training

E. McIntire C. Barros

Invalid Date

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

```
1 + 1
```

```
[1] 2
```

1 Introduction

2 Castor workflow with `setupProject`

Castor is a forest and land-use model used to simulate forest harvest and its effects on multiple forest values, which include not only timber, but also habitat for several wildlife (e.g. caribou, fisher). It is a fully open-source model, implemented in `SpaDES`, developed and maintained by researchers at the Forest Analysis and Inventory Branch, BC Ministry of Forests.

In this chapter, we demonstrate how to set up a Castor workflow using `setupProject` from the `SpaDES.project` package. The code was adapted from this Castor scenario, with some modifications to streamline the code and accommodate the use of `SpaDES.project` functions[1].

A bare-bones version of this example is also available in this .R script

2.1 Workflow setup

```r
## install/load necessary packages
repos <- c("predictiveecology.r-universe.dev", getOption("repos"))
install.packages(c("remotes", "DiagrammeR"), repos = repos)
```

```
package 'remotes' successfully unpacked and MD5 sums checked
package 'DiagrammeR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\cbarros\AppData\Local\Temp\Rtmp6vk4AY\downloaded_packages
```

```r
remotes::install_github("PredictiveEcology/SpaDES.project@transition")   ## to deal with mod
library(SpaDES.project)

## get Castor modules
setupFunctions(paths = list("projectPath" = "~/"),
               functions = c("PredictiveEcology/PredictiveEcology.org@training-book/tutos/ca
```

[1] `SpaDES.project` is currently being adapted to deal with modules nested in folders of GitHub repositories (instead of living in their own GitHub repositories), as is the case of Castor modules. Hence, the code in this example is subject to changes in the near future.

```
                             "PredictiveEcology/PredictiveEcology.org@training-book/tutos/cas
               overwrite = TRUE)
outMod <- getCastorModulesAndDB(paths = list("modulePath" = "~/tutos/castorExample/modules/"
                                             "projectPath" = "~/tutos/castorExample"),
                                modules = c("dataCastor",
                                            "growingStockCastor",
                                            "forestryCastor",
                                            "blockingCastor"),
                                dbURL = "https://drive.google.com/file/d/1-2POunzC7aFbkKK5Le
                                dbPath = "R/scenarios/comparison_stsm")

## set up the workflow paths, dependencies and modules
## as well as simulation parameters, (some) inputs and outputs
out <- setupProject(
  paths = list("inputPath" = "modules/forestryCastor/inputs",
               "outputPath" = "/R/scenarios/comparison_stsm/outputs",
               "modulePath" = "modules/",
               "cachePath" = "modules/forestryCastor",
               "projectPath" = "~/tutos/castorExample"),
  modules = names(outMod$modules),
  functions = "bcgov/castor@main/R/functions/R_Postgres.R",
  ## install and load
  require = "dplyr",
  ## install but don't load these:
  packages = c(
    "DBI",
    "keyring",
    "rgdal",
    "RPostgreSQL",
    "sp",
    "terra"
  ),
  params = "params.R",
  times = list(start = 0, end = 20),
  outputs = {
    data.frame(objectName = c("harvestReport",
                              "growingStockReport"))
  },
  scenario = {
    data.table(name = "stsm_base_case",
               description = paste("Priority queue = oldest first. Adjacency constraint",
                                   "= None. Includes roads (mst) and blocks (pre).",
                                   "Harvest flow = 147,300 m3/year in decade 1, 133,500",
                                   "m3/year in decade 2, 132,300 m3/year in decades 3 to",
                                   "14 and 135,400 m3/year in decades 15 to 25.",
                                   "Minimum harvest age = 80 and minimum harvest volume = 150
  },
  harvestFlow = {
    rbindlist(list(data.table(compartment = "tsa99",
                              partition = ' age > 79 AND vol > 149 ',
                              period = rep( seq (from = 1,
                                                 to = 1,
                                                 by = 1),
                                            1),
                              flow = 1473000,
                              partition_type = 'live'),
                   data.table(compartment = "tsa99",
                              partition = ' age > 79 AND vol > 149 ',
                              period = rep( seq (from = 2,
                                                 to = 2,
                                                 by = 1),
                                            1),
                              flow = 1335000,
                              partition_type = 'live'),
                   data.table(compartment = "tsa99",
                              partition = ' age > 79 AND vol > 149 ',
                              period = rep( seq (from = 3,
                                                 to = 14,
                                                 by = 1),
                                            1),
                              flow = 1323000,
                              partition_type = 'live'),
                   data.table(compartment = "tsa99",
                              partition = ' age > 79 AND vol > 149 ',
                              period = rep( seq (from = 15,
                                                 to = 25,
                                                 by = 1),
                                            1),
                              flow = 1354000,
                              partition_type = 'live')
    ))
  },
  Restart = TRUE
)
```

2.2 Initialise the model and inspect `simList`

`setupProject()` returns a names list containing values that can be passed to `simInit()` arguments.

We use `do.call()` to pass the whole list of arguments to `simInit`.

```r
## initialize simulation
castorInit <- do.call(SpaDES.core::simInit, out)
```

Another (more verbose) option would to call `simInit` directly:

```r
## initialize simulation
castorInit <- SpaDES.core::simInit(
  times = out$times,
  params = out$params,
  modules = out$modules,
  objects = list(scenario = out$scenario,
                 harvestFlow = out$harvestFlow)
)
```

Use the following functions to access workflow/model properties. `events()`, for instance will output the scheduled events, which at this point are only the `init` events of each module as you can see in the output below.

```r
## inspect the `simList`
SpaDES.core::params(castorInit)
SpaDES.core::inputs(castorInit)
SpaDES.core::outputs(castorInit)
SpaDES.core::times(castorInit)

## scheduled events
SpaDES.core::events(castorInit)
```

```
   eventTime         moduleName eventType eventPriority
       <num>             <char>    <char>         <num>
1:         0         checkpoint      init             0
2:         0               save      init             0
3:         0           progress      init             0
4:         0               load      init             0
5:         0         dataCastor      init             1
```

```
6:          0 growingStockCastor      init             1
7:          0     blockingCastor      init             1
8:          0     forestryCastor      init             1
```

2.3 Visualize the workflow

`moduleDiagram()` and `objectDiagram()` are great to visualise how each module interacts with the other. Recall that these interactions arise from object "exchanges" between modules, which are deduced by `simInit()` from module metadata (Figure **??**) – i.e., if a module's inputs are another's outputs, then the first module will follow the second.

```
SpaDES.core::moduleDiagram(castorInit)
SpaDES.core::objectDiagram(castorInit)
```

2.4 Run simulation

`spades()` runs the simulation, beginning with the execution of the `init` events. Notice how the result of `outputs()` differs from previously.

```
castorSim <- SpaDES.core::spades(castorInit)

## we now have outputs
SpaDES.core::outputs(castorSim)
```

```
          objectName
1      harvestReport
2 growingStockReport
                                                                     file     fun
1      C:/R/scenarios/comparison_stsm/outputs/harvestReport_year20.rds saveRDS
2 C:/R/scenarios/comparison_stsm/outputs/growingStockReport_year20.rds saveRDS
  package saveTime saved arguments
1    base       20  TRUE        NA
2    base       20  TRUE        NA
```

`completed(castorSim)` shows the chaining of events that was produced and run by `spades()`. The sequence of steps in the workflow therefore arises from each module's events and their scheduling, rather than being explicitly imposed by the user.

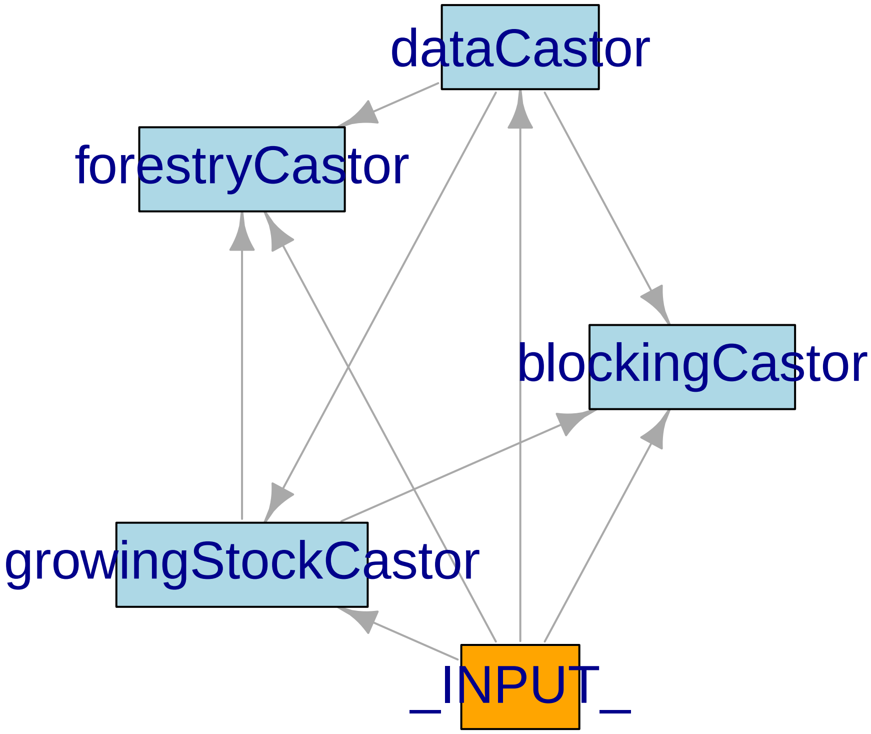

Figure 2.1: Diagram of module connections.

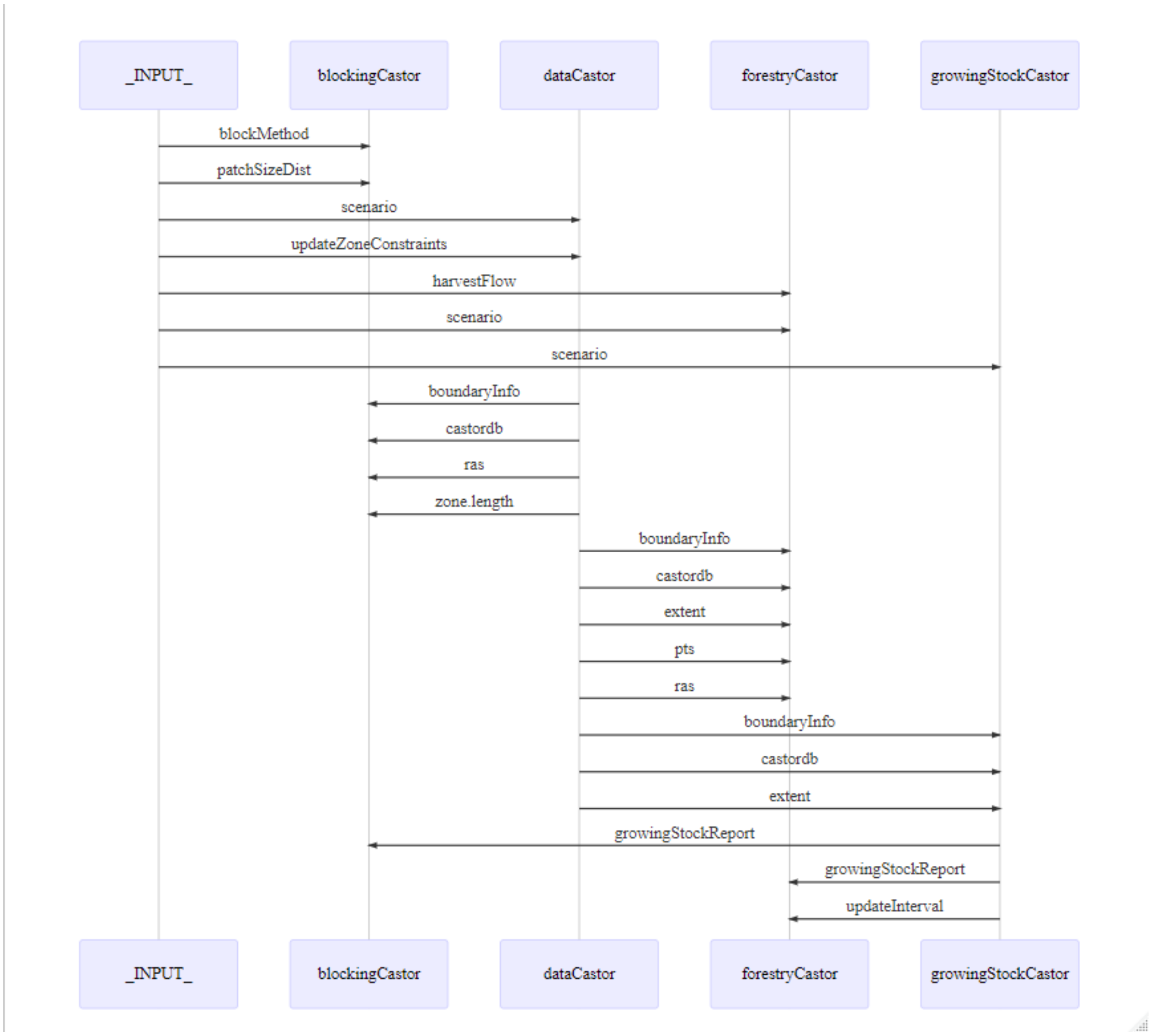

Figure 2.2: Diagram of module inter-dependencies with object names.

```r
SpaDES.core::completed(castorSim)
```

```
    eventTime          moduleName           eventType eventPriority
        <num>              <char>              <char>         <num>
 1:         0          checkpoint                init             0
 2:         0                save                init             0
 3:         0            progress                init             0
....
```

We suggest omitting the `blockingCastor` module in `setupProject()` and rerunning the workflow again to see how `spades` is capable of re-generating a new workflow with little effort from the user.

```r
modules <- c("dataCastor",
             "growingStockCastor",
             "forestryCastor")

out <- setupProject(
  paths = list("inputPath" = "modules/forestryCastor/inputs",
               "outputPath" = "/R/scenarios/comparison_stsm/outputs",
               "modulePath" = "modules/",
               "cachePath" = "modules/forestryCastor",
               "projectPath" = "~/tutos/castorExample/"),
  modules = modules,
  functions = "bcgov/castor@main/R/functions/R_Postgres.R",
  ## install and load
  require = "dplyr",
  ## install but don't load these:
  packages = c(
    "DBI",
    "keyring",
    "rgdal",
    "RPostgreSQL",
    "sp",
    "terra"
  ),
  params = "params.R",
  times = list(start = 0, end = 20),
  outputs = {
    data.frame(objectName = c("harvestReport",
                              "growingStockReport"))
```

```
},
scenario = {
  data.table(name = "stsm_base_case",
             description = paste("Priority queue = oldest first. Adjacency constraint",
                                 "= None. Includes roads (mst) and blocks (pre).",
                                 "Harvest flow = 147,300 m3/year in decade 1, 133,500",
                                 "m3/year in decade 2, 132,300 m3/year in decades 3 to",
                                 "14 and 135,400 m3/year in decades 15 to 25.",
                                 "Minimum harvest age = 80 and minimum harvest volume = 150
},
harvestFlow = {
  rbindlist(list(data.table(compartment = "tsa99",
                            partition = ' age > 79 AND vol > 149 ',
                            period = rep( seq (from = 1,
                                               to = 1,
                                               by = 1),
                                          1),
                            flow = 1473000,
                            partition_type = 'live'),
                 data.table(compartment = "tsa99",
                            partition = ' age > 79 AND vol > 149 ',
                            period = rep( seq (from = 2,
                                               to = 2,
                                               by = 1),
                                          1),
                            flow = 1335000,
                            partition_type = 'live'),
                 data.table(compartment = "tsa99",
                            partition = ' age > 79 AND vol > 149 ',
                            period = rep( seq (from = 3,
                                               to = 14,
                                               by = 1),
                                          1),
                            flow = 1323000,
                            partition_type = 'live'),
                 data.table(compartment = "tsa99",
                            partition = ' age > 79 AND vol > 149 ',
                            period = rep( seq (from = 15,
                                               to = 25,
                                               by = 1),
                                          1),
                            flow = 1354000,
                              partition_type = 'live')
    ))
  },
  Restart = TRUE
)

## initialize and run simulation in one go
castorSim2 <- do.call(SpaDES.core::simInitAndSpades, out)
```