Ecological Niche Modeling in Kepler

The *Ecological Niche Modeling in Kepler* guide is a tutorial style manual for scientists who want to execute and/or modify existing scientific workflows for this type of analysis.

Recommended prerequisite: *Getting Started with Kepler*

Table of Contents

1. Introduction

The Ecological Niche Modeling (ENM) Guide introduces the main approaches used in niche modeling analysis and describes the relevant components and workflows in Kepler. The guide contains instructions for using and modifying example ENM workflows. Once you are familiar with the general principles and workflows used in ENM, we recommend that you work through a couple of the demo workflows to get a feel for how easy it is to use and modify workflow components and how different sub-workflows can be combined to form novel analyses.

1.1. What is Ecological Niche Modeling?

Ecological niche modeling (ENM) is a conceptual framework for understanding and anticipating geographic and ecological phenomena related to biodiversity (Soberon and Peterson, 2005). The ecological niche of a species can be defined as the conjunction of conditions within which a species can maintain populations without input via immigration Grinnell (1917). Extensive research by diverse investigators has built the case that niches can be estimated based on associations between known geographic occurrences of species and features of landscapes summarized in digital GIS data layers (Huntley et al. 1995, Pearson et al. 2002, Peterson 2003, Araujo et al. 2005; *Figure 1A*).

The ability to predict ecological and geographic phenomena using ENM generates many opportunities for investigators. ENM can be used to model distributions of species in ecological space (*Figure 1B*), offering a view of the ecological requirements of a suitable habitat (e.g., annual precipitation is more than 5 mm and less than 12 mm). The ecological requirements can then be projected onto geographic space to interpolate a predicted geographic distribution of a species (*Figure 1C*). Models can incorporate scenarios of change, predicting the distribution of species introduced to new locations (i.e., the potential distribution of an invasive species) (*Figure 1D*), or the distribution of a species under changed climate conditions (*Figure 1E*).

The general approach used by ecological niche modeling is the following:

1. Predict species distributions based on collected occurrence data and environmental information (i.e., model the ecological data).
2. Validate predictions based on present-day species' occurrence data
3. Predict potential species distributions under future climate scenarios.

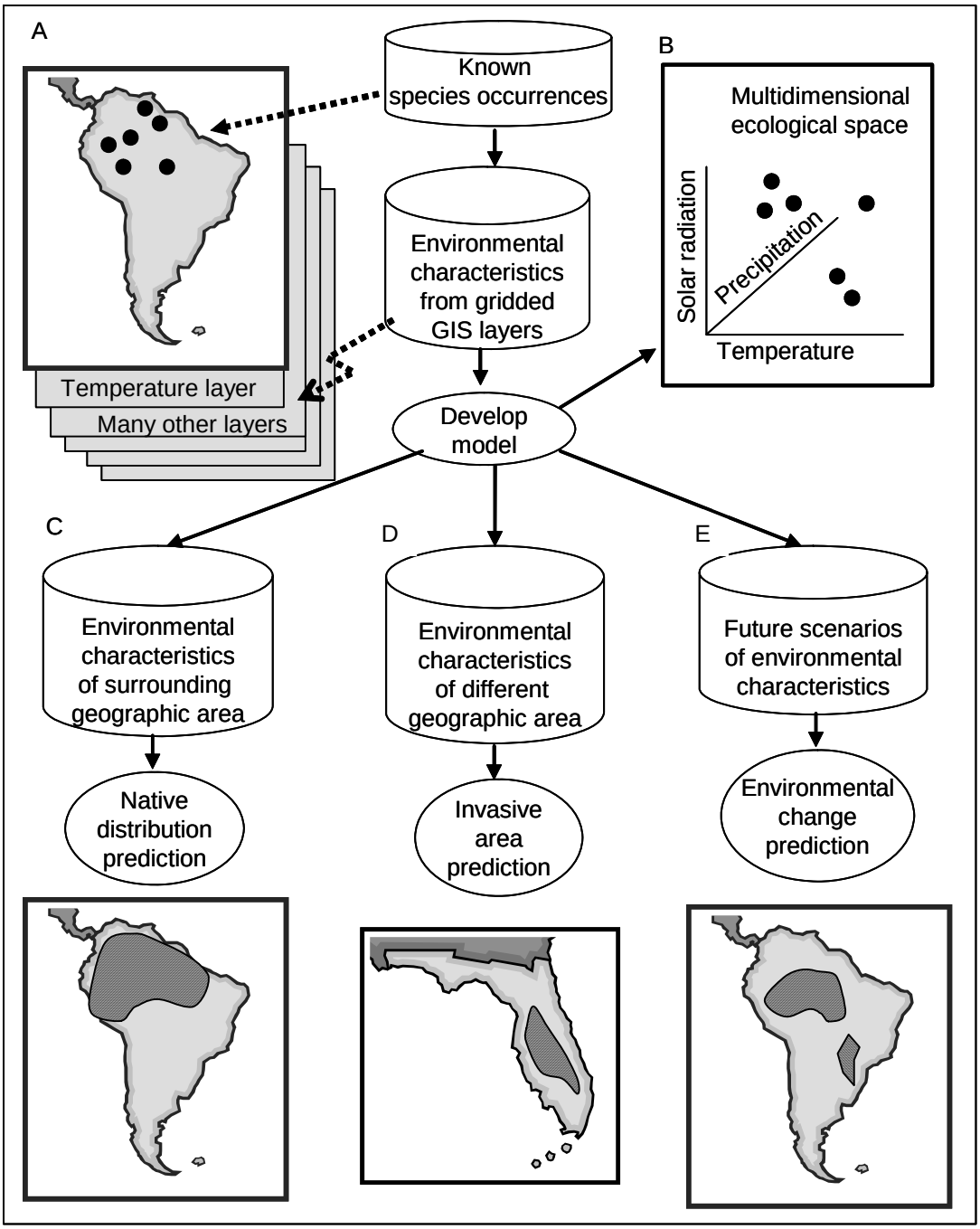


Figure 1: Ecological niche modeling approach and applications (modified from Pennington et al. 2007) (A) species' occurrence data are analyzed with environmental data to (B) identify the conditions that define a suitable habitat for the species (appropriate temperature, precipitation, etc). (C) This model is "mapped" to geographic space to predict the native distribution of the species; (D) The model is mapped to a new geographic space to predict the potential distribution of invasive species; and (E) the model is applied to a future climate scenario to predict the distribution of suitable habitat under the new conditions.

Numerous conceptual approaches and software tools can be used in ENM. Some very simple tools have seen very broad application (Nix, 1986). Further developments of niche modeling tools proceeded along two main lines: (1) multivariate statistical tools beginning with logistic regression (Mladenoff et al., 1995) and progressing through generalized linear and generalized additive models (Elith and Burgman, 2002); and (2) evolutionary computing applications such as genetic algorithms (Stockwell and Peters, 1999), neural networks (Pearson et al., 2002), and maximum entropy approaches (Phillips et al., 2004). Each of these two classes has its advantages and disadvantages for niche modeling, but the basic message is that many computational options exist for modeling ecological niches.

Although the number of studies using ENM is large---see a recent review and meta-analysis (Thomas et al., 2004) ---most have been limited by practical and technical limitations to between a few dozen and a few hundred species. The largest such study to date (Peterson, 2002) reviewed approximately 1800 species of Mexican birds, mammals, and butterflies.

1.2. History of Ecological Niche Modeling in Kepler

The National Science Foundation-funded Science Environment for Ecological Knowledge (SEEK) project -- the initial contributor to the Kepler project -- chose ENM as the prototype Kepler application. SEEK selected this application because there were clear gains to be made through applying cutting-edge technology to niche modeling. The scientists recruited for collaboration included approximately 20 leading ENM researchers from around the world (US, Mexico, Europe, South Africa, Australia, New Zealand and others). SEEK held three working meetings with this group: 1) a full group meeting where analysis and modeling tasks were discussed and important datasets, algorithms, and analytical environments were identified, 2) a small working group meeting where a specific research problem was fully specified from a workflow perspective, and 3) a full group meeting where the community was exposed to initial, prototype solutions.

Through the first two meetings, technical problems associated with ecological niche modeling were identified as:

- labor-intensive data preprocessing and preparation;
- distributed data, archived in dozens of museum collections and environmental repositories;
- heterogeneous computing environments required at different points in the workflow, including C and Java programs, statistical scripts written in MATLAB, SAS and/or R, and Geographic Information System (GIS) analyses;
- compute-intensive algorithm execution for multiple species, under multiple parameter sweeps. When the algorithm is stochastic, constructing a distribution of results for statistical comparison requires many iterations.

The outcomes of these meetings were two: 1) clearly delineated needs, allowing developers to focus on specific, articulated problems and test solutions against real problems, and 2) a showcase project designed both to demonstrate the power of scientific workflows in solving large-scale computational problems and to shed light on a still-open question: What is the magnitude of likely climate change effects on biodiversity across the Americas?

The project made use of the data resources of the distributed Mammal Networked Information System (MaNIS; Stein and Wieczorek, 2004) to carry out a review of likely climate change effects on the over 2000 mammal species of the Americas, constructing maps of potential species distributions under future climate scenarios. This analysis will be the broadest in taxonomic and geographic scope carried out to date, and the computational approach (the Kepler workflow) will be completely scalable and extensible to any region and any suite of taxa of interest.

2. Example ENM workflows

The basic operations necessary for ENM are straightforward and consist of data access (which may involve transforming data or merging data grids), sampling, model construction and validation, and model prediction in current or changed conditions. Each of these high-level steps requires numerous detailed steps that comprise one or more sub-workflows. Sub-workflows are standalone workflows that can be useful by themselves or combined to construct a more complex, nested workflow. The sub-workflow architecture used in Kepler allows maximum flexibility for creative workflow design and innovation.

This section introduces how the prototype workflow is constructed and broken into steps. *Figure 2* contains a high level overview of the ENM workflow; *Figure 3* delves into greater detail about how the workflow decomposes to a number of more complicated steps. Actual Kepler workflows—for single and multiple species—are described in Sections 2.1 and 2.2.

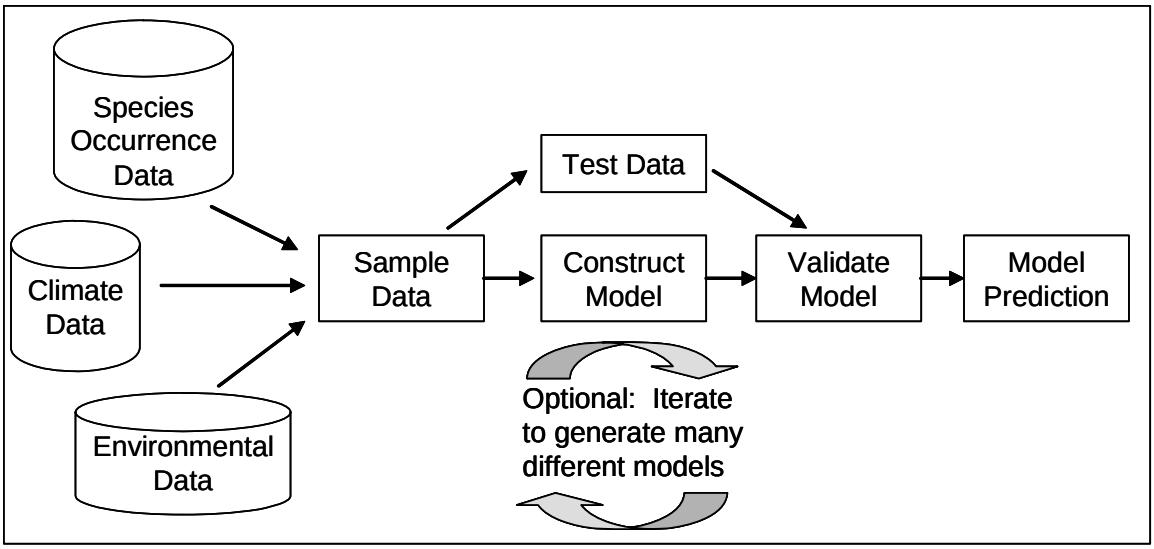


Figure 2: High-level ENM steps, each of which is decomposed into one or more sub-workflows that accomplish the details of that step. Depending on the modeling algorithm selected, the final workflow may be linear or iterative and include substantial looping.

The high-level ENM workflow decomposes to a number of more complicated steps (*Figure 3*) that are carried out by one or more sub-workflows (discussed in more detail in Sections 3 and 4). *Figure 3A* shows how each stream of data (the species occurrence, climate, and environmental data) is transformed for use in the workflow. *Figure 3B* shows more specifically how the predictive model is constructed and validated: A stochastic genetic algorithm (GARP; Stockwell and Peters, 1999) generates a different model for each workflow iteration. Hundreds or thousands of iterations may be run, from which the best subset of models is selected for prediction. *Figure 3C* shows how the models are used to generate many distribution maps for both current and future environmental conditions that are combined into maps of probability of occurrence. The ENM sub-workflows were constructed for the prototype application, but most are re-usable for similar applications.

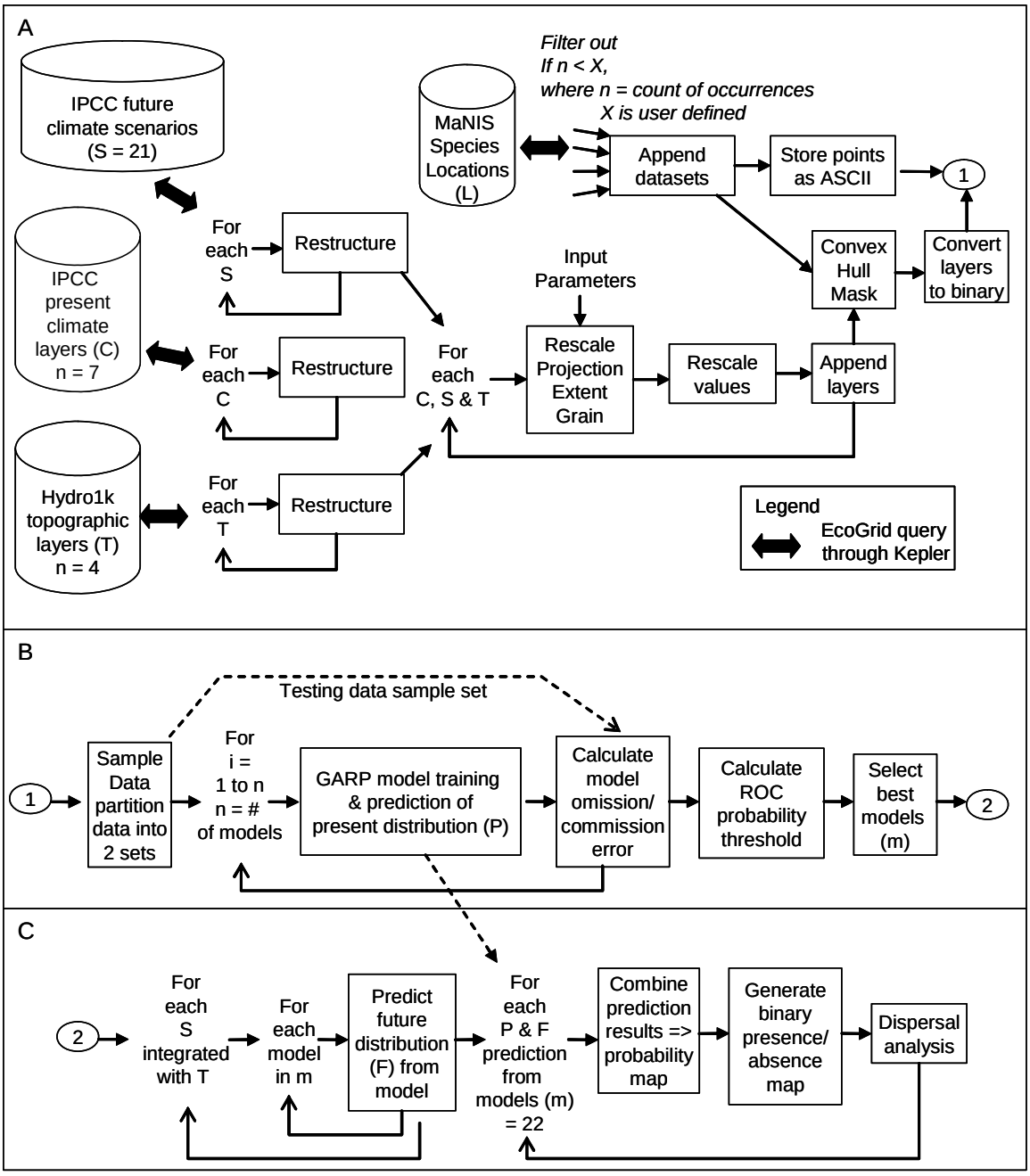


Figure 3: Decomposed prototype application. A) Data preparation. B) Sampling, model construction, and model validation. C) Prediction in current and future climate scenarios.

2.1.Single species distribution

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml

The single species workflow (*Figure 4*) included with the Kepler application predicts the probability of occurrence of a single species, *Mephitis mephitis*, based on collected occurrence points (see Section 3.1), IPCC climate data (see Section 3.2), and Hydro1 topographical data (see

Section 3.3). The workflow "bundles" the data into a format that can be sampled (see Section 3.5) and creates a "mask" file used to isolate a specific geographic extent (see Section 3.6). After the data have been prepared, the workflow uses the GARP (Genetic Algorithm for Rule Set Production) algorithm to generate models, which are tested, and then used to project the predicted species' distribution onto a complete map of North and South America (see Section 4). Predictions are made using both historical and future IPCC climate data (*Figure 5*).

GARP is a stochastic algorithm; each time it is run on a given dataset, it will generate a different model. Typically, the algorithm is run hundreds of times for a given species dataset, the results are analyzed to determine the best models, and the results of the best models are compiled and returned.

NOTE: In the interest of minimizing run-times, the example workflow iterates the GARP algorithm only 10 times, which is insufficient for reliable statistics. To increase the number of iterations, change the value of the `number_Of_Iterations` parameter before running the workflow.

NOTE: The workflow requires that R, an application used for statistical analyses, be installed on the local system. R is included with the Kepler installer for Windows; currently, the installer for Mac does not include R. For more information about downloading and installing R, see http://www.r-project.org/.

NOTE: Only some of the future climate data are included in the sample data processed by the workflow (only data for 2020 are included).

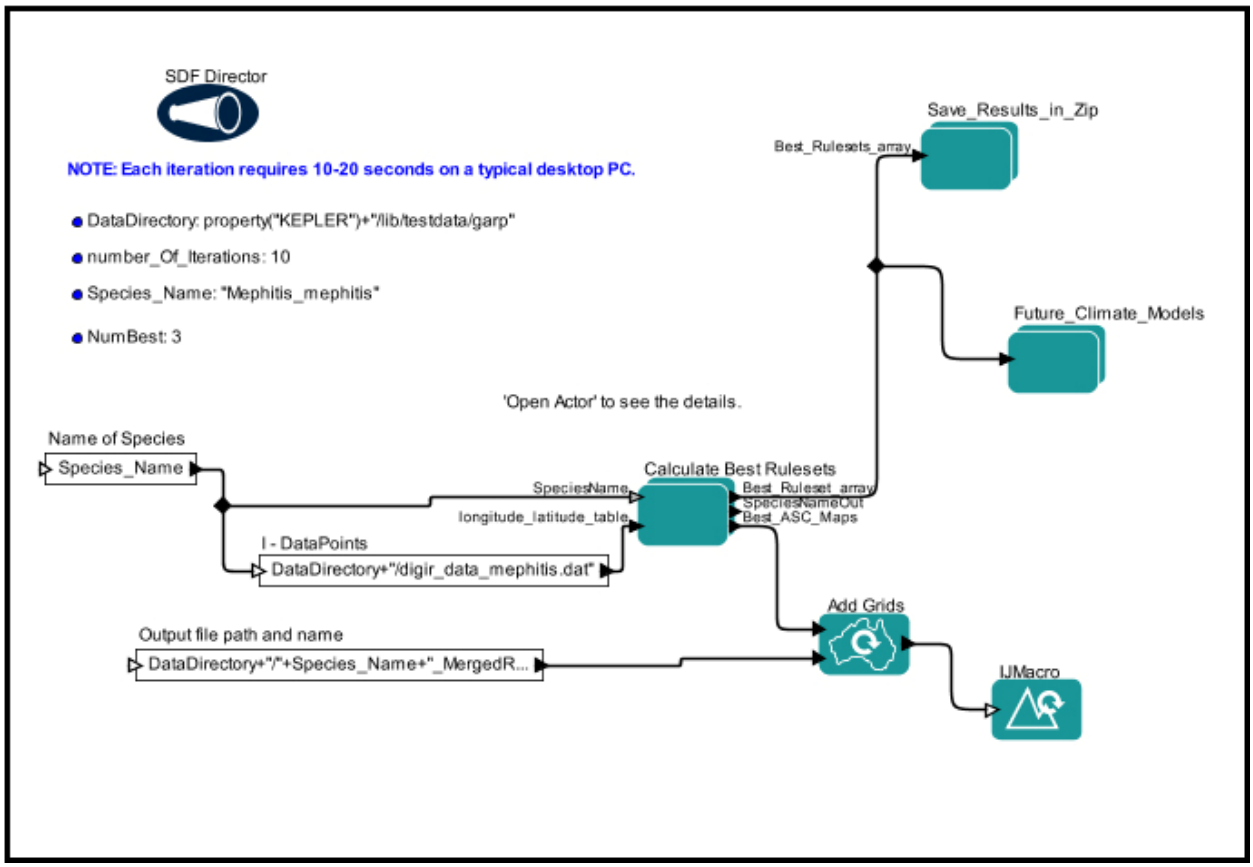


Figure 4: The GARP_SingleSpecies_BestRuleSet-IV.xml workflow, discussed in more detail in Sections 3 and 4.

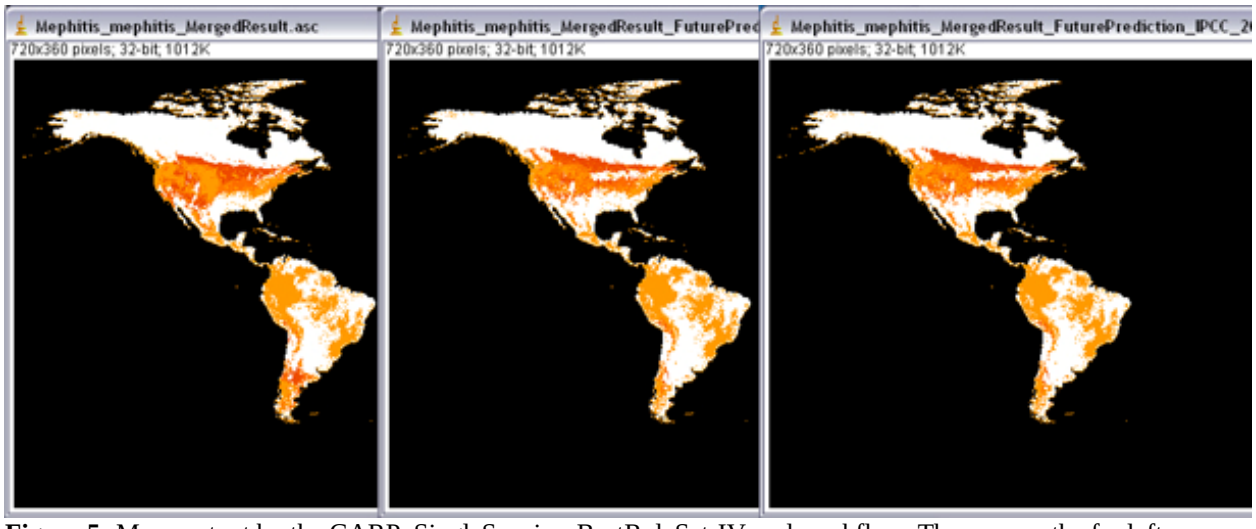


Figure 5: Maps output by the GARP_SingleSpecies_BestRuleSet-IV.xml workflow. The map on the far left displays a predicted distribution of *Mephitis mephitis* based on historical climate data. The map in the center displays a prediction based on future climate data for 2020. The map on the far right displays a prediction based on future climate data for 2050. The workflow also outputs a list of files used to generate the predictions (not pictured).

The GARP prediction (*Figure 5*) is displayed using ImageJ, an application used to display and process a wide variety of images (tiffs, gifs, jpegs, etc.) For more information about ImageJ, see

http://rsb.info.nih.gov/ij/. Areas with a higher probability of presence are displayed in a brighter color. White indicates areas where no prediction can be determined based on the data.

2.2. Multiple species distribution

See $KEPLER/demos/SEEK/GARP_MultipleSpecies-V.xml

The multiple species distribution workflow (*Figure 6*) included with Kepler extends the single species workflow discussed in Section 2.1 so that it can process multiple species. The workflow generates three maps for each species: a predicted distribution using historical climate data, a predicted distribution using future climate data for 2020, and a predicted distribution using future climate data for 2050. The example workflow runs predictions for *Mephitis mephitis* and a species of *Zapus* (*Figure 7*).

NOTE: In the interest of minimizing run-times, the example workflow iterates the GARP algorithm only 10 times, which is insufficient for reliable statistics. To increase the number of iterations, change the value of the `number_Of_Iterations` parameter in the *Single Species GARP Model* actor before running the workflow.

NOTE: The workflow requires that R, an application used for statistical analyses, be installed on the local system. R is included with the Kepler installer for Windows; currently, the installer for Mac does not include R. For more information about downloading and installing R, see http://www.r-project.org/.

NOTE: Only some of the future climate data are included in the sample data processed by the workflow (only data for 2020 are included).

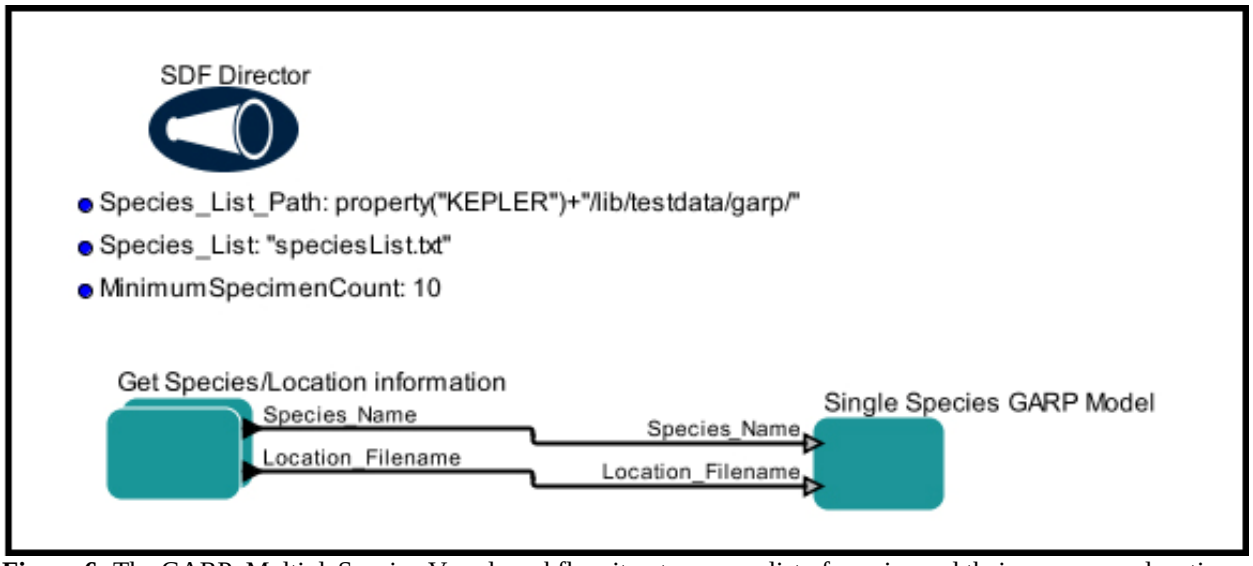


Figure 6: The GARP_MultipleSpecies-V.xml workflow iterates over a list of species and their occurrence locations, which are indexed in a text file. Parameters specify the location and name of the list file, as well as a threshold value specifying the minimum number of occurrence locations that must be available before the data will be processed.

The deceptively simple top level of the GARP_MultipleSpecies-V.xml workflow consists of two composite actors: one that iterates over a list of species and their associated occurrence point files, and a second that contains a set of nested sub-workflows that comprise a full ENM workflow (i.e., a slightly modified version of the GARP_SingleSpecies_BestRuleSet-IV.xml workflow). See Section 3.1.2 for more information about preparing species/location information for use with this workflow.

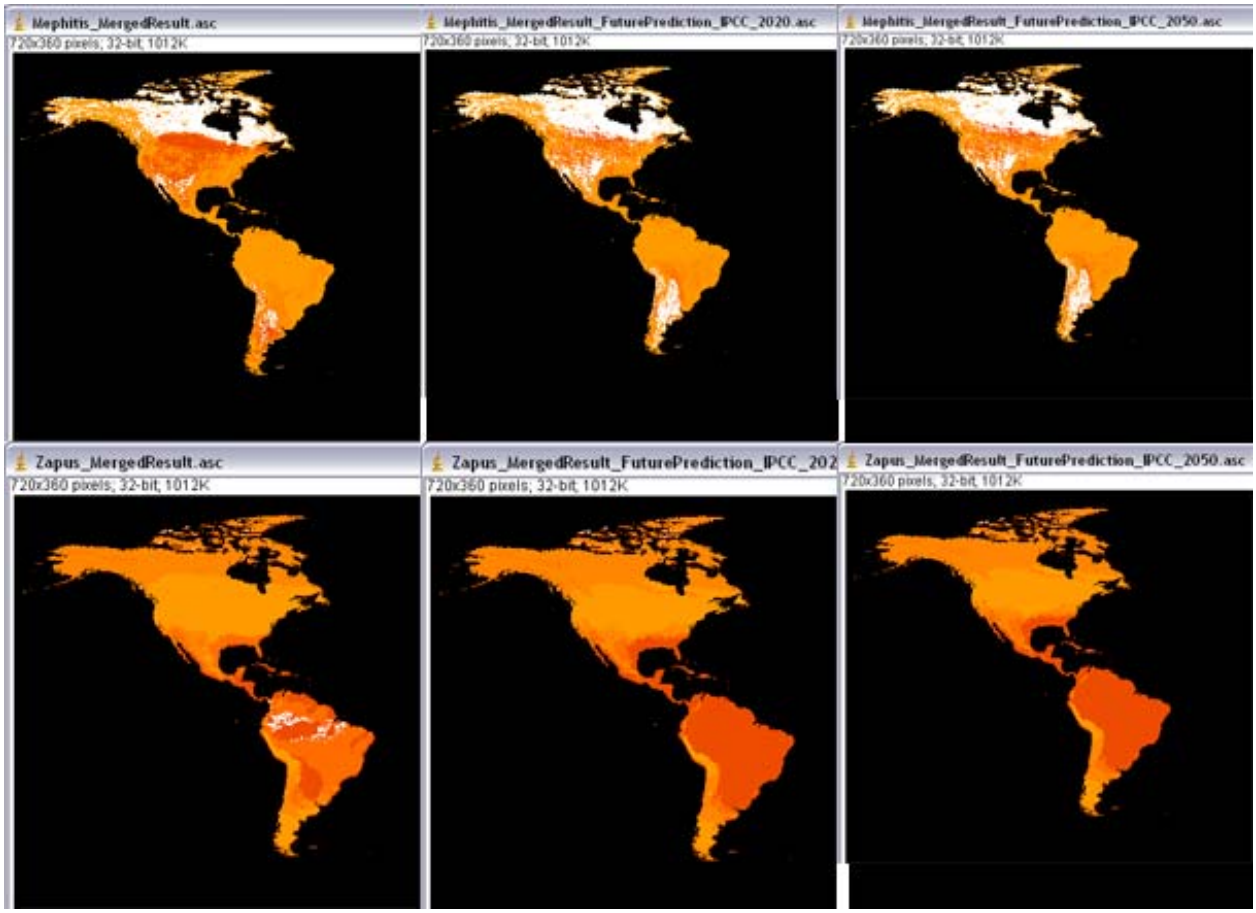


Figure 7: Maps output by the GARP_MultipleSpecies-V.xml workflow. The example workflow outputs three maps for each of the two species run*: Mephitis mephitis* (top) and *Zapus* (bottom). The map on the far left displays a predicted distribution based on historical climate data. The map in the center displays a predicted distribution based on future climate data for 2020. The map on the far right displays a predicted distribution based on future climate data for 2050. The workflow also outputs a list of files used to generate the predictions (not pictured).

The GARP prediction (*Figure 7*) is displayed using ImageJ, an application used to display and process a wide variety of images (tiffs, gifs, jpegs, etc.) For more information about ImageJ, see http://rsb.info.nih.gov/ij/. Areas with a higher probability of presence are displayed in a brighter color. White indicates areas where no prediction can be determined based on the data.

3. Data access and preparation workflows

Data used by the ENM workflows consist of a set of species occurrence points and a set of "environmental layers", which are geospatial grid files that contain information about climate, topography, and any other environmental data of interest. Transforming source data into properly formatted environmental layers often involves several steps, and in most cases the environmental layers should be processed before the ENM workflow is run (See Sections 3.2–3.6). Since environmental layers change infrequently, they can be stored and used many times in any given analysis or for multiple different niche analyses.

A global, 0.1 degree resolution set of environmental layers from the historical IPCC climate and Hydro1K data has been prepared and is available on the Kepler EarthGrid. See Section 3.4 for more information about accessing and using that dataset.

3.1. Species occurrence points

Species occurrence data is captured as point data, with each point consisting of a longitude and latitude of occurrence. These data can be obtained from any source, and may require reformatting before they can be used in ENM workflows. This section explains how to integrate species occurrence data from a text file or from distributed museum collections.

3.1.1 Single species: Text file

(See $KEPLER/demos/SEEK/garpModel_ImageJ.xml)

A plain ASCII text file can be used to input occurrence points for a particular species. The garpModel_ImageJ.xml workflow (*Figure 8*) uses occurrence data from a text file. The workflow passes the occurrence data, as well as a set of environmental layers, to three GARP actors that create a model, which can be used with other models to predict a species' distribution.

Note: GARP is a stochastic algorithm; each time it is run on a given dataset, it will generate a different model. Typically, the algorithm is run hundreds of times for a given occurrence dataset, the results are analyzed to determine the best models, and the results of the best models are compiled and returned. The garpModel_ImageJ.xml workflow iterates the GARP algorithm only once, generating a single model. Use the GARP_SingleSpecies_BestRuleSet-IV.xml workflow discussed in Section 2.1 to iterate the GARP algorithm and generate a more accurate prediction.

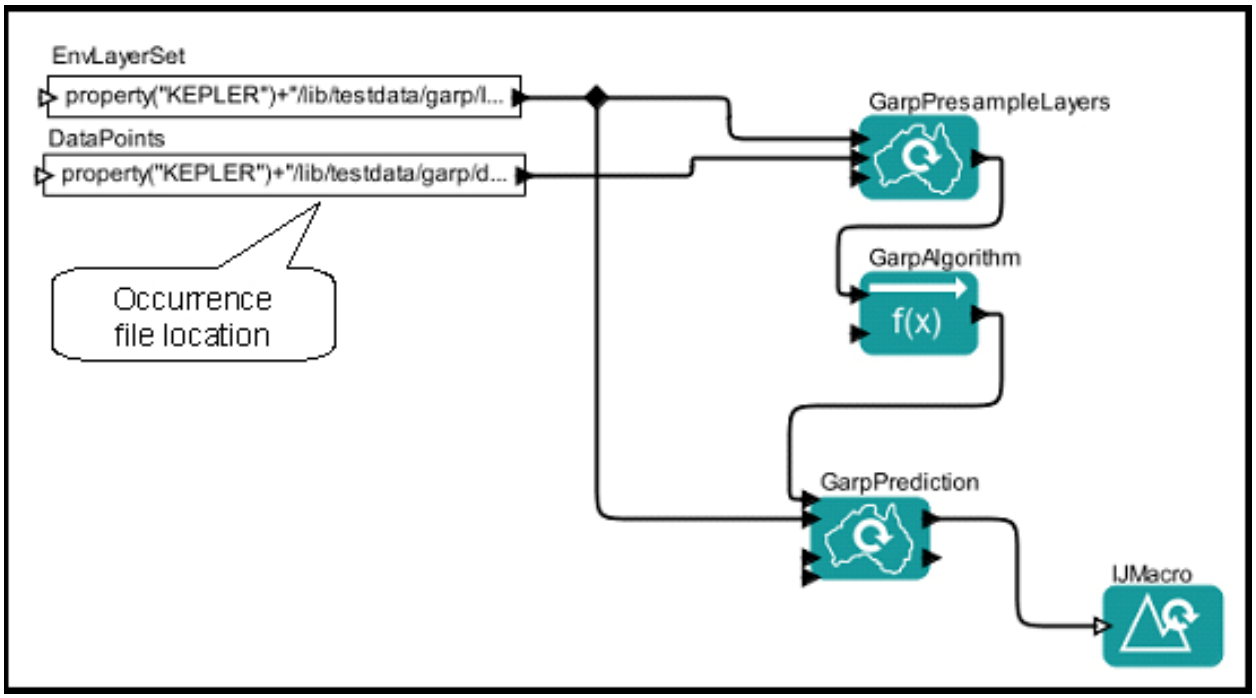


Figure 8: The garpModel_ImageJ.xml workflow reads species occurrence data from a text file.

In the above workflow, occurrence data is input via an *Expression* actor labeled "DataPoints." To see the full path of the data file, open the workflow in Kepler and double-click the *DataPoints* actor. Note that the expression '`property("KEPLER")`' returns the path to the directory in which Kepler is installed. Using an expression of this type allows the path to be evaluated properly no matter where the Kepler system is installed in the file system. In this example, the text file is in the "lib/testdata/garp" sub-directory of the Kepler installation, and the name of the file is "digir_data_mephitis.dat".

The data file consists of a set of occurrence points for the species *mephitis* (skunk), one occurrence point per line, listed as tab-delimited longitude/latitude combinations. If the file is opened with a text editor, the occurrence data will look something like the following:

```
-121.9182	46.742
-122.66639	47.29028
-124.25833	48.25472
-67.4607	44.6707
-78.8802	39.6965
-98.80037	35.75488
-98.80037	35.75488
-124.10306	48.20444
-124.10306	48.20444
…
```

The garpModel_ImageJ.xml workflow generates a predicted distribution for *mephitis* based on the input occurrence data and environmental layers, and displays the prediction using an open source image processing package called ImageJ (*Figure 9*).

Note: ImageJ is an image processing system created by the National Institute of Health (http://rsb.info.nih.gov/ij/). The *ImageJ* actor can be used to process and display many types of images; simply pass the image file name to the actor's input port.

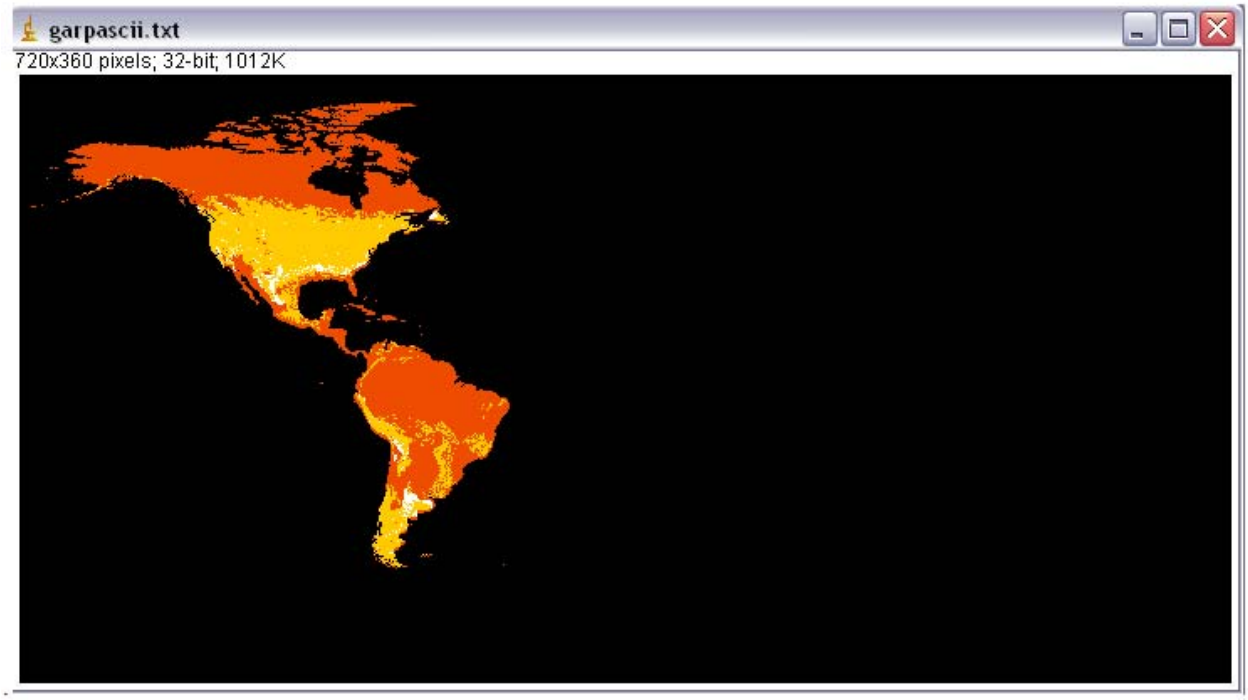


Figure 9: The output of the garpModel_ImageJ.xml workflow. Predicted species presence is indicated with color: brighter colors indicate a higher probability of presence. White indicates areas where no predication can be determined from the data.

3.1.2 Multiple species: Text file

(See $KEPLER/demos/SEEK/GARP_MultipleSpecies-V.xml)

Workflows can read occurrence data for multiple species using text file sources. The GARP_MultipleSpecies-V.xml workflow (*Figure 6*), introduced in section 2.2, is an example of this usage.

The deceptively simple top level of the GARP_MultipleSpecies-V.xml workflow consists of two composite actors: one that iterates over a list of species and their associated occurrence point data files (*Get Species/Location information*), and a second that contains a set of nested sub-workflows that comprise a full ENM workflow (*Single Species GARP Model*). Parameters specify the location and name of a text file that contains the species index list, which is a straightforward list consisting of species' names and the names of corresponding occurrence data files:

Mephitis,digir_data_mephitis.dat
Zapus,digir_data_zapus.dat
…etc…

A parameter (*MinimumSpecimenCount*) also specifies the minimum number of occurrence points required for a species to be run through the workflow. In the example workflow, the parameter has a value of 10; any species that has fewer than 10 occurrence points will not be run by the workflow.

Right-click the *Get Species/Location Information* actor and select Open Actor to display the sub-workflow (*Figure 10*). The sub-workflow uses three *Expression* actors *(Path, FileName,* and *minimumCnt*) to input the file name and path of the occurrence data, as well as the required minimum number of occurrence points, to an *RExpression* actor. The *RExpression* actor iterates over the list of species, locates each occurrence data file, reads through the data, and checks the number of occurrence points against the minimum threshold. The *RExpression* actor saves the "valid" species to a new index file: the species name and the path to its occurrence dataset are separated by a comma; each species is saved on a separate line.

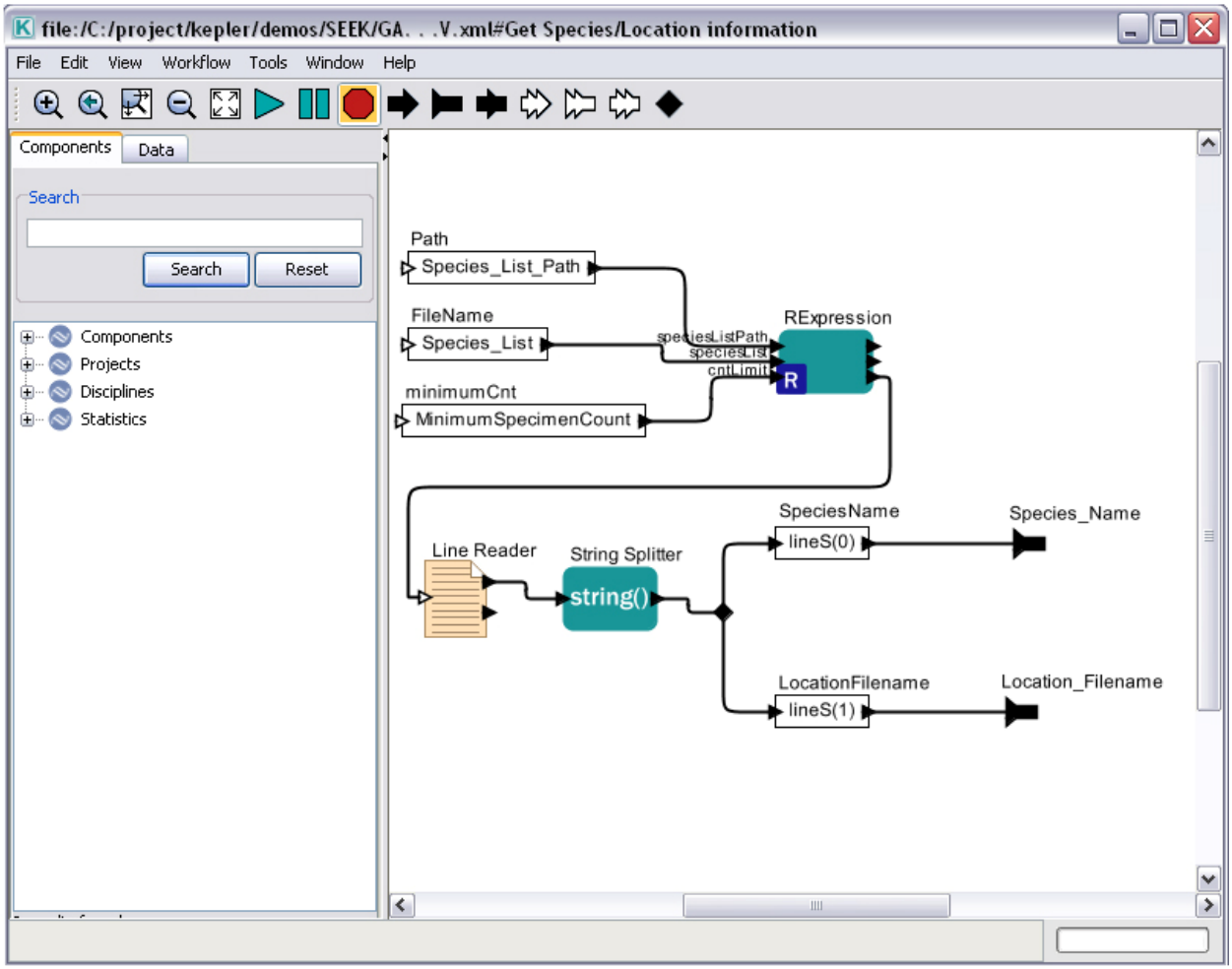


Figure 10: The *Get Species/Location Information* sub-workflow. Multiple sets of species' occurrence data are passed to the *RExpression* actor, which passes valid species' names and occurrence information to the remainder of the ENM workflow.

The *RExpression* actor outputs the file path of the revised species data index to a *Line Reader* actor that reads the file and outputs each line as a string. The output strings are split by the *String Splitter* actor, which splits strings at a character or characters specified by its `Regular expression` parameter. In this example, the character is a comma, ",". The actor outputs an array containing the split segments (e.g., { Mephitis,digir_data_mephitis.dat}). A relation is used to branch the actor's output so that it can be sent to multiple places in the workflow, in this case, to two *Expression* actors: *SpeciesName* and *LocationFilename*.

The *SpeciesName* and *LocationFilename* actors use the Ptolemy expression language to access the appropriate data. The value of the *SpeciesName* actor, `lineS(0)`, refers to the first element in the array passed to the actor via its `lineS` port. The value of the *LocationFilename* actor, `lineS(1)`, refers to the second element in the array. The species name and the path to its occurrence data are output by the workflow.

.

3.1.3 Single species: Distributed museum collections

(See $KEPLER/workflows/eco/garpModel_ImageJ_withData.xml)

The garpModel_ImageJ.xml workflow discussed in Section 3.1.1 can be modified to acquire species occurrence data from museum collections using the Distributed Generic Information Retrieval (DiGIR) protocol, an open source client/server protocol for retrieving distributed information using HTTP, XML, and UDDI (http://digir.net). The modified workflow, garpModel_ImageJ_withData.xml, is displayed in *Figure 11*.

Note: GARP is a stochastic algorithm; each time it is run on a given dataset, it will generate a different model. Typically, the algorithm is run hundreds of times for a given species dataset, the results are analyzed to determine the best models, and the results of the best models are compiled and returned. The garpModel_ImageJ_sithData.xml workflow iterates the GARP algorithm only once, generating a single model. Use the GARP_SingleSpecies_BestRuleSet-IV.xml workflow discussed in Section 2.1 to iterate the GARP algorithm and generate a more accurate prediction.

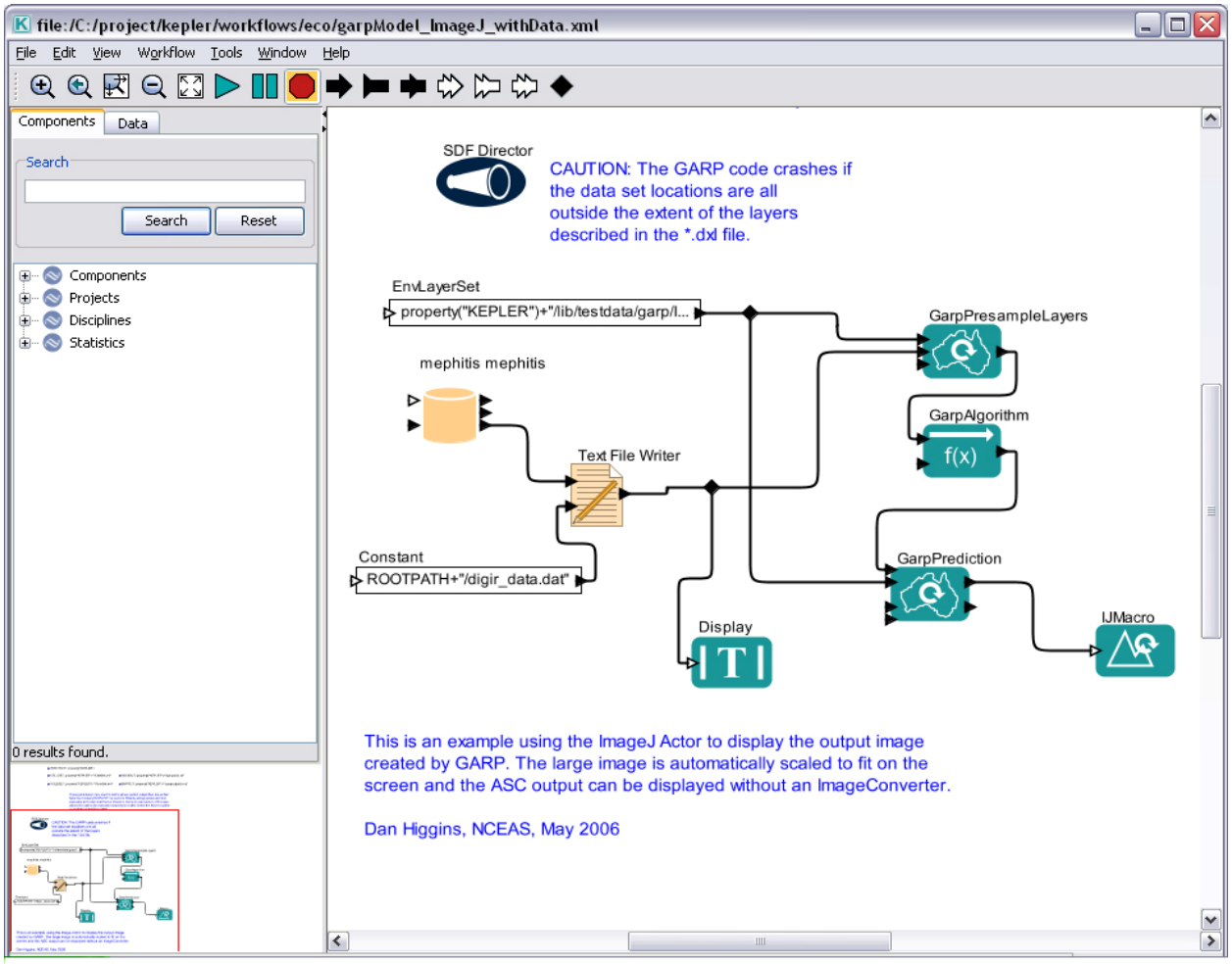


Figure 11: The garpModel_ImageJ_withData.xml workflow, which reads species' occurrence data from a museum collection using the DiGIR protocol.

The actor labeled “mephitis mephitis” is a *DarwinCoreDataSource* actor, which is obtained by searching for "mephitis mephitis" under the Kepler Data tab. The search returns any datasets that include the search term, hence datasets for subspecies such as 'mephitis mephitis avia' and 'mephitis mephitis estor' will be returned in addition to species-level data. The data within these sub-species datasets are included in the species-level dataset; therefore, they can be ignored.

Data returned via a search and dragged onto the Workflow canvas are stored temporarily in a cache and passed to the next actor as a single string. The garpModel_ImageJ_withData.xml workflow uses the *TextFileWriter* actor to write a permanent text file. The name of the text file is provided by the *Constant* actor (*Figure 12*). The *TextFileWriter* actor outputs the contents of the file to a *Display* actor, which displays the occurrence data.

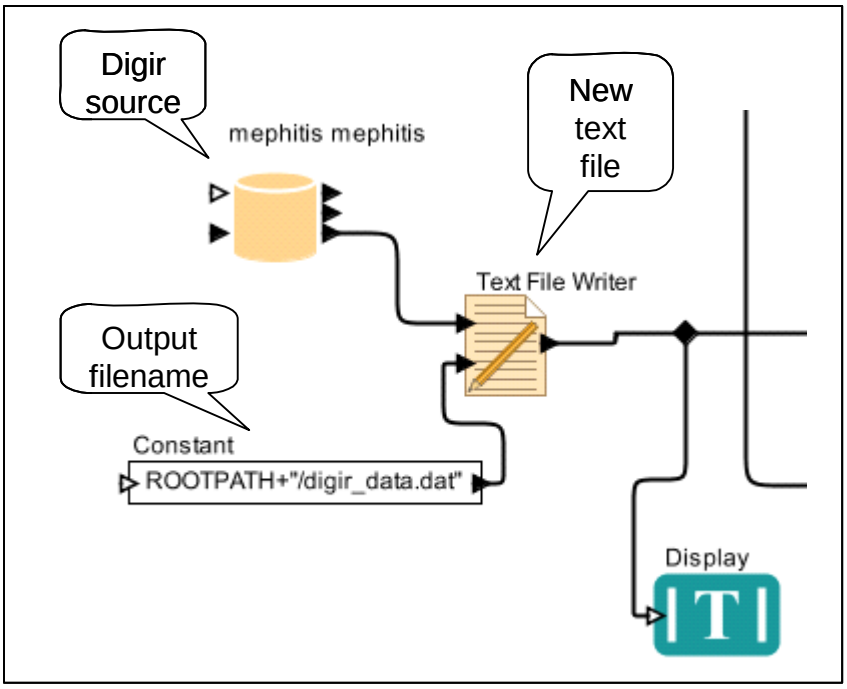


Figure 12: Using the DiGIR protocol to retrieve occurrence data from distributed museum collections.

.

The *DarwinCoreDataSource* actor (i.e., *mephitis mephitis*) returns much more information than the workflow requires. Right-click the *mephitis mephitis* actor and select the "Open Actor" menu item to see a list of the data fields that are returned. Returned field names and corresponding data types are listed at the top of the screen. Beneath the list of fields is a SQL Query Builder that can be used to filter the returned data (*Figure 13*). In this example workflow, the SQL Query Builder is used to reduce the returned information to only those fields required by the workflow.

The Query Builder is used to select a "Table" (e.g., Digir Table:mephitis mephitis) , a desired "Field" (e.g., DecimalLatitude), and to specify whether that field should be included in the selection. The Query Builder's Operator and Criteria settings specify additional selection criteria. In this example, every record with a longitude and latitude is desired, and the selection criteria are designed to select any record with valid values for both (i.e., any record with latitude or longitude less than 360 degrees). The only records that will be excluded are those without entries, or those that have invalid entries, for longitude or latitude.

Note: Valid latitudes should be in the range of -90 to +90, but missing data is often coded as a very large number, so 360 degrees works to screen invalid records in most cases.

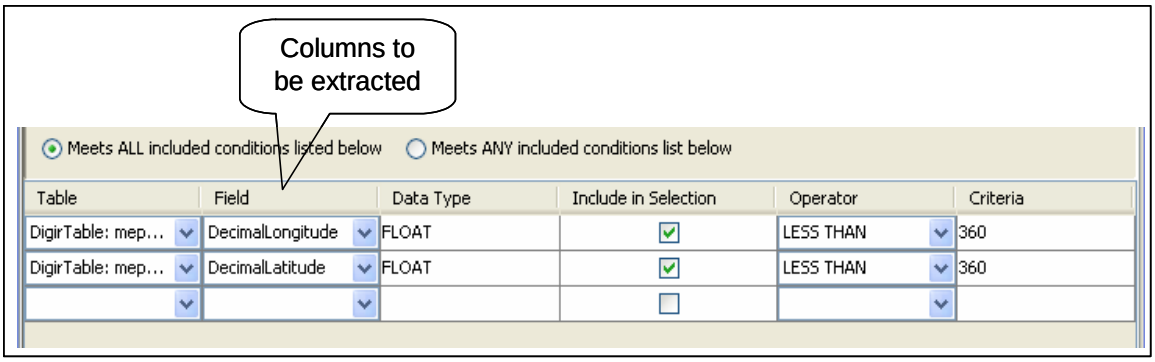


Figure 13: The SQL Query Builder is used to extract specific data from the dataset returned by the *DarwinCoreDataSource* actor (named *mephitis mephitis* in this example).

By default, the *DarwinCoreDataSource* actor returns data "As Field," using an output port for each field in the dataset. In this example, the entire dataset needs to be output as a table. Double-click the *mephitis mephitis* actor to see the actor's parameters. The output type of the actor is set to "As Table" via the actor's outputType parameter (*Figure 14*).

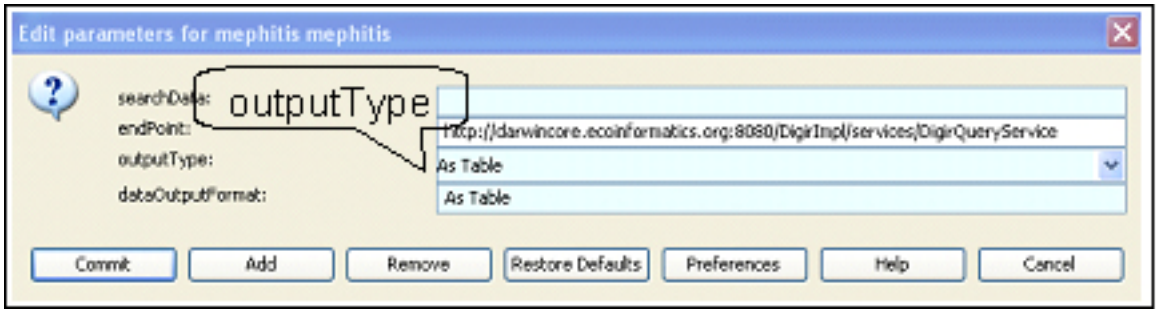


Figure 14: The output type of the *DarwinCoreDataSource* actor is set to "As Table" via the actor's outputType parameter.

3.1.4 Multiple species: Distributed museum collections

(See $Kepler/workflows/test/DarwinCore_Test.xml)

The DarwinCore_Test.xml workflow (*Figure 15*) uses the DiGIR protocol to access occurrence data for multiple species from museum collections. The workflow takes advantage of the *DarwinCoreDataSource* actor's `speciesName` port, via which the actor receives the names of each species to download. Once the occurrence data has been downloaded to the local cache, the workflow creates and saves a data file for each set of occurrence points.

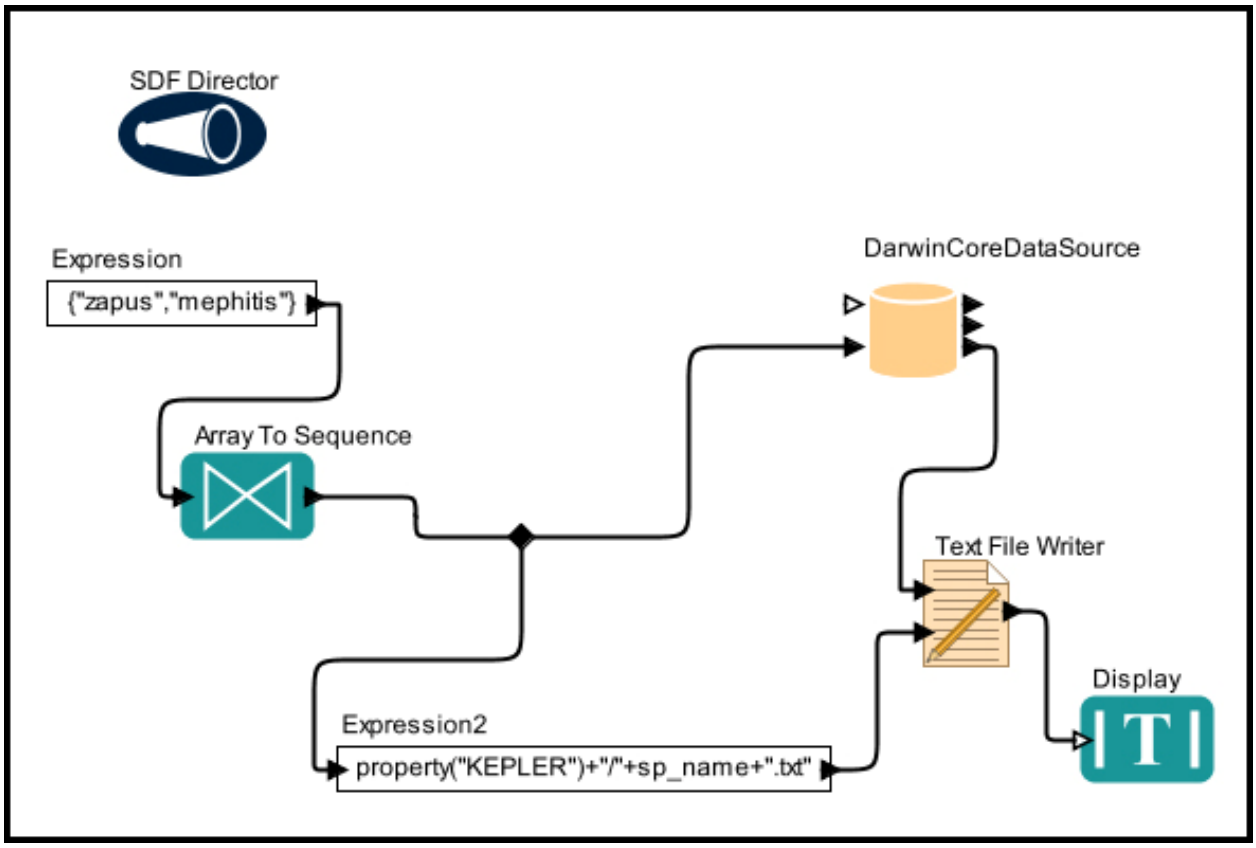


Figure 15: The DarwinCore_Test.xml workflow, which uses the DiGIR protocol to access occurrence data for multiple species, and creates a species data index file that can be used by the GARP_MultipleSpecies-V.xml workflow.

The workflow uses an *Expression* actor to identify the names of species. The list of species is specified as an array ({"zapus","mephitis"}). The *Array To Sequence* actor translates the array into a sequence of individual species names, and a Relation is used to branch the output names so that they can be fed to two downstream actors: an *Expression* actor (named *Expression2*), as well as the *DarwinCoreDataSource* actor. Note that the `arrayLength` parameter of the *Array To Sequence* actor must be set to match the number of species in the input array, in this case, 2.

The *Expression2* actor uses the Ptolemy expression language to create unique file names that will be applied to the downloaded occurrence datasets. The expression, `property("KEPLER")+"/"+sp_name+".txt"`, instructs Kepler to "build" a file path in the Kepler home directory. The `sp_name` in the expression references the value passed to the actor via the `sp_name` port, in this case, the species name. Depending on where Kepler has been installed, the generated file paths might look something like this: C:/kepler/zapus.txt and C:/kepler/mephitis.txt

The *DarwinCoreDataSource* actor is configured to retrieve the occurrence data for each species received via its `speciesName` port. The actor accesses the remote DiGIR specimen table and outputs a table containing only the relevant data: the latitude and longitude of each occurrence point, as well as the name of the species. Right-click the actor and select the Open Actor menu

item to view the Query Builder, which is used to filter the data so that only required fields are returned (*Figure 16*).

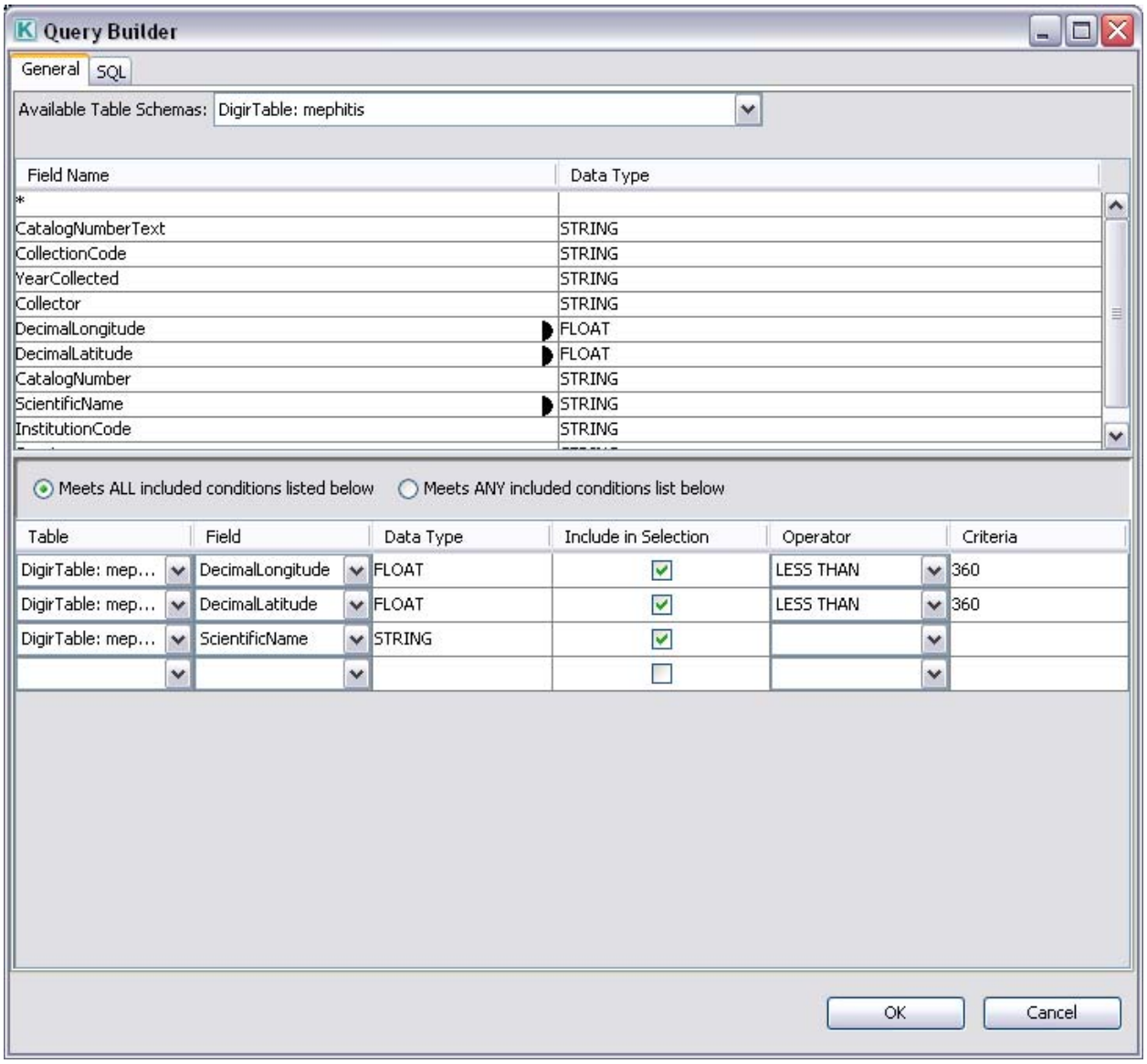


Figure 16: The Query Builder for the *DarwinCoreDataSource* actor, accessed by right-clicking the actor and selecting Open Actor from the drop-down menu.

The Query Builder lists the complete set of data fields returned by the actor, as well as the data type of each. Beneath the list of fields is a SQL query builder that can be used to filter the returned data. The Query Builder is used to select a "Table" (e.g., Digir Table:mephitis mephitis) and desired field (e.g., DecimalLatitude, DecimalLongitude, and ScientificName). The Query Builder's Operator and Criteria settings fields specify additional selection criteria. In this example, every record with a longitude and latitude is desired, and the selection criteria are designed to select any record with valid values for both (i.e., any record with latitude or longitude less than 360 degrees). The only records that will be excluded are those without entries, or those that have invalid entries, for longitude or latitude.

Note: Valid latitudes should be in the range of -90 to +90, but missing data are often coded as a very large number, so the 360 degree value works to screen invalid records in most cases.

The *DarwinCoreDataSource* actor outputs the occurrence data to the *Text File Writer* actor, which reads the string and writes it to a file. The *Text File Writer* actor saves the data to the path specified via its `fileToWrite` port.

Note: Because retrieving data from DiGIR collections takes several minutes per species, the first run of the DarwinCore_Test.xml workflow may take fifteen to twenty minutes. Subsequent workflow executions are much faster because the data is cached locally during the first execution.

3.2. Climate layers

Global climate data is provided by the Intergovernmental Panel on Climate Change (IPCC) Data Distribution Centre, both for historical climate conditions (1961-1990) and for predictions of future climate conditions (2020, 2050, 2080) based on multiple global circulation models (GCMs) (See http://www.ipcc.ch/). Data are available for six GCMs run independently by different climate modeling centers around the world, using different assumptions and change scenarios, resulting in 21 GCM/scenario combinations.

The Kepler EarthGrid contains climate data copied from the IPCC website and stored in the Storage Resource Broker at San Diego Supercomputer Center. These data can be easily searched and accessed by Kepler via the application's Data tab. The version of the data currently available on the Kepler EarthGrid is in the process of being replaced by a newer version from the IPCC website. The new version will be obtained and placed on the EarthGrid soon.

IPCC data consists of predictions for ten variables: cloud cover, diurnal temperature range, ground frost frequency, maximum temperature, minimum temperature, precipitation, radiance, vapor pressure, wet day frequency, and wind speed. Not all variables are available for every GCM. Data are stored in a longitude/latitude based grid, organized by month.

To use historical and/or future IPCC data in the ENM workflow, the data must first be transformed into a consistent format: ASCII Grid files (*.asc). Each grid file or "layer" must have the same geographic extent and resolution.

Several workflows have been designed to transform IPCC data for use with ENM workflows. Each of these workflows is discussed in more details in the following sections. Because historical and future datasets differ, the process is somewhat different for each type of data.

3.2.1 Working with historical IPCC datasets

(see "/$KEPLER/workflows/eco/IPCC_Base_Layers.xml")

The IPCC_Base_Layers.xml workflow (*Figure 17*) transforms historical climate data into a format that can be used by ENM workflows. The workflow reads 10 climate change datasets (cloud cover, diurnal temperature range, ground frost frequency, etc) from the Kepler EarthGrid, and transforms and stores the data as ASCII Grid files (*.asc). The minimum, maximum, or average values for each of the ten climate variables are summarized seasonally and/or annually and are stored in the location specified by the workflow's `OutputDir` parameter (e.g., C:/IPCCLayers/).

Note: Each version of climate data released by the IPCC must be converted only once, as converted data are saved and can be reused for every ENM run.

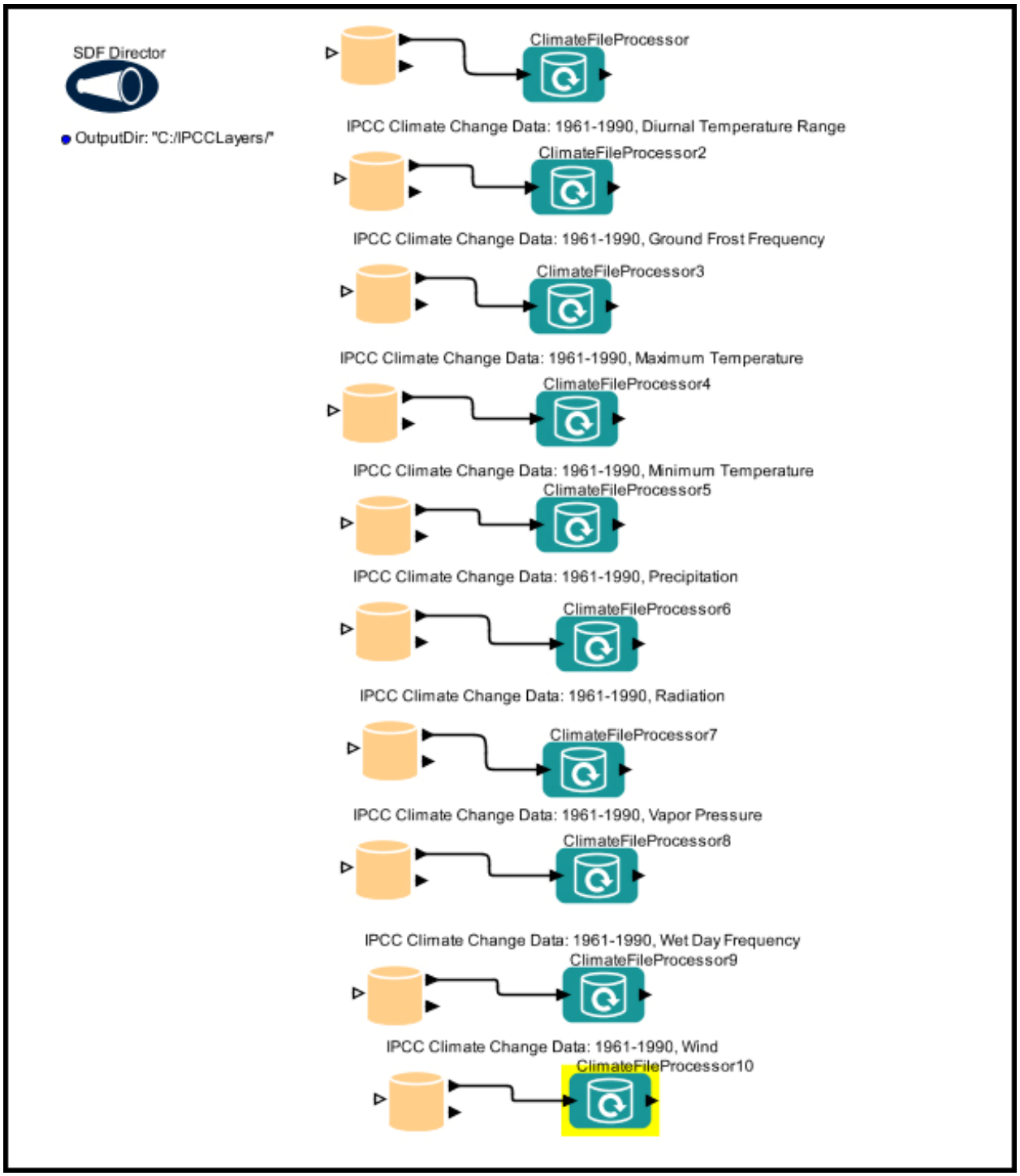


Figure 17: The IPCC_Base_Layers.xml workflow, which is used to transform and save historical climate data into a format that can be used by ENM workflows.

The data in the IPCC_Base_Layers.xml workflow are retrieved via a search from the Data tab. Dragging and dropping a returned dataset to the Workflow canvas instantiates a *EML2DataSource* actor, which functions much like the *DarwinCoreDataSource* actor discussed

in Sections 3.1.3 and 3.1.4. Retrieved data are stored in the Kepler cache, and the cache file name is passed to the *ClimateFileProcessor* actor.

Note: To learn more about the data, right-click the *EML2DataSource* actor and select "Get Metadata". The metadata contains detailed information about the extent, resolution, and content of the dataset.

The *ClimateFileProcessor* actor summarizes the data seasonally and/or annually (*Figure 18*). Data are output in ASCII Grid format (*.asc) and are stored in the location specified by the *ClimateFileProcessor* actor's `baseOutputFileName` parameter. The value of the `baseOutputFileName` parameter references the directory specified by the workflow's `OutputDir` parameter ("C://IPCCLayers/") and specifies an individual file name (e.g., ccov6109) to use for the layer.

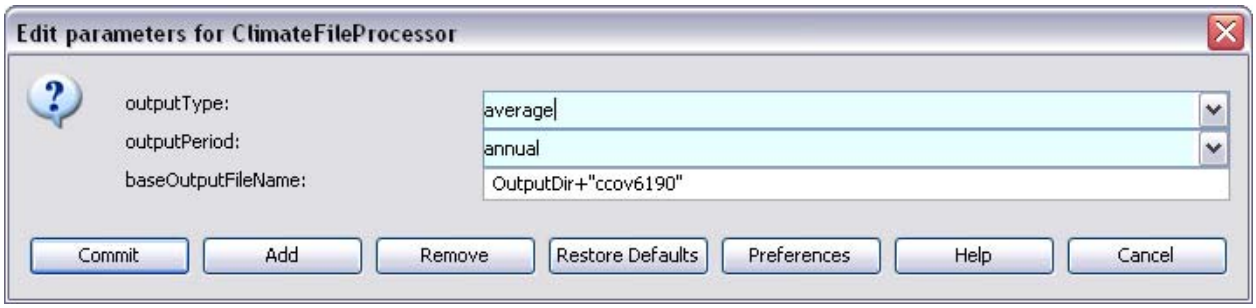


Figure 18: *ClimateFileProcessor* actor parameters. outputPeriod specifies the time period over which the data will be summarized (annual, winter, spring, summer, or fall) and the outputType parameter specifies the summarization method (average, maximum, minimum).

3.2.2 Working with IPCC future climate change datasets

(See "$KEPLER/workflows/test/ IPCC_Change_*.xml")

Because IPCC predicted future climate change datasets differ in format from historical datasets, additional transformation must occur on the data before they can be compared to historical climate data or used in the same ENM workflows.

Future climate change data, unlike historical data, is recorded as a “delta” value, or change from the historical value. In addition, future predictions (which vary in resolution by GCM) have a coarser resolution than historical predictions (which have a 0.5 degree resolution). To make future and historical data compatible, the following conversions must take place:

A. The future data must be converted to ASCII grid format and summarized seasonally and/or annually.
B. The future data must be converted to the same format as the historical data; the resolution, measurement standard (and in some circumstances, the extent) of the future data must be modified to match that of the historical data
C. The "delta" temperature values must be converted to temperature values, accomplished by adding the predicted temperature change to the historical value.

The IPCC_Change_Summary.xml workflow (*Figure 19*) translates six future climate change datasets into ASCII grid files (*.asc).

Note: Each version of climate data released by the IPCC must be converted only once, as converted data are saved and can be reused for every ENM run.

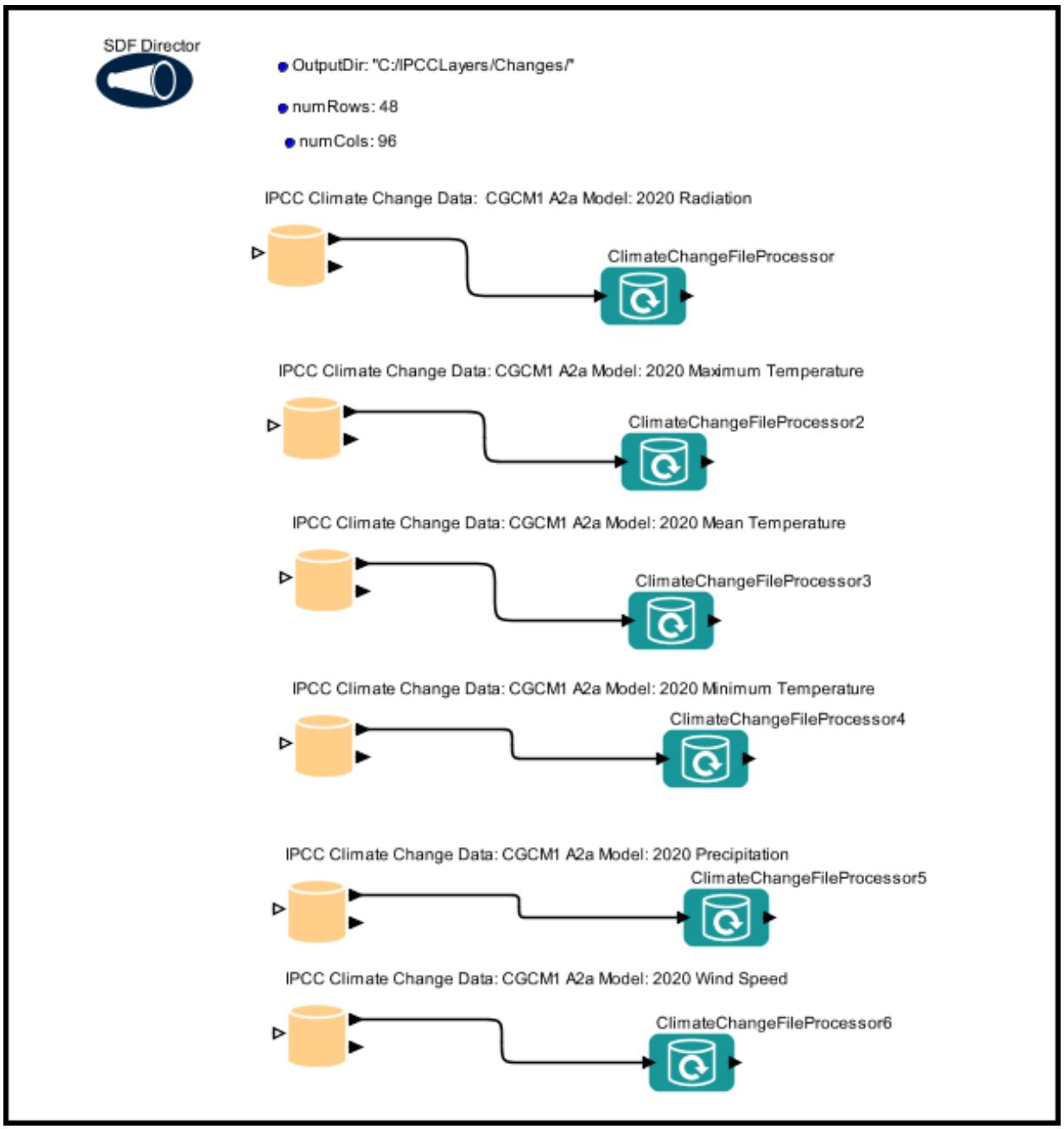


Figure 19: The IPCC_Change_Summary.xml workflow, which saves six IPCC future climate change datasets in *.asc format.

The *ClimateChangeFileProcessor* actor functions much like the *ClimateFileProcessor* actor discussed in Section 3.2.1. Users specify the time period for summarization (annual, winter, spring, summer, fall) and the type of summarization (average, minimum, maximum). The *ClimateChangeFileProcessor* actor's `rowsParameter` and `colsParameter` (*Figure 20*) reference workflow parameters (`numRows` and `numCols`) to specify the number of rows and columns in the output *.asc grid file (48 and 96, respectively).

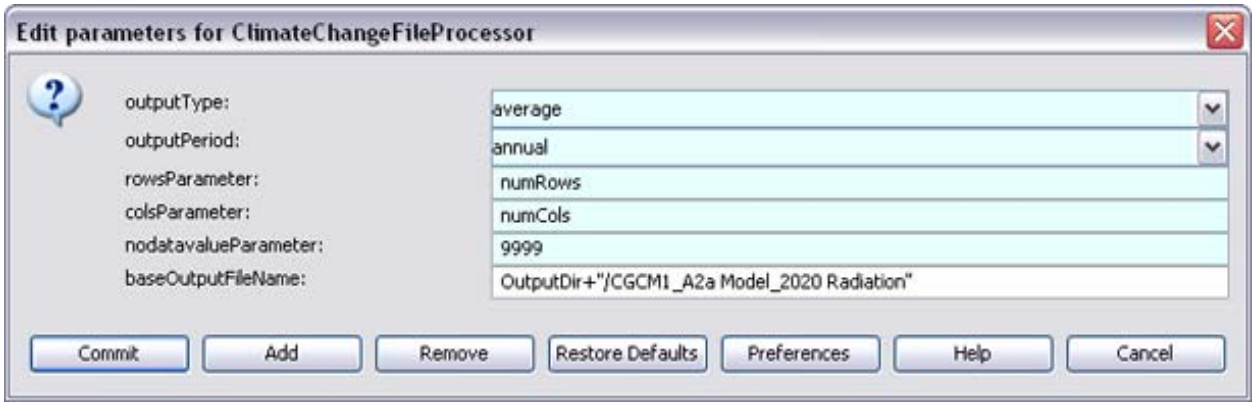


Figure 20: The *ClimateChangeFileProcessor* actor's parameters. The `rowsParameter` and `colsParameter` reference workflow parameters defined on the Workflow canvas.

Although the IPCC_Change_Summary.xml workflow converts the future climate data to *.asc files, the data are not yet compatible with the historical datasets: the resolution, extent, and measurement standard must all be transformed to match the historical standards before the data can be used in ENM workflows. The "IPPC_Change_MaxTemp.xml" workflow (*Figure 21)* is an example of a workflow that can be used to adjust one set of future climate data (maximum temperature data) to match the historical datasets.

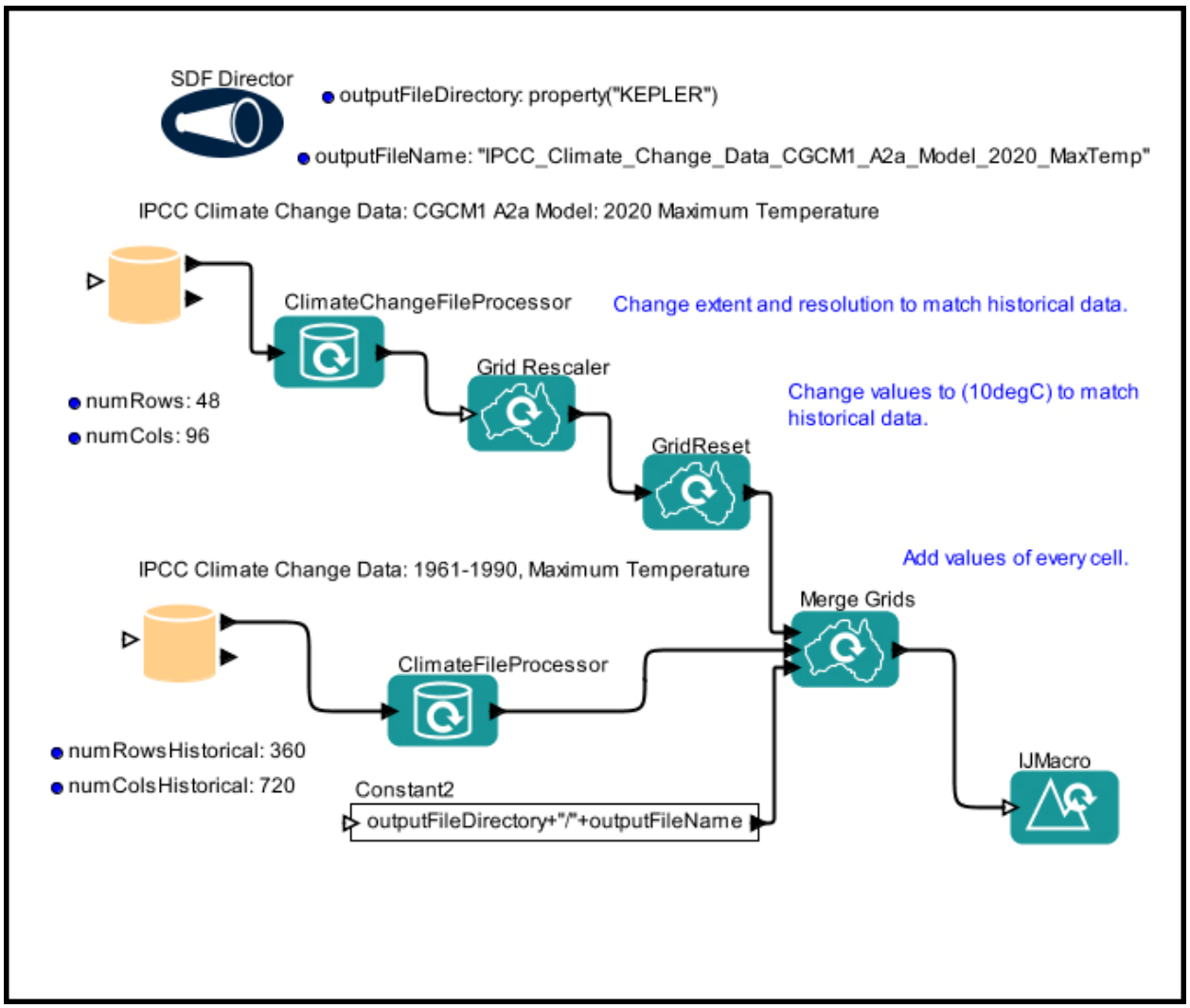


Figure 21: The IPPC_Change_MaxTemp.xml workflow rescales an ASCII grid file and adjusts the resolution and extent to match that of historical datasets. Additional workflows (IPPC_Change_MinTemp.xml, IPPC_Change_Precip.xml, IPCC_Change_Wind.xml, etc) are available for working with other predicted climate change variables.

The workflow reads historical and future datasets from the Kepler EarthGrid and uses the *ClimateFileProcessor* and *ClimateChangeFileProcessor* actors to convert the data to *.asc grid files. Once the data have been converted, they are output to a *Grid Rescaler* actor.

The *Grid Rescaler* actor uses an interpolation algorithm (either "Nearest Neighbor" or "Inverse Distance") to increase the resolution of the input file. The actor can also use its `numrows` and `numcols` parameters to reduce or "clip" the extent of the grid, which is necessary if the historical climate data used in the workflow are reduced in extent from the original data (*Figure 22*). The values of these parameters reference the `numRowsHistorical` and `numColsHistorical` parameters defined on the Workflow canvas (360 and 720, respectively). The actor's `yllcorner` and `xllcorner` parameters specify the latitude and longitude of the lower left corner for the transformed grid, and the `cellsize` parameter

specifies the grid resolution, in this case .5 degrees. The *Grid Rescaler* saves the transformed file to a file specified by the `outputFileName` parameter.

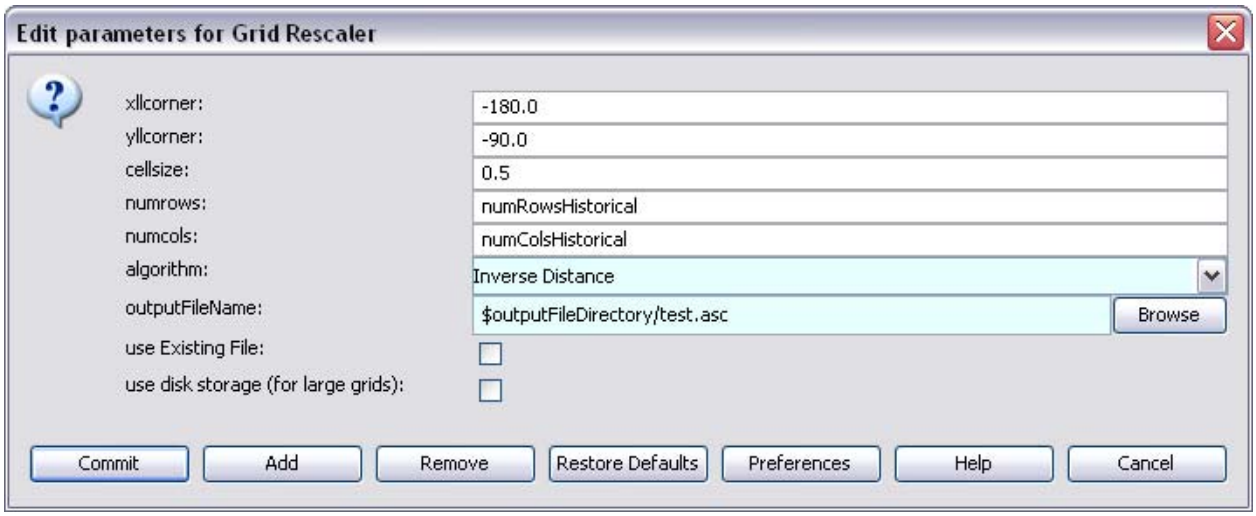


Figure 22: The *Grid Rescaler* parameters, which are used to transform the extent and resolution of future climate prediction data so that they match that of historical data.

The *Grid Rescaler* actor outputs the transformed data to the *GridReset* actor, which scales the "delta" temperature values to match the measurement standard used in the historical data. The actor multiplies each cell value by a specified multiplication factor (10 degrees Celsius).

Note: The measurement standard for each dataset is described in the dataset Metadata. Right-click the *IPCC Climate Change Data: CGCM1 A2a Model: 2020 Minimum Temperature* actor and select Get Metadata to view this information.

The *GridReset* actor saves the rescaled data and outputs the file name to the *MergeGrids* actor, which adds the "delta" temperature to the historical value for each grid cell. An *ImageJ* actor displays the result, which is now ready for use in niche modeling (*Figure 23*)

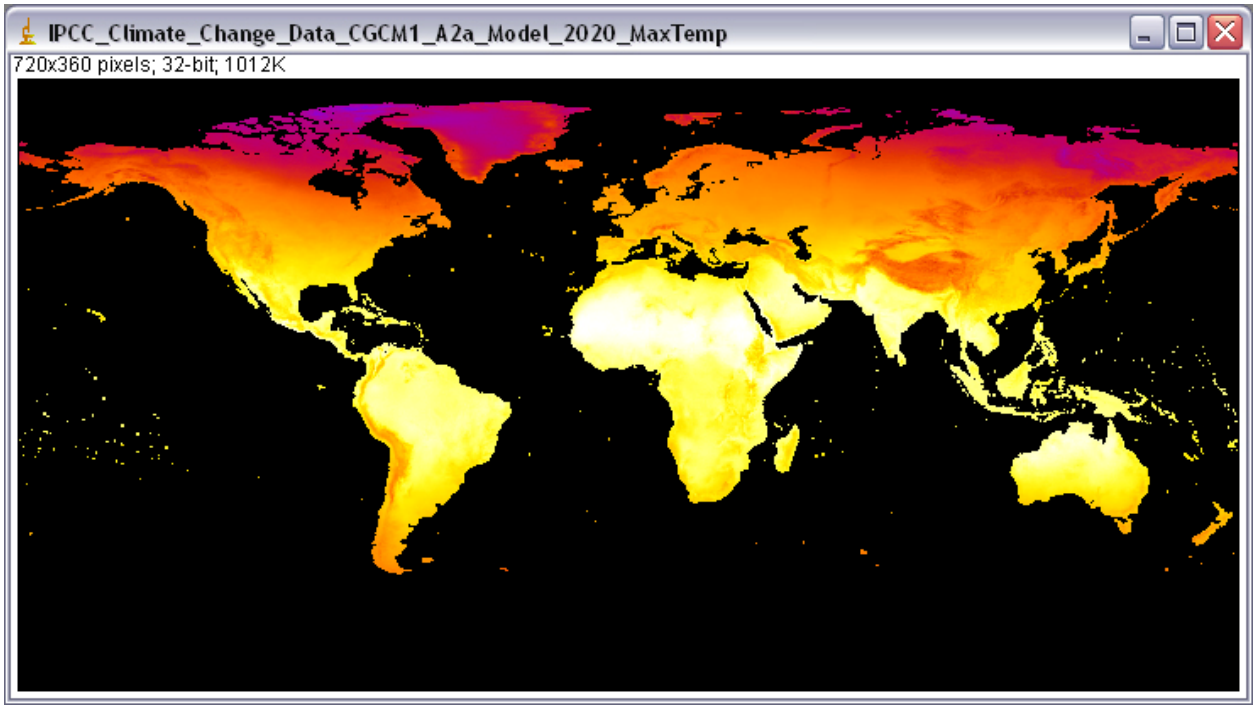


Figure 23: Output generated by the IPPC_Change_MaxTemp.xml workflow. The map displays future maximum climate data in a format that is compatible with historical climate data.

3.3. Topographic layers

See ($KEPLER/workflows/eco/GDAL-h1K_NS.xml)

Topographical data are provided by HYDRO1k, a geographic database providing comprehensive and consistent global coverage of topographically derived datasets. Developed at the U.S. Geological Survey's (USGS) EROS Data Center, HYDRO1k provides a standard suite of 1 km resolution georeferenced datasets by continent (For more information, see http://lpdaac.usgs.gov/gtopo30/hydro/readme.asp).

Available HYDRO1k datasets include:

Digital Elevation Model (DEM)	The Digital Elevation Model forms the basis of all the additional HYDRO1k datasets.
Slope	The Slope dataset describes the maximum change in the elevations between each cell and its eight neighbors.
Aspect	The Aspect dataset describes the direction of maximum rate of change in the elevations between each cell and its eight neighbors.
Flow Accumulation	Developed from the flow direction layer, the Flow Accumulation dataset defines the number of cells that flow into each downslope cell.
Compound Topographic	The Compound Topographic Index (CTI), commonly referred to as

Index (CTI)	the Wetness Index, is a function of the upstream contributing area and the slope of the landscape.

Table 1: Available HYDRO1k datasets.

The HYDRO1k datasets are stored in zipped, Band Interleaved By Line (*.bil) format. Each zipped file contains raw data and several text files containing metadata.

Before HYDRO1k datasets can be used for niche modeling, they must be transformed to match the spatial characteristics of the climate data. The GDAL-h1K_NS.xml workflow (*Figure 24*) is designed to do just that: it downloads the HYDRO1k data for North and South America from a remote server (or, if the data have already been downloaded, accesses them from a local cache), extracts the data, converts the HYDRO1k map projection (which uses a Lambert Azimuthal Equal Area coordinate system) to a format that uses a latitude/longitude system, converts the raster format (GeoTiff) to one used by GARP (ASC raster grid), rescales the raster file, and combines the two "cleaned up" files into a single map.

NOTE: The GDAL-h1K_NS.xml workflow uses digital elevation models for North and South America, but the workflow can easily be used with other topographic files; simply replace the DEM data source actors with the desired topographic data source, and update the corresponding mask files (discussed later).

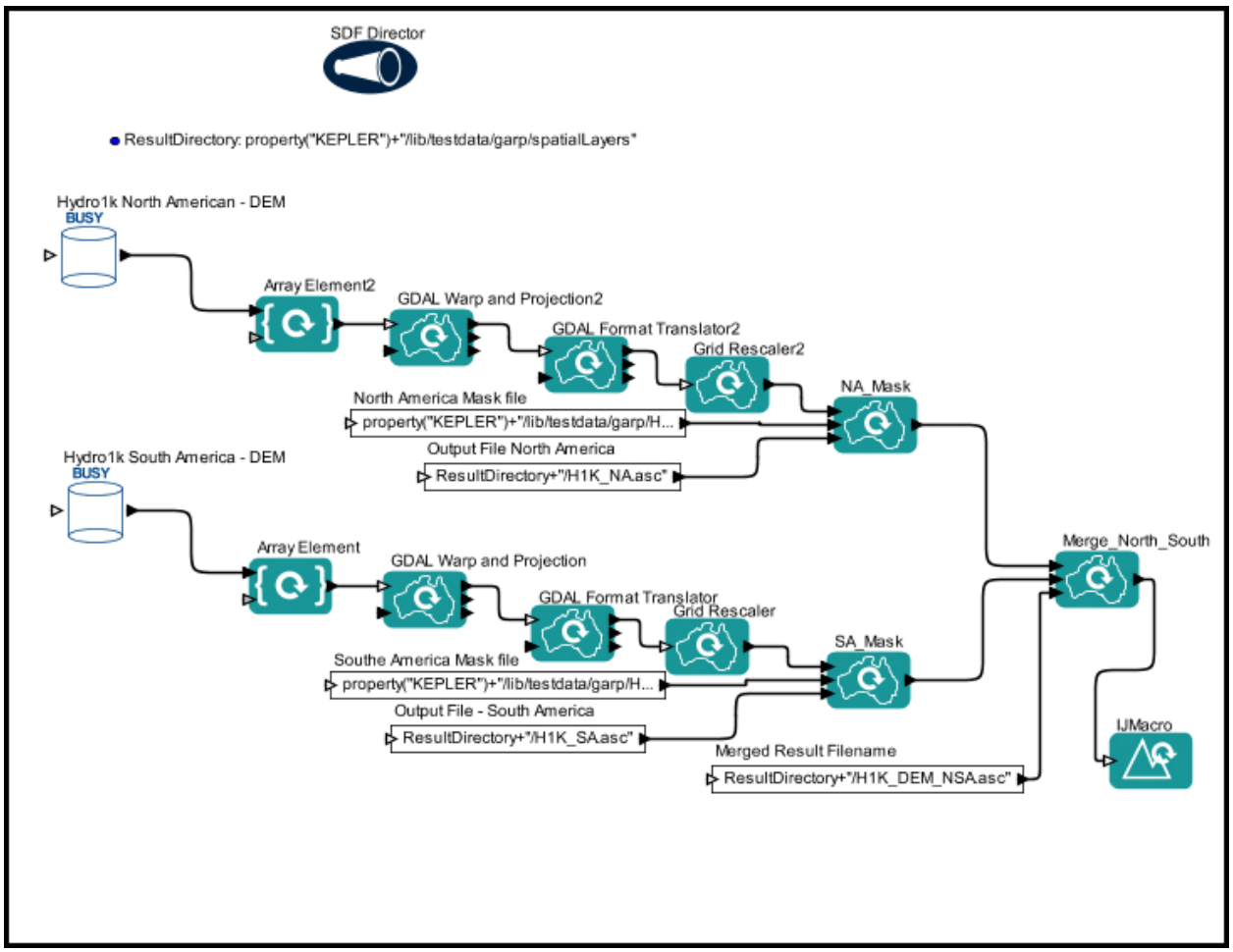


Figure 24: The GDAL-h1K_NS.xml workflow. The first time the workflow is opened, the data source actors (Hydro1k North America DEM and Hydro1k South America DEM) will show a "Busy" status as they download data from a remote server. The initial download may take as long as 30 minutes. Once data is stored in the local cache, the data are more immediately available. Because of the high resolution of the data, this workflow requires 30-45 minutes to execute once the data are downloaded.

In the GDAL-h1K_NS.xml workflow, HYDRO1k data are accessed via the Kepler EarthGrid. To locate topographical data, search for "Hydro1k" under the Data tab, and drag and drop a dataset on to the Workflow canvas. The data source actor will download the zipped data and save the data to the Kepler cache. Note that the initial download may take as long as 30 minutes with a reasonably fast PC. Once the data are saved to the cache, they become more immediately available.

Once the HYDRO1k data have been downloaded to the cache, they must be extracted from the zip file. The `Data Output Format` parameter (*Figure 25*) instructs the *Hydro1k North America DEM* and *Hydro1k South America DEM* actors to unzip the downloaded data into the Kepler cache and output the file name of the dataset (actually an array of file names: the file name of the raw data as well as the file names of the associated meta data files).

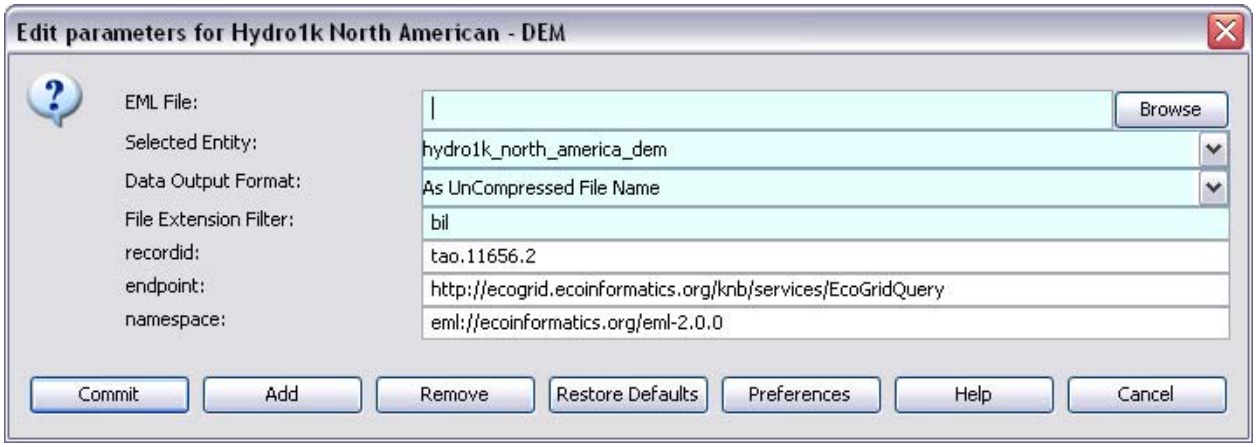


Figure 25: The parameters for the *Hydro1k North American –DEM* actor. Selecting "As UnCompressed File Name" as the value of the `Data Output Format` parameter instructs the actor to unzip the dataset into the Kepler cache.

The *ArrayElement* actor reads the array of file names output by the data source actors and extracts the first element, which is the name of the raw dataset. The file name of the dataset is then passed to downstream actors for further data transformations.

The *GDAL Warp and Projection* and *GDAL Format Translator* actors convert the data to the proper spatial format. The actors both use the Geospatial Data Abstraction Library (GDAL), an open source software package designed to read, write, and manipulate a wide variety of Geographical Information System (GIS) raster grid files. Details of GDAL are documented at http://www.gdal.org/index.html.

The *GDAL Warp and Projection* actor converts the data's coordinate system (Lambert Azimuthal Equal Area) to a longitude/latitude projection. Refer to http://www.gdal.org/gdal_utilities.html#gdalwarp for more information. After the projection has been converted, the actor outputs the result as a GeoTiff file. The *GDAL Format Translator* actor transforms the GeoTiff file into a byte format (ASC grid) used by ENM workflows. Refer to http://www.gdal.org/gdal_utilities.html for further information.

Once the data have been converted to the proper file and projection format, the *Grid Rescaler* actor ensures that the files have the proper resolution and extent. *Grid Rescaler* parameters are used to set the x and y values for the lower left corner of the output grid, the cell size, and the number of desired rows and columns (*Figure 26*). Either the "Nearest neighbor" or "Inverse distance" weighted algorithms can be used to calculate output cell values. If the "Use Existing File" checkbox is selected, the actor will check to see if a file with the output file name already exists. If so, then the actor skips all actions except for returning the existing file name (i.e., the actor does not "re-translate" the source data). Selecting the "use Existing File" parameter can help avoid lengthy rescaling calculations that have already been completed in prior runs. If the checkbox is not selected, any existing output file with the same name will simply be overwritten.

The *Grid Rescaler* actor outputs the file names of the transformed grid files. If the `outputFileName` parameter is empty, then the output file name is just the input file name plus the suffix ".outn" (where n is an index). If the `outputFileName` parameter is a

directory, all output is put in that directory; otherwise output is placed in the same directory as the input file(s). In the example workflow, the `outputFileName` parameter for the *Grid Rescaler2* actor is set to `$ResultDirectory/NewTest_NA.asc,` which instructs the actor to save the data to a file named NewTest_NA.asc, in the Kepler home directory under /lib/testdata/garp/spatialLayer/.

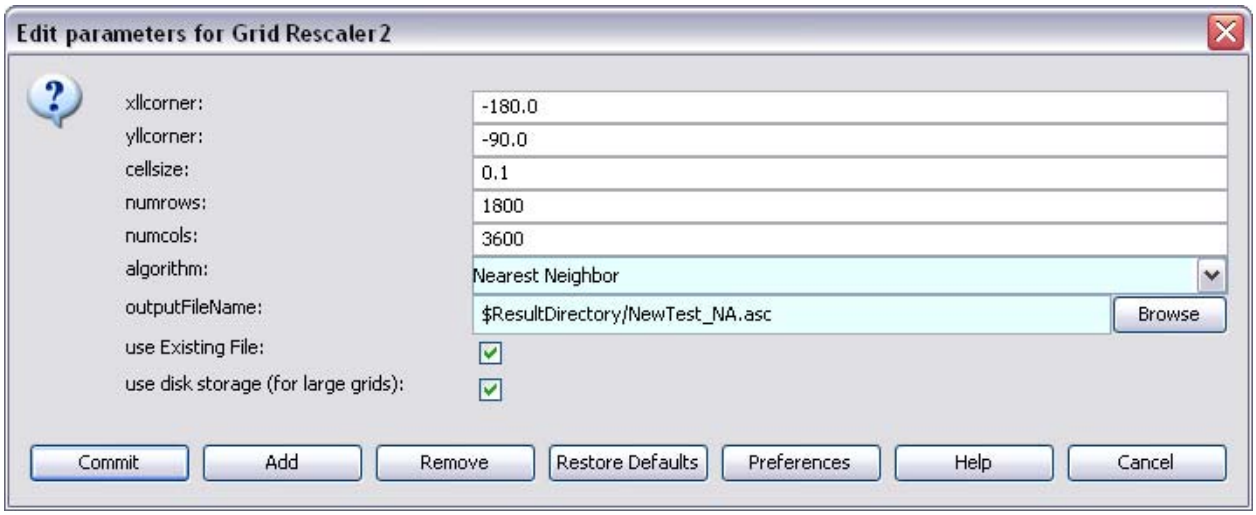


Figure 26: Parameters of the *Grid Rescaler2* actor. Note that the "use Existing File" parameter has been selected, instructing the actor to return the file name of an existing output file if one exists.

Once the *Grid Rescaler* actors have transformed the ASC grid files, non-continental areas of the files must be "masked." The workflow uses two *Merge Grid* actors (*NA_Mask* and *SA_Mask*) to mask the maps. The *Merge Grid* actors receive the transformed HYDRO1 data as well as the name of a mask file. In addition, the actor receives a file name to use for the output masked file (e.g., "ResultDirectory+"/H1K_NA.asc"). Masked areas (e.g., oceans) will be assigned a value of "NO_DATA".

Lastly, the processed topographical data for the North and South American continents are mosaicked into a single grid with the *Merge_North_South* actor, and the results are displayed with an *ImageJ* actor (*Figure 27*). The transformed topographical data layers are now ready to be used with the ENM workflow.

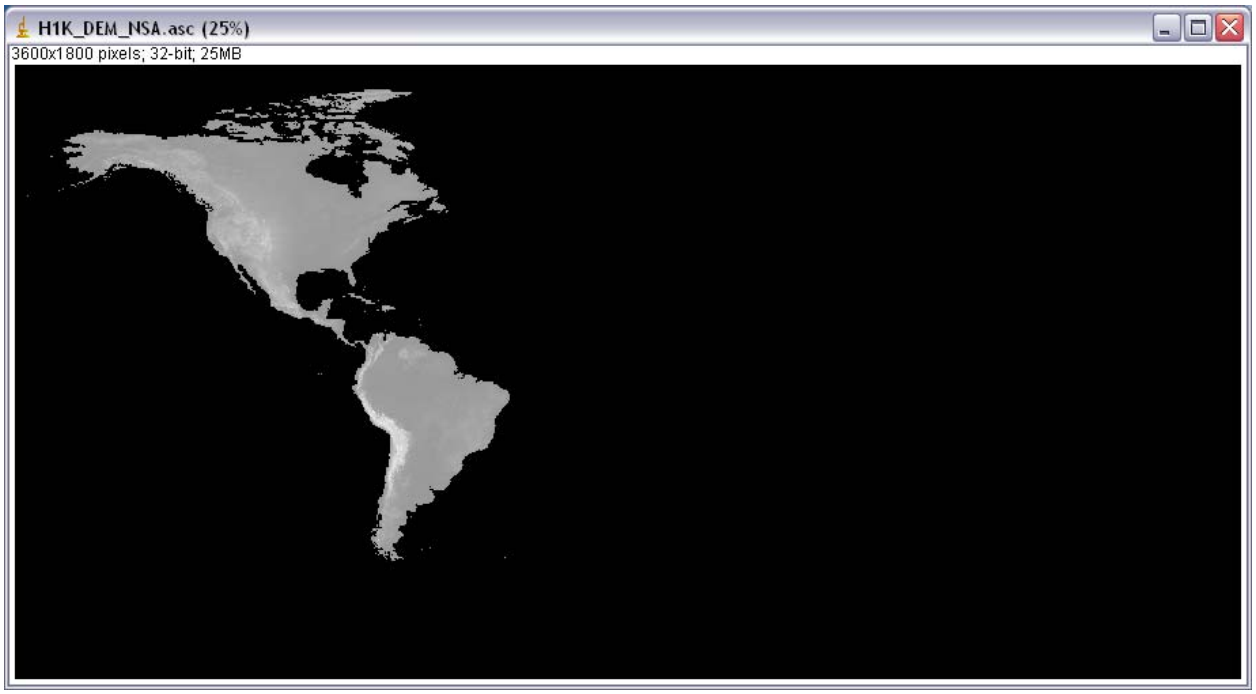


Figure 27: A topographical map of North and South America, output by the *GDAL-h1K_NS.xml* workflow.

3.4. Ready-to-use climate and topographic layers on the EarthGrid

(See $KEPLER/workflows/eco/BaselineLayers.xml)

The Kepler EarthGrid contains a base set of ready-to-use environmental layers created from the historical IPCC climate data (10 layers) and HYDRO1k data (5 layers). The prepared data are global in extent and have a cell resolution of 0.1 degrees. All of the data layers have been preprocessed to be in the *.asc format and all are scaled to the same size.

To retrieve the prepared data, search for "IPCC" on the EarthGrid via the Kepler Data tab. The dataset (named “Baseline Layers for ENM/GARP Calculations (IPCC/Hydro1K)") will appear at the bottom of the returned search results.

The prepared dataset is in a zipped file that can be downloaded and unzipped with the BaselineLayers.xml workflow (*Figure 28*). The BaselineLayers.xml workflow outputs the file names of the individual data files, which are stored temporarily in the Kepler cache (~/.kepler/unzip/). Cached files can later be moved to a permanent location on the local host.

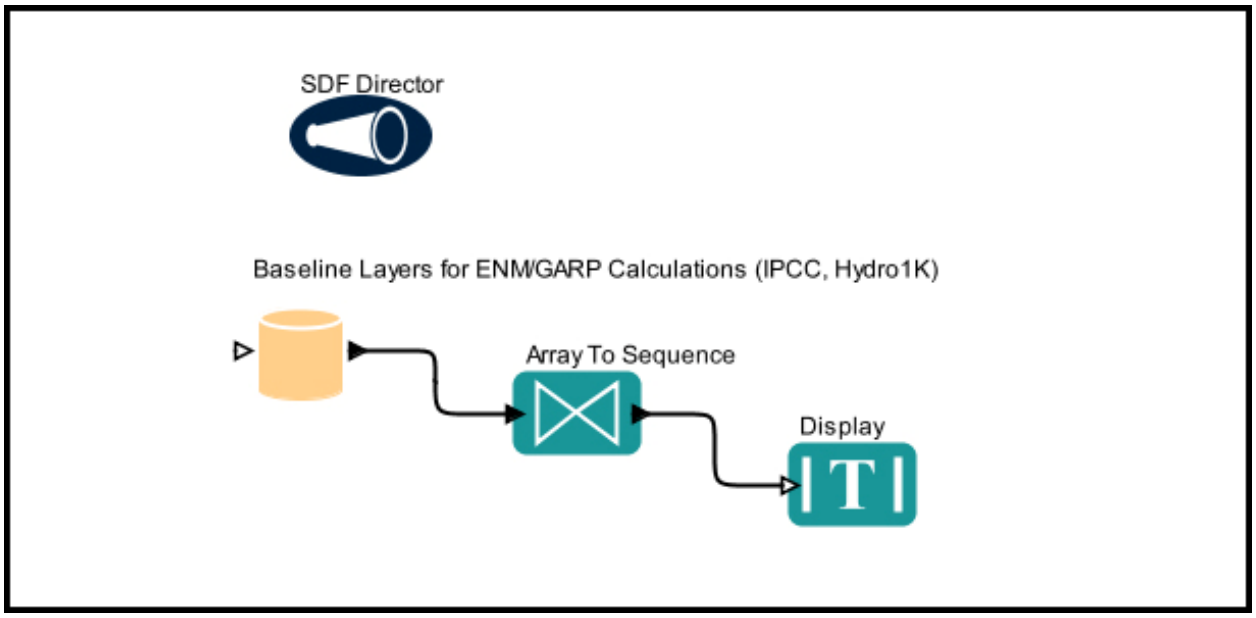


Figure 28: The BaselineLayers.xml workflow, which is used to open and display a base set of ready-to-use environmental layers created from the historical IPCC climate (10 layers) and Hydro1K (5 layers) data.

Once the data have been downloaded to the cache, they must be extracted from the zip file. The *Baseline Layers for ENM/GARP Calculations (IPCC, Hydro1K)* actor's `Data Output Format` parameter instructs the actor to unzip the downloaded data into the Kepler cache and output an array of the contained .asc datasets. The *Array To Sequence* actor then transform the array of file names into a sequence of file names that are displayed by the *Display* actor.

Before the data can be used in the ENM workflow, they must be collected into a single dataset that can be sampled. The $KEPLER/workflows/eco/BaselineLayers_2.xml can be used to unzip and translate the layers into a format (*.raw) that is ready for use with the ENM workflow. See Section 3.4.1 for more information.

3.4.1 Step-by-step: Integrating ready-to-use layers with the ENM workflow

Use the following steps to incorporate a set of ready-to-use environmental layers into the single species' distribution workflow (GARP_SingleSpecies_BestRuleSet-IV.xml). The layers are stored on the Kepler EarthGrid and have been pre-processed to have a common resolution and extent.

1. Open the BaselineLayers_2.xml workflow ($KEPLER/workflows/eco/BaselineLayers_2.xml)
2. Run the workflow. The workflow unzips the environmental layers, translates them to the *.raw format used by GARP, and saves the layers as well as a summary file ("world.dxl") to the kepler cache in the '.kepler/unzip/' directory. The "world.dxl" file contains an index of all of the environmental layer files. Open it with a text editor to see the complete list of files. Note that the index includes a mask file. This mask file "masks" all area outside of the Western hemisphere. To include the entire globe, create a new mask file (see steps 3 and 4)

3. Delete the mask.raw file.
4. Copy any of the environmental layers unzipped by the BaselineLayers_2.xml workflow (e.g., "ccov6190_average_ann.raw") and name the copied file "mask.raw". This new mask will include both the Western and Eastern hemispheres. Note: the layer file works as a mask because GARP doesn't look at the "real data" when considering a mask file—just the area where data exists.
5. Copy the "world.dxl" file and the unzipped environmental layers and mask file into the data directory used by the workflow (by default, `property("KEPLER")+"/lib/testdata/garp"`). Note that `property("KEPLER")` references the directory in which the Kepler application is installed.
6. Make a copy of the "world.dxl" file and name it "world1.dxl". The copied file will be used by the workflow to create a mask around the occurrence points. For more information, see Section 3.6.
7. Open the "world1.dxl" file with a text editor and change the following line:

```
<EnvLayer Type="Mask" Id="mask" Title="mask"
MatrixType="RawByteMatrixInDisk" MatrixFileName="mask.raw"
ValueUnits="" MapUnits="" CoordSys="" LayerType="" Rows="1800"
Columns="3600" XMin="-180.0" XMax="180.0" YMin="-90.0" YMax="90.0"
CellSize="0.1" MinValue="-74.0" MaxValue="6562.0"
ValueCoef="26.229249011857707" />

to:

<EnvLayer Type="Mask" Id="mask1" Title="mask1"
MatrixType="RawByteMatrixInDisk" MatrixFileName="mask1.raw"
ValueUnits="" MapUnits="" CoordSys="" LayerType="" Rows="1800"
Columns="3600" XMin="-180.0" XMax="180.0" YMin="-90.0" YMax="90.0"
CellSize="0.1" MinValue="-74.0" MaxValue="6562.0"
ValueCoef="26.229249011857707" />
```

8. Open the GARP_SingleSpecies_BestRuleSet-IV.xml workflow.
9. Delete the *Future_Climate_Models* sub-workflow. Note: prepared climate layers include only historical IPCC data. To use the *Future_Climate_models* sub-workflow, you must first prepare future climate scenario layers. See Section 3.2.2 for more information.
10. Right-click the *Calculate Best Rulesets* actor and select "Open Actor."
11. Change the value of the `cellSize` parameter to `.1`; change the value of the `numXCells` parameter to `3600`; change the value of the `numYCells` parameter to `1800` (the new extent and resolution of the environmental layer set).
12. Change the value of the *II-EnvLayerSet* actor to `DataDirectory+"/world1.dxl"`
13. Change the value of the *II-EnvLayerSet2* actor to `DataDirectory+"/world.dxl"`
14. Right-click the *Create Best ASC Maps* actor and select "Open Actor." Change the value of the `EnvLayerSet` port-parameter to `"world.dxl"`
15. Run the workflow. Note: because the extent and resolution of the files is so high, the workflow may take 30-40 minutes to execute. *Figure 29* shows a global projection created with the Mephitis mephitis data.

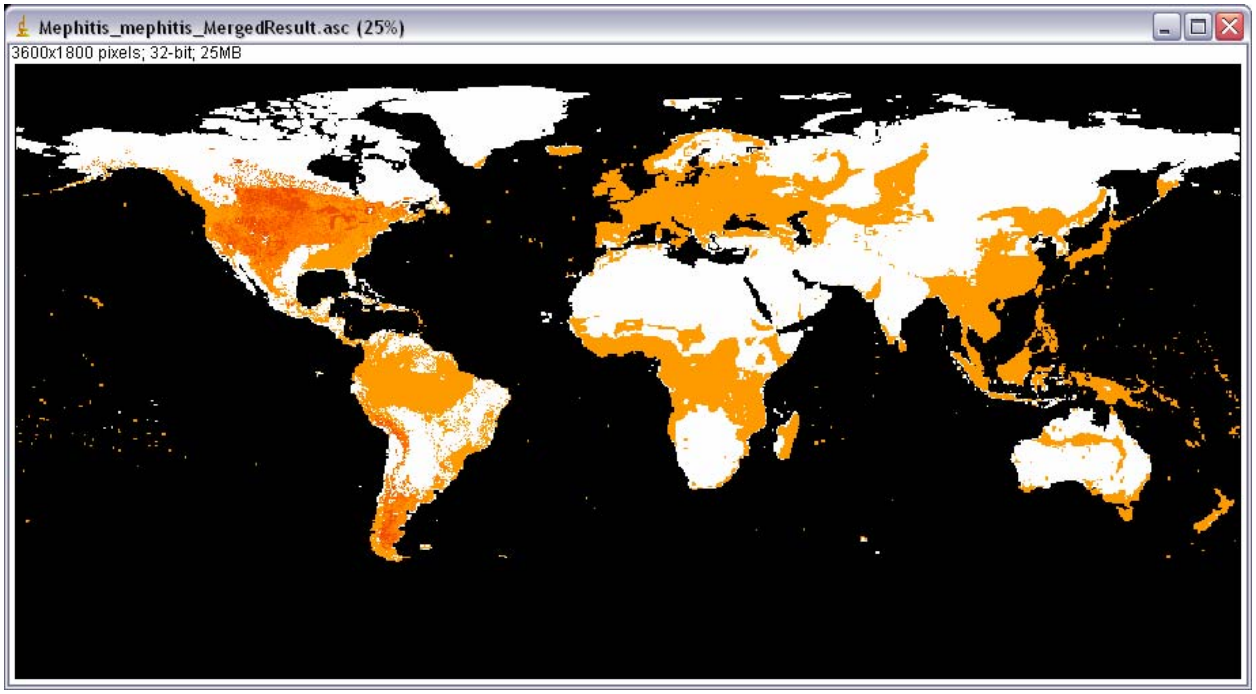


Figure 29: A global prediction of suitable habitat using Mephitis mephitis occurrence data. Brighter colors indicate a higher probability of presence; white indicates area where no prediction can be generated based on the data.

3.5. Collecting layers to be sampled

(See $KEPLER/workflows/test/ASC2RAWTest.xml and $KEPLER/workflows/eco/BaselineLayers_2.xml)

All of the environmental layers--climatic, topographic, plus any others that may be desired – must be collected into a set that can be sampled at each species occurrence point. In order to be collected and used together, the layers must each have a comparable projection, extent, and resolution. This data standardization is accomplished with the workflows discussed in the previous sections.

Precisely how the layers get collected depends on the input requirements of the modeling algorithm that is used in the workflow. The ENM prototype uses the GARP algorithm, which requires that each ASCII grid (*.asc) be converted to a custom binary format (*.raw), and that information about the layer set be summarized in a text file (*.dxl).

The ASC2RAWTest.xml workflow (*Figure 30*) translates environmental layers (.asc grid files) to binary files appropriately formatted for use with GARP.

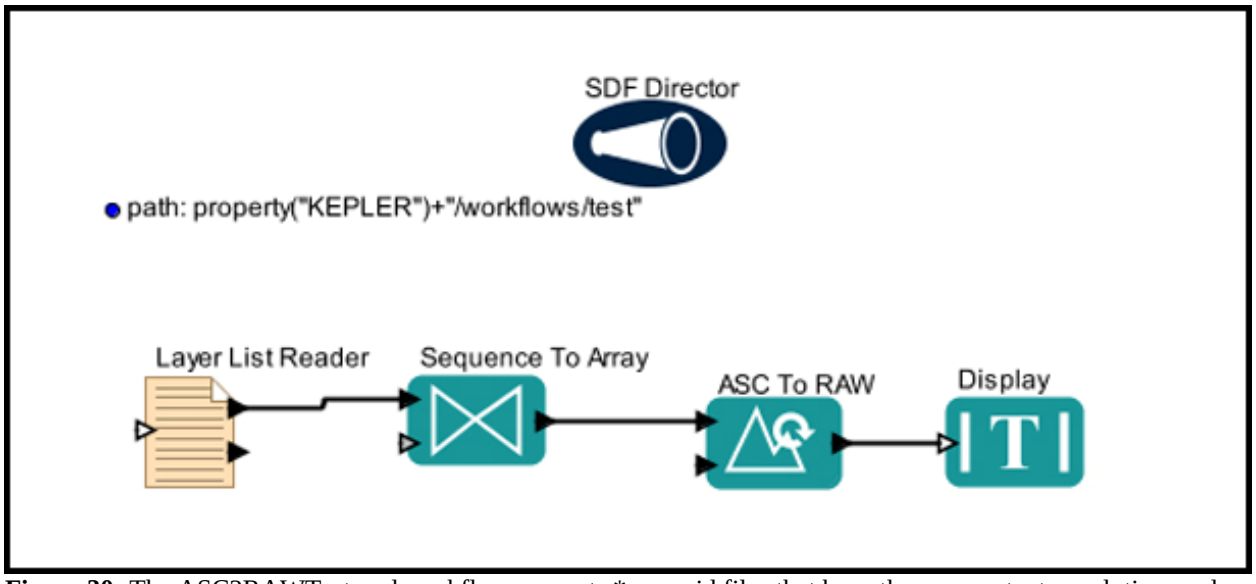


Figure 30: The ASC2RAWTest.xml workflow converts *.asc grid files that have the same extent, resolution, and projection to the customary binary format used by the GARP algorithm.

The workflow's `path` parameter (`property("KEPLER")+"/workflows/test"`) specifies the path to the working directory that contains a list of *.asc grids to be included in the GARP set. The list is a simple text file that specifies the location of each environmental layer, as shown in the below example:

C:/Garp/Ascii/NorthAmerica/frs6190_ann.asc
C:/Garp/Ascii/NorthAmerica/h_aspect.asc
C:/Garp/Ascii/NorthAmerica/h_dem.asc
C:/Garp/Ascii/NorthAmerica/h_flowacc.asc

The *Layer List Reader* actor reads the layer list text file, specified via the actor's `fileOrURL` parameter. The actor outputs the list as a sequence of file paths. The *Sequence To Array* actor then converts the sequence of paths into an array of paths, which is output to the *ASC To RAW* actor.

The *ASC To Raw* actor converts the *.asc files to binary files (*.raw). First, however, the actor checks to make sure that all of the values in the input *.asc grids fall within the range (0 to 255) required by binary formats. If values fall outside the required range, the actor automatically scales the values to fall within the range. If values fall inside the required range, the actor checks the `scaleRaw` parameter (*Figure 31*) to determine if it should scale the values or leave them intact.

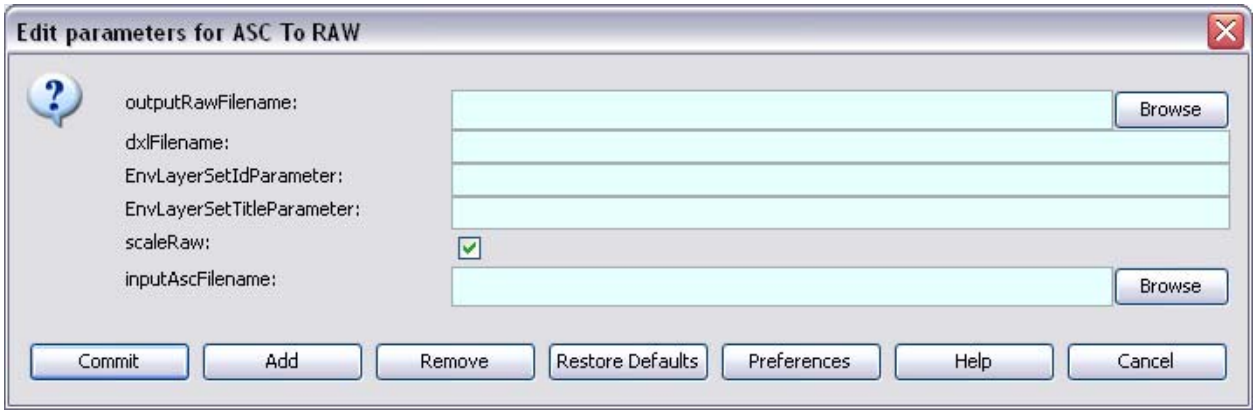


Figure 31: *ASC To Raw* actor `parameters.` Because no values for the `outputRawFilename` and `dxlFileName` parameters are specified, the actor will write the *.raw files to the same directory that the source *.asc files are in (C:/Garp/Ascii/NorthAmerica/). To write the files to another directory enter the directory name in the `outputRawFilename` and `dxlFileName` parameters.

Once the input data have been scaled, the *ASC To Raw* actor generates a .dxl file, which the GARP algorithm uses to identify the layer set. The .dxl file is an XML file that contains header information: a user-defined layer set title and identification number (both specified with *ASC To Raw* parameters), the geographic extent and resolution of the layer set, a list of the environmental layers, and (optionally) a mask layer that defines the area of interest. The generated .dxl file is saved and its file path is sent to the *Display* actor.

Note: The *ASC To Raw* actor can also be used to convert a single .asc layer file to a .raw file for use with GARP. In this case, the *ASC To Raw* actor will output the file path of the converted raw file. When the *ASC To Raw* actor used to convert a single .asc file, the input file path should be connected to the actor's `singleFilenamePort.`

Kepler also includes a workflow specifically designed to create environmental layers for use with GARP. The BaselineLayers_2.xml workflow (*Figure 32*) is very similar to the BaselineLayers.xml workflow discussed in Section 3.4; however, the BaselineLayers_2.xml workflow uses the *ASC To Raw* actor to directly convert the downloaded environmental layers into a *.raw format for use with GARP.

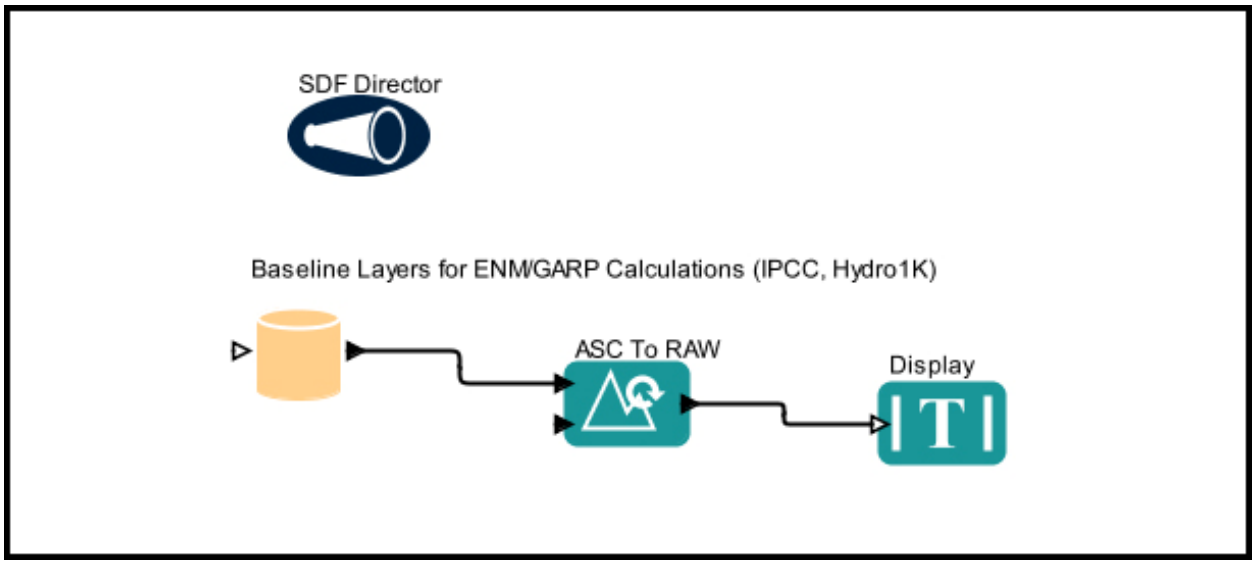
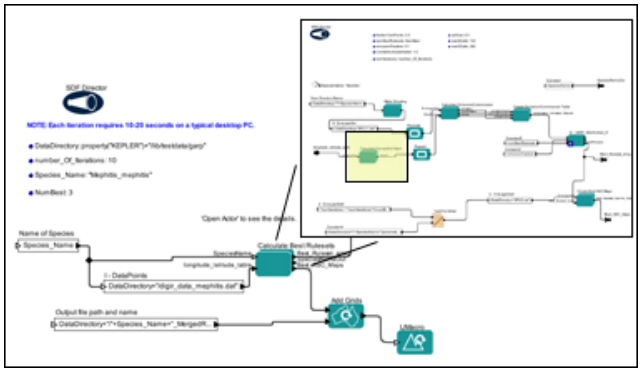


Figure 32: The BaselineLayers_2.xml workflow, designed to format environmental layers for use with GARP from a set of preprocessed climate and topographical layers that are found on the Kepler EarthGrid.

3.6. Creating a mask (generating species' absence points)

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml > *CalculateBestRulesets* sub-workflow > *CalculateConvexHullMask* sub-workflow



Masks are used to help generate species "absence" points, which are required by many robust statistical techniques that require both species occurrence and absence points. Because absence points are rarely available, particularly in broad scale studies, pseudo-absence points are often randomly generated as substitutes. These points must be chosen from within the broad range of the species in order to represent environmental information relevant to the model. In cases in which the geographic extent to be modeled predicatively is much larger than the extent of the known species range, a mask is used to specify the area from which pseudo-absence points may be selected.

The GARP_SingleSpecies_BestRuleSet-IV.xml sub-workflow generates a mask by constructing an area that bounds the known occurrence points, “growing” the area by some user-specified amount, generating a mask (an *.asc grid file in which all cells outside of the identified species

occurrence area are set to “NO_DATA”), and converting the mask to the format required by the GARP algorithm (*.raw).

The *CalculateConvexHullMask* sub-workflow (*Figure 33*) is nested inside the *CalculateBestRulesets* composite actor found in the GARP_SingleSpecies_BestRuleSet-IV.xml workflow.

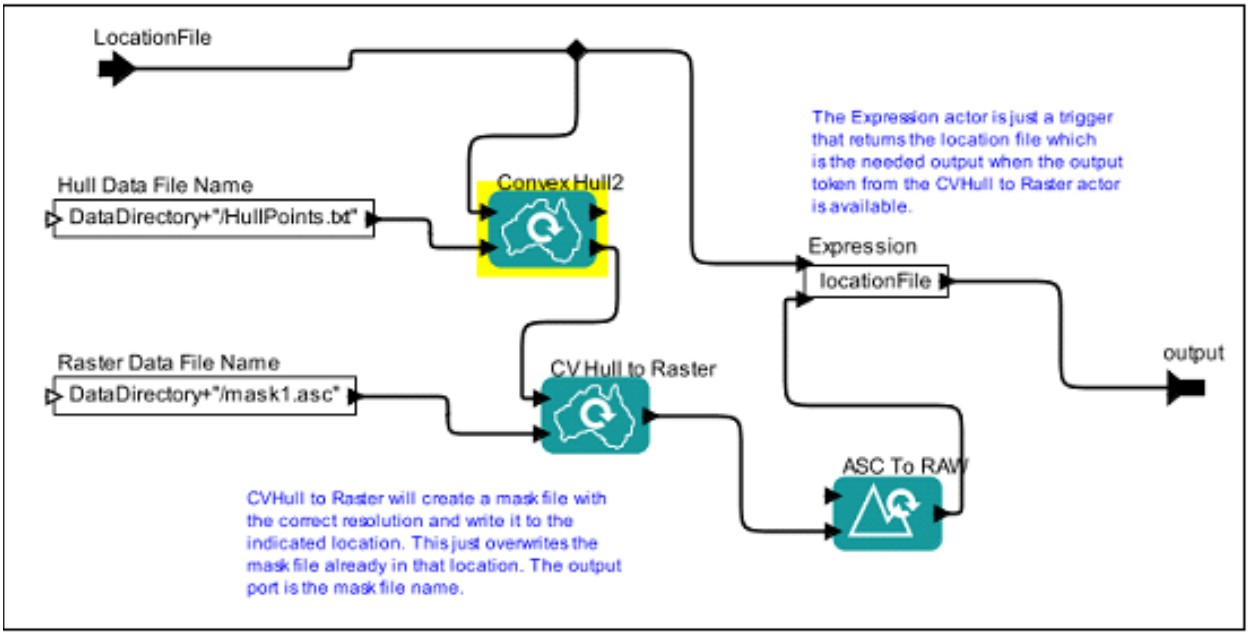


Figure 33: The *CalculateConvexHullMask* sub-workflow found inside the *CalculateBestRulesets* composite actor in the GARP_SingleSpecies_BestRuleSet-IV.xml workflow. This ENM sub-workflow constructs a convex hull around the input species occurrence points, increases its area by a user-defined scaling factor, converts that area to a mask grid, and transforms the mask to the binary format required by the GARP algorithm.

The *CalculateConvexHullMask* sub-workflow requires as input a text file containing species occurrence data (see Section 3.1 for more information). The data are captured as point data, with each point consisting of a longitude and latitude of occurrence. The containing workflow passes the name of the occurrence data file to the sub-workflow via the `LocationFile` port.

The *Convex Hull* actor receives the file name of the occurrence point data and constructs a convex hull (i.e., the smallest polygon that contains the given points). If desired, the constructed polygon can be scaled (made larger or smaller) by a user-defined multiplication factor. The actor saves the points that define the convex hull to a file named via the *Convex Hull* actor's `hullFileName` port. In *Figure 33*, the convex hull file name is `DataDirectory +"/HullPoints.txt"`. `DataDirectory` is a parameter defined at the top level of the workflow (the value is `property("KEPLER")+"/lib/testdata/garp"`). The sub-workflow inherits the defined value from the containing workflow.

The name and location of the convex hull file are passed to the *CV Hull to Raster* actor, which creates and saves a mask file with the correct resolution and extent. The resolution and extent are specified via the actor's parameters (*Figure 34*). Note that the values for the `cellsize,`

`numrows`, and `numcols` parameters refer to values defined and passed by the containing workflow. The values are .5, 360, and 720 respectively.

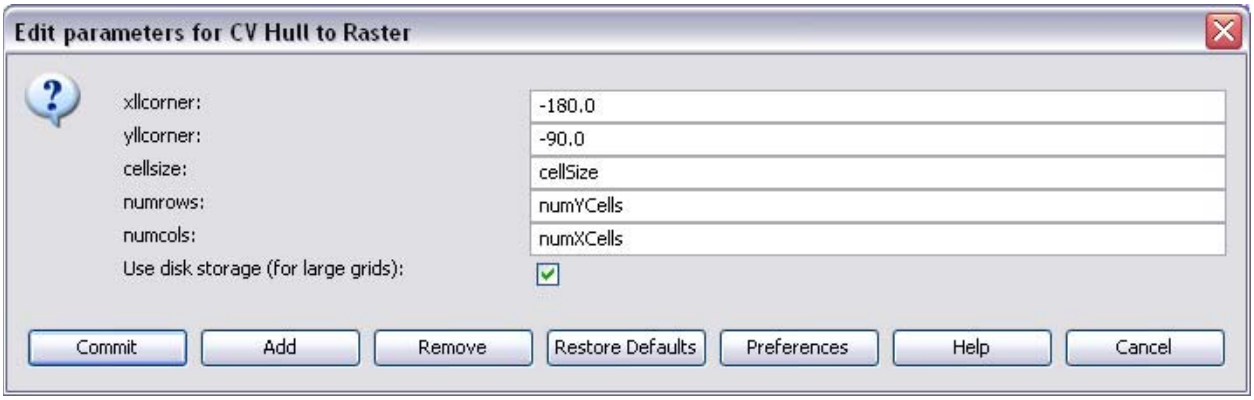


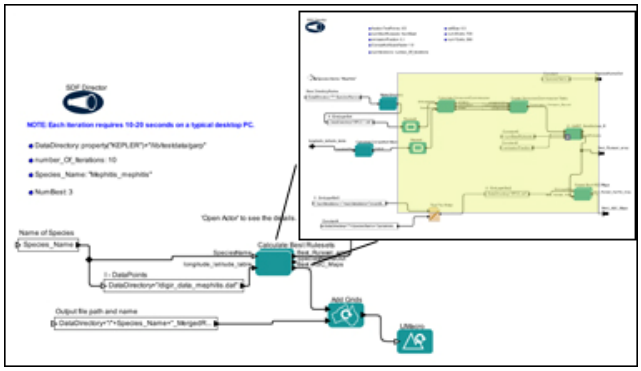
Figure 34: The parameters of the *CV Hull to Raster* actor. Note that the values for the `cellsize`, `numrows`, and `numcols` parameters refer to values defined by the containing workflow. The values are .5, 360, and 720 respectively.

The *CV Hull to Raster* actor writes the mask file to the location specified via the `rasterFileName` port (`DataDirectory+"/mask1.asc"`) and outputs the mask file name. The *ASC to RAW* actor converts the *.asc mask to the binary format required by GARP (*.raw) and passes the file name of the mask to an *Expression* actor. The *Expression* actor is used as a "trigger". Once it receives the name of the mask file (i.e., the *Calculate ConvexHull Mask* sub-workflow has completed processing), the *Expression* actor outputs the name of the occurrence point dataset, which is output to the containing workflow.

Note: The mask file must be specified within the *.dxl file that summarizes the layer set. Since the source IPCC1.dxl file used by the workflow already references a layer called "mask1.raw," the *ASC to Raw* actor simply overwrites that existing file with the new mask. Also note that the IPCC.dxl file, which is included in the data directory as well, is identical to the IPCC1.dxl file except that it does *not* contain the convex hull mask. The IPCC.dxl data include the entire Western hemisphere, and are used when projecting the model (created using the more limited area specified by the mask in the IPCC1.dxl file) to all of North and South America.

4. GARP sub-workflows

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml > *Calculate Best Rulesets* sub-workflow.



GARP (Genetic Algorithm for Rule Set Production) is a genetic algorithm that creates an ecological niche model representing the environmental conditions where a species would be able to maintain populations (for more information, see http://www.lifemapper.org/desktopgarp/). GARP is a stochastic algorithm; each time it is run on a given dataset, it will generate a different model. Typically, the algorithm is run hundreds of times for a given species dataset, the results are analyzed to determine the best models, and the results of the best models are compiled and returned. The GARP sub-workflows are thus iterative, and will be "cycled through" a user-specified number of times.

The portion of the GARP_SingleSpecies_BestRuleSet-IV.xml workflow displayed in *Figure 35* contains the actors used for iterative GARP processing.

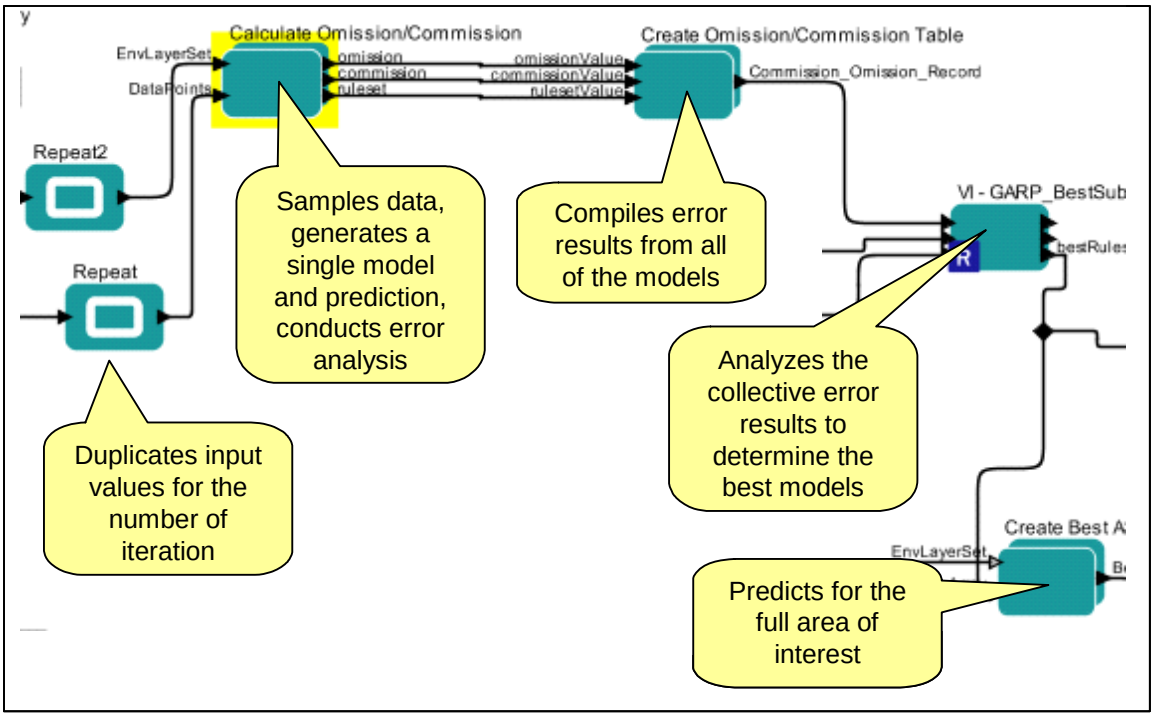


Figure 35: Overview of the GARP modeling approach, which comprises a portion of the GARP_SingleSpecies_BestRuleSet-IV.xml workflow.

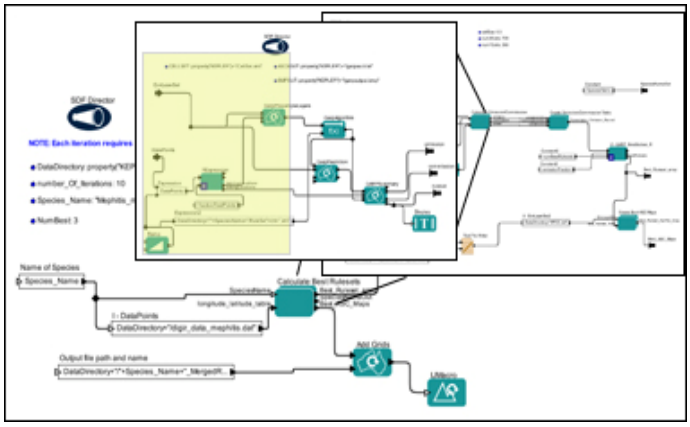
For each iteration, input data (an environmental layer set with a corresponding mask file, as well as a list of occurrence data points) are passed to the GARP algorithm. Occurrence data points are divided into two sets. One set is used to sample the environmental layer set and generate a model that is used to predict species occurrence; the second set of occurrence data is used to test the prediction and generate error metrics.

Error metrics are compiled into a table, which is then analyzed to determine the best predictive models. Selected models are used to predict species occurrence over the entire area of interest (i.e., the full extent of the environmental layers).

The following sections provide more detail about each step in this overall approach.

4.1. Sampling environmental layers

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml > *Calculate Best Rulesets* sub-workflow > *Calculate Omission/Commission* sub-workflow



Model validation requires that some portion of the known species occurrence points be used to test the predictive accuracy of the generated model. An accurate test depends on those points not being used to generate the model, and standard modeling practice is to divide occurrence points into two sets: a set that will be used to "train" the model, and a second set that will be used to test the model. The training set is used to sample the environmental layers and construct a new dataset that contains a vector of environmental characteristics for each occurrence point. The testing set is used after a model is generated.

The sampling process is executed by the *Calculate Omission/Commission* sub-workflow, which is nested inside the *Calculate Best Rulesets* sub-workflow in the GARP_SingleSpecies_BestRuleSet-IV.xml workflow.

The workflow subdivides the occurrence data into two sets with the *RExpression* actor (*Figure 36*). The actor's R script contains a function that randomly samples the occurrence points. The fraction of points to be included in the testing set is specified with the `FractionTestPoints` parameter, which is set to `.5` via a parameter set on the Workflow canvas.

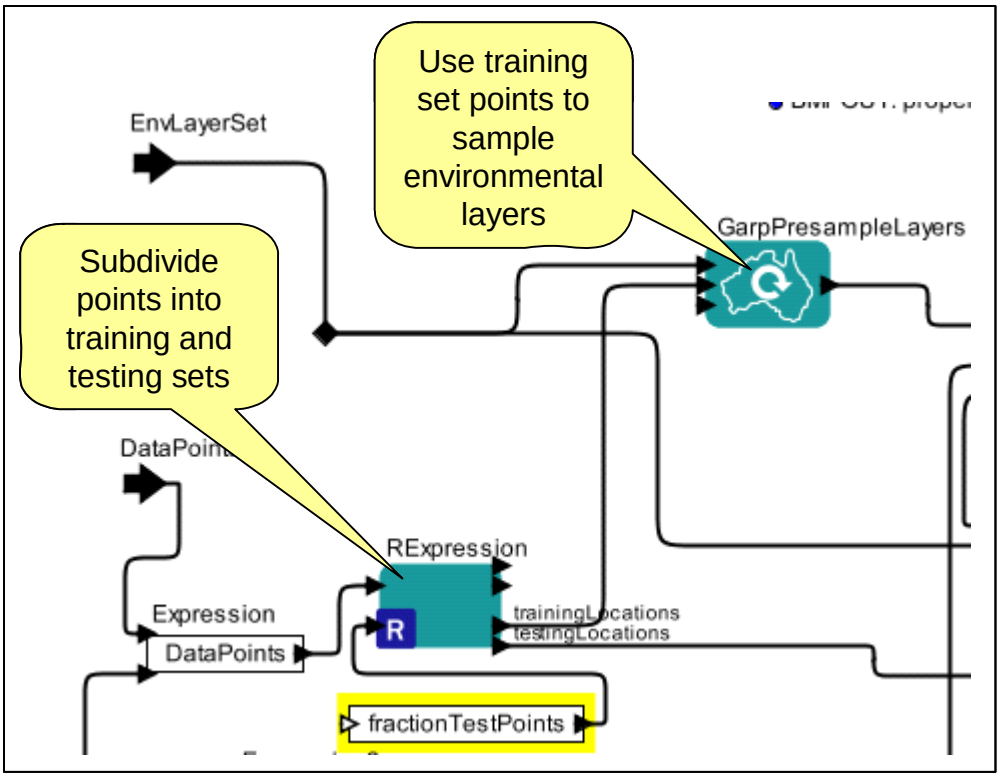


Figure 36: Portion of the *Calculate Omission/Commission* sub-workflow that divides the occurrence data into training and testing sets and then uses the training set to sample the environmental layers. The testing set is used after a model is generated to conduct error analysis and validate the model.

A *Ramp* actor provides input to the *RExpression* actor, controlling how the entire sub-workflow is iterated. The *Ramp* actor is the equivalent of a "for loop" in many traditional computer languages. Its parameters include an initial value, the amount the value is incremented each time the actor fires (the 'step'), and the upper limit of the value (the firingCountLimit) (*Figure 37*).

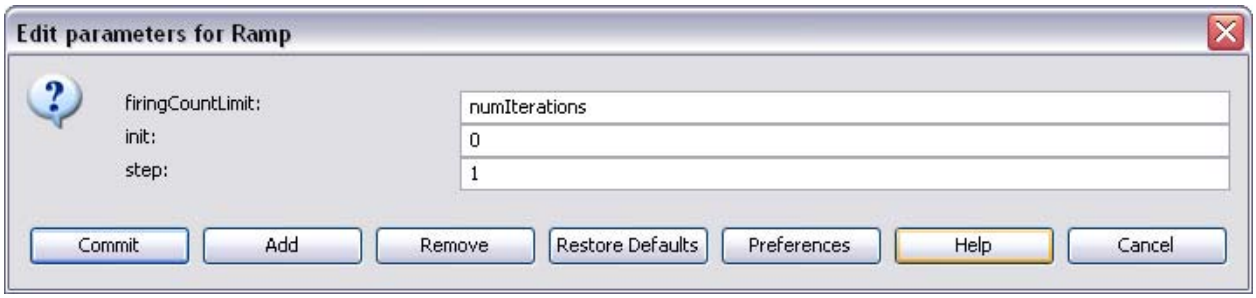


Figure 37: The *Ramp* actor parameters from the prototype ENM *Calculate Omission/Commission* sub-workflow. The `firingCountLimit` is set to the value "numIterations", which is defined as 10 by a workflow parameter.

The training set of occurrence points is output to the *GarpPresampleLayers* actor, which also receives the file name of the *.dxl file that describes the environmental layers and the mask file generated by the *CalculateConvexHullMask* sub-workflow discussed in Section 3.6. The *GarpPresampleLayers* actor selects pseudo-absence points from within the area specified by the

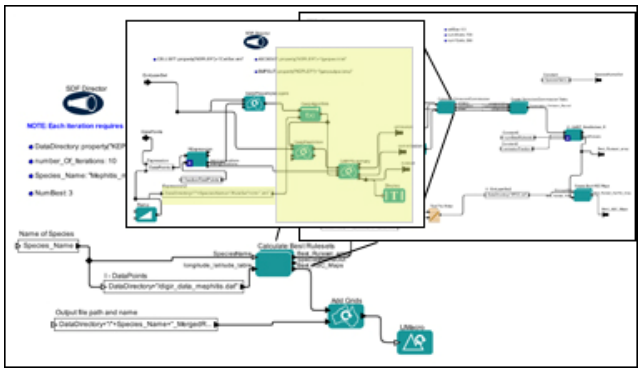
mask. The pseudo-absence points are added to the set of occurrence points so that both presence and absence points will be available to the GARP algorithm.

The *GarpPresampleLayers* actor samples the set of environmental layers at each point in the training set, and outputs a dataset containing the environmental characteristics at each sampled occurrence point.

CAUTION: The GARP code crashes if the occurrence locations are all outside the extent of the environmental layers described in the *.dxl file.

4.2. Constructing a model and generating error statistics

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml> *Calculate Best Rulesets* sub-workflow > *Calculate Omission/Commission* sub-workflow



Once species occurrence points have been divided into two sets ("training" and "testing"), and the environmental data for each point in the training set has been sampled, the GARP algorithm can be used to generate a model that predicts where a species is likely to be present based on the sampled environmental factors. The generated model is then tested to see how well it performs.

The *GarpAlgorithm* actor (*Figure 38*) reads the sampled data output by the *GARPPresampleLayers* actor and uses them to generate a set of rules about climate conditions under which a species might survive (e.g., "If annual precipitation is more than 5 mm and less than 12 mm then the species is present."). The generated ruleset is saved to a file. Users specify the location and name for the ruleset file via an *Expression* actor, which passes its value to the *GarpAlgorithm* actor. In the example workflow, the *Expression* actor has the value `DataDirectory+"/"+SpeciesName+"/RuleSet"+cnt+".xml`, which evaluates to a unique name with each iteration of the workflow. The specified expression uses the `SpeciesName`, defined at the top-level of the workflow (Mephitis_mephitis), and the `cnt`, which references the value passed to the *Expression* actor via its `cnt` port.

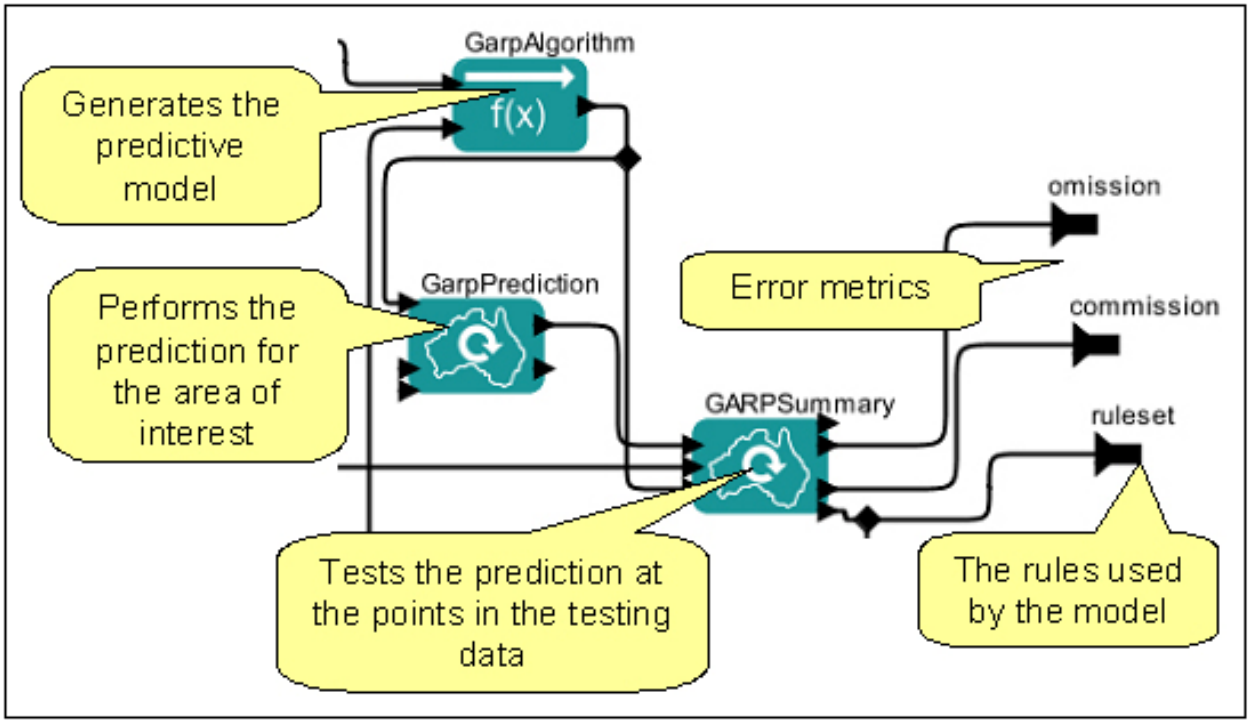


Figure 38: GARP model generation, prediction, and error calculation.

The *GarpAlgorithm* actor outputs the file name of the ruleset to the *GarpPrediction* actor, which uses the rules to predict species presence or absence at each cell within the dataset (which includes the environmental layers and mask file. Masked cells are not considered). Presence is indicated with a value of "1"; absence with a value of "0". If the *GarpPrediction* actor cannot make a prediction for a given cell, it outputs the value "254". The actor outputs both an ASCII file (*.asc) and a bitmap file (*.bmp) of the resulting prediction.

After a model has been created, the *GarpAlgorithm* actor outputs it to the *GARPSummary* actor, which tests the prediction for accuracy. The *GARPSummary* actor uses the testing set of occurrence points (discussed in Section 4.1) and compares each point to the corresponding point in the model to see if it has been correctly predicted or not. The testing set includes both known occurrence points and pseudo-absence points generated by the *GarpPresampleLayers* actor.

If the model prediction differs from the testing data, the *GARPSummary* actor generates an error. The number of these errors compared to the total number of testing points is reported as a percentage of omission or commission errors:

Omission errors: Errors of omission occur when a species is present but the model predicts it to be absent. These are true model errors.

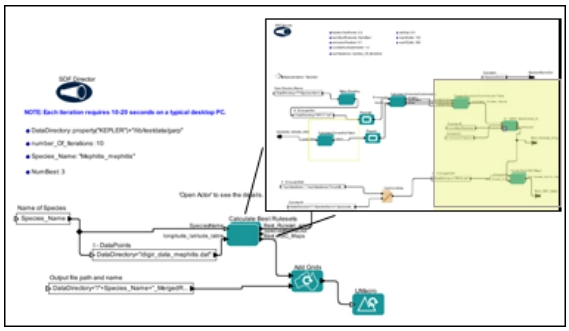
Commission errors: Errors of commission occur when a species is predicted to be present but is really absent. Because absence points are randomly generated (pseudo-absence), a commission error may not truly be an error (the model has no way of knowing if the species is really present or not). Since there is no way to calculate true commission error based on pseudo-absence points, a more general method is used: the commission error for each model is the proportion of

predicted species presence area to the total area. Very high or very low proportions are more likely to be erroneous.

The *GARPSummary* actor outputs the values of the omission and commission errors, which are used to determine and select the best models. In addition, the actor outputs the file name of the ruleset used to generate each prediction. The ruleset is used by downstream actors to "recreate" the selected, best models.

4.3. Selecting the best models and making predictions

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml > *Calculate Best Rulesets* sub-workflow



After the workflow has used the GARP stochastic algorithm to generate a number of models, and then calculated omission and commission errors for each one, it must compile and compare the error results, select the best models, and generate the best predicted species distribution (*Figure 39*).

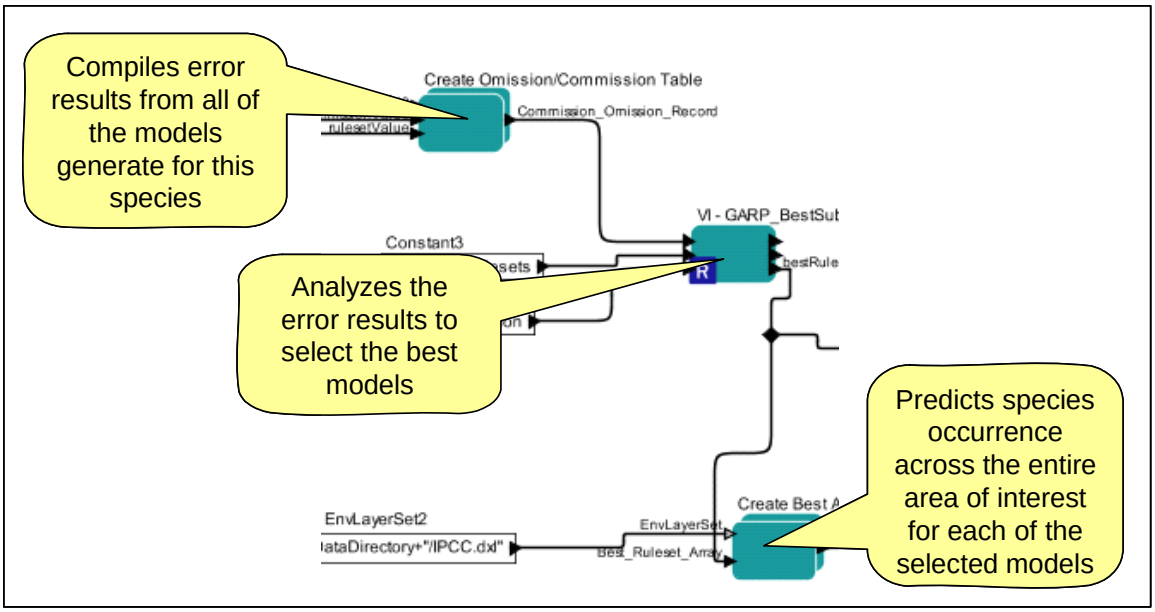


Figure 39: Portion of the *Calculate Best Rulesets* sub-workflow used to select the best models and make predictions from those models.

The *CreateOmissionCommisssionTable* sub-workflow (*Figure 40*) uses three *Sequence to Array* actors to convert the sequence of data tokens produced by the previous sub-workflow into arrays, which are output to a *Record Assembler* actor. The *Record Assembler* actor has three user-defined input ports: `commission, omission, ruleset,` which receive the corresponding arrays. The actor assembles the data into a single record, which is output. A record is a composite data type consisting of one or more elements. Each element is named and can have a distinct type. For example, {number=1, name="dog"} is a record containing two elements. The first element, named "number", contains an integer value. The second element, named "name", contains a string value.

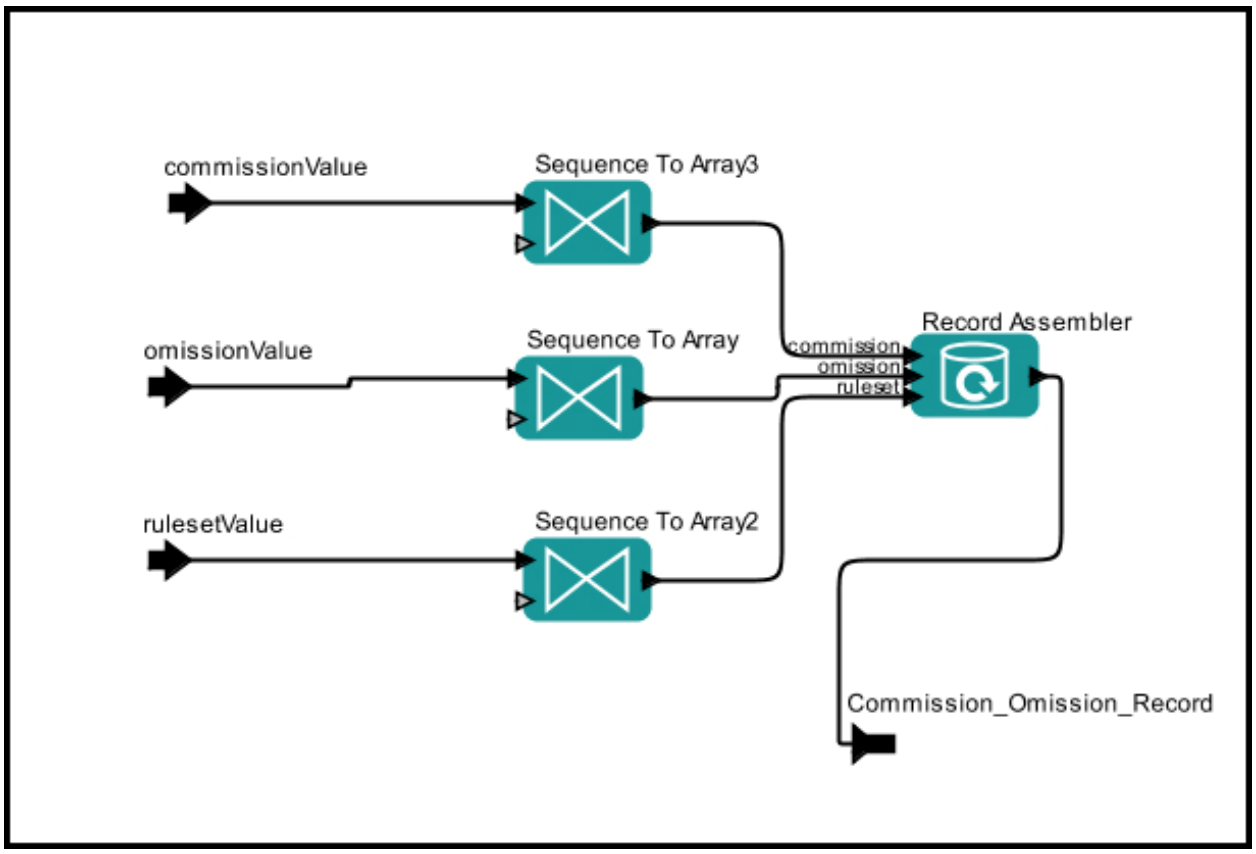


Figure 40: The *Create Omission/Commission Table* sub-workflow. Actors in this sub-worflow assemble the ruleset and the omission and commission error data into a single table that is used by the *VI - GARP_BestSubset_R* actor.

The *Create Omission/Commission Table* sub-workflow outputs the compiled data to the *VI - GARP_BestSubset_R* actor, which is a custom *RExpression* actor that reads the table and selects the best models based on a comparison of omission and commission errors.

Omission Error Test: Generally, the *VI - GARP_BestSubset_R* actor receives a user-specified "omission threshold" via its input ports. The omission threshold can be specified in one of two ways: (1) as an error "cutoff" that disqualifies models that exceed the stated error (e.g., select models with an omission error less than some percentage), or (2) as a number of models to select (e.g., select the 200 models with the lowest omission error). Only models that meet the specified threshold will be acceptable.

Commission Error Test: The commission error for each model is the proportion of predicted species presence area to the total area. Assuming that very high or very low proportions are more likely to be in error, the *GARP_BestSubset_R* actor selects models that are closest to the median. Generally, the actor selects 50% of the models, excluding the 25% of the models that predict the highest presence area and the 25% that predict the lowest presence area. The commission testing criteria are specified in the R-script used by the actor.

Models that pass both the omission error threshold test and the commission error test are compiled as the best subset of models. These models are passed as an array to the *Create Best*

ASC Maps sub-workflow (*Figure 41*). A Relation is used to branch the array of best rulesets so that it is also output to the containing workflow.

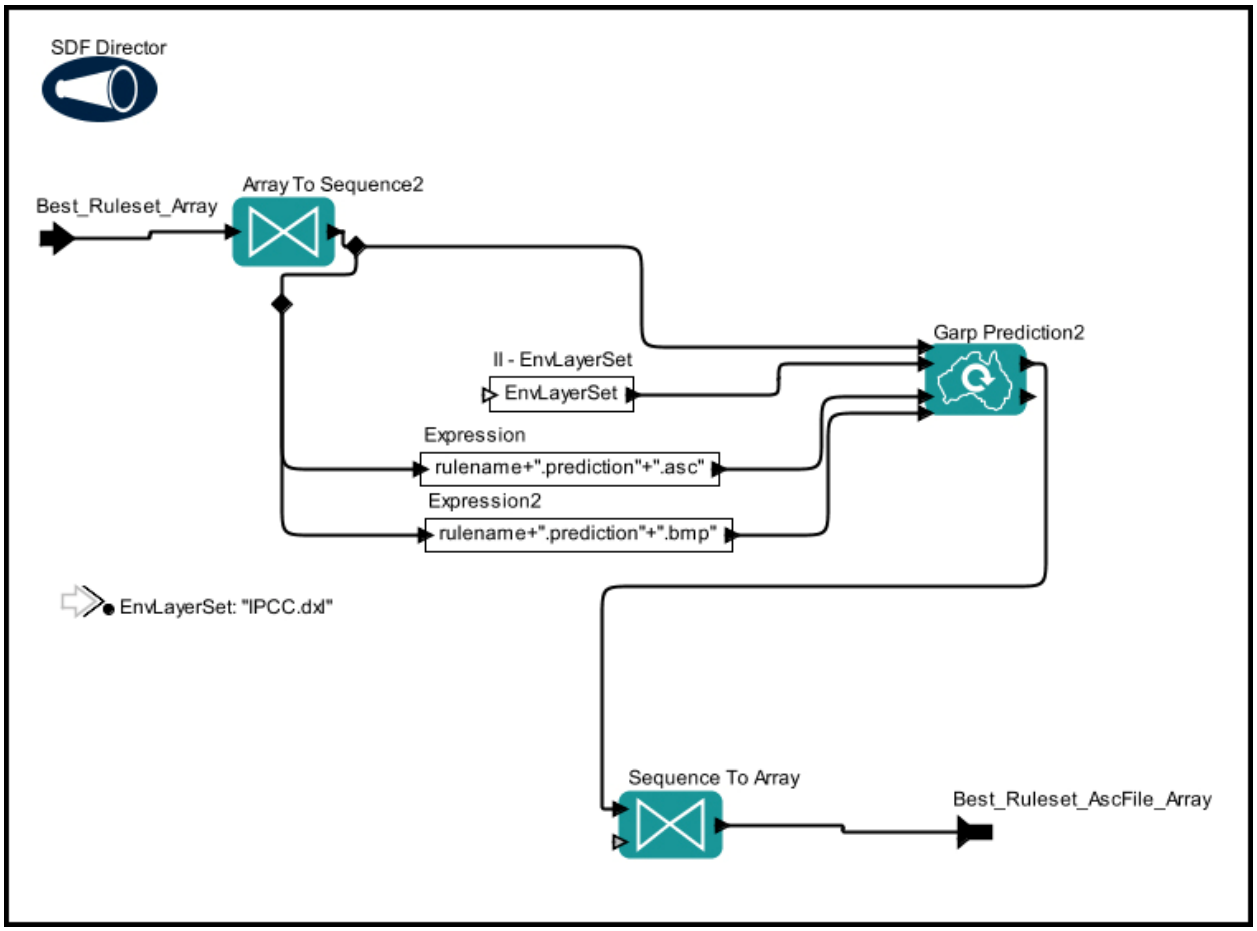


Figure 41: The *Create Best ASC Maps* sub-workflow, which "re-assembles" the best species distribution models using the GARP ruleset and environmental layers used earlier in the workflow.

The *Array To Sequence2* actor converts the array of ruleset names into a sequence, which is fed to the *Garp Prediction2* actor. In addition to the ruleset names, the *Garp Prediction2* actor receives the name of an environmental layer set (IPCC.dxl), as well as file names to use for the *.asc and *.bmp files that the actor generates. Note that the IPCC.dxl file includes data for all of North and South America (not just the masked region initially used to generate the best models). In this workflow step, the best models are used to predict species occurrence across the entire area of interest included in the environmental layers.

The *Garp Prediction2* actor outputs a sequence of *.asc file names for each of the "best" GARP models. The *Sequence To Array* actor converts the sequence into an array of file names and outputs it to an *Add Grids* actor in the containing workflow (*Figure 42*).

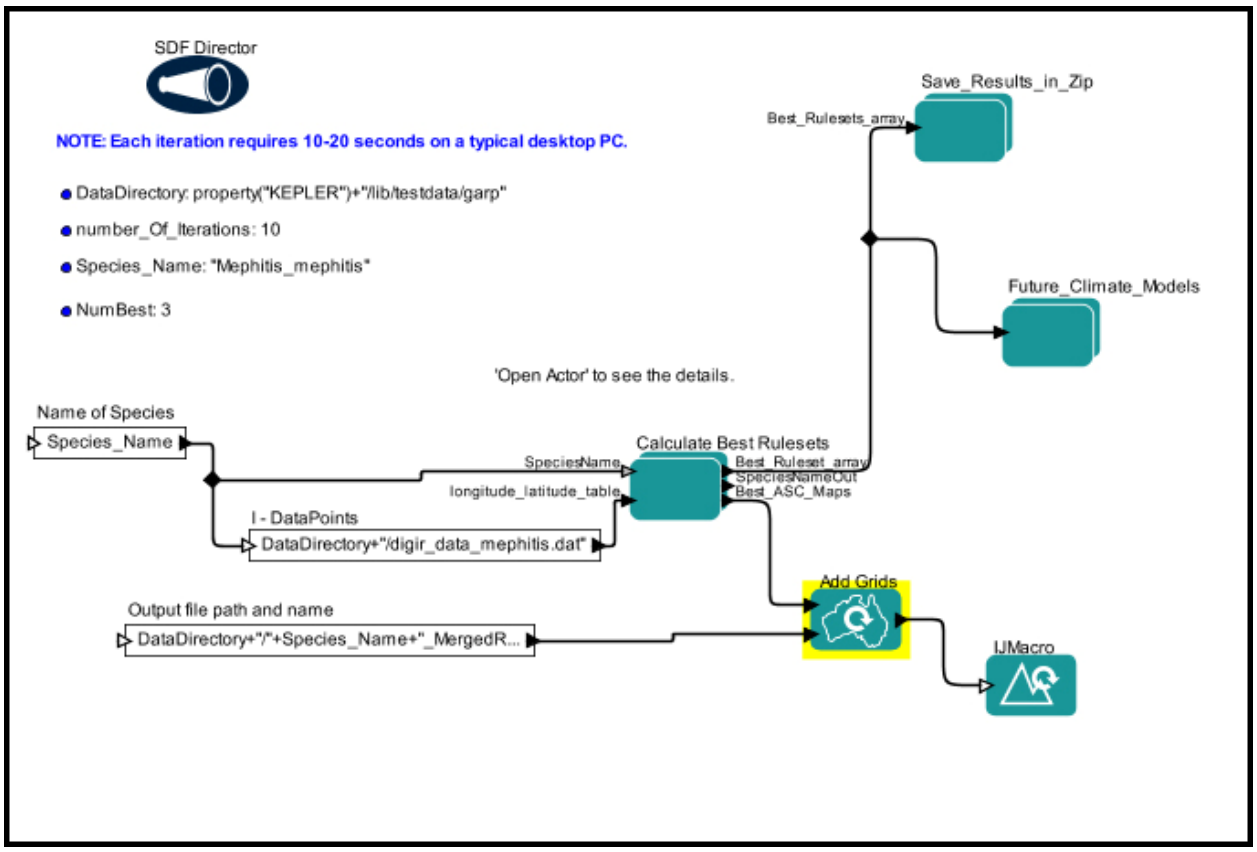


Figure 42: The array of *.asc files generated by the best GARP models in the *Calculate Best Rulesets* sub-workflow is passed to the *Add Grids* actor in the top-level workflow.

The *Add Grids* actor reads the geospatial image files (*.asc), merges the files by adding the cell values, and outputs the name of the merged file. Grid cells that have the most predictions of presence across the best subset of models have the highest probability of species occurrence. Thus, cell values (1 for presence, 0 for absence, or 254 for cells where no prediction can be determined) are added for all pixels in the input file list. If the *Add Grids* actor merges 3 best predictions and in each case presence is predicted, the result will be '3' (if only one of the results predicts a presence, the merged value will be 1). Similarly, a result of 0 indicates that all of the individual predictions predict absence at that location. Any sum of 254 or more indicates that one of the predictions did not determine presence/absence for the location. Values of 3 are more probable than 2 or 1, but values greater than 254 are hard to evaluate.

The *Add Grids* actor outputs the merged prediction, which is displayed by the *IJMacro* actor (*Figure 43*). The *IJMacro* actor uses ImageJ, an application used to display and process a wide variety of images (tiffs, gifs, jpegs, etc.). For more information about ImageJ, see http://rsb.info.nih.gov/ij/. The actor is set to display areas with a higher probability of presence in a brighter color. White indicates areas where no prediction can be determined based on the data.

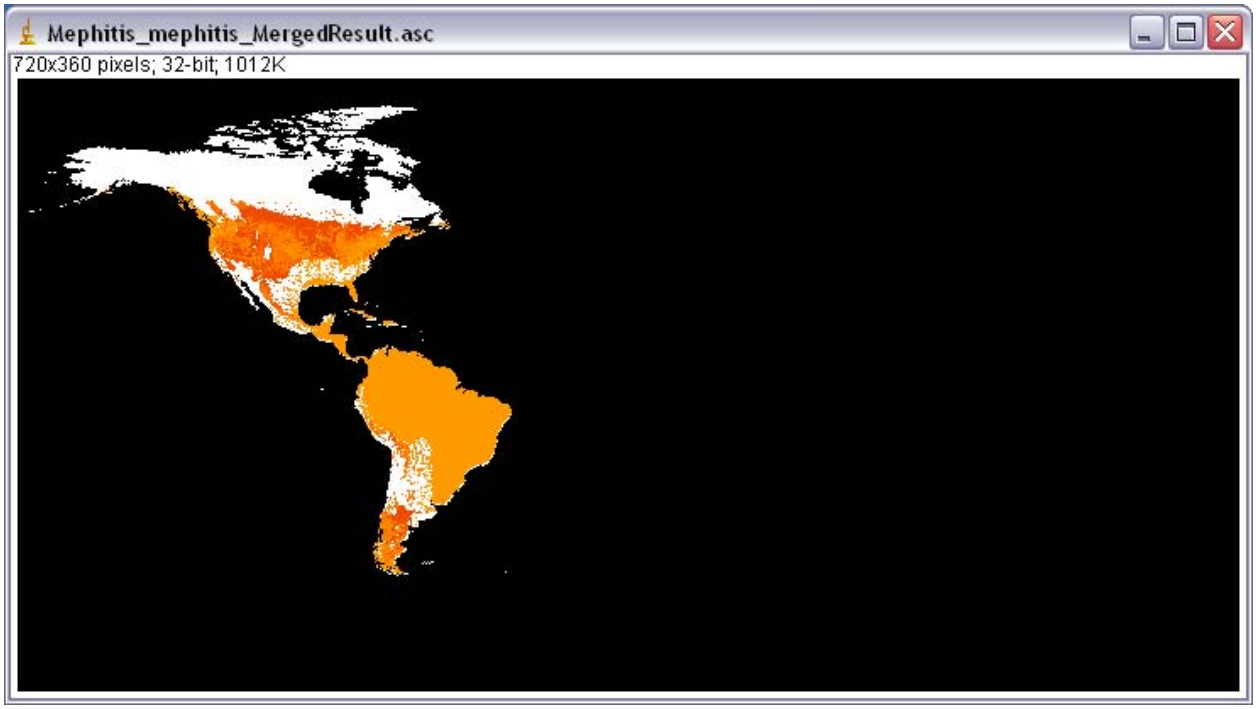


Figure 43: Output of the *IJMacro* actor. The map shows the species distribution predicted by the combined "best" GARP models. Brighter color indicates areas with a higher probability of presence; white indicates areas where no prediction can be determined.

Roll over the prediction map to display the value of each cell, displayed in the ImageJ toolbar (*Figure 44*).

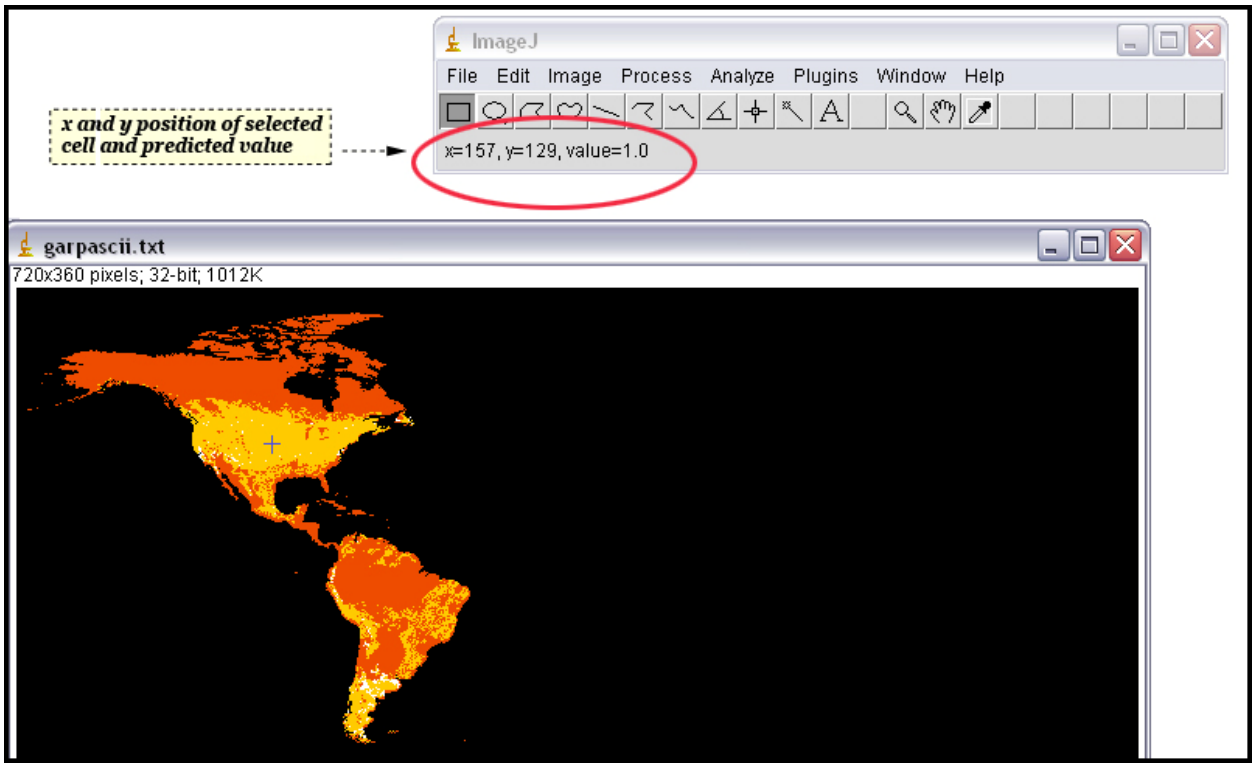
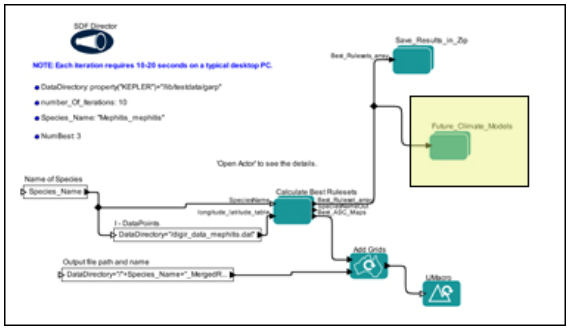


Figure 44: The ImageJ toolbar displays the cell value of each mapped point when the cursor is over it. In the example, the cursor is over a point (157, 129) that has a value of 1, which indicates species presence. White areas indicate places where no prediction can be determined.

The colors and intensity of the colors used to indicate presence and absence can be customized. See the User Manual for more information.

4.4. Predicting species distributions under future climate change scenarios

See $KEPLER/demos/SEEK/GARP_SingleSpecies_BestRuleSet-IV.xml > *Future_Climate_Models* sub-workflow



The final step in the ENM workflow uses the best models generated by the GARP algorithm using historical climate datasets to predict species' distribution under future climate change scenarios. The *Future_Climate_Models* sub-workflow (*Figure 44*) predicts distributions based on environmental layer sets that include future climate data instead of historical climate data

The workflow reads an array of the best GARP rulesets, prepares environmental layer sets containing future climate data, generates a prediction, and outputs a map of each scenario.

NOTE: This sub-workflow is designed to create predictions for multiple future climate scenarios (e.g., IPCC_2020, IPCC_2050, etc); however, the current workflow does not fully demonstrate this potential because only the IPCC_2020 future climate data have been processed. In order to actually predict results for 2050, the "climateList.txt" containing references to the data for 2050 must be updated to point to the correct data. See below for more information.

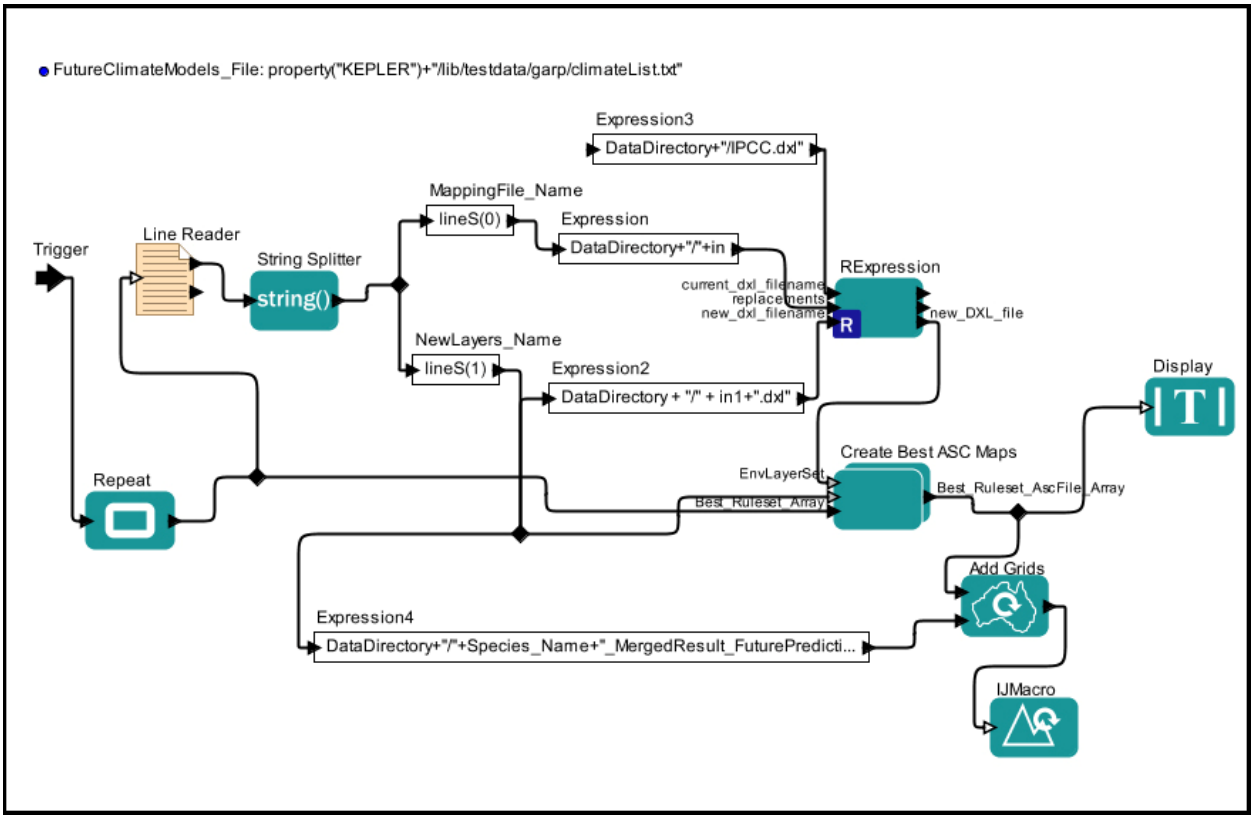


Figure 45: The *Future_Climate_Models* sub-workflow, which creates predictions based on environmental layer sets that include future climate data instead of historical climate data.

The *Future_Climate_Models* sub-workflow is triggered by the arrival of an array of the best rulesets, output by the *Calculate Best Rulesets* sub-workflow. The rulesets are created by the GARP algorithm, and contain rules about climate conditions under which a species might survive.

The array of rulesets is passed to a *Repeat* actor, which is used to control the workflow iterations. Its `numberOfTimes` parameter is set to 2, instructing the workflow to repeat twice. The data are then branched via a Relation: one set is fed to a *LineReader* actor (used to trigger the data preparation process); the other is input to the *Create Best ASC Maps* sub-workflow (discussed in Section 4.3), which generates the predicted species distribution maps.

Preparing environmental layer sets for future climate predictions consists of four basic steps (*Figure 45*):

1. Reading a list of climate scenarios to run
2. Mapping historical data to future data for each scenario
3. Specifying a name and path for the future dataset
4. Constructing a new layer set (*.dxl) for use with GARP

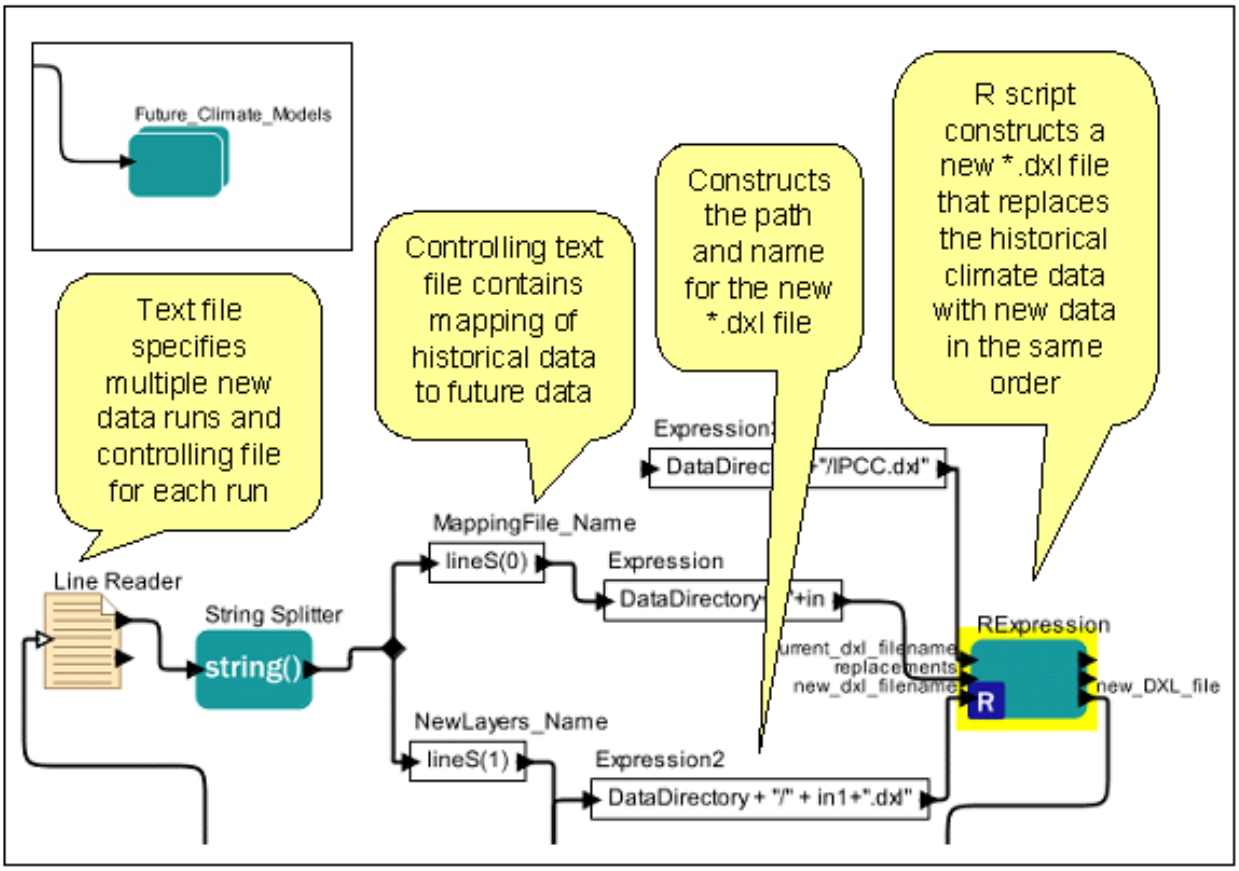


Figure 46: The major steps for preparing data used to make predictions based on future climate scenarios.

The workflow uses a *Line Reader* actor to read a list of climate scenarios that should be run. The *Line Reader* actor reads the file specified by its `fileOrURL` parameter (property("KEPLER")+"/lib/testdata/garp/climateList.txt"). The specified file contains a list of future climate data and an identifier for each:

```
NewClimate.txt,IPCC_2020
NewClimate.txt,IPCC_2050
```

Note that the data source for both IPCC_2020 and IPCC_2050 is "NewClimate.txt." The "NewClimate.txt" file is a list of text files that contain future climate data for 2020. The example workflow uses the same file ("NewClimate.txt") for both scenarios because the data for 2050 has not yet been generated. In a real workflow run, the dataset for 2050 would be a unique file.

Note: The climate variables (e.g., temperature, rainfall, etc) in the historical and future datasets must be specified in the same order.

The *Line Reader* actor outputs each line of the "climateList.txt" file as a string, which is passed to a *String Splitter* actor. The *String Splitter* actor "chops" each string at the "," and outputs the segments an array (e.g., {NewClimate.txt, IPCC_2020}). This output is branched to two *Expression* actors. The *MappingFile_Name* actor references the name of the climate data source (e.g., "NewClimate.txt") with the expression `lineS(0)`. This expression identifies the first (i.e., "0") element in the array that is input via the actor's `lineS` port. The *NewLayers_Name* actor references the second element in the array (e.g., "IPCC_2020").

The workflow's *Expression* actor constructs the path to the future climate dataset. The actor references the `DataDirectory` parameter defined by the containing workflow (`property("KEPLER")+"/lib/testdata/garp"`) as well as the value passed to its `"in"` port (e.g., `NewClimate.txt`), and combines the two to form a path. This path is passed to a customized *RExpression* actor, which uses an R-script to perform the actual data replacement and generate a new layer set (*.dxl) containing the future climate data. The *Expression2* actor constructs a name for the new *.dxl file.

The new environmental layer set is passed to the *Create Best ASC Maps* actor (described in Section 4.3). Output maps are combined with an *Add Grids* actor (Section 4.3.) and displayed with an *IJMacro* actor (*Figure 46*)

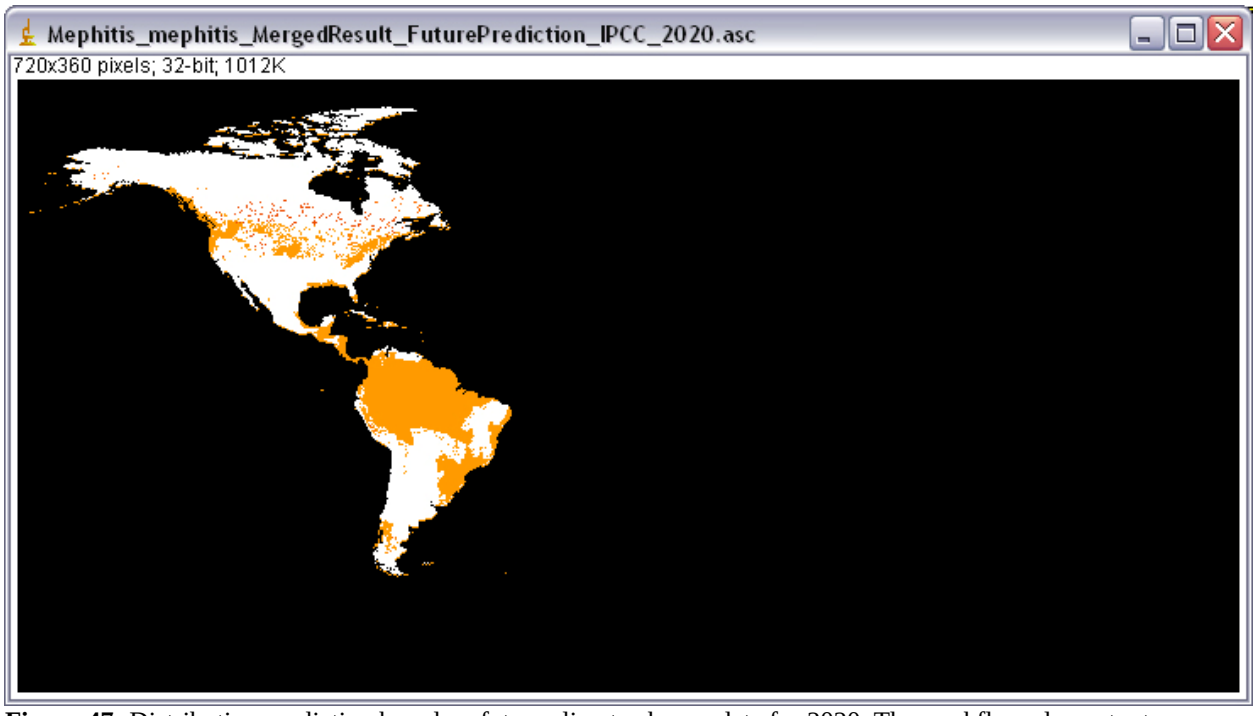


Figure 47: Distribution prediction based on future climate change data for 2020. The workflow also outputs a map for 2050; however, this map is based on 2020 data and is included for illustrative purposes only. Brighter colors indicate presence; white indicates areas where no prediction can be determined.

5. Common modifications

One of the strengths of the hierarchical, component-based design of Kepler is that workflow modifications are more easily accomplished than they are with "single program" approaches, which combine all functionality into a single application. Still, modifications to ENM workflows may require some effort, especially if new data or algorithms must be incorporated into Kepler. Modifications will become much simpler as the community of niche modelers creates and adds data and models to Kepler. New workflows can be shared among users by compositing the workflow and exporting it as a *.kar file (right-click => Export archive). Users with appropriate permissions can upload the composite actor to the Kepler repository for distribution with the Kepler actor libraries. See the Kepler User Manual for more information about creating and sharing kar files.

5.1. Saving and sharing workflows

ENM workflows (or any other workflow for that matter) can be saved and shared in a few easy steps:

1. Save the workflow by selecting Save from the File menu. The workflow is saved as an XML file that can be emailed or posted to a Web server. If the workflow is posted to a website, users can open the workflow by selecting Open URL from Kepler's File menu and typing in the URL of the remote workflow. Workflows sent via email can be opened via the File > Open File menu item.

   NOTE: An easy way to add additional instructions or contextual information (e.g., how to download and run the workflow) is simply to create a web page with the instructions that includes a link to the workflow. Users will be able to open the web page and navigate to the workflow via the Open URL option.

2. If the workflow contains actors that are not included in Kepler's default actor library (or that users may not have on their local machines), upload the actors to the repository. The Kepler Repository allows users to both upload and download workflow components to a centralized server where they can be searched and re-used. This functionality will be included in the next beta release (beta4) of Kepler, and is currently available in the Kepler nightly build. For more information about downloading and running the nightly build, see the User Manual.

   To upload an actor to the repository, right-click the actor and select Upload to Repository from the drop-down menu (*Figure 48*). Note that each actor should have a unique and descriptive name. To rename an actor, right-click it and select Customize Name from the drop-down menu.

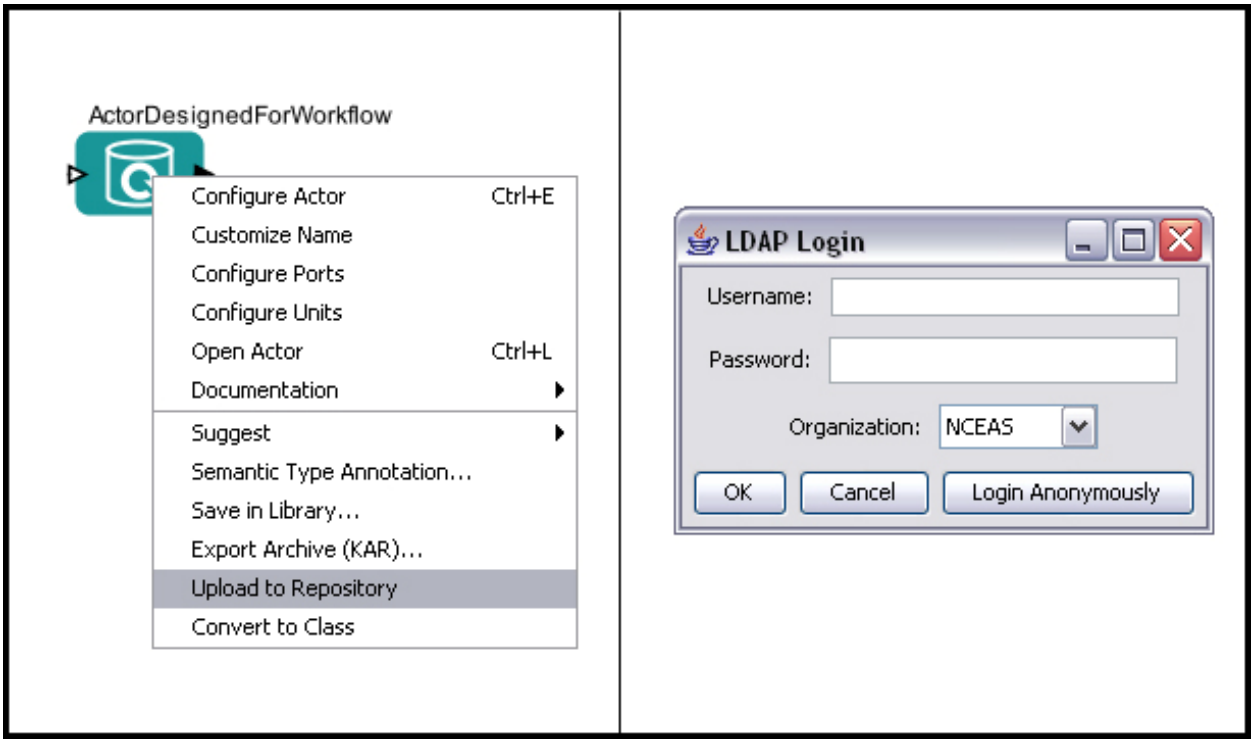


Figure 48: Uploading actors to the Kepler repository. Right-click the actor and select the Upload to Repository menu item (left). Log in to the Kepler repository using the pop-up authentication dialog (right).

3. Enter a username, password, and organization OR, if you do not have a user account, click the Login Anonymously button to upload the actor without credentials. A status bar appears as the actor is uploaded.

4. Click "Yes" in the dialog box that asks whether the component should be publicly accessible in the library. Each component must have a unique LSID, which identifies it. The system will automatically assign an LSID if necessary. A confirmation screen appears when the upload is complete.

5. Users interested in sharing the workflow must download the required actors from the repository in order for the workflow to load properly. To search for and download actors from the repository, select the "Search repository" checkbox in the Components tab and type in the name of the required component (*Figure 49*). The component will automatically download when a user drags and drops the search result onto the Workflow canvas. Note: This functionality will be included in the next beta release (beta4) of Kepler, and is currently available in the Kepler nightly build. For more information about downloading and running the nightly build, see the User Manual.

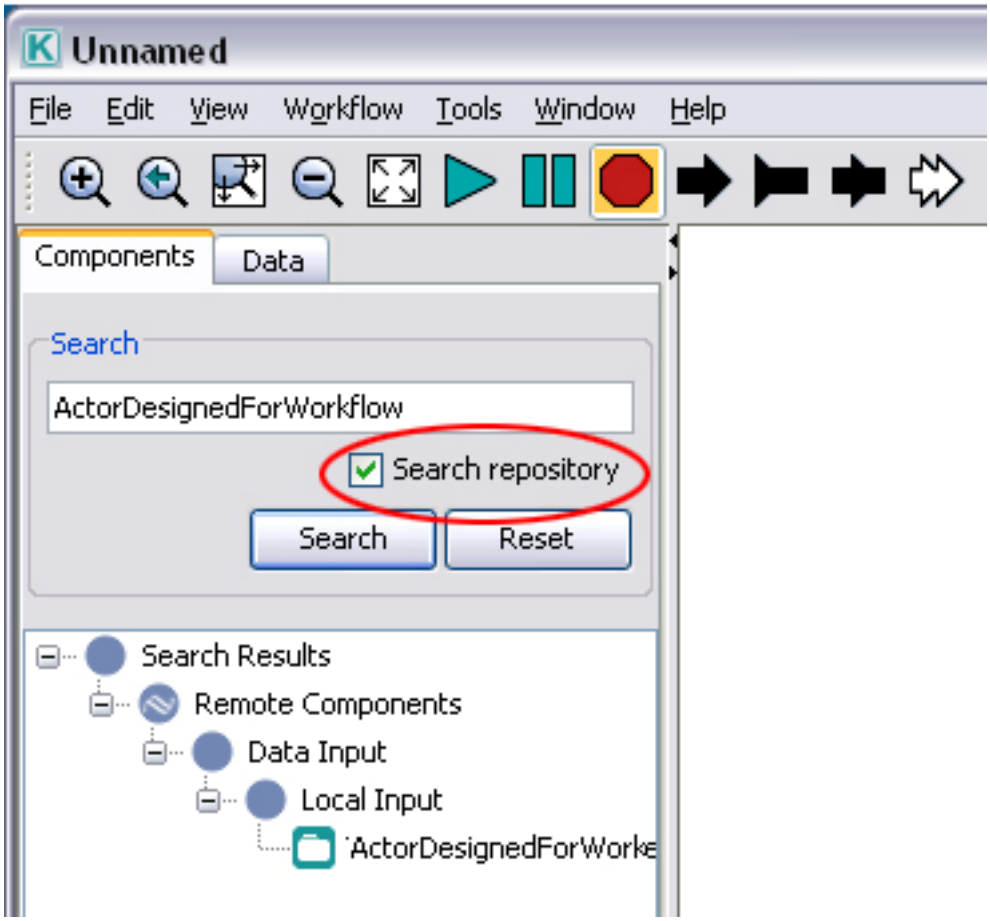


Figure 49: Searching the Kepler repository for components. This feature is currently available in the Kepler nightly builds, and will be incorporated into the next beta release of Kepler (beta4)

~~NOT CURRENTLY WORKING~~ ~~Workflows can also be stored and transported in the Kepler archive format (*.kar), which is a zip file that includes everything necessary for importing and running the workflow on any system that has Kepler installed. To save and share a workflow:~~

~~1.Save the workflow as a *.kar file by selecting "Export archive (KAR)" from the File menu. Select a folder to save the archive in and type a name for the archived file (e.g., "ENMWorkflow.kar").~~
~~2.Share the *.kar file with colleagues (e.g., post it to a project website where it can be downloaded).~~
~~3.Users can download the archived workflow and then open it by selecting "Import archive (KAR)" from Kepler's File menu.~~

5.2. Changing input data

Adding new data may require making those data available on the Kepler EarthGrid (see Kepler's User Manual for more information). Alternatively, data can be stored as a text file and workflows can manipulate those data into the correct format.

Different sources of species occurrence data can be incorporated into ENM workflows simply by formatting the data as a text file with tab delimited longitude/latitude values, and substituting the new file for the existing occurrence file used in the workflow. Alternatively, a dataset that contains occurrence information can be added to the Kepler EarthGrid and accessed via the Data Search tab. In this case, the appropriate columns can be extracted by the workflows that make use of Digir data (see Sections 3.1.3 and 3.1.4).

Changing environmental datasets requires a bit more effort because all layers must use a longitude/latitude coordinate system and have comparable extents and resolutions. The processing required depends on the characteristics of the source data. This processing can be conducted inside or outside of Kepler.

Once the set of comparable layers has been constructed, the layers can be "collected" into a set (described by a *.dxl file) and incorporated into the example ENM workflow as discussed in Section 3.5.

Existing datasets that could potentially be of interest to users are:

- WildFinder Database: ~30,000 species in four taxa (amphibians, reptiles, birds, and mammals): http://www.worldwildlife.org/science/data/wildfinder.cfm
- Birds of the Western Hemisphere: http://www.natureserve.org/getData/birdMaps.jsp
- World's Amphibians: http://www.natureserve.org/getData/amphibianMaps.jsp
- Mammals of the Western Hemisphere: http://www.natureserve.org/getData/mammalMaps.jsp
- USGS Tree Species Range Maps: 600+ tree species in SHP polygon format: http://esp.cr.usgs.gov/data/atlas/little/
- Global monthly min/max temperature, precipitation, and 19 bioclimatic variables at 30 arc-seconds (~1 km) resolution: http://www.worldclim.org/current.htm
- Global 10 minute resolution climatology dataset: http://www.cru.uea.ac.uk/cru/data/tmc.htm
- 3) NASA MODIS global EcoSystem (land cover) dataset at 1 minute resolution: http://modis-atmos.gsfc.nasa.gov/ECOSYSTEM/
- 4) NASA MODIS global NDVI dataset at 1 minute resolution (2000-2004 average): http://modis-atmos.gsfc.nasa.gov/NDVI/index.html
- Terrestrial Ecoregions of the World (800+ ecoregions): http://www.worldwildlife.org/science/data/terreco.cfm

5.3. Replacing the modeling algorithm

Adding new modeling algorithms requires use of a) the *CommandLine* actor for code that runs from a command line, b) scripting in R and use of the *RExpression* actor, or c) “wrapping” the code in Kepler with Java. Please see the actor documentation for more information about using the *CommandLine* and *RExpression* actors. Wrapping code with Java is discussed in the Kepler Developer documents. Once the new algorithm is available in Kepler other workflow modifications may be necessary to accommodate different input/output formats.

The GARP model originally consisted of a single program that presampled environmental layers based on a training set of occurrence data, generated a model, and conducted error analysis (see http://www.lifemapper.org/desktopgarp/). In Kepler these functions are separated into distinct actors: *GarpPresampleLayers, GarpAlgorithm, GarpPrediction*, and *GarpSummary*. Because of this modularity, the output from the *GarpPreseampleLayers* actor (a text file that includes a code for present/absent, location information, and a set of environmental characteristics at each location point) can easily be reformatted for input to a different algorithm.

A number of statistical algorithms are available in R that could be used instead of the GARP algorithm (see the R documentation for explicit information). If the *GarpAlgorithm* and *GarpPrediction* actors are replaced with R actors, it is also likely that the error analysis portion of the workflow (the *GARPSummary* actor) will have to be replaced because it requires input in a custom format.

If the *GarpPresampleLayers* actor is not used then there is no need for a *.dxl file. The *GarpPresampleLayers* actor is the only actor that requires a *.dxl file. However, a set of environmental layers will still need to be constructed and sampled in some way in order to use any kind of modeling algorithm.

5.4. Changing the output display settings

The ImageJ application, which is used to display the GARP predictions, can be customized to display presence and absence data with different color palettes and color intensities. The default settings display areas with a higher probability of presence with a brighter color. Areas where no prediction can be determined are indicated with white. For more information about using ImageJ to manipulate image data, see the User Manual.

5.5. Running a high performance computing version

Multiple species ENM workflows can very easily be converted to "high performance computing applications" simply by running each species independently on its own processor. Because there is no interaction or dependency between the tasks of predicting the distribution of each species, the workflow is well suited for parallel processing (in fact such workflows are known as “embarrassingly” parallel). A simple script can be written to launch the simultaneous runs on a cluster. Kepler is currently being modified to implement control of parallel processing from within the Kepler environment. This functionality is expected to be available in 2008.

6. References

Araujo, M.B., Cabeza, M., Thuiller, W., Hannah, L., and Williams, P.H. (2004). Would climate change drive species out of reserves? An assessment of existing reserve-selection methods. Global Change Biology 10:1618-1626.

Elith, J. and Burgman, M. (2002). Predictions and their validation: Rare plants in the Central Highlands, Victoria, In: Predicting Species Occurrences: Issues of Scale and Accuracy, (Eds: Scott, J.M. and Heglund, P. J. and Morrison, M. L.), Island Press:Washington, D.C.

Grinnell, J. (1917). Field tests of theories concerning distributional control. American naturalist 51:115-128.

Huntley, B., Bartlein, P.J., and Prentice, I.C. (1989). Climatic control of the distribution and abundance of Beech in Europe and North America. Journal of Biogeography 16:551-560.

Mladenoff, D. J., Sickley, T. A., Haight, R. G. and Wydeven, A. P. (1995). A regional landscape analysis and prediction of favorable gray wolf habitat in the northern Great Lakes region, Conservation Biology 9, 279—294.

Nix, H. A. (2002). A biogeographic analysis of Australian elapid snakes, In: (R. Longmore, ed) Atlas of elapid snakes of Australia, pages 4-15, Canberra:Australian Government Publishing Service.

Pearson, R.G., Dawson, T.P, Berry, P.M., and Harrison, P.A. (2002). Spatial evaluation of climate impact on the envelope of species. Ecological Modelling 154:289-300.

Pennington, D., Higgins, D., Peterson, A.T., Jones, M.B., Ludaescher, B., and Bowers, S., 2007, Ecological Niche Modeling Using the Kepler Workflow System. In: *Workflows for e-Science* (I. Taylor, D. Gannon, E. Deelman, and M. Shields, eds.), Springer-Verlag.

Peterson, A.T. (2003). Predicting the geography of species' invasions via ecological niche modeling. Quarerly Review of Biology 78:419-433.

Peterson, A. T., Ortega-Huerta, M. A., Bartley, J., Sanchez-Cordero, V., Soberon, J., Buddemeier, R. H. and Stockwell, D. R. B., (2002). Future projections for Mexican faunas under global climate change scenarios, Nature 416, 626-629.

Phillips, S. J., Dudik, M., and Schapire, R. E. (2004), A maximum entropy approach to species distribution modeling, In: Proceedings of the International Conference on Machine Learning.

Soberon, J. and Peterson, A. T., 2005, Interpretation of models of fundamental ecological niches and species' distributional areas, Biodiversity Informatics 2:1-10.

Stein, B. R. and Wieczorek, J., (2004). Mammals of the world: MaNIS as an example of data integration in a distributed network environment, Biodiversity Informatics 1, 14-22.

Stockwell, D. and Peters, D. (1999). The GARP modelling system: Problems and solutions to automated spatial prediction, International Journal of Geographical Information Science 13, 143-158.

Thomas, C. D., Cameron, A., Green, R. E., Bakkenes, M., Beaumont, L. J., Collingham, Y. C., Erasmus, B. F. N., Ferreira de Siqueira, M., Grainger, A., Hannah, L., Hughes, L., Huntley, B., Van Jaarsveld, A. S., Midgely, G. E., Miles, L., Ortega-Huerta, M. A., Peterson, A. T., Phillips, O. L., and Williams, S. E. (2004), Extinction risk from climate change, Nature 427, 145-148.