We can compare each run to see how the yields have changed with each change we have made.

Function: multiple flow outputs - *ComPlot*

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 therefore provide another function: comPlot.

This requires the same input as before, but this time we will introduce some of the other flow outputs, which should have been saved into separate folders as described earlier. Current maximum of 3 flows.

```
#Read in the example flows:
flow1 <- read.csv("./Run_1/saveFlow1", sep="", header = F) #Files have been
saved in separate folders
flow2 <- read.csv("./Run_2/saveFlow1", sep="", header = F)
flow3 <- read.csv("./Run_3/saveFlow1", sep="", header = F)
#flow4 <- read.csv("./saveFlow4", sep="")

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

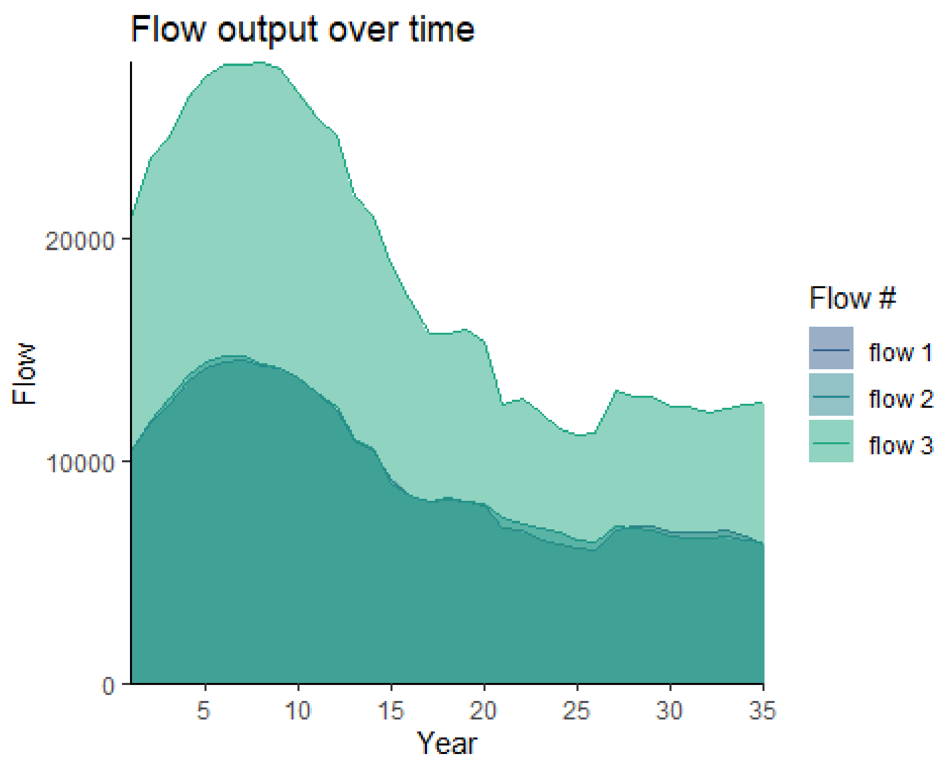

Figure 26. The amount of flow (habitat area) attained over time as a result of the three separate Habplan management schedule (flow1, flow2, and flow3). Each flow is represented by a different shade of blue.

```
#Read in the example flows:
flow1 <- read.csv("./Run_1/saveFlow2", sep="") #Files have been saved in
separate folders
flow2 <- read.csv("./Run_2/saveFlow2", sep="")
flow3 <- read.csv("./Run_3/saveFlow2", sep="")
#flow4 <- read.csv("./saveFlow4", sep="")

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

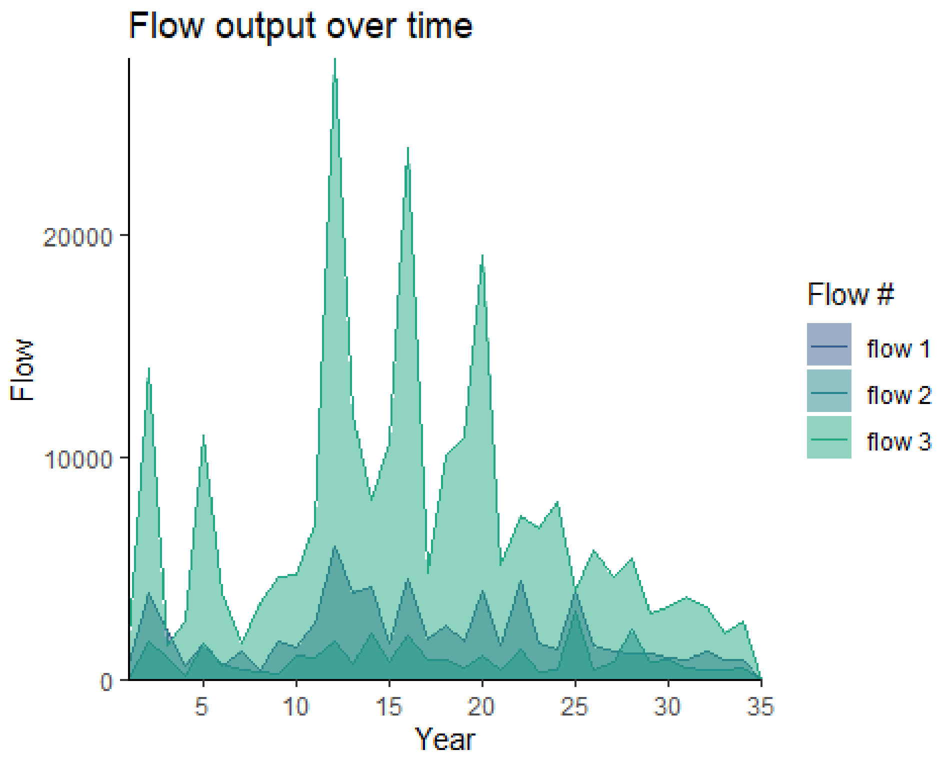

Figure 27. The amount of flow (harvested pine pulpwood) attained over time as a result of the three separate Habplan management schedule (flow1, flow2, and flow3). Each flow is represented by a different shade of blue.

By visualizing the flows in comparison to one another from each run, we can now clearly see the little differences found between the first and second run, but the changes we made to the third run have greatly increased our averge HSI and harvested pine pulpwood - ultimately providing more habitat for our species of interest, but maximizing yield and profits also.

Function: saving the top schedule to shapefile - *StandSched*

From this example, we are happy with the final run and will apply the recommended regime to each stand based on the output schedule from Habplan. Our next function pulls the information from the best schedule saved in the working directory (from Habplan), and attaches this to the stand shapefile.

As long as the file is still in the working directory (saveSched), then all we need to do is input the stand shapefile and run the function.

```
#Run function to add schedule to shapefile
standSched(site.shp)
```

If an error message comes up that states the following:

Error: unable to find an inherited method for function 'writeVector' for signature 'x = "data.frame", filename = "character"'

then run the next section in R to manually save the top schedule as a shapefile.

```
  #Read in saved schedule from habplan run
  sched <- read.csv("./saveSched")
  #Change column headings so that it's easier to work with
  colnames(sched) <- c("id", "StdID", "sched")
  #Remove any blank spaces that may have been brought in from the import
  sched$StdID <- gsub(" ", "", sched$StdID)
  #Add the new column
  new.shp <- merge(site.shp, sched, by = "StdID")
  #Save to the working directory
  writeVector(new.shp, "./Site_with_schedule.shp", filetype="ESRI Shapefile",
              overwrite=TRUE)
```

A new .shp file can now be found in the working directory titled Site_with_schedule.shp.

For the purposes of creating flow files, running Habplan, and comparing possible management schedules output from Habplan, you can now stop at this point of the vignette. However, next we provide two additional functions for creating custom HSI flow files, and including a blocksize flow subcomponent to the Habplan run.

Function: HSI calculation - *HSIcalc*

We are going to work through two examples of using basal area (BA) to calculate HSI. We have provided a function that will compute an HSI based on a user-specified formula, and combine this with our stand data. We provide examples of simple HSI formulas, but any formula can be used, so long as the inputs are present in the std.data dataset. We use a single independent variable (BA), but a function of multiple independent variables can also be used.

Our first example looks at a negative linear effect of BA on HSI. The equation will need to be provided in form of an R function.

```r
#Create function to apply equation for HSI
#Define the independent variable (x), slope (m), and y-intercept (b) of the
linear equation
hsi.func <- function(x = std.data$BA, m = -0.02, b = 2) {
  return(m * x + b)
  }
```

We can now insert this equation into the *HSIcalc* function, which calculates HSI from the minimum to maximum values of all x values in the std.data dataset, and then adds a new data column to our stand data for HSI values.

```r
#The function will create a new column with the HSI values
new.data <- HSIcalc(std.data, hsi.func)

#Summarise the HSI data
summary(new.data$HSI)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.05188 0.40419 0.61323 0.54688 0.73023 0.88072
#Plot data against BA to show relationship
ggplot(data = new.data) +
  geom_point(aes(x = BA, y = HSI, color = HSI),
            size = 3) +
  scale_color_viridis_c(option = "plasma") +
  theme_classic()
```

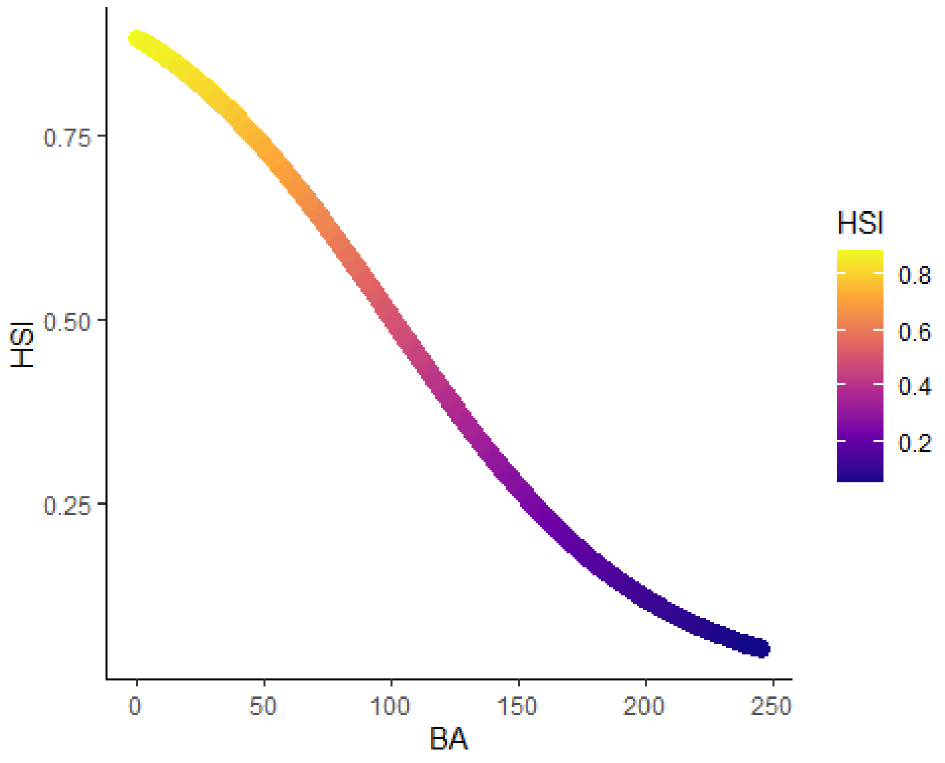

Figure 28. A negative linear effect of BA on HSI. Color represents change in HSI values.

Our second example looks at a quadratic relationship of BA on HSI. Again, the equation will need to be provided in form of an R function:

```
#Create function to apply equation for HSI
#Use a, b, and c to change shape of graph
hsi.func <- function(x = std.data$BA, a = -0.0006, b = 125, c = 3.5) {
  return(a * (x - b)^2 + c)
  }
```

We can now insert this equation into the *HSIcalc* function to add a column to our stand data for HSI values.

```
#The function will create a new column with the HSI values
new.data <- HSIcalc(std.data, hsi.func)

#Summarise the HSI data
summary(new.data$HSI)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
#> 0.002816 0.394109 0.774463 0.661753 0.938009 0.970688
#Plot data against BA to show relationship
ggplot(data = new.data) +
  geom_point(aes(x = BA, y = HSI, color = HSI),
          size = 3) +
  scale_color_viridis_c(option = "plasma") +
  theme_classic()
```

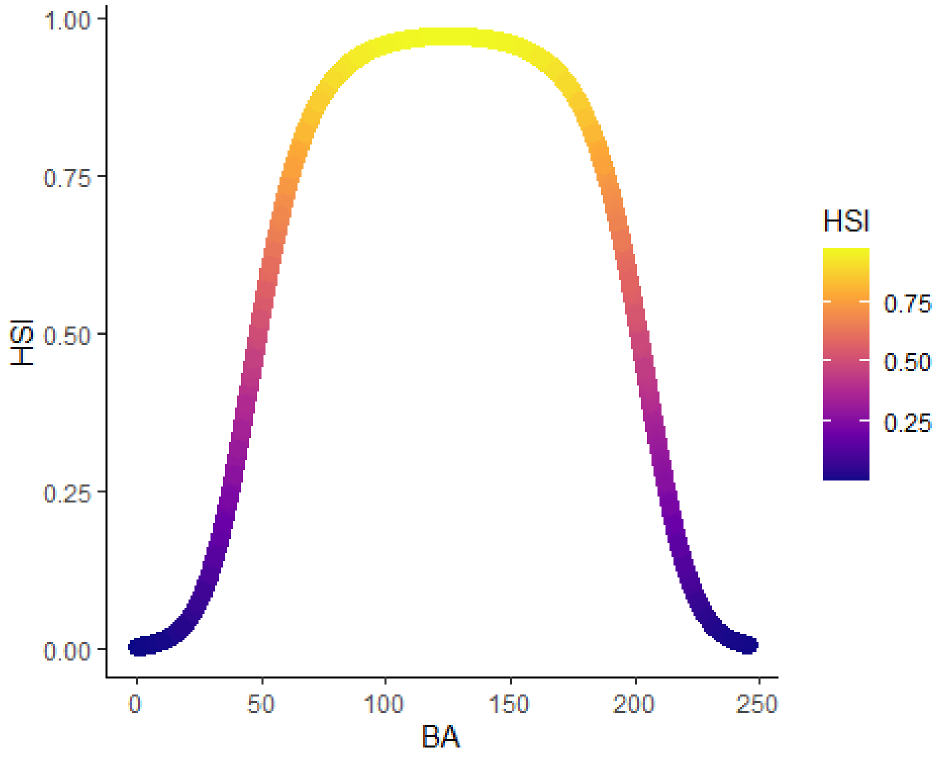

Figure 29. A quadratic effect of BA on HSI. Color represents change in HSI values.

Function: create block data - *HabBlock*

We will not be including the block data for the purpose of this vignette due to processing time. However, below we provide instructions for including a block size constraints to the Habplan run. Block size constraints are a sub-component of flow components. For a comprehensive guide to the purpose of block size constraints, please see section 9 of the Habplan user manual.

We can now run the function *HabBlock* from the HabplanR package. This takes the SpatVector shapefile, and stand data/info from earlier in the vignette and creates block data (defines which polygons are adjacent to each other). The block data will save directly to the working directory.

```
#HabBlock will create a block data file in the working directory
HabBlock(std.data = std.data, std.info = std.info, site.shp = site.shp,
         block.title = "block1")

#Since this block size component file is within the working directory
already,
#we just need to specify the file name below (b1.file), and input the
```

```
#specific parameters for the component.

#Info for f1 block size component (green up) ----
b1.file <- "block1.txt"
b1.notBlock <- "1-4" #which regimes do NOT contribute to block sizes
b1.greenUp <- "3" #3 year green up period
b1.min <- "1" #minimum allowable block size
b1.max <- "1000" #maximum allowable block size
b1.goal <- "1" #a value of 1.0 means all blocks must comply
b1.weight <- "1.0" #good practice to start this at 1
b1.title <- "block1.txt"
#Combine f1 components for writing
block1 <- c('<block title="BK1(1) Component">',
             paste0('<file value="', b1.file, '" />'),
             paste0('<notBlock value="', b1.notBlock, '" />'),
             paste0('<greenUp value="', b1.greenUp, '" />'),
             paste0('<min value="', b1.min, '" />'),
             paste0('<max value="', b1.max, '" />'),
             paste0('<goal value="', b1.goal, '" />'),
             paste0('<weight value="', b1.weight, '" />'),
             paste0('<title value="', b1.title, '" />'),
             '<bounds height="330" width="366" x="553" y="443" />',
             "</block>")
```

You can include as many block size components as there are flow components, which can be included as above but changing the block title and file to match the number of the flow component.

We can then include block size constraints to the project file using the *writeProj* function.

```
#Provide each of these flow component objects to the function and run
writeProj(f1.comp = f1.comp, block1 = block1, f2.comp= f2.comp)
```

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.