A guide to Eclipse and the R plug-in StatET

Longhow Lam

Version 01-12-2007

longhowlam@gmail.com

Contents

List of Figures

1. Introduction

The open source data analysis system R has gained a lot of attention, not only because it is freely available. It has proven to be an effective tool for data manipulation, data analysis, creating graphs and developing new (statistical) methods. R has welcomed many new users (from other systems) over the last years. This guide will not explain how to use R. See the R manuals [1] or the introduction to R document [2] that can be downloaded from www.splusbook.com for a detailed introduction to the R system. The purpose of this guide is to introduce the Eclipse Platform and the plug-in StatET for R users.

The base R system does contain an editor for writing R code. However, users will soon find out the limited capabilities of the internal editor. In principle any text editor can be used to write R programs (scripts). However, this document describes Eclipse for the following reasons:

- Like R, Eclipse is open source. It is freely available from www.eclipse.org.
- Although it is free, Eclipse does contain many features that a good editor should have.
- It is more than a text editor, it is an environment that allows you to easily maintain multiple R scripts in the form of projects.

In addition, the Eclipse Platform is an open environment that can be extended. It allows users to write so called plug-ins. Such a plug-in can customize the Eclipse environment for a certain programming language or for a certain task. The plug-in 'StatET' for R, created by Stephan Wahlbrink (www.walware.de/goto/statet) adds the following features:

- Run R code 'in Eclipse' by sending it to R.
- Syntax coloring of R key words.
- Insert predefined blocks of R code (templates).
- Supports writing R documentation files (*.Rd files).
- Run R CMD tools from Eclipse.

The remainder of this guide will first describe the basics of Eclipse and then the features of the StatET plug in.

1.1. Acknowledgments

I want to thank Stephan Wahlbrink for giving me Eclipse tips and tricks, descriptions of his StatET plug-in and for giving me feedback on this document.

1.2. Obtaining and installing the software

1.2.1. The Eclipse system

Eclipse is a Java program, so it needs the Java Runtime Environment (JRE) to run. You need JRE version 1.4 or above, which can be downloaded from http://java.sun.com. The Eclipse platform can be downloaded from the website www.eclipse.org, the latest version of the StatET works with Eclipse version 3.3. For windows, take some time to download the 139 MB zip file, 'Eclipse Classic' / Eclipse SDK. The installation is straight forward no installation wizard has to be run. Since Eclipse does not alter the Windows registery just unzip the zip file to any location you want, for example to C:/Program Files/ Eclipse.

Run Eclipse by starting eclipse.exe. During the start Eclipse will ask the user to choose a *workspace folder*. This folder is the default location that Eclipse will use will to store projects, see section 2.2.1.

1.2.2. The StatET plug-in

The StatET plug-in can be installed from within Eclipse.

- Go the 'Help' menu, then select 'Software Updates' and then 'Find and install'.
- In the 'Install/Update' dialog select 'Search for new Features to install' and click 'Next'.
- A new dialog will appear, click the 'New Remote Site' button and enter a name in the 'Name' field for example 'StatET' and enter in the 'URL' field:

  http://www.walware.de/eclipseupdates/

  A dialog as in Figure 1.2 will appear.
- Click 'Finish' and in the next dialog check the check box 'StatET' and click 'Next'. Then accept the terms in the license agreement and click again 'Next'.
- The next dialog will allow the user to the select a location for the plug-in. The default location is usually good enough, click 'Finish'.

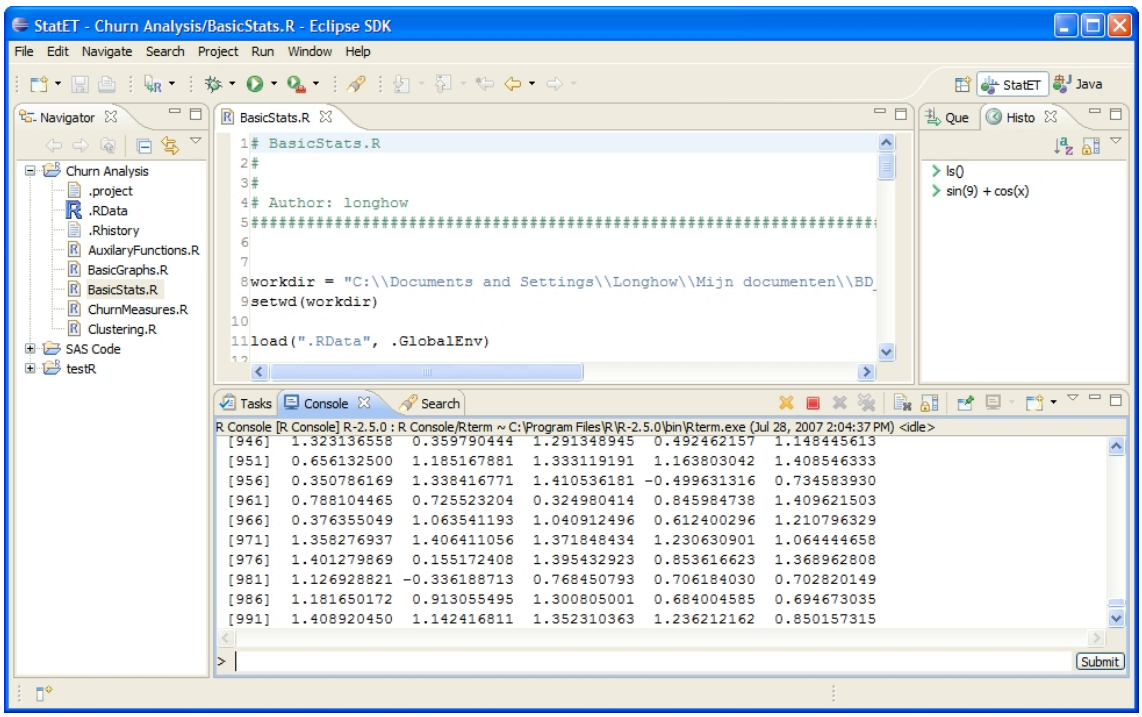

Figure 1.1.: The Eclipse workbench.

After the plug-in is installed Eclipse must be restarted. One way to see if the plug-in is installed is to go to the 'Help' menu and select 'About Eclipse' then click the button 'Plug-in details' and a list with all the plug-ins will appear as in Figure 1.3.

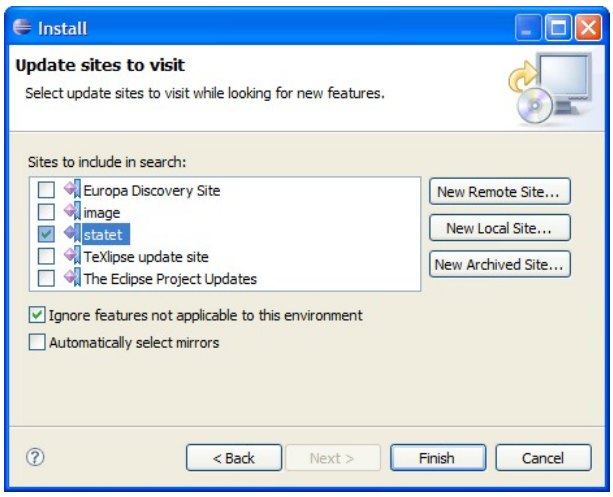

Figure 1.2.: The install dialog with the Statet plug-in ready to install.

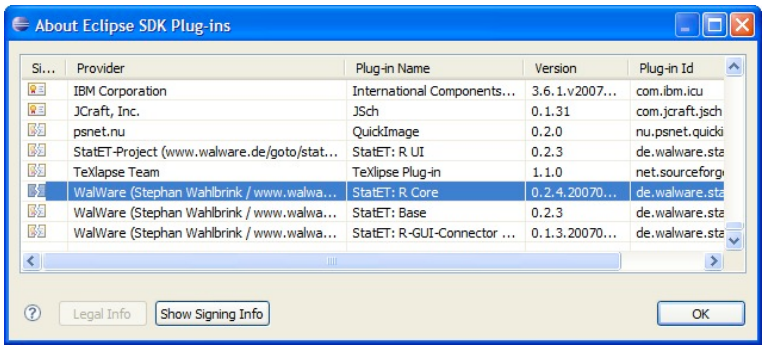

Figure 1.3.: A dialog displaying all the plug-ins that are currently available.

2. The basics of the Eclipse Environment

Before explaining the specific R features provided by the StatET plug-in, a few basic Eclipse features are explained. You will find these features very handy, not only for developing R programs. For a complete description of all the features of Eclipse, see the online Eclipse documentation [3].

2.1. The Workbench and Perspectives

The term *workbench* in Eclipse refers to the development environment on the desktop. It contains different windows, editors and views which are categorized in tab groups, as displayed in Figure 1.2.1. The views that are organized by tab groups are not static they can be dragged to different locations to suit your preferred layout. Select a tab and drag it to some other location.

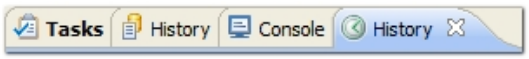

Figure 2.1.: A group of tabs that can be customized.

A workbench contains one or more *perspectives*. A perspective is a set of views and layout of tab groups, it is designed to facilitate the user to do a specific task. For example, the Java perspective helps the user write Java code by displaying certain views that are useful while editing Java code.

When the StatET plug-in is installed then the StatET perspective is available within the Eclipse workbench. To open this perspective: Go to the 'Window' menu and select 'Open perspective'. A list of perspective (as in Figure 2.1) will appear when 'Other' is selected. Now select 'StatET'.

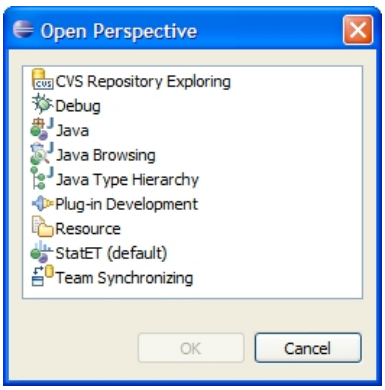

Figure 2.2.: The available perspectives in the Eclipse workbench.

2.2. Projects in Eclipse

2.2.1. Creating a project

The main structure in Eclipse to organize your work is a *project*. To create and edit R script files you need at least one project. To create a new project:

- Go to the File menu and select New > Project.
- A 'New Project' dialog as in Figure 2.3 will appear.

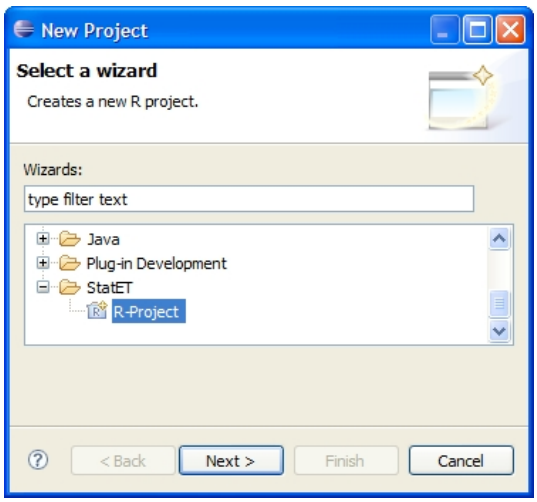

Figure 2.3.: The available project types in the Eclipse.

- Select 'R-Project' in the StatET folder, and click 'Next'.
- Type a name for the project and choose a location for the project. Eclipse will choose by default the 'workspace folder' as the location for all new projects. At

this location Eclipse will create a new folder (named after the project name), and it will put all relevant project information in this folder.

- A new project is created and it will be visible in the 'Project Explorer' view or 'Navigator' view, as displayed in Figure 2.4.

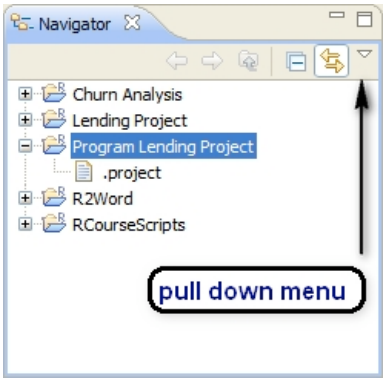

Figure 2.4.: The new project in the navigator view.

Once a project has been created it acts as a sort of container, new or existing *resources* can be added to the project. These resources could be any type of file, for example a jpeg picture an mp3 music file. In the case of an R project the resources are usually R script files, so plain text files with the extension .R. Note that you can have more than one project in the workbench. Repeat the above steps to insert another new project.

The .Project file

The `.project` file provides a complete description of the project, suitable for recreating it in the workbench if it is exported and then imported. You should not edit this file manually, it is even better to hide files beginning with a '.'. Go to the pull down menu in the 'Navigotor' view and select 'Filters...' then select the checkbox '`.*`'.

2.2.2. Adding new resources to a project

To add a new resource, for example an R script file, to the project:

- Right click on the project name in the 'Navigator' view and select 'New'.
- Select 'R-script file' and enter a name for the script file.

A new R script file is now created and is visible under the project in the 'Navigator' view. The file itself is located in the project directory, which in turn is located in the workspace folder. In case of doubt, right click on the file name in the 'Navigator' view and select properties. A dialog will appear showing the location of the file. With the creation of

a new R script file the StatET plug-in has put some comments at the beginning of the file. In Section 3.4.2 we'll explain how to modify these comments.

2.2.3. Adding existing resources to a project

To add an existing file (an R script file for example) to a project:

- Right click on the project name in the 'Navigator' view and select 'Import...'.
- In the dialog that appears, select 'File System' and click 'Next'.
- Browse to the directory with the file(s), and select the file(s) you want to add.

The file(s) will now appear in the project in the 'Navigator' view. It is important to realize that the files you have added are copied from their original location to the project directory. Any change you will make to the file in the project will not change the original file but will change the file in the project directory.

Another way to import a resource into a project is by means of drag and drop. From a Windows explorer window, files can be dragged and dropped into a project in the 'Navigator' view in Eclipse.

2.2.4. Adding Folders to a project

A project can contain folders. This will be useful when there are many files in a project, you can then use folders to organize the files.

- Right click on a project in the 'Navigator View' and select New > Folder.
- Enter a name for the folder.

Once a folder has been created you can insert new or existing resources into a folder, as described in the sections above. Alternatively, you can drag and drop existing resources from one folder to another.

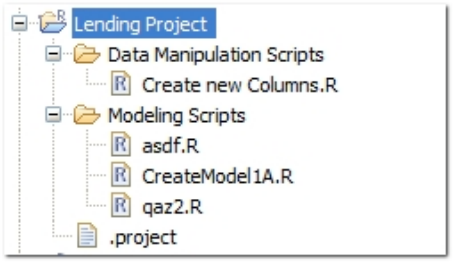

Figure 2.5.: Folders in a project.

2.2.5. Adding Linked Resources

Eclipse allows you to add so called *linked resources*. In the project a link to an existing file can be created as follows:

- Right click on the project name in the 'Navigator' view.
- Select 'New > File', a 'New File' dialog will appear.
- Click on the 'Advanced' button and check the 'Link to file in file system' check box.
- Browse to the file you want to link and click 'Finish'.

In the project an icon will appear with a small arrow, indicating that this is a linked resource, see Figure 2.6. Double click on the icon to open the file. Any change you will make will be applied to the file that you are linked to.

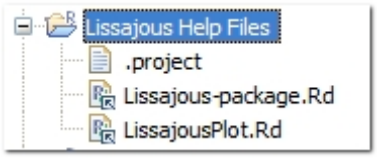

Figure 2.6.: Linked resources in a project.

Note that also folders and complete projects can be created as linked resources in a similar way.

2.3. Working with text files

To open a file in a project, double click on its name in the 'Navigator' view. Each type of file in the project can be associated with a different type of editor. R script files (*.R) are opened by default in an 'R script editor' in the workbench, and R documentation files (*.Rd) are opened with the 'R Documentation editor'. Right-click on the file name and select 'Open With' to open the file with another editor. This could be an external editor, in which case the file is opened outside the Eclipse workbench.

In the Eclipse environment you can open multiple (text) files, by default these (text) files are organized as different tabs on one tab group like Figure 2.7. In this setting only one text file is open and visible. You may want to view two or more (text) files at the same time, then drag a tab outside its tab group. You could create a setting like in Figure 2.8.

Figure 2.7.: A tab group with different text editors.

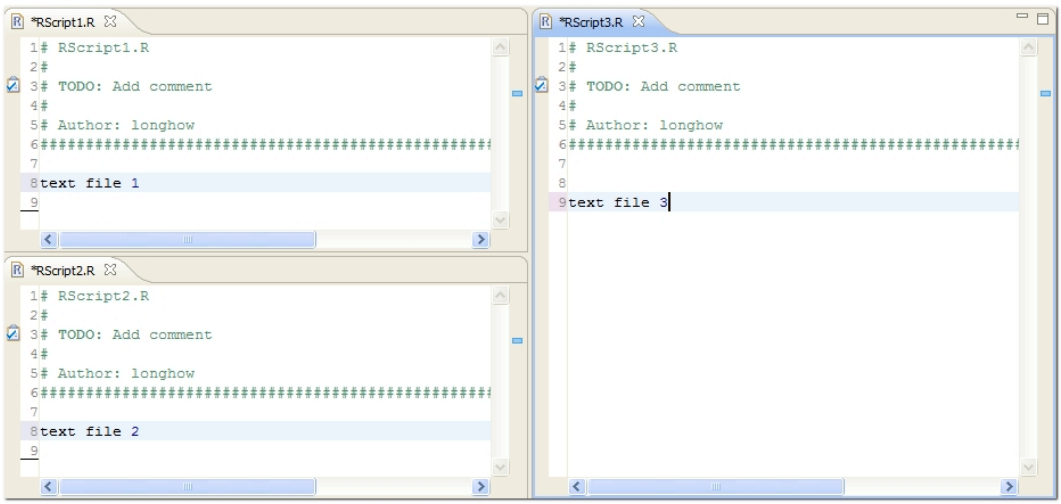

Figure 2.8.: Multiple text files visible.

2.3.1. Line numbers, Quick diff and Word completion

Two useful features to show are the line numbers and the quick diff. In an open text editor in the workbench right-click in the small gray strip on the left side of the editor (the gutter). In the context menu that appears select 'Quick Diff' and 'Show Line Numbers'. The 'Quick Diff' identifies the lines in the editor that have changed since the last save. For example, lines that have been deleted (a small black line), or lines where new text is entered (line numbers are shaded). Hover over these areas with your mouse to see the old text.

Word completion is a handy feature. In a text editor you can complete some typed in characters to a word occurring in all currently open text editors. Type in some characters and press `alt+/`.

2.3.2. The History of a file

A handy feature in Eclipse is the local history of a file. Each time you save a file, Eclipse will log the changes you have made. You can easily recall previous versions of the file.

- Right click on a file of a project in the 'Navigator' view.
- Select 'Compare With > Local History'.
- The history view will appear as in figure 2.10.

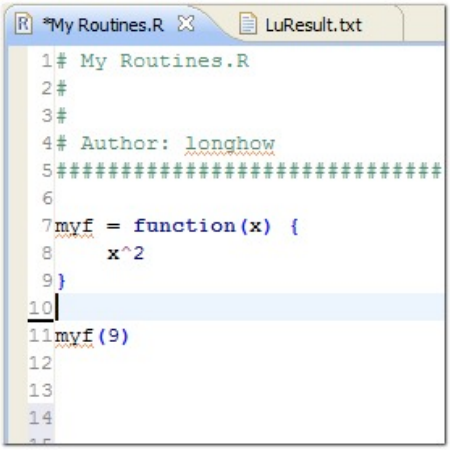

Figure 2.9.: The gutter with line numbers and 'Quick Diff'

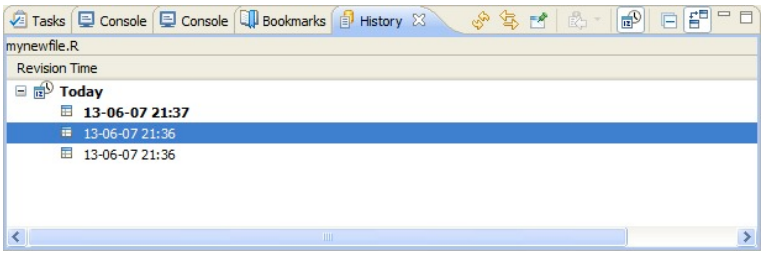

Figure 2.10.: The history view that lists all the versions

In the history view the current and all the previous versions of the file are displayed. To open a previous version, right click on a version and select 'Open'. This will open the previous version in read-only mode. To compare a previous version with the current version right click and select 'Compare current with Local'. A compare editor will open where the two versions are displayed. See figure 2.11.

2.3.3. Comparing files

Instead of comparing different versions of one file, Eclipse also allows you to compare two different files with each other. Select two files in the 'Navigator' view, right click and select 'Compare with each other'. A compare editor as in figure 2.11 will display the two different files and their differences.

2.3.4. Bookmarks and task tags

Bookmarks are useful when you want to remember a certain location in a file. To insert a bookmark in a file:

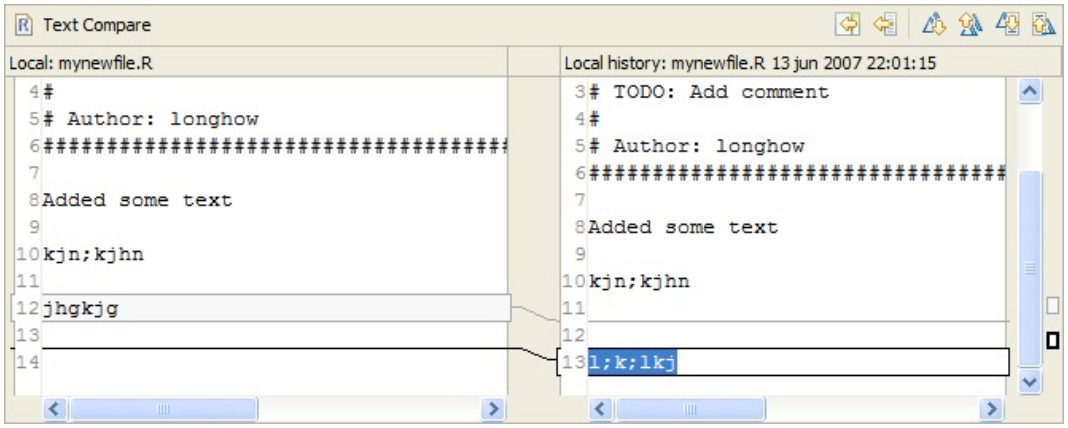

Figure 2.11.: The Compare Editor allowing the user to browse through changes between versions

- Right click in the gutter on the line where you want a bookmark.
- Select 'Add Bookmark...' and enter a name for the bookmark.

Light blue bookmarks symbols will appear in the gutter, as in figure 2.12. To get a view of all bookmarks go to the 'Window' menu and select 'Show View > Bookmarks'.

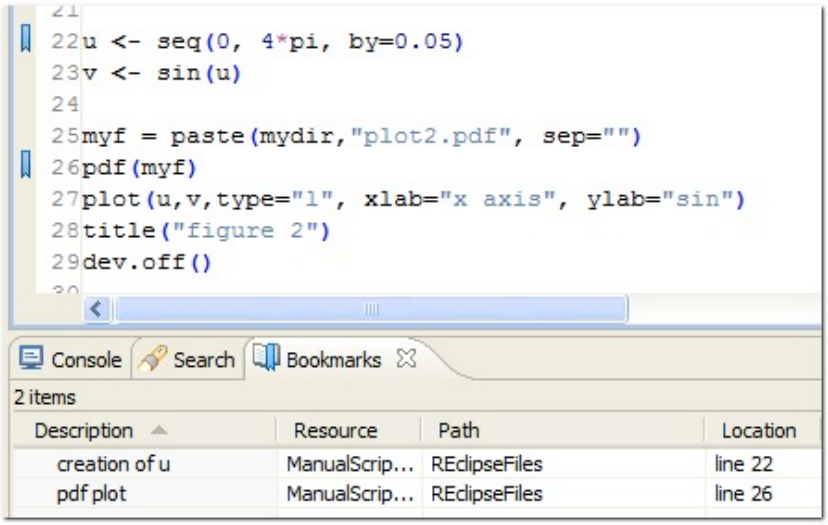

Figure 2.12.: Bookmarks in a text file

It is also possible to add a bookmark that refers to an entire file. Select a file in the project in the 'Navigator' view, then go to the menu 'Edit > Add Bookmark...'.

Tasks allow the user to be remembered of certain uncompleted tasks that need to be done. Proceed as follows:

- Right click in the gutter on the line where you want to add a task.

- Select 'Add Task...' and enter a name for the task.

A little task icon will appear in the gutter, as in figure 2.13. To get a view of all tasks go to the 'Window' menu and select 'Show View > Tasks'.

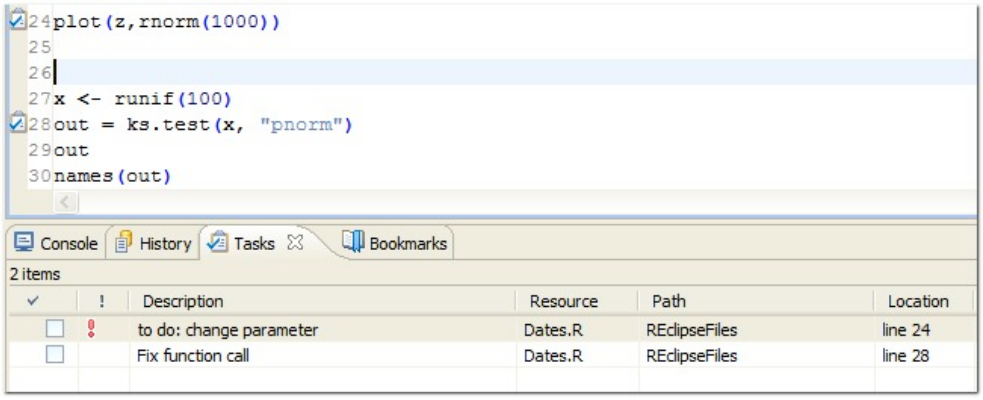

Figure 2.13.: Tasks in a text file help you to remember to finish uncompleted tasks

2.3.5. Text/Code folding

The text editor for R files in the StatET plug-in supports code folding when writing functions. By default code folding is automatically enabled. That means when you write an R function, the editor will automatically put the code between the opening bracket end the closing bracket of the function in a *folding area*. See Figure 2.14.

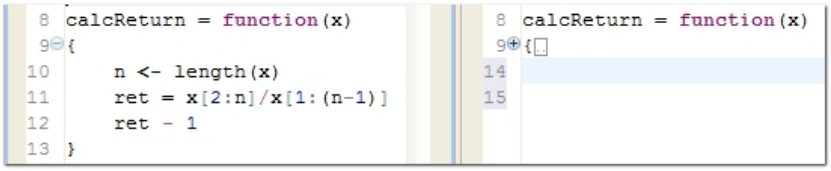

Figure 2.14.: Code in a folding area and code that is collapsed.

In the gutter of the text editor you will find small nodes on the left of the opening bracket. Click these nodes to fold or unfold the corresponding code. Right click somewhere in the gutter to open a context menu. In this menu you can:

- Enable / disable folding.
- Collapse all folding areas in the editor.
- Expand all folding areas in the editor.

2.4. Searching for text

2.4.1. A single file

To search for a text in an open file in the editor, press ctrl-F. A 'Find/Replace' dialog as in figure 2.15 will appear.

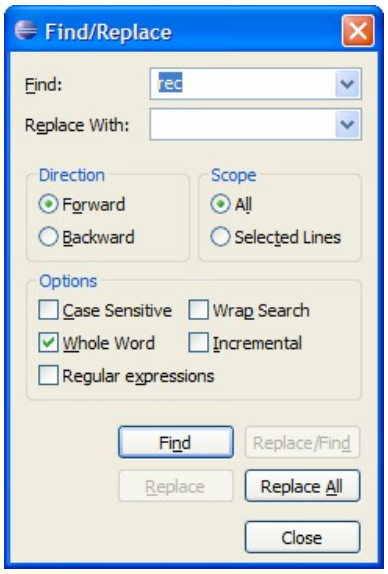

Figure 2.15.: The Find / Replace dialog

Type a search string and click 'Find'. Unselect 'Whole Word' to search for the search string as part of a larger string. So searching for 'tan' will also find rectangle. Eclipse allows you to specify search strings using regular expressions. With regular expressions you can formulate advanced search strings. It is beyond the scope of this document to explain regular expressions in detail. Just a few examples.

- Select the 'Regular expressions' checkbox. And enter a regular expression.
- `gr[ae]y` will find the words grey and gray but not groy.
- `gr.y` will find any single character, so gray, groy and gr5y.
- `gr.*y` will find gray, graaxsdy or gry.
- The expression `[0-9]+` will find any number, so 458, tr5w and 786jhf.

An alternative to searching for strings in an editor is the *Incremental Find* and the *Reverse Incremental Find*. With the editor in focus:

- Press ctrl-J, you will enter the incremental find mode.
- This is indicated in the status bar at the bottom of the Eclipse window.

- Type in some characters, Eclipse will try to find these characters.

If the characters are found these characters are highlighted in the text editor, otherwise the status bar will display 'Not found'. Press ctrl-J to move forward from the current cursor position and try to find the characters again. Use ctrl-shift-J to search upwards in the text.

2.4.2. Multiple files

To search for a text in multiple files go the menu 'Search' and select 'File...'. A dialog as in figure 2.16 will appear. Or alternatively press ctrl+H or click on the 'Search' button on the main Eclipse tool bar.

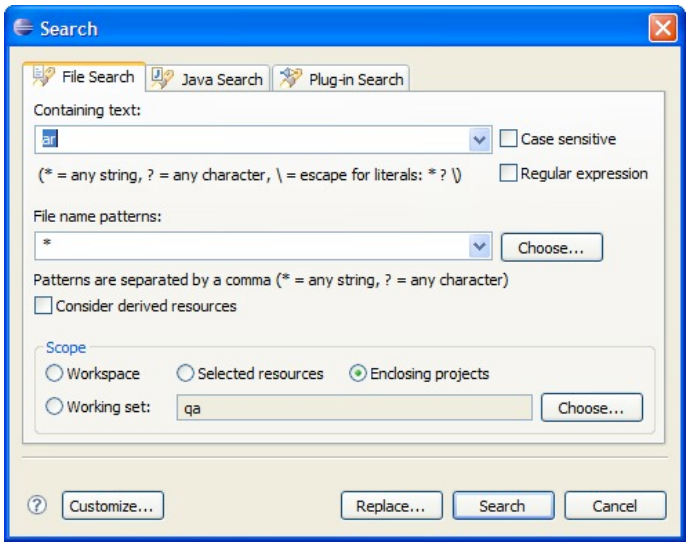

Figure 2.16.: The search dialog allows the user to search in multiple files

Proceed as follows:

- Enter a search string in the 'Containing text' field. If this is a regular expression select the 'Regular expressions' checkbox.
- Enter a pattern in the 'File name Patterns:' field to specify in which type files you want to search. For example *.R for R script files, or just * for all files.
- Select the scope to specify in which files you want to search.
  - Workspace, all files in all the projects of your Eclipse workspace.
  - Selected resources, only the files that are selected in the 'Navigator' view.
  - Enclosing projects, all the files in the current project.
  - Working set, a set files that you can create by clicking the 'Choose' button.

The results of the search are displayed in the 'Search' view, as in Figure 2.17.

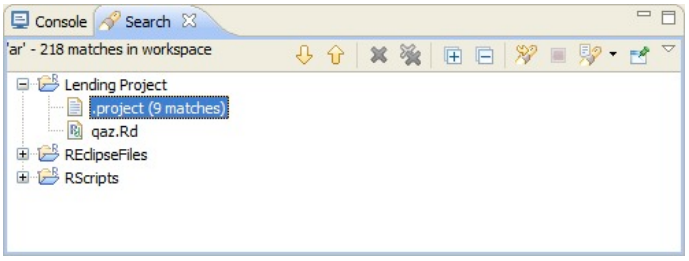

Figure 2.17.: The search view that displays locations of the search string.

In the 'Search view' you can double click a certain file to open it in the text editor. The found search strings are highlighted in the editor and in the gutter you will see small arrows. When the 'Search view' is selected you can see a little up and down arrow. Click these arrows move the cursor to the next or previous match. To refresh the 'Search view' press F5 and the same search will be done again.

2.5. Eclipse preferences

To set the Eclipse preferences, open the 'Preferences' dialog as displayed in Figure 2.18 by going to the menu 'Window > Preferences...'. The preferences are structured in a tree structure on the left side of the dialog, there are just too many preferences to mention here. Only a few of them are high-lighted in this section, experiment with the preferences dialog to find out more.

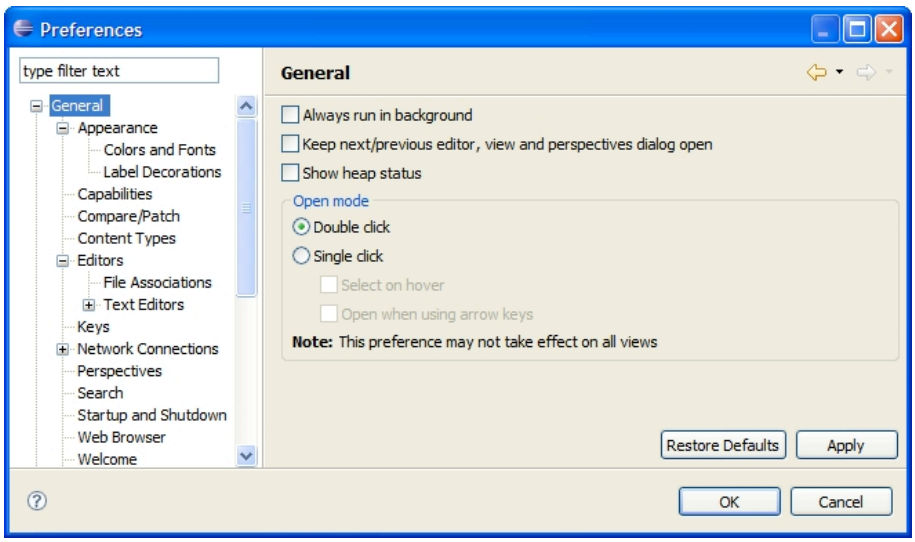

Figure 2.18.: The preferences dialog.

Appearance

Unfold 'General' in the preferences tree and select 'Appearance', here you can set the colors and fonts for the different elements of eclipse. For example the colors and fonts used in the text file compare environment.

Shortcut keys

In Eclipse many keys are already assigned to a certain task or command. In the preferences dialog unfold 'General' in the tree and select 'Keys'. Then in the dialog select a command to change or set the key binding. For example, if you are used to F3 in SAS, set key binding for the command 'Run Selection/Current Line in R' code to F3. As a shortcut to the shortcut keys press Ctrl+Shift+L anywhere in the workbench. Eclipse will display a pop up window with a list of all active shortcut keys.

The local history

The settings for the 'Local History' can be found under the General > Workspace settings. You can increase the low values. For example, set 'Days to keep files' to 365, the 'Entries per File' to 10,000 and 'Maximum file size' to 10. These new settings should be enough to track a complete year of changes.

Auto Run

The left hand side of the preferences dialog shows the main categories, you can find 'Auto Run' under the 'Run/Debug' folder. The 'Auto Run' preference enables you to start certain tools at start up. For example if you want to automatically start the R console you should check the check box 'Enable run at startup of:'.

3. The StatET plug-in

3.1. Configuring R

If you want to run R code in an R console inside Eclipse or you want to run the R CMD tools, then you need to set up an R environment first. To set up an R environment go to the menu 'Window > Preferences...', the 'Preferences' dialog will appear as in Figure 3.2. Expand the 'StatET' node in the tree and expand 'R Interaction', you will see 'R Environments' as in Figure 3.1.

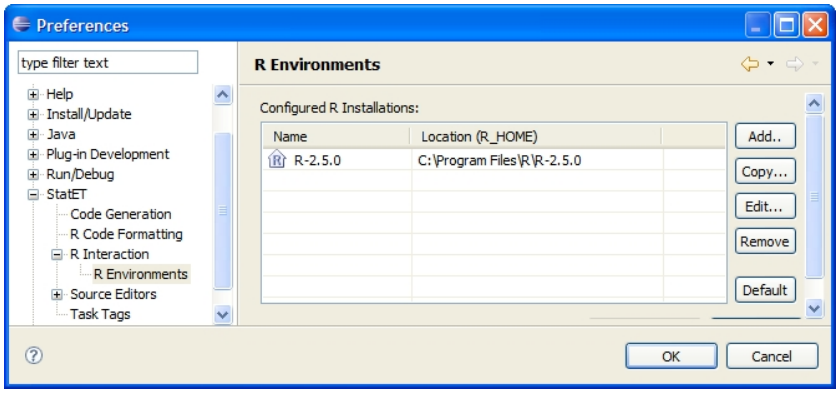

Figure 3.1.: Setting up an R environment

Perform the following steps:

- Click Add, a dialog appears where you can enter a name for the specific R configuration, for example: R-2.5.0.
- Enter the location of the R system, for example 'C:\Program Files\R\R-2.5.0'.

You can add more R configurations. This can be useful when you have installed different versions of R, and you want to run some R code in a specific version. One configuration must be assigned as default R configuration, select an R configuration that you have created and click the button 'Default'.

Once you have set up the R environment you need to choose how to interact with R. You can either send R code from the editor in Eclipse to an R console inside Eclipse or to an external RGui application.

Go to the menu 'Window > Preferences...', the 'Preferences' dialog will appear as in Figure 3.2. Expand the 'StatET' node in the tree and select 'R Interaction'. Then choose an interaction type.

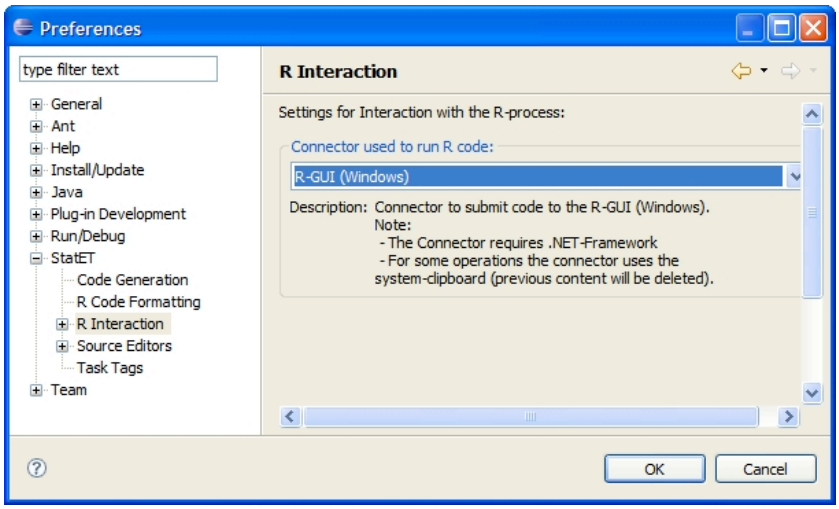

Figure 3.2.: The preferences dialog displaying the preferences for Eclipse and the plug-ins.

To send code using an external RGui application you will need to start that application first. If you want to use the R console as described below, select 'New Console inside Eclipse'.

3.2. The R Console in Eclipse

To send code to an R console inside the Eclipse environment you must also start the console first. However, this console must be started from within Eclipse. Proceed as follows:

- Go to the menu 'Run > Open Run Dialog...'. A dialog as in Figure 3.3 will appear.
- In the run dialog select 'R Console' and click the 'New launch configuration' button.
- Enter a name for the configuration, for example 'R Console'.
- In the 'main' tab there is a field options where you can enter R start up options. For example, `--silent` causes R to start up without printing start up messages.
- Go to the 'R Config' tab and select an R configuration.
- Also enter a working directory for R. You can leave it blank then the R system will choose the default working directory. Or you could enter variables like `${project_path}`. Then in this case the working directory for the R session will

be set to the selected project in the 'Navigator' view. The Appendix lists Eclipse variables that you can use.

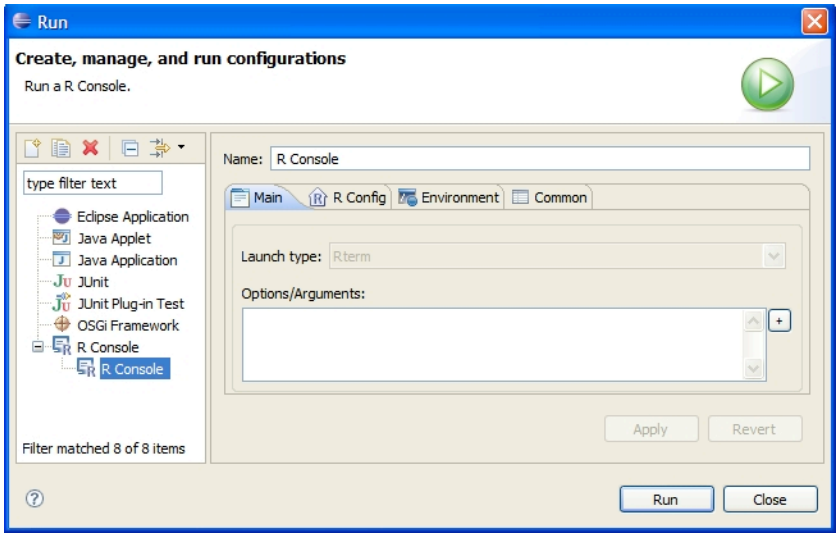

Figure 3.3.: The run dialog to configure the internal R console.

Now click on 'Apply' to save the configuration and click on 'Run' to run the R console. A view as in Figure 3.5 will be visible. Note that sometimes you may not run the R console, the 'Run' button is grayed. This can happen for example when no project is selected but you have set R to start with `${project_path}` as the working directory.

Once the R Console has been setup, the next time you start Eclipse it can be ran again from the 'Run As' button menu on the main toolbar of Eclipse as high lighted in Figure 3.4. Or alternatively you can automatically run the R console at the startup of Eclipse, as described in section 2.5.

Figure 3.4.: The 'Run As' button menu high-lighted on the toolbar.

The R console consists of two parts, as Figure 3.5 displays. There is an upper part displaying R output of statements you have entered. The lower part is a input field used to enter an R statement. When your cursor is in this input field there are a few key combinations that you can use, given by Table 3.1

3.2.1. Running R code

Using either the RGui or the internal R console, there are several ways to run R code in an Eclipse Editor. If the editor with the R code has the focus go to the 'little R button' on the Eclipse toolbar and select from one of the following options.

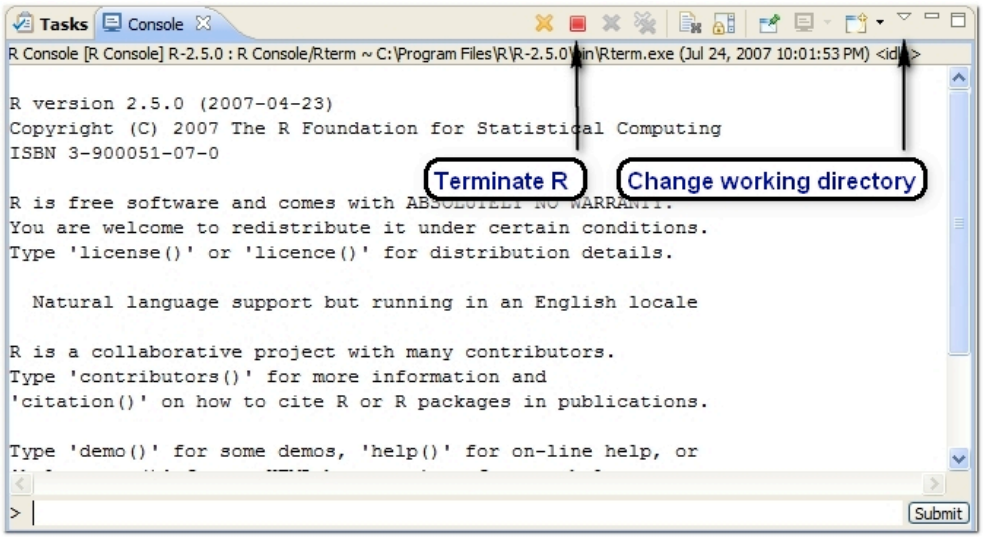

Figure 3.5.: The R console inside Eclipse, with an extra line to enter R statements.

Figure 3.6.: Little R button on the Eclipse toolbar.

Run R Script in R submitting directly The complete contents of the text editor is submitted to R. Depending on your preferences, the code (statement per statement) will be sent to an external RGui application or to the R console in Eclipse.

Run R Script in R via 'source' The complete contents of the text editor is entered in R through the `source` command in R. In contrast to the above method, only one command is run in R: `source( <selected text file> )`

Run R selection/current line in R If a block of code is selected then only that block of code is sent to R. If nothing is selected, only the current line (the line of code where the cursor is) is sent to R.

Run R selection/current line in R and go to the next line Does the same as the above but moves the cursor to the next line in the text editor. This might be useful to run some block of statements line by line.

Run Function Definition in R This option is not accessible from the menu but is accessed by the short cut key ctr+R, ctr+F. In a text editor, if the cursor is in a function definition then use the short cut key to submit the function definition. StatET

Key combination	Console effect
(arrow) Up and Down	Scroll through previously entered statements
Ctrl+alt+Up/Down	Scroll through previously entered statements, but only display the statements that start with the letter combinations in front of the cursor.
Shift+(arrow) Up	Scroll output one line up
Shift+(arrow) Down	Scroll output one line down
Shift+PageUp	Scroll output one page up
Shift+PageDown	Scroll output one page down
Shift+Ctrl+Home	Scroll output to the start
Shift+Ctrl+End	Scroll output to the end
F1	Displays a view with help options

Table 3.1.: Some useful key combinations in the R console

will recognize the beginning and the ending of the function. The user does not need to select the whole function and use 'run R selection' to define the function.

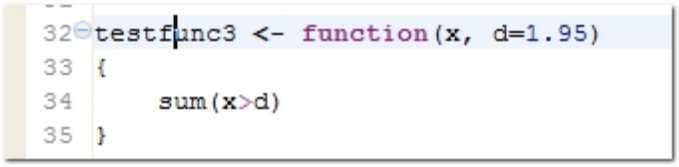

Figure 3.7.: Run function definition. Place the cursor in a function definition and use Ctrl+r, Ctrl+F

For convenience, the above ways to run R code can be accessed by short cut keys as described in Section 2.5.

3.3. The Cmd History and Queue views

The 'Cmd History' view records every statement that you have sent to an R console inside Eclipse. If it is not opened, you can open it by going to the menu 'Window > Show View > Other...'. A dialog will appear where you can unfold 'StatET' and then select 'Cmd History'. Click OK and the History view will be part of the workbench, as displayed in Figure 3.8. Use the Up and Down arrow keys to scroll through the list of statements. When a statement is selected in the History view, it can be submitted again by hitting the enter key or right click and select 'Submit Again'. Hover the mouse cursor over an entry in the 'Cmd History' view, a small time stamp will appear that indicates when the command was submitted.

The Queue view, as in Figure 3.9, lists all the statements that are submitted to R and have not finished or executed yet. It can be opened in the same way as the History view.

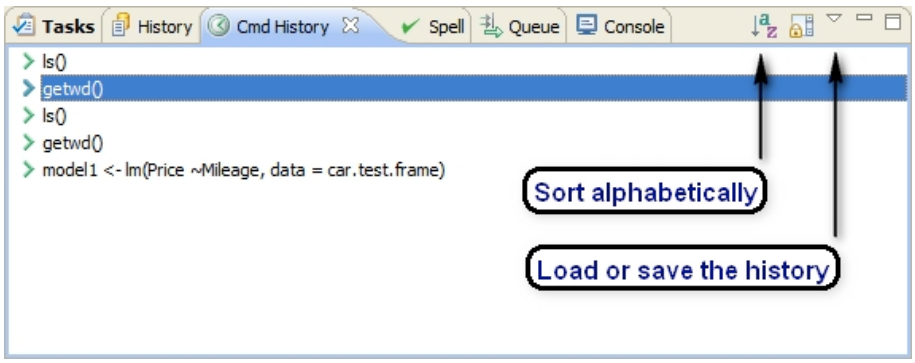

Figure 3.8.: The History view with previously entered or submitted statements.

The upper part displays the statement that is currently being executed by R. The lower part displays the other statements that are going to be executed. The 'Pause button' on the Queue view can be used to pause the execution. The current statement will be executed and finished but the other statements are paused until the 'Pause button' is clicked again.

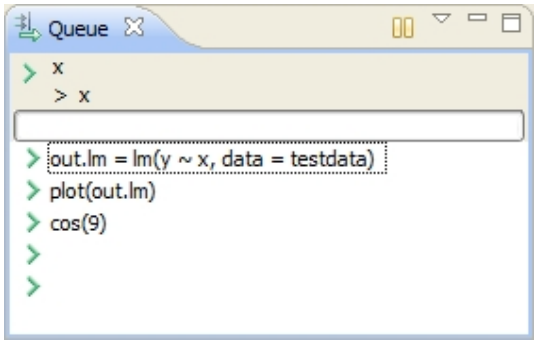

Figure 3.9.: The Queue view displaying statements that are waiting to be executed.

3.4. The StatET preferences and their corresponding features

In the (StatET) preferences dialog as displayed in Figure 3.2 there are more preferences you can set. The next sections will high light some these settings that may be useful in writing R programs.

3.4.1. Code templates

The StatET plug-in allows you to predefine code templates that can be inserted when you edit R script files. There are already some predefined code templates, to add new

code templates:

- Go to preferences dialog and select: StatET > Source Editors > R Templates.
- A dialog with the already available R templates will be displayed, as in Figure 3.10.
- Click the 'New...' button to insert a new template. A dialog as in Figure 3.11 will pop up.

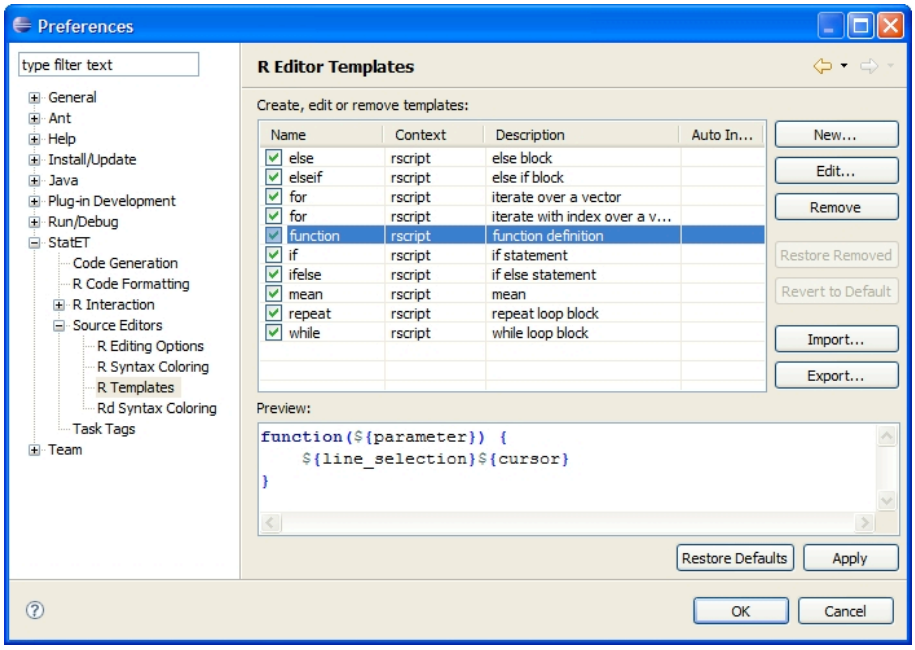

Figure 3.10.: The available code templates.

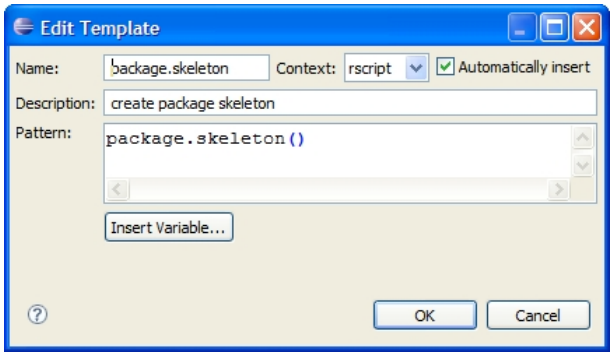

Figure 3.11.: Inserting a new code template

- In the 'New Template' dialog fill in a name for the template.
- Fill in a description for the template, and type in the actual code that will be inserted.

- The 'Insert Variable' button can be used to insert useful variables inside the code. Such as the current date or time.

To insert a code template in an editor press Ctrl+space to see a list of all predefined code templates. Note that only the code templates will be displayed whose name start with the same characters as the characters directly before the cursor. For example, type '`myf = fu`' in an editor and press Ctrl+space. Only the code templates starting with 'fu' are displayed.

3.4.2. Code generation

In addition to inserting predefined code templates at certain locations in an R script file, you can have Eclipse insert certain code at the beginning of an R script file when you create a new R script file. Go to preferences dialog and select: StatET > Code generation. Edit the code for new R script files or new R documentation files.

3.4.3. Syntax coloring

The editor for writing R programs and R documentation files supports syntax coloring. Different colors can be assigned to different elements of the R language. Go the preferences dialog and select StatET > Source Editors > R Syntax Coloring. An 'R Syntax Coloring' dialog as in figure 3.12 will appear.

Select a code element in the 'Elements:' field to change the color settings for those code elements.

The StatET plug-in allows the user to modify the default identifiers that will be colored, as well as adding custom identifiers that also should be colored in the editor.

- Go the 'Preferences' dialog, and unfold the 'StatET' node.
- Unfold 'Source Editors' and select 'R Identifiers Groups'.
- On the right-hand side of the dialog you can now select a group.
- Now add or remove keywords / symbols. Click OK when done.

The keyword / symbols you have added will now also be highlighted / colored according to the syntax coloring settings.

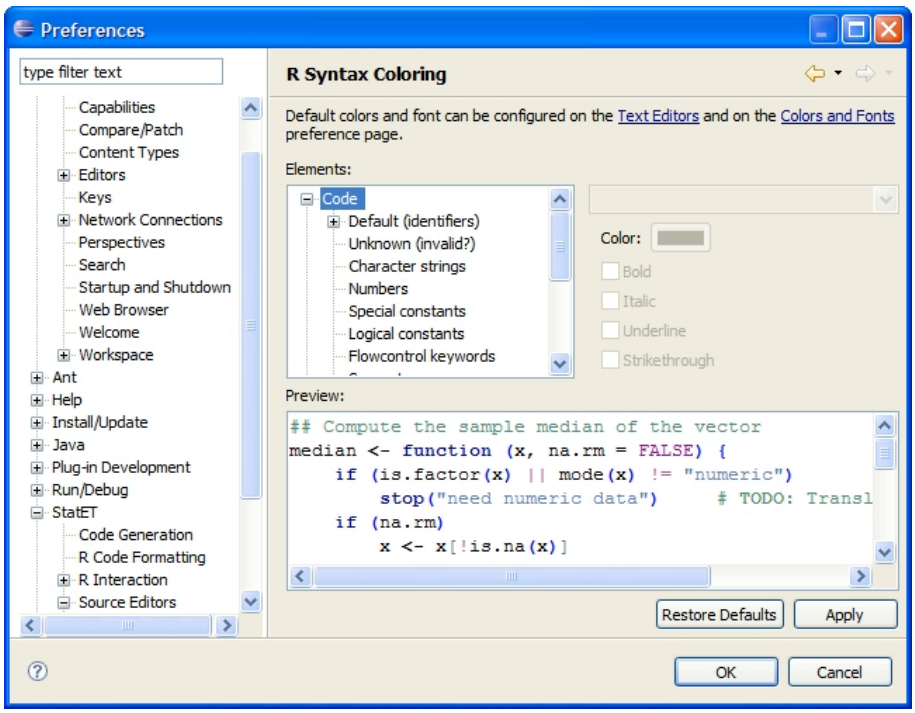

Figure 3.12.: The preferences dialog for setting syntax coloring

3.4.4. R Code formatting

The setting in the 'R Code Formatting' dialog, see Figure 3.13, allow the user to control the indentation. I.e. the amount of whitespace (spaces and tabs) at the beginning of a new line. You can choose to either use leading tabs or leading spaces.

Within a block of {} you can choose to indent again when starting a new {} block. For example, the following code:

```
test = function(x){
  if(x > 0){
    z = 2*myf(x)
```

starts with a one level indent after the starting of the `if` construct. This is controlled by the setting 'Indent within '{}' blocks:' in the R Code Formatting dialog.

When a block of code does not have the indentation that corresponds with the current settings, you can correct the indentation so that it matches the settings. Select the block of code and use 'Correct Indentation' in the 'Source' menu. For example, the following code:

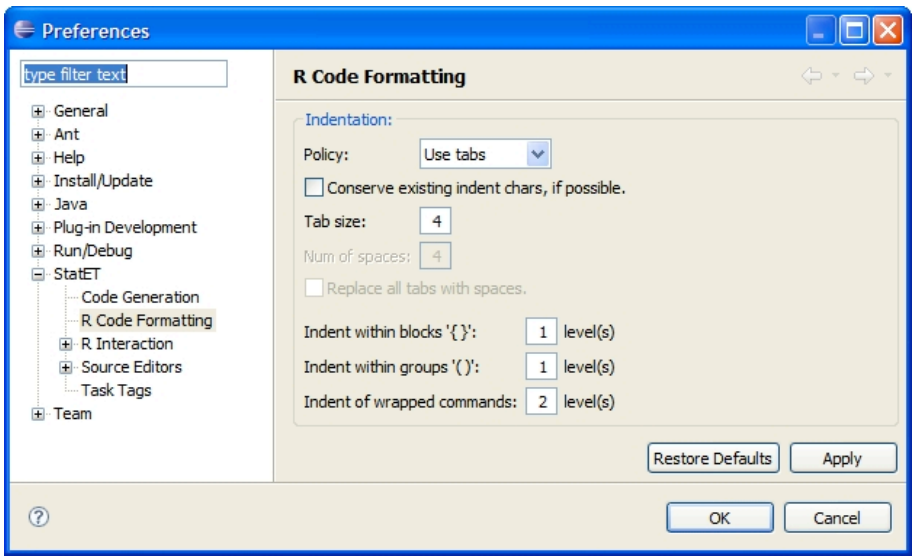

Figure 3.13.: The preferences dialog for setting syntax coloring

```
test = function(x){
if(x > 0){
z = 2*myf(x)
}
}
```

would be corrected to

```
test = function(x){
  if(x > 0){
    z = 2*myf(x)
  }
}
```

3.4.5. R Editing Options

The settings you will find in the 'R Editing Options' dialog allow you to modify the *smart insert* settings.

- Correct indentation when pasting. If this is selected then the indentation of code that is pasted from the clipboard to the R editor will automatically be corrected to match the current indentation settings.
- Automatically close brackets. If this is selected then the R editor will automatically insert a closing bracket when an opening bracket is typed into the editor.

3.4.6. Some StatET shortcuts

In addition to the many shortcut keys that the Eclipse environment already provides, the StatET plug-in provides some useful shortcuts as well. These shortcuts can be changed to your own preferences as described in section 2.5.

Key combination	**meaning**
Ctrl+R, S	Run R code via source
Ctrl+R, ctr+D	Run R code by submitting it directly
Ctrl+R, 1	If a word is selected in the editor, run help('*selected word*')
Ctrl+Shift+#	Add documentation comment. These are double # in front of each selected line
Ctrl+Shift+C	Toggle comment. place or remove single comment symbols #.
(Shift)+tab	Shift a selected block of text to the right or to the left.
Alt+Shift+Up	Select enclosing element
Alt+Shift+Left	Select previous element
Alt+Shift+Right	Select next element
Alt+Shift+Down	Restore last selection

Table 3.2.: Some useful StatET shortcuts

3.5. Writing R help files

The R system allows the user to write R documentation files (help files) for user written functions. These files have extension Rd, and need to be written in a special markup language that resembles the LaTeXmarkup language. Eclipse together with the StatET plug-in is a convenient environment to write and maintain these help files. As with R syntax, key words in the markup language for R help files are color highlighted. There are three ways to start the process of writing R documentation files.

Start from a new file

- Right click on a project in the 'Navigator view'.
- Select New > Other > StatET > R Documentation file.
- Choose a name for the help file.

The new Rd file will appear in the project and is visible in the text editor.

The `prompt` function in R

If you have an R function, the `prompt` function in R can generate a template R documentation file from the function. Submit the following R commands:

```
myrdfile = "Myfunc.Rd"
prompt(myf,myrdfile)
```

Then R will create the Rd file 'myfunc.Rd'. In Eclipse, right click on a project and select 'Import...' to import the Rd file into the project for further editing.

The `package.skeleton` function in R

When you create an R package (see for example [2]) with multiple functions, then the R function `package.skeleton` creates a skeleton for the R package. Among other things, this skeleton contains a directory with template Rd files for each function in the package. You can then import these Rd files in Eclipse for further editing and maintenance. See for example Figure 3.14.

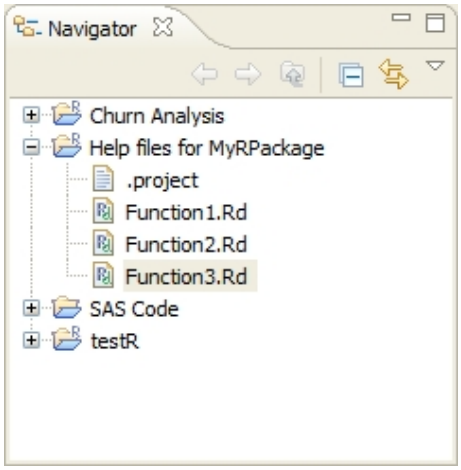

Figure 3.14.: A project consisting of R help files.

3.6. R CMD tools

From within Eclipse you can run R command tools, the StatET plug-in has a customized 'External Tools' dialog to configure R command tools. Go to the menu 'Run > External Tools > Open External Tools Dialog...', a dialog as in Figure 3.15 will appear. The next sub sections give some examples on how to configure and run these tools.

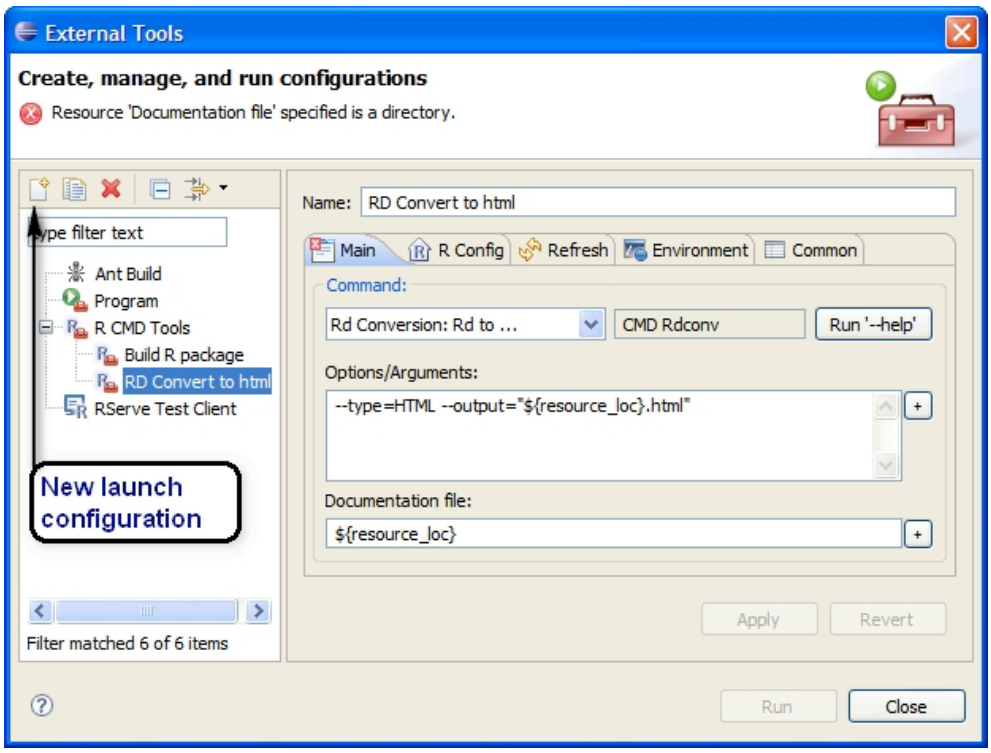

Figure 3.15.: The External Tools dialog to set R command tools.

3.6.1. Converting Rd files

R documentation files can be converted to LaTeXor HTML by running the tool `Rcmd Rdconv`.

- Select 'R CMD Tools' in the External Tools dialog and click 'New Launch Configuration' button.
- Then on the right-hand side of the dialog, enter a name for this configuration. For example, 'Rd Convert to html'.
- Select the specific R command, in this case 'Rd Conversion: Rd to ...'.
- In the Option/Argument field enter:

  ```
  --type=HTML --output="${resource_loc}.html"
  ```

  The `type` argument specifies the conversion to html, the `output` argument specifies the location and name of the html file. In this case, we took the eclipse variable `$resource_loc`. But to make sure the R command tool does not overwrite the original Rd file, an extra '.html' is placed after the name. If there are spaces in the name you must enclose the name with double quotes.
- In the 'R Config' tab select the R environment and the working directory.

- In the 'Refresh' tab you can select 'Refresh resources upon completion'. When you have selected to generate the html file in the same directory as the Rd file, Eclipse will then refresh the 'Navigator' view so that the newly generated html file is visible.
- It is not necessary to set the tabs Environment and Common.

After the settings are done click 'Apply' and close the 'External Tools' dialog. To convert an Rd file, select it in the Navigator view or put the text editor in focus. Open the external tools dialog again and select the 'Rd Convert to html' and click 'Run'. Note that once an external tool has run, the Eclipse toolbar has a quick link to the external tool. In our case you can run the 'Rd Convert to html' tool by selecting it from the toolbar.

Figure 3.16.: Quick link on the toolbar to external tools.

3.6.2. Building an R package

Assuming that R and the necessary tools for building an R package are installed, the StatET plug-in can be configured to build an R package from within Eclipse. Proceed as follows:

- Open the External Tools dialog, and click the 'New launch configuration' button.
- Enter a name for this tool, say 'Build R Package'.
- Select the command 'Add-on Packages: Build'.
- The `R CMD Build` tool can be run with specific options. These options are entered in the 'Options/Argument:' field. For example, to build a pre-compiled binary package, enter the option `--binary`.
- Specify the package directory. This should be the directory containing the necessary ingredients to build an R package, so for example the 'DESCRIPTION' file, the sub directories: 'data', 'man' and 'R'. If these ingredients are imported in an Eclipse project you can set the package directory to `${project_loc}`.
- In the 'R Config' tab select a working directory. If you fill in for example `${project_loc}` then the resulting R package will be created in the directory of the project.

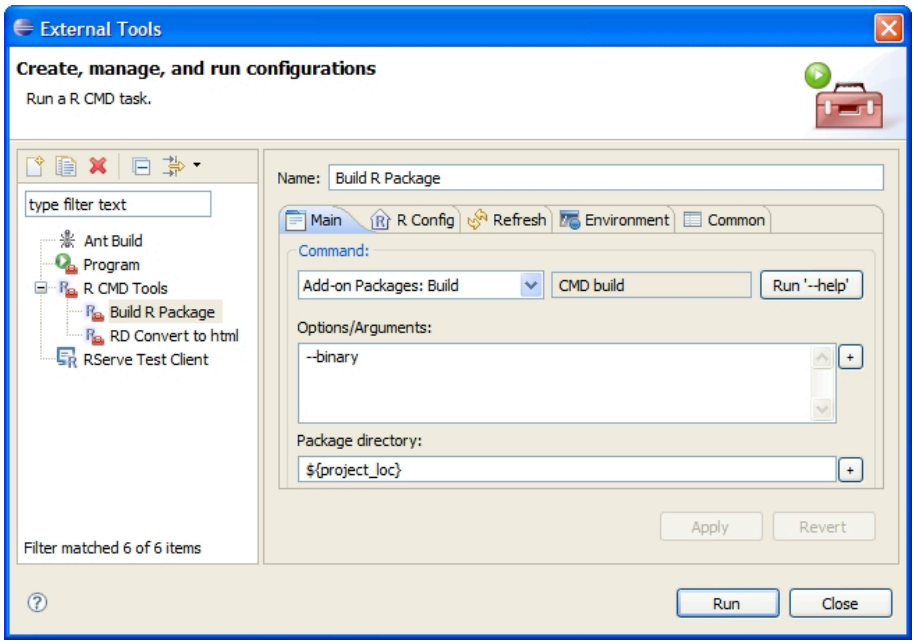

Figure 3.17.: Building an R package from the external tools.

A. Some Eclipse variables

This appendix lists some of the Eclipse variables that you can use in the 'Run' and 'External Tools' dialogs. The following text is taken verbatim from the Eclipse help file.

From the Eclipse help

Let's assume your Eclipse workspace directory is `c:\eclipse\workspace` and you have two projects, MyProject1 and MyProject2. The first project, MyProject1, is located inside the workspace directory, the second project, MyProject2, is located outside the workspace directory at `c:\projects\MyProject2`. Lets look at how the variable examples below will be expanded when an external tool is run, if the resource /MyProject2/MyFolder/MyFile.txt is selected.

Variable Examples	**Expanded Results**
`${workspace_loc}`	c:\eclipse\workspace
`${workspace_loc:/MyProject1/MyFile.txt}`	c:\eclipse\workspace\MyProject\MyFile.txt
`${workspace_loc:/MyProject2/MyFile.txt}`	c:\projects\MyProject2\MyFile.txt
`${project_loc}`	c:\projects\MyProject2
`${project_loc:/MyProject1/MyFile.txt}`	c:\eclipse\workspace\MyProject
`${container_loc}`	c:\projects\MyProject2\MyFolder
`${resource_loc}`	c:\projects\MyProject2\MyFile.txt
`${project_path}`	/MyProject2
`${container_path}`	/MyProject2/MyFolder
`${resource_path}`	/MyProject2/MyFolder/MyFile.txt
`${project_name}`	MyProject2
`${container_name}`	MyFolder
`${resource_name}`	MyFile.txt
`${build_type}`	none

Bibliography

[1] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

[2] Longhow Lam, *An introduction to R*. www.splusbook.com, 2007.

[3] IBM Corporation and others, *The Eclipse documentation*. http://www.eclipse.org/documentation, 2007.

Index