

Fort Collins Science Center

User Documentation for the Software for Assisted Habitat Modeling (SAHM) package in VisTrails.

By Colin Talbert, Marian Talbert

Report Series XXXX–XXXX

U.S. Department of the Interior

U.S. Geological Survey

U.S. Department of the Interior

KEN SALAZAR, Secretary

U.S. Geological Survey

Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia: 201x  
Revised and reprinted: 201x

For more information on the USGS—the Federal source for science about the Earth,  
its natural and living resources, natural hazards, and the environment—visit   
<http://www.usgs.gov> or call 1–888–ASK–USGS

For an overview of USGS information products, including maps, imagery, and publications,  
visit <http://www.usgs.gov/pubprod>

To order this and other USGS information products, visit <http://store.usgs.gov>

Suggested citation:  
Author1, F.N., Author2, Firstname, 2001, Title of the publication: Place of publication   
(unless it is a corporate entity), Publisher, number or volume, page numbers; information   
on how to obtain if it’s not from the group above.

Any use of trade, product, or firm names is for descriptive purposes only and does not imply   
endorsement by the U.S. Government.

Although this report is in the public domain, permission must be secured from the individual   
copyright owners to reproduce any copyrighted material contained within this report.

**Contents**

Introduction 1

Installation of SAHM and VisTrails 2

Windows Installation 2

VisTrails Primer 11

Creating and Editing Workflows 12

Executing Workflows 13

Viewing Output in the Spreadsheet 14

Workflow History Tree View 15

Parameter Exploration 17

Using the SAHM Package 18

Basic Operation. 18

The session folder. 19

Typical SAHM workflow 20

Inputs 21

***Field Data*** 22

***Template Layer*** 23

***Covariates*** 23

***Predictor*** 24

***PredictorListFile*** 24

***Individual Predictors selector modules.*** 25

Pre-processing 27

***Field Data Query*** 27

***Field Data Aggregate and Weight*** 29

***Project, Aggregate, Resample, Clip (PARC)*** 30

How PARC works: 32

***Raster Format Converter*** 34

Pre-modeling Data Manipulation 34

***Merged Dataset Builder.*** 35

***ModelSelectionSplit and ModelEvaluationSplit.*** 36

***ModelSelectionCrossValidation*** 38

***CovariateCorrelationAndSelection*** 41

*Modeling* 43

***Output Produced by All models*** 43

***Generalized Linear Model (GLM)*** 52

***Multivariate Adaptive Regression Splines (MARS)*** 52

***RandomForest*** 53

***BoostedRegressionTree (BRT)*** 53

***Maxent*** 54

***Select and Test the Final Model*** 54

Viewing Output 56

***Spreadsheet*** 56

***SAHM Model Output Viewer Cell*** 57

***SAHM Spatial Output Viewer Cell*** 57

Glossary 60

**Figures**

**Figure 1.** The main VisTrails Builder Window of the VisTrails 2.0 program. 11

**Figure 2.** The VisTrails Spreadsheet. 15

**Figure 3.** VisTrails History View. 16

**Figure 4.** VisTrails Parameter Exploration View. 18

**Figure 5.** Stages in a typical SAHM workflow. 21

**Figure 6.** A calibration plot showing good calibration. Note that the logit curve follows the diagonal quite closely. The intercept is not significantly different than zero and the slope is not significantly different than 1. 46

**Figure 7.** These plots show poorly calibrated models. Positive bias occurs when the predicted values are higher than the observed probabilities of occurrence. Negative bias occurs when the predicted probababilities of occurrence are lower than the observed probabilities. Spread error 1 indicates that predicted probabilities of occurrence are higher than than they should be . 47

**Figure 8.** Confusion matricies for a data with no evaluation split and with a test\training split. A good model fit will have relatively high percentages in the red bars indicating that the predicted presence and predicted absence of observations agree well with the observed values of presence and absence and thus the model has good discrimination. Large discrepancies between the training and evaluation data in this plot could indicate model overfitting. 49

**Tables**

**Table 1.** Module execution color interpretation. 13

User Documentation for the Software for Automated Assisted Habitat Modeling (SAHM) package in VisTrails.

By Colin Talbert, Marian Talbert, Drew Ignizio

# Introduction

The Software for Assisted Habitat Modeling (SAHM) has been created to both expedite habitat modeling and help maintain a record of the various input data, pre- and post- processing steps and modeling options incorporated in the construction of a species distribution model. The four main advantages to using the combined VisTrail: SAHM package for species distribution modeling are:

1. formalization and tractable recording of the entire modeling process
2. easier collaboration through a common modeling framework
3. a user-friendly graphical interface to manage file input, model runs, and output
4. extensibility to incorporate future and additional modeling routines and tools.

This user guide provides detailed instructions on how to install the SAHM package within VisTrails and then presents information on the use of the package. A step-by-step tutorial to create cheatgrass habitat suitability maps for Rocky Mountain National Park, USA, is provided in appendix A.

# Installation of SAHM and VisTrails

## Windows Installation

The following detailed steps can be used to install and setup the VisTrails and the SAHM package on Windows. The various components are all freely available but require separate downloads and installations. These include VisTrails 2.0, R (developed under version 2.14.0), Maxent (developed under version 3.3.3k), GDAL (developed under version 1.8.0), and the GDAL Python bindings for Python 2.7. While not terribly difficult each component must be installed correctly and configured for the individual computer. The most difficult step in this process is getting the GDAL Python bindings installed into the Python installation that comes with VisTrails. Previous installations of any of the above components or PyQt might cause compatibility problems with VisTrails SAHM. These issues can be difficult to troubleshoot and fix. Particular care should be taken to ensure that the versions of each component are compatible. For example if you install the 64-bit version of VisTrails you should also install the 64-bit version of the GDAL Python bindings.

**NOTE: You must have Administrative Rights on the computer you are trying to install on.**

1. **Download and install VisTrails.**

Ready to install executables are available from <http://www.vistrails.org/index.php/Downloads>. Choose the most current release which is currently 2.0-alpha. The bundle is available for both 32-bit and 64-bit machines (to view the specs of a computer to determine which version you should get, go to Control Panel>System and Security>System). Install this application on your system using the provided installers. You will need administrative rights to complete this installation. Unlike some installations which will prompt a user to login as an administrator during the installation, a user should be logged in with admin privileges before attempting the installation, or else it will fail. Complete instructions and documentation are available from their site. Note that for the purpose of these instructions I installed my VisTrails to C:\VisTrails so subsequent instructions will be relative to this location.

1. **Download and install GDAL.**

32-bit and 64 bit versions are available from http://www.gisinternals.com/sdk/

A user should choose the 32-bit or 64-bit version to match their system architecture. We tested these instructions with the 'release-1600' version.

Specifically we used “Generic Installer for the GDAL Core Components”:

<http://www.gisinternals.com/sdk/Download.aspx?file=release-1600-x64-gdal-1-8-mapserver-6-0\gdal-18-1600-x64-core.msi> for 64-bit architecture and

<http://www.gisinternals.com/sdk/Download.aspx?file=release-1600-gdal-1-8-mapserver-6-0\gdal-18-1600-core.msi> for 32-bit architecture.

When it finishes downloading double click the msi and choose the 'complete installation' option.

The final step for GDAL is to add its location to your Path variable and add a couple of required Environmental (System) Variables to your system. On your machine’s system, navigate to Control Panel > System and Security > Advanced System Settings (on the left) > Environmental Variables



Click on the Environment Variables tab and add a new system variable called 'GDAL\_DATA' that points to the 'gdal-data' folder within the GDAL install location. Note this location will be specific to where you installed GDAL and might be C:\Program Files (x86)\GDAL\gdal-data if you installed the 32-bit version on a 64-bit machine.



Next add a new system variable called 'PROJ\_LIB' that points to the 'projlib' folder in the GDAL install location:



Now click on and edit the 'Path' variable. Append the path to the root GDAL folder to the variable value for the 'Path' variable. Include a semicolon at the end, with no spaces, 

when done modifying the 'Path' System Variable:



1. **Download and install the GDAL bindings for Python.**

These are available from the same location we obtained the GDAL core from, <http://www.gisinternals.com/sdk/> Once again match the 32-bit or 64-bit architecture of your system and be sure to grab the version for Python 2.7.

Specifically we used “Installer for the GDAL python bindings (requires to install the GDAL core)”:

<http://www.gisinternals.com/sdk/Download.aspx?file=release-1600-x64-gdal-1-8-mapserver-6-0\GDAL-1.8.1.win-amd64-py2.7.msi> for 64-bit architecture and

<http://www.gisinternals.com/sdk/Download.aspx?file=release-1600-gdal-1-8-mapserver-6-0\GDAL-1.8.1.win32-py2.7.msi> for 32-bit architecture.

Once it finishes downloading double click to install. For the installation of these bindings you must choose ‘Python from another location’ and then point to the python installation used by VisTrails. In our case I used C:\VisTrails\vistrails\Python27\_64 Note that if you've installed the 32-bit version of VisTrails this will be C:\VisTrails\vistrails\Python27 When providing the location after specifying "Python from another location," a user should be providing the location in which the python.exe program resides. We want to provide the location where the bindings should be installed, rather than pointing to the full path of the actual python.exe file.



The final thing to do is copy the DLL files from the GDAL install location to the VisTrails folder in the VisTrails installation.

-In GDAL install location (C:\Program Files\GDAL), sort all files by type. Select all the DLL files (these might be labeled 'Application extension') and copy them. Paste the files here: C:\VisTrails\vistrails



1. **Download and install the SAHM package.**

This is available from <http://www.fort.usgs.gov/xxxxxxxxxxxxxxxxxx>. Unzip this into the VisTrails packages directory which is at C:\VisTrails\vistrails\packages.

1. **Download and install the external software R**

R is available from <http://www.r-project.org/>. If you have a previously installed version of R you might be OK with it but we have only tested our code with version 2.14.0 for Windows. All required libraries will be programmatically downloaded and installed assuming you have internet connectivity at the time you first run an R module. The following libraries are loaded as needed ade4, foreign, gbm, lattice, mda, PresenceAbsence, randomForest, raster, rgdal, ROCR, sp, spatstat, survival, tcltk2, and tools.

1. **Download and install the external software Maxent**

The Maxent package is available for download from: <http://www.cs.princeton.edu/~schapire/maxent/>

1. **Configure the SAHM package to point to where the above packages are installed.**

Once all the dependencies are installed we just need to let the SAHM package know where the various applications are installed. To do this open the VisTrails application and select Edit -> Preferences -> Module Packages (tab). Select and enable sahm in the list of Disabled packages. 

Click Configure.



By double clicking in the 'Value' area you can change the path to point to the location where various components were installed. In our case we used:

**gdal\_path** = C:\Program Files

note: this is the folder that contains the GDAL folder

**r\_path**  = C:\Program Files\R\R-2.14.0\bin

note: this is the bin folder within your R installation

**maxent\_path** = C:\Maxent

note: this is the folder that contains the maxent.jar file

**Additional configuration:**

While you are setting the above you could also change the following configuration variables:

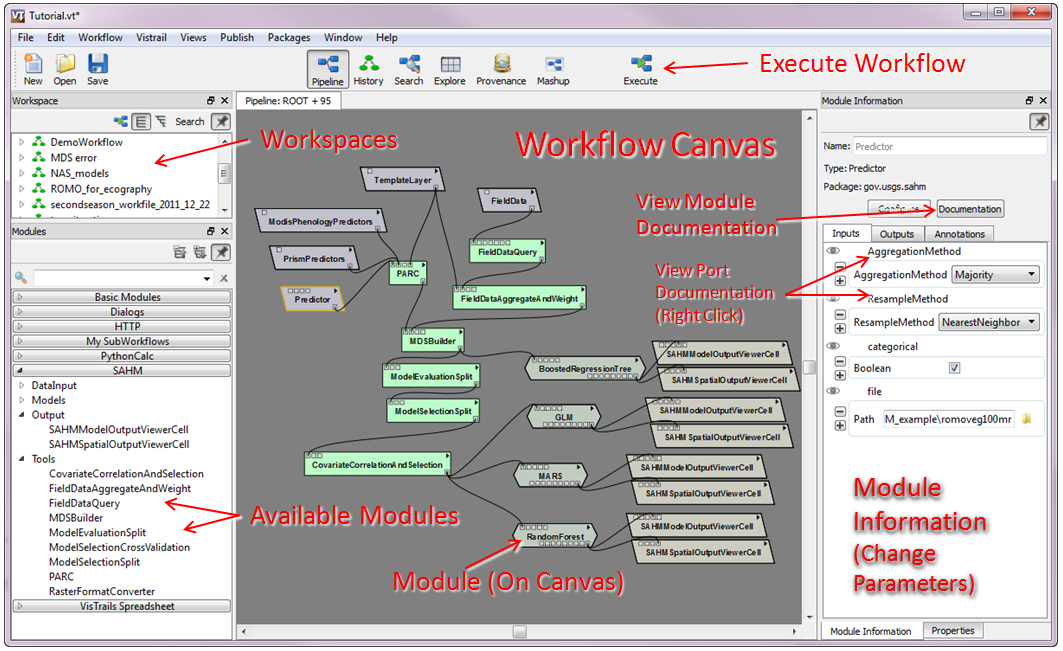
**output\_directory** = The default location you want SAHM to store your session folders.

**verbose** = Whether to print out all output messages to console screen. This

should stay the default value of True.

# VisTrails Primer

VisTrails is covered in detail in their own documentation and much of the content won’t be repeated here. The complete documentation is available at <http://www.vistrails.org/usersguide/VisTrails_Documentation/VisTrails_Documentation.html> This section is meant as a quick introduction to the highlights of VisTrails.



1. The main VisTrails Builder Window of the VisTrails 2.0 program.

## Creating and Editing Workflows

When you first open VisTrails you will see the window above with a blank Workflow Canvas. In the Workflow Canvas window you will view, edit and create workflows. A workflow is composed of individual Modules that each perform a specific function. The workflow as well as a complete history (provenance) of all changes and parameters used can be saved to a VisTrails (.vt) file. The Available Module are show to the left of the Workflow Canvas. These are arranged by the packages they belong to. Some packages are distributed with the VisTrails application others, such as SAHM, are created and distributed by others to enable discipline specific functionality within VisTrails.

Modules are added to a workflow by dragging the item from the Available Modules list onto the Workflow Canvas. You will notice on each module a series of small squares on the top and bottom of its shape. These represent input and output 'ports' or parameters for the module. The inputs and parameters are along the top and outputs are on the bottom. By hovering over a port you will get a pop-up text box that gives the port name and data type. Complete Module documentation is available by either clicking the small triangle in the upper right of the Module and selecting 'View Documentation' or by clicking the 'Documentation' button in the Module Information panel. Additional detailed documentation about each port, including the modules and ports it can be connected to is available by left clicking the port name in the Module Information panel and selecting 'View Documentation'.

Modules are connected into a workflow by clicking and dragging from the output port of one module into the appropriate input port of the next module. There are numerous valid ways of connecting modules depending on the processing required. When starting out it can be difficult to know which modules connect where. It will be helpful at first to open and modify an existing workflow instead of starting from scratch. The module and port documentation as well as the package documentation (this document) can be informative as well. Modules or connections that are no longer needed can be removed by selecting them and clicking 'Delete'.

As workflows become larger you might find it helpful to resize and navigate around the workflow canvas. The scroll wheel and sideways scroll wheel action move around (pan) the workflow canvas. Panning can also be accomplished by holding shift and the left mouse button and moving the mouse. By holding down the right mouse button and moving the mouse up or down you can zoom in and out. A clicking control+r will re-center the canvas.

## Executing Workflows

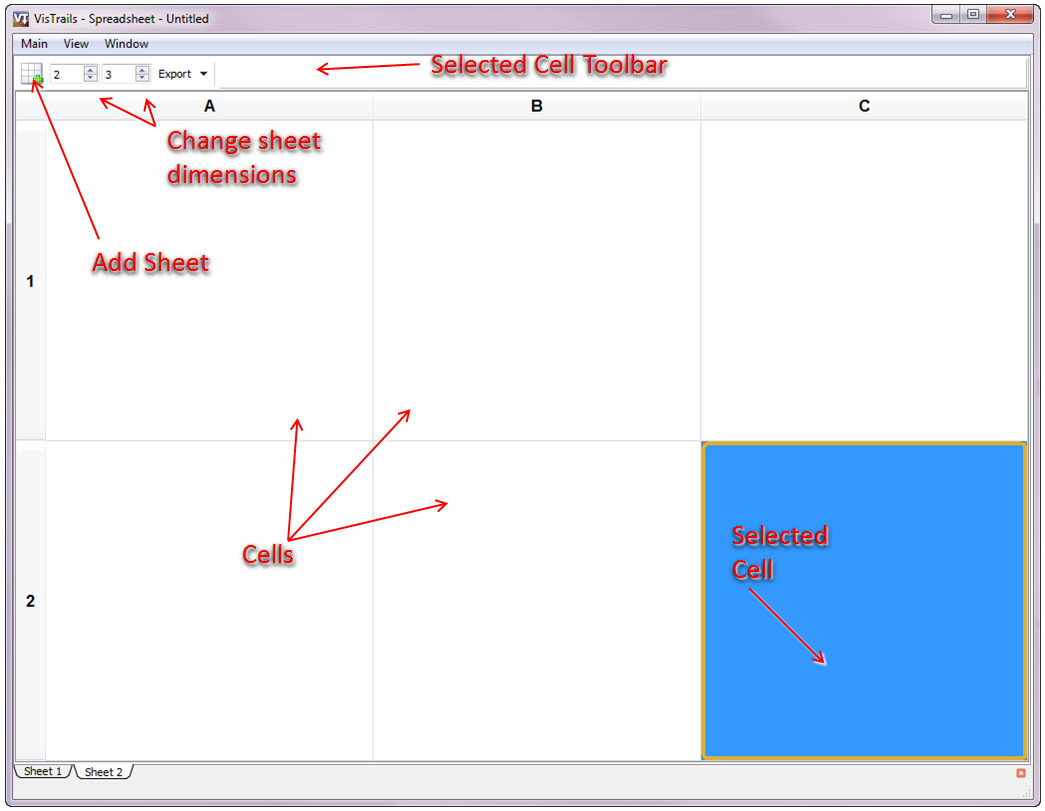
Once a workflow has been set up and configured it can be run by clicking on the 'Execute Workflow' button on the top of the main window. VisTrails will sequentially run through the execution of each module starting with the ones that have no inputs feeding into them and proceeding down the workflow. VisTrails caches the results of each models execution so that subsequent runs will not rerun the modules that completed successfully unless one of their inputs or parameters were changed. All modules downstream of a rerun module will be rerun. To override this property you can click Workflow -> Erase Cache Contents and subsequent runs will run all modules. As execution progresses the color of the modules in a workflow will change according to the table below.

1. Module execution color interpretation.

|  |  |  |  |
| --- | --- | --- | --- |
| Module Color | Meaning | Example | |
| Green | Module was executed |  |
| Red | Module execution failed |  | |
| Yellow | Module is currently being executed |  | |
| Orange | Module was cached |  | |
| Lilac | Module was not executed |  | |

## Viewing Output in the Spreadsheet

VisTrails has a spreadsheet window for viewing the various output produced by a workflow or series of workflows. Certain modules allow outputs or files to added to the spreadsheet. By default these modules will be added to the next available cell but the cell location can also be explicitly specified. Each specific type of spreadsheet cell module can display a particular type of file and will allow for particular types of output manipulation.

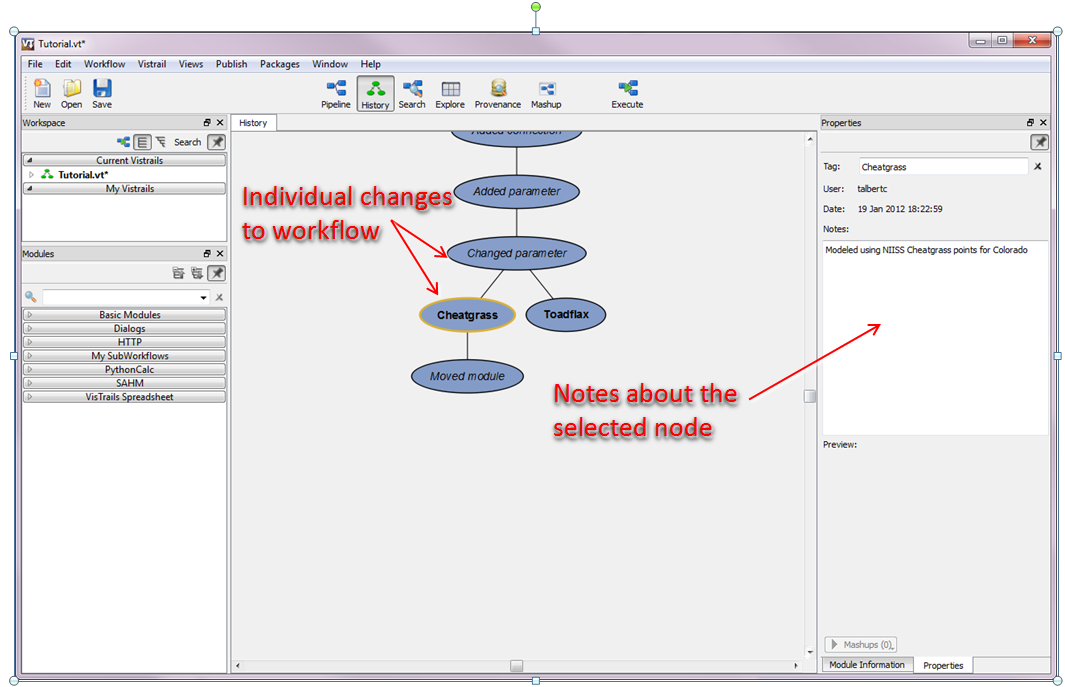


1. The VisTrails Spreadsheet.

The number of rows and columns displayed on a spreadsheet should be changed to match the output from a workflow or series of workflows. Additional sheets can be added for subsequent runs and sheets can be renamed to organize results.

## Workflow History Tree View

One of the strengths of VisTrails is that it tracks a complete history of all steps in the evolution of a workflow. If you would like to review or return to a previous step of your history it can be done using the History Tree View which is accessed with the 'History' button at the top of the main window. Since the History Tree View allows a full tree structure, relabeling of nodes on the history, and additional annotation this is a convenient and powerful means of documenting the process resulting in a set of final results. Additionally it can allow a user to quickly return to a previous workflow state if a particular line of processing proves unfruitful.



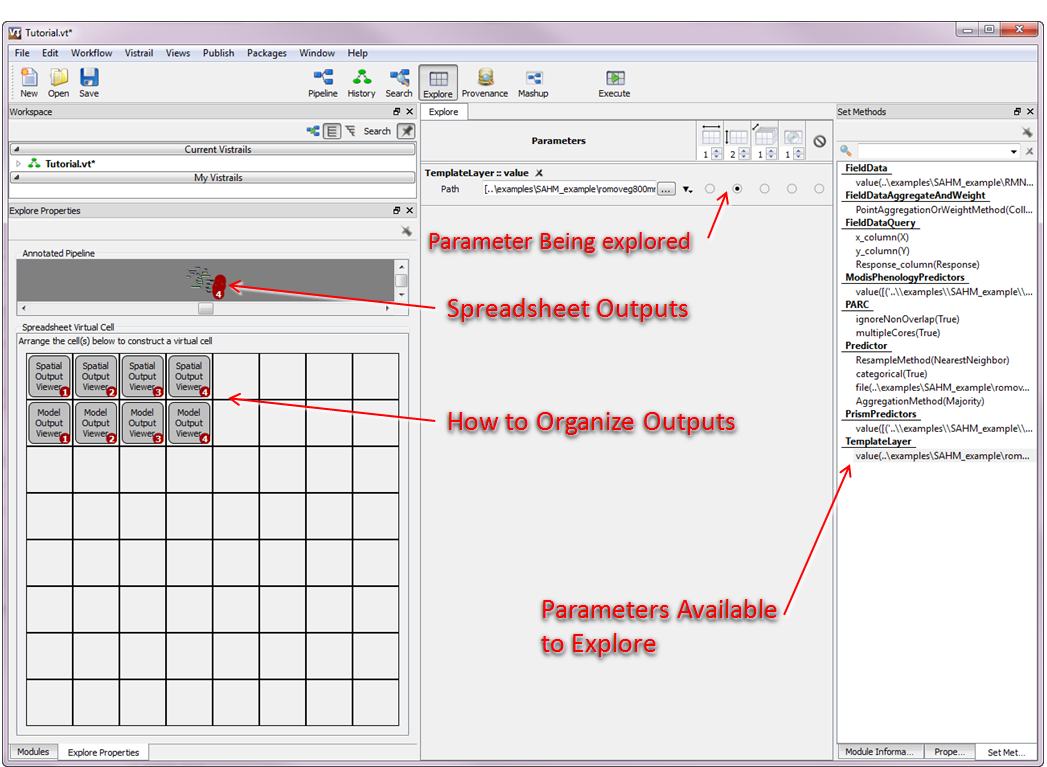
1. VisTrails History View.

The same navigation and panning functionality described in the Creating and Editing workflows section applies to the History View Canvas. To return to a previous workflow state select the node in the appropriate node on the tree and when you change back to the Pipeline View (Workflow Canvas) the modules, parameters, and connections will be as they were at that point in time. To change the label on a node double click on the text in that node and write a more meaningful tag. Additional notes about a node can be entered into the notes field in the properties tab to the right of the main canvas.

In addition to the history tab the Provenance tab provides functionality to view the results of each individual run executed on a workflow. This provides an execution log that complements the information stored in the history view.

## Parameter Exploration

The Parameter Exploration tab includes tools for easily seeing how changes to a parameter or series of parameters affects workflow output. This might be a series of input files or a range of values for a given parameter. To explore a parameter drag the name from the list in the 'Set Methods' tab to the right onto the 'Explore Canvas'. Next Change the values to include how you want to explore the parameter. Clicking the button with three ellipses opens a wizard to assist with this. Next decide how you want each iteration's output to be organized on the output spreadsheet. You can specify rows, columns, additional sheets, or as a movie over time. This is set with the radio buttons next to the parameter being explored on the 'Explore Canvas'. Finally you need to arrange the outputs from each individual iteration using the Spreadsheet Virtual Cell tab to the right of the 'Explore Canvas'.



1. VisTrails Parameter Exploration View.

# Using the SAHM Package

## Basic Operation.

The SAHM Package contains a series of modules useful for all stages of the species habitat modeling process. Each of these modules has comprehensive documentation that is available directly from within VisTrails. To access the general documentation for a module select it on the workflow canvas and click the documentation button on the module information panel. To access detailed documentation about each of the ports for a selected module right click on the port name and click 'View Documentation'. This brings up a form that displays a definition for the port, whether the port is mandatory, the default value used, The possible options available for the port, and the other modules/ports that port is commonly connected to in SAHM. The extensive information provided in the module and port level documentation is not repeated in this document.

1. VisTrails SAHM documentation access.

When running SAHM in VisTrails many useful messages, warnings and errors are printed to the VisTrails console window. It is a good idea to periodically check over this output as workflows are run. This same information is also written to a session log file titled sessionLog.txt which is saved in the session folder (see below). This log file also contains additional information not displayed on the console such as starting times of individual processes, commands sent to R, etc. When SAHM encounters problems in the execution of a model execution will stop and the module that encountered the problem will turn red. An error message is available in a pop-up tool tip if you hover over this module.

## The session folder.

All of the outputs produced during a single session of SAHM in VisTrails are saved to a single directory which we refer to as a ‘session folder’. When you first start up SAHM in VisTrails a new uniquely named folder is created in the working directory specified in the SAHM configuration. If you would like to save your outputs to a different folder you can select Packages -> SAHM -> Change Session Folder. This can also be useful if you would like to continue processing started in a previous session.



Within the session folder individual outputs are generally uniquely, sequentially named so that subsequent outputs do not overwrite previous outputs. For example the the first output from the Field Data Aggregate and Weight module will be called FDAW\_1.csv. The next time this module is run the output will be FDAW\_2.csv. In the case of modules that produce several related outputs the group will be placed in a folder named with the same unique sequential convention. For example all of the outputs produced by a single model run will be placed in a single folder. The exceptions to this pattern are the Covariate Correlation and Selection module which will have the name from its selectionName parameter and the PARC module which will be send its output to a folder named according to the template layer used.

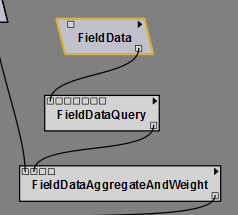
## Typical SAHM workflow

The typical SAHM workflow can be divided into five general stages: specifying inputs, pre-processing, preliminary model analysis, modeling, and viewing output. The modules within SAHM are given subtle colors and/or shapes to differentiate the stages.

1. Stages in a typical SAHM workflow.

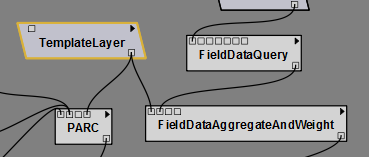
## Inputs

Inputs for SAHM must all currently exist as files on the file system. Some such as FieldData and the Template layer will be specified by navigating directly to a single file. Others such as the PredictorListFile allow a user to specify multiple inputs by creating a CSV file which indicates multiple individual files. Other commonly used inputs can be specified the Layers.csv file in the SAHM package install directory. The inputs listed here become easily available to users by name from picklists in the [category name]predictors modules.

****

### Field Data

The fundamental input to a modeling session is the field data file. SAHM currently allow for presence only, presence/absence, and count data as field data. At a minimum this file contains rows with the X and Y coordinates and the recorded detection of a species at that point. See the module level documentation in VisTrails for the specifics of required file format. It is also possible to supply a file that contains additional inputs and use the Field Data Query Module to subset/reformat the output (see Field Data Query Module documentation for more information).

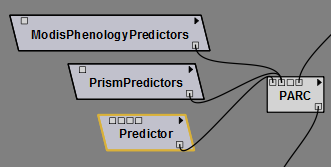


### Template Layer

The second fundamental input in an analysis is the template layer. It is used to define the extent and resolution that will be used in all subsequent analyses. The TemplateLayer is a raster data layer with a defined coordinate system, a known cell size, and an extent that defines the study area. The data type and values in this raster are not important. All additional raster layers used in the analysis will be resampled and reprojected as needed to match the template, snapped to the template, and clipped to have an extent that matches the template. Users should ensure that additional covariates considered in the analysis have complete coverage of the template layer used.

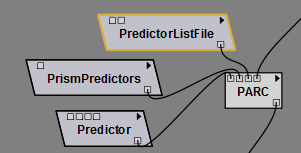
### Covariates

The SAHM package provides several ways of specifying which covariates or predictors will be included in subsequent analysis.



### Predictor

The Predictor module allows a user to select a single raster layer for consideration in the modeled analysis. Since this predictor will be going into the PARC module the user must specify the aggregation method, resampling method, whether the layer if categorical.



### PredictorListFile

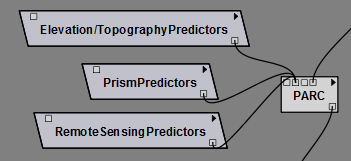
The PredictorListFile module allows a user to load a .csv file containing a list of rasters for consideration in the modeled analysis. The .csv file should contain a header row and four columns containing the following information, in order, for each input raster.

Column 1: The full file path to the input raster layer including the drive .

Column 2: A binary value indicating whether the input layer is categorical or not. A value of "0" indicates that an input raster is non-categorical data (continuous), while a value of "1" indicates that an input raster is categorical data.

Column 3: The resampling method employed to interpolate new cell values when transforming the raster layer to the coordinate space or cell size of the template layer, if necessary. The resampling type should be specified using one of the following values: "nearestneighbor," "bilinear," "cubic," or "lanczos."

Column 4: The aggregation method to be used in the event that the raster layer must be up-scaled to match the template layer (e.g., generalizing a 10 m input layer to a 100 m output layer). Care should be taken to ensure that the aggregation method that best preserves the integrity of the data is used. The aggregation should be specified using one of the following values: "Min," "Mean," "Max," "Majority," or "None."



### Individual Predictors selector modules.

The SAHM package allows users to set up dynamic Predictor selection widget which allow users to easily select from commonly used predictors. Examples might include Daymet, PRISM, MODIS, etc. For this to work the layers must exist on the local file system and the user must configure a csv file which contains the locations and parameters for each. This csv is located in the SAHM package and is called layers.csv. The first column (FileName) contains the full path to each individual file. The second column (LayerName) is a short descriptive name for this layer. The third column (Group) specifies which widget this layer will be found in. All of the distinct entries in the Group column will generate individual widgets with similar data. In our case we found it convenient to separate climate layers from phenology layers, from topology layers. The fourth column (Source) provides a means of grouping layers within an individual layer. Each unique entry within a widgets layers will generate a collapsible group of similar layers. The fifth and final column (Categorical) contains a ‘Y’, ‘N’ Boolean value indicating whether this layer is categorical. If it is categorical Resampling will default to ‘nearest neighbor’ and Aggregation will default to ‘majority’. If it is not categorical Resampling will default to ‘bilinear’ and Aggregation will default to ‘mean’. See the original layers.csv for an example of how to set up this file.

The resulting widgets derived from this file are named the unique Group column name followed by ‘Predictors’. For example you might end up with a ClimatePredictors widget and a PrismPredictors widget. To select individual layers from each widget click the small triangle in the widget’s upper left and select ‘Edit Configuration’ in the dropdown. A box will pop up which contains the layers managed by this widget. Checking the boxes at the left will include that layer in the model. Optionally you can also change the categorical, resampling and aggregation parameters used in subsequent processing.

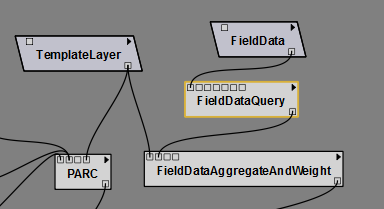


Within the widget convenience functions to select all layers or switch the selection are provided via buttons at the bottom. The Query box can be used to easily add/remove multiple layers. To use enter text in the Query box that is unique to the layers that you want to include or remove. The full paths to the layers will be individually searched for the provided text. For example I might want to include all layers that have MonthlyPrecip in the path. Note that the query string is case sensitive.

When the final selection is made click OK to continue.

## Pre-processing

Before SAHM can begin the process of modeling it must first sync the inputs which can come in a variety of formats, projections and coordinate systems. This processing creates a single 'stack' of predictor rasters each with the same extent, cell size, coordinate system, and projection. It also cleans up the original FieldData by removing extraneous rows and columns from the original file and collapsing or weighting the field data points to match the resolution of the template layer used. Some of these steps can sometimes be omitted depending on the formats and conditions of your various inputs.

****

#### Field Data Query

Often raw field data come to us in a format that contains more information than we need to include in any single model. This can take the form of additional columns that contain extraneous information, additional columns that contain occurrence data for additional species, or rows that from a time period, collection method, or species that we are not interested in modeling. The Field Data Query module contains functionality to reformat this output into the format used by the SAHM package.

At a minimum you must specify the columns that contain the X, Y, and response. Columns can be specified with either a positional argument (1, 2, 3, etc) if you want to select the first, second, third etc column. Note these numbers start from 1. Alternatively you can select a column based on name by entering the text of the column name found in the header. Finally you must select the response type of either Presence(Absence) or Count. If none is selected it will default to Presence(Absence).

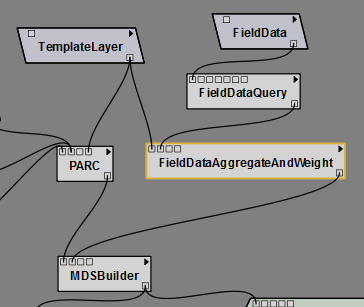
If you would also like to select by rows you can use the Query and Query\_column methods. The Query\_column is the column that contains the data you would like to query on. It will be selected similarly to how columns are selected for X, Y, and response. For the Query column you can either enter a single value or enter an equality statement with x used as a placeholder to represent the values in the query column.

*For example:*

*x < 2005 (would return values less than 2005)*

*x == 2000 or x == 2009 (would return 2000 or 2009)*

*The syntax is python in case you want to create an involved query.*



### Field Data Aggregate and Weight

In many instances data collected in the field can be at a different spatial resolution than we are modeling at. For example we might have observations collected every five meters along a 200 m. transect when we are modeling with covariates with 1000 m. cells. When running species distribution models (SDMs) such as those contained in SAHM, spatial issues need to be addressed in order to avoid introduction of pseudo-replication. For instance, considering multiple field data observations which are all spatially located in the same modeled pixel will generate replicate values or redundant information. When running a model, this redundancy causes pseudo-replication and can negatively influence model development. The FieldDataAggregateAndWeight tool helps aggregate field data locations so only one field data observation is represented per pixel or multiple points are down-weighted proportionately.

Currently only GLM, MARS, and Boosted Regression Trees accept weights. Any Weights column will be ignored by Random Forest.

### 

### Project, Aggregate, Resample, Clip (PARC)

The Projection, Aggregation, Resampling, and Clipping (PARC) module is a powerful utility that automates the preparation steps required for using raster layers in most geospatial modeling packages. In order to successfully consider multiple environmental predictors in raster format, each layer must have coincident cells (pixels) of the same size, have the same coordinate system (and projection, if applicable), and the same geographic extent. The PARC module ensures that all of these conditions are met for the input layers by transforming and or reprojecting each raster to match the coordinate system of the template layer. This process usually involves aggregation (necessary when an input raster layer must be up-scaled to match the template layer— e.g., generalizing a 10 m input layer to a 100 m output layer), and or resampling (necessary for interpolating new cell values when transforming the raster layer to the coordinate space or cell size of the template layer). Lastly, each raster predictor layer is clipped to match the extent of the template layer.

The settings used during these processing steps follow a particular set of decision rules designed to preserve the integrity of data as much as possible. However, it is important for a user to understand how these processing steps may modify the data inputs.

The PARC module accepts four kinds of inputs:

1. Predictor List: A user should not have to populate this field. This field is populated at execution time with selections made from the pre-loaded .csv file of raster predictor layer inputs (specified during the SAHM install) and connected to the PARC module in the visual display.
2. ListRasterWithPARCInfoCSV: The .csv file list corresponds to the Predictor List File element and allows a user to load a .csv file containing a list of rasters for consideration in the modeled analysis. For additional information, please see the documentation for the Predictor List File element.
3. Predictor: The predictor input allows a user to select a single raster predictor layer to be considered in the analysis. It is recommended that a user add this input as a separate element in the visual display (and then link it to the PARC module) so that the aggregation and resampling settings can be established. The PARC module can accept multiple predictor elements. For additional information, please see the documentation for the Predictor element.
4. Template Layer: The template layer is a raster data layer with a defined coordinate system, a known cell size, and an extent that defines the study area. This raster layer serves as the template for all the other inputs in the analysis. All additional raster layers used in the analysis will be resampled and reprojected as needed to match the template, snapped to the template, and clipped to have an extent that matches the template. Users should ensure that any additional layers considered in the analysis have coverage within the extent of the template layer. The template layer is a required input for the PARC module.

Additional Methods:

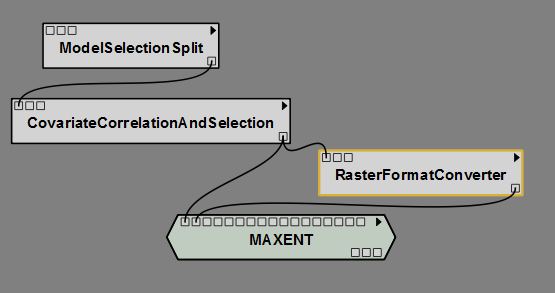
1. multipleCores: If checked individual layers will be processed in parallel on each of the available cores on a machine. This can lock up a machine until processing is finished but processing times are much, much faster.
2. ignoreNonOverlap: If checked the requirement that every covariate completely covers the template is ignored. The output extent will be the intersection of the covariates extents. The output will be snapped to the template raster. Use this with caution.

#### How PARC works:

PARC uses a combination of GDAL (Which used Proj.4), numpy, and python code to accomplish its task. The processing steps that PARC uses are fairly complex and the geospatial implications of the algorithm it used should be considered fully. The default transformation chosen by GDAL will be used in for reprojection but in some cases this is not optimal and can introduce significant errors. In particular be careful with switching datums, using unusual projections or datums, using projections or datums far outside of their intended area of accuracy. When in doubt you might want to perform reprojections in an external GIS package or consult with a GIS professional. For information on the transformations used see <http://www.remotesensing.org/geotiff/proj_list/>

For each raster going into PARC the following steps are performed:

1. The ratio of source cell size to template cell size is determined. If the source raster cell dimension is greater than half of the template raster cell dimension no aggregation will be performed. Otherwise aggregation will be performed if specified. The units used in the comparison are the units in the templates projection.
2. The source raster is projected into the template projection and extent using the resampling algorithm specified. If no aggregation is to be used the output cell size will be identical to the template cell size. If aggregation is to be performed the cell size is calculated to be the value closest to the original source cell size that results in template cells containing even numbers of source cells. For example if aggregating from 30m to 100m the temporary raster dimension would be 25m. These new pixels will be aligned (snapped) with the origin of the template such that a given number of whole pixels falls completely within each template pixel. The source pixel type and nodata value will be maintained.
3. If aggregation is performed on the resulting layer, all of the reprojected source pixels that fall within a given template pixel are considered using the specified algorithm. Currently these are: minimum, maximum, mean or majority. The resulting value is written to the corresponding pixel location in the output raster.



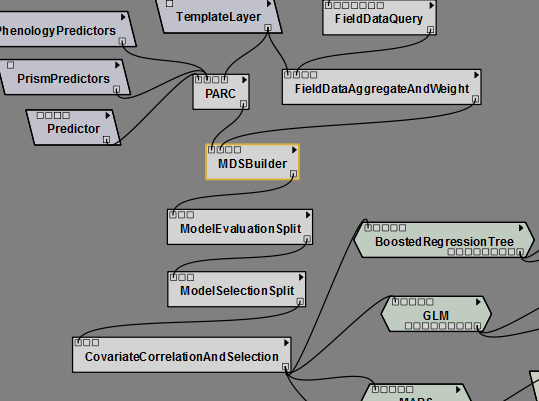
### Raster Format Converter

The RasterFormatConverter module allows a user to easily convert between raster file types for a group of rasters. This group can be specified as either all the rasters in a single directory or the rasters specified in a single MDS file (see below). All outputs will be sent to a folder named "ConvertedRasters" (followed by an underscore and a number corresponding to the run sequence of the module) within the user’s current VisTrail’s session folder. Typically this module will be used within a workflow to convert the geotiff format used by the rest of the modules to the ascii format needed by Maxent. But the following file formats are accepted for both inputs and outputs: Arc/Info ASCII Grid, ESRI BIL, ERDAS Imagine, and JPEG and others. See the compiled by default options at <http://www.gdal.org/formats_list.html> for a complete list of the accepted file types.

## Pre-modeling Data Manipulation

Much of the remaining workflow uses a file format we devised called Merged Dataset or MDS. This format combines the X, Y, response columns of the field data file with the pixel values extracted from the individual covariate rasters at each point. In this regard it is similar to the Samples With Data (SWD) format used by Maxent. The MDS format uses three header lines to record and transmit additional pieces of information as well. The first header line contains the short names for each covariate. Additionally for categorical covariates the work \_categorical is appended to the end of the name. This will indicate to subsequent models how to treat this variable. The second line in an MDS file contains zeros and ones which indicate whether to include or remove this covariate from subsequent models. The third line contains a file path to the location of the raster used to extract values.

In addition to the above items the MDS file stores a few bits of information about its provenance. In cell 1, 2 is the template used for the field data aggregate and weight. In cell 2,2 is the field data file used. Cell 2, 3 contains the name of any selection applied to the MDS. Cell 1,3 contains the raster used for the PARC operation (this should always be identical to cell 1, 2). Cell 2,3 contains the folder that PARC output was saved to. This info is used internally for display in subsequent models and is ignored otherwise.

****

### Merged Dataset Builder.

Prior to subsequent data splitting, checking, and modeling the covariate rasters data at each of our field data points must be extracted. The Merged Data Set (MDS) Builder module performs this extraction and additionally can be used to add background points to the output file. A probability surface can also be used to determine the inclusion probability of having a background point assigned to a given pixel. The output from the MDSBuilder module will be a csv in MDS format. This format will be used as input and output for all of the subsequent before the actual model modules.

### 

### ModelSelectionSplit and ModelEvaluationSplit.

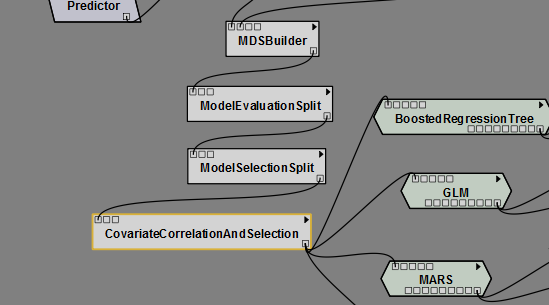
The ModelEvaluationSplit module provides the opportunity to reserve a specified portion of the data for producing and reporting evaluation metrics on an independent test set following model exploration and selection. The ModelEvaluationSplit must be applied before the CovariateCorrelationAndSelection module. The nearly identical ModelSelectionSplit reserves a portion of the data from the model fitting process but reports the evaluation metrics on all models not just the those selected as the final models to be reported in the analysis. This module should be placed directly before the CovariateCorrelationAndSelection. If both a ModelEvaluationSplit and a ModelSelectionSplit are specified then the training portion of the ModelEvaluationSplit will be further partitioned by the ModelSelectionSplit thus the ModelEvaluationSplit should come first in the workflow. Both of these algorithms stratify the splits by the response. That is, the ratio of presence to absence points should be nearly equal in the testing and training split. If a ModelSelectionSplit is included evaluation metrics applied to the reserved data will be reported in the textual output, model evaluation plots including AUC plots as well as the across model plots and the csv. Both of these modules ignore background points and treat all observations with values greater than 0 as presence for the purpose of stratification by response. The decision of whether to use the ModelSelectionSplit and\or ModelEvaluationSplit along with selection of an appropriate training proportion and ratio of presence to absences should be considered in light of the number of observations that are available, the number of predictors under consideration, the number of factor levels for any categorical predictors, and the expected strength between predictors and the response. If the ratio of observations to predictors is small, less than 10 for example, ModelSelectionCrossValidation might be preferable. One should be extremely cautious in both the use of categorical predictors with many levels and the decision to split the data when such predictors are desired. Any factor levels found in a test split or fold (in cross-validation) that was not present in the corresponding training data will result in model failure.

### ModelSelectionCrossValidation

The ModelSelectionCrossValidation module provides another tool for comparing models by splitting the field data observations into cross validation folds. This should not be used with the ModelSelectionSplit but can be used with the ModelEvaluationSplit in which case only the training portion of the ModelEvaluationSplit is partitioned into folds. If specified then the individual models will fit a model using all of the data and report this as the training results. Following the model fitting step sub-models with be fit to each set of n-1 folds and then evaluation metrics calculated on the remaining fold. These will show up as box plots in the AUC plot, means and standard deviations are reported in textual output and box plots in across model comparison plots. Evaluation metrics for each individual fold are reported in the across model comparison csv. The cross validation method incorporated here was originally written for evaluation of MARS models by Leathwick et. al. 2006. The current implementation does not attempt any sort of model averaging but rather is only used for calculation of evaluation metrics. The ModelSelectionCrossValidation module makes better use of data then the ModelSelectionSplit as it uses all of the data to fit the final model but can be substantially more time consuming.

Under most circumstances the cross validation evaluation metrics reported by this module do not indicate how the the model might perform if applied to an independent set of data but rather are to be used only for model selection purposes. The first issue is that when cross validation is applied any feature selection based on the relationship between the response and the predictors must be carried out on each cross validation training set. The CovariateCorrelationAndSelection module includes an exploration of the relationship between the predictors and the response and thus would need to be carried out for each for each cross validation training set. The second issue is that it is invalid to use an evaluation metric for model selection and then report that metric for only the best performing model without acknowledgement to the total number of models that were considered and the range of the evaluation metrics. One should be extremely cautious in both the use of categorical predictors with many levels and the decision to split the data when such predictors are desired. Any factor levels found in a fold (in cross-validation) that was not present in the corresponding training data will result in model failure. This issue is exacerbated in ModelSelectionCrossValidation because splitting the data into n folds will allow n opportunities for failure in contrast to the ModelSelectionSplit which only has 1 opportunity for failure.

A trade-off exists in selecting the number of folds to use for cross validation. When the number of folds is close to the total number of observations the prediction error is nearly unbiased as the cross validation sample size is nearly equal to the total sample size but because the training sets are nearly identical in this case variance of the prediction error can be quite high (Hastie et. al 2009).

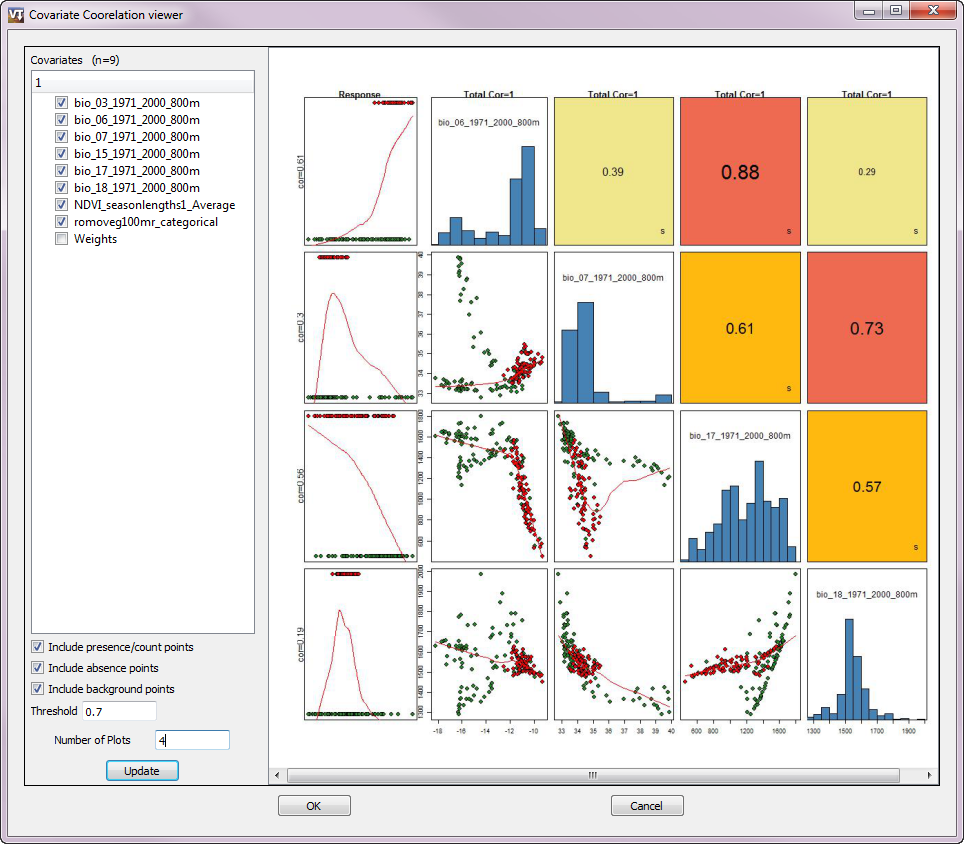
****

### CovariateCorrelationAndSelection

The CovariateCorrelationAndSelection view provides a breakpoint in the modeling workflow for the user to assess how well each variable explains the distribution of the sampled data points and to remove any variables that may exhibit high correlation with others.

The display shows the n variables that have the highest total number of correlations above a threshold with other predictors using the maximum of the Pearson, Spearman and Kendall coefficient. The column heading over each variable displays the number of other variables with which the environmental predictor is correlated using the user supplied threshold which defaults to .7. Radio buttons are available to limit the display and correlation calculations to any combination of presence, absence, or background points. The first column in the plot shows the relationship between the response and each predictor. Row labels indicate the maximum of the Spearman and Pearson correlation coefficient and a locally weighted smooth has been added to help distinguish the nature of the relationship.

The remaining plots make up a square with histograms for each variable displayed on the diagonal. Their respective graphical display and correlation with other variables can be found by locating the row/column intersection between each (above and below the diagonal). The scatter plot along with a locally weight smooth is shown below the diagonal. Presence records are represented by red points, absence by green, and background are yellow. Above the diagonal is the correlation coefficient between the two predictors. If Spearman or Kendall correlation coefficient is larger than the Pearson correlation coefficient then an s or k will show up in the bottom right corner of this box.



A user is provided with the opportunity to select a new set of the environmental predictor variables and “Update” the Covariate Correlation screen to investigate the relationships among the new variables selected. Variables with a high degree of correlation with other variables should generally be unchecked in their respective radio buttons, and will be excluded from subsequent analysis steps in the model workflow. Decisions about which of two correlated variables to remove can be based on many considerations including biological understanding about which is expected to exhibit an influence on the organism under consideration, the strength of the relationship between each predictor and response, and the number of other predictors with which each variable is also correlated.

Multiple iterations can be run at this screen, allowing the user to investigate the relationships among the environmental predictor variables and choose the most appropriate set to be used in the subsequent modeling. When the desired set of variables has been chosen, the “OK” button is selected and processing will resume in the VisTrails workflow.

## **Modeling**

Currently five models are available in the SAHM package. Maxent is used for modeling presence only data. Generalized Linear Model (GLM), Multivariate Adaptive Regression Splines (MARS), Boosted Regression Tree, and Random Forest are appropriate for fitting models to presence/absence data as well as count data all of these except Random Forest work under the assumption that a count response can be modeled as Poisson. Random Forest is less restrictive. GLM, MARS, Boosted Regression Tree, and Random Forest are all implemented in R and have similar options and input requirements and produce similar output. A brief discussion of each model is provided here followed by a discussion of the interpretation of common output. Input options are described in the user manual as well as the module documentation within VisTrails. Maxent was written in Java and has quite different requirements and will be discussed separately.

### Generalized Linear Model (GLM)

This is basically linear regression adapted to binary presence-absence or count data. We used a bidirectional stepwise procedure to select covariates to be used in the model. That is, we began with a null model and calculated the AIC (Akaike Information Criterion) score for each covariate which could be added to the model. AIC is a measure of how well the model fits the data with a penalty based on the number of covariates in the model. In the first step, we add the covariate with the best AIC score. In the next step we calculate AIC scores for all two-covariate models and again add the covariate that most improves the AIC, and so on. At each step, we also look at the change in AIC from dropping each covariate currently in the model. The stepwise procedure ends when no additions or removals result in an improvement in AIC.

### Multivariate Adaptive Regression Splines (MARS)

MARS is a non-parametric technique that builds flexible models by fitting piecewise logistic regressions. In effect, it is similar to GLM except that rather than fitting a straight line response to each predictor, piecewise functions of each predictor are fit, which allows MARS to better accommodate nonlinear response to predictors and also reduces the risk that outlying observations might have high leverage. The model is deliberately over-fit and then pruned back. The original code was developed from that provided in the supporting material of Leathwick and Elith 2006 which contains more details on how model fitting occurs.

### RandomForest

Random forest is a machine learning ensemble classifier. Numerous (in our case, 1,000) decision trees are computed using random subsets of the covariates. Each tree gets one vote, and whichever class gets the best vote “wins”. The relative importance of each covariate is assessed by the change in a fit statistic, on average, for trees that include it. Random forest models automatically model interactions and nonlinear relationships. According to Breiman, cross-validation or separation of data into test\training splits is not required with Random Forest because internally random forest uses a bootstrap sample to fit individual trees observations not used to fit a given tree are called out-of-bag. Predictions can then be calculated for each observation using only trees for which the observation was out-of-bag (OOB). All evaluation metrics for the training data are based on OOB predictions and thus should be similar to the results from applying evaluation metrics to independent test data. The OOB confusion matrix does not agree with the training data confusion matrix because the threshold for the former is based on wining OOB votes while the later has a threshold based on the threshold optimization method selected by the user. Several parameters can be set for random Forest these are discussed in detail and their defaults are listed in the randomForest function documentation in the randomForest reference manual (<http://cran.r-project.org/web/packages/randomForest/index.html>).

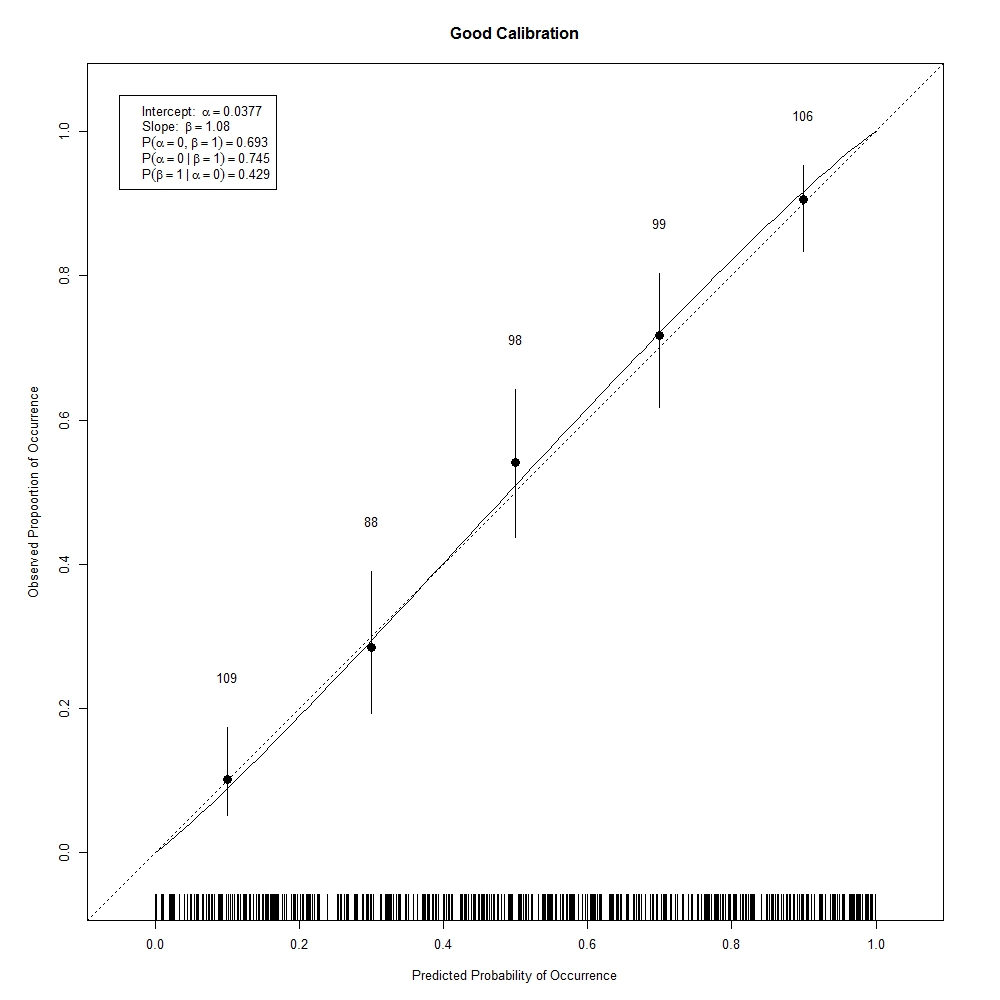
### BoostedRegressionTree (BRT)

BRT is also based on numerous decision trees. BRT starts with a single decision tree, then adds a tree that best explains error in the first tree, and so on. Like random forest, BRT models automatically model interactions and nonlinear relationships, and are robust to missing observations. Our implementation makes approximately 1,000 trees, and incorporates advanced algorithms for tuning the model settings, simplifying the model using a cross-validation technique, and for detecting important interactions between covariates. If more than 500 presence or absence records are found a random subset will be used for learning rate estimation and model simplification but all data will be used in the final model fitting step. The cross-validation step within BRT should not be confused with that provided by the Model Selection Cross Validation step. The former is used to optimize parameter values when defaults are not provided while the later is used to select models based on between model comparisons of evaluation metrics. All discussion of cross-validation related to setting parameters in refers to the algorithm used for parameter optimization and does not affect the cross validation split selected by Model Selection and Cross Validation.

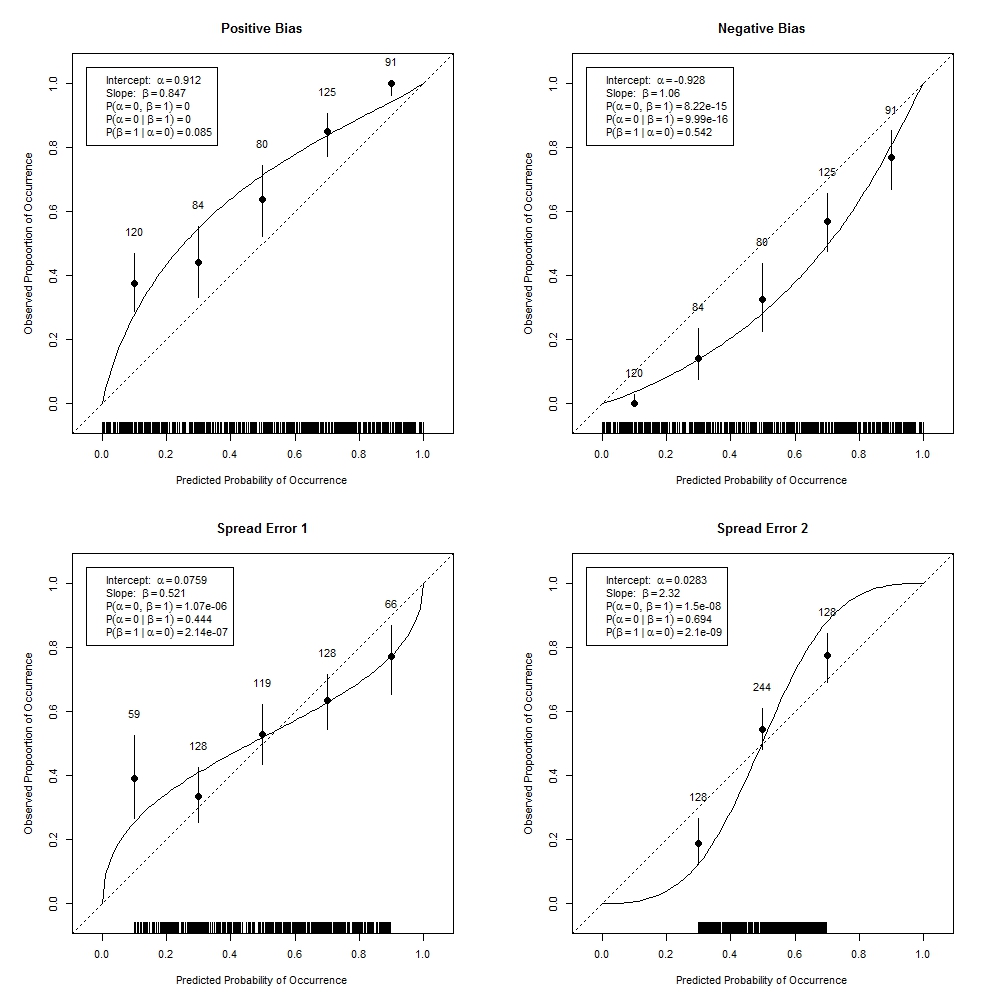
#### Output Produced by All models

Several files should be produced by a successful model run while an unsuccessful model run will produce a log file which will indicate the issue that caused execution to halt and possibly some other files depending on where in the workflow execution was halted. When a model is successful the following files are produce in a folder in the VisTrails workspace:

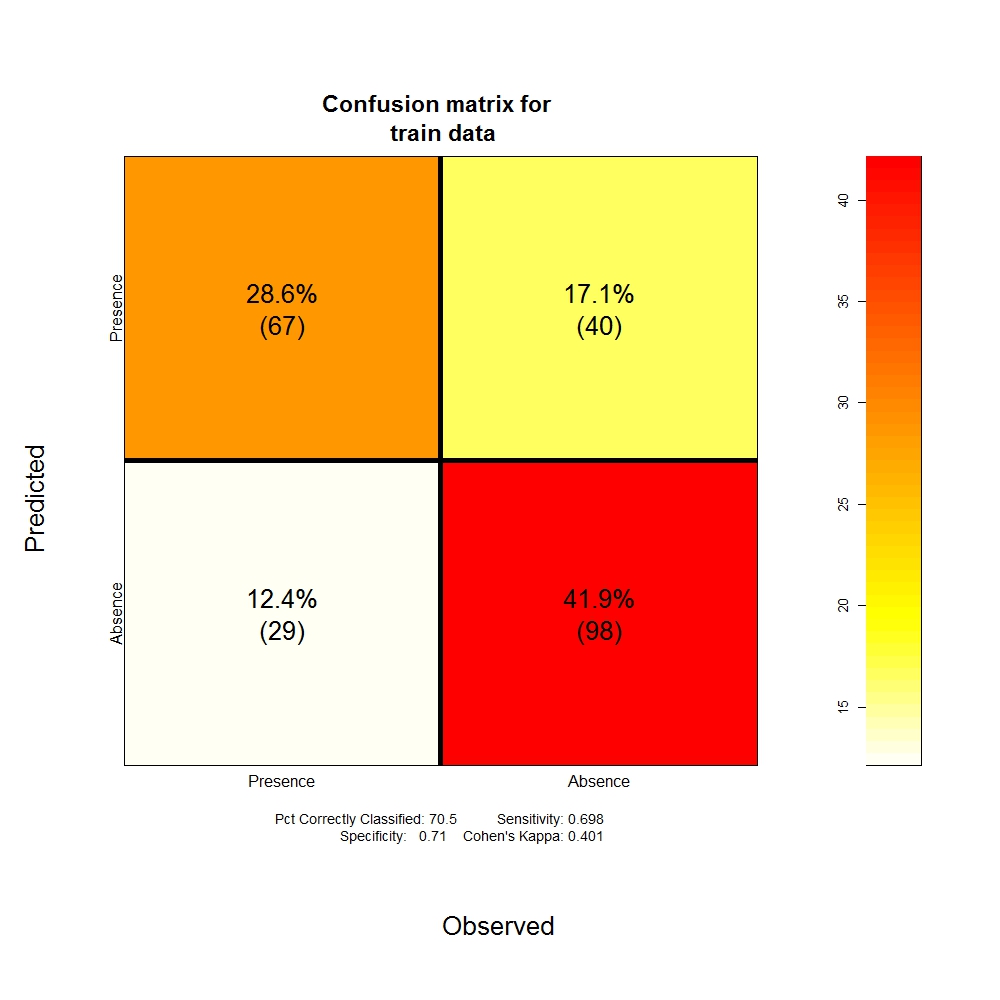
1. “model”\_output.txt : This file contains a summary of the model fit. The information contained here includes the number of presence observations (counts equal to or greater than 1 for count models), the number of absence points, the number of covariates that were considered by the model selection algorithm. Note all of these can differ from the numbers in the original .mds due to incomplete records being deleted, and predictors with only one unique value being removed. The random number seed is recorded if applicable which allows completely reproducible results as well as a summary of the model fit. Evaluation Statistics are reported for the data used to fit the model as well as for the test or cross-validation split if applicable. References for how to interpret most of these are ubiquitous in the literature but it is worth mentioning that interpretation of the calibration statistics is described by Pearce and Ferrier 2000 as well as Miller and Hui 1991.
2. “model”\_modelEvalPlot.jpg : For binary data this will be a Receiver operating characteristic curve. Which shows the relationship between sensitivity and specificity as threshold for discretizing continuous predictions into presence absence is varied. The threshold selected using the specified ThresholdOptimizationMethod is shown. If a model selection test\training split was specified the ROC curve for this will be shown in red and if a cross validation split was specified box plots along the x axis will summarize the cross validation results. If the model fits well both sensitivity and specificity should be well above the diagonal line. If there is a strong disparity between the curves for the training data and either the testing split or cross validation standard deviation curves this can be indicative of model overfitting. These plots and the evaluation metrics based on the confusion matrix describe the models ability to discriminate between presence and absence points. The AUC value, or area under the ROC curve, is the probability that the model will rank a randomly chosen presence observation higher than a randomly chosen absence observation. For count data this display will show several standard plots for assessment of model residuals.
3. “model”\_CalibrationPlot: Calibration plots and statistics describe the goodness-of-fit between the predicted values and the actual observations. These are especially useful for identifying problems with overfitting or underfitting when separate data is used for model fitting and model evaluation. These plots and statistics can be used to determine how reliably the model will predict if a site is occupied or unoccupied (Pearce and Ferrier 2000). The calibration plot shows the predicted probability of occurrence plotted against the actual proportions of occurrence for each of 5 bins along the probability axis. A logistic regression model is fit to the logits of the predicted probabilities of occurrence and is shown on the plot. This line is a logistic curve because we are not using the logit transform of the predicted and observed. The intercept and slope of this line should be 0 and 1 respectively. Test statistics are reported and significant values indicate that the predicted values have a different mean or spread than the observed data. The five plots below show several different patterns that can be identified using calibration plots and statistics. Currently calibration plots are only produced for presence absence models. Poor model calibration means that the model predictions do not match well the probability of occurrence at sites. Poor model calibration is not always associated with poor discrimination and so models with poor calibration but that still have high AUC scores can still be useful for determining which sites are more likely than others to be occupied see Pearce and Ferrier 2000 for more discussion on what can be appropriately quantified with models exhibiting either poor calibration or discrimination. The calibration plot and related statistics are only reported for presence absence models.

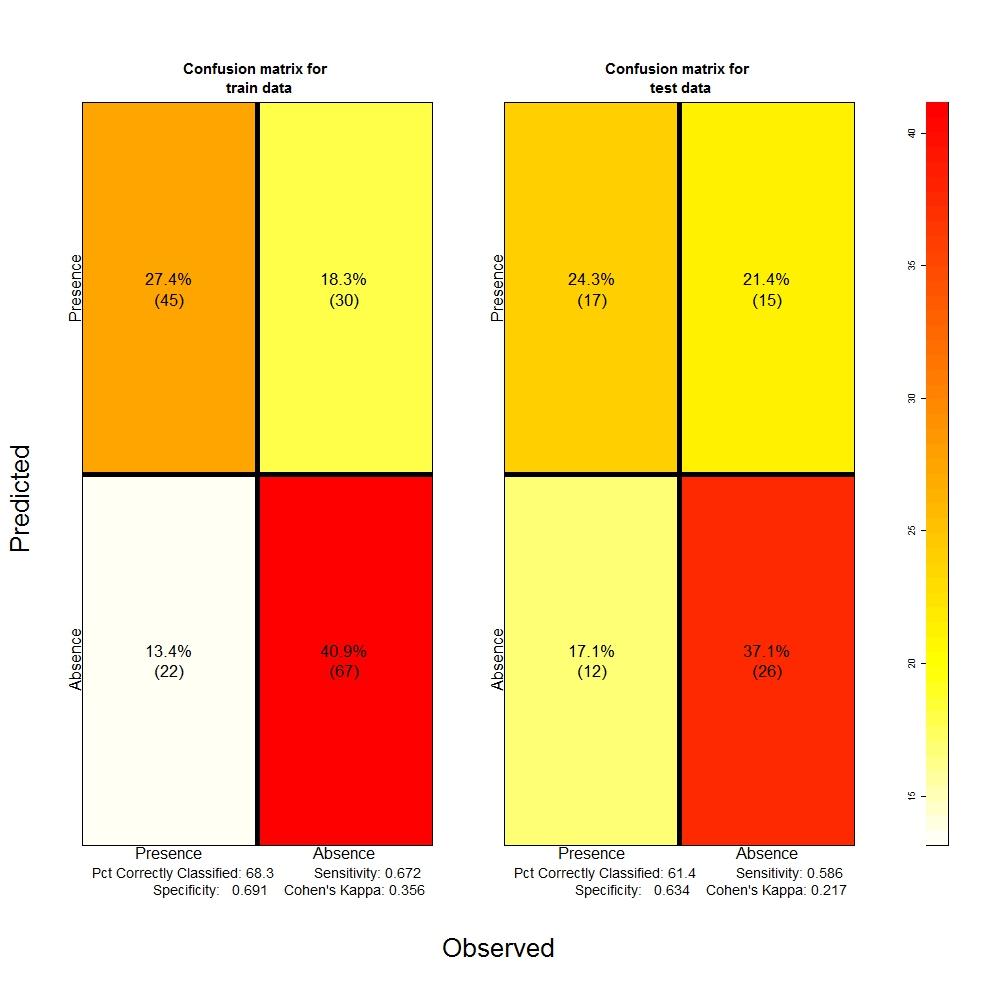


1. A calibration plot showing good calibration. Note that the logit curve follows the diagonal quite closely. The intercept is not significantly different than zero and the slope is not significantly different than 1.

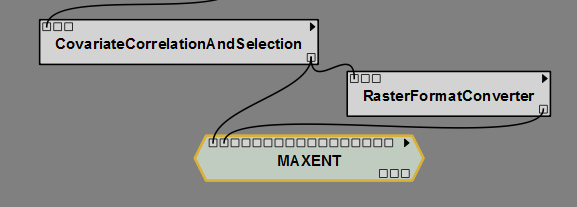


1. These plots show poorly calibrated models. Positive bias occurs when the predicted values are higher than the observed probabilities of occurrence. Negative bias occurs when the predicted probabilities of occurrence are lower than the observed probabilities. Spread error 1 indicates that predicted probabilities of occurrence are higher than than they should be .

“model”.confusion.matrix.jpg: The confusion matrix shows the percent of predicted and observed values in each of the presence and absence classes. For predicted values this is based on the threshold used to discretize the predicted values. If a test\training split or cross-validation was specified then the percentages for the training split and for the test or total over all evaluation folds will be shown in the same plot. Several evaluation metrics are based on the discretization of the continuous predictions and could be seen as accompanying this plot. These include the percent correctly classified, sensitivity, specificity, Cohen’s kappa, and the true skill statistic which are reported below the plot. 



1. Confusion matricies for a data with no evaluation split and with a test\training split. A good model fit will have relatively high percentages along the diagonals indicating that the model predicted presence and predicted absence agree well with the observed values of presence and absence and thus the model has good discrimination. Large discrepancies between the training and evaluation data in this plot could indicate model overfitting.
2. “model”\_response\_curves.pdf: Model response curves show the relationship between each predictor included in the model and the fitted values, while holding all other predictors constant at their means. MARS response curves are shown on a logit scale thus the response axis will not necessarily be bounded on the 0 to 1 interval. BRT response curves will show response surfaces for any interaction terms included in the final model along with the percent relative influence. These surfaces can be examined to assure that the relationships agree with the biological understanding of the organism under study. If several models under consideration exhibit similar performance in terms of their evaluation metrics consideration of the response curves can sometimes be used to select the better of the models.
3. “model”.resid.plot: Model residual plots show the spatial relationship between the model deviance residuals. Most models assume residuals will be independent thus spatial pattern in the deviance residuals can be indicative of a problem with the model fit and inference based on the fit. It can for example indicate that important predictors were not included in the model and can be compared with the spatial pattern of predictors that were not included in the model. Whether or not a significant spatial pattern exists in model residuals can at times be difficult to assess visually. We hope to add correlograms of Moran’s I soon. Unfortunately statistical tests based on the Moran’s I statistic for residuals of binary response models lack statistical justification and thus cannot be used to test for a significant spatial pattern (Bivand 2008). See Dormann 2007 for more discussion on evaluation of model residuals and spatial models that are appropriate for species distribution modeling. Residual plots can also be used to determine if certain observations contribute disproportionately to the deviance of the fitted model. For a binary response model deviance residuals with absolute values greater than 2 can be indicative of a problem.
4. “model”\_prob\_map.tif: If specified using MakeProbabilityMap=TRUE then a surface of predicted values is produced based on the tiffs in the input .mds file and the fitted model. These can but do not always indicate the probability of finding the species at a given site. For example if model calibration is poor then these will not agree well with the true probabilities though discrimination between presence and absences might still be good. Also if background points are used in place of true absence observations then calculation of the probability of occurrence is not generally possible.
5. “model”\_bin\_map.tif: If specified using MakeBinaryMap=TRUE then a surface of binary observations is produced by discretizing the probability using the best threshold given the selected threshold optimization. This map indicates whether one could expect each site to be occupied or unoccupied based on the model.
6. “model”\_MESS\_map and “model\_Mod\_Map: If specified by selecting makeMESMap=TRUE the the MESS and MOD surfaces will be produced. The MESS surface is the multivariate environment similarity surface and shows how well each point fits into the univariate ranges of the points for which the model was fit. Negative values in this map indicate that the point is out of the range of the training data. The MoD map is related and indicates which variable was furthest from the range of the observations used for model training. The MESS map takes the minimum value of a statistic calculate for each predictor and thus cannot diagnose hidden extrapolation as one might do using a hat matrix. This surface is only calculated for variables that are selected in the model selection step within each model fitting algorithm so that variables that do not significantly affect the occurrence of the organism over the range of the training data will not be included in the MESS map even though these predictors might be significant outside the range in which the model was fit. Random Forest never drops predictors so if one wishes to compare the MESS and Mod map before and after insignificant predictors were dropped, one can compare the MESS map of a Random Forest fit to that produced from the other model fit algorithms as long as they were fit using the same dataset. See Elith et. al. 2010 for details on how the MESS map calculations are performed.
7. Evaluation metrics appended to AcrossModel...csv and .jpg: An across model csv is produced to track several evaluation metrics across model runs and if at least two models have been run then a .jpg will accompany this. The name of the csv indicates the type of response as well as the type of model selection split that was specified and separate .csv’s and .jpg’s will be produced for each combination. The across model plot will show the value of several evaluation metrics for each model with x labels indicating the folder name where further output for model can be found. The best performing model for each of the training and evaluation data will have their metrics displayed. This plot will be used by the select and test the final models option and is discussed in more detail in that section.

****

### Maxent

Maxent is a machine learning technique that minimizes relative entropy between the probability densities estimated for the species and that estimated for the available environment. The model requires presence data for the species of interest and background locations representative of the sampled environment. Detailed information on the technique is provided in several papers including Philips et al 2004, Philips et al 2006, and Elith et al 2010. We incorporated the program described in those papers into SAHM, exposing the same options available in the GUI implementation. Detailed information about the program can be found at http://www.cs.princeton.edu/~schapire/maxent/.

### Select and Test the Final Model

If multiple models are run in the same workspace then model output is complied across models into a jpg and a csv. The select and test final model option (Packages drop down, SAHM, Select and Test the final model) can then be used to select from among the fitted models, evaluate these models on the data withheld using the model evaluation split, consolidate evaluation metrics for these best performing models, produce probability surfaces, binary surfaces, MESS and MoD maps. Once Select and Test the Final model has been chosen the user will be given the choice between any Appended Output files that were created for different combinations of model selection splits and response types. A graphic will pop up, if more than one model was selected, with the appended output results showing the best performing models marked with their numeric values for each metric. x labels indicate the folder name where all output for each model can be found. Folder output will be left blank for models that failed at some point during execution. If a test\train split was used for model selection evaluation metrics will be shown for each as bars and if a cross validation split was used then bar plots showing results from each fold will be shown. Once the desired models are selected then the requested output will be produced for the best performing models.

## Viewing Output

### Spreadsheet

VisTrails has a spreadsheet to view individual outputs side by side. Besides the built in cells for viewing text, webpages, etc SAHM has two specialized cells for viewing model outputs. The output from several model runs can be viewed side by side in individual spreadsheet cells.



****

### SAHM Model Output Viewer Cell

This widget is for viewing the textual and graph output from R. It contains tabs for the model output report, response curves, AUC plot, and calibration graph, confusion matrix and map of residuals. Within any of the graphs you can zoom in or out with the mouse wheel and pan by moving the mouse while holding down the left mouse.



### SAHM Spatial Output Viewer Cell

The SAHM Spatial Output Viewer Cell provides a convenient means for viewing the numerous spatial outputs produced by individual model runs as well as the input presence and absence points and background points if applicable. The spatial viewer displays the outputs in an interactive Matplotlib chart which functions much like a full GIS. ****

Attached to each cell is a toolbar that allows changing of the displayed layer and the overlaid points

**Spatial Output Viewer Toolbar:**

When panning you can zoom by holding a right click and moving the mouse up for zoom out or down for zoom in.

When using the Zoom rectangle a right click rectangle selection will zoom in, a left will zoom out.

Reference, 2000, National water quality inventory—1998 report, U.S. Environmental Protection Agency Report EPA–841–F–00–006, 45 p.

# Glossary

Glossary Term Glossary definition.

References

Baddeley, A. and Turner, R. (2005). Spatstat: an R package for analyzing spatial point patterns. Journal of Statistical Software 12 (6), 1-42. ISSN: 1548-7660.

Bivand, R.S., Pebesma, E.J., and Gόmez-Rubio, V. (2008). Applied Spatial Data Analysis with R. Springer New York, NY.

Dormann, C.F., McPherson, J.M., Araujo, M.B., Bivand, R., Bolliger, J., et al. (2007). Methods to account for spatial autocorrelation in the analysis of species distributional data: a review. Ecography 30:609–28.

Elith, J., Leathwick, J.R. and Hastie, T. (2008). A working guide to boosted regression trees. Journal of Animal Ecology, 77, 802–813.

Elith, J., Kearney, M., Phillips, S. (2010). The art of modeling range-shifting species. Methods Ecol Evol 1:330–342.

Elith, J., S. J. Phillips, T. Hastie, M. Dudík, Y. E. Chee, and C. J. Yates. 2011. A statistical explanation of MaxEnt for ecologists. Diversity and Distributions 17:43-57.

Fielding, A. H., and Bell, J. F. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation, 24, 38-49.

Freeman, E. (2007). PresenceAbsence: An R Package for Presence-Absence Model Evaluation. USDA Forest Service, Rocky Mountain Research Station, 507 25th street,

Ogden, UT, USA.

Grosjean, Ph. (2011). SciViews: A GUI API for R. UMONS, Mons, Belgium.

Hastie, T., Tibshirani, R., Friedman J.H., (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer-Verlag. 744 pp. 2nd ed.

Hastie, T. and Tibshirani., R. mda: Mixture and flexible discriminant analysis. Ported to R by Leisch, F., Hornik, K. and Ripley B. D. (2011). R package version 0.4-2.

Hijmans, R. J., and van Etten, J. (2011). raster: Geographic analysis and modeling with raster data. R package version 1.9-41.

Keitt, T. H. Bivand, R., Pebesma, E., and Rowlingson, B. (2011). rgdal: Bindings for the Geospatial

Data Abstraction Library. R package version 0.7-4.

Leathwick J.R., Elith, J., Hastie, T. (2006). Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. Ecological Modelling 199: 188-96

Liaw, A. and Wiener M. (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Miller, M.E., Hui, S.L., Tierney, W.M. (1991). Validation techniques for logistic regression models. Statistics in Medicine 10: 1213-26

Pearce, J., and S. Ferrier. (2000). Evaluating the predictive performance of habitat models developed using logistic regression. Ecological Modelling 133:225–245.

Pebesma, E.J., Bivand, R.S. (2005). Classes and methods for spatial data in R. R News 5(2).

Phillips, S. J., R. P. Anderson, and R. E. Schapire. 2006. Maximum entropy modeling of species geographic distributions. Ecological Modelling 190:231-259.

Phillips, S. J., M. Dudik, and R. E. Schapire. 2004. A maximum entropy approach to species distribution modeling. Pages 655-662 in Proceedings of the Twenty-First International Conference on Machine Learning.

R Development Core Team (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Ridgeway, G. (2010). gbm: Generalized Boosted Regression Models. R package version 1.6-3.1.

Therneau, T. and original Splus->R port by Lumley, T. (2011). survival: Survival analysis, including penalised likelihood. R package version 2.36-10.

Appendix A: SAHM VisTrails Tutorial, modeling cheatgrass in Rocky Mountain Park.

# Start with the input data:

1. For our example we obtained cheatgrass (Bromus tectorum) presence absence data from the National Institute of Invasive Species Science (NIISS) <http://www.niiss.org/cwis438/websites/niiss/home.php?WebSiteID=1> Nick please add in instructions, information about NIISS, etc.
2. In VisTrails drag in a FieldData Module from the DataInput section. In the Module Information pane to the right click on the 'value' port on the Inputs tab. This opens a path box below the port name. Click on the folder icon to navigate to the file you downloaded from NIISS, or use the RMNP\_Bromus.csv supplied with the SAHM\_example.
3. Since the file format that NIISS supplies does not match that used by SAHM we must reformat our input using the FieldaDataQuery module. Drag a FieldaDataQuery from the Tools section onto our canvas. Connect the output….