Dan Higgins
July 8, 2005

Environmental Niche Model Workflow

**Workflow for Determining Best GARP RuleSets (Single Species)**

The basic top-level tasks that must be done to determine the best GARP Rulesets for a single species are indicated in the conceptual workflow of Figure 1. Each Kepler composite actor shown in the workflow is identified by a Roman numeral (e.g. I, II, …). These tasks will be discussed first in general and then in detail.

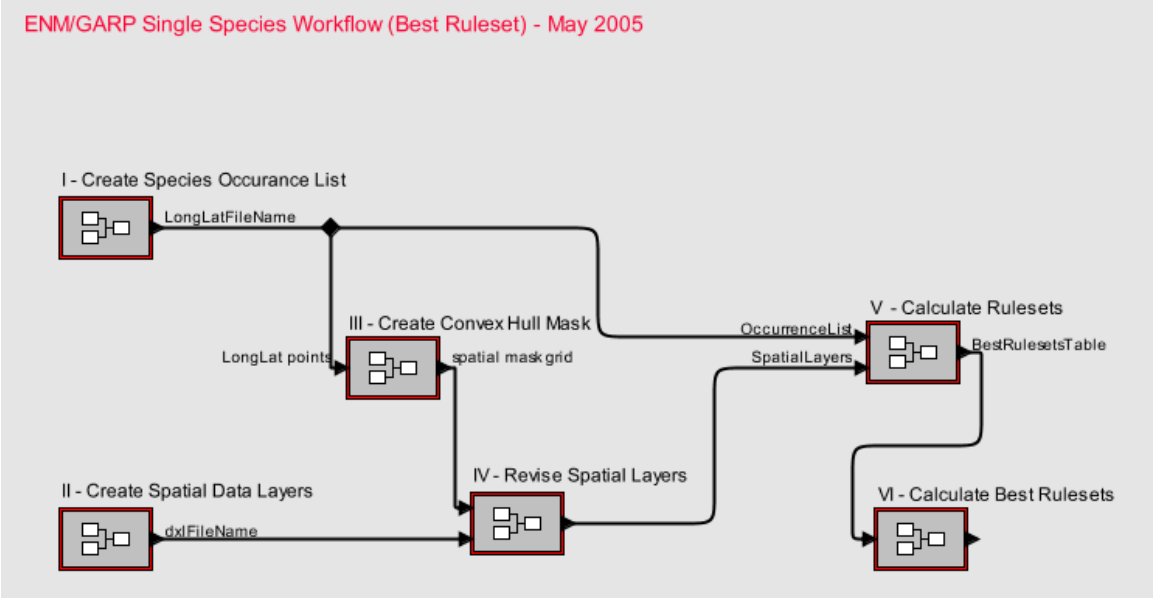


Figure 1 – Top Level Workflow Blocks

I – Create Species Occurrence List

This block will create a file which lists the longitude and latitude of known locations where the species is known to exist. Currently, this data just comes from a search of DigiR for the species name.

II – Create Spatial Data Layers

A set of spatial data layers must be created with climate and geographic data used in the GARP genetic algorithm. Hydro1K and IPCC data sources are currently the sources for this data. All layers to be used are converted to grids with the same resolution and extent.

III – Create Convex Hull Mask

A convex hull (a 'minimum' bounding polygon) that includes all input locations is calculated, perhaps scaled (i.e. made larger by a factor determined by the user), and then converted to a 'mask' grid indicating the region to be considered in the calculation.

IV – Revise Spatial Layers

The spatial layers used by GARP include a 'mask' layer. All points outside the mask are ignored in the calculation. In this case, the mask layer is replaced by the mask grid created from the species location convex hull. (Otherwise, the spatial layers are independent of the species locations.)

V – Calculate Rulesets

This block takes the occurrence list and the spatial layers and uses GARP to calculate 'rulesets' for predicted species locations. The rulesets predict species locations but may be different each time GARP is run (i.e. the results have statistical variations). The calculation is repeated a large number of times (~100-500). Each time, omission/commission values are calculated for the result using the input occurrence data (perhaps divided into training/testing sets). These values are saved, along with the ruleset, so that the 'best' rulesets can be determined.

VI – Calculate 'Best' Rulesets

The table of results generated in Block V is examined to determine the 'best' rulesets. Those 'best rulesets' are saved for use in further climate change calculations.

The general 'recipe' (provided by Ricardo Periera)used for determining these best rulesets is:

1) Get a set of presence data points (species occurrences - x, y coordinates) to test.
2) Project the GARP model onto geography (map generated by GarpProjection actor)
3) Overlay the presence points from item #1 onto the map generated on #2. The percentage of those points that fall in a pixed not predicted present is your OMISSION. Say, out of 100 points, only 45 fall on white pixels, the other 55 fall on black ones, your omission is 55% or 0.55.
4) Commission, when we don't have real absence points (our case) is the proportion of area predicted present with regard to the total area of interest, not counting masked pixels. So if 40% of the area is predicted present, your commission error is 40% or 0.40.
5) Then, select those GARP runs that show omission below a certain omission threshold, say 5 or 10%.
6) From those runs selected in #5, sort them by commission error, and then get the 50% of the models that are around the median value for commission. If you got, say, 20 models in item #5, now you have 10 models that make up your best subset of models.
7) Sum up the maps for the best subset of models in item #6, that is your final prediction map for your species.

Inside Block V – Calculate Rulesets

Block V of the overall workflow is the task where the GARP algorithm is iterated numerous times to get a statistical sample of predicted species distributions. The tasks

that must be carried out as part of this block are shown in Figure 2. (This is the workflow that is seen when one 'Looks Inside' the composite actor of Block V. Most of the actors here are composites; they are labeled with capital letters (A, B, …).

Every task in this workflow is repeated many times – i.e. inside a repeating loop. The 'Ramp' actor drives that loop. A 'Ramp' is basically just a counter where the user sets the minimum, maximum, and step size.

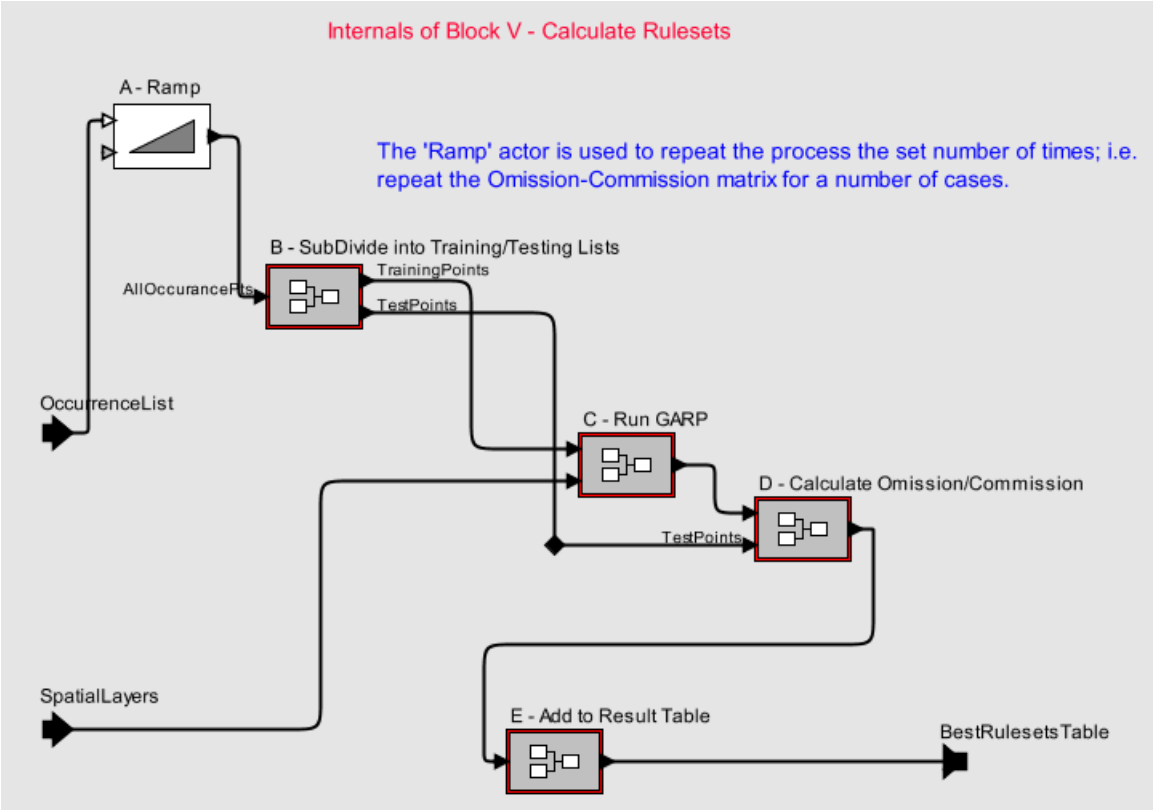


Figure 2 – Internals of Block V – Calculate Rulesets

On each iteration, the input occurrence data must be divided into two sets (Step B in Figure2), one used as training points for the GARP calculation and the second used as testing points to evaluate GARP results. Currently, this subdivision is simply random. GARP is then run using the training points. Omission and commission values are then calculated (Step D) and saved in a results table along with the ruleset generated by the GARP calculation (Step E). The ruleset is saved so that once the 'best' results are determined, the ruleset can be applied to generate new predictions for new climate layers.

## Implementation

Previous discussions have been purely conceptual – (meaning that the workflows didn't actually run!). An actual Kepler implementation of the workflow has been created and the top-level workflow is shown in Figure 3.

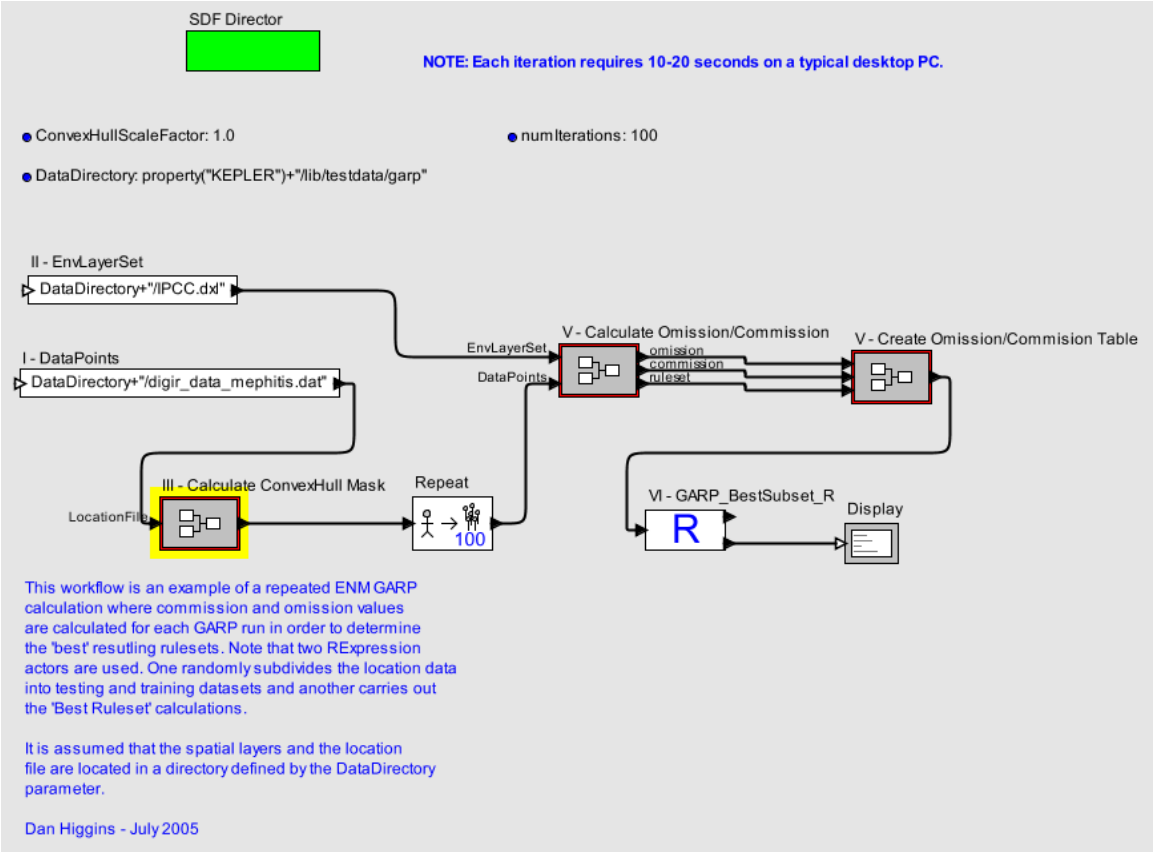



Figure 3 – Implementation of GARP Single Species Calculation. (This workflow is in the 'workflows/eco/' subdirectory under the name: "GARP_SingleSpecies_BestRuleSet_July2005.xml")

This implementation is organized slightly differently than the conceptual workflow but the same roman numerals are used to identify corresponding blocks.

Note that in this case, blocks I and II (creating known location lists and spatial layer information) have been reduced to simple file names. It is being assumed that various pre-processing has been done off-line and the resulting files are in the correct 'raw' format for GARP processing and have been stored in the indicated locations under the indicated names. Example files have been created from IPCC and Hydro1K data and saved in the locations shown. Details of how this was done will be described along with the associated workflows elsewhere, but the point here is that this is a separate step that can be done prior to the multiple GARP runs.

Block III calculates the convex hull and creates a mask file limiting calculations to the area enclosed but this convex hull. If you "Look Inside" the compositeActor III in Figure 3, you will find the workflow shown in Figure 4.

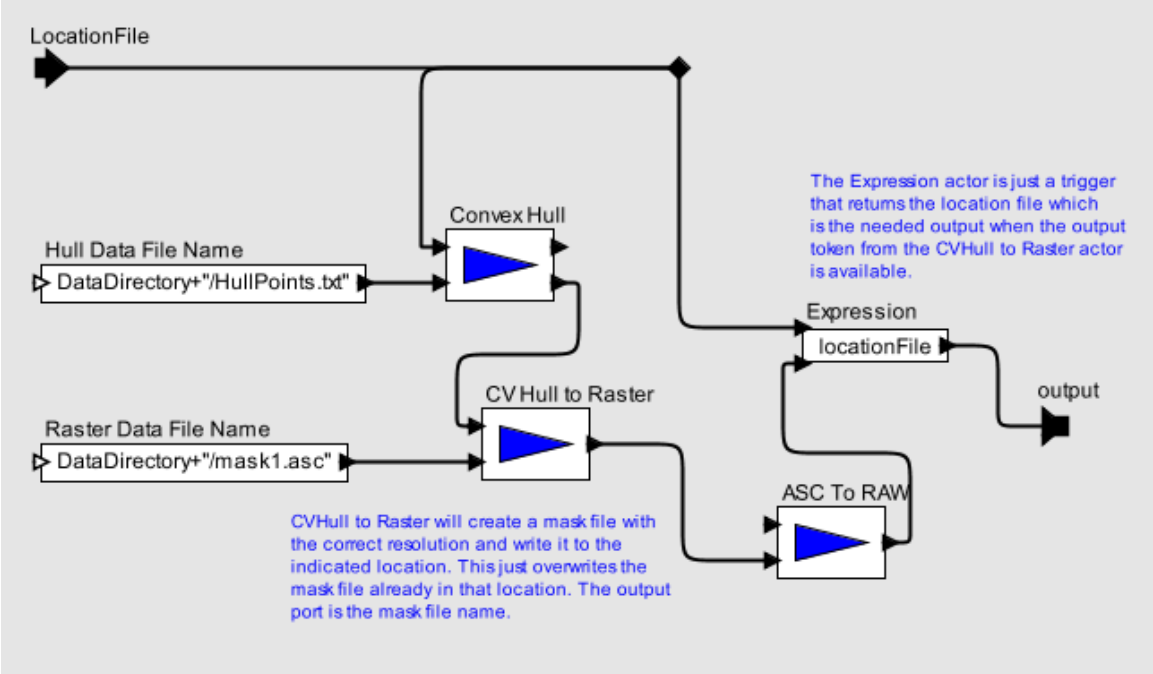



Figure 4 – Inside of Block III (Calculate Convex Hull Mask

This subworkflow calculates the Convex Hull as a set of points, saves that to a file, then uses that file to create a spatial grid 'mask' file. Note that the *.asc grid must be converted to the *.raw format used by GARP. In this case, the mask1.raw file is just overwritten so that the XML-based IPCC1.dxl file, which summarizes all the spatial layers, need not be changed.

Looking inside Block V – Calculate Omission/Commission shows the subworkflow of Figure 5. Although this part of the workflow basically just executes a single GARP calculation, there is some rather messy plumbing required. In particular, notice the "Ramp" actor. This actor is used to create an index that is added to the name of RuleSet file (e.g "RuleSet5.xml" when the Ramp count is '5'), so that each is distinct.

There is also an RExpression actor which is used to randomly subdivide the location datapoints for every iteration. The R script used is:

```
# read the original data
df <- read.table(fileName)
# get number of rows (i.e. number of lines)
lll <- length(df$V1)
fraction <- 0.5
# create a list of subset indices
sss <- sample(1:lll, size=(fraction*lll))
# create 2 subsets
df1 <- df[sss,]
# write output file
```

```
write.table(df1, file="FirstSubset.txt", sep="\t", row.names=FALSE,
col.names=FALSE)
trainingLocations <- paste(getwd(),"/FirstSubset.txt",sep="")
trainingLocations
df2 <- df[-(sss),]
# write output file
write.table(df2, file="SecondSubset.txt", sep="\t", row.names=FALSE,
col.names=FALSE)
testingLocations <- paste(getwd(),"/SecondSubset.txt",sep="")
df1
df2
```

The primary command that creates the random samples is '`sample`'. The size of each sample is determined by the '`fraction`' variable that has a default value of `0.5`. This gives training subsets and testing subsets of the same size.

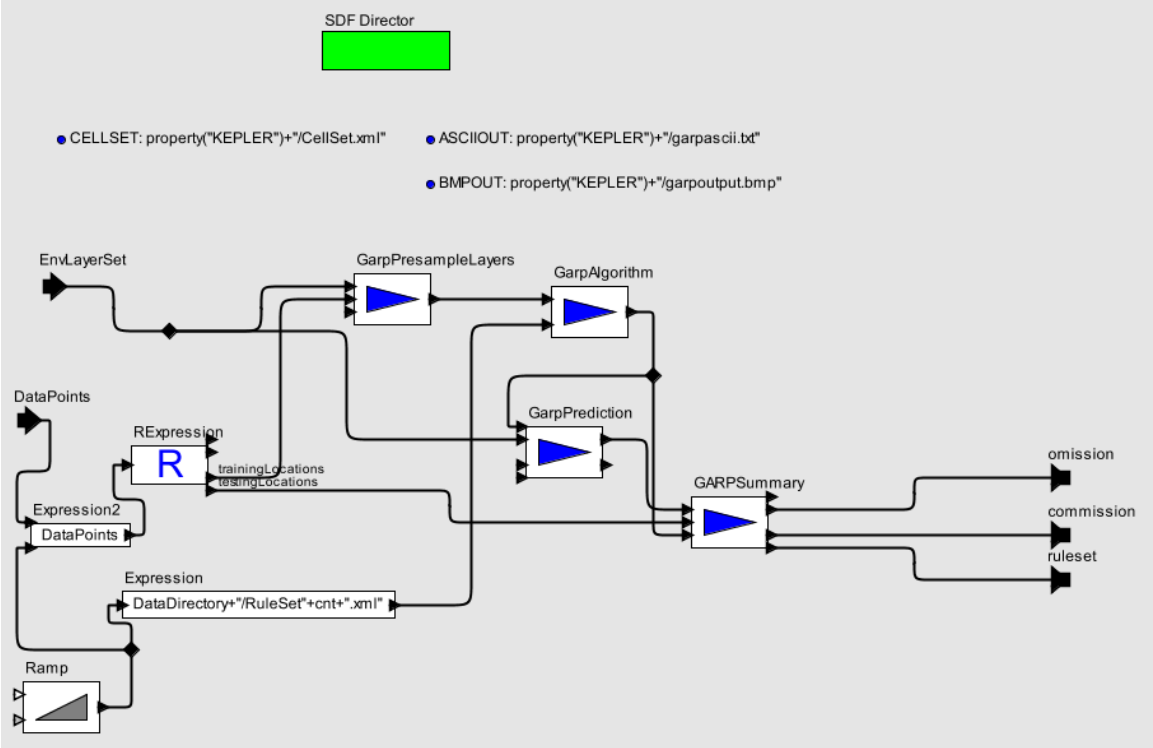


Figure 5 - Inside Block V – Calculate Omission/Commission

The GARPSummary actor actually calculates the omission and commission values and outputs them, along with the associated ruleset. Each set of these three parameters is then added as a 'row' in a table by the next composite in the top-level workflow (Figure 3). Looking inside this composite gives the subworkflow shown in Figure 6.

Figure 6 shows how a sequence of commission, omission, and ruleset values are first converted to arrays and the put into a Kepler/Ptolemy record. This record can be thought of as a three-column table. It can then passed to an REpression actor where it automatically becomes an R Dataframe.

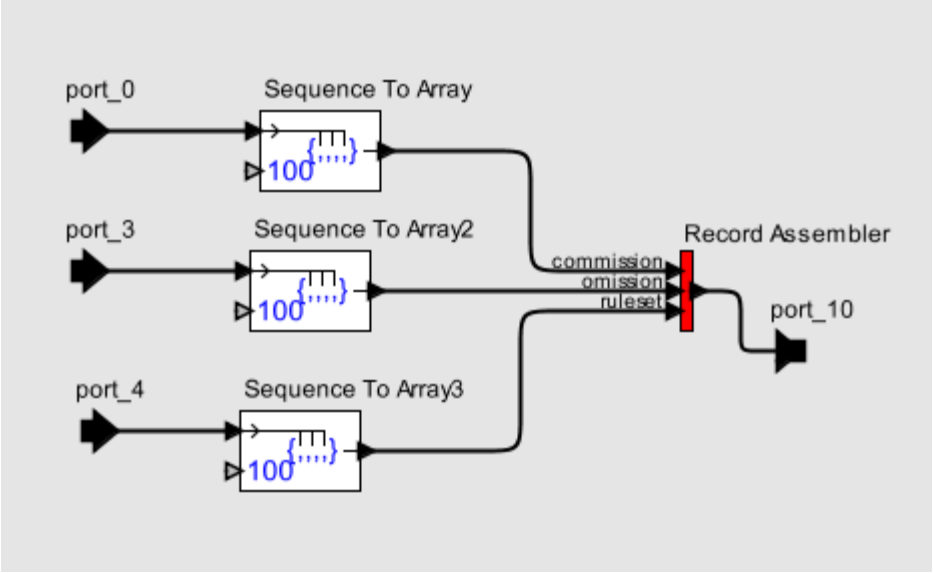


Figure 6 – Inside the Create Omission/Commission Table Composite Actor

Now consider the RExpression actor in Figure 3 (VI – GARP_BestSubset_R). The R script for this actor is:

```
df
df1 <- df[order(df$commission),]
df1
df2 <- df1[1:(0.1*length(df1$commission)),]
df3 <- df2[order(df2$omission),]
df3
l <- length(df3$commission)
l
df4 <- df3[(0.25*l):(0.75*l),]
df4
```

This script sorts the table (dataframe df) by the `df$commission` column using the `order` function. It then creates a new table with the largest 10% of the commission values. (The fixed value of '`0.1`' should probably be defined as a variable so that it can more easily be varied by the user.) It then takes the middle 50% of the remaining values (i.e. the values from `0.25` to `0.75` of the length).

Finally, note the 'numIterations' parameter near the top of Figure 3 with a displayed value of 100. The value of that parameter is the number of times the workflow is to be iterated. The SDF Director is set to repeat that number of times and the parameter is used in several other places in the workflow, including inside composite actors.

Note that supplying a simple constant file name for the environmental layer lists a total of 'numIterations' times is not computational intensive, but we really don't want to calculate the same convex hull repeatedly. Thus, the 'Repeat' actor appears in Figure 3. This actor just repeats the token from the previous actor 'numIterations' times and thus avoids repeating the previous calculation.