Running ORGANON in R

Peter Gould[1] and David Marshall[2]
[1]Pacific Northwest Research Station, Olympia, WA. 360-753-7677 pgould@fs.fed.us
[2]Weyerhaeuser Co, Federal Way, WA.

November 28, 2011

The core functions of the ORGANON growth model can be executed in R through dynamic link libraries (dlls). Running ORGANON through R can be a powerful way to combine and automate the steps required to run ORGANON and summarize the results using the analytical and graphic capabilities of R. Pre-compiled versions of the ORGANON dll are available on the ORGANON website; however, at this time these dlls are not compatible with R.

Fortunately, the source code for the ORGANON dlls is available and can be edited and compiled fairly easily to create dlls that are compatible with R. This can be done with the free FORTRAN compiler gfortran. Gfortran is part of the GCC collection of compilers and it is run in Windows in a unix-type environment called minGW. The compiler can be installed as part of a normal installation of minGW.

The following steps describe how to create R-compatible dlls on a computer running a Windows OS.

Step 1: Download and install MinGW

The installer can be found here:
http://sourceforge.net/projects/mingw/files/
Do not install MinGW in a path that contains spaces. Select the FORTRAN compiler when given the option to install components (you can also install other compilers if you wish).

Step 2: Add MinGW/bin to your path variable

This step is not necessary but it simplifies calling gfortran from the Windows command prompt. On the Windows desktop, right click the **My Computer** icon and select **Properties.** Click the **Advanced** tab then **Environmental Variables.** Select the **Path** variable and click **Edit.** Add the path to the bin folder of the MinGW installation to the path variable, separated from the rest of the variable by a semicolon. In my installation I added "**;C:\minGW\bin"** (without the parenthesis).

Step3: Download the ORGANON source code and unzip folders.

The source code can be downloaded at:
http://www.cof.orst.edu/cof/fr/research/organon/ORGANON%20DLLS%20SOURCE%20CODE.ZIP
Unzip the downloaded folder and then unzip the four folders that contain the source code for each dll (e.g., RUNDLL SOURCE CODE.zip).

Step 4: Change the syntax in the source code to make it compatible with gfortran.
The following two issues need to be fixed:

- All lines that contain "DLL_EXPORT" should be deleted or commented-out by putting a C in the first column of the line. This issues is found in the following files: PREPARE.FOR, EXECUTE2.FOR , ORGVOL.FOR , and WOODQUAL.FOR.
- The functions **JIFIX()** and **INT4()** need to be replaced by **LONG().** This issue is found in ORGVOL.FOR and VOLEQNS.FOR.

The changes can be made by opening each .FOR file in Notepad and using the find and replace tools.

Optional Changes
Since the compiled dlls are not part of an "official" version of ORGANON, it is a good idea to add a subroutine that will provide some information on what changes were made. For example, I created a file in Notepad to add a subroutine that provides information on the revisions I made. The file contains the subroutine "REVISION" with a character variable "HISTORY":

```
C       RECORD CHANGES WHEN THE DLL IS RECOMPILED
        SUBROUTINE REVISION(HISTORY)
        CHARACTER*500 HISTORY
        HISTORY='Recompiled by Peter Gould Nov 13,2011. Minor changes'
     1 //'were made for compatibility with gfortran. No substantive'
     2 //' changes were made. Contact: pgould@fs.fed.us'
        END
```

To include this subroutine, a copy of the file REVISION.FOR needs to be added to each folder containing source code and REVISION.FOR needs to be included in the list of source files when compiling each dll (see below).

Step 5. Compile the dlls

Open the Windows command prompt.

Before compiling each dll, navigate in the command prompt to the folder that contains the source code for the dll. For example, to get ready to compile ORGEDIT on my computer I typed:

cd C:\AdataFolder\ORGANON\EDITDLL SOURCE CODE

To compile each dll, type the following commands in the command prompt:

Command for ORGEDIT:
gfortran -shared -static-libgcc -o ORGEDIT.dll DIAMCAL.FOR COMFILES.FOR START2.FOR PREPARE.FOR

Command for ORGRUN:
gfortran -shared -static-libgcc -static-libgfortran -o ORGRUN.dll CRNGROW.FOR DIAGRO.FOR EXECUTE2.FOR GROW.FOR GROWTH_MODS.FOR HTGROWTH.FOR MORTALITY.FOR STATS.FOR SUBMAX.FOR TRIPLE.FOR WHPHG.FOR

Command for ORGVOL:
gfortran -shared -static-libgcc -o ORGVOL.dll ORGVOL.FOR VOLEQNS.FOR

Command for ORGWQ:
gfortran -shared -static-libgcc -o ORGWQ.dll COMFILES2.FOR WOODQ2.FOR WOODQUAL.FOR

Step 6: Place all of the newly compiled dlls in a convenient folder
By default, the dlls will be placed in the same folder as the source code. They should be put into a more convenient folder where they can be called by R.

Running the dlls in R

An example R script is attached. Keep the following in mind when writing your own script:

- Dlls are loaded into R using dyn.load(). The ORGANON subroutines are called using .Fortran(). For example, to call the "prepare" routine use .Fortran("prepare",…).

- Each variable needed by the dll needs to be initialized and of the right type and dimension. The "prepare" and "execute" subroutines expect arrays of length 2000 for variables like DBH and HT. Real variable types (non-integer numbers) must be cast as singles within the DLL call using the R function as.single(). Integers need to be cast using as.integer().

- The dll return a list object that contains the inputed and updated values. The variables in the list are not labeled, but the entries are in the same order as the variables in the

.Fortran statement. Elements in lists are referenced using double brackets. For example, to get the updated DBH values after calling “execute” use the references [[47]].

- The ORGVOL dll is designed to calculate volumes for one tree at a time as opposed to “prepare” and “execute” which use arrays. Use the mapply function to avoid a much-slower loop (see example script).

```
###Example of calling ORGANON dlls from R
###Peter Gould November 18, 20111
###load dll
##DEFINE THE DIRECTORY WHERE THE DLLS WERE PLACED
setwd("C:/AdataFolder/ORGANON/COMPILE")
dyn.load("ORGEDIT.dll", type="Fortran")
dyn.load("ORGRUN.dll", type="Fortran")
dyn.load("ORGVOL.dll", type="Fortran")

##show example of the revision subroutine
##uncomment to run it
#.Fortran("revision", "")

## run prepare first
## initialize variables
VERSION = 1
NPTS = 2
NTREES = 10
STAGE = 40
BHAGE = 37
blanks = rep(0, 2000)
SPECIES = blanks
USER = blanks
DBH = blanks
HT = blanks
CR = blanks
EXPAN = blanks
RADGRO = blanks
SPECIES[1:10] = rep(202, 10)
IEVEN = 1
DBH[1:10] = round(runif(10)*10 + 10, 1) ##random normal DBH
EXPAN[1:10] = rep(20, 10)
RVARS = rep(0, 30)
RVARS[1] = 125
SERROR = rep(0, 13)
TERROR = rep(0, 2000*6)
SWARNING = rep(0, 8)
TWARNING = blanks
IERROR = 0
```

```
IRAD = 0
GROWTH = blanks
ACALIB = rep(0, 3*18)
prepare =.Fortran("prepare", as.integer(VERSION), as.integer(NPTS),
as.integer(NTREES), as.integer(STAGE), as.integer(BHAGE),as.integer(SPECIES),
as.integer(USER), as.integer(IEVEN), as.single(DBH), as.single(HT),
as.single(CR), as.single(EXPAN), as.single(RADGRO), as.single(RVARS),
as.integer(SERROR), as.integer(TERROR), as.integer(SWARNING),
as.integer(TWARNING), as.integer(IERROR), as.integer(IRAD),
as.single(GROWTH), as.single(ACALIB))

##show ht and diameter
getht = prepare[[10]][1:NTREES]
getdbh = prepare[[9]][1:NTREES]
getcr = prepare[[11]][1:NTREES]
windows()
plot(getdbh, getht, xlab="DBH", ylab="Heights predicted by ORGANON")
windows()
plot(getdbh, getcr, xlab="DBH", ylab="Crown ratios predicted by ORGANON")

##Initialize remaining variables
CYCLEG = 0
TREENO = blanks
TREENO[1:10] = 1:10
PTNO  = blanks
PTNO[1:10] = rep(c(1, 2), each=5)
INDS = rep(0, 30)
INDS[4] = 1
INDS[9] = 1
##DBH initialized above above
DBH1 = DBH
NTREES1 = NTREES
EXPAN1 = EXPAN
HT1 = prepare[[10]] ##extracted from prepare
CR1 = prepare[[11]] ##extracted from prepare
SCR1 = blanks
MGEXP = blanks
ACALIB = prepare[[22]]
PN = rep(0, 5)
YSF = rep(0, 5)
BABT = 0
BART = rep(0, 5)
NPR = rep(0, 5)
YST = rep(0, 5)
PRAGE = rep(blanks, 3)
PRLH = rep(blanks, 3)
PRDBH = rep(blanks, 3)
PRHT = rep(blanks, 3)
PRCR = rep(blanks, 3)
```

```
PREXP = rep(blanks, 3)
BRCNT = rep(blanks, 3)
BRHT = rep(blanks, 40)
BRDIA = rep(blanks, 40)
JCORE = rep(blanks, 40)
SERROR = rep(0, 35)
TERROR = rep(blanks, 6)
SWARNING = rep(0, 9)
TWARNING = blanks
DGRO = blanks
HGRO = blanks
CRCHNG = blanks
SCRCHNG = blanks
MORTEXP = blanks
NTREES2 = 10
DBH2 = blanks
HT2 = blanks
CR2 = blanks
SCR2 = blanks
EXPAN2 = blanks
STOR = rep(0, 30)

##run repeatedly
out1 = data.frame(PERIOD = 0, PLOT = PTNO, TREE=TREENO, DBH = DBH1, HT=HT1,
CR=CR1, SCR=SCR1, EXPAN=EXPAN1)
out1 = out1[1:NTREES1, ]
for(k in 1:20){
##make the call here
grow = .Fortran("execute", as.integer(CYCLEG), as.integer(VERSION),
as.integer(NPTS), as.integer(NTREES1), as.integer(STAGE), as.integer(BHAGE),
as.integer(TREENO), as.integer(PTNO), as.integer(SPECIES), as.integer(USER),
as.integer(INDS), as.single(DBH1), as.single(HT1), as.single(CR1),
as.single(SCR1), as.single(EXPAN1), as.single(MGEXP),
as.single(RVARS), as.single(ACALIB), as.single(PN), as.single(YSF),
as.single(BABT), as.single(BART), as.single(YST), as.integer(NPR),
as.integer(PRAGE), as.single(PRLH), as.single(PRDBH), as.single(PRHT),
as.single(PRCR), as.single(PREXP), as.integer(BRCNT), as.integer(BRHT),
as.integer(BRDIA), as.integer(JCORE), as.integer(SERROR), as.integer(TERROR),
as.integer(SWARNING), as.integer(TWARNING), as.integer(IERROR),
as.single(DGRO), as.single(HGRO), as.single(CRCHNG), as.single(SCRCHNG),
as.single(MORTEXP), as.integer(NTREES2), as.single(DBH2), as.single(HT2),
as.single(CR2), as.single(SCR2), as.single(EXPAN2), as.single(STOR))
        ##Pull out updated values
        CYCLEG = k
        NTREES1 = grow[[46]]
        DBH1 = grow[[47]]
        HT1 = grow[[48]]
        CR1 = grow[[49]]
        SCR1 = grow[[50]]
```

```
        EXPAN1 = grow[[51]]
        STOR = grow[[52]]
        out2 = data.frame(PERIOD = k, PLOT = PTNO, TREE=TREENO, DBH = DBH1,
HT=HT1, CR=CR1, SCR=SCR1, EXPAN=EXPAN1)
        out2 = out2[1:NTREES1, ]
        out1 = rbind(out1, out2)
}

###Plot values of basal area over time
BA1 = with(out1, aggregate(list(BA=DBH**2*0.005454*EXPAN),
list(YEAR=PERIOD*5, PLOT=PLOT), sum))
BA2 = with(BA1, aggregate(list(BA=BA), list(YEAR=YEAR), mean))
windows()
with(BA2, plot(YEAR, BA, type="b", xlab="Year", ylab="BA (sqft/acre)"))

#########################################################################
## use a function to calculate volumes
returnVol = function(DBH, HT, CR){
  VERSION = 1
  CFTD = 6
  CFSH = 1.2
  LOGLL = 32.0
  LOGML = 8.0
  LOGTD = 6.0
  LOGSH = 0.5
  LOGTA = 8.0
  SP = 202
  VERROR = rep(0, 5)
  TERROR = rep(0, 4)
  VWARNING = rep(0, 5)
  TWARNING = 0
  IERROR = 0
  CFVOL = 0
  BFVOL = 0
  orgvol =.Fortran("volcal", as.integer(VERSION), as.integer(SP),
as.single(CFTD), as.single(CFSH), as.single(LOGLL), as.single(LOGML),
as.single(LOGTD), as.single(LOGSH),as.single(LOGTA), as.single(DBH),
as.single(HT), as.single(CR), as.integer(VERROR), as.integer(TERROR),
as.integer(VWARNING), as.integer(TWARNING), as.integer(IERROR),
as.single(CFVOL), as.single(BFVOL))

  ##check results
  return(as.numeric(orgvol[[18]]))
}

##calculate and plot volumes
##note use of mapply function
out1$VOL = mapply(returnVol, out1$DBH, out1$HT, out1$CR)
###Plot values of volume over time
```

```
VOL1 = with(out1, aggregate(list(VOL=VOL*EXPAN, TPA=EXPAN),
list(YEAR=PERIOD*5, PLOT=PLOT), sum))
VOL2 = with(VOL1, aggregate(list(VOL=VOL, TPA=TPA), list(YEAR=YEAR), mean))
windows()
with(VOL2, plot(YEAR, VOL, type="b", xlab="Year", ylab="VOL (cubic
ft/acre)"))
```

This is a simple example to get you started. The process could be organized into R functions for easier use.

Also, in addition to ORGANON, the CONIFER young stand model can also be run through R (http://www.fs.fed.us/psw/programs/ecology_of_western_forests/projects/conifers/). The CONIFERS model has been compiled as an R library RCONIFERS that can be downloaded from the CRAN site on the R website (http://www.r-project.org/). This model can grow trees from planting to be handed off to ORGANON to grow to rotation.