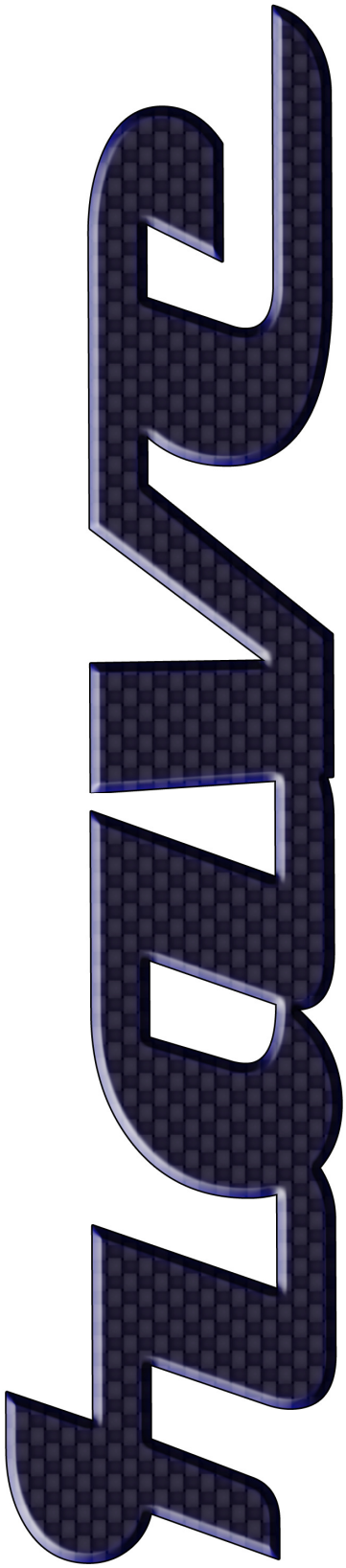

RAVE USER MANUAL

VERSION 0.9.0

Contents

Introduction

RAVE is a matlab-based tool that provides a point-and-click interface to many of MATLAB's visualization, optimization, metamodeling, and data analysis functions.

If you haven't already done so, you should read the RAVE Installation Guide and Release Notes for information on correctly installing RAVE.

Please note that RAVE is still very new, and will have some bugs, as well as many missing features. I'm sure there will be times when you say "I wish RAVE could do X" or even "I can't believe that RAVE can't do Y." PLEASE let me know when this happens by emailing mdaskilewicz@asdl.gatech.edu. Your feedback will truly help us make RAVE into a useful **free** tool for decision support and data analysis. We update rave very frequently, so depending on the complexity of your request, you could see it implemented in just a few days.

This user manual is still very sparse, but hopefully has enough information to get you started. If you have ANY questions, please email me!

Getting Started

First, follow the installation instructions in the RAVE Install Guide.

To start rave, open MATLAB and type "rave". If you haven't used rave before, a setup GUI will run.

Once rave has started, there are two rules:

1. You must load a data set before you can do anything else

2. After loading a data set, you must create a graph before you can do anything else. If at any time there are no graphs in the workspace, the only thing you'll be able to do is create a graph.

Also note: The "continuous" tab on the "New Graph" GUI will be disabled until you load a function using the model tab. Continuous graphs require functions to generate data on-the-fly, so they can't be created until you load a function.

User Interface

The main rave interface is shown in Figure 1.

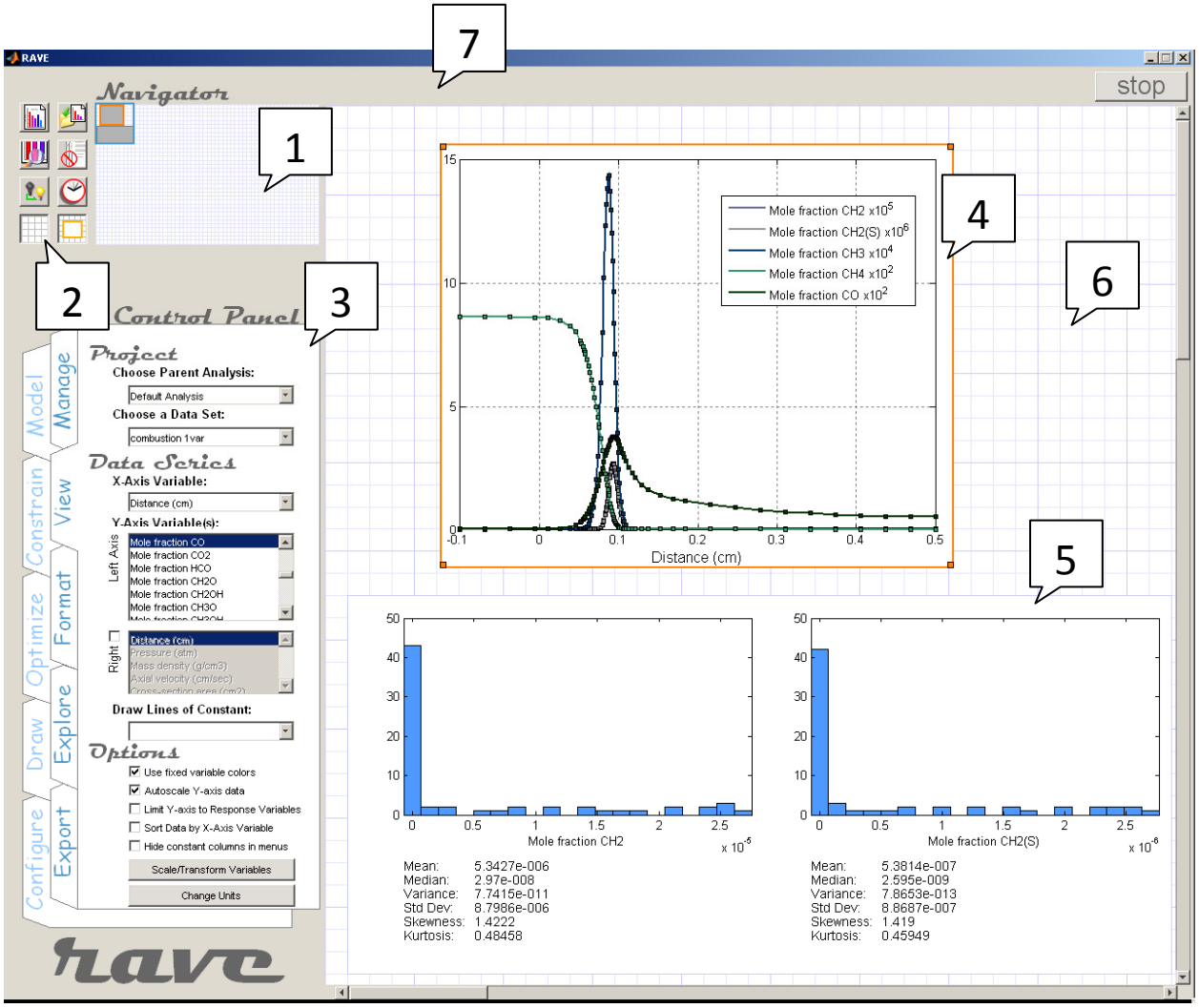

Figure 1. Interface overview

The numbered elements are:

1. Navigator
2. Untabbed Controls
3. Tabs
4. Selected Graph
5. Unselected Graph
6. Workspace
7. Infobar

Tabs

Most of rave's controls are located in the tab pane at the left of the screen. There are ten tabs, each containing controls related to a particular type of task.

The ten tabs are:

- Manage – Load and edit data sets and analyses, create new graphs, save/load projects
- View – Pick which variables are displayed on the current graph
- Format – Pick colors, labels, and other appearance options for the current graph
- Explore – Overlay additional data visualizations on the current graph
- Export – Save or print graphs or data
- Model – Load functions, create metamodels and objective functions
- Constrain – Create and view constraints
- Optimize – Run an optimizer
- Draw – Add powerpoint style annotations to graphs or workspace
- Configure – Change the way rave operates

The content of the view, format, and explore tabs depends on which graph is currently selected. Each time you select a new graph, the current tab will update to reflect the settings of the new current graph.

To change tabs, simply click the name of the desired tab. NOTE: you can't change tabs until you have created at least one graph. If there are ever no graphs in the workspace (because you deleted them all) you won't be able to change tabs until you create a graph.

Using the Format Tab

Most changes on the format tab are made by simply clicking the desired formatting options. The currently selected options are colored blue and unselected options are gray.

The Workspace

The workspace is the blank canvas on which you will place graphs, tables, images, text, and. You can make the workspace larger or smaller by changing the preference "wsheight" and "wswidth" before you start rave. (There is currently no way to resize the workspace once rave has started.)

Using the Navigator

The Navigator in the top right of the screen shows a zoomed out view of the entire workspace. Each graph is shown as a rectangle. Gray rectangles indicate vectorized graphs. Violet rectangles indicate rasterized graphs. The orange border shows the currently selected graph. The blue rectangle shows the region of the screen you are currently looking at.

Moving around the workspace

You can move your current view of the workspace by:

- Using the scroll bars located along the edges of the screen
- Clicking an empty region of the workspace and dragging it

- Clicking and/or dragging the “Navigator”
- Using the scroll wheel on your mouse (if a menu or other scrollable object does not have focus)

Changing the Workspace appearance

You can change the color of the workspace background and the gridlines on the Configure tab.

You can show/hide the gridlines by clicking the gridline button in the Untabbed Controls.

You can show/hide the orange border around the current graph by clicking the border button in the Untabbed Controls.

You can hide all controls for a full screen view of the workspace by clicking the “no tabs” button in the Untabbed controls. To make the controls reappear, hit the Esc key (you must first click inside the rave window at least once). When in full screen mode, grid lines and the orange border around the current graph disappear. Note: Currently there is no way to change which graph is the current graph while in full screen mode, but this will be fixed in the future.

Untabbed Controls

There are eight buttons in the top left of the screen to access commonly used functions without having to change tabs.

The buttons in the first column are:

- New graph: click this to open the new graph dialog box
- Zoom: this doesn’t do anything yet
- Show hidden data: click this to unhide all hidden data points
- Gridlines: click this to show/hide the workspace grid lines

The buttons in the second column are:

- Put away graph: Click this to “put away” the currently selected graph. It will disappear, but you will be able to restore it later if you’d like. (Note: This doesn’t work yet)
- Hide tabs: Click this for a full screen view of the workspace.
- Renderer: Click this to change between the painters and opengl renderers. Using opengl may speed up certain tasks, but lowers the quality of all graphics. Generally you want to be using painters. (if you don’t understand this, don’t worry about it. Just don’t click this button.)
- Border: Click this to show/hide the orange border around the currently selected graph.

Infobar

The infobar is the empty region at the top of the screen. Informational text and instructions are displayed here during certain tasks.

The debug button

There is a "debug" button in the top right corner that enters debug mode and allows you to manually type commands or view all variables rave has stored. If you click this by mistake, you can just keep working and ignore it, but it is better to type "return" at the command prompt to exit debug mode. (Sometimes just being in debug mode causes matlab to crash for no reason.)

Data Sets and Analyses

RAVE groups information into two main structures: data sets and analyses.

Each data set is created by loading a .txt or Excel file that contains a data table. You can load as many data sets as you want within a single RAVE session. Each data set is completely independent of other data sets within RAVE – there is no way to do any sort of linking between data sets.

Data sets MUST be in a "flat file" format. This means the first line of the data file contains variable names as column headers, and the rest of the file contains one data point on each row. The files can either be tab/comma delimited or fixed width.

Each graph shows data from a single data set, and belongs to an "analysis." All graphs that belong to the same analysis are linked, so that changing any of the analysis properties (by interacting with a graph) causes all graphs in the same analysis to update instantly to reflect those changes. By using multiple analyses, you can (for example) manage multiple sets of row coloration, and quickly switch between the different colors.

When you start RAVE, a single analysis (named "default") is created automatically. You can create additional analyses, if you need them, in the Manage Analyses window.

Every data set "belongs to" every analysis. Creating a new data set does not create a new analysis to "go with it," instead that data set now exists in every analysis you have already created, and has the default properties (all points visible, unselected, default color (blue), and no constraints).

Properties of Data Sets include:

- The set of variables, their names, and their values for each row in the data set
- An indicator of whether each variable is independent or dependent
- The class of each independent variable (continuous, discrete, etc…)
- For each dependent variable: The name of a matlab function that calculates this variable's values, and the index of which output of the function corresponds to this variable.
- For each dependent variable: A list of the other variables that are inputs to the function that calculates this variable.
- A color associated with each variable (used as the default line color for that variable)
- The units of each variable (coming soon!)

- The “preference” for each variable, used in decision making techniques
- An indicator of whether each variable is considered to be an “objective” or not
- A single variable that is flagged as being the “ranking variable,” whose values are used whenever a method needs to identify the “best” row in the data set

Properties of Analyses:

- A list of “current values” for each independent variable, used for making things like prediction profilers
- A range or list of allowable values for each independent variable, used for settings limits on prediction profilers and optimizers
- A color associated with each row in the data set (limited to 10 total colors)
- An indicator of which rows in the data set are currently selected
- An indicator of which rows in the data set are currently visible
- A set of constraints that acts on the data set and a default line color associated with each

Note: I realize it would be more useful if preferences were a property of analyses instead of data sets, but that would be very hard to implement for reasons I won’t go in to.

Loading a new data set

The first thing you should do when you start RAVE is load a data set. Most functions (including changing tabs) will be disabled until you load a data set and create a graph. To load a data set, click the “manage data sets” button on the manage tab, and then click the browse button in the window that pops up. Select the file you want to use and hit “Create this Data Set.”

Supported file types are .txt, . csv, and ,xls files. If you use other data formats in your work that you would like supported in RAVE, please let me know!

After a couple seconds, the data set name should appear in the lower half of the window under the words “Manage Data Sets”. You can hit “Done” or use the buttons on the right to modify the data set if necessary. For example, you can use the Define Data Types button to make some variables discrete.

When you close the Manage Data Sets window (by hitting “Done”), you are ready to create a graph.

Changing Data Set Properties

For now, the only things you can change for a data set are the “class” of each variable and the target value of each variable. Both of these can be changed from the Manage Data GUI (top button on the manage tab). Once you change a variable’s class, you should go to the Manage Analyses GUI to check/change the allowable values for each variable. Note that these values are properties of the *analysis*, not the data set so that you can easily manage multiple sets of allowable ranges for a single data set.

Changing Analysis Properties

Row selection, color, and visibility

Note: In a future release, these controls will be moved to the "Explore" tab.

Row selection, coloration, and visibility are all changed by selecting data on a graph (using the options on the Format tab), then (if desired) changing the data's color using the row color buttons or hiding the data using the hide button. These are shown in Figure 2 for a scatter plot.

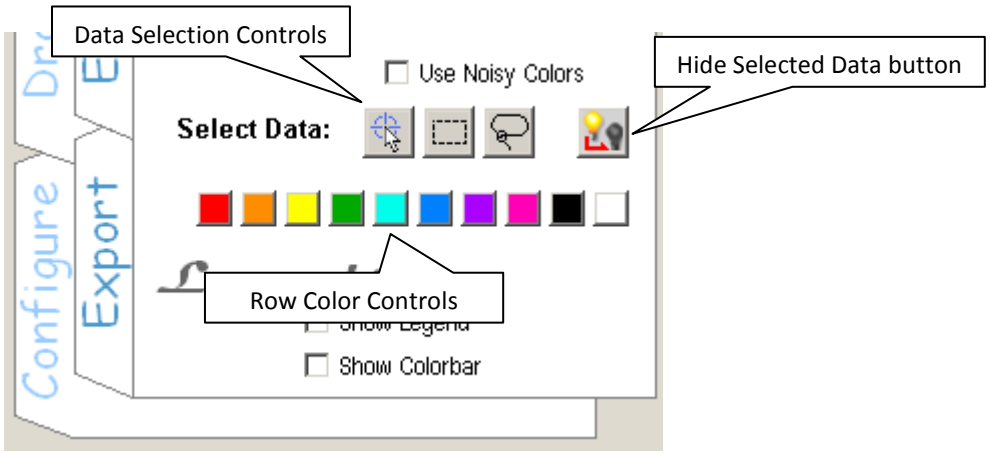

Figure 2. Data selection, visibility, and row color controls for a scatter plot

Changing row colors

To change a row's color, select the desired row(s), and click the Row Color Control button of the color you would like to make the rows. You are limited to ten different colors to be used by all rows in a single rave session.

Customizing the available row colors

Although you are limited to 10 colors, you can change which 10 colors are available by holding CTRL and clicking on the Row Color Controls. You can select from the colors available on any of the figure's colormaps. If you would like to use a color that is not in offered in the pop-up window, you will have to edit one of the three custom colormaps (the last three columns of the pop-up) to make your desired colors appear. See the section of this document on editing custom colormaps.

Changing a Graph's Data Set or Analysis

Every graph's data set and analysis can be changed using the pull-down menus located at the top of the View tab when that graph is selected. By default, each new graph you create will belong to the first analysis (named "default" unless you changed its name) and will show data from the first data set.

Moving all Graphs in one Analysis to another Analysis

Coming in a future release.

Functions and Modeling

Using MATLAB functions with RAVE

Note: in the future I would like to accommodate more "first line" formats. If you have a good idea for how to make this less burdensome on the user, let me know.

RAVE can run just about any matlab function, BUT the function must have its first line in the form:

function [output1, output2, output3, …] = functionname(data, prefs, targets)

Also, **THE FUNCTION MUST BE VECTORIZED**! This means it must be capable of running multiple points with a single call and returning vectors of results. If your function is not truly vectorized, you can easily vectorize it by just wrapping the whole thing with a for loop to loop through every row of the input data.

"Data" is an array that contains all input variables to this function (which you will specify when you load this function in RAVE) in the order that they appear in your data table. It is recommended that you write your functions to simply take in ALL the unmodeled variables in your data table so when you load it in RAVE you don't need to worry about telling it which variables to use as inputs.

"Prefs" is a **vector** of preferences for the variables in data, and "targets" is a **vector** of the "best values" for the variables in data. (These vectors are created by RAVE.) These are rarely used, but if you're making some sort of OEC or objective function you might need them. **Even if you do not use these vectors in your function, you must include them as inputs!!!!** All you have to do is list them in the first line of the function as above… RAVE will handle the rest.

Output1, output2, etc… is a list of all outputs from your function**. Each output must be a column vector.**

Note that you can name the inputs/outputs whatever you want, all that matters is that there are exactly three inputs, of which the first is the data array, and that each output is listed in a separate variable. When you import the function into RAVE, the resulting variables will have the same names you use for the output variables in your function file.

Using non-numerical data with MATLAB functions

If your MATLAB function takes inputs whose values are strings (aka words), it will work exactly the same as described above, EXCEPT the input variable data will be a cell array instead of a numerical array. Each number or string will appear in its own cell, and the cell array will have dimensions (# of runs) x (number of input variables) Note: I haven't actually tested any functions that take strings as inputs yet.

Using txt/JMP functions with RAVE

You should be able to use any simple rse or neural network from JMP in rave without making any changes. Just open it in JMP, highlight the whole thing, hit ctrl+c, and then ctrl+v to paste it into rave (there is no paste button yet, have to use the keyboard) If you use more complicated functions however, this might not work. Also some simple functions have different names in jmp vs matlab which I may not have written into the parser. If your function doesn't work, let me know and I'll try to update the parser to handle it.

Note: JMP functions currently do not support non-numerical (text) data

Using simple functions with RAVE

In theory you can use the "load TXT function" dialog box to type in any simple function, like a sum or product of two variables in your data set. Just make sure you spell all variable names correctly.

Using objective functions with RAVE

There is a separate dialog in RAVE to create "objective functions," which differ from normal responses in that they are functions of preferences/targets as well as data (but note that all matlab functions you use with rave need preferences/targets as inputs even if you don't use them.) The dialog will auto-generate OEC or TOPSIS matlab functions of your data and load them into RAVE, or you can type your own objective function. If you have a particular class of objective functions you use frequently, talk to me about having it added as a template in RAVE. Note: you can use ANY function as an objective function when running an optimizer, not just "objective functions" as defined in this paragraph.

Graphs

In rave lingo, a graph is anything that is created from the Create New Graph dialog window, and includes tables and controls that are placed on the workspace.

Note: The following graphs don't work yet: Spider/radar plot (though parallel coordinates are a better alternative to these), response profiler (though you can create one from the exploretab for scatter plots), overlay line plot, Pareto plot, map, all tables, and all controls except independent variable sliders and preference sliders.

There are five classes of graphs, with the most important distinction being between data-based graphs and equation-based graphs. The five graph types are:

- Data-based graphs - These can be created for any data set. They directly plot the data included in the data set.
- Equation-based graphs – These can only be created for data sets that have functions associated with them (loaded using the Model tab or a rvefun metadata file). These do not directly use the data set, but generate their own data on the fly by evaluating the associated functions.
- Special Graphs – Images and graphs that serve a very particular purpose.
- Tables – Tables of data. Note: These currently don't work.
- Controls – Variable value sliders or other menus that can be deployed to the workspace to create interactive graphs, dashboards, and calculators.

Creating, Moving, and Resizing Graphs

To create a graph, open the Create New Graph dialog by clicking the Create New button on the Manage Tab, or by clicking the New Graph button in the Untabbed Controls. Select the type of graph you want to create by clicking its button, and then place the graph on the workspace by clicking and dragging the mouse to size the graph.

You can select a graph by clicking the empty space between the graph itself and the solid colored rectangle that surrounds the graph and its axis labels. This gives it the (usually orange) resizing border and makes the tabs update to show the information related to this graph. When you create a new graph, it is automatically selected. In rave lingo, the selected graph is called the "current graph".

You can resize the current graph (the one with the orange border) by clicking and dragging the small squares at each corner (note, sometimes the lower right square cannot be clicked because it is too close to the axes label. If that happens, resize the graph using the other three corners). While resizing a graph, the infobar displays the dimensions of the graph. By default, graphs are snapped to ¼" sizes; you can enable smooth resizing on the Configure Tab.

You can move a graph (not just the current graph) without resizing it by holding CTRL and clicking in the empty space between the graph and its bounding box, then dragging the graph to the desired position.

Changing Graph Titles and Axis Labels and Limits

Each graph can be given a title or axis labels by selecting the graph and clicking on the Format tab.

The axis limits for each graph can be changed on the Format tab. Axis limits may be specified as numbers, or mathematical expressions that evaluate to a number, for example “pi/2” (entered without quotes).

For matrix plots, you must first select a sub-graph within the matrix by clicking it while on the format tab. It will then be highlighted in orange, and you can change its limits or labels by using the controls in the format tab. Note that the changes you make will also affect other sub-graphs in the same row/column of the matrix as necessary.

Customizing Lines Drawn on a Graph

Many lines that have been added to a graph, such as constraints, can be customized by **double-clicking** on them to open the property editor. If you want to change the color of a line, it can also be done by **right-clicking** on the line to open the color editor. Changes made using the right click will be “permanent,” while changes made by double-clicking only effect the current line, and will be lost if the line is redrawn for any reason. Some lines can be deleted by **shift-clicking** on the line, while other lines (those that are integral to the graph) cannot be deleted. **Single-clicking** on a line will bring it to the top of the image.

Constraints and Optimization

I don't have a good "user manual" for this yet, but everything should work with the following exceptions:

- the MATLAB optimizer gamultiobj, which rave calls "Controlled-elitist NSGA-II" does not support constraints. For constrained multiobjective optimization, use the other NSGA-II algorithm (written specifically for rave)
- I haven't tested equality constraints.
- RAVE currently does not distinguish between "less than" and "less than or equal to", which for continuous optimization doesn't really matter, but for discrete optimization does matter.
- I think the optimizers do not currently support discrete variables. All variables will be treated as continuous between their upper and lower bounds. (I don't have much hope for ever getting this supported for the MATLAB toolbox optimizers, but it can be implemented for custom-written optimizers)
- The "pause" button for optimization only works for NSGA-II.
- Optimizer animation only works if the currently selected graph is a scatter plot.

(Ok, now that I look at it, that's a pretty long list. I'll work on fixing these as soon as possible. At the very least, the NSGA-II optimizer works very well. It can also be used for single objective optimization, although it runs slower with only 1 objective.)

Constraints must be enetered on the Constrain tab. You can either enter them as a mathematical expression (using normal matlab syntax), or for more complicated constraints, you might find it easier to write a dedicated matlab function for the constraint using the same format as other rave functions. Then in the constrain tab your constraint will just be <constraintvariablename> <=0

Make sure when you enter constraints, you don't type out the "<=0" part of it, instead use the buttons to determine the type of inequality, and enter the numerical value in the box to the right of the inequanlity buttons.

To run multiple optimizers in a row, instead of clicking "Start", click "batch", then use the menu's to set up the next optimizer, and click "batch" again. Repeat as needed. When they are all set up, click "start". They will run and the results will be saved in a directory of your choosing. Note: I haven't tested this with any optimizer but NSGA-II.

Saving and Loading Sessions and Metadata

Sessions can be saved and loaded from the Manage tab. If you set up .rve file type associations in your operating system, you can reopen saved sessions by double clicking them. Otherwise, you can reopen them by starting rave, then clicking "load project" from the manage tab.

Working with Metadata

Metadata files contain additional information about your data that is not contained in the data file itself. Things like functions that act on the data, constraints, and row colors are contained in the metadata files.

There are eight types of metadata files that RAVE searches for. Each type of metadata has a different six-letter file extension (described below), and is a plain ascii text file (tab or comma delimited) with a particular format (also described below). RAVE can also generate these files for you, so you typically don't need to create your own, but the formats are described here in case you want to edit them manually.

Whenever you load a data set, RAVE will also search the folder the data file is in for metadata files with the same filename (minus the extension) and will load those as well. It will load any of them it finds, even if you don't have all nine. Optionally, you can choose not to load the metadata when you load the initial data file, and can instead select individual metadata files to load later as you need them. The initial search will only find up to one file of each type, but you can load as many files of each type as you want later on. For example, if you have two sets of constraints stored in separate files, you can load them both individually after you create the data set. But during the initial data loading, a metadata file will be loaded only if its filename matches the name of the data file you loaded.

The first time you use a particular data set, you won't have any metadata (unless you manually create the files), but once you start doing things in RAVE like changing row colors or adding constraints, you can save metadata files so that next time you work with this data set you won't need to redo all those things. Metadata is therefore something like a midway point between starting from scratch and saving the entire RAVE project. Metadata only saves the information listed below, not anything that is visible on the rave workspace (graphs, controls, etc). So if you start a new RAVE session and load a data set with metadata you'll still start with a blank workspace, but your data state will be just like it was when you saved the metadata.

Metadata can also be loaded during a rave session to quickly make many changes to your data state, but keep all your graphs/controls etc. For example, you could load a metadata file that contains row colors to quickly change the row colors of your data set to some predefined state without manually selecting/recoloring the data.

The eight metadata file types

Note, RAVE comes with a sample data set that includes all eight of these files, so you can view those as examples

With the exception of the .rvefun file, the ordering of variables within the metadata files generally doesn't matter. When the files are read, the variables are not identified by their order, but by their names, so the names must match those listed in the initial data set exactly, including spaces and capitalization. Also, except for the .rvecur and .rvepoi files, you can omit any variables from any

metadata files to just use the usual default values that those variables would have been assigned if you hadn't loaded any metadata file.

*.rvefun Functions that act on this data set.

The .rvefun file contains a list of all functions that act on this data set. Loading a .rvefun file is the same as loading each of these functions using the model tab, but the rvefun file completely automates the process without asking for any user input.

Each VARIABLE that is an output of a function appears on a different row in this file. This file has no header row, the first variable appears on the first row. The format for each row is (with a tab/comma between each entry):

1) Variable name as you want it to appear in RAVE (must obey normal matlab rules except that it may contain spaces). If this variable is considered an "objective function" (i.e., if it's value is a function of preferences and targets), precede its name with an asterisk. Example: *MyObjective.
2) Full path to function file. This is either a .m or a .txt function. If you created a function by pasting text in or typing manually into the function editor, this will point to the m file that was created in your rave default directory.
3) The output of the function that corresponds to the variable named in (1). I.e., if you function has 5 outputs and this variable is the third output, this value will be 3.

4-n) The names of all variables that are inputs to this function. If the file specified in (2) is a .txt file, you can omit these (the variables will be determined by parsing the text file). If the file in (2) is a .m file, these are required. IMPORTANT: RAVE only supports feed-forward function analysis, so this list can only include variables that either appear as columns in your initial data set, or appear as the first entry in a previous row in this file. Although in theory you could just list every previous variable in this list, that will lead to excessive function evaluations within rave, as each input that is itself a function will be also be evaluated each time this function is evaluated (if all your functions are lightning fast, that might not be a big deal).

Note that if you have a single function that exports 5 variables, and you want to include all of them, you will have 5 rows where entries 1 and 3 are different for each row, and 2 and 4-n are the same.

Also note that the during the initial data loading, the .rvefun file (if it is found) is always loaded first, so the remaining metadata files can refer to variables defined in this file. If you don't have a rvefun file, the remaining metadata files can only refer to variables included in your initial data set (or that have otherwise been created before loading the metadata files).

*.rveval Variable Classes and List of allowable values

File format: Each variable appears in its own row, which has the following fields (separated by delimiter) If you omit any variables, they will be treated using the default rave settings.

1) Variable name (exactly as it appears in original data file)
2) Variable class, choose from: constant, continuous, discrete, logical, integer, or string

3-n) Remaining columns contain numerical values that define the allowable values for this variable. Depending on the data type specified in (2), the interpretation of these values is different. If the

variable named in (1) is a function output as defined by the .rvefun file, the values listed here don't define the allowable values, but define the min/max range of variability, which will be used to set initial ranges on things like profilers, contours, and colormaps.

Interpretation of columns 3-n is:

- For constant variables: the first value listed defines the value of the variable. Any other values in the row have no effect.
- For continuous or integer variables: the min/max of the values that appear here define the min/max allowable values for the variable. All other values in the row have no effect.
- For discrete or string variables: Each unique entry in the row is an allowed value for the discrete variable. Duplicate entries have no effect.
- For logical variables: If all entries in the row are 0, this variable can only be "off", if all entries are 1, this variable can only be "on". Otherwise, the values are ignored and variable can be on or off. Duplicate entries have no effect.

For variables modeled by functions: the min/max values that appear in the row define the min/max "known" values for the variable. All other values in the row have no effect.

These min/max values are used as the default limits when preparing contour plots, colormaps, derivative profilers, etc, however if in the course of creating those graphs, new values outside this range are found (for example, by using an optimizer) these ranges will automatically be widened to encompass the newly discovered bounds.

*.rvecon Constraints

This file contains constraints that act on the data set and functions defined in the rvefun file. Each constraint appears on its own row, and must be of the form (expression)<=0. Note that RAVE does not distinguish between <= and <. Each row consists of the following entries, separated by tabs:

(1) The constraint expression, excluding the "<=0". I.e., just the left hand side of the constraint inequality. Only variables in the initial data set or listed in the first column of the .rvefun file can be included in this expression.
(2) (optional) The constraint color, which will be used to draw this constraint whenever it appears on a graph. Must be in matlab color format, including the square brackets (i.e., "[1,0,0]" for red) This entry may be excluded for any/all constraints, and they will be drawn using the default color instead.

*.rvecur Current point

This file contains the definition of the current point, used as the starting point for derivative profilers, contour plots, etc. This file MUST contain a value for each variable that appears in the initial data set. If values are also included for other variables (such as those in the rvefun file) those values will have no

effect. Each row in this file defines the current point for one analysis, so if you include more than one row of values, multiple analyses will be created.

File format: First row contains all variable names included in the intial data set. You MUST include ALL of those variables.

Subsequent rows contain the current value for the corresponding variables named in the first row. If any of the variables are "strings", this value should NOT be the string itself, but its index if all the allowable strings (as defined by the .rveval file) were sorted alphabetically. For example, if a string variable can have values a,b,c, or d, and the "current point" value is "b", it should be recorded in this file "2", not "b". If there is more than one row of values, multiple analyses will be created. (See note at end of this section)

*.rveopt Target values and optimization info

This file contains information used by optimizers: the "best" value for each variable, and the "preference" for that variable. Any variables not listed in this file will be given a target of '-inf' (i.e., minimize) and a preference of '0'.

File format: First row contains variable names included in the initial data set or the .rvefun file. You can omit any of the variables you don't care about.

Second row contains the target value for each variable named in the first row, either '-inf' to minimize, 'inf' to maximize, or a numerical value for target matching.

Third row contains the preference for each variable named in the first row. These are used by some objective functions to form weighted aggregate objectives. These can be any numerical value between 0 (don't care at all) and 1 (care a whole lot). Ideally, the values in this row should sum to 1.

*.rvecol Variable Colors

Each variable in RAVE is assigned a unique color, so that each time it appears in a graph it is drawn in the same color. You can use this file to define those colors. Any variables you omit from this file will be assigned a color automatically be RAVE.

File format: First row contains variable names included in the initial data set or the .rvefun file. You can omit any of the variables you don't care about.

Second row contains a 3-element colorspec vector for each variable named in the first row. These vectors must contain the square brackets, e.g. enter [1,0,0] for red. All values must be between 0 and 1, i.e. in the typical matlab format.

*.rverow Row Colors, Visibility, and Selection

This file no header row, and has the same number of rows as your initial data set. Each row contains 3*n columns, where n is the number of analyses you wish to create. Within each analysis, the first column indicates the row color (an integer between 1 and 10), the second column indicates whether this row is

visible or not (1 for visible, 0 for invisible) and the third row indicates whether this row is currently selected or not (1 for selected, 0 for unselected). Usually you will want to start with all rows visible and unselected. Note that for this file, colors are defined by a single integer instead of a 3-element vector as for the .rveccl file. These integers are mapped to actual colors as defined by your raveprefs file. So all rows of the same color should have the same value, but you specify the actual color by changing your raveprefs file.

If you wish to have more than one analysis, simply repeat the 3 column pattern for each analysis you want to create, so for two analyses your column order would be: rowcolor1, visibility1, selection1, rowcolor2, visibility2, selection2. (See note about multiple analyses at end of section)

*.rvepoi Points of Interest

In addition to the data in your original data set, you can specify a set of "points of interest," which are special data points that are particularly interesting to you. For example, the result of an optimization might be a point of interest. These do not appear in the same manner as the regular data, but can be individually viewed/hidden on any graph, and each point has its own unique marker, not affected by the formatting of the current graph. NOTE: you probably want to keep this list fairly short.

The format of this file is identical to the initial data set: variable names in the first row and data values in subsequent rows. HOWEVER, each row (except the first) also has 5 additional columns, which contain (in the order listed below):

1st additional column: Marker shape. A single character of the following: s,p,h,o,v,^ ,+,.

(That last . is a legal option) These mean: square, 5-sided star, 6-sided star, circle, triangle, cross, point.

2nd additional column: Marker size. An integer value indicating the size of the marker. Typically in the 4-20 range.

3rd additional column: Marker fill color. A 3-element color vector, including the square brackets. E.g. [1,0,0] for red. OR the word 'none' (without quotes) to have no fill.

4th additional column: Marker edge color. A 3-element color vector, including the square brackets. E.g. [1,0,0] for red. OR the word 'none' (without quotes) to have no edge.

5th additional column: Edge width. An integer value indicating how thick to draw the edge. You probably want either 1 or 2.

A note about using meta-data with multiple analyses

Note, at this time, metadata only supports a single analysis

Two meta data files can be used to create more than one analysis: the .rverow file and the .rvecur file. If both files contain data for the same number of analyses, that many analyses will be created. If one file contains data for 1 analysis and the other file contains data for n analyses, n analyses will be created by

duplicating the data contained in the file with 1 analysis to fill all n analyses. If one data set contains data for n analyses and the other contains data for m analyses, where n<m, n analyses will be created by using only the data for the first n analyses from each file (and ignoring the remaining data).

A note about loading meta-data later in a RAVE session

If you load a meta data file during your rave session, it must contain the correct number of rows/columns contained in your data set at the time you load the file. For example, if you load a data set that contains 300 rows, and you have a meta-data .rverow file that also contains 300 rows, you won't be able to load that meta file if you've added new data to the data set after you loaded it (for example, by appending the results of an optimizer) so that the dat set now contains more (or less) than 300 rows.

Similarly, suppose you have a .rvecst file that contains constraints that are functions of a variable Y, which is not in your initial data set or in the .rvefun file you're already loaded. You cannot load this meta data file until you load whichever function outputs the variable Y, either by using the model tab or by loading another .rvefun file.

Preferences and Customization

Most options in RAVE are stored in the raveprefs.mat file, which you should keep in your normal MATLAB working directory (NOT in the RAVE directory).

To modify these preferences, navigate to the directory that contains raveprefs.mat, and type "load raveprefs" at the MATLAB command prompt. Then double-click on "raveprefs" in your Workspace window (or equivalently, enter the command "openvar raveprefs"). Use this table to make any changes you like. Note that preferences whose value is listed as "<1x1 struct>" contain additional preferences that can be viewed by double-clicking them.

After you make any changes, save your changes by entering the command "save raveprefs raveprefs".

(I know this is annoying, but it would be more annoying for me to have to make an edit preferences gui.)

Some of the more useful preferences are described below, followed by a list of all preferences.

Changing the Workspace background and gridline colors

The workspace background and gridline colors can be changed on the Configure tab. The minor gridline color is determined by averaging the major gridline color and the workspace background color.

Changing the Custom Colormaps

Many colors in RAVE must be chosen from predefined colormaps. RAVE has access to nine of the built-in MATLAB colormaps, three additional colorblind-friendly colormaps, and three custom colormaps. ~~To edit the custom colormaps, open any colormap chooser dialog (for example, on the Format Tab of a scatter plot), and click the "Modify" button to the left of the colormap you'd like to edit. (There will be an error beep, but no error actually occurs.) Note that the colors of the choose colormap dialog button's won't update until next time you open the dialog.~~

The colormap editing GUI is temporarily disabled because it used MATLAB source code that can't be distributed. Until I (or someone else) make a new one, colormaps can only be modified by…

Alternatively, you can edit the colormaps directly in the raveprefs.mat file. They are stored in raveprefs.colormap1, .colormap2, and .colormap3. Each colormap should be 5 colors long.

Changing the Fonts

Fonts can be changed from the configure tab, or by running raveinstall.m.

Unfortunately there is no way (yet) to change the graph axis label fonts, but this will be implemented in the future.

Preference Glossary

This is a list of all the preferences and their allowable values.

preferredposition – the default size/position that rave occupies on your screen

animation – a group of preferences dealing with exporting animations

animation.format – "gif" or "avi" to export animations in that file format.

animation.loop – "1" to loop a gif or "0" to have the gif only run once. Has no effect if animation.format is "avi".

animation.fps – number of frames per second to run animation at. Not sure if this affects "avi" but it definitely affects gifs. Set to "0" to make animation play in real time (which may be very slow).

colormap1-**colormap3** are custom colormaps. Enter a list of 5 colors and it'll interpret between them. These will show up as options when you try to change the colormap on something. I believe at the moment they must be exactly 5 colors long. These can also be customized within rave from the "Select colormap" dialog box.

colors is a structure with various object types beneath it. If you know how structures and colors work, you can edit these to customize rave's colors to your liking.

colors.tabbg - the color of the background for the control tabs on the left of the screen, and all objects that share this color. Default = [1 1 1] (white)

colors.tabfc - the color of the lines that draw the tabs. Default = [.5 .5 .5] (gray)

colors.wsbg - the color of the workspace background. Default = [1 1 1] (white)

colors.wsgrid - the color of the workspace grid lines. Default = [.8 .8 1] (lavender)

colors.frame - the color of the RAVE figure background. Default = [0.875 0.855 0.815] (beige)

colors.title – the color of "title" text. Default = [.4 .4 .4] (dark gray)

colors.text – the color of highlight text. Default = [.3 .6 .8] (bright blue)

colors.texttint – the color of secondary highlight text, currently used only by 2nd column of tab names. Default = [.6 .8 1] (light blue)

colors.buttonbg – the color of pushbuttons. Default = [.8784 .8667 .8471] NOTE: Changing this will cause buttons to revert to the old boxy look-and-feel on Windows Vista/7.

colors.togglebuttonbg – the color of togglebuttons. Do not set this to the default value for colors.buttonbg listed above in Windows Vista/7, or you won't be able to tell when the button is depressed.

colors.constraint – the default constraint color. Default = [1 0 0] (red). Can be changed for individual constraints by right clicking them.

colors.textbox – the default color for text in textboxes and calculated values. Default = [0 0 0] (black)

defaultdir – this must point to a folder that exists, and is also on your path. (add it to your path as above) This is where rave dumps any "behind the scenes" files you don't need to look at, so you should not just use your normal working directory. If you load a txt function, it gets written to an m file in here.

fonts – is a list of preferences that affect fonts.

fonts.textbox – the default font used when you create a new textbox or calculated value. Default = Verdana

fonts.title – the font used for titles. Default – Magneto (the rave font)

fonts.titlescale – a multiplier on the fontsize used by titles. If you change fonts.title, you may need to use this to shrink large fonts so the text stays within its intended bounds. Default =1

fonts.tab – the font used for tab names, and other text that shares the same font as the tab names. Default = Comic Sans MS

fonts.tabscale – same as fonts.titlescale, but for the tab font.

automovemode – currently unused

snaptogrid – set to "1" to snap to grid or "0" to not. Can be changed from Configure tab.

gridmode – set to "1" "2" or "3" to show no grid lines, major grid lines, or minor grid lines. Can be changed from Configure tab.

gridsize – an integer number. The number of pixels between major grid lines.

alwaysrasterize – set to "1" to rasterize graphs when they get deselected, or "0" to not. Can be changed from Configure tab.

interpreter – set to "tex" or "none" to interpret (or not) variable names as tex strings. This makes the symbols ^ and _ create super/subscripts, among other things. Can be changed from Configure tab.

defaultvariablecolors – either "accel" or "rave" to change the default order in which variable colors are assigned. May not actually do anything yet…

defaultcolormap – the name of the default colormap, but I think this doesn't do anything yet.

annotationcolor – doesn't do anything yet

defaultrowcolor – the index (see "rowcolors" below) of the color to use as the starting color for all rows when you load a data set. The default is 6, which corresponds to blue when using the default rowcolors.

rowcolors – a list of 10 integers that point to colors in RAVE's colormap, indicating the 10 colors that can be used as row colors. Don't edit this manually, instead ctrl-click on the row color buttons on the Format tab to change these.

alwaysupdate – set to “1” to instantly update all graphs as you drag a slider (computationally expensive) or “0” to only update the current graph, and update all other graphs once you let go of the mouse button. Can be changed from configure tab.

gpgpu – do not change this value. Should be “0”

gpusyntax – currently unused

logicaltrue and logicalfalse - are any two strings of text. This controls how logical variables are labeled on axes tick marks. Ex (‘Yes’, ‘No’)

tabscale – is a multiplier (Default = 1) to increase or decrease the size of the control tabs, for use on very large or small monitors. Very buggy, best not to change this for now…

autoselectbest –set to “1” to automatically select the “best” row in your data set any time the definition of “best” changes (e.g. you change preferences). Doesn’t work yet, but coming very very soon!

matrixspacing – an integer number of pixels to put between the graphs in a scatter plot matrix. Default =2.

profilerhspacing – an integer number of pixels to put between columns in a prediction profiler. Default = 0

profilervspacing – an integer number of pixels to put between rows in a prediction profiler Default. = 0

os is either “classic” “xp” “vista” or “windows7”. This preference slightly changes the window placement to attempt to make it look maximized. In the future I’ll have better options here. RAVE does its best to fill your screen, but it may be off by a few pixels. (See “nudge” below). This only matters when “ravemonitor” is set to “1”.

wsheight and **wswidth** are the width and height of the workspace in pixels. Change them here if you want to make it bigger (or smaller). I recommend making these values integer-multiples of the “gridsize” preference. At the moment you cannot change this while RAVE is running, so you’re stuck with whatever size you start with.

cellnumber is your cell phone number, if you want to be able to receive text alerts. NOTE: you must enter this as a string (put it in single quotes, no spaces or dashes)

cellcarrier is your cell phone carrier. Choose from: ‘verizon’ ‘att’ ‘cingular’ ‘alltel’ ‘boost mobile’ ‘nextel’ ‘sprint nextel’ ‘t-mobile’ ‘virgin mobile’. If you’re not on the list, let me know and I’ll try to add your carrier.

sendstringsasstrings is 0 or 1 to sends string inputs to your functions as strings, or convert them to numerical indices and send those instead. If you don’t have any functions whose input variables can be strings, you don’t need to worry about this. (Not sure if this works correctly yet)

Troubleshooting

Problem: There was an error while loading a MATLAB function from the model tab

If you get an error while loading a function, it will not break your RAVE session. Make sure the first line of the function is formatted correctly (n outputs, 3 inputs). If possible, fix the error and reload the function.

Problem: RAVE is slow

In general, RAVE should run reasonably quickly on an average 2010 desktop. If RAVE is running slower than usual, it is probably because you had MATLAB open for a while without using it, so that the functions it usually keeps in RAM got written to your pagefile. It should speed up as you begin using it again.

RAVE is known to be slow in the following cases:

- The first time you rasterize a graph during a session there will be a fairly long pause
- Sometimes when you click the “Manage Data Sets” button there is a pause
- Sometimes when you click the “New Graph” button it takes a while for all text on the gui to appear.

RAVE’s speed is mostly linked to the number of objects on the screen. Thus a scatter plot with 10,000 points will slow RAVE down more than a line with 10,000 points (10,000 objects vs 1 object). Use the “always rasterize” option on the configure tab to turn complex graphs into raster images.

If you notice RAVE running consistently slowly when you think it shouldn’t, please submit it as a bug report.