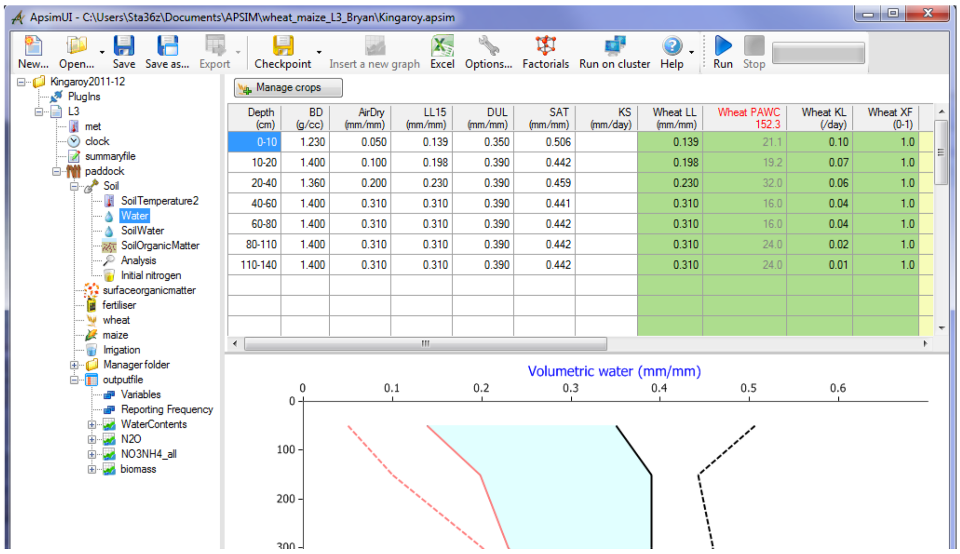
Extending the capabilities of agriculture simulators using R, an introduction to the apsimr package

by Bryan Stanfill

Introduction

The **A**gricultural **P**roduction **S**ystem s**IM**ulator (APSIM) is a widely used, powerful and highly complex computer program (Keating et al., 2003; Holzworth et al., 2014). Based on information about weather, soil properties, farming practices and land use, APSIM can predict crop and environmental outcomes such as yield, nitrogen runoff and sediment loss as a function of time and space.

APSIM is currently run either from a clunky and unappealing user interface (Figure 1) where limited analysis and visualization tools available, or from the command line, which requires a deep understanding of APSIM and a substantial amount of ad hoc programming. We propose the **apsimr** to ease the pain of editing and running several APSIM simulations using the UI or command line. The **apsimr** package includes functions to create, edit, run and analyze APSIM simulations using R. Additionally, **apsimr** acts as a bridge between the **sensitivity** package and APSIM, which provides APSIM users with access to a wide variety of uncertainty quantification tools (Pujol et al., 2014). Other packages that could be of interest for uncertainty/sensitivity analysis of APSIM are **spartan** and **multisensi** (Alden et al., 2014; Lamboni et al., 2011).


ApsimUI - C:\Users\Sta36z\Documents\APSIM\wheat_maize_L3_Bryan\Kingaroy.apsim

Depth (cm)	BD (g/cc)	AirDry (mm/mm)	LL15 (mm/mm)	DUL (mm/mm)	SAT (mm/mm)	KS (mm/day)	Wheat LL (mm/mm)	Wheat PAWC 152.3	Wheat KL (/day)	Wheat XF (0-1)
0-10	1.230	0.050	0.139	0.350	0.506		0.139	21.1	0.10	1.0
10-20	1.400	0.100	0.198	0.390	0.442		0.198	19.2	0.07	1.0
20-40	1.360	0.200	0.230	0.390	0.459		0.230	32.0	0.06	1.0
40-60	1.400	0.310	0.310	0.390	0.441		0.310	16.0	0.04	1.0
60-80	1.400	0.310	0.310	0.390	0.442		0.310	16.0	0.04	1.0
80-110	1.400	0.310	0.310	0.390	0.442		0.310	24.0	0.02	1.0
110-140	1.400	0.310	0.310	0.390	0.442		0.310	24.0	0.01	1.0

Figure 1: The user interface for APSIM version 7.6.

In the remainder of this vignette we give a general overview of APSIM then explain how the **apsimr** package can be used to run, edit, visualize and analyze APSIM using R.

APSIM Overview

This section gives a brief account of how APSIM works because it will inform the construction of the package. For a detailed description see Keating et al. (2003), or Holzworth et al. (2014) and the references therein.

APSIM is composed of several independent modules, each of which controls a specific aspect of the simulation. Modules control everything from crop growth to soil water balance, and soil nitrogen levels to farming practices (e.g. how much fertilizer to apply and when). Crop growth statistics are produced for each day in the simulation and are determined by daily weather information, soil characteristics and management choices. The weather data is provided by the user in the form of a `.met` file. Soil characteristics and management choices are controlled by the soil and management modules, respectively.

Each simulation is defined by a `.apsim` file, which controls when and if each module is called. Each module relies on a separate `.xml` file for instructions including parameter values. The module `.xml` files can have any name, but are often titled `Soil.xml`, `Wheat.xml` or something similar. Note that `.apsim` files are actually `.xml` files with a different extension. The `.apsim` simulation file controls the simulation metadata, such as what outputs to report, how often to report those outputs (e.g. daily, weekly), where to find the module files and where to write the results file.

If APSIM is executed successfully, a text file with the extension `.sum` is produced, which contains a detailed summary of the simulation including possible errors. When an APSIM simulation has run successfully, an additional text file with the extension `.out` is written, which contains the specified outcomes at the specified time steps. The user can specify the file names as well as where the files are written. By default, both files inherit the name of the simulation and are written in the same directory as the `.apsim` simulation fill. For example, if no output name or location is specified by the user then the results of the simulation `Millet.apsim` currently stored on the user's desktop, then APSIM will create files `Millet.sum` and `Millet.out` on the user's desktop.

Run and Visualize APSIM

The function used to run APSIM from R is called `apsim`. Its only required argument is `exe`, which is the path to the APSIM executable file on your machine. Additional arguments include `wd` and `files`. The `wd` argument specifies the working directory to which the results will be written and is set to the current working directory by default. The argument `files` is the list of `.apsim` simulation files to be run, which is set to all `.apsim` files in the specified working directory by default.

```r
apsimExe <-"C:/Program Files (x86)/Apsim75-r3008/Model/Apsim.exe"
apsimWd <- "~/APSIM"
toRun <- c("Canopy.apsim", "Continuous Wheat.apsim")
results <- apsim(exe = apsimExe, wd = apsimWd, files = toRun)
```

The results of a call to `apsim`, including the variable `results` above, will be of class `"apsim"`. The **apsimr** package includes a `plot` routine for objects of the class `"apsim"` and is modeled after the `plot.lm` function.

After `apsim` has been called, one can visualize all of the results as a function of time in separate plots with a call to `plot.apsim`. To visualize the results of the "Continuous Wheat.apsim" simulation do the following.

```r
plot(results$"Continuous Wheat", geom = 'line')
```

Instead of cycling through each of the plots, one can visualize all variables on one faceted plot by setting `one_plot=TRUE` (Figure 2).

```r
plot(results$"Continuous Wheat", one_plot = TRUE, geom = 'line') + theme_bw()
```

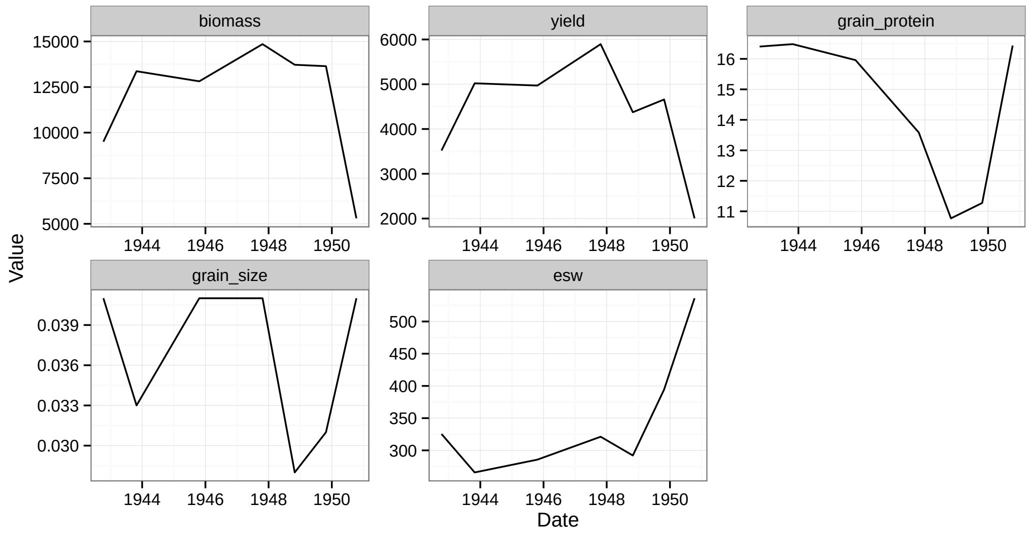


Figure 2: Figures produced by `plot.apsim` with argument `one_plot = TRUE`.

Variables can also be plotted individually using the `y` argument. To plot only the yield results for the simulation “Continuous Wheat.apsim" set `y='yield'` (Figure 3).

```
plot(results$"Continuous Wheat", y = 'yield') + geom_line(colour = 'red') + theme_b
```

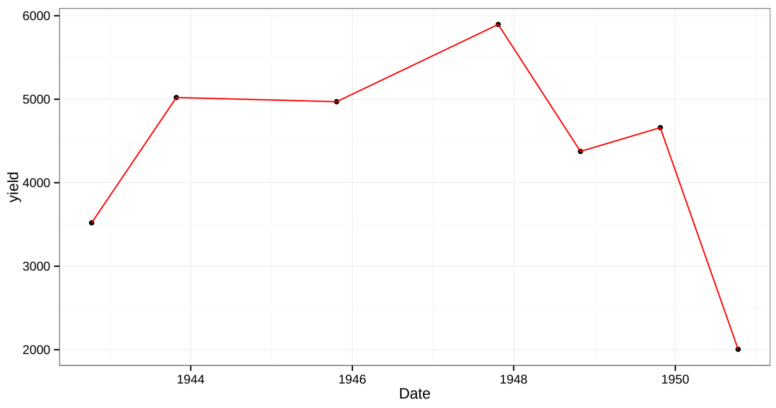


Figure 3: Plot produced by `plot.apsim` with argument `y = 'yield'`.

Editing APSIM

To edit an APSIM simulation file, i.e. files ending in `.apsim`, use the `edit_apsim` function. In this example the file we want to edit is called “Canopy.apsim" which is in the directory "∼/APSIM".

```
apsimFile <- "Canopy.apsim"
apsimWd <- "~/APSIM"
```

In this example the variables to be changed are the soil water thickness and the carbon to nitrogen ratio in the soil organic matter. In “Canopy.apsim" these variables correspond to the elements `soilWater/Thickness` and `SoilOrganicMatter/SoilCN`. The soil water thickness needs to be specified for all 11 layers of the soil considered while the carbon to nitrogen ratio is controlled by a single value. In this example, the top two layers of soil are edited to have a thickness of 200 while the last nine layers have a thickness of 300. The carbon to nitrogen ratio is changed to 10.

```
apsimVar <- c("SoilWater/Thickness", "SoilOrganicMatter/SoilCN")
apsimValue <- list(c(rep(200, 2), rep(300, 9)), 10)
```

The next snippit edits the .apsim file without overwriting it. This results in the new file being written into the working directory specified by `wd` with the additional tag "-edited.apsim". In this case the new file "Canopy-edited.apsim" is identical to "Canopy.apsim" except for the variables that have been changed by the above code snippet (up to differences in spacing and tabbing). If the user instead specifies `overwrite=TRUE` then "Canopy.apsim" is overwritten with the new variable values.

```
edit_apsim(file = apsimFile, wd = apsimWd, var = apsimVar,
           value = apsimValue, overwrite = FALSE)
```

To edit a module file, such as "Soil.xml," use the `edit_sim_file`. This function works in much the same way that `edit_apsim` works so the example below is given without commentary. Note: if `edit_sim_file` is used to edit a file with the ending `.apsim`, then `edit_apsim` will be used to edit the file and a warning will be issued. The same is true if `edit_apsim` is called with a file ending `.xml`.

```
simFile <- "Soil.xml"
simVar <- c("nitrification_pot", "dnit_nitrf_loss","wfnit_values")
simValue <- list(abs(rnorm(1)), abs(rnorm(1)), c(0,2,2,1))
edit_sim_file(file = simFile, wd = apsimWd, var = simVar,
              value = simValue, overwrite = FALSE)
```

Sensitivity Analysis

The function `apsim_vector` is a "vectorized" version of `apsim`, meaning the editing and running of APSIM is done with the same call making the process more automated. The `apsim_vector` function can be used to bridge APSIM and the package **sensitivity**.

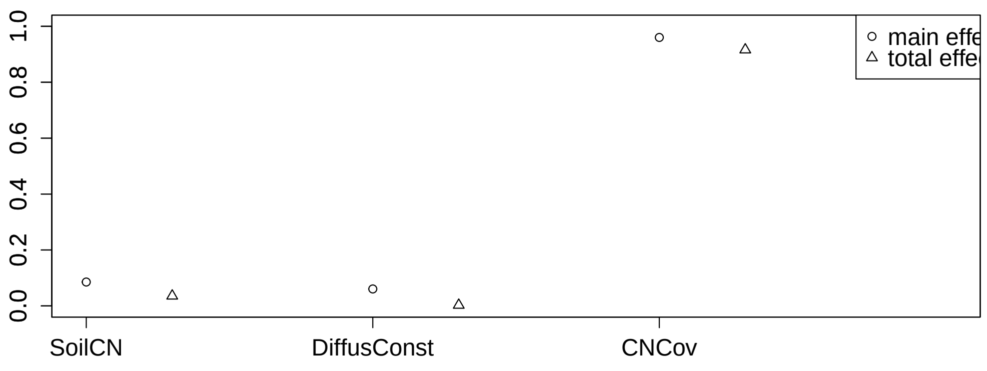
The idea is as follows: the function `apsim` returns a data frame that includes different outputs over the specified time frame at specified time intervals. The `apsim_vector` function takes that data frame and translates it into a univaraite outcome according to the user specified function `g`. This way, the function `apsim_vector` can be used as the `model` argument for the, e.g., `soboljansen` function within the `sensitivity` package. The function `g` is allowed to return a univariate or multivariate outcome, but only univariate outcomes are allowed by the **sensitivity** package.

Below is an example of a `g` function and how sensitivity analysis of APSIM can be accomplished. First, load the sensitivity package and define a `g` function. In this example we are interested in how sensitive the average cowpea yield is to initial values in the soil organic matter and soil water. Therefore we define the `g` function so that only the average cowpea yield is saved every time APSIM is executed. We define the three inputs of interest in the `vars` argument: the soil organic matters soil carbon (`SoiCN`), the soilwaters diffuse constant (`DiffusConst`) and the carbon-nitrogen covariance (`CNCov`).

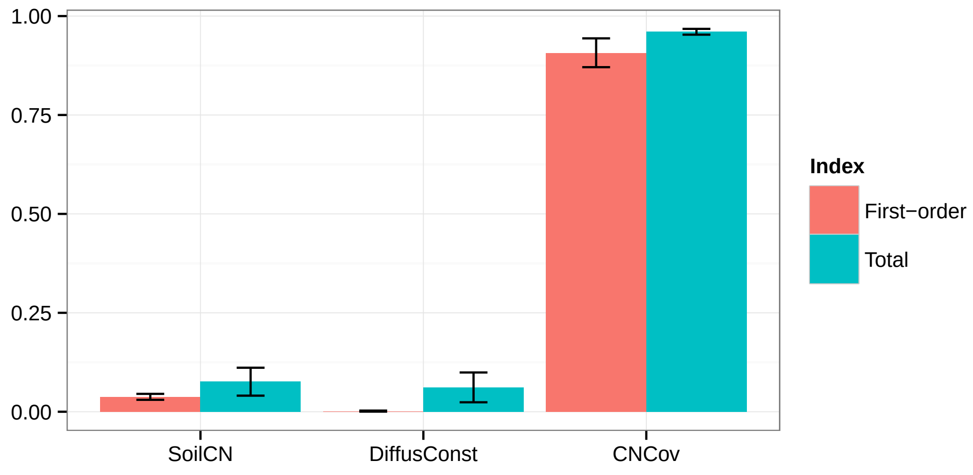
```
library(sensitivity)

meanYield<-function(x){
  return(mean(x$lai_cowpea))
}

vars <- c("SoilOrganicMatter/SoilCN", "SoilWater/DiffusConst", "SoilWater/CNCov")
```

```r
n <- 20
X1 <- data.frame(SoilCN = runif(n, 5, 25),
                 DiffusConst = runif(n, 20, 50), CNCov = runif(n, 0, 1))
X2 <- data.frame(SoilCN = runif(n, 5, 25),
                 DiffusConst = runif(n, 20, 50), CNCov = runif(n, 0, 1))

sobolResults <- soboljansen(model = apsim_vector, X1, X2, exe = apsimExe, wd = apsimWd,
                        vars = vars, to.run = apsimFile, g = meanYield, overwrite = TRUE)
plot(sobolResults)
```

The **sensitivity** package includes several methods to perform sensitivity analysis. For this example we chose the `soboljansen` function which estimates first and total Sobol' indices as described in Saltelli et al. (2010). The `soboljansen` function requires two data frames, `X1` and `X2`, to define the points in the input space at which to execute the function specified by the `model` argument. The function in the **apsimr** package to use for sensitivity analysis is `apsim_vector`, therefore we set `model=apsim_vector`. We define the `X1` and `X2` matrices so that each variable is an independent uniform random variable on an appropriate interval. The `soboljansen` function allows for additional arguments to be passed to the `model` argument. The function `apsim_vector` requires the following additional arguments: the APSIM executable location `exe`, the working directory to find the .apsim file to run `wd`, and if the edits .apsim file should be overwritten or not `overwrite`. The results of this sensitivity analysis are give in Figure 4a, which was produced by the `plot` routine for the output of the call to `soboljansen`. *Note:* the function `soboljansen` requires $p \times n$ model evaluations, where n is a user specified number of model evaluations (500 here) and p is the number of input parameter considered (3 here).



(a) Index estimates using `soboljansen` function.



(b) Index estimates using `apsim_emul_sa` function with `method = "singleGAM"`

Figure 4: The results of the sensitivity analysis of the APSIM Canopy simulation for the average cowpea yield over time.

An alternative, and potentially more efficient, method for sensitivity analysis is via emulators. The function `apsim_emul_sa` estimates sensitivity indices for each input (column of `X`) for the computer program specified by the `model` argument. If computer model runs corresponding to the input matrix `X` are available then the one can forgo the `model` argument and include the computer model runs directly using the `y` argument. *Note:* the `y` argument takes precedence over the `model` argument and an error is returned if the number of rows in `X` does not match the length of `y`.

There are two methods to emulate the computer model defined by `model`: a single GAM emulation method or a separate GAM method differentiated by `method = "singleGAM"` and `method = "separateGAM"`, respectively. Data frames consisting of first-order and total sensitivity index estimates along with a standard error, bias, lower and upper confidence bounds are returned along with the fitted values and residuals. The fitted values and residuals are provided so that the assumptions of the emulator can be verified, e.g. homoscedasticity, normality and agreement with the true computer model. The separate GAM option typically takes less time to run than the single GAM method, but it cannot be used to estimate the total sensitivity indices while the latter can.

The `apsim_emul_sa` function returns an object of class `"gamSA"`, which has its own plotting routine included in the package. The example code below demonstrates how to call the `apsim_emul_sa` function and plot the results of the `method = "singleGAM"` option. Figure 4b is the result of `plot(emulRes)`. Note: an error is returned if less than $8p^2 - 4p + 1$ model runs are available to fit the single GAM where p is the number of parameters included in the analysis.

```
n <- 61
emulX <- data.frame(SoilCN = runif(n, 5, 25),
                    DiffusConst = runif(n, 20, 50), CNCov = runif(n, 0, 1))
emulRes <- apsim_emul_sa(model = apsim_vector, X = emulX, method = "singleGAM",
                         exe = apsimExe, wd = apsimWd, vars = vars, to.run = apsimFile,
                         g = meanYield, overwrite = TRUE)
plot(emulRes)
```

The ability of `apsim_emul_sa` to include model runs instead of the model itself is a crucial difference between it and functions in **sensitivity**. This difference makes multivariate global sensitivity analysis of APSIM possible because functions of the computer model output can be analyzed as well. Below we use principal components to identify the five most important dimensions of APSIM simulations of cowpea. We then perform a sensitivity analysis on the coefficients of these five dimensions in order to identify the parameters that the simulation of cowpea is most sensitivity to. See Campbell et al. (2006) for more about multivariate global sensitivity analysis.

```
rawYield <- function(x){
  return(x$lai_cowpea)
}

cowpeaY <- apsim_vector(X = emulX, exe = apsimExe, wd = apsimWd,
                        vars = vars, to.run = apsimFile, g = rawYield,
                        overwrite = TRUE)

vCowpea<-var(cowpeaY)
pcsY<-svd(vCowpea)$u
N<-nrow(cowpeaY)
ones<-matrix(rep(1,N),ncol=1)
yBar<-(1/N)*ones%*%t(ones)%*%cowpeaY
cowpeaPCS<-(cowpeaY-yBar)%*%pcsY

mgsaRes <- vector("list",5)

for(i in 1:5){
  mgsaRes[[i]] <- apsim_emul_sa(y = cowpeaPCS[,i], X = emulX, method = "separateGAM")
}
```

Bibliography

K. Alden, M. Read, P. S. Andrews, J. Timmis, and M. Coles. Applying spartan to understand parameter uncertainty in simulations. *The R Journal*, 6(2):1–10, 2014. [p1]

K. Campbell, M. D. McKay, and B. J. Williams. Sensitivity analysis when model outputs are functions. *Reliability Engineering & System Safety*, 91(10):1468–1472, 2006. [p7]

D. P. Holzworth, N. I. Huth, E. J. Zurcher, N. I. Herrmann, G. McLean, K. Chenu, E. J. van Oosterom, V. Snow, C. Murphy, A. D. Moore, et al. APSIM–evolution towards a new generation of agricultural systems simulation. *Environmental Modelling & Software*, 2014. [p1]

B. A. Keating, P. S. Carberry, G. L. Hammer, M. E. Probert, M. J. Robertson, D. Holzworth, N. I. Huth, J. N. Hargreaves, H. Meinke, Z. Hochman, et al. An overview of APSIM, a model designed for farming systems simulation. *European Journal of Agronomy*, 18(3): 267–288, 2003. [p1]

M. Lamboni, H. Monod, and D. Makowski. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. *Reliability Engineering & System Safety*, 96(4):450–459, 2011. [p1]

G. Pujol, B. Iooss, A. Janon, P. Lemaitre, L. Gilquin, L. L. Gratiet, T. Touati, B. Ramos, J. Fruth, and S. D. Veiga. *sensitivity: Sensitivity Analysis*, 2014. URL `http://CRAN.R-project.org/package=sensitivity`. R package version 1.9. [p1]

A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010. [p5]