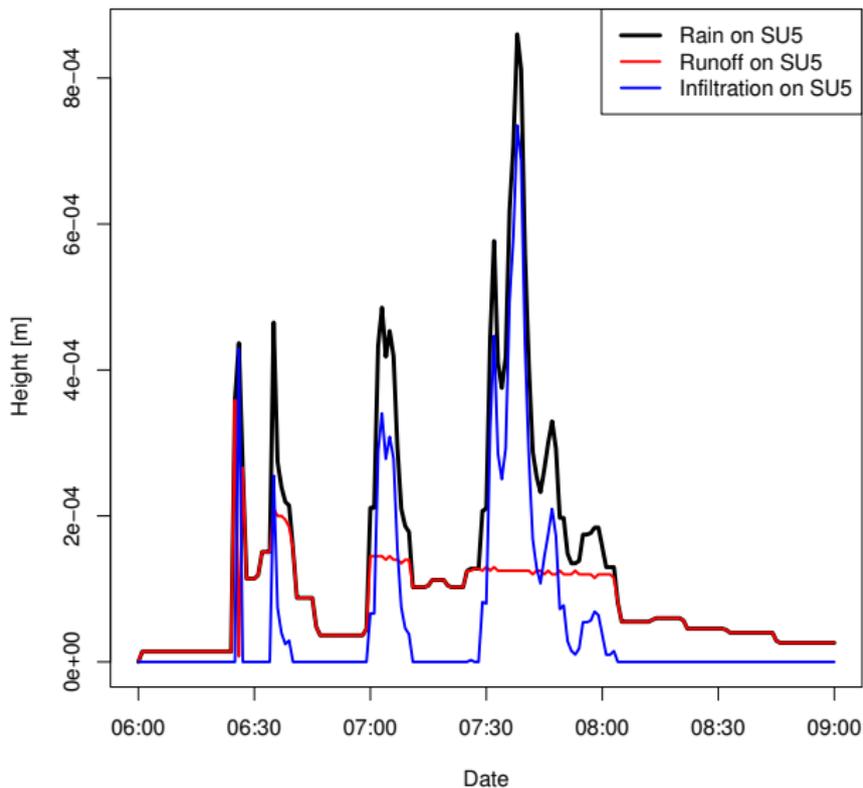
## Accès aux résultats et visualisation graphique

```
# execution of the simulation
OpenFLUID.runSimulation(OFsimu)

# loading of results as R data
OFrain = OpenFLUID.loadResult(OFsimu,"SU",5,"water.atm-surf.H.rain")
OFrunoff = OpenFLUID.loadResult(OFsimu,"SU",5,"water.surf.H.runoff")
OFinfiltration = OpenFLUID.loadResult(OFsimu,"SU",5,"water.surf.H.infiltration")

# creation of the graphic plot
plot(OFrain[,1],OFrain[,2],type='l',xlab='Date',ylab='Height_[m]')
lines(OFinfiltration[,1],OFinfiltration[,2],col='red')
lines(OFrunoff[,1],OFrunoff[,2],type='l',col='blue')
legend("topright",c('Rain_on_SU5','Runoff_on_SU5','Infiltration_on_SU5'))
```

# Visualisation graphique des résultats



# Calcul de la hauteur d'eau maximum

Calcul de la **hauteur d'eau** maximum à l'exutoire au cours de la simulation, en utilisant la fonction `max` de R

## Calcul de la hauteur d'eau maximum à l'exutoire

```
# execution of the simulation
OpenFLUID.runSimulation(OFsimu)

# loading of simulated water level as R data
OFHlevel = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.H.level-rs")

# compute of the maximul water level
OFmaxLevel = max(OFHlevel[,2])
```

## Calcul du volume d'eau cumulé

Calcul du volume d'eau cumulé à l'exutoire à partir du **débit à l'exutoire** et de la valeur du **pas de temps du modèle**, en utilisant la fonction `sum` de R

### Calcul du volume d'eau cumulé à l'exutoire

```
# execution of the simulation
OpenFLUID.runSimulation(OFsimu)

# loading of simulated downstream as R data
OFQdownstream = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.Q.downstream-rs")

# get the default time step of the simulation
OFdeltaT = OpenFLUID.getDefaultDeltaT(OFsimu)

# compute of the cumulated water volume
OFcumOutlet = sum(OFQdownstream[,2]) * OFdeltaT
```

# Optimisation de paramètres

## Contexte

Optimisation des **paramètres du modèle** et des **attributs spatiaux** pour le débit à l'exutoire **en fonction de mesures de terrain**

Les paramètres à optimiser sont la célérité dans le réseau  $C_{RS}$  et la largeur du tronçon de réseau à l'exutoire  $L_{RS3}$

⇒ Définition sous R d'une **fonction coût** qui lance la simulation et calcule la RMSE entre mesuré et simulé à partir:

- d'un vecteur contenant  $C_{RS}$  et  $L_{RS3}$
- de la solution de référence vers laquelle doit tendre la simulation

L'optimisation vise à **faire tendre cette fonction coût vers 0**, en utilisant des fonctions R dédiées à l'optimisation.

# Optimisation de paramètres

## Code R (fonctions principales)

### Calcul de la fonction coût

```
OF.runSingle <- function(X,ref) {  
  
  # redefine simulator parameters and input data according to vector X  
  OpenFLUID.setSimulatorParam(OFsimu,"water.surf.transfer-rs.hayami","meancel",X[1])  
  OpenFLUID.setAttribute(OFsimu,"RS",3,"width",X[2])  
  
  OpenFLUID.addVariablesExportAsCSV(OFsimu,'RS')  
  OpenFLUID.runSimulation(OFsimu) # run simulation with new data set  
  
  outRS3 = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.H.level-rs")  
  out = sqrt(sum((outRS3[,2]-ref[,2])**2)) # compute objective function  
  return(out)  
  
}
```

### Optimisation de paramètres

```
OFoptim = optim(par=c(0.01,0.2),fn=OF.runSingle,method='L-BFGS-B',  
               lower=0.1*c(0.01,0.2),upper=5.0*c(0.01,0.2),ref=OFrefoptim)
```



# Analyse de sensibilité d'un modèle

## Principes

Analyse de la **sensibilité du modèle** au paramètre  $C_{RS}$  et aux attributs spatiaux  $\theta_I$   $SU5$ ,  $L_{RS3}$  pour le calcul du débit cumulé et du maximum de la hauteur d'eau à l'exutoire

Principes de l'analyse de Morris :

- définition d'un plan d'expérience optimisé de parcours de l'espace des facteurs
- lancement des simulations pour les  $n$  jeux de facteurs générés
- analyse des écarts relatifs sur la variable de sortie considérée

## Analyse de Morris

```
library("sensitivity") # load of sensitivity package
OFmorrisQ <- morris(model=OF.runMulti,r=500,
                   design=list(type="oat", levels=25, grid.jump=12),
                   binf=c(0.1,0.05,0.1),bsup=c(1.0,0.35,1.0),varOut="Q")
plot(OFmorrisQ,xlim=c(0,1),ylim=c(0,1))
```

# Analyse de sensibilité d'un modèle

## Code R pour multi-simulation

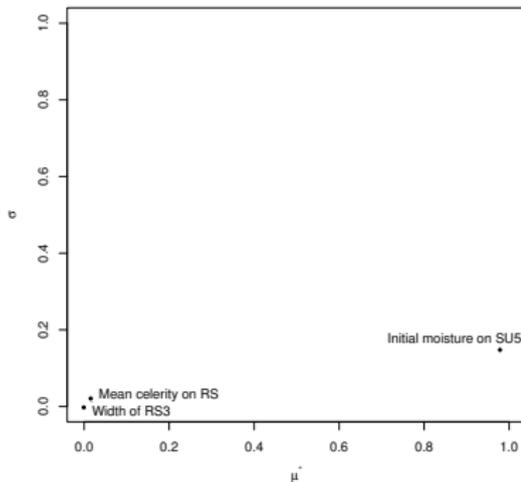
### Multi-simulation

```
OF.runMulti <- function(X,varOut) {  
  
  for (i in seq(1:nrow(X))) { # loop on the lines of the factors vector  
    C = X[i,1]; H = X[i,2]; W = X[i,3] # redefine factors  
    OpenFLUID.setSimulatorParam(OFsimu,"water.surf.transfer-rs.hayami","meancel",C)  
    OpenFLUID.setAttribute(OFsimu,"SU",5,"thetaini",H)  
    OpenFLUID.setAttribute(OFsimu,"RS",3,"width",W)  
  
    OpenFLUID.addVariablesExportAsCSV(OFsimu,'RS')  
    OpenFLUID.runSimulation(OFsimu) # run simulation with new data set  
  
    # choice of output variables  
    if (varOut=="Q") {  
      outRS3 = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.Q.downstream-rs")  
      out[i] = sum(outRS3[,2])*OpenFLUID.getDeltaT(OFsimu)  
    }  
    else if (varOut=="H") {  
      outRS3 = OpenFLUID.loadResult(OFsimu,"RS",3,"water.surf.H.level-rs")  
      out[i] = max(outRS3[,2])  
    }  
  }  
  return(out)  
}
```

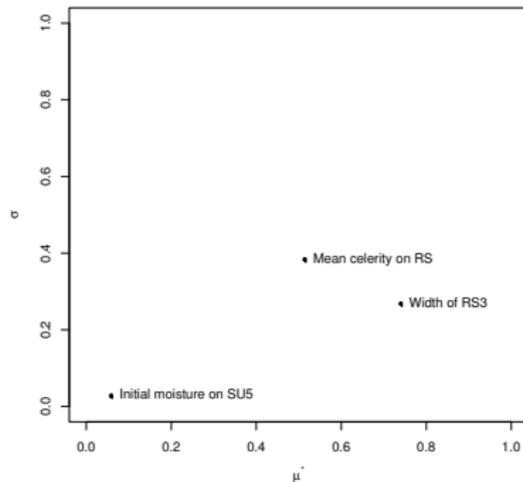
# Analyse de sensibilité d'un modèle

## Résultats

Morris sensitivity analysis on the downstream



Morris sensitivity analysis on the water level



Sensibilité des paramètres et attributs spatiaux analysés pour le calcul du débit et de la hauteur d'eau à l'exutoire

- en abscisse: sensibilité **directe** (tous les autres facteurs fixés)
- en ordonnée: sensibilité **indirecte** (en interaction avec les autres facteurs)

# Compléments

## Systèmes supportés pour ROpenFLUID:

- Linux (packaging prévu pour Windows, non testé sous Mac)
- R versions 2.14+ et 3.xx

## Documentations complémentaires ROpenFLUID:

- Manuel ROpenFLUID

<https://www.openfluid-project.org/resources/docs/manuals/en/openfluid/2.1.8/ROpenFLUID/html/index.html>

- Tutoriel ROpenFLUID (Mai 2015)

[http://www.openfluid-project.org/resources/docs/trainings/2015-05/ropenfluid\\_fr.pdf](http://www.openfluid-project.org/resources/docs/trainings/2015-05/ropenfluid_fr.pdf)

# Liste des commandes ROpenFLUID I

**OpenFLUID.addExtraObserversPaths**(*paths*)  
**OpenFLUID.addExtraSimulatorsPaths**(*paths*)  
**OpenFLUID.addVariablesExportAsCSV**(*ofblob, unitclass*)  
**OpenFLUID.createAttribute**(*ofblob, unitclass, attrname, attrval*)  
**OpenFLUID.getAttribute**(*ofblob, unitclass, unitid, attrname*)  
**OpenFLUID.getAttributes**(*ofblob, unitids, unitidsAsRownames, unitclass, attrnames*)  
**OpenFLUID.getAttributesNames**(*ofblob, unitclass*)  
**OpenFLUID.getDefaultDeltaT**(*ofblob*)  
**OpenFLUID.getExtraObserversPaths**()  
**OpenFLUID.getExtraSimulatorsPaths**()  
**OpenFLUID.getGeneratorParam**(*ofblob, unitclass, varname, paramname*)  
**OpenFLUID.getGeneratorParamNames**(*ofblob, unitclass, varname*)  
**OpenFLUID.getGeneratorParams**(*ofblob, unitclass, varname, paramnames*)  
**OpenFLUID.getGeneratorsVarNames**(*ofblob, unitclass*)  
**OpenFLUID.getModelGlobalParam**(*ofblob, paramname*)  
**OpenFLUID.getModelGlobalParamNames**(*ofblob*)  
**OpenFLUID.getModelGlobalParams**(*ofblob, paramnames*)  
**OpenFLUID.getObserverParam**(*ofblob, obsid, paramname*)  
**OpenFLUID.getObserverParamNames**(*ofblob, obsid*)  
**OpenFLUID.getObserverParams**(*ofblob, obsid, paramnames*)  
**OpenFLUID.getObserversIDs**(*ofblob*)

# Liste des commandes ROpenFLUID II

**OpenFLUID.getObserversPaths()**  
**OpenFLUID.getPeriodBeginDate(*ofblob*)**  
**OpenFLUID.getPeriodEndDate(*ofblob*)**  
**OpenFLUID.getSimulatorParam(*ofblob*,*simid*,*paramname*)**  
**OpenFLUID.getSimulatorParamNames(*ofblob*,*simid*)**  
**OpenFLUID.getSimulatorParams(*ofblob*,*simid*, *paramnames*)**  
**OpenFLUID.getSimulatorsIDs(*ofblob*)**  
**OpenFLUID.getSimulatorsPaths()**  
**OpenFLUID.getUnitsClasses(*ofblob*)**  
**OpenFLUID.getUnitsIDs(*ofblob*,*unitclass*)**  
**OpenFLUID.getVersion()**  
**OpenFLUID.loadResult(*ofblob*,*unitclass*,*unitid*,*varname*)**  
**OpenFLUID.loadResultFile(*filepath*)**  
**OpenFLUID.openDataset(*path*)**  
**OpenFLUID.openProject(*path*)**  
**OpenFLUID.printSimulationInfo(*ofblob*)**  
**OpenFLUID.removeAttribute(*ofblob*,*unitclass*,*attrname*)**  
**OpenFLUID.removeModelGlobalParam(*ofblob*,*paramname*)**  
**OpenFLUID.removeObserverParam(*ofblob*,*obsid*,*paramname*)**  
**OpenFLUID.removeSimulatorParam(*ofblob*,*simid*,*paramname*)**  
**OpenFLUID.resetExtraObserversPaths()**

# Liste des commandes ROpenFLUID III

**OpenFLUID.resetExtraSimulatorsPaths()**  
**OpenFLUID.runProject(*path*)**  
**OpenFLUID.runSimulation(*ofblob*)**  
**OpenFLUID.setAttribute(*ofblob,unitclass,unitid,attrname,attrval*)**  
**OpenFLUID.setAttributes(*ofblob,unitclass,attrvals*)**  
**OpenFLUID.setCurrentOutputDir(*path*)**  
**OpenFLUID.setDefaultDeltaT(*ofblob,deltat*)**  
**OpenFLUID.setGeneratorParam(*ofblob,unitclass,varname,paramname,paramval*)**  
**OpenFLUID.setGeneratorParams(*ofblob,unitclass,varname,paramvals*)**  
**OpenFLUID.setModelGlobalParam(*ofblob,paramname,paramval*)**  
**OpenFLUID.setModelGlobalParams(*ofblob,paramvals*)**  
**OpenFLUID.setObserverParam(*ofblob,obsid,paramname,paramval*)**  
**OpenFLUID.setObserverParams(*ofblob,obsid,paramvals*)**  
**OpenFLUID.setPeriodBeginDate(*ofblob,begindate*)**  
**OpenFLUID.setPeriodEndDate(*ofblob,enddate*)**  
**OpenFLUID.setSimulatorParam(*ofblob,simid,paramname,paramval*)**  
**OpenFLUID.setSimulatorParams(*ofblob,simid,paramvals*)**