Example_1

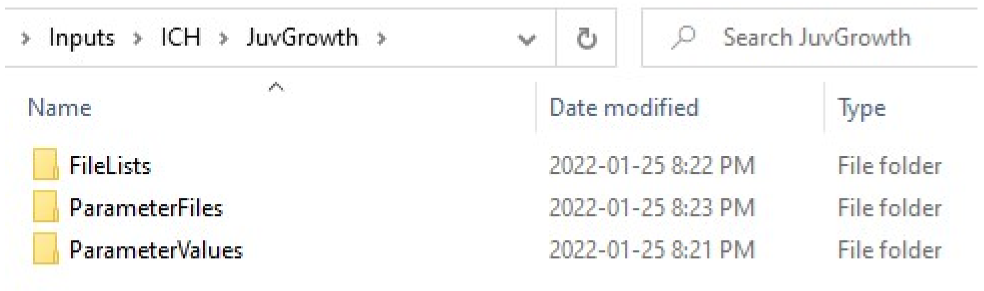
BVRC SORTIE and rsortie Workshop

March 2022

In this example, we will take a base SORTIE parameter file, substitute the values of each tree species for each diameter class (tree list) with new initial stand conditions. We will also change the length of the run. We will do this in SORTIE via the GUI and using rsortie.

Folder structure:

While creating a folder structure is entirely up to the user, we typically default to a system that helps us keep track of the various files required to create and store SORTIE parameter files. Within the main folder where you want to house SORTIE parameter files, we create a folder to contain the textfiles that specify which files will be used for any parameter file creation (FileLists). We create a second folder that contains the SORTIE parameter files (ParameterFiles) and a third folder that contains any of the new values that we wish to use to create new SORTIE parameter files (ParameterValues)



Name	Date modified	Type
FileLists	2022-01-25 8:22 PM	File folder
ParameterFiles	2022-01-25 8:23 PM	File folder
ParameterValues	2022-01-25 8:21 PM	File folder

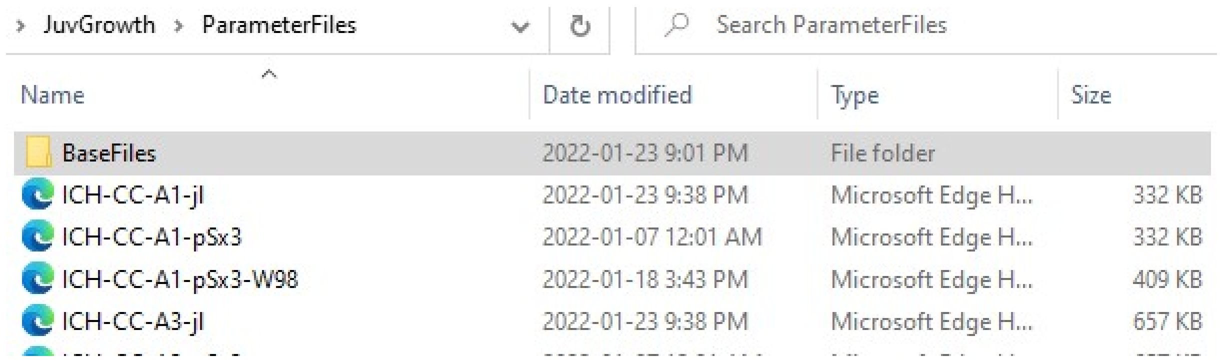
Figure 1: Folder structure hierarchy

Additionally, to keep track of the base SORTIE parameter files, we store those within a sub folder in the ParameterFiles folder (BaseFiles), and all newly created SORTIE parameter files are populated into the main ParameterFiles folder.

This folder structure will be called explicitly in the rsortie example below, but generally keeping track of the files used in makeing new SORTIE parameter file is important. If mostly using the GUI, we recommend using a file within this folder structure to annotate your workflow. For more information on the structure and file requirements for rsortie see https://aclason.github.io/rsortie/articles/prepare_inputs.html.

Setting initial stand conditions and run length

To run SORTIE, you must first have a valid parameter file. These files are parameterized by data and therefore represent the stand dynamics for forests defined by the developers. Here, we will use the freely available parameter file that comes with downloading SORTIE (GMF), as well as the SORTIE parameter



Name	Date modified	Type	Size
BaseFiles	2022-01-23 9:01 PM	File folder	
ICH-CC-A1-jl	2022-01-23 9:38 PM	Microsoft Edge H...	332 KB
ICH-CC-A1-pSx3	2022-01-07 12:01 AM	Microsoft Edge H...	332 KB
ICH-CC-A1-pSx3-W98	2022-01-18 3:43 PM	Microsoft Edge H...	409 KB
ICH-CC-A3-jl	2022-01-23 9:38 PM	Microsoft Edge H...	657 KB

Figure 2: Folder structure hierarchy 2

files developed by Dave Coates, Charlie Canham and others in BC. For more information on the inputs required to run SORTIE, visit http://sortie-nd.org/help/manuals/index.html and https://aclason.github.io/rsortie/articles/use__rsortie.html. For more information on the SORTIE parameter files used in BC see https://forests-bvcentre.ca/sortie-for-bc-ecosystems.

In this example, We will start with a base parameter file (SBS-01.xml), define the starting forest conditions at time 0 (FR01Inits.csv, etc.), and update run details (d.csv).

Making new parameter files in the SORTIE graphical user interface (GUI)

Create new files

1. Open SORTIE
2. Open the base file (SBS-01.xml) in the SORTIE GUI
3. Open the csvs for the initial stand conditions we want to update (“FR01.csv”,“FR02.csv”,etc) in excel
4. Select Edit > Model Settings
5. Select Tree Population
6. Copy the stems/ha in each diameter class from csvs to the base parameter file. Hit OK.

- make sure the species, size class are lined up properly to match the base parameter file in SORTIE
- init.dens.1 refers to “Initial Density Seedlings (#/ha)”
- the diameter values represents the upper diameter size

7. Select Edit > Model Settings > Plot
8. Update the timesteps to 4. Hit OK

Update outputs

1. Select Edit > Output options
2. Update both detailed and summary output file locations to the correct location on your computer. Hit OK
3. Save the parameter file: File > save parameter file. Give it a new name and save it in the correct location

OPTIONAL Repeat for the remaining stands

Run sortie

1. Model > Run to run the current parameter file or
2. Model > RunBatch to select all the newly created parameter files to run in sequence

Note - a batch run is run in serial (one after the other) not in parallel (run simultaneously on different cores). runSortiePar (in development) in the rsortie package uses parallel processing functions in R to execute parallel runs

Open outputs

1. Select File > Open output file and navigate to the detailed output file created by the above run, or provided in the workshop
2. Select line graphs in upper right and navigate to live adults absolute density
3. Select histograms DBH histogram
4. explore any of the other output options of interest

Extract tree data from SORTIE

To export tree data from SORTIE for analysis in other programs (excel, R, etc.) use the "batch extract detailed output files" option in the tools menu. Every tree from every time step with output options selected prior to the run as part of parameter file creation will be included in these exported tables. For instance, tree data often includes variables such as x, y coordinates, DBH, Height etc.

1. Tools > batch extract detailed output files
2. navigate the the detailed output file or files that you want to extract tree data from. Add files to batch and select Next
3. Select tables > Timestep tree writer and browse to the location where the files should be extracted to
4. Assign a root name for the extracted tree data select OK
5. 1 text file for every timestep will be created
6. Open this text file in an excel or text editor, or see below for reading in outputs using rsortie

Making new parameter files in rsortie

```
library(rsortie)
```

First, let's get familiar with the structure of SORTIE parameter files and the translation between variables defined by users in R and the variables contained within the existing parameter file.

Open a SORTIE parameter file as an .xml

Select either the base parameter file (SBS-01.xml), or one of the new files you've created. Right click and open it with an internet browser (internet explorer or Microsoft edge work well). Note the structure of the .xml and the naming convention of behaviours and values within the SORTIE parameter file.

- Find the behaviour name for initial density of a specific tree species
- Note that species in the R environment will have to be passed to match those in the .xml
- Find a behaviour related to growth in the behaviour list

- Note the list position
- scroll down and find that behaviour elsewhere in the .xml, and note that the list position is part of the behaviour name

Open rsortie variable names

```
head(VariableNames)
tail(VariableNames)
```

The VariableNames object is essential to using rsortie, to connect the variables defined by a user in an R environment to those in the SORTIE parameter file.

See https://aclason.github.io/rsortie/articles/prepare_inputs.html for detailed information on the VariableNames object and how to edit it.

makeFiles

The makeFiles function may be the only function required by a user to update a SORTIE parameter file with new values. Many of the functions in the rsortie package are called from makeFiles and so are not necessarily accessed directly by users.

```
# to learn more about makeFiles
`?`(makeFiles)
```

If you want to learn more about `makeFiles()`, follow the example provided in the function documentation and read more at https://aclason.github.io/rsortie/articles/using_makeFiles.html

To substitute new values for the starting tree conditions in rsortie, start by setting your pathways.

rsortie allows pathway arguments to be passed to the makeFiles function in order to accommodate the folder structure outlined above, or any user-defined pathways. If you are using a Rstudio project, you can use relative pathways, which helps make code more reproducible on other computers. Here, you can change the local path to your machine, and update the base, newxmls, and newvals pathways to the appropriate directories based on where you have saved your files, and where you want your new files to go.

```
# exmple of the local root pathway.  In this case, all the
# files are within the Inputs directory:

# Change this to your root input directory
loc_path <- "D:/GitHub/rsortieWorkshopFiles/Example1/Inputs/"

# where are your base parameter files?
My_basePath <- paste0(loc_path, "ParameterFiles/BaseFiles/")

# where do you want to put newly created parameter files?
My_newxmlPath <- paste0(loc_path, "ParameterFiles/")

# where are the values that will substitute those in the
# base file?
My_newvalsPath <- paste0(loc_path, "ParameterValues/")
```

Next, you need to provide the actual names of files that will be used to create new parameter files. This file list can be a text or csv file, or you can create it in R and access by that object. We are providing an example here where we read in a csv that contains the files needed to create new SORTIE parameter files

List of files

```
read.csv(paste0(loc_path, "FileLists/Init_ForestCarbon.csv"))
```

Notice the column names and the numbers associated with each file. For more information on specifying this file see https://aclason.github.io/rsortie/articles/prepare_inputs.html. What does type 0 for "SBS-01.xml" mean?

IMPORTANT - does one of those files include an updated output file location???

Tree list

```
read.csv(paste0(My_newvalsPath, "FR01Inits.csv"))
```

Notice the first row contains NAs. For more information on creating the tree list see https://aclason.github.io/rsortie/articles/prepare_inputs.html.

Run details

In this example, we've included a second csv of new values (d.csv) that contains the output directories for the runs. In many cases, we also include details for the run, as we pass this file to all newly created parameter files, and if they are all run for the same number of timesteps, we can include that in this "run details" file. For more information on how the different files are combined to make new parameter files see https://aclason.github.io/rsortie/articles/prepare_inputs.html.

The example we are working on here is based off a recent study where stands were measured at different times since fire (chronosequence). We wanted to use SORTIE to model each stand to 100 years since fire to calculate change in carbon over time. This means each new parameter file created in rsortie, required a different number of timesteps for each SORTIE run. So we just added the timesteps for each newly created parameter file to the tree list to substitute the timesteps uniquely for each stand.

```
read.csv(paste0(My_newvalsPath, "d.csv"))
```

Notice the double backslashes in the output file name. If you try and create this file in R, be sure you understand how forward and backslashes work in R.

Update outputs

If you want to try running the newly created parameter files in the SORTIE GUI or using the runSORTIE function in rsortie, you must include a valid output directory to be pasted into the new parameter file: Update the output pathway to valid local directory

```
RunDetails <- fread(paste0(My_newvalsPath, "d.csv"))
RunDetails
# where do you want outputs? Update this to a valid
# directory on your computer
Output_path <- "D:\\GitHub\\rsortieWorkshopFiles\\Example1\\Outputs\\"
# must use double backslash not single if writing this in R
# Update Run details with new directory
RunDetails[V1 == "ShortOutput", `:=`(Western_Hemlock, Output_path)]
RunDetails[V1 == "Output", `:=`(Western_Hemlock, Output_path)]

# view the update
RunDetails

# write out as a csv for makefiles to access.
write.csv(RunDetails, paste0(My_newvalsPath, "d.csv"), row.names = FALSE)
```

What would happen if you don't overwrite the run details file called "d.csv"?

Create new files with rsortie

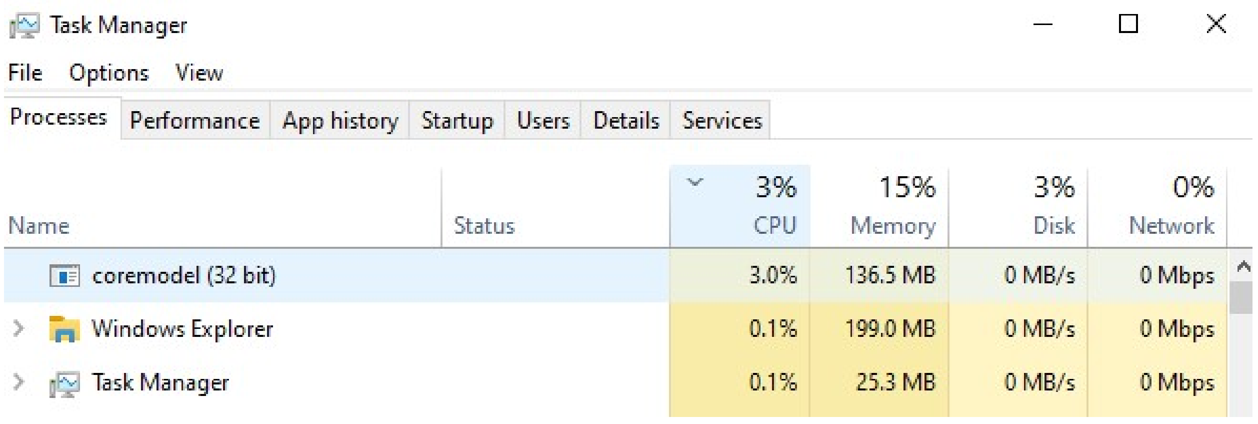
```
MyFiles <- read.csv(paste0(loc_path, "FileLists/Init_ForestCarbon.csv"))
makeFiles(lstFiles = MyFiles, path_basexmls = My_basePath, path_newxmls = My_newxmlPath,
    path_newvals = My_newvalsPath)
```

Find the newly created parameter files. Does the naming convention make sense? Open the newly created files to see whether the expected substitutions happened. How many timesteps are being run? Check the output pathway and file name near the bottom of the .xml - is it a valid pathway on your computer?

Run the new parameter files

```
`?`(runSortie)
# ex - run the first new parameter file
runSortie(paste0(My_newxmlPath, "SBS-01-FR01Inits-d.xml"), sortie_loc = 0)
```

If you want to check on whether the file is running, open your task manager and look for coremodel:



Name	Status	3% CPU	15% Memory	3% Disk	0% Network
coremodel (32 bit)		3.0%	136.5 MB	0 MB/s	0 Mbps
Windows Explorer		0.1%	199.0 MB	0 MB/s	0 Mbps
Task Manager		0.1%	25.3 MB	0 MB/s	0 Mbps

Figure 3: check to make sure SORTIE is running in the background

We often run multiple SORTIE parameter files simultaneously through rsortie using parallel processing. We won't describe this function in detail here as it requires more testing (`runSortiePar()`), but it is part of our current rsortie development branch.

SORTIE outputs

Extract outputs in rsortie

We often use the SORTIE GUI to extract results as it is faster and more tested than the rsortie versions. The functions to process SORTIE outputs in rsortie are still in development, but here is an example of how to use them:

1. Extract files for the detailed output .tar.gz
2. Parse the resulting files into a dataframe and save it or continue to work with it in R

we recommend saving the file (dt_table) after step 2, as parsing is slow, where as reading in saved csvs is very fast

```r
out_path <- "D:/GitHub/rsortieWorkshopFiles/Example1/Outputs/"
plotID <- c("FR01", "FR02")  #list the plots you ran

# Use ExtractFiles to untar the detailed output file
if (dir.exists(paste0(out_path, "extracted"))) {
    # if there's already an extracted folder, don't extract
    # files again
    ListofExtractedFiles <- paste0(out_path, "extracted/", list.files(paste0(out_path,
        "extracted")))
    ListofExtractedFiles <- grep("csv.gz", ListofExtractedFiles,
        invert = TRUE, value = TRUE)
} else {
    # otherwise the function automatically creates it
    ListofExtractedFiles <- extractFiles(itype = 0, exname = out_path)
    ListofExtractedFiles <- grep("csv.gz", ListofExtractedFiles,
        invert = TRUE, value = TRUE)
}

for (pl in 1:length(plotID)) {
    # get the file names > 0
    a <- grep(paste0("det_", "([0-9]+)"), grep(plotID[pl], ListofExtractedFiles,
        value = TRUE), value = TRUE)
    dt_table <- data.table()
    for (ix in 1:length(a)) {
        print(paste0("parsing: ", a[ix]))
        iy <- strsplit(a[ix], ".xml.gz")[[1]]
        t <- parseXML(a[ix])
        dt <- as.data.table(t)
        dt[, `:=`(timestep = as.numeric(strsplit(iy, "det_")[[1]][2]),
            plotID = plotID[pl])]
        dt_table <- rbind(dt_table, dt, fill = TRUE)
    }
    write.csv(dt_table, paste0(out_path, plotID[pl], "run_outputs.csv"))
}
```

Batch extract outputs from SORTIE GUI

The alternative to extracting and parsing in rsortie is to batch extract detailed outputs using the GUI. We often use this as it can be significantly faster than extracting and parsing in R, especially when dealing with large stands or long simulations.

1. Tools > batch extract detailed output files
2. navigate the the detailed output file or files that you want to extract tree data from. Add files to batch and select Next
3. Select tables > Timestep tree writer and browse to the location where the files should be extracted to
4. Assign a root name for the extracted tree data select OK

Read outputs into R

After extracting (and/or extracting and parsing), the outputs will be in different formats depending on your approach. Here, you can create a data table that contains all trees in all stands for every timestep. Choose which code to run below based on whether you extracted SORTIE outputs using Batch Extract in the SORTIE GUI, or extract and parse in rsortie.

In both cases, we read all the trees for every timestep from the SORTIE simulation, but then we clip the outputs to a 1 ha plot in the centre of the run to reduce computation in the next steps.

```
out_path <- "D:/GitHub/rsortieWorkshopFiles/Example1/Outputs/"
plotID <- c("FR01","FR02")

###### IF READING IN OUTPUTS PARSED IN R ########
# all years are contained within a csv if extracted
# and parsed using above code
out_names <- paste0(plotID,c("run_outputs.csv","run_outputs.csv"))

out_DT <- data.table()
for(pl in 1:length(out_names)){
  dt <- fread(paste0(out_path,out_names[pl]),header=T)
  #just keep 1ha of trees
  dt <- dt[X >50 & X <150 & Y>50 & Y <150]
  out_DT <- rbind(out_DT,dt,fill=TRUE)
}
setnames(out_DT, "tree_species", "Species")

\

###### IF READING IN OUTPUTS EXTRACTED IN SORTIE ########
#each year has it's own file if batch extract from SORTIE
out_names <- paste0("ext_SBS-01-",plotID,c("Inits-d_det_"))

out_DT <- data.table()
for(pl in 1:length(plotID)){
  PlID <- plotID[pl]
  yrs <- seq(0,4) #hard coded for 4 years
  for(i in 1:length(yrs)){
    dt <- fread(paste0(out_path,out_names[pl],yrs[i]), sep="\t",
                header=T,na.strings = "--", skip=1)
    dt[,':='(timestep = yrs[i],plotID = PlID)]
    #just keep 1ha of trees
    dt <- dt[X >50 & X <150 & Y>50 & Y <150]
    out_DT <- rbind(out_DT,dt,fill=TRUE)
  }
}
#############
#have a look at the data table
out_DT
```

Example graphs from SORTIE output

As a last step, explore how you can use SORTIE outputs to calculate live tree carbon! For this you need to source functions stored in a separate R script.

```
source("D:/Github/rsortieWorkshopFiles/Example1/R/CarbonFunctions.R")

# Format data table to match biomass calculation code
out_DT <- out_DT[, `:=`(SO_sp, ifelse(Species == "Western_Larch",
    "Lw", ifelse(Species == "Douglas_Fir", "Fd", ifelse(Species ==
        "Subalpine_Fir", "Bl", ifelse(Species == "Interior_Spruce",
        "Sx", ifelse(Species == "Lodgepole_Pine", "Pl", ifelse(Species ==
            "Trembling_Aspen", "At", ifelse(Species == "Black_Cottonwood",
            "Ac", ifelse(Species == "Paper_Birch", "Ep", Species)))))))))]
# Format and just keep live trees
out_DT[, `:=`(Tree_class, ifelse(Type == "Seedling" | Type ==
    "Sapling" | Type == "Adult", "L", "D"))]
Trees_SO <- out_DT[!is.na(DBH) & Height > 1.3 | !is.na(DBH) &
    is.na(Height)]
Trees_SO[, `:=`(Tree_class, ifelse(Tree_class == "L", 2, 4))]

# Calculate carbon in live trees by timestep and plot ID
g <- vector()
for (i in 1:nrow(Trees_SO)) {
    g[i] <- TreeCarbonFN(Species = Trees_SO[i, SO_sp], DBH = Trees_SO[i,
        DBH], HT = Trees_SO[i, Height], Tree_class = Trees_SO[i,
        Tree_class])
}
# xkg/(100x100m) = xkg/1ha * 1Mg/1000kg = x Mg/1000
Trees_SO[, `:=`(CarbonPerHa, g/1000)]
Trees_SO_live <- Trees_SO[!is.na(CarbonPerHa)]
FR_Sor_trees <- Trees_SO_live[, .(LiveCperHa = sum(CarbonPerHa)),
    by = c("plotID", "timestep")]

# Graph the trends over time in carbon from SORTIE
library(ggplot2)

ggplot(FR_Sor_trees, aes(x = timestep, y = LiveCperHa, colour = plotID)) +
    geom_point() + theme_minimal() + ylab(expression("Carbon Mg/ha")) +
    xlab("Time since fire") + ylim(0, 200) + theme(strip.text.x = element_text(face = "bold"))
```