



Getting Started with Kepler

The *Getting Started with Kepler* guide is a tutorial style manual for scientists who want to create and execute scientific workflows.

Table of Contents

1. Introduction
 - What is Kepler?
 - What are scientific workflows?
2. Basic components of Kepler
 - Director and Actors
 - Director and Actor Icons
 - Ports and Port parameter
 - Relations
 - Parameters
3. Downloading and installing Kepler
 - System Requirements
 - Installing on Windows
 - Installing on Macintosh
 - Installing on Linux
4. Basic Operations in Kepler
 - Starting Kepler
 - For Windows and Macintosh platforms
 - For the Linux platform:
 - Opening an existing scientific workflow
 - Example 1: Opening the Lotka-Volterra workflow
 - Running an existing scientific workflow
 - Example 2: Running the Lotka-Volterra existing workflow with default parameters
 - Example 3: Running the Lotka-Volterra existing workflow with adjusted parameters
 - Editing an existing scientific workflow
 - Basic searching in Kepler
 - Searching for available data for use within Kepler
 - Searching for available analytic processes for use within Kepler

- Basic creation of a scientific workflow
- 5. Sample scientific workflows
 - Sample Workflow 1 – 01-SimpleAddition
 - Sample Workflow 2 – 05-LinearRegression.xml
 - Sample Workflow 3 – 06-WebServicesAndDataTransformation.xml
- 6. Appendices
 - A. Technical overview of Ptolemy II – the foundation of Kepler
 - B. Technical overview of Kepler
 - C. Actor Reference

1. Introduction

What is Kepler?

Kepler is a software application for the analysis and modeling of scientific data. Kepler eliminates most of the programming typically required to create executable models by instead creating visual representations of these processes. These representations of models, or “scientific workflows,” visualize the flow of data among discrete analysis and modeling components (Figure 1).

Kepler allows domain scientists to create their own executable scientific workflows by simply dragging and dropping components onto a workflow creation area and connecting the components to construct a specific data flow, creating a visual model of the analytical portion of their research. Kepler represents the overall workflow visually so that it is easy to understand how data flow from one component to another. The resulting workflow can be saved in a text format, emailed to colleagues, and/or published for sharing with colleagues worldwide.

Kepler is based on distributed computing technologies (see Appendices, Section 5.2) that allow scientists to share their data and workflows with other scientists and to use data and analytical workflows from others around the world. Kepler also provides access to a continually expanding, geographically distributed set of data repositories, computing resources, and workflow libraries (e.g., ecological data from field stations, specimen data from museum collections, data from the geosciences, etc.).

What are Scientific Workflows?

Scientific workflows are formal models of the flow of data among processing components. These models reflect the ways that scientists typically work by exporting and importing data among various software packages. Kepler automates these low-level data processing tasks so that scientists can focus instead on the scientific questions of interest.

Scientific workflows can be simple and linear, but more often they are complex and non-linear. Increasingly used in scientific domains such as biology, chemistry, ecology, geology, oceanography, and physics, scientific workflows can be used to process and

share both archived and streaming sensor data, images (medical and satellite), simulation output (global climate change models), and observational data recorded by a scientist.

In general, scientific workflows build upon the strengths of visual models, providing:

- documentation of all aspects of an analysis;
- visual communication of analytical steps;
- ease of testing and debugging;
- reproducibility of a given project with little effort; and
- reuse of part or all of a workflow in a different project.

Scientific workflows feature the following additional benefits:

- integration of multiple computing environments into a single environment (e.g. functionality from different software packages may be accessed and used within a single workflow from a single environment);
- automated access and use of distributed resources (e.g., data, software, or hardware) via web service and grid technologies; and
- system functionality designed to assist with integration of heterogeneous components derived from multiple computing environments and distributed resources.

To date, most scientific workflows have involved a variety of software programs and sophisticated programming language. Kepler builds upon the open-source Ptolemy II visual modeling system (<http://ptolemy.eecs.berkeley.edu/ptolemyII/>), creating a single source for scientists. The result is a user-friendly program that allows scientists to create their own scientific workflows without having to integrate several different software programs or enlist the assistance of computer programmers.

In Kepler, an icon represents each step or workflow component, formally describing input, output, parameters, and processing associated with that step. Workflows may represent theoretical models or observational analyses. In the latter, data entered into the workflow are typically the first component.

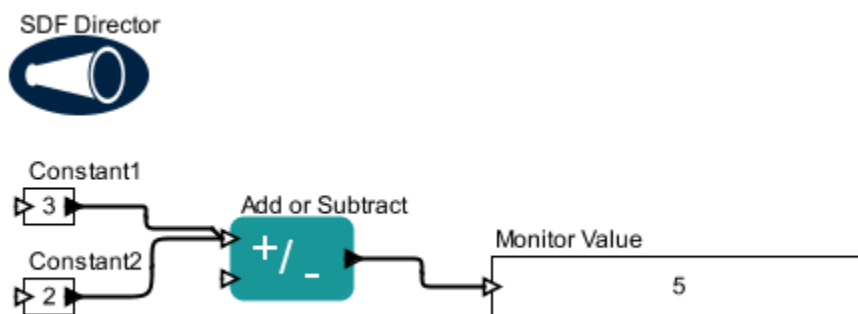


Figure 1: A simple scientific workflow developed in Kepler

Figure 1 illustrates a simple example of a workflow that adds two numbers together and displays the result. The two components labeled *Constant* and *Constant2* (or actors in Kepler terminology) produce constant values that flow to the *Add or Subtract* actor, which subsequently processes these values. The *Add or Subtract* actor calculates the sum of all of the input values, and then produces that sum. The *Display* actor opens a small window and displays the sum, in this case the value 5.

A number of ready-to-use, analytical components come standard with Kepler, including generic mathematical, statistical, and signal processing components and components for data input, manipulation, and display. R- or Matlab-based statistical, image processing, and GIS functionality are available through direct links to these external packages. It is also possible to create new components or wrap existing components from other programs (e.g., C programs) for use within Kepler. For more information about the design of Kepler and Ptolemy, upon which Kepler is based, see the Appendix.

2. Basic Components in Kepler

Director and Actors

Kepler uses a director/actor metaphor to visually represent workflow execution and the various components of a workflow. Execution is represented by a director and the executable components are represented by actors. A director controls (or directs) the execution of a workflow. The actors take their execution instructions from the director. In other words, actors specify *what* processing occurs while the director specifies *when* it occurs.

Every workflow must have a director that controls the execution of the workflow using a particular model of computation. Different models of computation are possible, each represented by its own director. For example, workflow execution can be synchronous, with processing occurring one component at a time in a designated sequence. Alternatively, workflow components can execute in parallel, with one or more components running simultaneously. Kepler represents each model of computation using a different director component. A small set of commonly used directors come pre-packaged with Kepler, but more are available in the underlying Ptolemy II software that can be accessed as needed. For more detailed discussion of workflow models of computation, see Appendices A and B and refer to Ptolemy II documentation.

Kepler identifies two kinds of actors: atomic and composite. Atomic actors represent a single data source or operation. Composite actors are collections or sets of actors bundled together to perform more complex operations within a hierarchical, nested workflow. In essence, an entire workflow can be represented as a composite actor, then be included as a component within a higher-level workflow. In more complex hierarchical workflows, it is possible to have different directors at different levels.

Kepler provides a large set of packaged actors for creating and editing scientific workflows. The current base set includes three types:

- general operational actors or atomic actors (e.g., integral, data transformation, R program);
- model actors (e.g., statistical, probabilistic); and
- display actors (e.g., text, image, graph).

Actors can be added to Kepler for an individual’s exclusive use and/or can be made available to others.

Director and Actor Icons

In Kepler, a single class of icons represents directors, while actors are divided into functional categories, with each category assigned to visually related icons as listed below. For a complete list of actors, including their specific icons and a description of each actor, see Appendix D, “Actor Reference.”







Icon	Name	Description
	Director	Stand-alone component in a scientific workflow that directs the other components (the actors) in their execution
	Atomic Actor	A connected component in a scientific workflow; can be a single data source or an operation upon data
	Display Actor	Shows the output of the workflow in text or graphical format
	Model Actor	Allows the creation and execution of an analytic rule set
	Composite Actor	Collections or sets of actors bundled together to perform more complex operations within a hierarchical, nested workflow.
	Data	Datasets located from the grid containing either data or metadata or both.

Table 1: The major Kepler icons

Ports and Port Parameters

Each actor in a workflow can contain one or more ports used to consume or produce data and communicate with other actors in the workflow. Actors are connected in a workflow via their ports. The link that represents data flow between one actor port and another actor port is called a channel. Ports are categorized into three types:

- input port – for data consumed by the actor;
- output port – for data produced by the actor; and
- input/output port – for data both consumed and produced by the actor.

Each port is configured to be either a “singular” or “multiple” port. A single input port consumes only a single value and can be connected to only a single channel, whereas a multiple input port can take several inputs and therefore can be connected to multiple channels.

Port parameters are used when designing a workflow when it is unclear whether a needed value will be specified through a parameter or passed through a port. A function is incorporated into the actor that couples a parameter with a specific port. The parameter provides a persistent default value that is used if no data are being passed through the port. When the port is active, the parameter value is ignored.

Relations

Relations are used to direct the same input or output to more than one other port. For example, a scientist might wish to direct the output of an operational actor to both another operational actor for further processing, but also to a display actor to visualize the data at that specific reference point.

Parameters

Parameters are configurable values that can be attached to a workflow or to individual directors or actors. For example, the scatter plot display actor allows the user to set the scale of the graph on the X and Y axis, as well as designate titles on the graph. Simulation model actors can be configured to control certain aspects of the simulation, such as initial values. Director parameters control the number of workflow iterations, and the relevant criteria for each iteration.

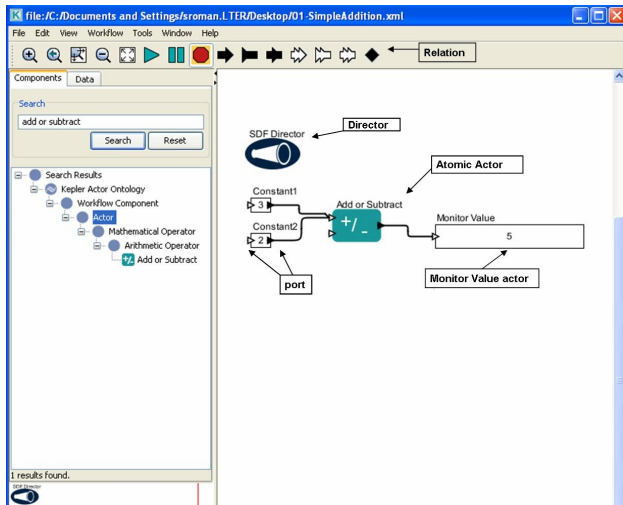


Figure 2: Main window of Kepler with some of the major workflow functions highlighted

The next sections include information on downloading and installing Kepler, and then provide step-by-step examples of how to open, edit, and run different scientific workflows.

3. Downloading and Installing Kepler

Kepler is an open-source, cross-platform software program that can run on Windows, Macintosh, or Linux-based platforms. Kepler can be downloaded from the project web site: <http://kepler-project.org>.

Although Kepler has been designed to be a stable and full-featured application, the software is still a research prototype, and Kepler releases are a continual work in progress. Kepler users are encouraged to suggest improvements for the documentation, new features, new actors and components, etc., as well as notify the designers of bugs and other problems. Community involvement in the on-going development of Kepler has proved valuable because it allows the system to quickly adapt to the needs of practicing scientists. To stay abreast of changes and updates, subscribe to the Kepler users mailing list at <http://mercury.nceas.ucsb.edu/eoinformatics/mailman/listinfo/kepler-users>.

System Requirements

To run Kepler your system will need at least:

- 270 MB (283,688,960 bytes) of disk space
- 128 MB of RAM
- Java 1.4 or higher

To download and install Kepler, follow the instructions for your system. Downloading the installer files may be time consuming depending upon your connection.

NOTE: Java 1.4 or higher is required and can be obtained from Sun's Java website at: <http://java.sun.com/j2se/downloads/> or check with your system administrator. Some installations offer a version that includes Java 1.4 and others do not.

Installing on Windows

Two versions of Kepler are available for installation on a Windows machine: one with Java; and one without Java. If you do not have a Java virtual machine 1.4 (or higher) installed, be sure to download the package that includes Java or download and install from Sun's Java website. Follow these steps to download and install Kepler for Windows:

1. Click on the desired Windows version (with or without Java): [kepler-1.0.0alpha6-jre.exe --- Windows Installer Including Java JRE \(recommended\)](#) or [kepler-1.0.0alpha6.exe --- Windows Installer without JRE](#)
2. Save the executable install file to your computer.
3. Once the file has been saved to your computer, double click on the install file.
4. After the install wizard appears, follow the steps presented to complete the Kepler installation process.
5. Once the installation process is complete, a Kepler application icon will appear on your desktop (*Figure 3*).

Installing on Macintosh

Java is included in all Mac OSX installs and thus can be assumed to be present. Follow these steps to download and install Kepler for Macintosh systems:

- 1 Click on the following link: [kepler-1.0.0alpha6-MacOSX.zip --- MacOSX Installer](#).
- 2 Save the zipped install file to your computer.
- 3 Once the file has been saved to your computer, the zipped install file should automatically begin to extract itself. (If the extraction does not start automatically, manually extract the zip file by double clicking on it.)
- 4 Once the extraction is complete, an install icon (*Figure 3*) will appear on your desktop. Double click the install icon.
- 5 After the install wizard appears, follow the steps presented to complete the Kepler installation process.
- 6 Once the installation process is complete, a Kepler application icon will appear on your desktop (*Figure 3*).

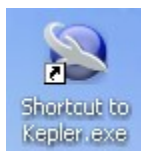


Figure 3: Graphic of install icon

Installing on Linux

After making sure you have Java 1.4 or higher installed, follow these steps to download and install Kepler on Linux:

1. Click on the following link: [kepler-1.0.0alpha6.zip --- Linux Installer](#).
2. Save the zipped install file to your computer.
3. NOTE: This step performs the installation via the command line. Open a shell window and extract the zipped install file to the desired directory. This completes the Kepler installation.

4 Basic Operations in Kepler

This section will cover the basic operations in Kepler: including starting the application, opening and running an existing workflow, and some of the basic functions needed to edit, design and create your own workflows in Kepler.

Starting Kepler

To start Kepler, follow the instructions for your platform.

For Windows and Macintosh platforms:

Double click on the Kepler icon (Figure 3) on the desktop.

For the Linux platform:

1. Open a shell window.
2. Change to the directory into which you installed Kepler.
3. Type `sh ./kepler.sh`.

Two windows will open on your desktop. The first is the Kepler command window where programming instructions executed by the Kepler program may be displayed. You can minimize this window but do not close it or you will exit the Kepler program.

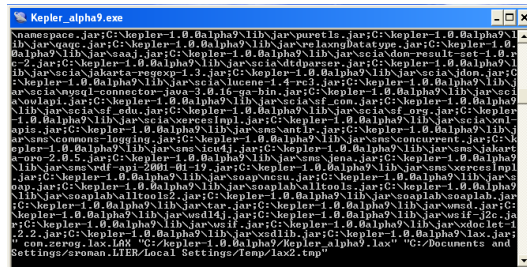


Figure 4: The Kepler command window

NOTE: This window must be kept open while using Kepler. Closing this window will exit the Kepler program.

The second window that will open is the main Kepler application window (Figure 5). From this window you can access and run sample and existing scientific workflows and/or create your own custom scientific workflow. Each time you open an existing workflow or create a new workflow, a new application window will open. This allows you to work on multiple workflows simultaneously and compare, copy, and paste components between workflows.

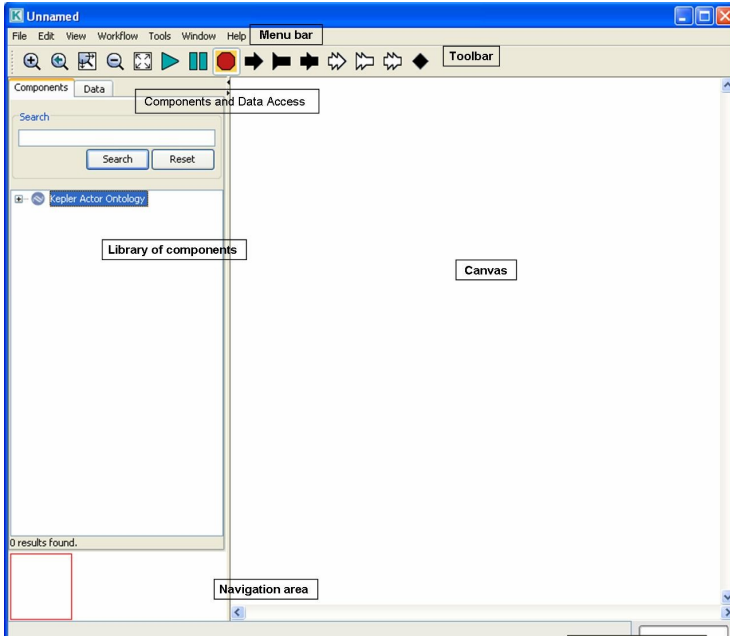


Figure 5: Empty Kepler window with major sections annotated

The major sections of the Kepler application window are:

- Menu bar – provides a list of all available functions within Kepler.
- Toolbar – provides access to the most commonly used functions in Kepler.
- Components and Data Access – contains a components tab and actor tab, both of which contain a search function and display the library of available components and/or search results for data and components.
- Workflow canvas – provides space for displaying and creating workflows.
- Navigation area – allows a portion of a workflow to be displayed if it is too big to fit into the workflow canvas area.

The Kepler toolbar is designed to contain the most commonly used functions (see *Figure 6*).

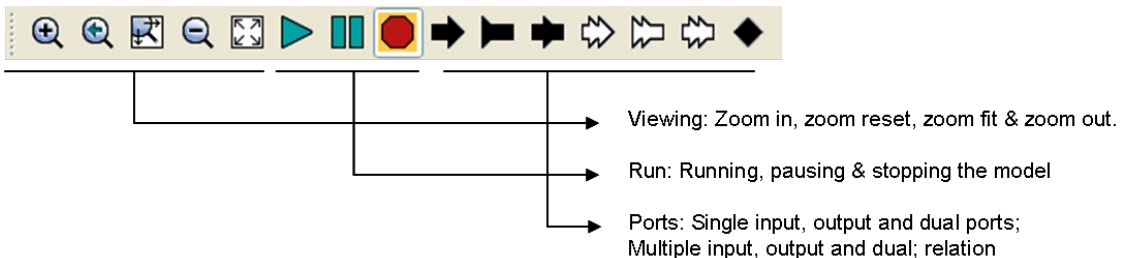


Figure 6: Annotated Kepler toolbar

The main sections of the toolbar include:

- Viewing – allows you to zoom in, reset, fit and zoom out of the model on the Kepler canvas.

- Running – allows you to run, pause and stop the model without opening the run window.
- Ports – allows you to add single (black), multi (white) input and output ports and relations to workflows.

Opening an Existing Scientific Workflow

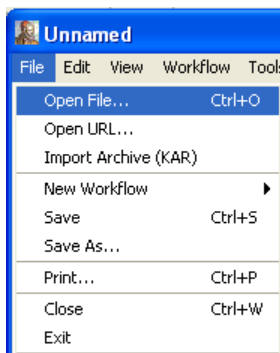
This section will walk you through the steps of opening a workflow in Kepler. We begin the generic steps and then specify the steps to open a pre-existing workflow in Kepler. To open any existing workflow:

1. From the menu bar select *File*, then *Open File*. A standard file dialog box will appear.
2. If the file dialog box does not open to the /kepler directory, then navigate to the /kepler directory, then to the /workflows directory and locate the desired workflow, or navigate the workflow subdirectories until you locate the desired workflow.
3. Double click on the desired workflow to open it. The workflow will appear in the workflow canvas area of the application window.

Example 1: Opening the Lotka-Volterra workflow

In this example we will open a specific workflow: the classic predator pray model the Lotka-Volterra workflow. To open this workflow:

1. From the menu bar select *File*, then *Open File*. A standard file dialog box will appear (*Figure 7*).
2. Navigate to the *kepler/demos/getting-started* directory and locate the file named *02-LotkaVolterraPredatorPrey.xml* (*Figure 7*).
3. Double click on the file named *02-LotkaVolterraPredatorPrey.xml*. The Lotka-Volterra workflow appears in the workflow canvas area of the application window (*Figure 9*).



This graphic needs to be added once we add these workflows

Figure 7: Graphic showing navigation to Lotka Volterra workflow with highlight of file selected.

Running an Existing Scientific Workflow

This section will walk through the steps of running an existing workflow.

To run any existing scientific workflow:

1. Open the desired workflow.
2. From the toolbar select the *Run* button.
3. The workflow will execute and produce whatever output was specified.

OR

1. Open the desired workflow.
2. From the menu bar, select *Workflow*, then *Run Window*. A “Run Window” dialog box will appear.
3. If the workflow has parameters, they will appear in the “Run Window” dialog box. If desired, adjust any parameters and then click on the *Run* button.
4. The workflow will execute and produce whatever output was specified.
5. If desired during workflow execution, select the *Pause*, *Resume* or *Stop* buttons.

NOTE: The ability to pause, resume or stop the workflow execution is not available when selecting the Run button or Run menu item from the workflow menu. Additionally, the option to adjust workflow parameters, if they exist, will not be presented when selecting the Run button. The workflow will execute with the default parameter values.

Example 2: Running the Lotka-Volterra existing workflow with default parameters

This model uses the continuous time domain in Kepler to solve two coupled differential equations: one that models the predator population; and one that models the prey population. The results are plotted as they are calculated showing both populations change and a phase diagram of the dynamics. To run the Lotka-Volterra workflow:

1. Open the workflow file named *LVPredPrey.xml*.
2. From the menu bar, select *Run*.
3. The Lotka-Volterra workflow will execute with the default parameters and produce two graphs. The graph labeled *TimedPlotter* depicts the interaction of predator and prey over time (i.e., the cyclical changes of the predator and prey populations over time predicted by the model). The graph labeled *XYPlotter* depicts a phase portrait or the population cycle around the equilibrium (i.e., the predator population against the prey population). Together these graphs show how the predator and prey populations are linked: as prey increases, the number of predators increase. (*Figure 8*)

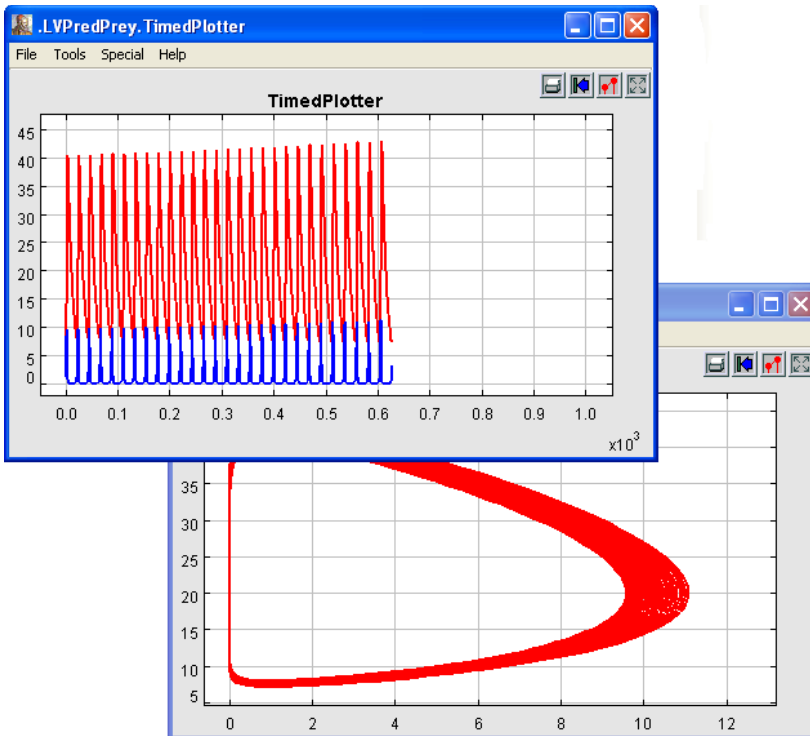


Figure 8: Graphics from running the Lotka-Volterra model

Example 3: Running the Lotka-Volterra existing workflow with adjusted parameters

Before running the Lotka-Volterra workflow with adjusted parameters it is useful to provide some background and a scenario.

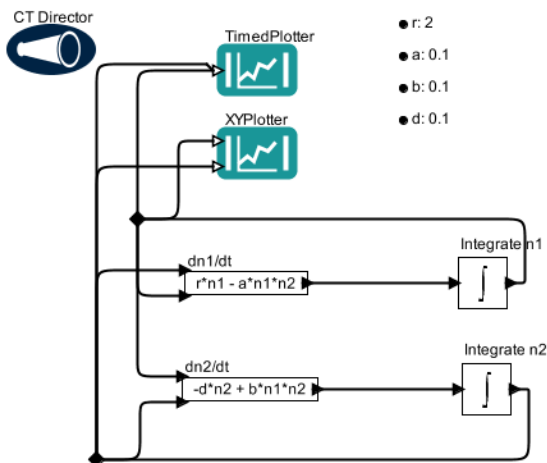


Figure 9: Graphic of Lotka-Volterra workflow

The model was developed independently by Lotka (1925) and Volterra (1926) and is made up of two differential equations. One describes how the prey population changes

($dn1/dt = r*n1 - a*n1*n2$), and the second equation describes how the predator population changes ($dn2/dt = -d*n2 + b*n1*n2$).

The Lotka-Volterra model is based on certain assumptions:

- the prey has unlimited resources;
- the prey's only threat is the predator;
- the predator is a specialist (i.e., the predator's only food supply is the prey); and
- the predator's growth depends on the prey it catches.

The Lotka-Volterra model as represented in Kepler as a scientific workflow contains:

- six actors - two plotters, two equations, and two integral functions;
- one director; and
- four workflow parameters.

NOTE: The director has several configurable parameters as do the two plotter actors.

The critical assumptions above provide the basis for the workflow parameters. The workflow parameters and their defaults are as follows:

Parameter	Default Value	Description
R	2	the intrinsic rate of growth of prey in the absence of predation
A	0.1	capture efficiency of a predator or death rate of prey due to predation
B	0.1	proportion of consumed prey biomass converted into predator biomass (efficiency of turning prey into new predators)
D	0.1	death rate of the predator

Table 2: Description of the default parameters for the Lotka-Volterra workflow

In the differential equations used in the workflow, the variable n1 represents prey density, and the variable n2 represents predator density.

When changing parameters in a workflow, the assumptions of the model must be kept in mind. For example, if creating a Lotka-Volterra model with rabbits as prey and foxes as predators, the following assumptions can be made with regard to how the rabbit population changes in response to fox population behavior:

- the rabbit population grows exponentially unless it is controlled by a predator;
- rabbit mortality is determined by fox predation;
- foxes eat rabbits at a rate proportional to the number of encounters;
- the fox population growth rate is determined by the number of rabbits they eat and their efficiency of converting the eaten rabbits into new baby foxes; and

- fox mortality is determined by natural processes.

If you think of each run of the model in terms of rates at which these processes would occur, then you can think of changing the parameters in terms of percent of change over time.

To run the Lotka-Volterra workflow with adjusted parameters:

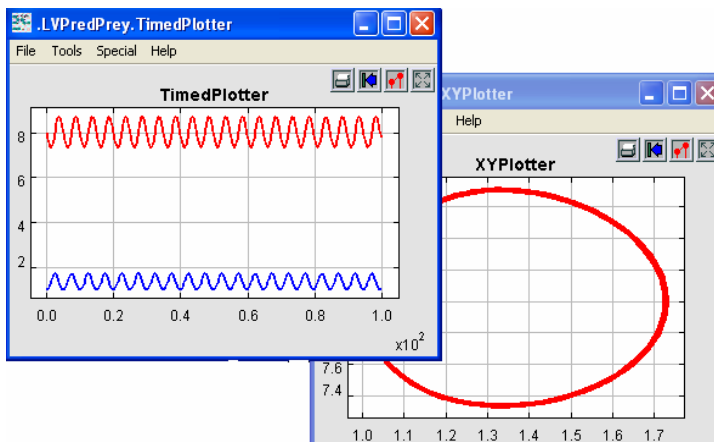
1. Open the workflow file named *LVPredPrey.xml*.
2. From the menu bar, select *Run Window*. The *Run Window* will appear.
3. Notice there are two sets of parameters – one for the workflow and a set for the director. In this example, you will make adjustments to both sets of parameters.
4. Adjust the workflow parameters as listed in Table 3.

Parameter	Value	Description
R	4	the intrinsic rate of growth of prey in the absence of predation
A	0.05	capture efficiency of a predator or death rate of prey due to predation
B	0.03	proportion of consumed prey biomass converted into predator biomass (efficiency of turning prey into new predators)
D	0.04	death rate of the predator

Table 3: Description of the suggested parameters for the Lotka-Volterra workflow

5. Adjust the value of the *stopTime* director parameter to 100.
6. In the Run window, click the *Run* button.

The Lotka-Volterra workflow will execute with the adjusted parameters and produce the two graphs: 1) the TimedPlotter graph and 2) the XYPlotter graph. Note that with the changes in the parameters, the relationship between the predator and prey populations are still linked but the relationship has changed.



10: Graphs showing the run of the Lotka Volterra model with adjusted parameters

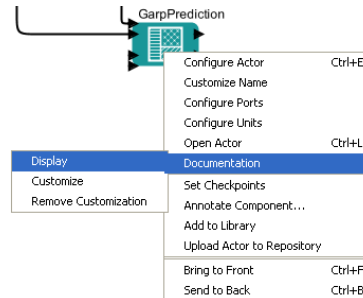
Editing an Existing Scientific Workflow

There are two ways to edit an existing scientific workflow:

- substitute a different data set for the current data set; or
- substitute one or more analytical processes in the workflow with other analytical processes (e.g., substitute a neural network model actor for a probabilistic model actor).

Whether substituting data or processes, it is important to understand the required inputs and outputs of the actors involved.

NOTE: To see a high-level description of an actor, right click on that actor, and then select the Documentation menu item (Figure 11). A dialog box will appear which describes the main function of an actor and its required inputs and output. When finished with this dialog, click the OK



button and it will disappear.

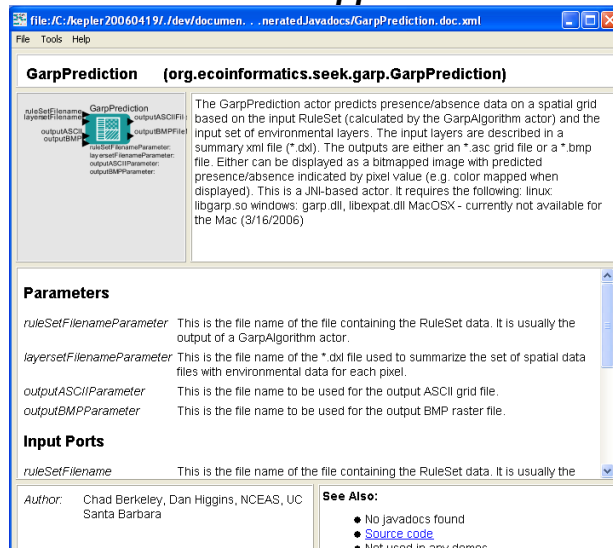


Figure 11: Graphic showing how to get documentation and the subsequent display

To edit an existing scientific workflow:

1. Open the desired workflow.
2. Identify which workflow component is the target for substitution.
3. Select the target component (data actor or processing actor). The component will be highlighted in a thick orange border, indicating it is selected.

4. Press the *Delete* key on your keyboard. The previously highlighted component will disappear from the workflow canvas.
5. From the components area on the left, drag either an appropriate data file or processing actor to the workflow canvas.
6. Connect the appropriate input and output ports.
7. Run the workflow.
8. To save the workflow, go to the menu bar and select either *Save* (to save over the existing workflow) or *Save As* (to save as a new workflow). If using the *Save As* option, enter a new workflow name when prompted.

Example 4: Editing/substituting analytical processes in the Image J Workflow

In this example, we are going to show you how two different actors can perform the same function. In this example we are going to work with the Image Display workflow found in that come standard with Kepler and we will substitute the *Browser* actor for the *Image J* actor to get the same output. The image we are using is that of a species distribution of the species *Mephitis* throughout North and South America from the GARP (Genetic Algorithm for Rule-set Production). GARP is a genetic algorithm that creates an ecological niche model for a species that represents the environmental conditions where that species would be able to maintain populations. For more information on GARP go to <http://www.lifemapper.org/desktopgarp/> . GARP was originally developed by David Stockwell, at the [San Diego Supercomputer Center](#). The 03-Image-Display.xml workflow uses the *ImageJ* actor to show the distribution. To edit the Image Display workflow:

1. Open 03-Image-Display.xml.
2. Identify which workflow component is the target for substitution.
3. Select the target component (data actor or processing actor). The component will be highlighted in a thick yellow border, indicating it is selected (e.g., the *Image J* actor).

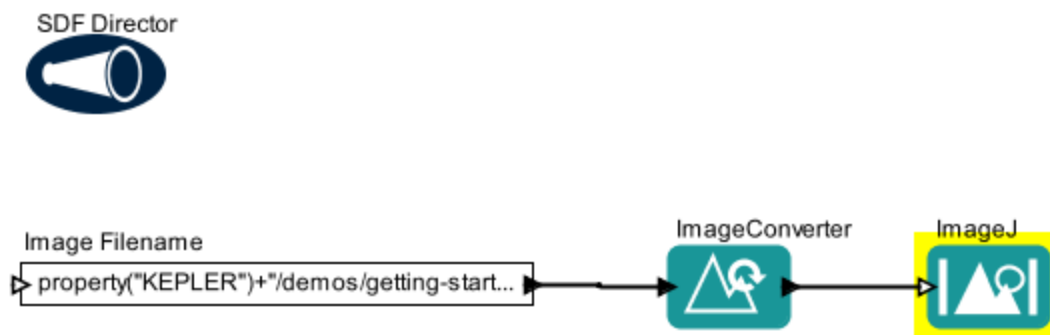


Figure 12: Image Display workflow showing Image J actor highlighted

4. Press the *Delete* key on your keyboard. The previously highlighted component will disappear from the workflow canvas.
5. From the components area on the left, drag either an appropriate data file or processing actor to the workflow canvas (e.g., *Browser Display*).

6. Connect the appropriate ports (the output port of the *Image Converter* actor to the input port of the *Browser Display* actor). To do this left click and hold on the output port (black triangle) on the right side of the *Image Converter* actor, drag the pointer to the upper input port on the left side of the *Browser Display* actor, and then release the mouse. If the connection is made you'll see a thick black line. If the connection is not completely made, the line will be thin.
7. Run the workflow.
8. If desired, save the workflow. From the menu bar, select either *Save* (to save over the existing workflow) or *Save As* (to save as a new workflow). If using the *Save As* option, enter a new workflow name when prompted.

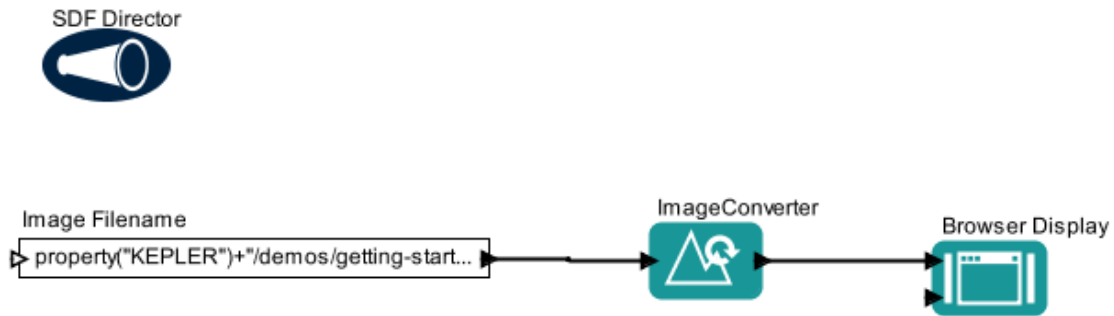


Figure 13: The *Image Display* workflow with the *Browser* actor substituted for the *ImageJ* actor.

NOTE: Sometimes the easiest way to connect actors is to go from the output port to the input port.

Basic Searching in Kepler

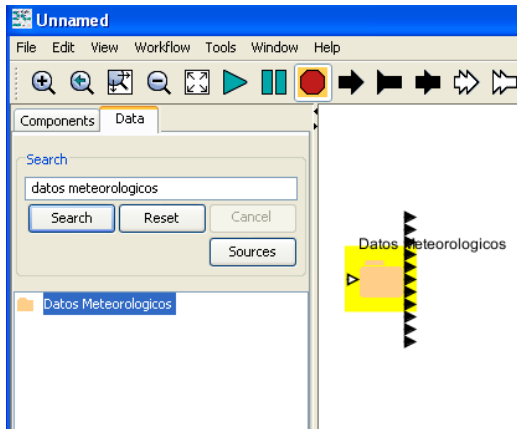
Kepler provides a searching mechanism to locate data and analytical processing components both locally and remotely (on the Grid). All sources, local and remote, will be searched unless otherwise configured. The examples given in this section describe searching for data and components in Kepler.

Searching for available data for use within Kepler

Kepler provides access to search for data from the EcoGrid through resources stored on the KNB Metacat <http://knb.ecoinformatics.org>, the KU Digir <http://www.specifysoftware.org/Informatics/informaticsdigir/> and the GEON <http://www.geongrid.org/> databases. To search for data on the EcoGrid through Kepler:

1. In the Components and Data Access area to the left of the workflow canvas, select the *Data* tab.
2. Type in the desired search string (e.g., Datos Meterologicos).
3. Click on the *Search* button.
4. When the search is complete, a list of search results will be displayed in the area below the Search group box.

5. To use one or more data actors in a workflow, simply drag the desired files to the workflow canvas.
6. Information about the data can be revealed in two ways: roll over the Data output ports tool tip to show the name and type of data; Left click on the data package to get metadata.



14: Graph of searching for and locating Datos Meteorologicos

NOTE: Configuration of sources to be searched and the documents retrieved can be done by clicking on the Source button and selecting the desired sources to be searched.

Searching for available analytical processes for use within Kepler

Kepler comes standard with over 200 workflow components and the ability to modify and create your own. This allows you to create an innumerable number of workflows with a variety of analytic functions. The available default set of Kepler processing components is displayed below the search box, and is organized by function. To search for processing components:

1. In the *Components and Data Access* area to the left of the workflow canvas, select the *Components* tab.
2. Type in the desired search string.
3. Click on the *Search* button.
4. When the search is complete, a list of search results will be displayed in the area below the Search group box. This replaces the default list of components organized by function. If you would like to clear the search results and re-display the default list, click on the *Reset* button.
5. To use one or more processing actors in a workflow, simply drag the desired components to the workflow canvas.

NOTE: If you know which component you want to use and its location, you can navigate to it within the tree, and then drag it to the workflow canvas for use in a workflow.

Basic Creation of a Scientific Workflow

One of the strengths of Kepler is the ability to design, create and save your own executable workflows. The general steps in creating a workflow are as follows:

1. Create a conceptual (paper or other medium) model of your scientific workflow.
2. Open the Kepler application.
3. Map the data and actor components available in Kepler to your conceptual model.
4. Select a director for your workflow and drag it to the workflow canvas.
5. Drag the desired data and actor components to the workflow canvas.
6. Connect the data and actor components.
7. Save the workflow.

The examples given in this section will show you how to begin to create your own workflows. The first workflow is the classic “Hello World” example that demonstrates how easy it is to create a functioning workflow in Kepler. The second example is more practical and shows you how to use your own desktop data in a workflow.

Example 1: Creating a Hello World workflow:

To create this model, begin by thinking about the type data being used (e.g., text or string data); the type of output desired (e.g., textual or image display); and the type of director needed to execute this model (e.g., in a SDF director, the data will flow through the actors based on the order in the model, and will run continuously). The Hello World model requires a constant actor, a text display actor, and a SDF director.

1. Open Kepler. A blank workflow canvas will open.
2. In the Components and Data Access area, select the *Components tab*, then navigate to the *Workflow Components/Directors* directory.
3. Drag the director titled *SDF Directory* to the top of the workflow canvas.
4. In the Components tab, type *Constant* in the Search box, then click the *Search button*.
5. After the search results are displayed, drag the actor titled *Constant* onto the workflow canvas a little ways below the *SDF Director*.
6. Configure the *Constant* actor by right clicking on the actor and go to *Configure Actor*

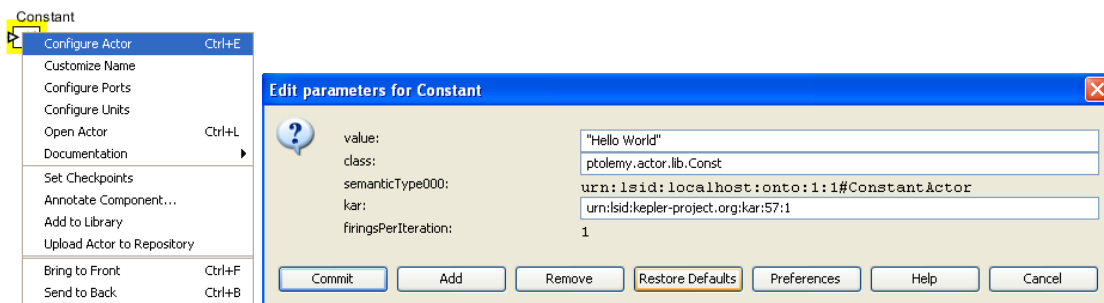


Figure 15: Configuring the Constant actor

7. Type in “Hello World” in the value dialogue box that opens in the *Edit parameters for Constant* window (Figure 15).
8. Search for the *Text Display* actor and drag and drop it to the screen.
9. Connect the output port of the *Constant* actor to the input port of the *Text Display* actor.
10. Run the model (Figure 16).

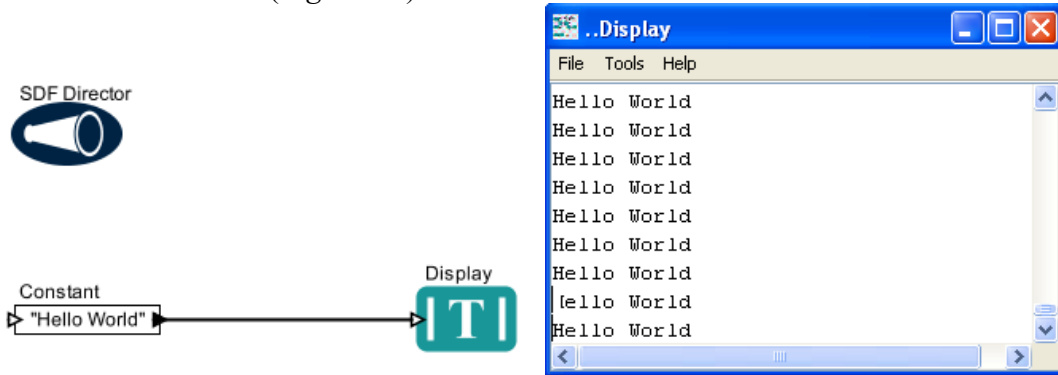


Figure 16: Hello World workflow and its output

NOTE: The SDF director handles regular computations that operate on streams. If not configured, the SDF director will continuously run the workflow, creating a loop. To have “Hello World” run a limited number of times, configure the SDF director iterations to the appropriate number of runs.

Example 2: Creating a simple workflow in Kepler using local data

In this example, you will create a simple workflow using an actor that will read a local data file, extract the data from a column within that file, and then calculate count averages for each of the species. In the creation of a workflow, reviewing the data is the first step (i.e., the limitations of the data determine which analytic methods are appropriate).

There are many ways to read a file in Kepler. One is to review the data table and another is to use an actor to take in the data file. In this example we will use an actor to review the data table. To determine which actor is appropriate, start with the format in which the data are saved. In this example, the data are saved in a text format. As such we will use the File to String Converter to view the data in a tabular format.

1. This model requires two actors: a *File to String Converter* actor and a *Text Display* actor, plus a *SDF Director*. If you have a blank workflow canvas open, go to step 2. If not, from the menu bar, select *File*, then *New Workflow*, and then *Blank*. A new workflow window will appear with a blank canvas.
2. In the Components and Data Access area, select the *Components* tab, and then navigate to the *Workflow Components/Directors* directory.
3. Drag the director titled *SDF Director* to the top of the workflow canvas.

4. In the Components tab, type “File to String Converter” in the Search box, then click the *Search button*.
5. After the search results are displayed, drag the actor titled *File to String Converter* onto the workflow canvas a little ways below the *SDF Director*.
6. Right click on the *File to String Converter* actor to display the Configure dialog.
7. Select the *Parameters tab*.
8. Click on the *Browse button* next to the *fileOrURL* parameter and navigate to the following file: *mollusc_abundance.txt*. These data come installed in Kepler and are located in the R folder.

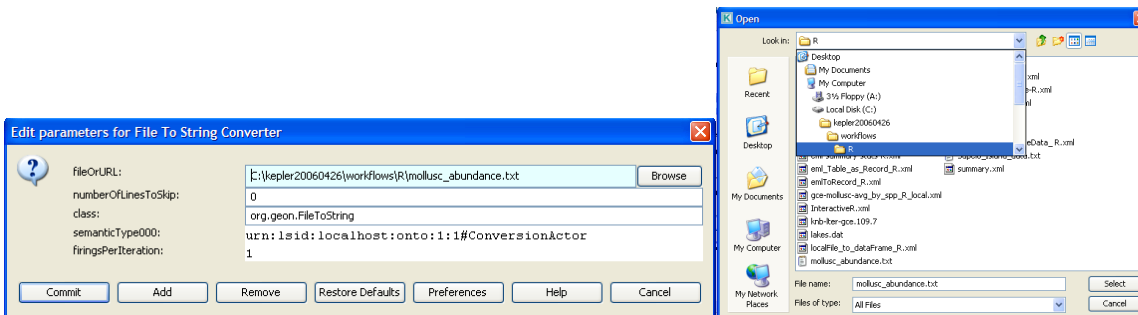


Figure 17: Locating desktop data in Kepler

9. Click on the *Commit button* at the bottom of the *Edit Parameters for File to String Converter* dialog box. The *File Reader* actor is now configured to read the specified file.
10. In the Components tab, navigate to the following directory: *Workflow Components/Outputs/Textual*. Locate the *Display* actor and drag in onto the workflow canvas to the right of the *File Reader* actor.
11. Connect the output port of the *File Reader* actor to the input port of the *Display* actor.
12. From the toolbar, select the *Run button*. A pop up window will appear, displaying the contents of the file in tabular format.
13. From the menu bar, select *File*, then *Save* and when prompted, save the newly created workflow to the workflows directory with the name “*reading data xml*.”

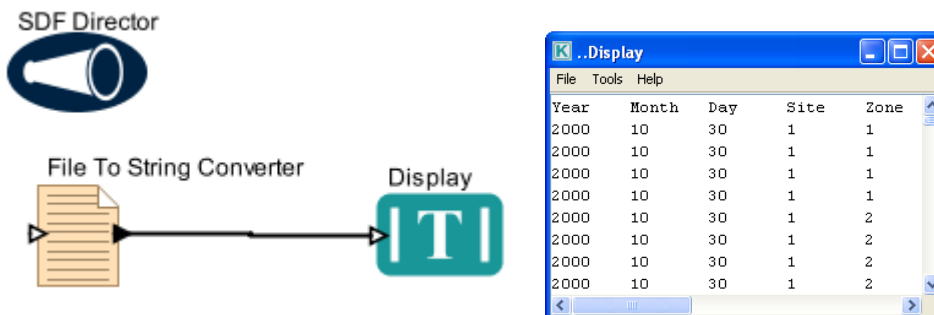


Figure 18: Working with desktop data workflow

5 Sample Scientific Workflows

In this section a small set of sample scientific workflows that come standard with Kepler are explained and step by step instructions of how to create these workflows are provided.

Sample Workflow 1 – 01-SimpleAddition

Name	Simple Addition Workflow
Filename	/kepler/demo/getting-started/01-SimpleAddition.xml
Detailed Description	This workflow adds two numbers together and displays the result. The Constant actor simply produces a constant value that is sent along the connection to the 'Add or Subtract' actor, which subsequently processes these values. The 'Add or Subtract' actor calculates the sum of all of the input values, and then produces that sum on its output port, which is connected to the 'Monitor Value' actor. The Monitor Value actor displays the sum, in this case the value '5'. The overall workflow represents the analysis visually so that it is easy to understand how data flows from one component to another.
Assumptions	The constant actor has numeric values
Director	SDF Director
Data	Data is generated in the constant actor
Actors	Constant, Add or Subtract, Monitor Value
Parameters	For the Constant actor: 2 and 3

This simple workflow adds two numbers together and displays the results. To create this workflow:

1. In the Components and Data Access area, select the *Components tab*.
2. Search for the *SDF Director* and drag and drop that to the screen.
3. Search for the *Constant* actor and drag and drop that to the screen twice.
4. Note the second actor is named *Constant2*.
5. Configure the *Constant* actor by left clicking on the actor and going to *Configure Actor*. At the *Edit Parameters for Constant* window, add 2 and 3 to the value dialogue box of each actor.
6. Search for the *Add or Subtract* actor, and drag and drop that to the screen. Configure this to calculate the sum of all of the input values.
7. Connect the two *Constant* actors to the *Add or Subtract* actor.
8. Search for the *Monitor Value* actor, and drag and drop that to the screen.
9. Connect the output of the *Add or Subtract* actor to the *Monitor Value* actor.

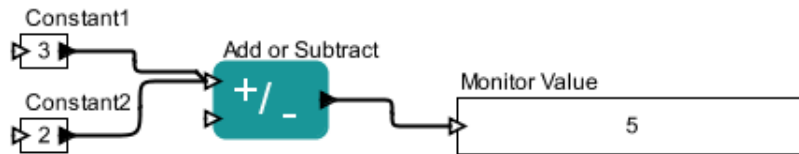


Figure 19: The $x+y$ workflow and its output

Sample Workflow 2 – 05-LinearRegression.xml

Name	Simple Linear Regression workflow using R
Filename	/kepler/demos/getting-started/LinearRegression.xml
Detailed Description	This is an example of carrying out a simple linear regression analysis using the RExpression actor. This workflow creates a scatter platter of the two variables from the Datos Meteorologicos data set and adds a regression line using the $Y = a + bX$ equation, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b , and a is the intercept (the value of y when $x = 0$).
Assumptions	A Linear Regression assumes linearity, independence, homoscedasticity and normality
Director	SDF director
Data	Datos Meteorologicos
Actors	R, Display, ImageJ
Parameters	Datos Meteorologicos: Data Output Format = As Column Vector SDF: iterations = 1; RExpression: script = res <- lm(BARO ~ T_AIR) res plot(T_AIR, BARO) abline(res); RExpression: input ports = 'T_AIR' and 'BARO.'

Linear Regression Example

The *RExpression Actor* has been created for inserting R commands and scripts into Kepler workflows. This actor makes it easy to integrate the powerful data manipulation and statistical functions of R into workflows. The actor is found under the Mathematical Expressions node in the actor tree. To implement the *RExpression Actor*, R must be installed on the computer running the Kepler application.

A very common use of R is to perform a Linear Regression. In this workflow, a search for data on the EcoGrid is run and the data found is used to create a workflow conducting a linear regression. In this example, the input data comes from two output ports (the data columns on Barometric Pressure and Air Temperature) of the *Datos Meteorologicos*, a data set of meteorological data from the La Hechicera station collected in 2001, instead of expression actors.

If you have problems creating this workflow, a stored version comes standard with Kepler at kepler/demos/getting-started/LinearRegression.xml.

This workflow is created from four actors: the *Datos Meteorologicos*, the *RExpression Actor*, the *ImageJ Actor* and the *Display Actor*; and the *SDF director*.

To create this sample linear regression workflow:

1. We start with a search for *Datos Meteorologicos* on the *EcoGrid*.
2. Switch the tab (in the left hand search window) from *Components* to *Data*.
3. Limit the search by clicking on the *Sources* button under the *Data* tab. *Datos Meteorologicos* is stored on the KNB Metacat, so the data source search can be limited to just those nodes on the grid.
4. Click on the green check marks in the boxes across from KU Digir EcoGrid QueryInterface and the GEON Search QueryInterface.
5. Click *Ok* to confirm and store the changes.
6. Type *Datos* in the search box and click on search.
7. Click on the yellow folder named *Datos Meteorologicos*. Drag and drop that to the screen.

NOTE: To find out more information about the dataset, left click on the folder and choose **Get Metadata**. Depending upon the amount of information entered by the provider, much valuable metadata can be obtained. For this particular data set, the **Attribute Name** allows the data to be read and incorporated into our R script. The definition of the **Attribute Name** allows you to choose which data to use, and the **type or value and measurement type** allows you decide which statistical models are appropriate to run.

Figure 19: Locating Metadata

5. Configure the *Data Output Format* of *Datos Meteorologicos* to *As Column Vector* so that the data type matches that of the R actor.
6. Left click on the actor and go to *Configure actor*.
7. At the edit parameters window, use the pull down menu and go to *As Column Vector*.
8. Click *Commit* for the change to take effect.

Figure 20: Configuring Datos Meteorologicos

NOTE: Datos Meteorologicos has a series of output ports. To locate the appropriate port mouse over the output ports to reveal the port tips.

Figure 21: Locating data ports

7. To finish creating the workflow, the *SDF Director* and remaining actors need to be added. Click on the *Components* tab.
8. Locate the *SDF Director* and drag and drop that to the screen.
9. Configure the *SDF Director* by left clicking on the actor. At the edit parameters window that opens, change the number of iteration to 1.
10. Click *Add* and then *Commit* for the changes to take effect.
11. Next, locate the *RExpression Actor* and drag and drop that to the screen. The *RExpression Actor* is located in the *Mathematical Operator* folder.
12. The *RExpression Actor* is configured with two output ports and the command 2+2. You need to add two input ports (T_AIR and BARO) and reconfigure the *RExpression* script.
13. Left click on the *RExpression Actor* and go to configure ports.
14. In the dialogue box that appears, click on *Add* twice.
15. Add two input ports with the names “T_AIR” and “BARO” (the names of the data). Move your cursor to the blank boxes that appear under the Name column and click in the appropriate box under the Show Name column. As you add input ports, click on the appropriate boxes under the Input column.

16. To configure the R script, left click on the actor and choose configure actor. In the dialogue box that appears, change the 2+2 command to the following:

```
res <- lm(BARO ~ T_AIR)

res

plot(T_AIR, BARO)

abline(res)
```

17. Next, add the text *Display Actor*. The text *Display Actor* is located under the components tab.
18. Connect the lower single output port of the *RExpression Actor* to the input port on the *Display Actor*.
19. Next locate the *ImageJ Actor*, drag and drop that to the screen, and connect the *RExpression Actor* to it. The *ImageJ Actor* is located under the components tab in the *External Reader Actor* folder.
20. Connect the upper single *output* port of the *RExpression Actor* to the *input* port on the *ImageJ Actor*.
21. You are now ready to run the workflow. The resulting workflow and graphic output are shown below.

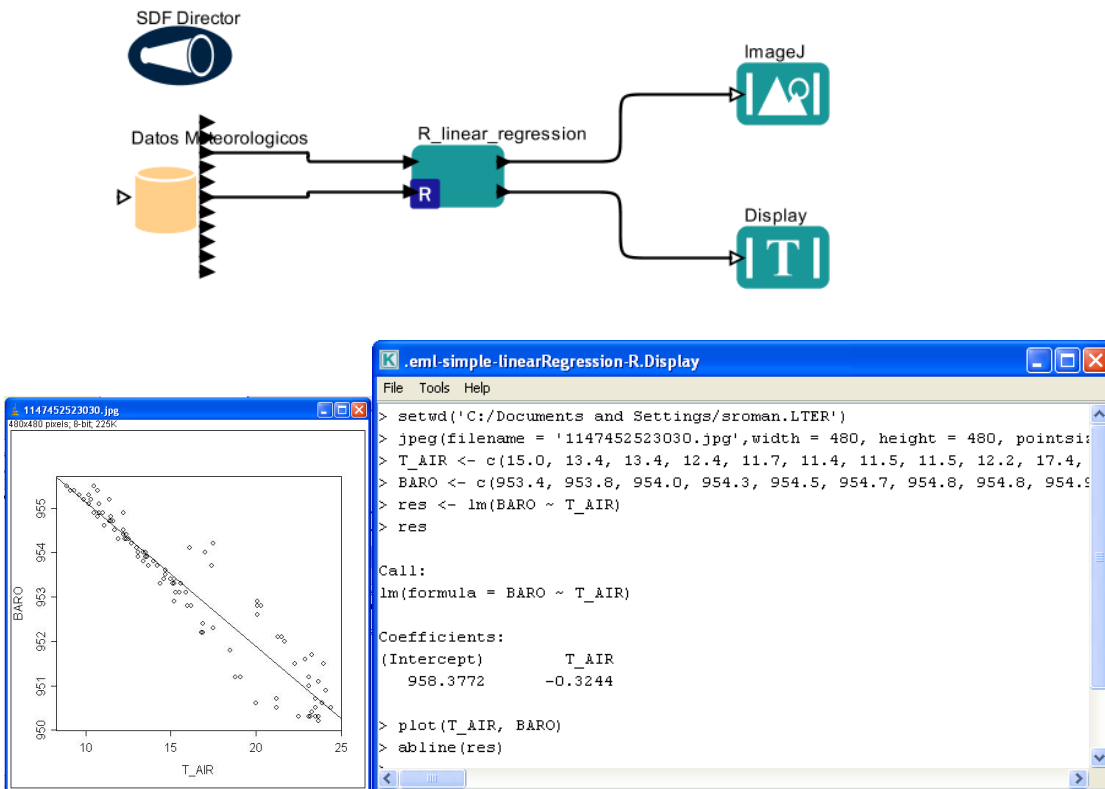


Figure 22: Linear Regression workflow and its output

In the left hand display window, the scatter plot of Barometric pressure to Air Temperature with a regression line is displayed. The graph shows a strong negative relationship between the two: as air temperature lowers, the Barometric pressure rises. In the right hand display, the data for Barometric Pressure and Air Temperature is graphed in the scatter plot. Additionally, the intercept on the Y-axis which 958.38 Barometric Pressure and the slope -0.32 for the linear regression equation $y=mx+b$ is represented.

The data type run through the equations can be changed or you can change the data set in total. When changing the data, remember to make sure that the data being used meets the assumptions mentioned above.

Sample Workflow 3 – 06-WebServicesAndDataTransformation.xml

Name	Webservices and Data Transformation Workflow
Filename	WebServicesAndDataTransformation.xml
Detailed Description	In this workflow a XXXXXXXXXXXX
Assumptions	Describe any assumptions the model is based on
Director	SDF Director
Data	The Data in this workflow is generated
Actors	Text Display, Typed composite actor, web services, Browser UI, File Writer,
Parameters	List of parameters used in the workflow, may include workflow, director, actor or port parameters

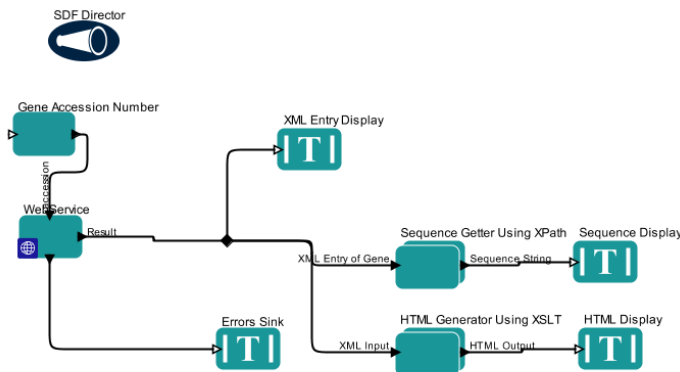


Figure 22: The Webservices workflow

Provide here any discussion about substituting different data or different processing components and any discussion about assumptions if needed, and/or about setting or configuring any parameters etc.

6 Appendix

A. Ptolemy II – The Foundation of Kepler

Ptolemy II is a software framework for heterogeneous, concurrent modeling and design, with a Java-based component assembly framework using a graphical interface called Vergil. The Ptolemy II software is a product of the Ptolemy project at the University of California at Berkeley, a project whose goal is “the use of well-defined models of computation that govern the interactions between components.”

As explained at the project’s website, Ptolemy II includes a number of *domains*, each of which realizes a model of computation. It also includes a component library and a number of support packages such as graphing, mathematics, plot, and data packages. For more information about Ptolemy II, see <http://ptolemy.eecs.berkeley.edu/index.html>.

Although not originally intended for scientific workflows, Ptolemy II provides support for dataflow-oriented models, which is a very important characteristic of scientific workflows. Because Ptolemy II provides an open-source, mature platform for model design and execution, including various models of computation, and is well documented and easily extensible, it was chosen as the foundation for Kepler.

B. Technical Overview of Kepler

Kepler is an open-source software tool for scientists to design and execute scientific workflows efficiently using emerging Grid-based approaches to distributed computation. Kepler is based on the Ptolemy II software and inherits much of its functionality, with particular emphasis on:

- model design and execution;
- synchronous dataflow computational model;
- process networks computational model;
- extensibility; and
- polymorphism.

From a technical perspective, the Kepler software adds:

- Web services;
- grid-based services;
- MoML (modeling markup language – an XML description of a Kepler workflow);
- ontology based processing (under development); and
- semantic-type checking (underdevelopment).

Kepler provides access to local and remote data through web and grid services. It provides a component that instantiates itself to perform remote calls as if they were running on the local machine. Kepler also offers a large library of Java-based components to display, manipulate and process data as it moves through execution in a

scientific workflow. Programmers can “wrap” external functionality in a Java wrapper so that it can be added to the component library. Programmers can also create new components using Java that can be used in Kepler.

Kepler uses a director/actor metaphor. Models of computation (directors) coordinate and direct the execution of the components (actors) of a scientific workflow. Actors can represent data or processing components.

Directors

A Director governs the execution within a Composite Actor. A composite actor that contains a director is said to be *opaque*, and the execution model within the composite actor is determined by the contained director. This director is called the *local director* of a composite actor. A composite actor is also aware of the director of its container, which is referred to as its *executive director*. A director may also be contained by a Composite Entity that is not a Composite Actor, in which case it acts like any other entity within that composite.

Actors

Actors are re-useable components that execute a variety of functions and communicate with other actors in a workflow. Actors communicate through ports. Messages (input/output data) are encapsulated in tokens. Messages are sent through ports. An Atomic Actor is an executable entity that cannot itself contain other actors. The Ports of Atomic Actors are constrained to be IOPorts. Derived classes may further constrain the ports by overriding the public method `newPort()` to create a port of the appropriate subclass, and the protected method `_addPort()` to throw an exception if its argument is a port that is not of the appropriate subclass. In this base class, the actor does nothing in the action methods.

A Composite Actor is an aggregation of actors. It may have a *local director*, which is an attribute of class `director` that is responsible for executing the contained actors. At the top level of a hierarchy, a composite actor (the top level Composite Actor of the topology) will normally exist with a local director, and no container. A composite actor at a lower level of the hierarchy may also have a local director. A composite actor with a local director is *opaque*, and serves the role of the *wormhole* from Ptolemy Classic. Its ports are opaque, but it can contain actors and relations. The top-level composite actor is also associated with a manager object that is responsible for managing any execution within the topology at a high level

Parameters

Parameters are named values that can be attached to a workflow or to individual directors or actors. When you select the "Configure Actor" menu (or double click on the actor) an "Edit Parameters" dialog appears containing parameters for the actor that can be specified by entering a value. Parameters can also be added directly to a workflow. The name and

value can be specified, and the name can then be used elsewhere in the workflow. For example, one could create a parameter named 'Count' with a value of '10'. Then in any actors on the workspace that need a count, one can enter the name 'Count' rather than the explicit value of '10'. This is particularly useful when one wants to change a parameter value. As the change needs to be made in only one place.

Parameters can thus be used as a secondary method for providing data to actors in a workflow. They may be thought of as 'global variables'; i.e. data that does not travel through the ports. Parameters can also be used in algebraic expressions as a way of providing even more complex data.

Ports

A Port is the interface of an Entity to any number of Relations. Normally, a Port is contained by an Entity, although a port may exist with no container. The role of a port is to aggregate a set of links to relations. Thus, for example, to represent a directed graph, entities can be created with two ports, one for incoming arcs and one for outgoing arcs. More generally, the arcs to an entity may be divided into any number of subsets, with one port representing each subset.

A Port can link to any instance of Relation. Derived classes may wish to constrain links to a subclass of Relation. To do this, subclasses should override the protected method [checkLink\(Relation\)](#) to throw an exception if its argument is a relation that is not of the appropriate subclass. Similarly, if a subclass wishes to constrain the containers of the port to be of a subclass of Entity, they should override the protected method [_checkContainer\(Entity\)](#).

Relations

A relation is an object representing an interconnection between entities used to broadcast the output from a single port to a number of places. The single port still has only one connection to it, a connection to a relation. Relations can also be used to control the routing of wires in the diagram. A relation is represented in the diagram by a black diamond and can be selected by clicking the black diamond button in the toolbar.

C. Actor Reference

Documentation for actors and directors is located at <http://www.kepler-project.org/nightly/docResults/generatedJavadocs>. Additionally, this documentation is available within the Kepler interface. To get documentation:

1. Left click on the actor or director
2. Go to *Documentation*
3. Then *Display*.

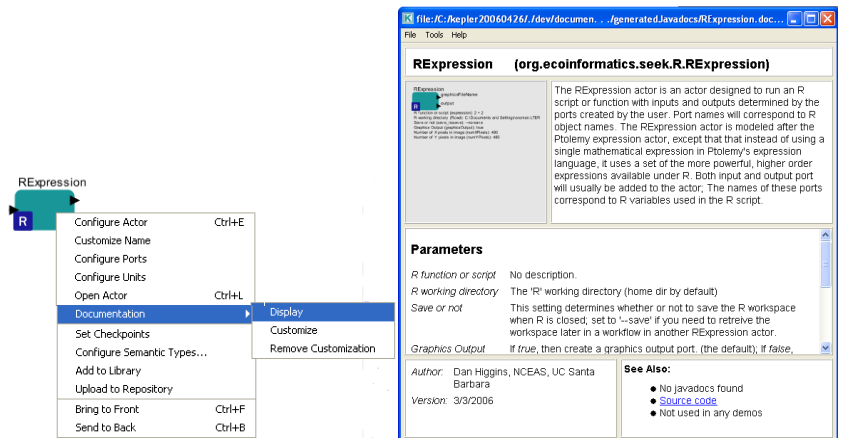


Figure 23: Actor documentation