

## StatSoft White Paper

# Integration Options and Features to Leverage Specialized R Functionality in *STATISTICA* and *WebSTATISTICA* Solutions

### Contents at a Glance

Overview and Summary .....	3
Basic Architecture and Features of R Support in <i>STATISTICA</i> .....	7
R Language Extensions: Passing Data to R and Retrieving Results .....	12
Executing R Scripts from within <i>STATISTICA Visual Basic</i> .....	16
R Support in <i>WebSTATISTICA</i> .....	20
Creating R-based <i>STATISTICA Data Miner</i> Nodes .....	23
Integrating R into ( <i>Web</i> ) <i>STATISTICA Enterprise</i> .....	24
Final Comments, and Some Caveats .....	27

For more information, see [www.StatSoft.com](http://www.StatSoft.com)

## Table of Contents

Overview and Summary .....	3
Basic Architecture and Features of R Support in <i>STATISTICA</i> .....	7
COM Interface to R Environment .....	8
R Integration Support Macro (R.svb) .....	8
R Scripts as Native <i>STATISTICA</i> Macros.....	8
Retrieving Results.....	10
R Language Extensions: Passing Data to R and Retrieving Results .....	12
ActiveDataSet .....	12
Spreadsheet (FilePathOrName) .....	12
RouteOutput (RObject, [SpreadsheetName], [SpreadsheetHeader]) .....	13
Uses (RPackageName, [AttachImports]) .....	15
Executing R Scripts from within <i>STATISTICA Visual Basic</i> .....	16
R Scripts are <i>STATISTICA</i> Macros.....	16
Passing Parameters to R Scripts: Collection Object.....	17
Managing Script Results.....	19
More Examples.....	19
R Support in <i>WebSTATISTICA</i> .....	20
<i>WebSTATISTICA</i> is a Powerful R Server .....	21
Off-Loading R Scripts from <i>STATISTICA</i> to <i>WebSTATISTICA</i> .....	21
Creating R-based <i>STATISTICA Data Miner</i> Nodes.....	23
Integrating R into ( <i>Web</i> ) <i>STATISTICA Enterprise</i> .....	24
Creating R-based Analysis Configurations.....	24
Calling R Scripts from SVB Analysis Configurations .....	26
Summary .....	26
Final Comments, and Some Caveats .....	27
Error Handling .....	27
Strengths and Limitations .....	27

## Overview and Summary

R is a programming language and environment for statistical computing (<http://www.r-project.org>).

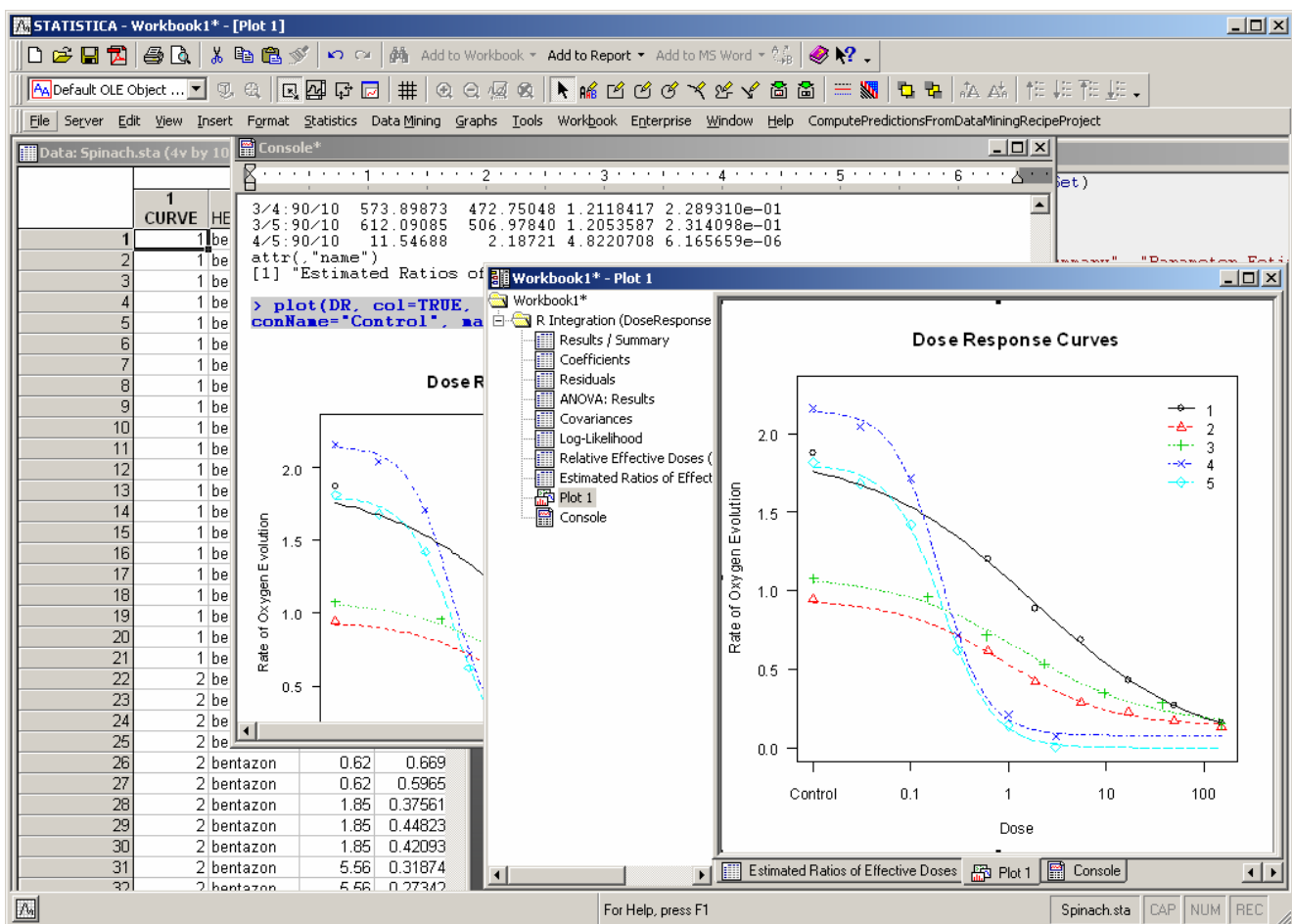
The R environment and its source code are freely available under the GNU GPL license; precompiled binaries exist for Microsoft Windows, Unix and Mac OS platforms. R uses a command line interface, but several graphical user interfaces are also available.

R is highly extensible: users can submit libraries ("packages") implementing a set of functions, usually for a specific area of their expertise/research. R community maintains several centralized repositories that make hundreds of such packages readily available to all users over the Internet. Many of these packages cater specifically to highly specialized audiences with particular data analysis needs.

The goal of this document is to provide a detailed description of new features that make the diversity and power of R fully available to users of all StatSoft Solutions. Furthermore, these features allow users to combine the unique capabilities of both *STATISTICA* and R platforms.

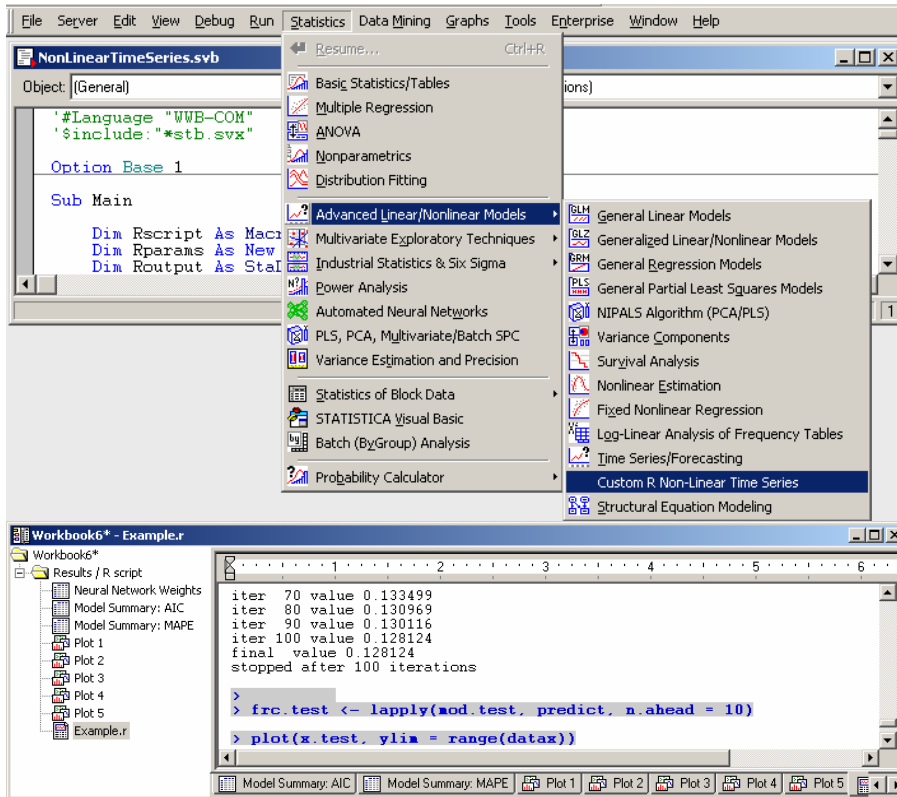
Starting with *STATISTICA* Version 8 MR3:

- Native R scripts can be run directly within *STATISTICA*, *WebSTATISTICA*, *STATISTICA Enterprise*
- R output can be retrieved as native *STATISTICA* spreadsheets and graphs, and managed via highly flexible *STATISTICA Workbook* containers

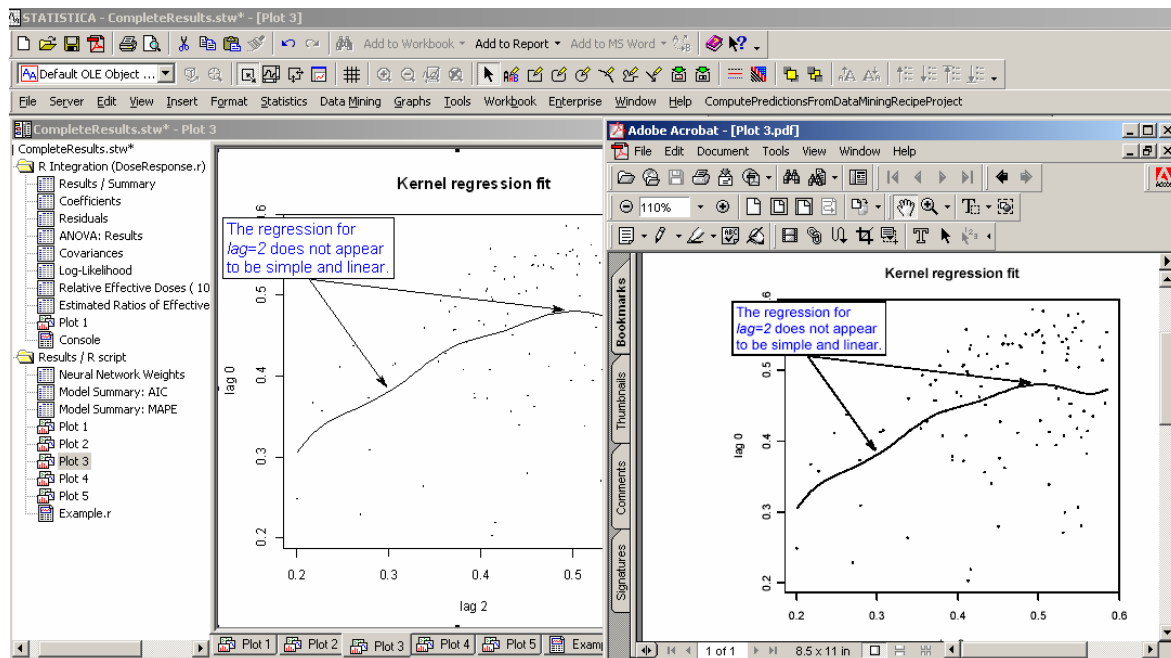


Thus, enterprises can now use the specialized routines and capabilities of R with *STATISTICA*, *STATISTICA Enterprise* and *WebSTATISTICA* servers to:

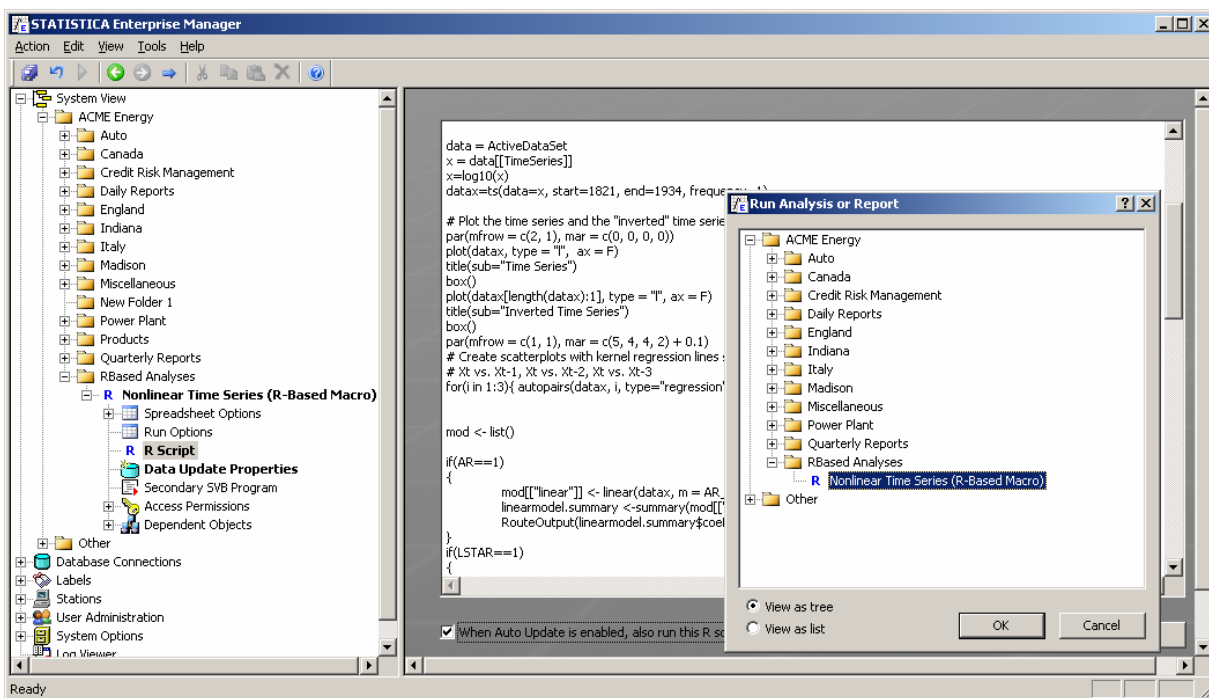
- Add new R-based “modules”



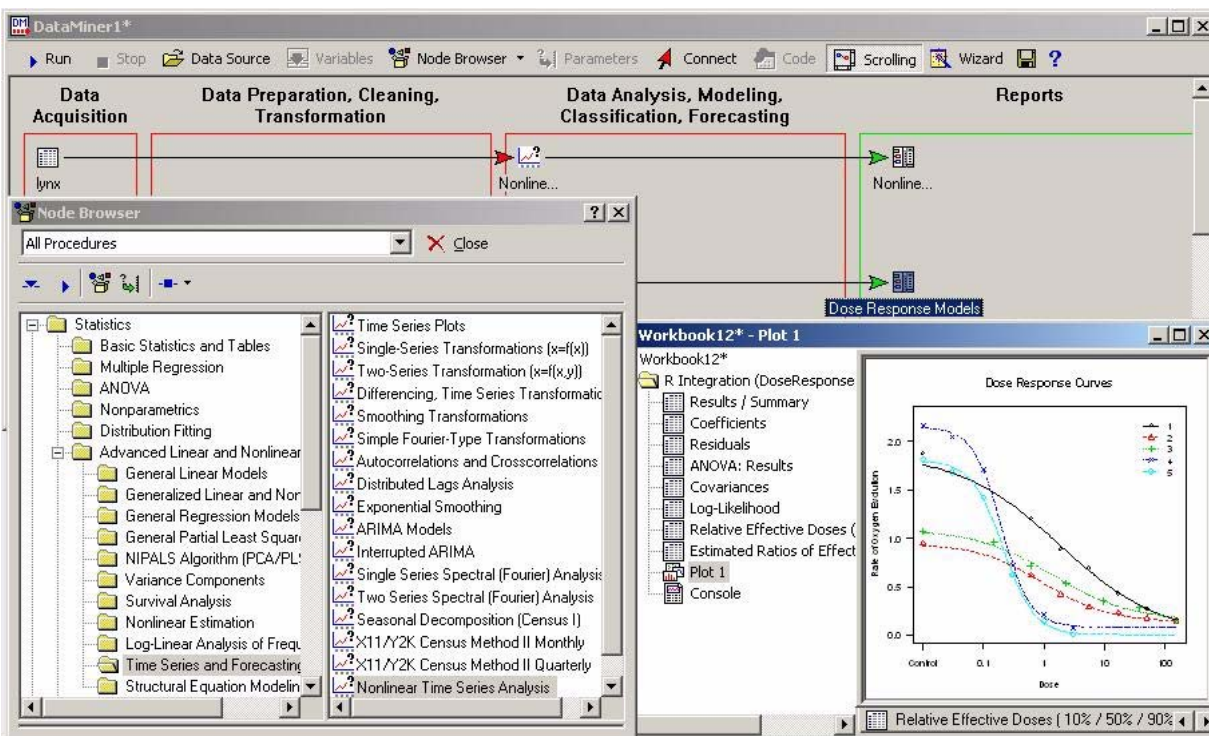
- Leverage *STATISTICA*'s superior graphics, flexible Spreadsheets, and convenient Workbook containers for various document types to handle output from R



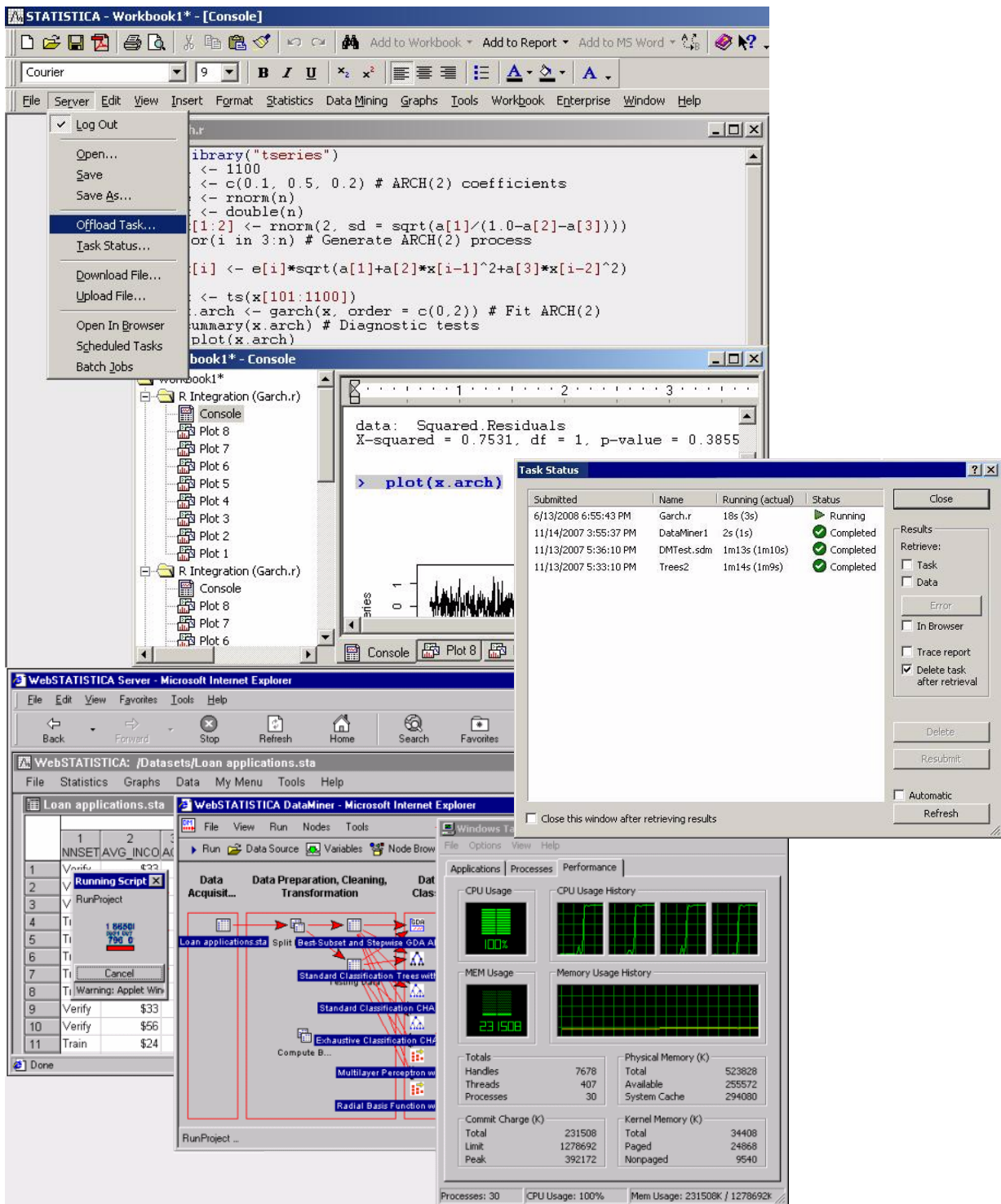
- Integrate R into *STATISTICA Enterprise* to make specialized R functionality available as reusable analysis templates to users not familiar with the R language, in a secure, role-based enterprise analysis system



- Add R-based analytic nodes to *STATISTICA Data Miner*, thus leveraging all R capabilities inside (Web)STATISTICA Data Miner Workspaces



- Build scalable R servers using *WebSTATISTICA* to handle security, load balancing, and to take advantage of multiple-processor servers to run R for demanding and/or validated enterprise applications



## Basic Architecture and Features of R Support in *STATISTICA*

R support in *STATISTICA* was designed to create an integrated *STATISTICA-R* platform that will enable users to run R programs ("scripts") directly inside (*Web*)*STATISTICA*, so that users can take full advantage of the specialized capabilities available in R.

The R Integration environment in *STATISTICA* was specifically designed to:

- Allow users to run R scripts "as is", and retrieve the results into *STATISTICA* reports
  - all R console output is copied into the report; R commands are highlighted
  - plots generated by the script are automatically embedded in the report as scalable images
  - these plots are also replicated as *STATISTICA* graphs (scalable "metafile" images are placed inside these graphs), thus enabling annotation using powerful graphical facilities in *STATISTICA*; the graphs can then be printed or exported into a variety of image formats
  - the reports can be edited, printed, saved as PDF files
- Provide R language extensions (keywords) for R scripts run from *STATISTICA* environment that:
  - transfer data from *STATISTICA* spreadsheets into R data frames
  - extract tabular data from R variables into *STATISTICA* spreadsheets; these "results" spreadsheets (as well as all graphs produced by the script) are returned according to Output Manager settings ("routed"), e.g. placed into *STATISTICA* workbooks
- Execute R scripts as native Macros from within *STATISTICA Visual Basic (SVB)* programs
  - scripts can be parameterized with a Collection of objects (numbers, strings, arrays, additional R code or overridden R functions, spreadsheets) that are mapped to R variables accessible to the script; this approach provides fine-grain control over script's behavior in repeated runs or when used as the backend for custom *STATISTICA* modules
  - by default all script output is routed by *STATISTICA* Output Manager; scripts may also be executed using a method that instead returns its output as a Document Collection, giving developers an easy way to extract specific analysis results that could be used for further processing, e.g. as input data for following analyses in *STATISTICA* or in R, or for graphing.

Taken together, these enhancements not only enable users to run R scripts directly in the *STATISTICA* desktop environment, but also provide a way to embed specialized R functionality into their SVB programs, custom interactive analysis modules, *Data Miner* nodes and *Enterprise* analysis configurations, or to offload such scripts to *WebSTATISTICA* for server-side processing.

Enterprise customers have the opportunity to integrate R into validated *STATISTICA Enterprise* solutions (e.g. for FDA-regulated industries) or to provide a very powerful *WebSTATISTICA*-based R Server environment.

**Developing and debugging R scripts.** One of the use cases that the R support in *STATISTICA* was *not* designed for is to supply a complete R development and debugging environment. The console application and tools supplied with standard R installation perform those functions very well, and are already familiar to R users and developers.

## COM Interface to R Environment

In order for R Integration in *STATISTICA* to work, the R environment has to be installed on the same computer (or on the server, in case of *WebSTATISTICA* or *Enterprise* solutions). The latest version of R environment can be obtained from the CRAN website (<http://cran.r-project.org>).

To access the R environment, *STATISTICA* uses the R COM Server (<http://rcom.univie.ac.at>) library distributed under GNU Lesser GPL license.

This library is automatically installed on the system during *STATISTICA* installation process if the R environment is detected, but can also be installed manually, either by using a standalone R COM Server installer or through the *STATISTICA* installer (“Integration with R” installer feature, also referred here as R Integration support).

R COM Server library provides a simple yet powerful COM (Component Object Model) interface to the R environment. This interface can be used directly by SVB programs in *STATISTICA* – an example of such a use case is included in *STATISTICA* distribution, located in `Examples\R\Dose Response` folder (open `Direct Interface To R via COM.stw` and run the embedded SVB macro).

But usage of such an interface directly by the end-users is very ineffective, sometimes unproductive, and usually inflexible. It may also significantly degrade the overall performance of interactions with R if done incorrectly.

Therefore the following architectural extensions have been added to the *STATISTICA* platform to provide a seamless and effective R Integration experience for end-users (the example mentioned above also demonstrates the significant reduction in the efforts required to implement the same analysis using the new built-in features: in *STATISTICA*, simply open and run `DoseResponse.r`).

## R Integration Support Macro (R.svb)

The installation of *STATISTICA* includes a *STATISTICA Visual Basic (SVB)* program file called *R.svb*. This file contains all the support code required to manage the interactions with R through COM. Specifically, when an R script is executed from inside *STATISTICA*, it is pre-parsed by this support macro, which handles *STATISTICA*-specific R language extensions (keywords), transfers spreadsheet data, script parameters and the R script itself to the R environment, runs the script, manages error conditions, and also handles the results returned from the R script, ensuring that they are properly transferred back to (*Web*)*STATISTICA* environment.

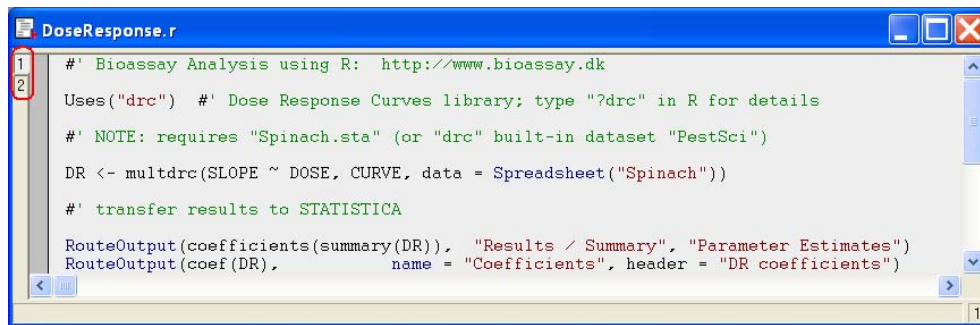
Although the *R.svb* macro is write-protected by default during the *STATISTICA* installation, it is accessible for inspection, and can be modified or enhanced to support new functionality required for specific use cases, although users should do so "at their own risk". This macro supports standalone execution to simplify debugging and testing of modifications.

## R Scripts as Native STATISTICA Macros

*STATISTICA* recognizes .R (and .S) file extensions as R scripts. Such files can be open through the *File* → *Open...* menu. It also registers (at the operating system level) .R/.S files as *STATISTICA* Macros, and therefore these files can be opened in *STATISTICA* from a file browser “by double-clicking”.



R scripts are displayed in slightly modified *STATISTICA Visual Basic* Macro windows. Such windows actually contain two scripts: the R script itself and the R Integration Support Macro (*R.svb*), accessible through two tabs in the upper-left corner (highlighted on the next screenshot).



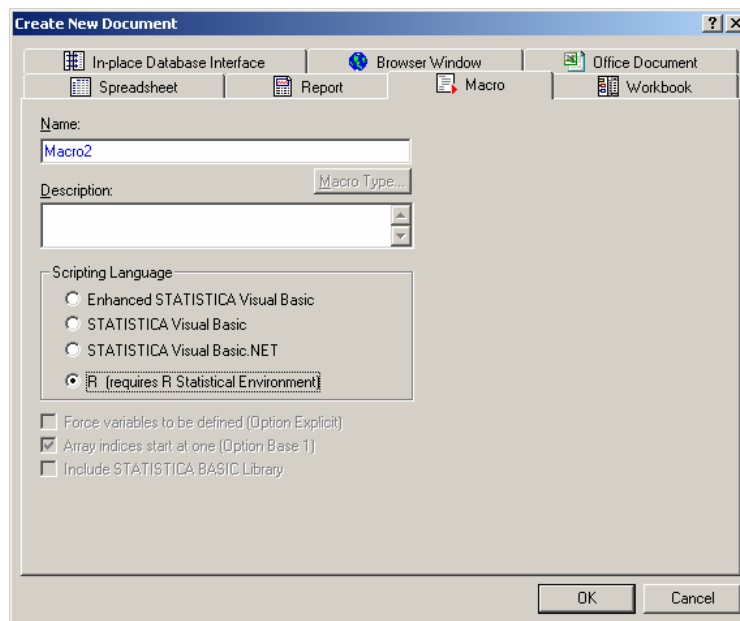
```

# ' Bioassay Analysis using R:  http://www.bioassay.dk
Uses("drc") #' Dose Response Curves library; type "?drc" in R for details
#' NOTE: requires "Spinach.sta" (or "drc" built-in dataset "PestSci")
DR <- multdrc(SLOPE ~ DOSE, CURVE, data = Spreadsheet("Spinach"))
#' transfer results to STATISTICA
RouteOutput(coefficients(summary(DR)), "Results / Summary", "Parameter Estimates")
RouteOutput(coef(DR), name = "Coefficients", header = "DR coefficients")


```

There is limited R code highlighting available (strings, language extensions, VB-style comments).

**Creating a new R script:** You can also create a new R script within *STATISTICA* - open the *New* document dialog (*File* → *New...*), switch to the *Macro* tab and select the *R Scripting Language* option (this option will only be available if R Integration support is installed on your machine):



This will open an empty R script window - you can now type or paste in an R program. R Integration support also includes an optional text file called *R.inc*, placed into the default installation directory along with *R.svb*: the contents of the file are copied to the beginning of each new R script created in this manner.

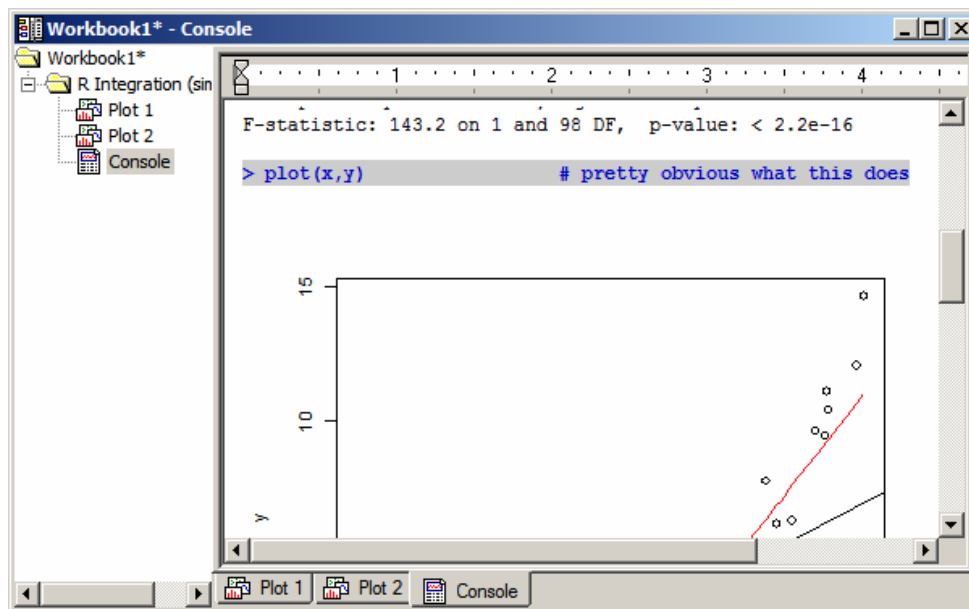
**Running R Scripts within *STATISTICA*:** click the  toolbar button or select the *Run* → *Run Macro* menu command (or press *F5*). This action will execute the *R.svb* macro for the currently active R script.

Although breakpoints are not supported for the R script itself, it's possible to set breakpoints and debug the *R.svb* macro on the second tab while running the R script.

Note that, in order to take advantage of R Integration features described in this document, R scripts should be executed from within (*Web*)*STATISTICA*. Although it is possible to develop and debug complex R programs within this environment, it was not particularly designed for these purposes; the R environment itself might be better suited for such activities.

## Retrieving Results

**Console Session:** The minimal output produced during execution of an R script is a *STATISTICA Report* that represents an R console session, including highlighted commands and any output generated by the R environment. Such a report will be produced even if the script is empty. The contents of this report can be edited and manipulated in the same way as one would edit any other *STATISTICA Report*.



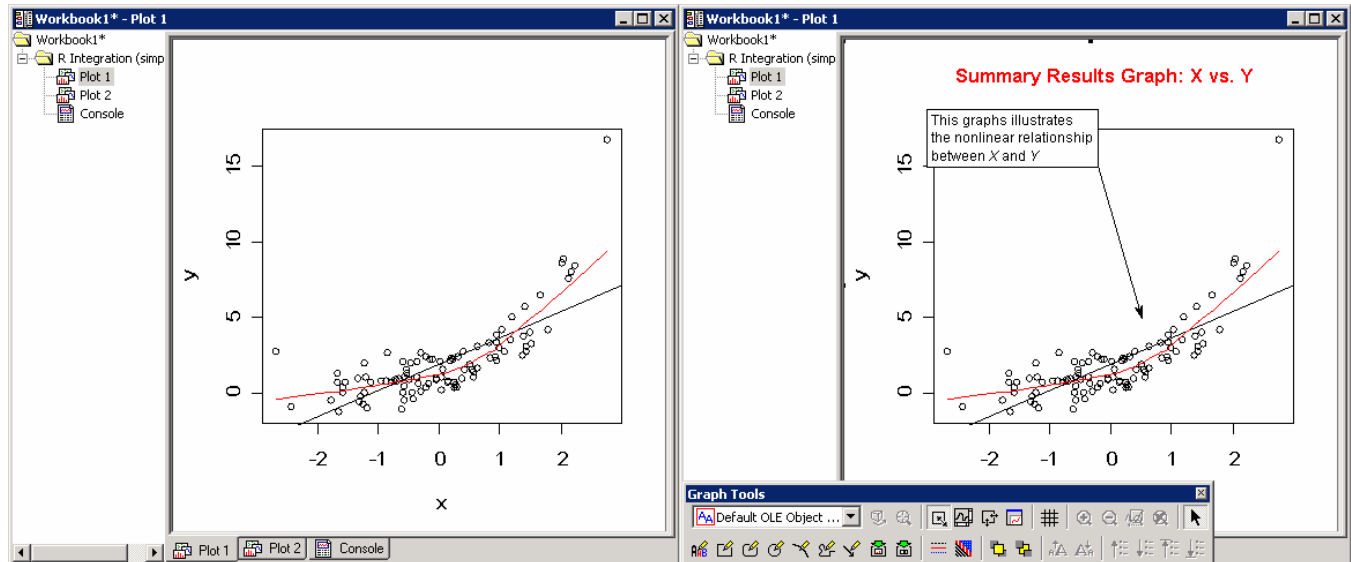
**Graphical Output:** All plots created during an R session are automatically transferred into *STATISTICA* environment as Windows Metafiles (vector graphics format), which means they can be resized without loss of quality.

These plots are placed into the R console session report, creating a natural flat report of the R session with embedded plots tied to the graphics commands (something that is not as easily achieved in R environment itself).

Moreover, the plots are also replicated as *STATISTICA Graphs* that become a part of the R script output. The metafile images are embedded into graph objects as locked "background" - this allows users to *annotate* R plots in *STATISTICA* using a familiar point-and-click interface with a set of text and drawing objects (like lines and arrows, rectangles and ellipses, polygons and pattern/color fill areas etc.). And since these annotations are anchored to relative positions on the plot area, they will remain correctly "attached" to the plot if the graph is resized.

Therefore, these graphs can now be flexibly "decorated" and further enhanced using all *STATISTICA* graphics tools, saved in other formats (e.g., JPG or GIF) or printed (e.g., to PDF files).

The individual R plot components (the structure of the plot) are not accessible for manipulation in *STATISTICA* graphs, and hence the rich capabilities of *STATISTICA* for creating and then further editing graphs (scaling, point markers, fit lines, etc.) are not available. However, integration between R and *STATISTICA* provides opportunities to extract data from R and then render important graphs inside the *STATISTICA* environment (by writing *STATISTICA Visual Basic* macros that will execute R scripts, extract results, and then post-process those results as necessary; this will be illustrated later).



## Unique *STATISTICA* Capabilities

Since R script output consists entirely of native *STATISTICA* objects:

- these objects are properly "routed" by *STATISTICA* according to Output Manager settings, which means that depending on user selection they could be placed into standalone windows or to a workbook
- each and all of them can be further managed using the extensive capabilities of the *STATISTICA* platform; e.g., they can be annotated, stored in a (compressed) workbook, exported as Microsoft Office documents, printed, saved as PDF file, converted to one of many popular formats, archived as a version-controlled, "auditable" items in *STATISTICA Document Management System*, shared among other users in a web-based client-server *WebSTATISTICA* environment etc.

## R Language Extensions: Passing Data to R and Retrieving Results

The R Integration Support Macro (*R.svb*) implements several extensions to R language – keywords and functions that can be used inside R scripts executed within *STATISTICA* environment. These extensions allow scripts to pass data to R and retrieve results from R environment.

**Important:** The R language is case sensitive, therefore R language extensions for *STATISTICA* are also case sensitive - they will only be recognized by the *STATISTICA* environment when typed exactly as shown.

Tabular data represented in *STATISTICA* in form of spreadsheets are mapped into the equivalent R structures – data frames. The mapping preserves as much information as possible for both formats: for example, text label variables in a spreadsheet become factor objects in a data frame, Missing Data values are mapped to *NA* indicators; data types, variable and case names are transferred both ways, etc.

### **ActiveDataSet**

The *ActiveDataSet* keyword has been adopted from the *STATISTICA Visual Basic* language and it performs the same function in R scripts: it references the active *STATISTICA* data spreadsheet.

In the desktop *STATISTICA* environment *active dataset* usually means the top-most visible spreadsheet which can act as a data source (it can also be a spreadsheet in a workbook selected as "Active Input"). This notion is redefined and extended for server-based environments (*WebSTATISTICA*, *Enterprise*), but the keyword is still valid and refers to the corresponding server-side mapping of the active data source.

If no active dataset is defined/available, the R script that uses it will fail (same is true for SVB macros).

If R Integration Support Macro encounters the *ActiveDataSet* keyword in the R script, it transfers the actual *STATISTICA* active dataset into R environment and assigns it to a variable of the same name. Therefore this keyword represents a data frame variable and can be handled as such in the script.

*Example:*

```
ActiveDataSet[1:5] # display a subset of active dataset (variables 1 through 5)
str(ActiveDataSet) # view the structure of the data frame
plot(ActiveDataSet$MEASURE01) # plot "MEASURE01" variable values
```

### **Spreadsheet (FilePathOrName)**

*FilePathOrName* – a literal string, e.g. "c:\datasets\sample.sta"

The *Spreadsheet (FilePathOrName)* extension allows you to load a specific *STATISTICA* data file and transfer the data in that file to an R data frame.

The behavior of this extension is similar to the *ActiveDataSet* keyword, but in this case R Integration Support code first loads the requested spreadsheet into *STATISTICA*, then transfers it to the R environment, replacing the *Spreadsheet ()* function call in the script with a reference to a data frame variable containing the spreadsheet.

Note that in the current implementation the *Spreadsheet ()* function is not preserved by *STATISTICA*: it is replaced by a respective ".Spreadsheet.N" variable, so this pseudo-function only accepts literal

strings as parameters – you can't pass a variable containing a string. This might change in future releases.

Similar to the *ActiveDataSet* keyword, the return value of the *Spreadsheet()* function should be treated as a data frame variable with the contents closely matching that of the corresponding *STATISTICA* spreadsheet.

Note that instead of using *Spreadsheet()* you could also use ***Spreadsheets.Open()***, which is mapped to the same implementation but might be more familiar to *STATISTICA Visual Basic* developers.

One useful feature supported by this function is the use of *default search paths* for spreadsheet files that are specified only as simple file name. This means that if the function parameter consists only of a file name, e.g. *Spreadsheet("some.sta")*, R Integration Support code will look for this file in several locations: first, it will check the folder where the R script itself is located (if it was saved to disk), then it will check the *Examples\Datasets* folder for the current *STATISTICA* installation. Support code will also append the default *.sta* file extension, if one is not present. Therefore the following options are available:

- R scripts can reference the accompanying datasets (placed in the same folder) simply by name
- Spreadsheets that are included in every *STATISTICA* installation as demonstration/example datasets can be referenced by name in much the same way as *built-in* R datasets

*Example:*

```
Spreadsheet("c:\myfiles\mydata.sta")      # display the dataset in R console
Spreadsheet("\\server\share\data.sta")    # same, but read from a network share
Spreadsheet("thisdemo.sta")              # file in the same folder as this script
...
# plot a histogram for MEASURE05 variable from STATISTICA example dataset Adstudy:
hist(Spreadsheet("Adstudy")$MEASURE05)
...
advert = Spreadsheet("Adstudy")$ADVERT    # retrieve a data frame variable (factor)
is.factor(advert)                        # => [1] TRUE
levels(advert)                            # => [1] "PEPSI" "COKE"
```

### ***RouteOutput (RObject, [SpreadsheetName], [SpreadsheetHeader])***

*RObject* – a data frame, matrix, array, number, or a string

*SpreadsheetName, SpreadsheetHeader* – literal string, e.g. "A Frequency Table"

This extension transfers various types of data from R environment into *STATISTICA* spreadsheets.

Although the function was introduced to retrieve tabular data (such as data frames, matrices or arrays) into spreadsheets, single-value data such as numbers or strings can be passed as well and will be placed in single-cell spreadsheets. *RObject* can be an R variable or a literal value.

The name of this extension, `RouteOutput()`, hints at the similarity of its behavior to the equivalent *STATISTICA Visual Basic* function: the "results" spreadsheets recreated by the function in *STATISTICA* environment become the standard output of the R script / analysis and follow Output Manager settings [select *File* → *Output Manager...* menu in *STATISTICA*], i.e. they are "routed" either to individual windows or to a workbook (or multiple workbooks for each analysis, with optional output reports, e.g. as a Microsoft Word document); most popular setting is a single *results workbook*.

Optional parameters `SpreadsheetName` and `SpreadsheetHeader` specify the name and header of the resulting spreadsheet. It is recommended to provide a value for the spreadsheet name for visual distinction in the tree-view of the results workbook.

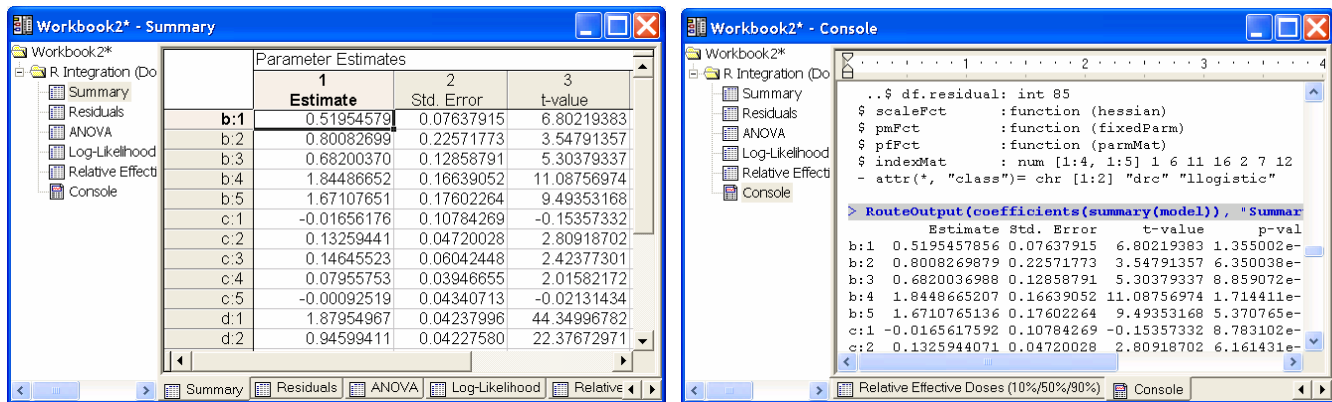
Note that R plots transferred into *STATISTICA* as native graphs do not require explicit output routing – all plots generated during a script run are automatically transferred and routed according to Output Manager settings.

**Important:** *Many functions in R, specifically the ones that perform statistical modeling, represent their results as structured objects, sometimes of significant complexity.* These objects cannot be reduced to a single table, and therefore cannot be handled by the `RouteOutput()` extension (they could be automatically traversed in search of tabular components, but since the object structures are specific to a particular method, such approach would generate a significant amount of “junk” output). However, since the results (the actual data of interest) are either stored in such objects as tabular components or produced by applying an object's method to some input data, this limitation does not pose any problems – particular results can be easily extracted from such a statistical model object and routed back to *STATISTICA*.

*Example:*

```
# build a statistical model (Dose Response Curves)
model <- multdrc(SLOPE ~ DOSE, CURVE, data = PestSci)
...
str(model) # display the complex structure of this model/results in R console
...
# transfer results to STATISTICA
RouteOutput(coefficients(summary(model)), "Summary", "Parameter Estimates")
RouteOutput(residuals(model), name = "Residuals", header = "Some Header")
RouteOutput(anova(model), "ANOVA") # inherits the default heading from data frame
RouteOutput(logLik(model), "Log-Likelihood")
RouteOutput(ED(model, c(10, 50, 90)), "Relative Effective Doses (10%/50%/90%)")
```

The results for this example may look like this:



### Uses (RPackageName, [AttachImports])

*RPackageName* – literal string, e.g. "drc"

*AttachImports* – boolean value, by default = FALSE

The *Uses (RPackageName)* function can be used to automatically ensure that the respective package (library) named *RPackageName* AND its *dependencies* (packages that are required in order for this package to run) are installed (on the computer/server where the R script is running) and loaded into the R environment. If some of these libraries are not present, they will be silently installed from CRAN repository (<http://cran.r-project.org>) and then loaded.

R defines several levels for specification of package dependencies. One of them is *Imports* – it lists packages *with namespaces* that have to be loaded for the package to run, but the namespaces do not need to be *attached* to the current/global environment, meaning that the package accesses such imported packages using namespace-qualified names, e.g. `tseriesChaos::mutual(x)` instead of `mutual(x)`. If you need to use some of these namespace-enclosed functions in your own R code, you can either explicitly use namespace qualification [`tseriesChaos::mutual(x)`], attach the library to current environment [`Uses("tseriesChaos")` or `library(tseriesChaos)`] or set the second optional *Uses()* parameter *AttachImports* to TRUE (this will attach all the imports of the package).

Note that this extension is not necessary for interaction between *STATISTICA* and R, and it's possible to implement it within R language itself, rather *it simplifies the process of conditional library installation and loading by encapsulating it in a single call.*

#### Example:

```
Uses("drc") # make sure that the respective package is installed and loaded
...
DR <- multdrc(SLOPE ~ DOSE, CURVE, data = PestSci) # call the package methods
```

This program fits dose response curves to the respective variables of the built-in *PestSci* dataset by calling *multdrc* function defined in "drc" package. *Uses("drc")* ensures that the function is available by installing and loading the package, if necessary.

## Executing R Scripts from within *STATISTICA Visual Basic*

A typical use case for leveraging specialized R functionality within *STATISTICA* is to call R scripts from inside a *STATISTICA Visual Basic (SVB)* macro. This way we can build new "modules" using *SVB* User Interface library and methods implemented in R scripts. Likewise, such functionality is required in order to create *STATISTICA Enterprise SVB* analysis configurations, or *STATISTICA Data Miner* nodes (for *STATISTICA* or *WebSTATISTICA*) that leverage R.

But in order to provide any non-trivial functionality within an R script in such use cases, we need to be able to "parameterize" that script with user-selected parameters, variables lists, input spreadsheets, etc. *STATISTICA* provides a simple and powerful way to pass such parameters to R scripts.

### R Scripts are *STATISTICA* Macros

In (almost) all cases *STATISTICA* treats R scripts in the same way as native *SVB* macros. This applies to the *STATISTICA* Object Model as well: *Macro* objects in *SVB* programs can now represent R scripts. Therefore, R scripts can be created, opened, edited, saved and executed from within *SVB* scripts.

This, in turn, means that R functionality is available in *STATISTICA Enterprise* analysis configurations and *STATISTICA Data Miner* nodes, since they are *SVB*-based.

Existing R script files can be open with `Macros.Open("path\to\some.r")` or created on-the-fly with `Macros.New()` and `Macro.Code`. Note that in the latter case *STATISTICA* needs help in distinguishing R scripts from *SVB* macros – this can be achieved either by specifying the name for a new macro with `.R` extension (even if you are not going to save it on disk), or by explicitly setting `Macro.Scripting` to 5 (R Macro Type). Run the scripts by calling `Macro.Execute`.

**Important.** The `Macro.Scripting` type for R scripts is 5 (later will be mapped to a symbolic constant).

*Example:*

```
Sub Main
  Dim R As New Macro
  R.Code = "ActiveDataSet" ' simple R script created on-the-fly
  R.Scripting = 5          ' R Macro Type = 5
  R.Execute
End Sub
```

This *STATISTICA Visual Basic* macro runs a simple R script containing only a single command `ActiveDataSet` which, as described in the previous section, is an R language extension for *STATISTICA* that will transfer (and in this case display) the currently active *STATISTICA* data file in R. For example, if you run this macro after opening example data file `Exp.sta`, a listing of that file will be displayed in a report window that represents the R console session:

	GROUP	GENDER	TIME	PAID	STRESS_R	CORRECT1	CORRECT2	CORRECT3
1	EXPERMTL	MALE	BEFORE	NOT_PAID	1.41	12	4	6
2	EXPERMTL	MALE	BEFORE	NOT_PAID	1.73	3	3	7
3	EXPERMTL	MALE	BEFORE	PAID	0.00	7	6	0
4	EXPERMTL	MALE	BEFORE	PAID	1.41	11	7	3
5	EXPERMTL	MALE	AFTER_1	NOT_PAID	12.83	8	2	7
6	EXPERMTL	MALE	AFTER_1	NOT_PAID	2.24	15	1	7



## Passing Parameters to R Scripts: *Collection* Object

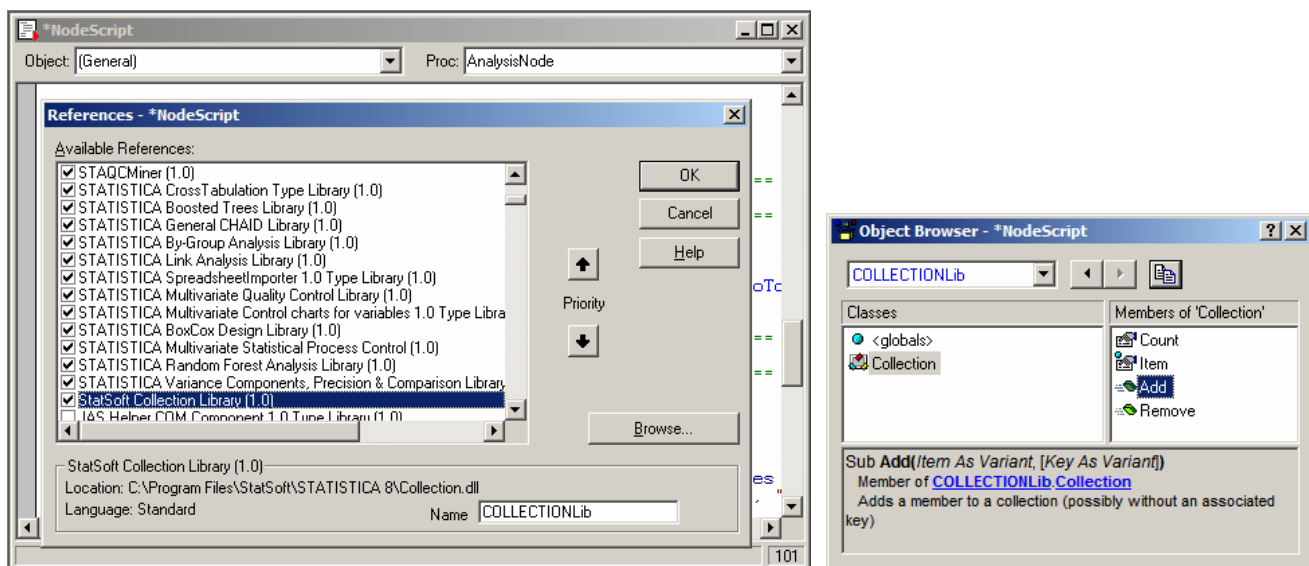
*Collection* is a COM library introduced with *STATISTICA* Version 8 MR3 that implements generic *Collection* objects. Such objects allow users to store keyword-tagged Variant values. Variants are COM structures that can store one of many supported data types, including arrays and references to other COM objects, e.g. Spreadsheets.

What this means for R Integration is that we can now store  $\{parameter\ name = parameter\ value\}$  pairs in such a *Collection*, then pass this parameter set to *STATISTICA* prior to R script execution, thus achieving script parameterization. The R Integration Support macro will transfer all the parameters from the *Collection* to the R environment as R variables named using the respective parameter keyword tags, allowing the script that is executed in this environment to reference all these variables (script parameters) directly by name.

R Integration supports the following types of named parameters: strings, numbers, arrays, Spreadsheets. Support for *STATISTICA* spreadsheets is particularly useful: for example, such a parameter can override the *ActiveDataSet* keyword, allowing a single script to serve as an R "engine" for an interactive module, *Enterprise* analysis configuration and *Data Miner* node implementations.

Also note that string parameters without a keyword tag are treated as R code that should be executed prior to execution of the script itself. This is analogous to *SVB* "hidden code" and can be used to, for example, define a common set of new functions or global variables/constants.

In order to use *Collection* objects, *SVB* scripts must include a reference to the *StatSoft Collection Library* (add it using *Tools* → *References* menu while editing the macro):



The *Collection* object has several generic properties and methods that are needed to manipulate the contents of a collection: *Count*, *Add(Item, [Key])*, *Remove(KeyOrIndex)*, and *Item(KeyOrIndex)* that returns an *Item* object with *Key* and *Value* properties. However, due to the use of so-called *default* object properties (*Item* is the default property of a *Collection* and it returns its *Value* property by default) interaction with a *Collection* object is reduced to intuitively clear assignment operations:

```
Dim param As New Collection
```

```
param("number") = 57
param("string") = "A string sample..."
```

Once you have assembled the parameter collection, execute the parameterized R script by calling *Macro.ExecuteWithArgument(Parameters As Collection)*.

**Example:**

```
Dim s1 As New Spreadsheet, s2 As New Spreadsheet ' ... populate s1, s2 with data
var1 = Array("CASE 1", "CASE 2")
var2 = Array(1, 2, 3, 4, 5)
' * don't use spaces in parameter names
' * some names are "locked" and can't be used [e.g. 'text', 'str', 'sample']

Dim param As New Collection
param("number") = 57
param("string") = "A STRING sample..."
param("string_array") = var1 ' add items with an assignment operator
param.Add(var2, "number_array") ' OR using explicit Add() method
param("SomeSpreadsheet") = s1
param("ActiveDataSet") = s2 ' override the value of 'ActiveDataSet' keyword
' string parameters without associated keys will be treated as R code and
' will be executed before the script - an analog of SVB 'hidden code'
' * define a function that will be available to the R script
param.Add("func <- function(x) { cat('Called func(x) with x =', x) }")
' * another way to define a global constant or variable
param.Add("STATISTICA.Version = '" & Version & "'")

' now run the R script with this collection of parameters
' (parameters become R variables - the script can reference them by name)
Dim m As Macro
m = Macros.Open(MacroDir & "\parameterized.r")
m.ExecuteWithArgument(param)
```

```
Workbook6* - parameterized.r
Workbook6*
  R Script Results
    parameterized.r

This script has input parameters:

number = 57
string = A string sample...
string_array = CASE 1 CASE 2
number_array = 1 2 3 4 5
SomeSpreadsheet =
  First Second
Case1      5    NA
Case2     NA    25

=====
> cat("Interesting fact: ", number, "- 1 =", number - 1)
Interesting fact: 57 - 1 = 56

> str(string_array) # display variable structure
chr [1:2] "CASE 1" "CASE 2"

> cat("STATISTICA Version:", STATISTICA.Version)
STATISTICA Version: 8.0

> if (exists("func")) func(string)
Called func(x) with x = A string sample...
```

## Managing Script Results

Both `Macro.Execute()` and `Macro.ExecuteWithArgument()` methods handle R script output (spreadsheets, graphs, report) exactly the same way as the output from *SVB* macros – it is routed by *STATISTICA* according to Output Manager settings, e.g. placed into a results workbook.

This approach might be sufficient for many use cases, but is not flexible enough in situations where the output from an R script is treated as *intermediate result* for subsequent processing in *STATISTICA*.

For example, an *SVB* script implementing a custom interactive module might implement a multiple stage analysis workflow, calling into R, analyzing the results, requesting additional inputs from the user and calling into R repeatedly with previous results as inputs. Or an *SVB* macro could use tabular results returned from R to produce graphs that are more expressive and flexible than what R is able to generate.

Since the items processed by *STATISTICA* Output Manager are intended to be a final representation of analysis results, it is not trivial to access them individually for further processing – one would need to know the type of representation selected by the user (single or multiple workbooks, individual windows) and programmatically iterate through the respective set of output items (windows, workbook contents represented as a tree), searching the item of interest by name.

### ***Macro.ExecuteNoRouteOutput([Parameters As Collection]) As StaDocCollection***

This method has been added to accommodate such special use cases: R script output is not "routed" at all, so the script runs *silently*; all the output from the script is collected into a *StaDocCollection* object (standard *STATISTICA* container for documents like spreadsheets, graphs, reports or workbooks) that is returned as the result of script execution. It also accepts an optional Collection for parameterization.

Now *SVB* macro developers can easily access individual components of R script output, e.g. to extract individual cell data from spreadsheets or to create a complex graph based on multiple columns from several spreadsheets.

#### **Example:**

```
Dim m As Macro
m = Macros.Open(MacroDir & "\some.r")

Dim Routput As StaDocCollection
Routput = m.ExecuteNoRouteOutput() ' silent, no output displayed

' do something with "Routput": extract and operate on individual documents

' trivial case: results are routed (displayed), as if from Macro.Execute()
RouteOutput(Routput, "R Script Results").Visible = True
```

## More Examples

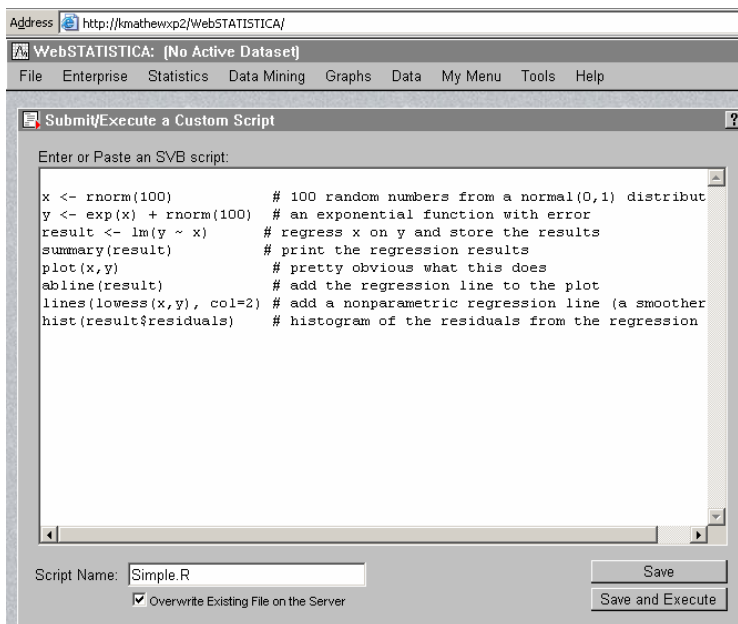
At this point we have demonstrated all functional components required to build custom applications within *STATISTICA* platform that can take advantage of the specialized functionality available in R.

All installations of *STATISTICA* now have a set of examples that provide a more detailed demonstration of the described features; you will find these examples in the `[STATISTICA]\Examples\R` folder. These examples may also be used as templates for your own development.

## R Support in *WebSTATISTICA*

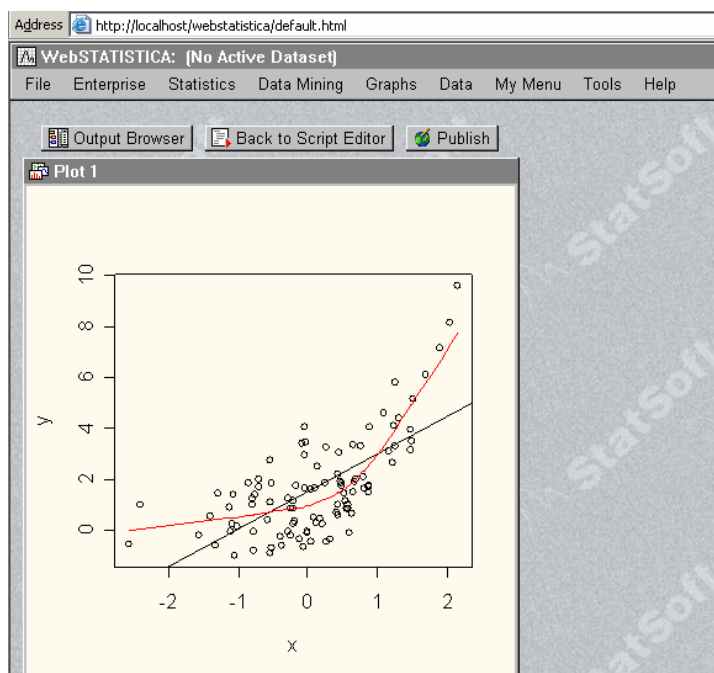
*STATISTICA* and *WebSTATISTICA* servers are based on identical *STATISTICA* libraries, and support identical functionality. This is true for R support in *STATISTICA* as well: you can execute R in *WebSTATISTICA* in much the same way as from desktop (thick-client) *STATISTICA*.

For example, sign into *WebSTATISTICA*, and select *Tools* → *Submit/Execute a Custom Script*; then type in the macro shown below (available as *standalone.r* in the *Examples\R* folder).



**Important:** Make sure to save the script with *.R* extension.

To run the script, click on *Save and Execute* or use the *File* menu to open and run it.



Note that this script does not require a data file. If you are using *ActiveDataSet* or *Spreadsheet()* extensions, keep in mind that the data sources have to be located on the server. *Spreadsheet()* should reference datasets by their *WebSTATISTICA* URI paths; *ActiveDataSet* will be mapped to the spreadsheet that you open in *WebSTATISTICA*, but this can be overridden by a parameter of the same name passed to the R script from *SVB*. Also, in order to execute any scripts in *WebSTATISTICA* the user must have the proper permissions on the server to do so.

## **WebSTATISTICA is a Powerful R Server**

*WebSTATISTICA* was designed as a powerful and flexible server/web-based analytical platform, relying on *STATISTICA Visual Basic* engine for diversity of its functionality, as well as extensibility.

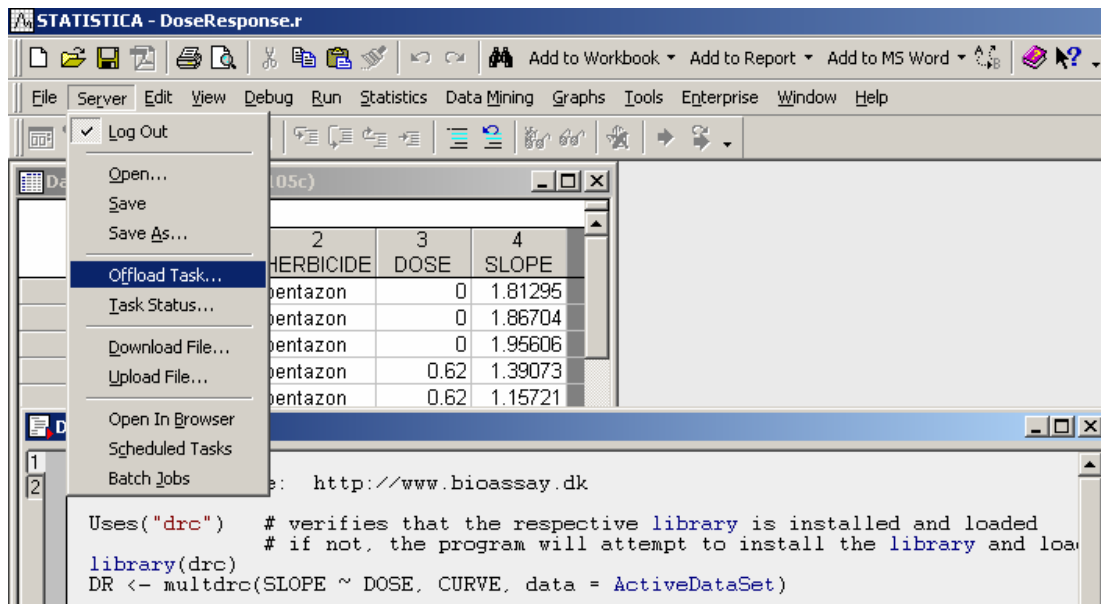
R scripts are handled by *WebSTATISTICA* server in much the same way as standard *SVB* macros. The third-party components that the *STATISTICA* platform relies upon to provide its R runtime environment (such as R library and R COM Server library) are also well suited for handling multiple simultaneous R sessions.

Thus, *WebSTATISTICA* represents an ideal platform for a powerful and flexible multi-processor R server that can handle a large number of users, providing security, scheduling, load balancing etc.

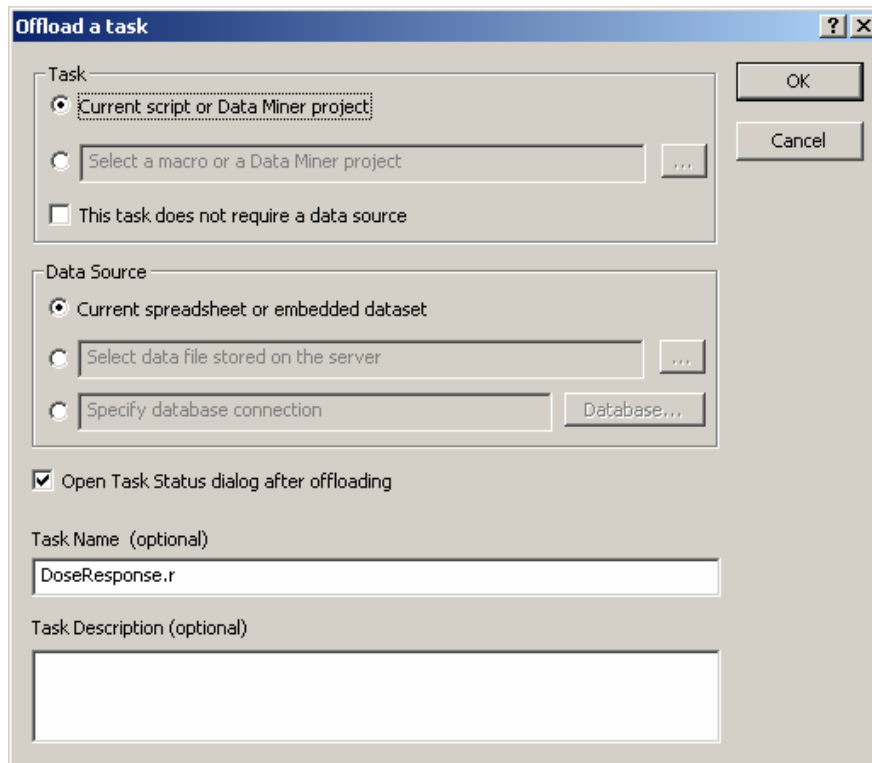
## **Off-Loading R Scripts from STATISTICA to WebSTATISTICA**

Like *SVB* scripts or *Data Miner* workspaces, R scripts can be offloaded from a desktop environment for processing on a *WebSTATISTICA* server. This is particularly useful when an R script is expected to require significant computing resources - users can submit such scripts to the server (e.g. overnight) and retrieve the results at a later time.

If your installation is set up to work with a *WebSTATISTICA* server (*Tools* → *Options: Server/Web*), you can offload any R "task" by selecting *Server* → *Offload Task...*:

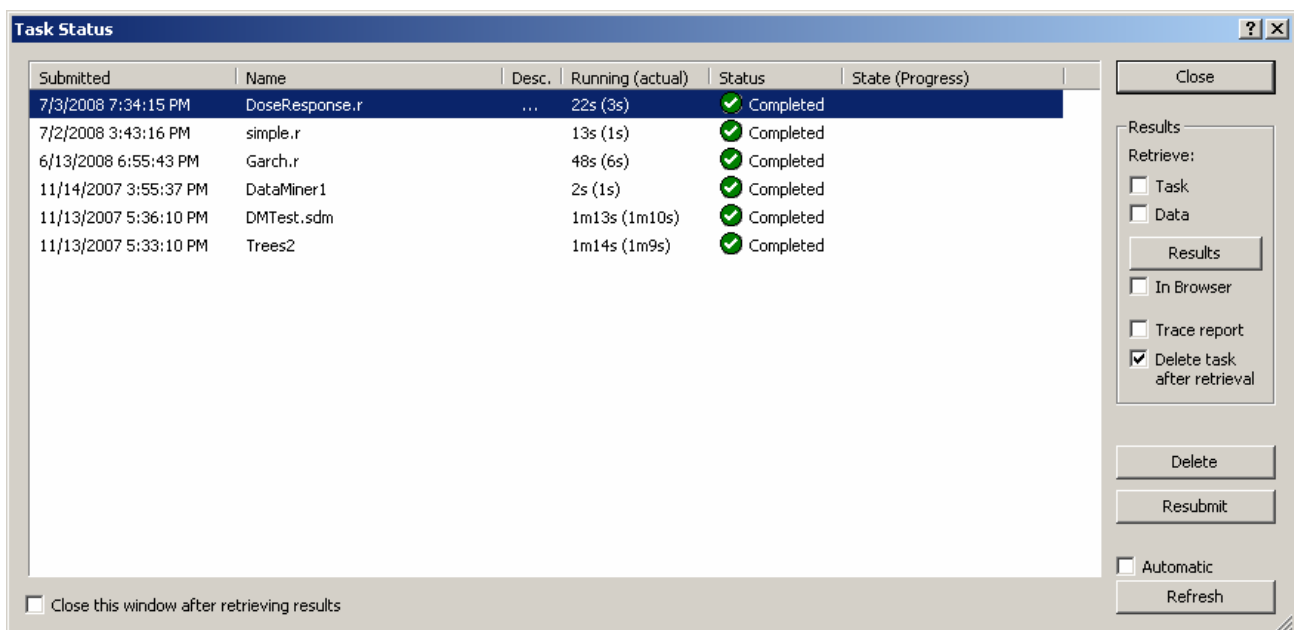


Options are provided to also transfer the data file to the server side; these options are described in detail in the *STATISTICA* documentation.



After clicking *OK*, the respective R script will execute on the *WebSTATISTICA* server (or scheduled to be executed, depending on the server load).

The progress of the analysis can be monitored via the *Server* → *Task Status...* dialog.




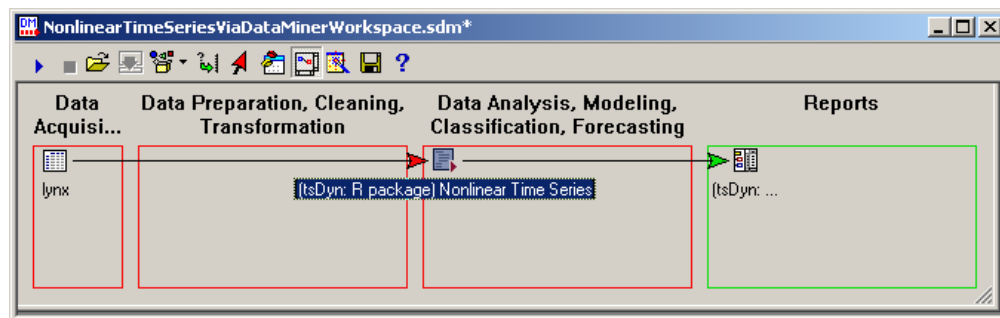
The results can be retrieved from the server via the *Results* button (or double-click on the task); the representation of the results of an offloaded task is equivalent to the same task running locally.

## Creating R-based *STATISTICA* Data Miner Nodes

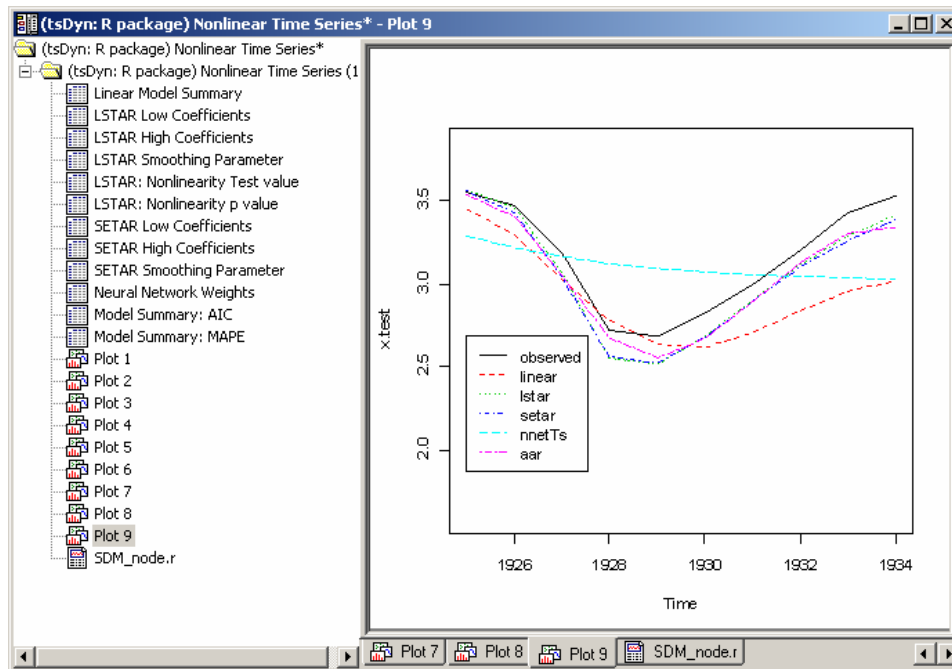
We can use the techniques described in previous chapters to implement a custom R-based *Data Miner* node: Edit the *SVB* node's source to generate a *Collection* based on the node's input parameters, use it to execute one or more parameterized R scripts, and retrieve the results as a *StaDataCollection* that becomes the output of the node.

For an example, see *[STATISTICA]\Examples\R\NonLinear Time Series\As Data Miner Node.sdm*:

Select the analysis node and click on the *Edit Code*  icon to review the underlying *SVB* code (*NonLinearTimeSeries.svb*); note how the corresponding *.dmi* file describes the node's input parameters; the R script file (*NonLinearTimeSeries.r*) performs nonlinear time series analysis using the *tsDyn* R package; this script can serve as a template for your own R-based *Data Miner* nodes.



The results node in *Reports* group is a workbook.



This example demonstrates that the ability to easily incorporate specialized R functionality into *STATISTICA* *Data Miner* provides very powerful opportunities for creating analytic workflows, or collections of data miner nodes and node-browser configurations dedicated to highly specialized analyses (such as nonlinear time series, dose response curve fitting, and so on).

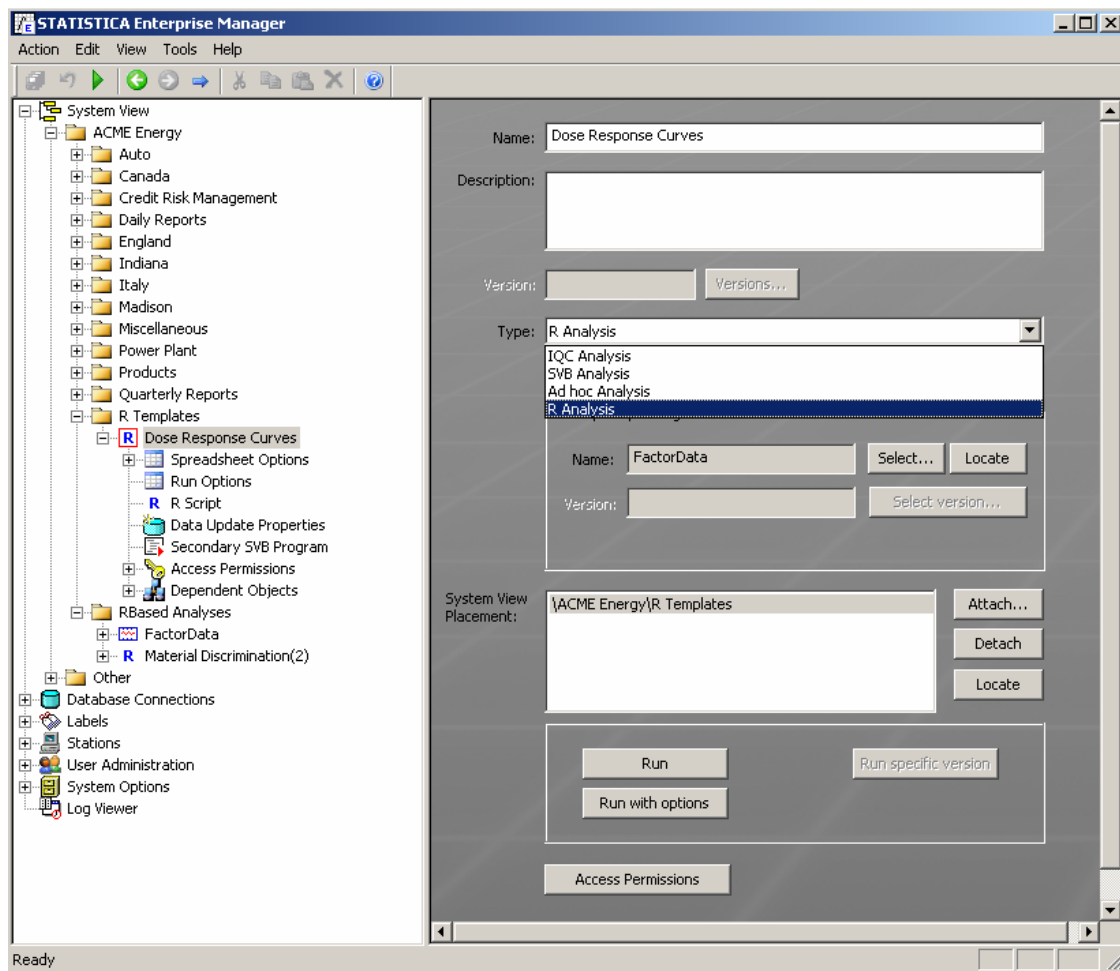
## Integrating R into (Web)STATISTICA Enterprise

R Integration in *STATISTICA* would be incomplete without support for (Web)STATISTICA Enterprise solutions: Now R programs may be used in or as analysis templates. This means R analyses can be distributed to end users who are not familiar with Visual Basic or R programming. Furthermore, because the (Web)STATISTICA Enterprise platform provides numerous options and features to specifically enable the application of templated analyses for validated applications (e.g. for manufacturers who need to comply with the requirements of FDA 21 CFR Part 11), such as version control and audit trails, these features make R available for deployment in mission-critical manufacturing applications.

### Creating R-based Analysis Configurations

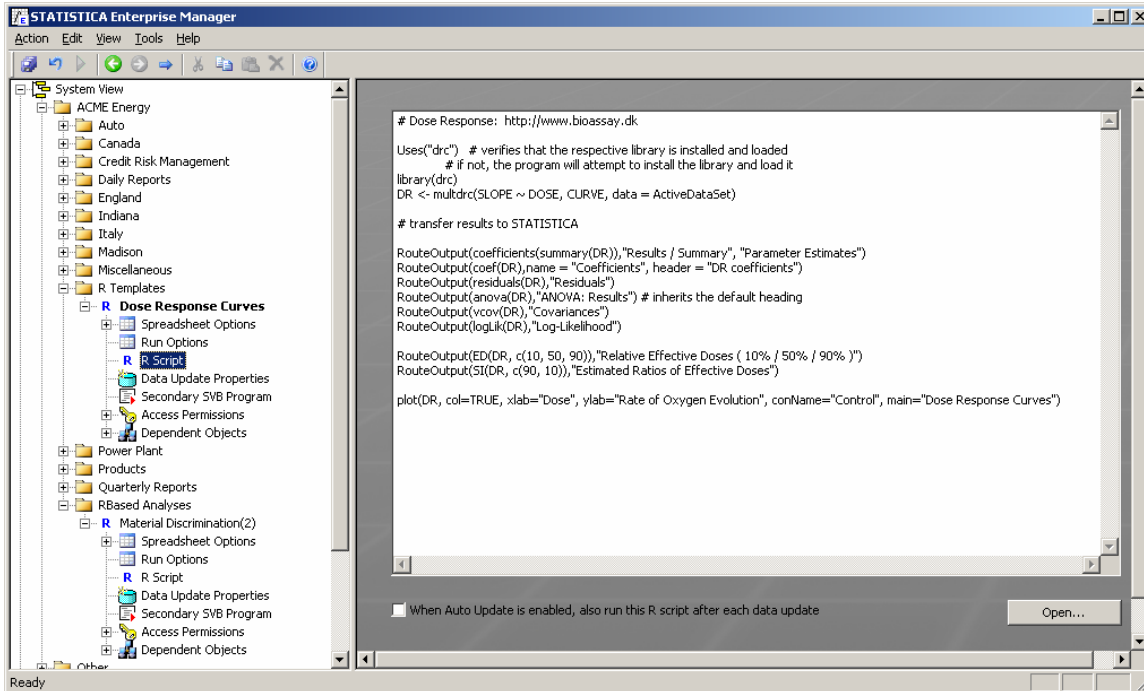
The *STATISTICA Enterprise* system implements standard control charting and SPC analyses out-of-the-box; non-trivial analysis templates are usually created using *STATISTICA Visual Basic (SVB)* (e.g., by using *STATISTICA* macro recording capabilities). In addition, (Web)STATISTICA Enterprise now recognizes R scripts, and can store and process them in the same way as SVB scripts.

When creating a new Analysis Configuration in *STATISTICA Enterprise*, select the *R Analysis* option:

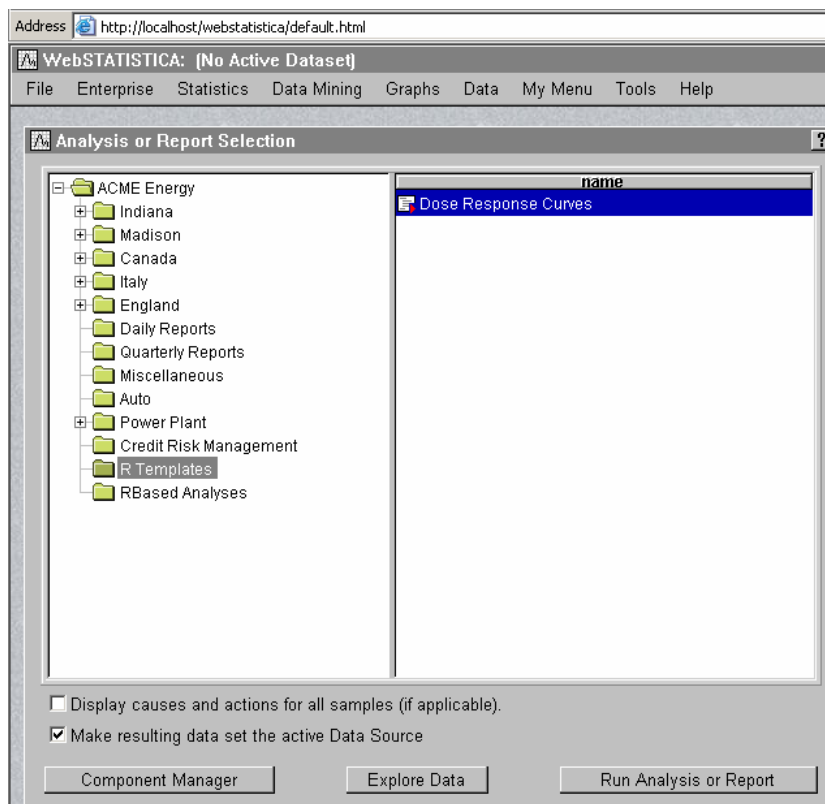


Then type, paste in, or load the respective R script; all other options are identical to those available for SVB-based analysis templates:





Depending on the configuration of the *STATISTICA Enterprise* system, this reusable analysis template is now available to the respective users of both *STATISTICA* and *WebSTATISTICA* environments, and the results of this template can be combined into standard reports.



Moreover, if the respective (*Web*)*STATISTICA Enterprise* installation is integrated with *STATISTICA Document Management System*, then these R scripts can be "locked down", i.e., versioned with complete audit trails (to support FDA 21 CFR Part 11 requirements).

## **Calling R Scripts from SVB Analysis Configurations**

Similarly, R functionality can also be utilized from inside *SVB* analysis templates, which can retrieve the results from R for further processing or display. In this case R scripts that are loaded for the respective templated analyses in *STATISTICA Enterprise* can either be placed into a secure folder repository from which they can be loaded (and where they can be managed via *STATISTICA Document Management* integrated with *STATISTICA Enterprise*), or they can be embedded into the *SVB* code (assign R script text to the *SVB* macro's *Code* property).

## **Summary**

Full integration of R environment into *STATISTICA Enterprise* allows organizations to utilize extensive and highly specialized functionality of R and leverage it inside a secure, enterprise-wide, role-based analysis system that insulates end users from implementation details and which can be deployed into validated manufacturing environments. Taken together with the ability to run R-based templates on the *WebSTATISTICA* server platform, and to extract results into pre-formatted standard reports, these capabilities make *STATISTICA Enterprise* arguably the most powerful enterprise analysis system available to date.

## Final Comments, and Some Caveats

The features provided in *STATISTICA* for integration with R are quite flexible, and make the thousands of highly specialized R functions and features available to all *STATISTICA* solutions. However, the users that are planning to exploit these features are advised to consider the following possible issues, in particular, certain system limitations of the R platform.

### **Error Handling**

Most of the error conditions generated within the R environment (e.g. syntax and runtime errors caused by an R script) or by the integration support libraries (e.g. broken R installation or missing components) are intercepted and handled by *STATISTICA*. Developers can use error handling facilities available in either environment, for example *On Error* handlers in *SVB* macros calling R scripts.

However, occasionally R programs can crash or hang (in the latter case program control does not return to *STATISTICA*). Therefore, careful validation of the respective R scripts is crucial for enterprise-level deployment of R analysis templates.

### **Strengths and Limitations**

**A word of caution regarding the quality of R algorithms:** R comes without warranty or guarantees. In practice, many (most) of the algorithms available in R are the result of diligent work over many years by one or a few individuals who are experts in the respective methodology or domain. However, this does not mean that the software was created following rigorous software development lifecycle methodology, or stringent standard operating procedures for software requirements gathering, design, implementation, and testing. Therefore, in order to build a mission-critical or validated application around a component that depends on R, it is absolutely critical that you carefully validate all results for the use cases to which the software is to be applied.

**A word of caution regarding scalability, large datasets, etc:** Another caveat regarding R that needs to be considered before building solutions around R concerns its basic architecture. Unlike *STATISTICA*, data in R must be (in practically all cases) resident in the computer's memory. This restriction, in combination with hardware-level and operating system-level memory limitations, may or may not pose an obstacle for any one individual user, but for sure will need to be considered carefully when building R-based server applications accessible to multiple users.