An R Interface to Organon and Cipsanon: `cipsr`

Nathaniel Osborne
Oregon State University
Peavy Hall A002B
Corvallis, Oregon 97331
nathaniel.osborne@oregonstate.edu

Professor Douglas Maguire
Oregon State University
Richardson Hall 312
Corvallis, Oregon 97331
doug.maguire@oregonstate.edu

Professor David Hann
Oregon State University
david.hann@oregonstate.edu

March 3, 2015

Abstract

Organon and Cipsanon are two models developed and maintained by the Center for Intensive Planted-forest Silviculture at Oregon State University. These models allow the prediction of growth, yield and wood quality attributes of individual trees under a variety of stand conditions and silvicultural treatments. This vignette is a brief introduction to the *cipsr* package, which provides an R interface to the Organon and Cipsanon models.

1 Introduction

The mission of the Center for Intensive Planted-forest Silviculture (CIPS) is to understand the interactive effects of genetics, silviculture, protection, competition, nutrition, and soils on the productivity, health, and sustainability of intensively-managed, planted-forests. This mission is centered around the development and maintenance of a comprehensive, science-based decision-support system for intensive silviculture of planted forests in the Pacific Northwest. Historically, this decision support system has been known as Organon, which can be accessed through a DOS console.

Organon was first developed by David Hann in the early 1980's as part of the Forestry Intensified Research (FIR) program. The original intent was to develop Organon as a variant of the PROGNOSIS model. That intent was changed with the advent of the IBM PC. To maximize the usefulness of Organon, Hann designed PC based software to run Organon, instead of running the model on a mainframe computer as PROGNOSIS had. Free from having to conform to the PROGNOSIS model structure, Hann was able to develop Organon based upon the structure of many models existing at the time (i.e. PROGNOSIS, CRYPTOS, CACTOS, SPS, and STEMS).

The Organon model is a system of many equations that support the prediction of individual tree growth, yield and wood quality attributes. There are three versions of Organon, each with their own set of equations. The first version developed in Organon was for Southwestern Oregon (SWO). Hann developed the Southwest Oregon version with Doug Maguire, John Scrivani, Dave Larsen, Martin Ritchie, Chao-Huan Wang, Abdel Azim Zumrawi, Dave Walters and Merlise Clyde. The work initially involved developing equations for predicting: total stem volume, merchantable stem volume, taper, stump diameter (all three incorporating CR), bark thickness, branch diameter up the stem (for estimating wood quality) and height growth rate for the six

major conifer species in the region; H/D, MCW, LCW, HCB, diameter growth rate and mortality rate equations for all 18 tree species measured on the plots; and maximum size-density trajectories for stands in the region. The Northwest Oregon (NWO) version of Organon was developed with a similar set of equations around the same time, based on data from the College of Forestry (CoF). Hann developed this version with Martin Ritchie, Chao-Huan Wang and Abdel Azim Zumrawi. The third version of Organon was developed for the Stand Management Cooperative (SMC). This version was developed using a system of permanent research plots in Southwest British Columbia, Western Washington, and Northwest Oregon. Access to these permanent plots allowed Hann to develop treatment response modifiers for thinning and fertilization. These were done in a manner that could then be applied to the SWO and NWO versions of Organon. Hann completed the SMC version of Organon with Dave Marshall and Mark Hanus.

Each version of Organon has been continually developed over the past thirty years. The Southwest Oregon version was further developed so it could be applied to older conifer and hardwood stands during the early 1990s. This work involved sampling stands with tree ages in excess of 350 years and stand which mostly consisted of hardwood trees. Hann completed this extensive sampling effort and model development with Mark Hanus. The SMC version was enhanced by continued data collection across SMC installations. Using the expanded SMC database, Hann, Dave Marshall, Peter Gould and Connie Harrington improved equations for H/D, HCB, diameter growth rate, and mortality rate equations for Oregon white oak and to develop genetic gain multipliers for the height growth rate and diameter growth rate of Douglas-fir.

The Center for Intensive Planted-forest Silviculture was established at Oregon State University in 2006 by Doug Maguire. During this time, Organon has been extensively developed by staff at CIPS. With collaboration from the OSU Hardwood Silviculture Cooperative (HSC) a Red-Alder version of Organon (RAP-Organon) was produced. Organon equations have also been enhanced, including the equations for predicting top height, H/D, MCW, LCW, crown profile, HCB, diameter increment, height increment, crown recession rate, mortality rate, size-density trajectory, and thinning multipliers for the dynamic equations. These improvements were made by Doug Maguire, David Hann, Aaron Weiskittel, Andrew Bluhm and Tzeng Lam.

The *cipsr* package is one of the latest improvements directed at the Organon model. This package provides an R interface to the Organon model, as well as a new model called Cipsanon. Cipsanon is an experimental spin-off of the Organon model. The Cipsanon model includes unique features not found in Organon: annualized predictions and the ability to predict site index based upon water-holding capacity and growing season precipitation. The *cipsr* package has also been developed to support coinciding research which enhances both models capacity to predict attributes important to wood quality.

Users may ask why R has been selected as the new platform for using Organon and Cipsanon, especially given its long history of development as a DOS console. Early in the development of Organon, it was unclear whether the model should have been developed for a PC or mainframe. We have faced a similar question in the mature years of Organon: should the model be distributed as software with a graphical user-interface (GUI) or through a programming interface? We found it best to distribute Organon and Cipsanon through *cipsr* in the R statistical programming environment for a number of reasons. Many models have become obsolete because advances in operating systems (OS) have outpaced software development. Code maintained in R, requires much less maintenance than code associated with a forest modeling software, decreasing the chance the model will fall out of currency with a users operating system. Using Organon in a programming environment offers many benefits beyond that offered in a typical forest model software. In *cipsr* a user may predict the growth, yield and wood quality of hundreds of forest stands at a time, instead of just one single stand. With this information the user can access R's powerful post-processing capabilities to apply statistical tests, run optimization routines, produce unique graphs or create customized tables. R is also a free software, which increases the

possibilities for collaboration among many individuals for a certain project or analysis task. If R does not appeal to you after these compelling arguments, you can still find access to Organon through a host of other methods including DLL files, an Excel application and the DOS console.

2 Installation and Loading

The *cipsr* package can be downloaded and installed from a repository located on the CIPS webpage (http://cips.forestry.oregonstate.edu/). This repository also contains dependencies for *cipsr* which are the *XLConnect*, *raster*, *sp* and *rgdal* packages. To install *cipsr* on your Windows computer, do the following:

1. Ensure your copy of R is version 3.0 or greater. You can download a new copy of R at http://www.r-project.org.
2. Update Java on your computer (https://www.java.com). If your computer if 64-bits, install both the 32- and 64-bit architectures.
3. Open R (i386 or 64x)
4. install.packages("cipsr", repos="http://cips.forestry.oregonstate.edu/sites/cips/files", dependencies=TRUE)

You may be asked to create a personal R directory after entering the `install.packages(...)` command. The default personal directory for most Windows users is located in My Documents. You can download *cipsr* to any directory, provided your R session is configured correctly.

Provided *cipsr* was properly installed, it can now be loaded. To load *cipsr* enter `library(cipsr)` in the R console. This command provides access to the *cipsr* functions `load.data`, `get.template`, `grow` and an example dataset formatted for use in *cipsr* named `cipsrexam`. The function `load.data` allows the user to load a Microsoft Excel (.xls) database into R, as long as it is correctly formatted. An Excel template formatted for use in *cipsr* can be created in your working directory using `get.template`. The function `grow` simulates forest growth, yield and wood quality attributes provided a *cipsr* formatted dataset.

3 Structure of the Input

Many users will find formatting data to be the most challenging aspect of using *cipsr*. Datasets used in *cipsr* have a very specific format. A given dataset should have a list of samples, units and activities. The samples component of a list contains individual tree information. In the units component, specifications for the simulation are defined as well as attributes of the forest unit. The activities component prescribes silvicultural activities to be imposed on each unit, like fertilization or thinning. An R list `cipsrexam`, is provided after loading *cipsr* as an example. In the code below, the first few rows of `cipsrexam` are shown:

```
> library(cipsr)
> lapply(cipsrexam,function(x) x[1:5,])

$samples
  unit sample tree expan user species  dbh tht    cr radgro
1    A      1    0    20   29     202 12.2  93 0.387      0
2    A      1    0    20   29     202 10.0  92 0.379      0
3    A      1    0    20   29     202 10.3  87 0.381      0
4    A      1    0    20   39     202 13.6  97 0.333      0
5    A      1    0    20   39     202  9.0  81 0.387      0
```

```
$units
  unit latitude longitude pptdd whc wantplot wanttable woodqual
1    A     0.00      0.00     0   0        2         1        1
2    B     0.00      0.00     0   0        2         1        0
3    2     0.00      0.00     0   0        0         0        1
4   12    43.03   -122.98     0   0        0         0        0
5   13     0.00      0.00     0   0        0         0        1
  model variant driver groyrs iseven partcut pastfert stage
1     1       3      0     50      1       0        0    24
2     2       3      0     50      1       0        0    24
3     1       1      0     50      1       0        0    76
4     2       1      1     50      1       0        0    75
5     1       1      0     50      1       0        0    77
  bhage  dfsi otsi dhcal ccal dgrocal triple maxsdi dfsdi wgsdi
1    20 125.0  0.0     1    1       1      0      1     0     0
2    20 125.0  0.0     1    1       1      0      1     0     0
3    71  78.8  0.0     1    1       1      0      1     0     0
4    70   0.0 83.1     1    1       1      0      1     0     0
5    72  97.5  0.0     1    1       1      0      1     0     0
  phsdi gdval ghval dfret genes snc core cftd cfsh logll logml
1     0     0     0     0     0   0    0    0    0    32     8
2     0     0     0     0     0   0    0    0    0    32     8
3     0     0     0     0     0   0    0    0    0    32     8
4     0     0     0     0     0   0    0    0    0    32     8
5     0     0     0     0     0   0    0    0    0    32     8
  logtd logsh logta
1     6   0.5     8
2     6   0.5     8
3     6   0.5     8
4     6   0.5     8
5     6   0.5     8

$activities
  unit trigger when what     how metric target
1    A    year   24 thin   below    rel   30.0
2    A    year   24 fert       N pounds  200.0
3    A     rel   70 thin uniform    rel   50.0
4    B     rel   40 thin uniform    bap  100.0
5    2    year   81 thin    user   prop    0.8
```

Some users will prefer to import their datasets for use in *cipsr* from Excel, rather than produce an R object formatted like `cipsrexam`. In Microsoft Excel, this workbook should consist of three worksheet tabs named: samples, units and activities). For an example Excel template enter the command `get.template()`. This command creates an Excel document called 'CIPSREXAM.xls' in your current R working directory. You can identify the current working directory using `getwd()`. CIPSREXAM.xls–or a file you create based off this template–may be loaded into R using the `load.data` command. The following is a demonstration of how to obtain the CIPSREXAM.xls file and then load it into R:

3.1 The samples component

unit

Identifies the sampled stand (unit) of trees.

sample
Identifies the sample nested within a unit.

tree
Identifies an individual tree within a sample and unit.

expan
Expansion factor (trees/acre).

user
Stand age (years) to remove a given tree. This value is associated with the user thinning approach which may be specified in *activities*. A value of 0 indicates that the user thinning approach will not be used.

species
Tree species identification code. The table below lists species names and their corresponding codes. It also indicates whether that species is supported within different model variants. SWO is Southwestern Oregon, NWO is Northwest Oregon and SMC is the Stand Management Cooperative variant.

Code	Species	SWO	NWO	SMC
015	White fir	Y	N	N
017	Grand fir	Y	Y	Y
081	Incense cedar	Y	N	N
117	Sugar pine	Y	N	N
122	Ponderosa pine	Y	N	N
202	Douglas-fir	Y	Y	Y
231	Pacific yew	Y	Y	Y
242	Western red cedar	Y	Y	Y
263	Western hemlock	Y	N	Y
312	Bigleaf maple	Y	Y	Y
351	Red alder	Y	Y	Y
361	Pacific madrone	Y	Y	Y
431	Golden chinkapin	Y	N	N
492	Pacific dogwood	Y	Y	Y
631	Tanoak	Y	N	N
805	Canyon live oak	Y	N	N
815	Oregon white oak	Y	Y	Y
818	Calif. black oak	Y	N	N
920	Willow	Y	Y	Y

dbh
Diameter at breast height (inches).

tht
Total tree height (feet). The total height of each tree is not required. If left equal zero, this value will be estimated.

cr
Crown ratio. This is the crown length of a tree (feet) divided by the total tree height (feet). If left equal zero, this value will be estimated. The crown ratio of each tree is not required. If left equal zero, this value will be estimated.

radgro
Five-year radial growth (inches). The radial growth of each tree is not required. If left equal zero, this value will be estimated.

3.2 The units component

unit
Identifier for the sampled stand (unit) of trees.

latitude
Latitude (decimal degrees) of the unit. This value is only necessary when using the Cipsanon model and you want to condition growth on whc and pptdd, but do not supply estimates of those variables.

longitude
Longitude (decimal degrees) of the unit. This value is only necessary when using the Cipsanon model and you want to condition growth on whc and pptdd, but do not supply estimates of those variables.

pptdd
Precipitation for degree days greater than or equal to 41 degrees F. A value of zero indicates that this value will be estimated by *cipsr*. A value for pptdd is only necessary when the Cipsanon model is in use and you want to condition growth using whc and pptdd, instead of site-index.

whc
Water holding capacity of the top 20 inches of soil. A value of zero indicates that this value will be estimated by *cipsr*. A value for whc is only necessary when the Cipsanon model is in use and you want to condition growth using whc and pptdd, instead of site-index.

wantplot
Indicates where to produce a series of descriptive graphs.

0 No graphs should be made

1 Make graphs in R session

2 Print graphs to a folder in the working directory

wanttable
Indicates whether or not to produce a table of results outside of R. A limit of 65,536 rows of Excel output is enforced. If row limit is exceeded, you will be informed and referred to the big data section of this vignette.

0 No table should be printed to the working directory

1 Print an Excel table to a folder the working directory

woodqual
Indicates whether or not to estimate wood quality attributes of individual trees.

0 Do not estimate wood quality attributes.

1 Estimate wood quality attributes.

model
Model to be used in the simulation.

1 Organon

2 Cipsanon

variant
Model variant to be used.

1 SWO - Southwest Oregon

2 NWO - Northwest Oregon

3 SMC - Stand Management Cooperative

driver
If Cipsanon is in use, this defines how to drive site productivity.

0 Use a traditional site index estimate

1 Do not use site index: condition growth on whc and pptdd. Only the SWO variant of Cipsanon allows this option.

groyrs

Number of years to grow the unit. If Organon is used, this number should only be in 5-year increments. If Cipsanon is used, this number can be in 1-year increments.

iseven

Indicates whether or not the unit is even-aged.

0 The unit is uneven aged

1 The unit is even aged

partcut

Indicates whether or not the unit has been partially harvested in the past.

0 The unit has not been partially harvested

1 The unit has been partially harvested

pastfert

Indicates whether or not the unit has been fertilized in the past.

0 The unit has not been fertilized

1 The unit has been fertilized

stage

Total age of the unit (years). If the unit is uneven-aged, this value must be zero.

bhage

Breast height age of the unit (years).

dfsi

Douglas-fir 50-year site index, depending on model variant.

NWO Hann and Scrivani (1987)

SWO Bruce (1981)

SMC Bruce (1981) for Organon and Flewelling et al. (2001) for Cipsanon

otsi

Other species 50-year site-index, depending on model variant.

NWO Flewelling et al. (2001) Western hemlock

SWO Hann and Scrivani (1987) ponderosa pine site

SMC Flewelling et al. (2001) Western hemlock

dhcal

Indicates whether or not diameter and height should be calibrated.

0 Do not calibrate diameter and height

1 Calibrate diameter and height

ccal

Indicates whether or not crown ratio should be calibrated.

0 Do not calibrate crown ratio

1 Calibrate crown ratio

dgrocal

Indicates whether or not diameter growth should be calibrated.

0 Do not calibrate diameter growth

1 Calibrate diameter growth

triple

Indicates whether or not to use tripling in the simulation.

0 Do not triple the tree list

1 Triple the tree list

maxsdi
Indicates whether or not to enforce a maximum size-density limit.

0 Do not enforce limit

1 Enforce a maximum size-density limit

dfsdi
Douglas-fir maximum size-density index. If left equal zero, a default maximum size-density of 520 is used.

wgsdi
White/Grand fir maximum size-density index. If left equal zero, a variant specific default maximum size-density is used.

phsdi
Ponderosa pine/Western hemlock maximum size density index. If left equal zero, a variant specific default maximum size-density is used.

gdval
Douglas-fir genetic worth value for diameter growth. If you are not conditioning growth on genetic worth, leave this value equal zero.

ghval
Douglas-fir genetic worth value for height growth. If you are not conditioning growth on genetic worth, leave this value equal zero.

dfret
Needle retention of Douglas-fir infected by Swiss-needle cast. If the unit is not infected with Swiss needle cast, leave this value equal zero.

genes
Use genetic worth values for diameter and height growth.

0 Do not use genetic worth values

1 Use genetic worth values for diameter and height growth

snc
Indicates whether a unit is infected with Swiss needle cast.

0 Unit is not infected with Swiss needle cast

1 Unit is infected with Swiss needle cast

core
Definition of the juvenile wood code when estimating wood quality attributes.

0 Assume age definition

1 Assume crown definition

cftd
Top diameter inside bark for cubic foot volume estimation (inches).

cfsh
Stump height for cubic foot volume estimation (feet).

logll
Log length for Scribner volume estimation (feet). If set to zero, a default log length of 32 feet is used.

logml
Minimum log length for Scribner volume estimation (feet). If set to zero, a default minimum log length of 8 feet is used.

logtd
Top diameter inside bark for Scribner volume estimation (inches). If set to zero, a default top diameter of 6 inches is used.

logsh
Stump height for Scribner volume estimation (feet). If set to zero, a default stump height of 0.5 feet is used.

logta
Trim allowance for Scribner volume estimation (inches). If set to zero, a default trim allowance of 8 inches is used.

3.3 The activities component

unit
Identifier for the sampled stand (unit) of trees.

trigger
Type of unit condition used to trigger a silvicultural activity.

year Stand age (years)

tpa Number of trees per acre (trees/acre)

bap Basal area per acre (sq. feet/acre)

qmd Quadratic mean diameter (inches)

sdi Stand density index

rel Relative stand density (percent)

when
Level that a given trigger must reach in order to initiate a silvicultural activity (what).

what
Silvicultural activity to be carried out if/when a trigger is met.

thin Thin the unit to some target

fert Fertilize the unit to some target

how
Instructions for how a silvicultural activity should be carried out.

uniform Remove trees without respect to social position in the stand

below Remove trees with smaller diameter before trees with larger diameter

user Remove trees at an age specified with a user code in the sample sheet

N Nitrogen fertilization

metric
Type of unit condition used to define the residual target of a silvicultural activity.

prop Proportion of the stand to remove

tpa Trees per acre (trees/ac)

bap Basal area per acre (sq. ft/ac)

sdi Stand density index (index)

rel Relative density (percent)

pounds Pounds of fertilizer to apply per acre

target
Intensity of silvicultural treatment in units defined by the metric.

4 Growth, Yield and Quality

An interface to the Organon and Cipsanon models is provided in *cipsr* by calling a series of dynamic link library (DLL) files. Each DLL file is compiled from Fortran source code specific to the Organon or Cipsanon model. A list DLL files used and distributed in *cipsr* is given below. These DLL files may be found in the libs folder of the *cipsr* package: path.package("cipsr") compiled as 32-bits and 64-bits for the Windows operating system.

ORGEDIT.dll Imputation of missing information and calibration

ORGRUN.dll Growth on a 5-year time step

ORGVOL.dll Cubic and Scribner volume yield

ORGWQ.dll Wood quality attributes from thinning and final harvest

CIPSEDIT.dll Imputation of missing information and calibration

CIPSRUN.dll Growth on a 1-year time step

CIPSVOL.dll Cubic and Scribner volume yield

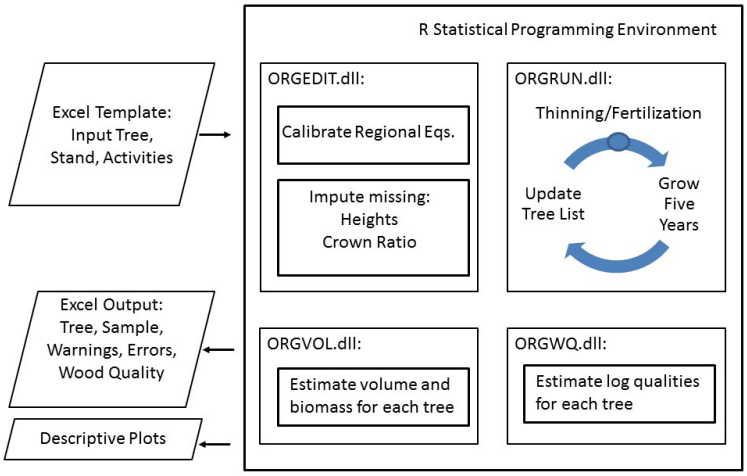

Figure 1: Flow of information in *cipsr* using the Organon model as an example.

In *cipsr*, each unit of data (i.e. in samples, units and activities) is processed by either the Organon or Cipsanon DLL files (Fig 1.) First, missing information in the samples sheet–like total heights or crown ratios–are imputed by an EDIT DLL. Next, each unit is grown to a certain age (defined by the *groyrs* field in the units sheet) by a RUN DLL. Silvicultural activities (prescribed in the activities sheet) are imposed during the simulation if their triggering conditions are satisfied by a treatment algorithm embedded in the *grow* function. If a thinning is applied, a call to estimate wood quality attributes will be made by a WQ subroutine. With or without thinning, a call to estimate wood quality attributes will be made at the end of the simulation when a final harvest is assumed. After all units from the input dataset are processed, a few basic stand level statistics are calculated (e.g. *bap* basal area per acre, *rel* relative density). After these attributes are calculated, descriptive plots and an Excel spreadsheet may be created upon user request.

4.1 Examples of Using the Grow Function

A few examples of using the function *grow* are given below.

Example 1. In this example, we acquire the *cipsr* template. The template is then loaded into R, and passed to the *grow* function.

```
library(cipsr)
get.template()
dat = load.data("CIPSREXAM.xls")
grown = grow(dat)
```

Example 2. In this example we use the already loaded *cipsrexam* object. This object is identical to the Excel template one can obtain using *get.template*. After renaming the *cipsrexam* object to something more meaningful, we remove all of the silvicultural activities planned in the activities sheet. After these minor changes to *cipsrexam*, we pass the database to the *grow* function.

```
library(cipsr)
dat = cipsrexam
dat$activities = dat$activities[0,]
grown = grow(dat, ProgressBar=FALSE)
```

Example 3. In this example, we load the *cipsrexam* object and rename it. After renaming the object, we completely replace the activities sheet with a single silvicultural treatment. The silvicultural treatment in this case is to remove 0.45-percent of the SDI of unit A from below whenever the SDI exceeds or equals 280. A modification is also made the units sheet of the input. Plots are to be drawn in R only for unit A. After growing the *dat* object, we subset output to unit A and only examine the behavior of SDI over time.

```
library(cipsr)
dat = cipsrexam
dat$activities = data.frame(unit="A", trigger="sdi", when=280,
   what="thin", how="below", metric="prop", target=0.45)
dat$units$wantplot = with(dat$units, ifelse(unit=="A",1,0))
grown = grow(dat)
subset(grown$samplelist, unit=="A", select=c(unit,subperiod,sdi))
```

Example 4. The objective of this example was to demonstrate applying a thinning triggered by a unit condition only one time. You will note that this is generally not the specification for thinning in *cipsr*, except for thinning triggered by a year or user-code. There are several steps in this analysis, but the general approach can be broken into two parts. First, units are grown with no silvicultural treatment. The year when thinning should be applied is found based on the output resulting from this simulation. Those years for treatment after passes into an object *dat*, along with a complete thinning prescription. The stand is then grown, with a thinning imposed only one time.

```
library(cipsr)
dat = cipsrexam
dat$activities = dat$activities[0,]

grown = grow(dat)$samplelist
when = lapply(split(grown,grown$unit), function(x){
                when = x$stage[min(which(x$rel>=50))]
                out = subset(x, stage==when, select=c(unit,stage)
                   )
                names(out)[names(out)=="stage"] <- "when"
                return(out)
        }
)
when = do.call("rbind",when)

when = merge(when, data.frame(trigger="year", what="thin", how="
   below", metric="rel", target=35))

dat$activities = when
dat$units$wantplot = 1
grown = grow(dat)$samplelist

by(grown,grown$unit,function(x){
```

```
                k = with(x, which(subperiod==1 & stage!=max(stage
                    )))
                x[sort(c(k,k+1), decreasing=TRUE),]
        }
)
```

5 Structure of the Output

Tabular Output Output from the grow function in *cipsr* can be retrieved in the R statistical computing environment or exported as an Excel spreadsheet. This capacity to produce output from *cipsr* in many ways was based on an expectation that some users may wish to continue their analysis in Excel, SAS, or other analytical frameworks. Output from the function `grow` has a format similar to the input. The output, whether obtained in R or Excel, is a list. The components of that list include information about individual tree and unit (sample) behavior over the simulation. Tree and unit level errors and warnings encountered during the simulation are also provided. These warnings and error messages can be particularly helpful to users not familiar with *cipsr*. Most errors and warnings can be resolved by modifying specifications for a given simulation in the units sheet of the input dataset. Variables contained within each component of the output list are defined as follows.

5.1 The treelist component

model
Indicates whether Organon or Cipsanon was used.

1 Organon

2 Cipsanon

unit
Unit identifier corresponding with the input dataset.

period
Growth period in the simulation.

subperiod
Growth subperiod in the simulation.

0 Non-harvest subperiod

1 Harvest subperiod

stage
Stand age (years).

user
Code used to mark trees for harvest at a given stand age (years).

tree
Tree identification number.

species
Species identification code. Refer to the already mentioned table of species codes.

dbh
Diameter at breast height (inches).

tht
Total tree height (feet).

cr
Crown ratio.

expan
Expansion factor (trees/acre).

mgexp
Cut tree expansion factor (trees/acre).

cfv
Cubic foot volume.

bfv
Scribner board foot volume.

5.2 The samplelist component

model
Indicates whether Organon or Cipsanon was used.

1 Organon

2 Cipsanon

unit
Unit identifier corresponding with the input dataset.

period
Growth period in the simulation.

subperiod
Growth subperiod in the simulation.

0 Non-harvest subperiod

1 Harvest subperiod

stage
Stand age (years).

bap
Basal area per acre (sq. feet/acre).

tpa
Number of trees per acre (trees/acre).

qmd
Quadratic mean diameter (inches).

sdi
Stand density index (index).

rel
Relative stand density (percent).

bfv
Board foot volume per acre (board feet/acre)

cfv
Cubic foot volume per acre (cu. feet/acre)

5.3 The woodquality component

unit
Unit identifier corresponding with the input dataset.

sample
Sample identifier corresponding with the input dataset.

period
Growth period in the simulation.

subperiod
Growth subperiod in the simulation.

0 Non-harvest subperiod

1 Harvest subperiod

stage
Stand age (years).

tree
Tree identification number.

mgexp
Cut tree expansion factor (trees/ac).

brht
Branch height at the point of insertion (feet).

brdia
Maximum branch diameter for the whorl (inches).

jcore
Diameter of the juvenile wood core (inches).

idib
Inside bark diameter (inches).

5.4 The standflags component

unit
Unit identifier corresponding with the input dataset.

flag
Warning or error message produced by the DLL.

routine
DLL subroutine which produced the error message.

level
Level of the message: tree or stand.

type
Indicator if the flag was a warning or error.

code
An error and warning message corresponding with the DLL documentation.

5.5 The treeflags component

unit
Unit identifier corresponding with the input dataset.

sample
Sample identifier corresponding with the input dataset.

tree
Tree identification number.

Species Code
Indicator for bad species code.

DBH
Indicator for bad diameter.

HT
Indicator for bad total tree height.

Crown Ratio
Indicator for bad crown ratio.

Expansion Factor
Indicator for bad expansion factor.

Shadow Crown Ratio
Indicator for bad shadow crown ratio.

Height to DBH Ratio
Indicator for total height to diameter ratio.

The remaining values of *treeflags* relate to improper specification of variables that control cubic and Scribner volume estimation in the *unit* component of the input dataset.

Graphical Output *cipsr* also offers users the capability to produce a series of descriptive plots. These plots, described in R or exported as .bmp files in the working directory, describe the evolution of stand level statistics as a function of time (Fig. 2). In the future, plots describing individual tree attributes will be made available.

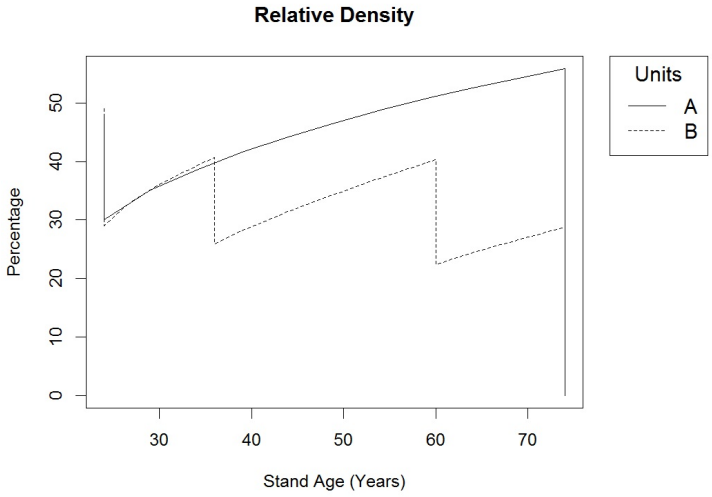

Figure 2: Example graph of relative density over time produced in *cipsr*.

6 Post-Processing Output

A Brief Introduction

One of the greatest strengths of *cipsr* is the statistical programming environment it is built upon. Almost any forestry analysis can be completed using the base packages of R. If you cannot find a solution in the base packages supplied in R, there are thousands of user-contributed packages (through CRAN and elsewhere) that help you meet an analysis goal. That goal could be to explore the development of one, ten or ten-thousand units of forestland.

A simple example of calling the `grow` function in *cipsr* and post-processing the output is given below. In this example, we simulate the growth and development of trees in the `cipsrexam` database up to 100-years stand age. With this information, we then compute the total harvest of Scribner and cubic volume among each unit. The final analysis step is to fit a linear model where the response is cubic volume at final felling, and the dependent variables are the site-index and

number of harvests taken from each unit. The purpose of this spurious analysis is demonstration - biometrics can wait for your analysis in *cipsr*.

To start our analysis we must first modify the model specifications in `cipsrexam`.

```
library(cipsr)
dat = cipsrexam
dat$units$groyrs=100-dat$units$stage
dat$units$model=2
dat$units$variant=ifelse(dat$units$unit %in% c(2,13),1,3)

dat = lapply(dat,function(x){
                x = subset(x,unit!="12")
                return(x)
        }
)

dat$units$driver = 0
dat$units$woodqual = 0
dat$units$wantplot = 0; dat$units$wanttable = 0
```

With the model setting properly specified, we can simulate the growth and yield of all the units in the *cipsrexam* object. With that information, we can extract the yields when the stand age is 100-years old for each unit.

```
grown = grow(dat)$samplelist

yield = subset(grown,stage==100 & subperiod==0 | stage!=100 &
   subperiod==1, select=c(unit,cfv,bfv,subperiod))
yield = aggregate(.~unit,data=yield,FUN=sum)
names(yield)[names(yield)=="subperiod"] <- "harvests"
yield = merge(yield, subset(dat$units,select=c(unit,dfsi)))
```

```
> print(yield)
```

```
  unit      cfv       bfv harvests  dfsi
1   13 11163.16  56102.63        1  97.5
2    2 12291.02  49543.13        1  78.8
3    A 51110.00 310043.35        2 125.0
4    B 35090.47 204695.78        3 125.0
```

Fitting our linear model, in the context of an analysis of variance (aov) is straightforward. We define our response and dependent terms from the yield object described above and use the *aov* function in R. Alternatively we could have used the functions *lm* or *anova*.

```
model = aov(cfv~dfsi+harvests,data=yield)
summary(model)
```

```
> summary(model)
```

```
            Df    Sum Sq   Mean Sq F value Pr(>F)
dfsi         1 856161041 856161041   3.420  0.316
harvests     1   6757959   6757959   0.027  0.896
Residuals    1 250303878 250303878
```

The Process Function

In March 2015, a stem bucking algorithm was added to *cipsr*. The stem bucking algorithm is incorporated into the function *process*, and allows a user to simulate harvesting of poles, saw and chip logs. Users can control the parameters of bucking by submitting arguments to the process function as a list. A set of default parameters can be obtained in R by calling the function: *processControl()*. The default parameters of process are also given in the table below.

pole
: Attempt to cut poles from each tree.

saw
: Attempt to cut saw logs from each tree.

chip
: Attempt to cut chip logs from each tree.

polell
: Maximum pole length to cut (ft).

poleml
: Minimum pole length to cut (ft).

poletd
: Minimum pole top diameter (in).

polebd
: Minimum diameter 6-feet from the pole base (in).

sawll
: Maximum saw log length (ft).

sawml
: Minimum saw log length (ft).

sawtd
: Minimum saw log top diameter (in).

chipll
: Maximum chip log length (ft).

sh
: Stump height (in).

ta
: Trim allowance (in).

By default, the *process* function prioritizes harvest of poles, then saw logs and finally chip logs. Poles are cut with emphasize upon maximizing size and grade given many dimensional specifications (Fig. 3). A requirement of 1.25 inches of sapwood in diameter at the base of each pole is also imposed. Saw logs are bucked into lumber classes using the equations of Fahey et al. (1991). Read more about these equations at 'Lumber and Veneer Recovery from Intensively Managed Young-Growth Douglas-fir' - Fahey et al. (1991) pp. 14.

An example of using the *process* function is given below. In this example, we explore the controls of *process* and make a simulation using the default control values. A second bucking simulation is made, but in this simulation we request the *process* function not harvest any poles.

```
treelist = grow(cipsrexam)$treelist
processControl()
complete = process(treelist,ProgressBar=TRUE)
partial = process(treelist,control=list(pole=FALSE))
```

McFarland Cascade®

STANDARD DIMENSIONS FOR DOUGLAS FIR AND SOUTHERN YELLOW PINE POLES

Class	7	6	5	4	3	2	1	H1	H2	H3	H4	H5	H6
Min. Top	4.77	5.41	6.05	6.68	7.32	7.96	8.59	9.23	9.87	10.50	11.14	11.78	12.41
Length of Pole	Minimum Diameter at 6 Feet from Butt (Inches)												
20	6.21	6.68	7.32	7.96	8.59	9.23	9.87						
25	6.84	7.32	8.12	8.75	9.39	10.03	10.66						
30	7.48	7.96	8.75	9.39	10.19	10.82	11.62						
35	7.96	8.59	9.23	10.03	10.82	11.62	12.41	13.21	13.85				
40		9.07	9.87	10.66	11.46	12.25	13.05	13.85	14.64	15.44	16.23		
45		9.55	10.35	11.14	11.94	12.89	13.69	14.48	15.44	16.23	17.03	17.83	18.62
50			10.82	11.62	12.41	13.37	14.32	15.12	16.07	16.87	17.67	18.62	19.42
55				12.10	12.89	13.85	14.80	15.76	16.55	17.51	18.46	19.26	20.21
60				12.41	13.37	14.32	15.28	16.23	17.19	18.14	18.94	19.89	20.85
65				12.89	13.85	14.80	15.76	16.71	17.67	18.62	19.58	20.53	21.49
70				13.21	14.32	15.28	16.23	17.19	18.14	19.26	20.21	21.17	21.96
75					14.64	15.60	16.71	17.67	18.78	19.74	20.69	21.65	22.60
80					14.96	16.07	17.19	18.14	19.10	20.21	21.17	22.12	23.08
85					15.28	16.39	17.51	18.62	19.58	20.69	21.65	22.76	23.71
90					15.60	16.87	17.83	18.94	20.05	21.17	22.12	23.24	24.19
95						17.19	18.14	19.42	20.53	21.49	22.60	23.71	24.67
100						17.51	18.62	19.74	20.85	21.96	23.08	24.19	25.15
105						17.83	18.94	20.05	21.33	22.44	23.55	24.51	25.62
110						18.14	19.26	20.53	21.65	22.76	23.87	24.99	26.10
115						18.46	19.58	20.85	21.96	23.08	24.35	25.46	26.58
120						18.78	19.89	21.17	22.28	23.55	24.67	25.78	27.06
125						18.94	20.21	21.49	22.60	23.87	24.99	26.26	27.37

Figure 3: Pole specifications implemented in the *process* function.

7 Big Data Solutions

The *cipsr* software has been designed to handle very large forestry analysis tasks. Still, some users may consider forestry problems which are large enough to require a customized solution. Examples are these problems are when: 90-percent of the available memory allocated to R is used; the output from a single simulation exceeds 1,000 megabytes; or if an Excel file to import or export has a least one sheet in excess of 65,536 rows. There are many definitions for big data, but if any of the already mentioned constraints is met in *cipsr*, we consider the problem one with a big data component.

In general, big data problems can be solved in three ways. Large problems can be reduced to smaller ones by limiting the scope of analysis, or making the analytical approach more efficient. If the problem cannot be reduced or made more efficient, the user can increase the amount of computer memory available for analysis. Memory can be increased in R by installing for RAM on the computer. In R, a limit of 8 TB of RAM can be used, in the 64-bit version. A third solution to big data problems is store objects on the hard disc and process them using RAM in chunks. The current structure of *cipsr* allows users to process large simulations in chunks, or make their analysis more efficient and smaller in scope. Two examples given below, demonstrate processing large datasets in *cipsr*. Both example use an approach of condensing the problem to minimizing memory use, and processing time in R.

Example 1. Extract harvest yields in chunks, retaining only the necessary data.

```
dat = lapply(cipsrexam, function(x) subset(x, unit=="A"))
dat$units$wantplot=0; dat$units$wanttable=0

dat = lapply(1:100, function(i){
                out = dat # Get the template from a single unit

                # Name the unit to be produced
                out = sapply(out, function(x){x$unit=i;x})

                # Randomly simulate the sample level data
                n = nrow(out$samples)
                dbh = out$samples$dbh
                out$samples$dbh = rnorm(n, mean(dbh), sd(dbh))
                out$samples$tht = 0; out$samples$cr = 0

                return(out) # Return the simulated data
        }
)

grown = lapply(dat,function(x){
        out = grow(x, ProgressBar=FALSE)$samplelist
        out = subset(out, subperiod==1 | period==max(period))
        out = aggregate(.~unit, data=out[c("unit","bfv","cfv")],
           FUN=sum)

        return(out)
        }
)
grown = do.call(rbind,grown)
```

Example 2. Extract the full array of cipsr output: constrained to final conditions.

```
grown = lapply(dat,function(x){
                out = grow(x, ProgressBar=FALSE)

                out = sapply(out,function(k){
                                if("period" %in% names(k)){
                                        k = subset(k, period==max
                                           (period))
                                }
                                return(k)
                        }
                )

                return(out)
        }
)

grown = do.call(Map, c(rbind, grown))
```