Interface to use SIMPLACE from R

Gunther Krauss

April 23, 2014

1 Introduction

This package provides methods to interact with the modelling framework SIMPLACE[1]. SIMPLACE is written in Java (and some parts in Scala) so one can access it from R via `rJava`. The purpose of this package is to simplify the interaction between R and SIMPLACE, by providing functions to:

- initialize and configure SIMPLACE
- load a simulation (solution and project)
- parameterize the simulation
- run whole simulation or run it stepwise
- get simulation output and convert it to formats suitable for R

2 Installing the Simplace Framework

For installing SIMPLACE, please consult the webpage www.simplace.net. A brief guide to install SIMPLACE:

- If you don't have installed Java, please install an appropriate version of the (JRE or JDK) from java.com
- Download the console mode of SIMPLACE from www.simplace.net
- Unpack the zip archive to your disk. You have to unpack the whole directory `SIMPLACE` and the directory must not be renamed.
- Install the `simplace` package in R:
  `install.packages('simplace',repos=c('http://r-forge.r-project.org','http://cran.r-project.org'))`

3 Basic Usage

The usage of SIMPLACE in R follows roughly this scheme:

- init SIMPLACE by providing the path to your simplace installation directory, your working directory and your outputs
- open a SIMPLACE project form a solution (and project) file
- create a list of simulation parameters you want to change
- create and run a Simulation

[1] **S**CIENTIFIC **I**MPACT ASSESSMENT AND **M**ODELLING **PL**ATFORM FOR **A**DVANCED **C**ROP AND **E**COSYSTEM MANAGEMENT.
See www.simplace.net for more information on SIMPLACE

- get the result from the simulation
- convert the result to a R object (`data.frame`, `list` etc.)

4 Troubleshooting

- Package `rJava` should be installed automatically with `simplace`. If not, install it manually: `install.packages('rJava')`
- Architecture of R and Java have to match. If you are using 64-bit Java, you have to use 64-bit R.
- If you want to use the development version instead of the console mode, make sure that the projects `simplace`, `lap`, `simplacerun` and `lapclient` are in a common directory and set the installation dir to this directory.

5 Example

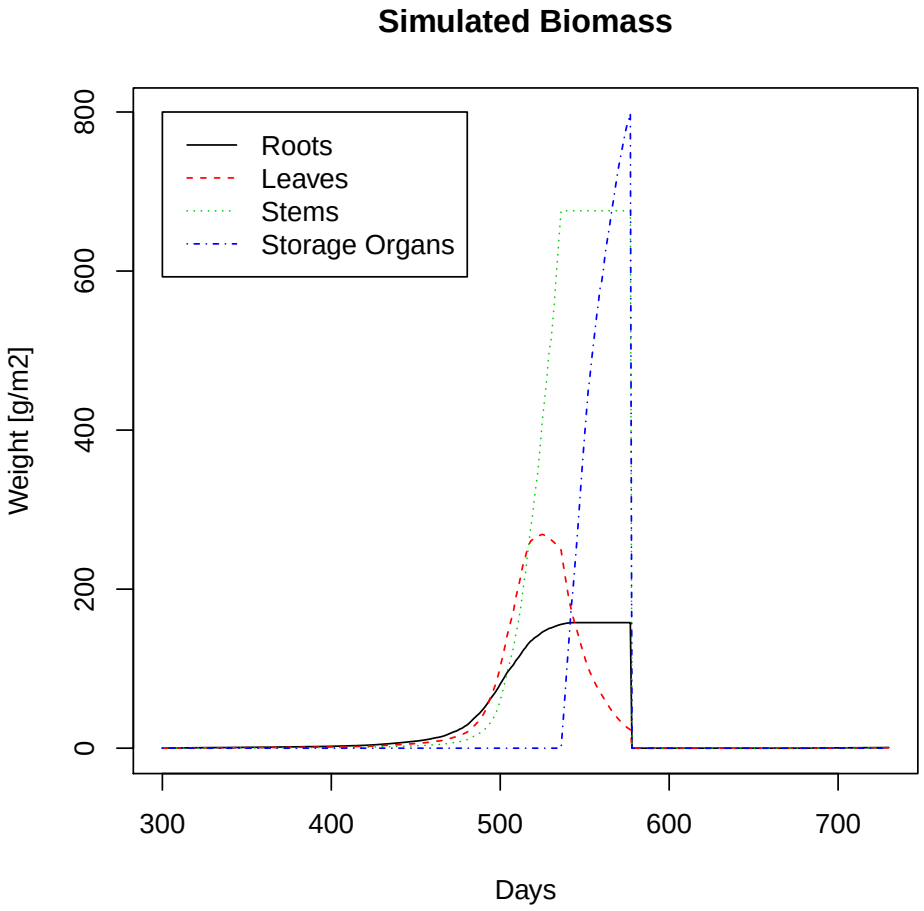
5.1 Run the simulation

```
> library(simplace)
> SimplaceInstallationDir <- "D:/java/simplace/"
> SimplaceWorkDir <- "D:/java/simplace/simplacerun/simulation/"
> SimplaceOutputDir <-  "D:/java/simplace/simplacerun/output/"
> Solution <- "D:/java/simplace/simplacerun/simulation/gk/solution/complete/Complete.sol.xml"
> simplace <- initSimplace(SimplaceInstallationDir,SimplaceWorkDir,SimplaceOutputDir)
> openProject(simplace, Solution)
> parameter <- list()
> parameter$enddate <- "31-12-1992"
> sid <- createSimulation(simplace,parameter)
> runSimulations(simplace)
> result <- getResult(simplace,"DIAGRAM_OUT", sid);
> closeProject(simplace)
```

After specifying the directories and the solution, the framework is initialized and the project opened. The end date of the simulation is (re)set and the simulation is run. After the run the result is retrieved.

5.2 Get the result and plot it

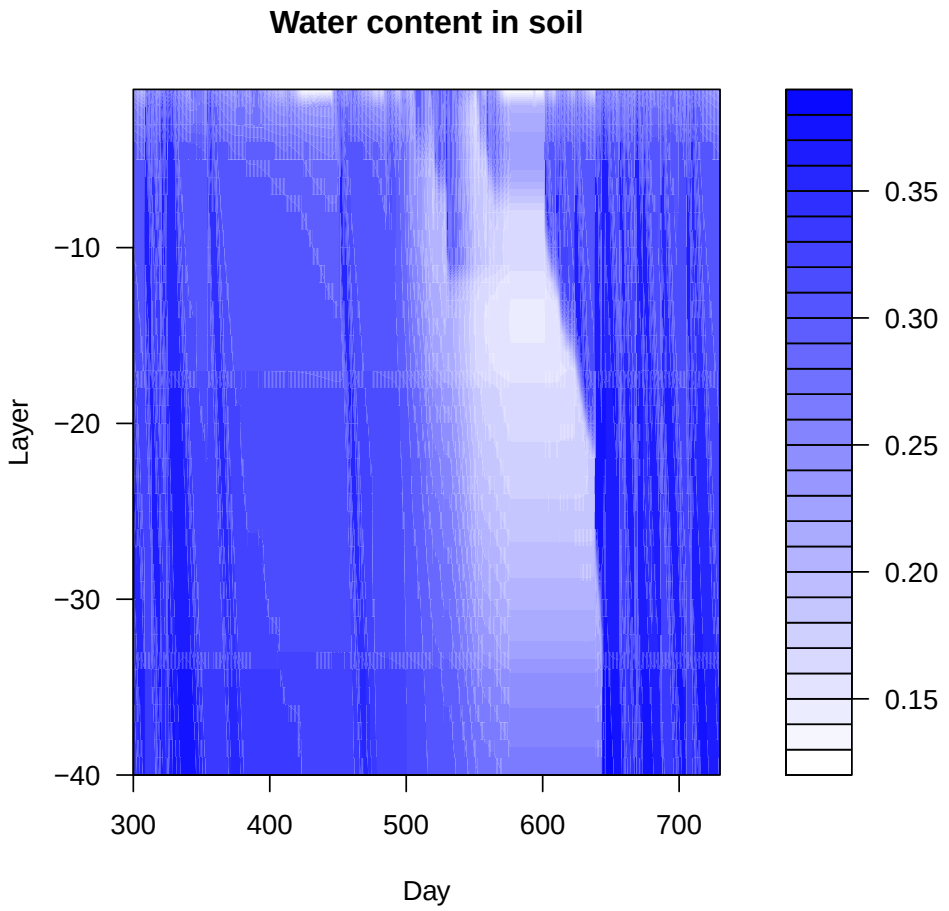
```
> resf <- resultToDataframe(result)
> dates <- 300:730
> weights <- resf[dates,
+                 c("TOP_LINE_Roots","TOP_LINE_Leaves","TOP_LINE_Stems","TOP_LINE_StorageOrgans")]
> matplot(dates,weights,type="l",xlab="Days",ylab="Weight [g/m2]",main="Simulated Biomass")
> legend(300,800,legend=c("Roots","Leaves","Stems","Storage Organs"),lty=1:4,col=1:4)
```



The result is converted to a dataframe. Interesting variables are extracted and then plotted.

5.3 Get arrays and plot them as contour plot

```
> resultlistexp <- resultToList(result,expand=TRUE)
> water <- resultlistexp$BOTTOM_ARRAY_VolumetricWaterContent
> wmat <- do.call(rbind,water)
> wmatpart <- wmat[dates,]
> layers <- dim(wmatpart)[2]
> filled.contour(dates,-(layers:1),wmatpart[,layers:1],
+                xlab="Day", ylab="Layer", main="Water content in soil",
+                color.palette = function(n){rgb((n:1)/n,(n:1)/n,1)})
```



As the result contains an array which holds the water content for 40 layers, it is transformed to a list and the array is expanded.

Contents