Similar to our other outputs from Habplan, we can read in the output flow files and plot them. We are going to compare the results to those of the Metropolis Hastings algorithm from the third run above.

```
#Read in the flow files
flow1 <- read.csv("./saveFlow1", sep="", header = F)
flow2 <- read.csv("./saveFlow2", sep="", header = F)


#Input the flow file into the function, and number of years
flowPlot(flow.data = flow1, nyear = 35)
```

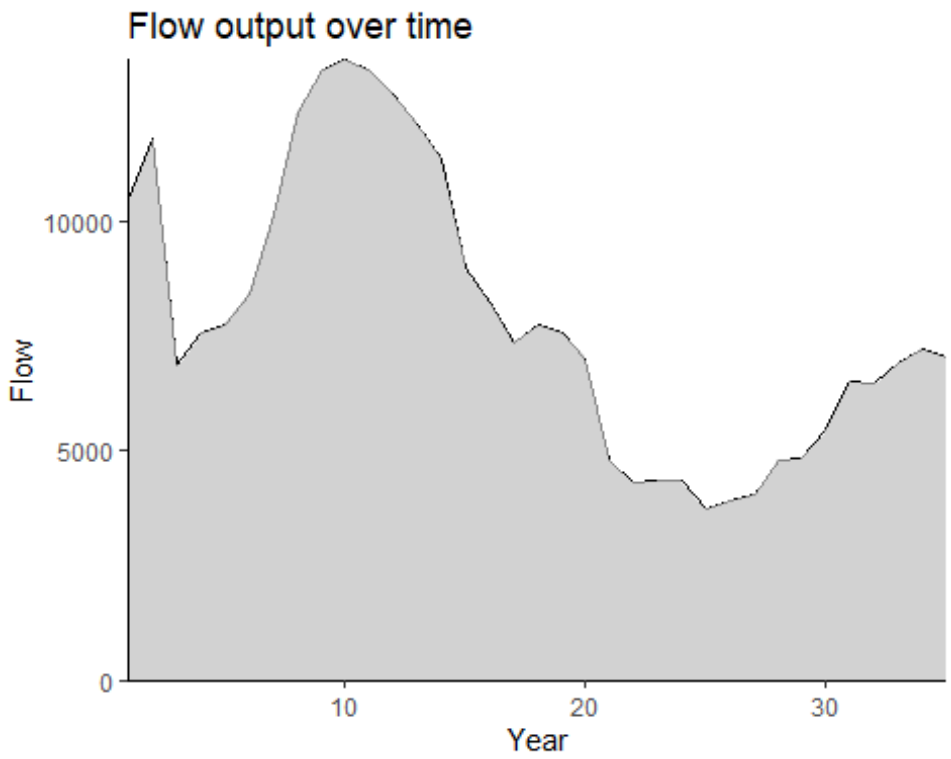

Figure 30. The amount of flow (habitat area) attained over time as a result of the suggested Habplan management schedule, based on Habplan linear programming.

```
#Input the flow file into the function, and number of years
flowPlot(flow.data = flow2, nyear = 35)
```

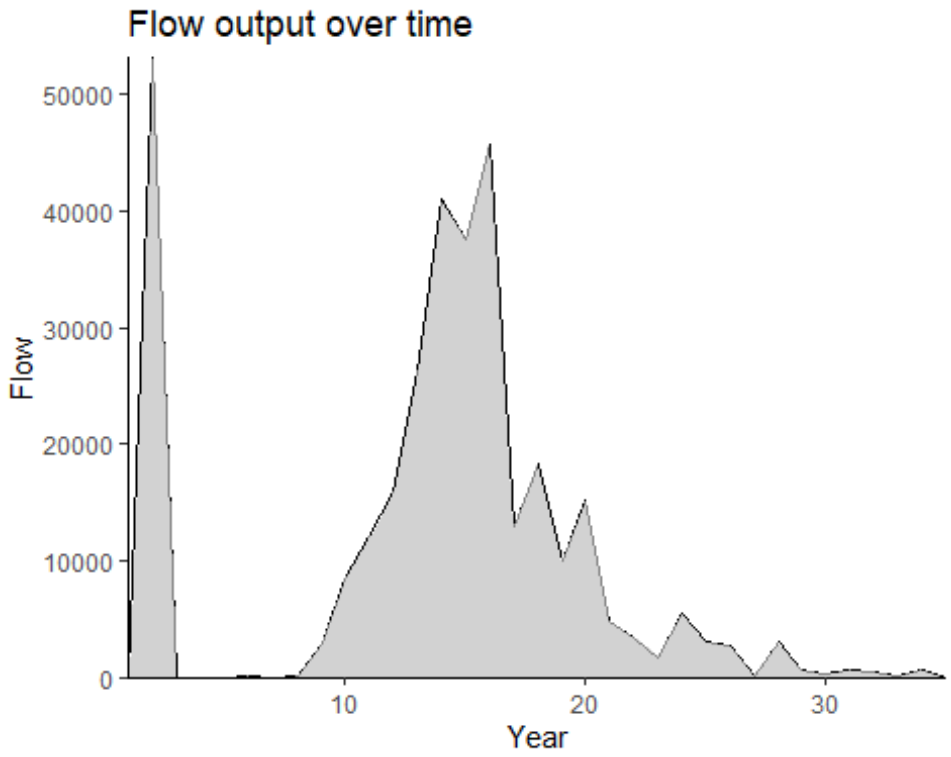

Figure 31. The amount of flow (harvested pine pulpwood) attained over time as a result of the suggested Habplan management schedule, based on Habplan linear programming.

Now that we have looked at the flows individually, we will use the *comPlot* function to compare the original Habplan runs with the LP Solver.

```
#Let's compare the flows across runs.
#For this example, we have saved the flow files in different folders
flow1 <- read_csv("./Run_2/saveFlow1", col_names = F)
#> Rows: 505 Columns: 73
#> ── Column specification
────────────────────────────────────────────────────────
#> Delimiter: ","
#> chr  (1): X2
#> dbl (72): X1, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ℹ Use `spec()` to retrieve the full column specification for this data.
#> ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
flow2 <- read_csv("./Run_3/saveFlow1", col_names = F)
#> Rows: 1010 Columns: 73
#> ── Column specification
────────────────────────────────────────────────────────
#> Delimiter: ","
```

```
#> chr  (1): X2
#> dbl (72): X1, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ⓘ Use `spec()` to retrieve the full column specification for this data.
#> ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
flow3 <- read_csv("./Run_LP/saveFlow1", col_names = F)
#> Rows: 505 Columns: 73
#> ── Column specification
────────────────────────────────────────────────────────
#> Delimiter: ","
#> chr  (1): X2
#> dbl (72): X1, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ⓘ Use `spec()` to retrieve the full column specification for this data.
#> ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
#flow4 <- read.csv("./saveFlow4", sep="")

#Run function with new flows
comPlot(flow.data.1 = flow1, flow.data.2 = flow2,
        flow.data.3 = flow3, nyear = 35)
```

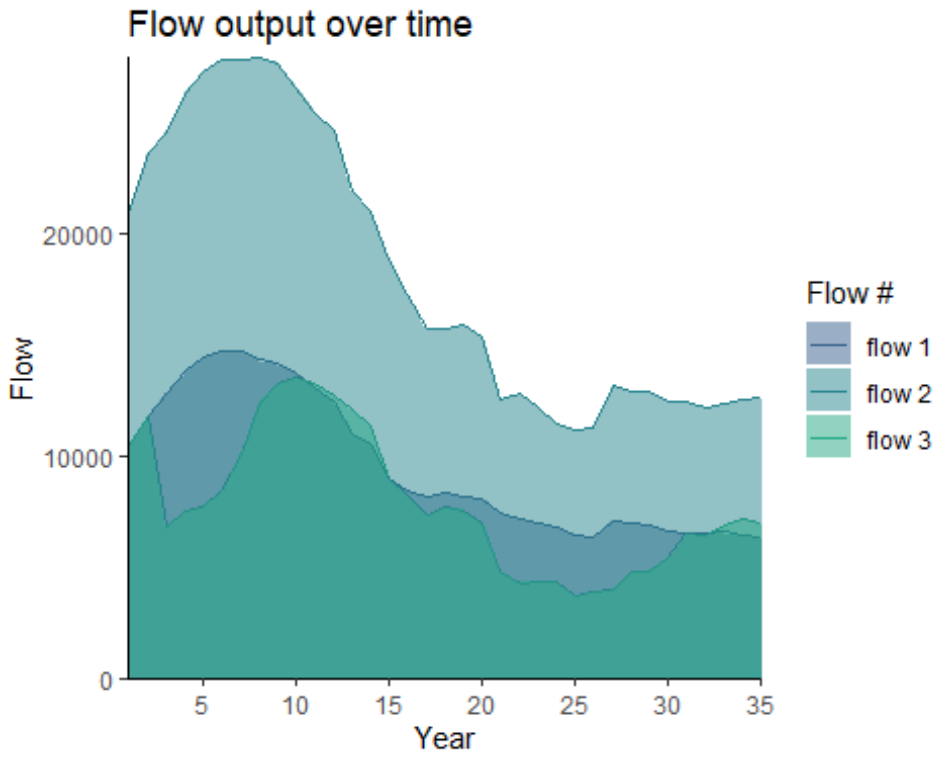

Figure 32. The amount of flow (habitat area) attained over time as a result of the three separate Habplan management schedule (flow1, flow2, and flow3 [linear programming]). Each flow is represented by a different shade of blue.

As can be seen from the above example, the suggested management regimes from the original Habplan run and the LP Solver can be very different. That's why it is important to look at each forestry problem using multiple methods and parameters, and plot them across time to find the "best" schedule.

Function: Carbon data conversion: *CarbConvert*

In the member-only HabplanR version, we have also incorporated a new function for converting Carbon data into Habplan flow files, similar to the *HabConvert* function introduced earlier in the vignette.

We need to provide the custom function with one argument only:

1. std.data: the flow information in csv format

We will read in the stand data with the carbon information embedded within.

```
#Load in stand and corresponding Carbon data
std.data <- read_csv("./HSM_OUT_FINAL_MerchStds_wCarbon.csv")
```

Now we can run the *CarbConvert* function. We only need to include our new std.data file for the function to run:

```
#Run function
carb.flow <- CarbonConvert(std.data = std.data)
```

The above function will save a flow file called "Carbon_data.dat" in our working directory

We will use a previously highlighted function of HabplanR, *WriteProj* to create a project file which can be loaded into Habplan to set parameters and other configurations. The only arguments needed for the actual function to work, are the object names for each flow component which needs to be included.

Since Habplan inputs allow for customization, we provide script which will allow users the same level of input manipulation. For our carbon workflow, we will only be working with one flow component: carbon.

```
#Overarching information for habplan run:
#Get the number of stands for later
npoly <- length(unique(std.data$StandKey))
#Configuration - see Habplan Manual
config <- "1,0,0,0,0,0"
#wd is the working directory where Habplan and data are stored
wd <- "FILEPATH HERE"
```

```
#iter is the number of iterations needed for the Habplan run
iter <- "10000"


f1.file <- "Carbon_data.dat"
f1.bygone <- ""
f1.time0 <- "100"
f1.goal0 <- "0.1"
f1.thlo <- "100"
f1.thhi <- "500"
f1.goalplus <- ".05"
f1.goalf <- "0.5"
f1.slope <- "0.0"
f1.weightf <- "1.0"
f1.weight0 <- "1.0"
f1.model <- "1,100;10,100;20,100;30,100"
f1.title <- "Carbon.dat"
#Combine f1 components for writing
f1.comp <- c('<flow title="F1 Component">',
             paste0('<file value="', f1.file, '" />'),
             paste0('<bygone value="', f1.bygone, '" />'),
             paste0('<time0 value="', f1.time0, '" />'),
             paste0('<goal0 value="', f1.goal0, '" />'),
             paste0('<threshLo value="', f1.thlo, '" />'),
             paste0('<threshHi value="', f1.thhi, '" />'),
             paste0('<goalPlus value="', f1.goalplus, '" />'),
             paste0('<goalF value="', f1.goalf, '" />'),
             paste0('<slope value="', f1.slope, '" />'),
             paste0('<weightF value="', f1.weightf, '" />'),
             paste0('<weight0 value="', f1.weight0, '" />'),
             paste0('<model value="', f1.model, '" />'),
             paste0('<title value="', f1.title, '" />'),
             '<bounds height="330" width="366" x="553" y="443" />',
             "</flow>")


#Provide each of these flow component objects to the function and run
writeProj(f1.comp = f1.comp)
```

Now we have a project file saved to our working directory. Let's open Habplan and see if this has worked. The Habplan program needs to be stored in the working directory that was assigned earlier, so that it can be run from RStudio below.

IMPORTANT - MUST RENAME OR MOVE SAVEFLOW1 AND SAVESCHED FROM WORKING DIRECTORY IF YOU DO NOT WANT THE FILES TO BE DELETED

```
#Open Habplan (if run from here, R functionality will cease until Habplan is
closed)
shell("h", wait=TRUE)
```

After running Habplan, we will have a carbon flow saved to our working directory. The option still exists to interactively watch the charts in a Habplan window. However, we provide a function to visualize each flow individually.

```
#Read in the Carbon flow file
flow1 <- read.csv("./saveFlow1", sep="", header = F)
```

IMPORTANT - if the flow1 object has more rows than number of stands something went wrong. Most likely, the saveFlow1 file was not removed from the working directory before Habplan was run.

```
#Input the flow file into the function, and number of years
flowPlot(flow.data = flow1, nyear = 36) #Carbon dataset has 36 years, not 35.
```

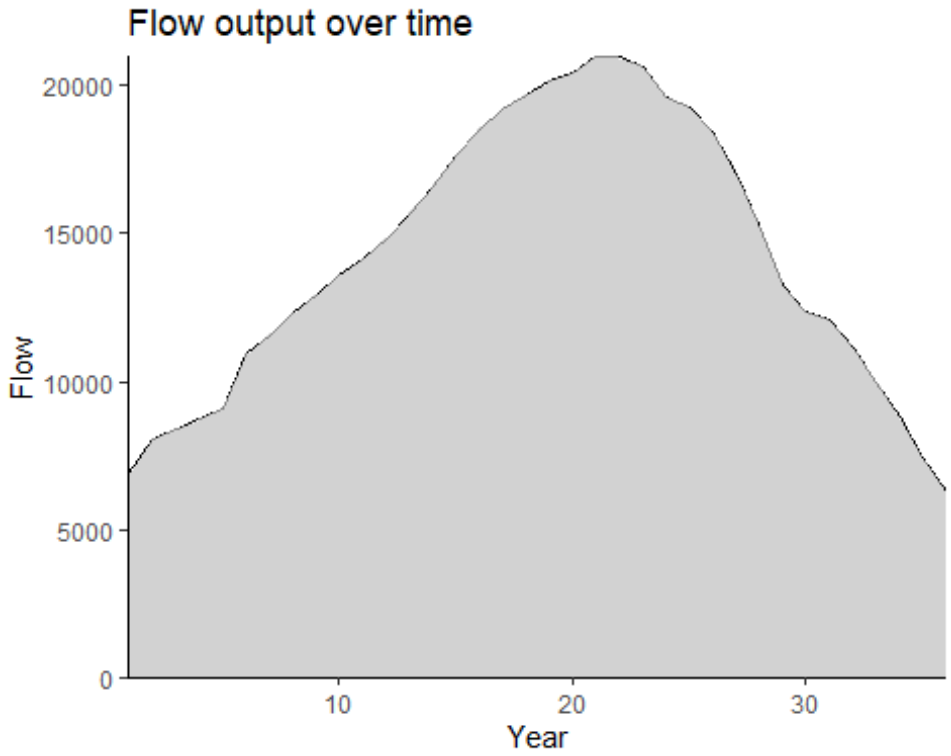

Figure 33. The amount of flow (sequestered carbon) attained over time as a result of the suggested Habplan management schedule.

Run 2

For this second run, we will maintain the overarching Habplan paramaters, but will now change the upper threshold (thhi), lower threshold (thlo), and model values.

```
f1.file <- "Carbon_data.dat"
f1.bygone <- ""
```

```
f1.time0 <- "5000"
f1.goal0 <- "0.1"
f1.thlo <- "5000"
f1.thhi <- "10000"
f1.goalplus <- ".05"
f1.goalf <- "0.5"
f1.slope <- "0.0"
f1.weightf <- "1.0"
f1.weight0 <- "1.0"
f1.model <- "1,10000;10,12000;20,12000;30,12000"
f1.title <- "Carbon.dat"
#Combine f1 components for writing
f1.comp <- c('<flow title="F1 Component">',
             paste0('<file value="', f1.file, '" />'),
             paste0('<bygone value="', f1.bygone, '" />'),
             paste0('<time0 value="', f1.time0, '" />'),
             paste0('<goal0 value="', f1.goal0, '" />'),
             paste0('<threshLo value="', f1.thlo, '" />'),
             paste0('<threshHi value="', f1.thhi, '" />'),
             paste0('<goalPlus value="', f1.goalplus, '" />'),
             paste0('<goalF value="', f1.goalf, '" />'),
             paste0('<slope value="', f1.slope, '" />'),
             paste0('<weightF value="', f1.weightf, '" />'),
             paste0('<weight0 value="', f1.weight0, '" />'),
             paste0('<model value="', f1.model, '" />'),
             paste0('<title value="', f1.title, '" />'),
             '<bounds height="330" width="366" x="553" y="443" />',
             "</flow>")


#Provide each of these flow component objects to the function and run
writeProj(f1.comp = f1.comp)
```

Now we have a project file saved to our working directory. Let's open Habplan and see if this has worked. The Habplan program needs to be stored in the working directory that was assigned earlier, so that it can be run from RStudio below.

IMPORTANT - MUST RENAME SAVEFLOW1 AND SAVESCHED FROM WORKING DIRECTORY IF YOU DO NOT WANT THE FILES TO BE DELETED

```
#Open Habplan (if run from here, R functionality will cease until Habplan is
closed)
shell("h", wait=TRUE)
```

After running Habplan, we will have a carbon flow saved to our working directory. The option still exists to interactively watch the charts in a Habplan window. However, we will visualize this flow using the flowPlot function.

```
#Read in the Carbon flow file
flow1 <- read.csv("./saveFlow1", sep="", header = F)
```

IMPORTANT - if the flow1 object has more rows than number of stands something went wrong. Most likely, the saveFlow1 file was not removed from the working directory before Habplan was run.

```
#Input the flow file into the function, and number of years
flowPlot(flow.data = flow1, nyear = 36)
```

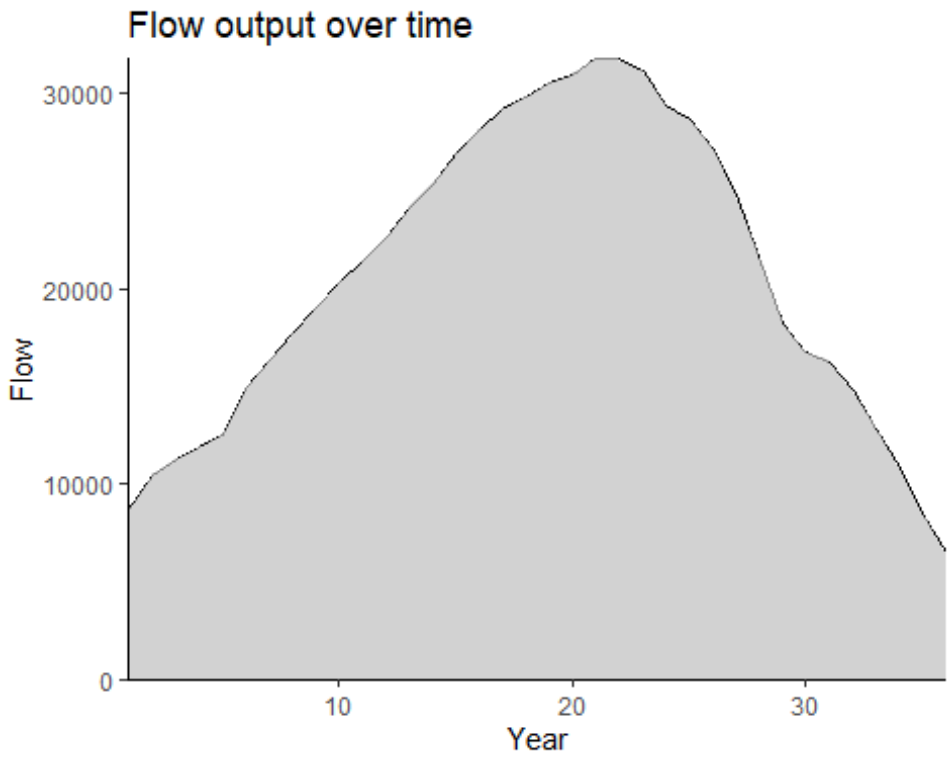

Figure 34. The amount of flow (sequestered carbon) attained over time as a result of the suggested Habplan management schedule.

Run 3

For this third run, we will maintain the overarching Habplan paramaters, but will now change the upper threshold (thhi), lower threshold (thlo), and model values.

```
f1.file <- "Carbon_data.dat"
f1.bygone <- ""
f1.time0 <- "10000"
f1.goal0 <- "0.1"
f1.thlo <- "3000"
f1.thhi <- "15000"
```

```
f1.goalplus <- ".05"
f1.goalf <- "0.5"
f1.slope <- "0.0"
f1.weightf <- "1.0"
f1.weight0 <- "1.0"
f1.model <- "1,15000;10,15000;20,15000;30,15000"
f1.title <- "Carbon.dat"
#Combine f1 components for writing
f1.comp <- c('<flow title="F1 Component">',
             paste0('<file value="', f1.file, '" />'),
             paste0('<bygone value="', f1.bygone, '" />'),
             paste0('<time0 value="', f1.time0, '" />'),
             paste0('<goal0 value="', f1.goal0, '" />'),
             paste0('<threshLo value="', f1.thlo, '" />'),
             paste0('<threshHi value="', f1.thhi, '" />'),
             paste0('<goalPlus value="', f1.goalplus, '" />'),
             paste0('<goalF value="', f1.goalf, '" />'),
             paste0('<slope value="', f1.slope, '" />'),
             paste0('<weightF value="', f1.weightf, '" />'),
             paste0('<weight0 value="', f1.weight0, '" />'),
             paste0('<model value="', f1.model, '" />'),
             paste0('<title value="', f1.title, '" />'),
             '<bounds height="330" width="366" x="553" y="443" />',
             "</flow>")


#Provide each of these flow component objects to the function and run
writeProj(f1.comp = f1.comp)
```

Now we have a project file saved to our working directory. Let's open Habplan and see if this has worked. The Habplan program needs to be stored in the working directory that was assigned earlier, so that it can be run from RStudio below.

IMPORTANT - MUST RENAME SAVEFLOW1 AND SAVESCHED FROM WORKING DIRECTORY IF YOU DO NOT WANT THE FILES TO BE DELETED

```
#Open Habplan (if run from here, R functionality will cease until Habplan is
closed)
shell("h", wait=TRUE)
```

After running Habplan, we will have a carbon flow saved to our working directory. The option still exists to interactively watch the charts in a Habplan window. However, we will visualize this flow using the flowPlot function.

```
#Read in the Carbon flow file
flow1 <- read.csv("./saveFlow1", sep="", header = F)
```

IMPORTANT - if the flow1 object has more rows than number of stands something went wrong. Most likely, the saveFlow1 file was not removed from the working directory before Habplan was run.

```
#Input the flow file into the function, and number of years
flowPlot(flow.data = flow1, nyear = 36)
```

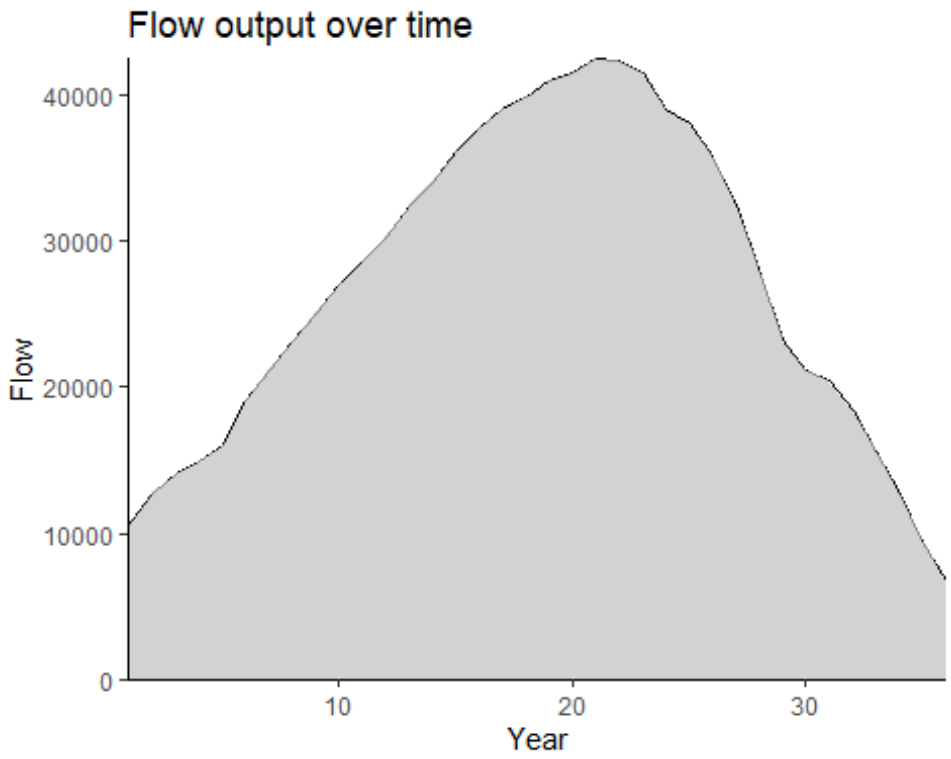

Figure 35. The amount of flow (sequestered carbon) attained over time as a result of the suggested Habplan management schedule.

The above will create a chart for one of the flows. However, it is also useful to be able to visualize the flows in relation to one another. We will therefore use the comPlot function, as we have demonstrated earlier.

This requires the same input as before, but this time we will introduce some of the other flows outputs. Current maximum of 3 flows.

```
#Let's look at these flows together
flow1 <- read_csv("./saveFlow1_1", col_names = F)
flow2 <- read_csv("./saveFlow1_2", col_names = F)
flow3 <- read_csv("./saveFlow1", col_names = F)
#flow4 <- read.csv("./saveFlow4", sep="")

#Run function with new flows
```

```
comPlot(flow.data.1 = flow1, flow.data.2 = flow2,
        flow.data.3 = flow3, nyear = 36) # 36 years in new Carbon data

#> Rows: 454 Columns: 75
#> ── Column specification
──────────────────────────────────────────────────────────
#> Delimiter: ","
#> chr  (1): X3
#> dbl (74): X1, X2, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ⓘ Use `spec()` to retrieve the full column specification for this data.
#> ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
#> Rows: 681 Columns: 75
#> ── Column specification
──────────────────────────────────────────────────────────
#> Delimiter: ","
#> chr  (1): X3
#> dbl (74): X1, X2, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ⓘ Use `spec()` to retrieve the full column specification for this data.
#> ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
#> Rows: 908 Columns: 75
#> ── Column specification
──────────────────────────────────────────────────────────
#> Delimiter: ","
#> chr  (1): X3
#> dbl (74): X1, X2, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15,
X16,...
#>
#> ⓘ Use `spec()` to retrieve the full column specification for this data.
#> ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

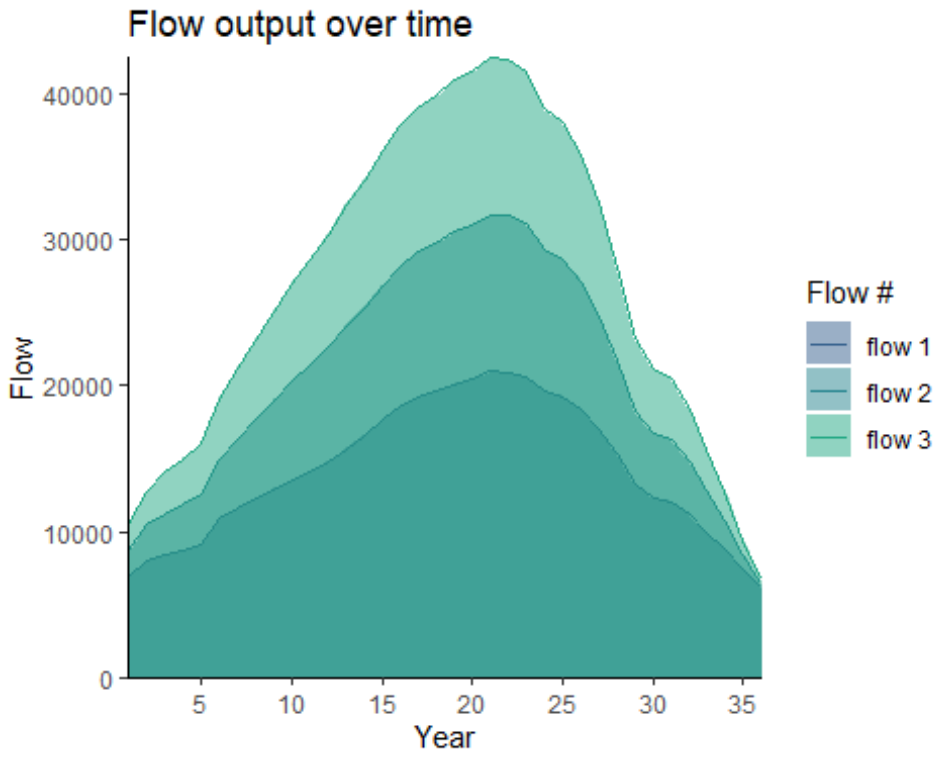

Figure 36. The amount of flow (sequestered carbon) attained over time as a result of the three separate Habplan management schedule (flow1, flow2, and flow3). Each flow is represented by a different shade of blue.

Summary

Above, we have provided very simple examples of how to use HabplanR to run a multi-objective forestry issue in Habplan. The values provided within the vignette are for demonstrative purposes only and do not represent real data. Real-world examples will likely have many more objectives to evaluate, alongside complex spatiotemporal considerations. If you encounter any issues with the HabplanR package, or require help with applying the functions to a novel dataset, feel free to contact Max Jones at maxdoltonjones@vt.edu. To access the member-only version, or for additional help, please contact Holly Munro at hmunro@ncasi.org.

Acknowledgements

We would like to thank the National Council for Air and Stream Improvement, Inc. for supporting the HabplanR package and vignette.

Appendix 1 - FVS variable names and descriptions

- RegimeKey: Unique number used to identify a specific management regime.
- Regime Activity: Descriptions of management regimes used within the Forest Vegetation Simulator growth model to project stand conditions. BA: Basal Area, LL: Loblolly Pine (Pinus taeda), TPA: Trees Per Acre.

- StandID: Unique forest stand ID.
- Year: Projected year by the growth model for that combination of forest stand and regime.
- Age: Age of forest stand.
- BA: Basal area.
- TPA: Trees per acre.
- Pine_Pulp_Tons: Tons of pine pulpwood.
- Pine_CNS_Tons: Tons of pine chip-n-saw.
- Pine_Saw_Tons: Tons of pine sawtimber.
- Hdwd_Pulp_Tons: Tons of hardwood pulpwood.
- Hdwd_Saw_Tons: Tons of hardwood sawtimber.
- Harv_P_Pulp_Tons: Harvested pine pulpwood (tons/acre).
- Harv_P_CNS_Tons: Harvested pine chip-n-saw (tons/acre).
- Harv_P_Saw_Tons: Harvested pine sawtimber (tons/acre).
- Harv_H_Pulp_Tons: Harvested hardwood pulpwood (tons/acre).
- Harv_H_Saw_Tons: Harvested hardwood sawtimber (tons/acre).
- HSI: Habitat Suitability Index.

Appendix 2 - List of HabplanR functions and brief description

- HSIcalc: Calculates HSI and inserts into dataset.
- HabConvert: Converts Forest Vegetation Simulator data (or similar) to Habplan flow file.
- WriteProj: Creates a project file with Habplan configurations/parameters.
- FlowPlot: Plots a single output flow file across time.
- ComPlot: Plots multiple flow files on a combined graph across time.
- StandSched: Adds the top schedule to stand shapefile.
- HabBlock: Creates block size constraint data (adjacency table).
- HabSpace: Spatial analysis of HSI flows.
- lpMPS: Create an MPS file to run the lp solver in Habplan.
- CarbonConvert: Converts growth model projection data with Carbon data to Habplan flow file.