Blaz Zupan, Gregor Leban, Janez Demsar, Tomaz Curk

Widgets and Visual Programming

Orange is a comprehensive, component-based framework for both experienced data mining and machine learning users and developers, and for those just entering the field. Orange widgets are GUI components that implement data mining tasks such as data management and preprocessing, visualization, modeling, evaluation, and other. Widgets can be assembled in a data mining application through visual programming in Orange Canvas.

When we started to design Orange, the data mining suite which should first of all satisfy our own appetites for fast prototyping of new machine learning methods, we thought that scripting is all we needed. To some point, we were right. Scripting is a powerful tool to combine existing components, prototype your own methods, and write programs for data analysis (that stay there so you can run them in exactly the same way them even after you have published the paper about them).

But as the time went by, we realized that something is missing. Data analysis is much about visualization. Oh, of course you can run a gnuplot from Python — the scripting environment for Orange — and plot a graph or two. But this is not we wanted. We were after interactive visualization. To plot a scatterplot, but be able to choose the data points from it. Visualize a classification tree, but be able to select nodes and corresponding data instances. Draw naive Bayesian nomogram, and use it for interactive classification and probability prediction. And above all that, wrap these visualization components with other standard components for data analysis, preprocessing and modeling in a schema that is easy to use yet powerful in enabling the user to visually program the data analysis procedure of her needs.

Orange Widgets

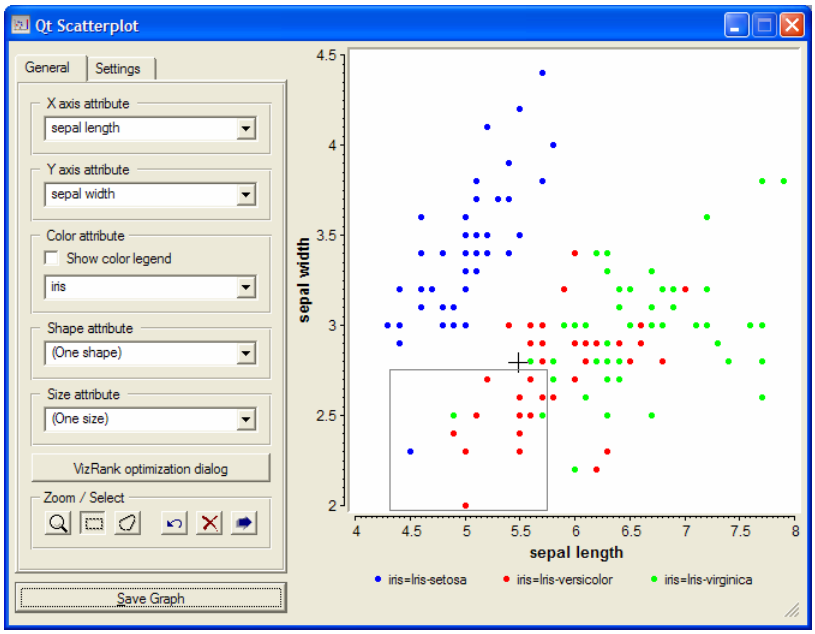
This is what took us to the concepts of visual programming and interactive graphical user interface components called widgets. Orange widgets are graphical user interface wrappers around data analysis algorithms implemented in Orange and Python.

For an example of Orange widget, consider a scatterplot widget on the next page. It shows a particular view of the data from the famous Iris data set (the data set is about classifying flowers to one of the three classes — Iris setosa, Iris versicolor and Iris virginica — based on the flower's attributes that include measurements of petal and sepal leaves). Orange widgets most often consist of the part that allows users to set some parameters of the methods implemented by a widget (left side of the widget), and the part with a particular visualization (right side). For our scatterplot, for instance, user can choose which attributes to display ("General" tab) or how particular elements on the scatterplot will look like in terms of size and color ("Settings" pane). Below find another ex-

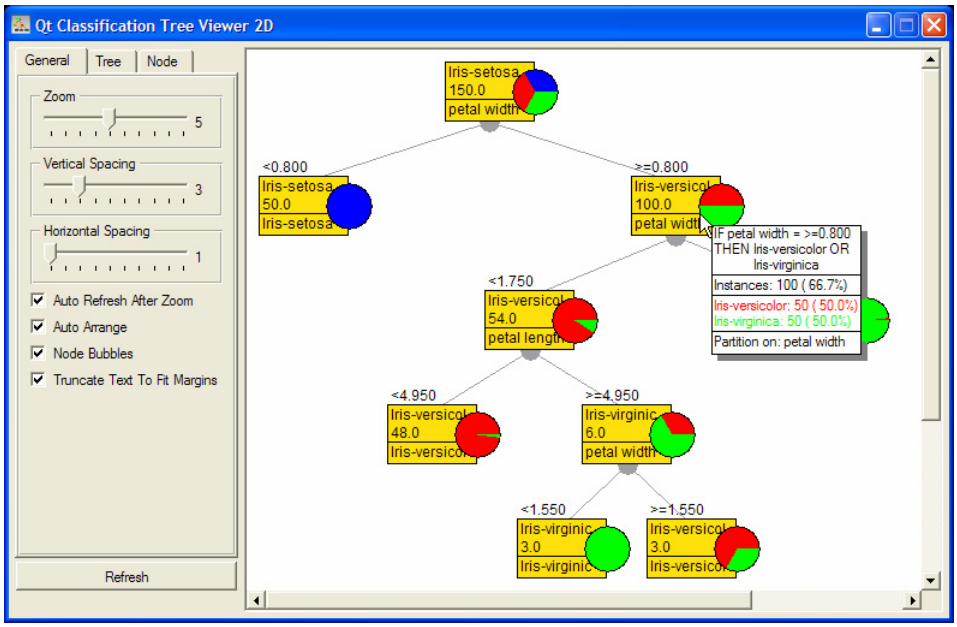
orange
DATA MINING
FRUITFUL&FUN

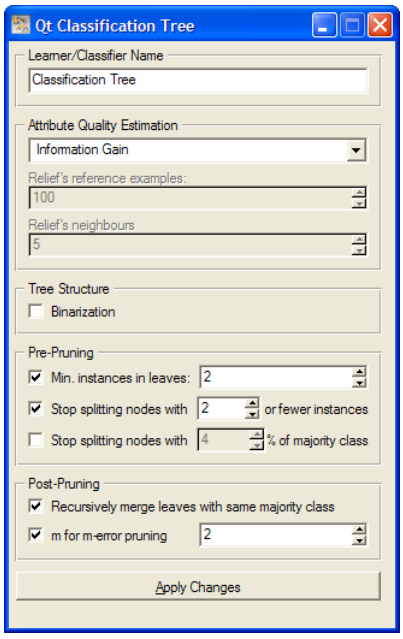
Orange Widgets most often include a part where user sets the parameters for particular data analysis or visualization method, and a part with visualization.

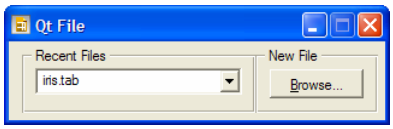
ample of the Orange Widget: a visualization of a classification tree. Again, the widget allows the user to set the parameters of visualization (three panes on the left), whereas the classification tree is shown on the right.



Orange widgets do not necessary include the visualization part. Some widgets are simpler. Like, for instance the File widget, where users specify which data file to read (snapshot on the right). Or Classification Tree widget (snapshot on





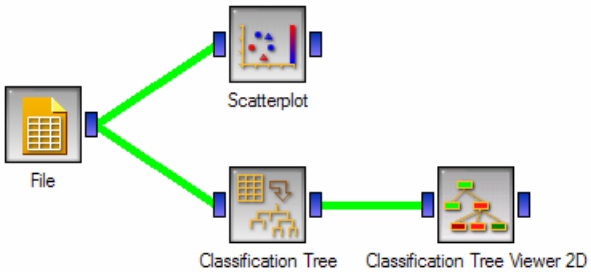


the left), which specifies the parameters for induction of classification trees.

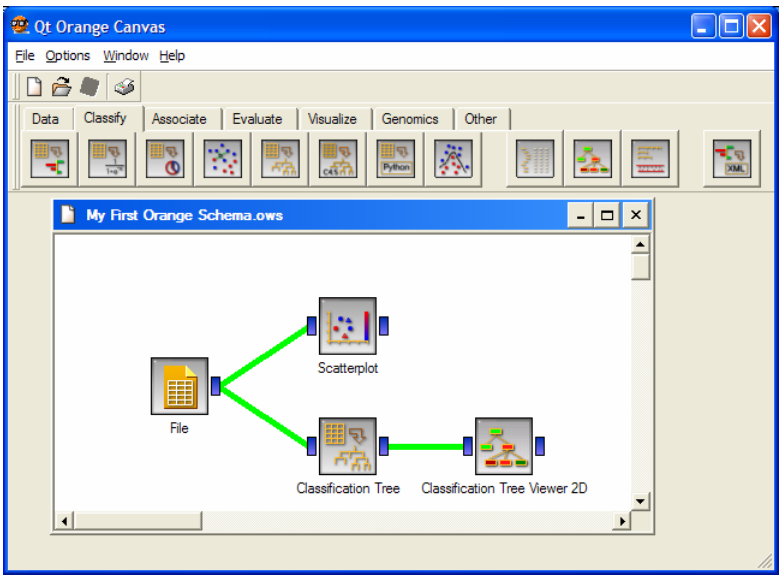
Widgets Communicate

Where did the data for the scatterplot come from? How does the tree visualization widget get the classification tree to display? Where is the data read by the File widget sent to?

Widgets communicate. They are interconnected with communication channels, and exchange packages of information called data tokens. Think of a widget as a box with the output connector on the right and the input connector left (blue rectangles on both sides of widget icons). The File widget sends the data it reads to both Scatterplot and Classification Tree. Once presented with the data, Classification Tree induces the classifier and sends it to the tree visualization widget.



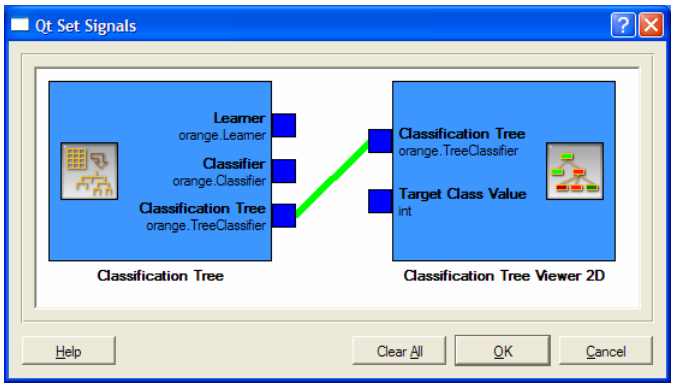
Widget Schemas and Orange Canvas



How do we connect the widgets? Widgets are Python objects derived from Qt and connecting them requires a Python script to establish a Qt signaling which, in turn, ... OK, you don't need to know this.

Actually, connecting the widgets is simple. To set up a schema - a system of interconnected widgets - you can (and always will) use an application called Orange Canvas (snapshot on the left). All you need to do is to find the widgets for your task, click them to appear on the canvas and intuitively connect them by dragging a line from the output connector of the source to the destination widget. Orange Canvas will take care of the dirty work.

Communication Channels

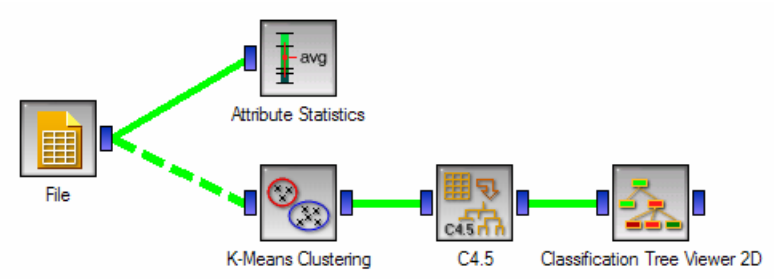


Each Orange widget would most often support a number of input and output channels, and would thus accordingly send and receive tokens of different types. To distinguish between them, the Orange channels are typed. For instance, the Classification Tree and Classification Tree Viewer widget can exchange data tokens of the type called "Classification Tree", which is (for those inspired to go deeper) an Orange object of the type orange.TreeClassifier. The particular connection established between the two widgets is shown in the window on the left. We can see that Classification Tree widget can also output the object called Learner, which is an Orange object that returns a classification tree once presented the data.

Widget channels are typed, making in easy for Orange to establish the right connections between widgets.

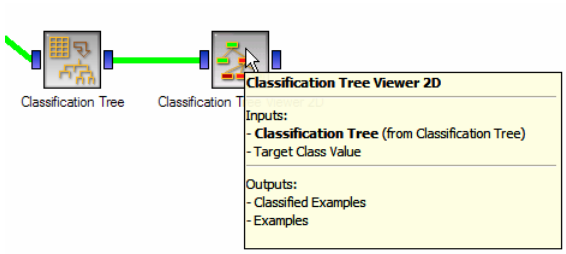
The typing also helps Orange to make sure we connect two compatible widgets, and when widgets are used in schemas we describe next, Orange can establish the right connections between two widgets automatically.

In Orange Canvas, a channel may be disabled or enabled. You may disable the channel if you want to have just a part of the schema active.



For instance, in the schema above, the changes in the File widgets would not propagate to *k*-means clustering, but one would see the updates in Attribute Statistics widgets. The communication mechanism that Orange uses would, however, remember most recent data token send out by the File widget, and enabling the channel with k-means clustering would propagate it immediately.

To find about particular channels a widget implements, position a mouse over the icon that represents the widget. Waiting there for a second will present all channel types the widget supports, and will show the ones in use in bold.



Using Schemas

Using schemas is just as intuitive as everything else. Double clicking the widget opens it. For instance, double clicking the Scatterplot icon on the canvas will show the scatter plot which, if provided by the data, will look like the first snapshot you have seen.

Before reading the next paragraph (you do have Orange installed and running on your computer, right?), try opening the File, Scatterplot and Classification Tree Viewer widget.

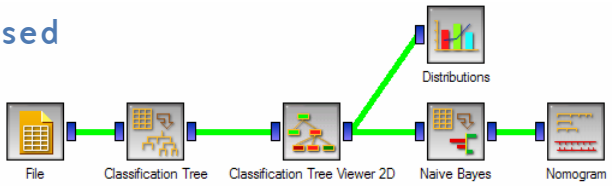
Tokens sent by the widgets propagate through the schema from the source widgets on. Changing the data file in the File widget from the schema with classification tree (say, from iris.tab to titanic.tab, both included in the Orange

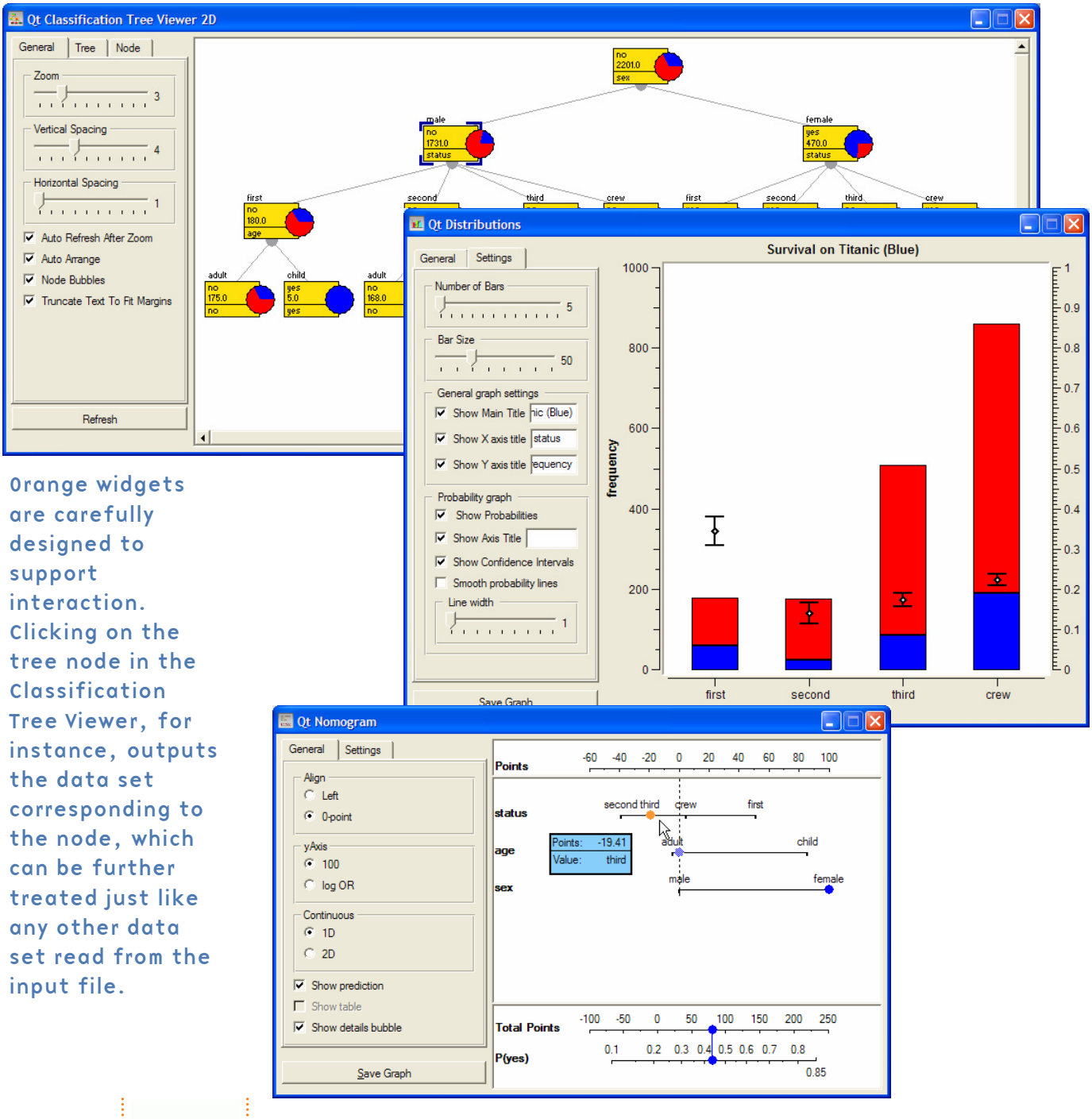
distribution) would send a data token to both Scatterplot and Classification Tree widget. Scatterplot will update its interface and the graph. Nothing much (in visual terms) will happen to classification tree, but the functional part of this widget will accept the data, build a corresponding classification tree and send a token with it to the viewer. If you have File and Classification Tree Viewer both open, you will see that changing the input file results in an update of the tree in the viewer. In the same way, any other changes, such as modifying any arguments in Classification Tree widget, automatically propagate down the schema.

So we came to the most important aspect of Orange Schemas: their interactivity. Consider the schema on the next page. It reads the data, induces a decision tree, and presents it. By clicking on tree nodes you can select the subsets of the data - examples belonging to a particular tree node - and use them for further analysis. The schema from the snapshot draws the class distribution graphs, dynamically, as we go from node to node, and even constructs naïve Bayesian classifiers and draws the corresponding nomograms for each selected subset.

orange
DATA MINING
FRUITFUL&FUN

Classification Tree-Based Data Exploration





Orange widgets are carefully designed to support interaction. Clicking on the tree node in the Classification Tree Viewer, for instance, outputs the data set corresponding to the node, which can be further treated just like any other data set read from the input file.

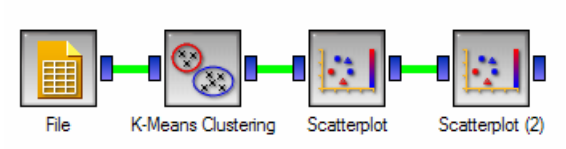
Visual Programming

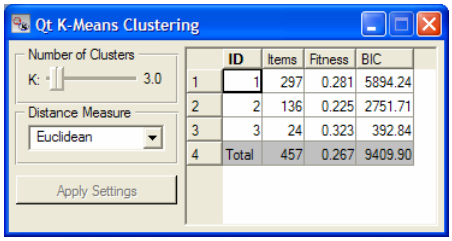
Versatility of Orange schema led us to call drawing schemas "visual programming." Before accusing us of being pretentious: is not programming all about taking the building blocks you have and arranging them in combinations that suit your particular task? "Programming" that Canvas allows for is undoubtedly simplistic, with no loops and conditional clauses, but that was exactly our purpose: to provide a simple tool that even a non-programmer can use to design procedures that he could never dream of with the usual, menu-based data mining tools. Combining these components together may be rather engaging and you may be surprised how easy is to come up with a particular (and useful!) combination one has not seen before.

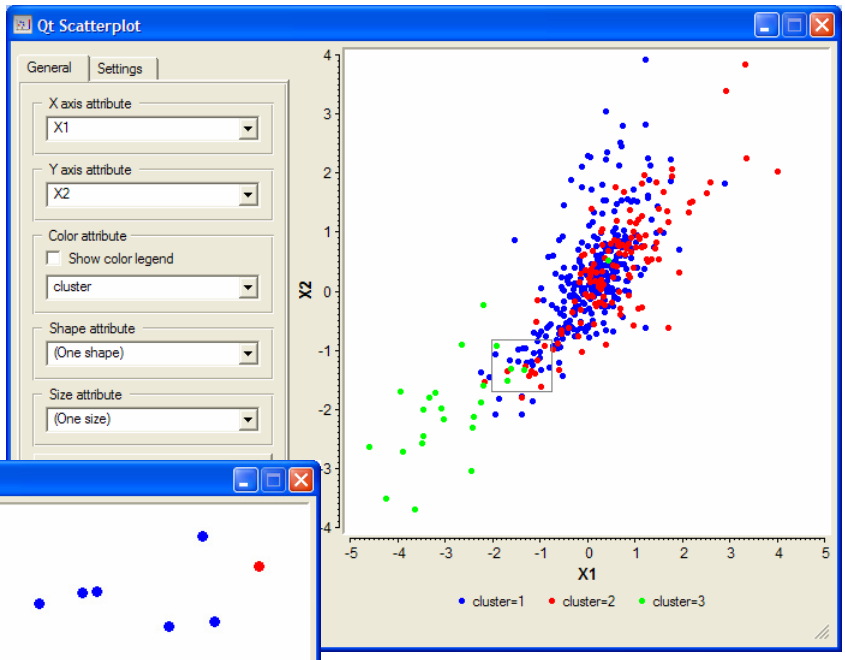
The one that is simple enough, but we quite like it for the example, is a magnifying glass. You can introduce the magnifier with a number of orange widgets, but the simplest one is with a scatterplot. Connect two scatterplots together, one after the other, and with appropriate set up the instances you select at first Scatterplot may be displayed in the "magnified" graph in the second scatterplot. The schema illustrating this concept is shown on the next page.

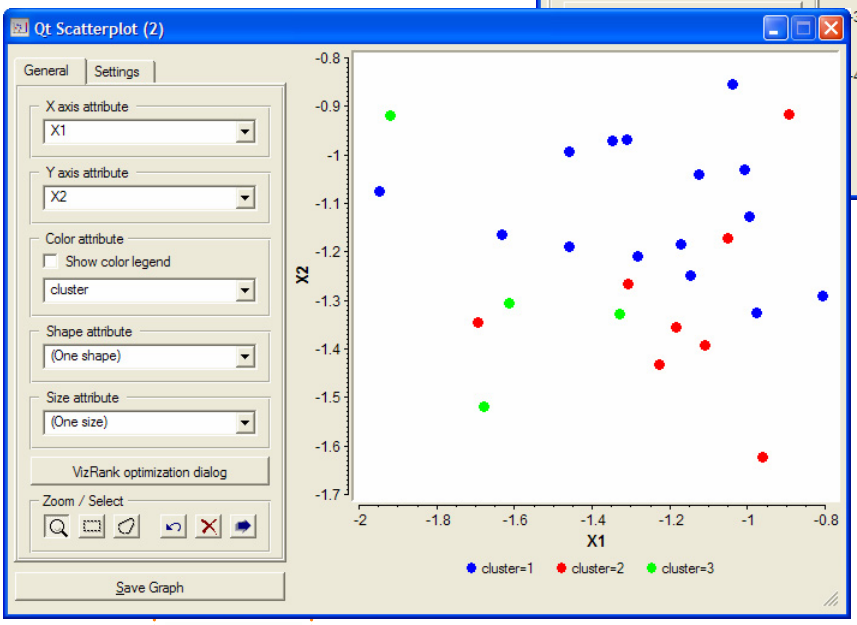
Magnifying Glass

Widgets are (almost) like Lego bricks: you can combine them in so many different ways that one gets often surprised to how easy is to design analysis schemas that one has never encountered before. The particular combination shown on this page is called a magnifying glass.

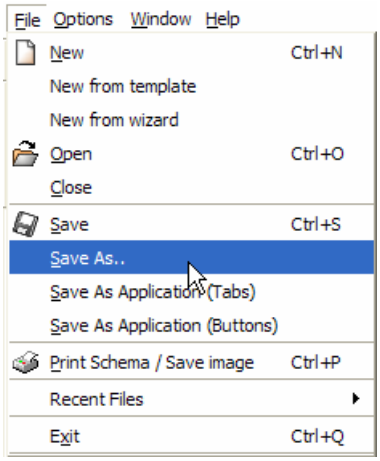








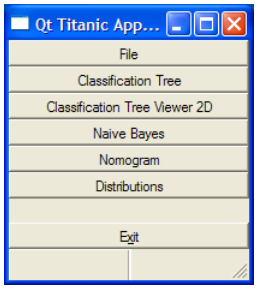
Saving Your Work

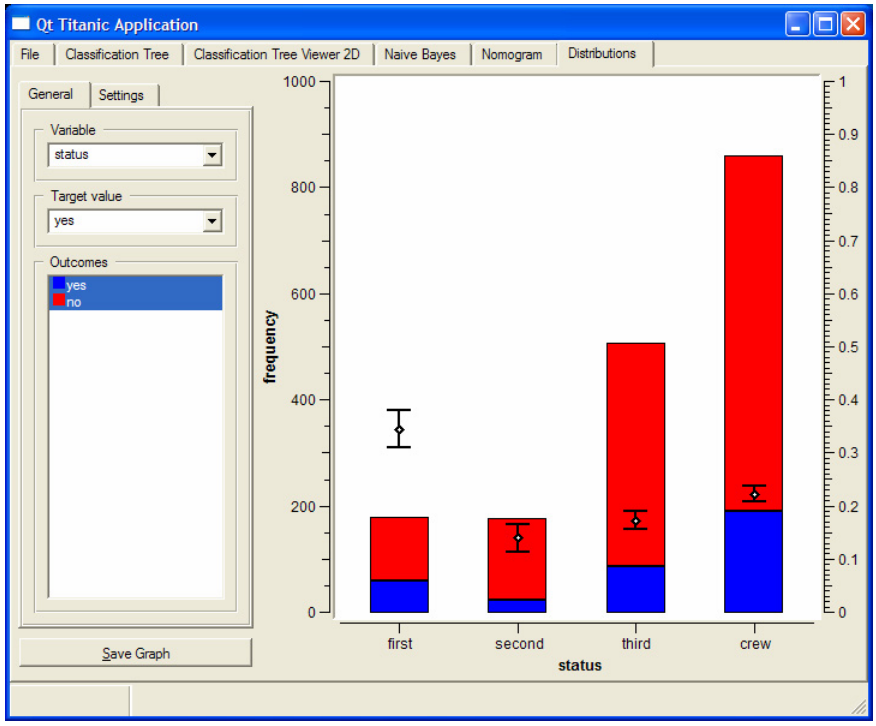


Schemas can be saved and retrieved with standard commands of Save, Save As ... and Open from the File menu. Besides the layout, saving your schema would also store any setting that you used with each of the widgets.

Orange Applications

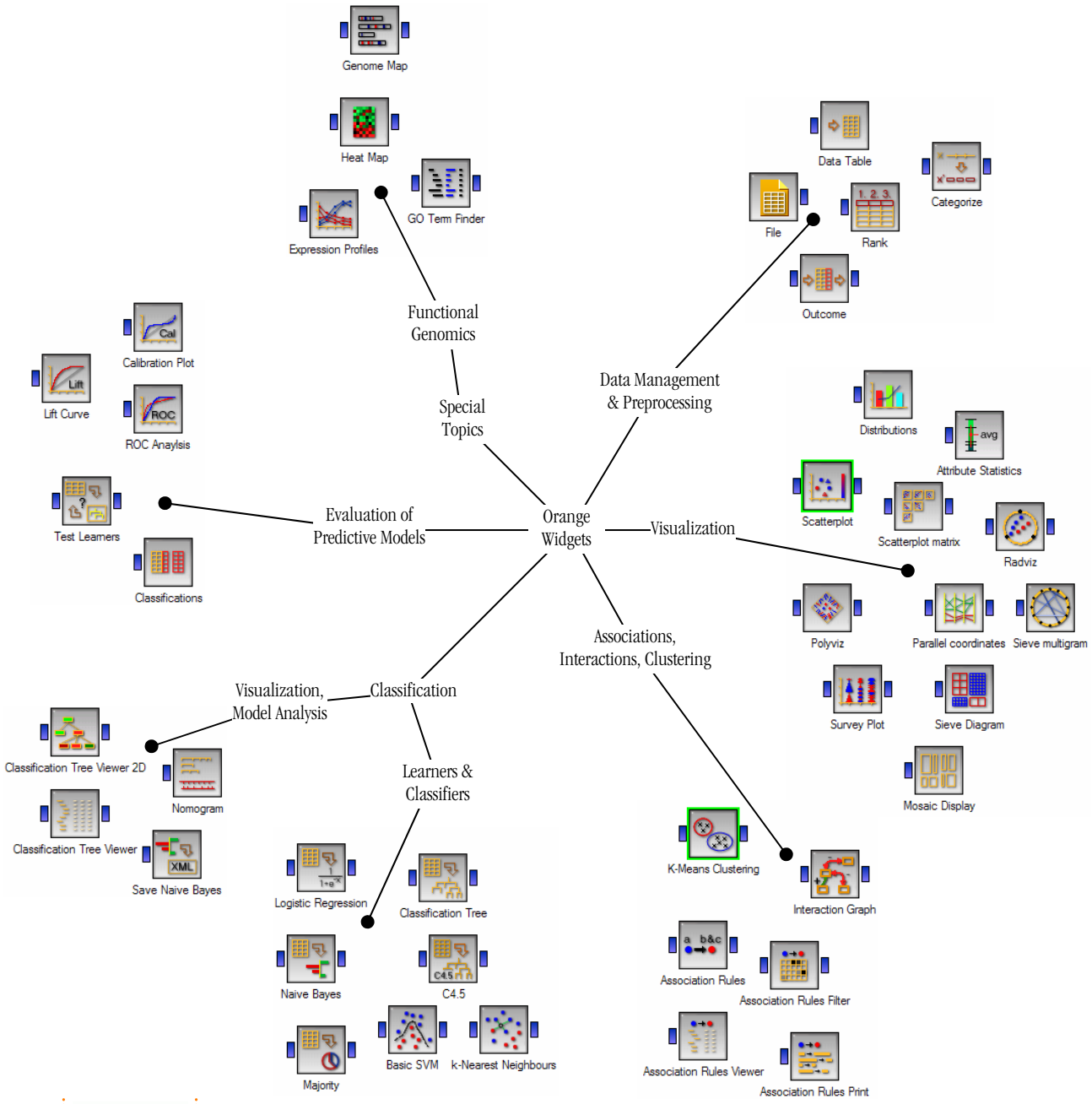
Orange schemas can also be saved as "stand alone applications". We use quotes here: the application will indeed be saved in a single, double-clickable, file, but to run it one will still require a complete Orange installation on the disk. There are two ways to save your application, depending on how you want to see the interface. In a tabbed application, the contents of every widget will appear in the separate pane, and in a buttoned application, each widget would be invoked by pressing on the button in the main control window. Here are the snapshots of how the two applications saved from the schema from the previous page would look like.





A Map of Orange Widgets

Orange currently includes about forty widgets, with a current rate of about one new widget coming out every three weeks. Widgets cover a wide range of data analysis tasks, including data management, preprocessing, visualization, modeling, evaluation and model analysis.



If you use Orange and Orange Canvas, send us a postcard with any comments and wishes for further development. Please use the following address:

Orange, AI Lab, Faculty of Computer and Information Science, University of Ljublana, Trzaska 25, SI-1000 Slovenia.

Acknowledgements

We are thankful for comments, encouragements and contributions of our colleagues and friends. We would first like to thank members of our AI Laboratory in Ljubljana for all the help and support in the development of the framework. Particular thanks go to Aleks Jakulin, Martin Mozina, Peter Juvan, and Ivan Bratko. Marko Kavcic developed the first prototype of widget communication mechanism and tested it on first few widgets. Some of association rules algorithms were implemented by Matjaz Jursic. Martin Znidarsic helped us in development of several very useful methods and in beta testing. Jure Zabkar programmed several modules in Python. Matjaz Kukar was involved in early conversations about Orange and, most importantly, introduced us to Python. Chad Shaw helped us with discussions on kernel-based probability estimators continuous attributes and classification. Gaj Vidmar was always available to help us answering questions on various problems we had with statistics. Martin Znidarsic and Daniel Vladusic used Orange even in times of its greatest instability and thus contributed their share by annoying us with bug reports. In porting Orange to various platforms we are in particular thankful to Ljupco Todorovski and Mark E. Fenner (Linux), Daniel Rubin (Solaris) and Larry Bugbee (Mac OS X).

Orange exists thanks to a number of open source projects. Python is used as a scripting language that connects the core components coded in C++. Qt saved us from having to prepare and maintain separate graphical interfaces for MS Windows, Linux and Mac OS X. Python to Qt interface is taken care by PyQt. Additional packets used are Numeric Python (a linear algebra module) and Qwt (a set of Qt widgets for technical applications) among with PyQwt that allows us to use it from Python.

A number of Orange components were build as an implementation of methods that stem from our research. This was generously supported by Slovene Ministry of Education, Science and Sport (the Program Grant on Artificial Intelligence), Slovene Ministry of Information Society (two smaller grants on development of open source programs), American Cancer Society (in collaboration with Baylor College of Medicine, grant on predictive models for outcomes of prostate cancer treatments), and USA's National Institute of Health (in collaboration with Baylor College of Medicine, program grant on functional genomics of *D. discoideum*).