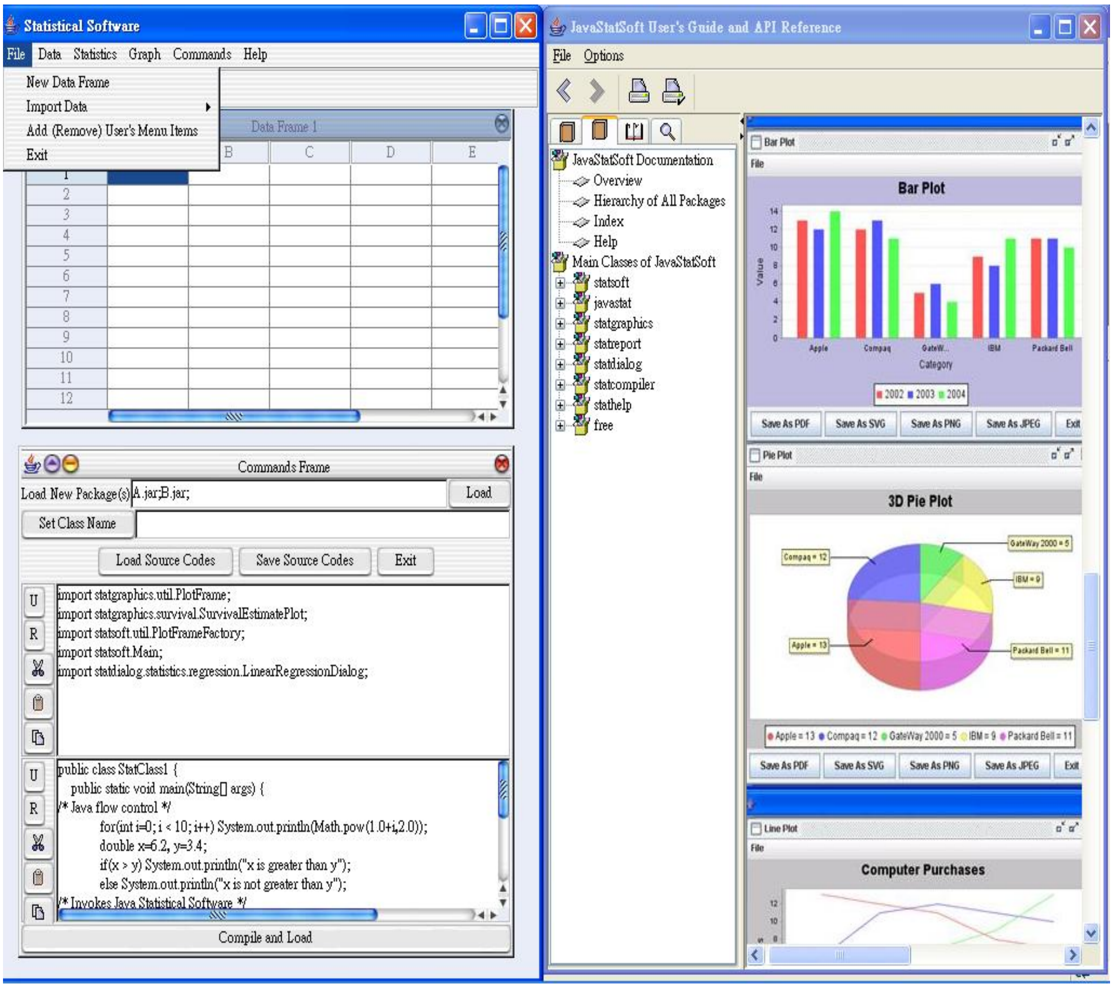
Java Statistical Software (JavaStatSoft)

A. Look and Feel



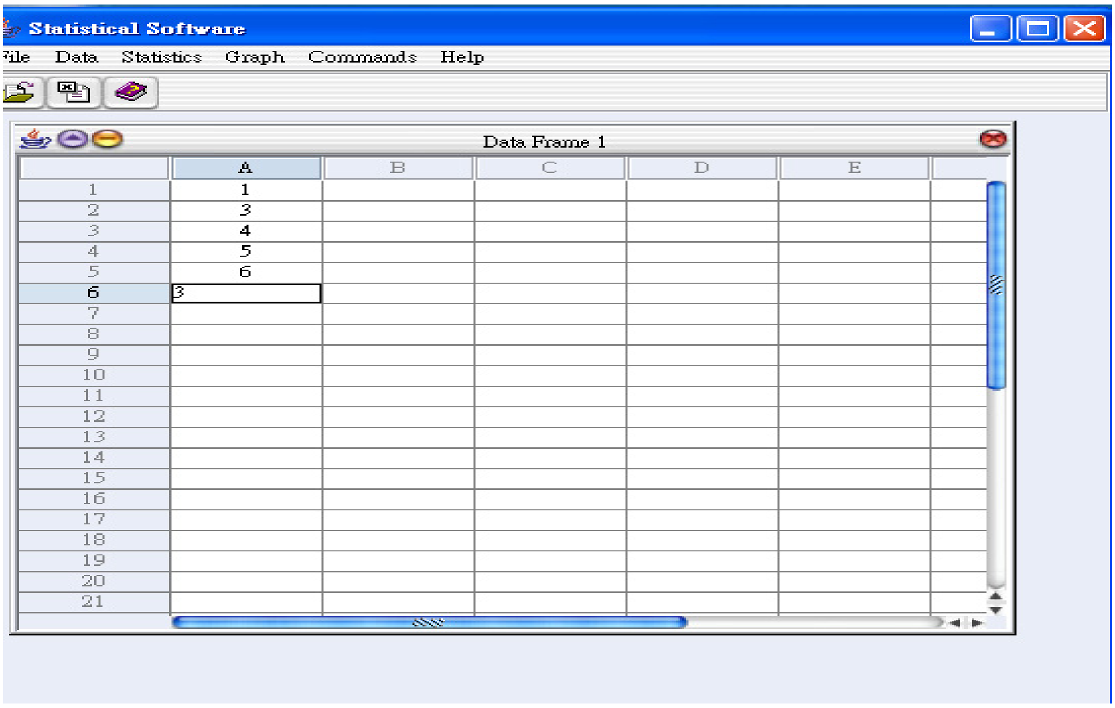
Load New Package(s): A.jar;B.jar;

```
import statgraphics.util.PlotFrame;
import statgraphics.survival.SurvivalEstimatePlot;
import statsoft.util.PlotFrameFactory;
import statsoft.Main;
import statdialog.statistics.regression.LinearRegressionDialog;
```

```
public class StatClass1 {
   public static void main(String[] args) {
/* Java flow control */
          for(int i=0; i < 10; i++) System.out.println(Math.pow(1.0+i,2.0));
          double x=6.2, y=3.4;
          if(x > y) System.out.println("x is greater than y");
          else System.out.println("x is not greater than y");
/* Invokes Java Statistical Software */
```

B. Data Input: To Spreadsheet, From File or From Database

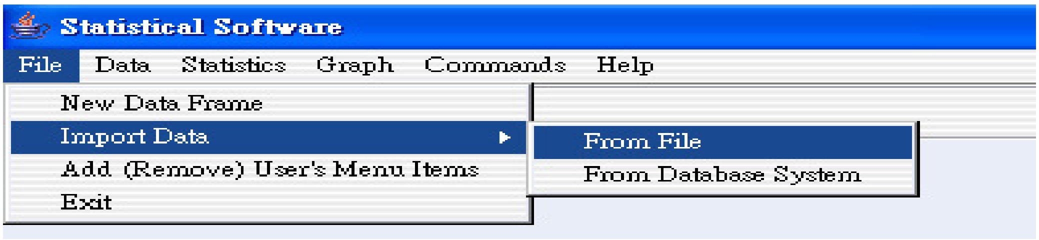
1. To spreadsheet



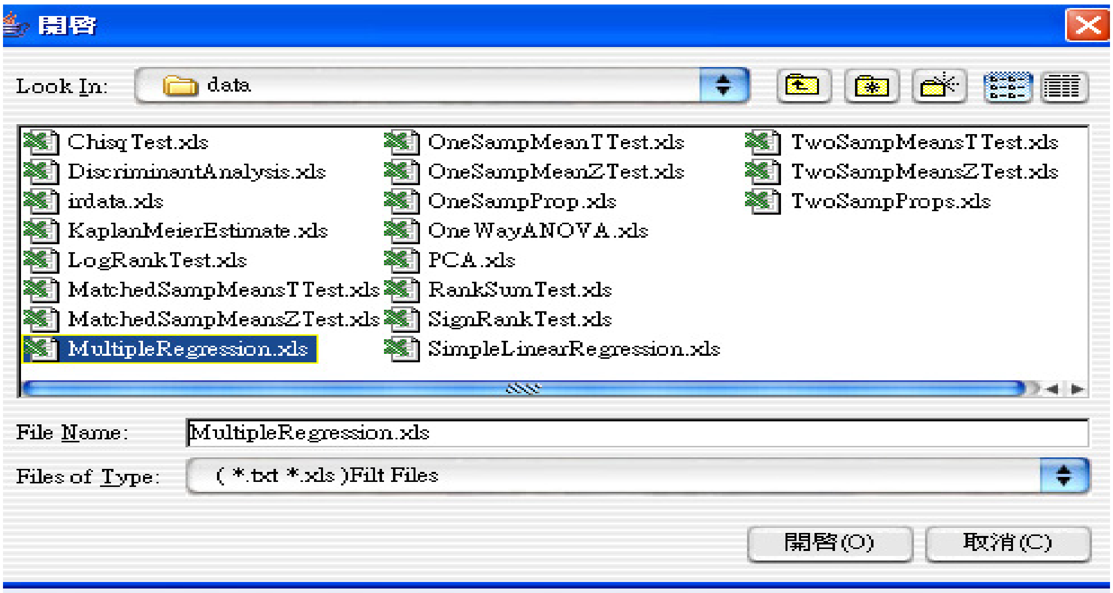
	A	B	C	D	E
1	1				
2	3				
3	4				
4	5				
5	6				
6	3				
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					

2. From file

(i)



(ii)



Look In: data

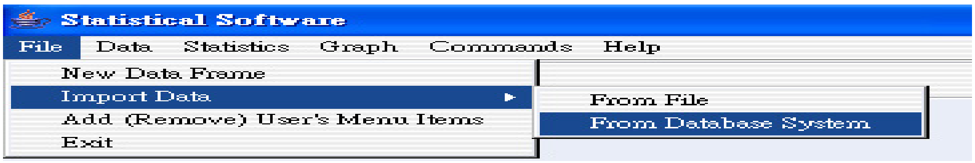
ChisqTest.xls
DiscriminantAnalysis.xls
irdata.xls
KaplanMeierEstimate.xls
LogRankTest.xls
MatchedSampMeansTTest.xls
MatchedSampMeansZTest.xls
MultipleRegression.xls
OneSampMeanTTest.xls
OneSampMeanZTest.xls
OneSampProp.xls
OneWayANOVA.xls
PCA.xls
RankSumTest.xls
SignRankTest.xls
SimpleLinearRegression.xls
TwoSampMeansTTest.xls
TwoSampMeansZTest.xls
TwoSampProps.xls

File Name: MultipleRegression.xls

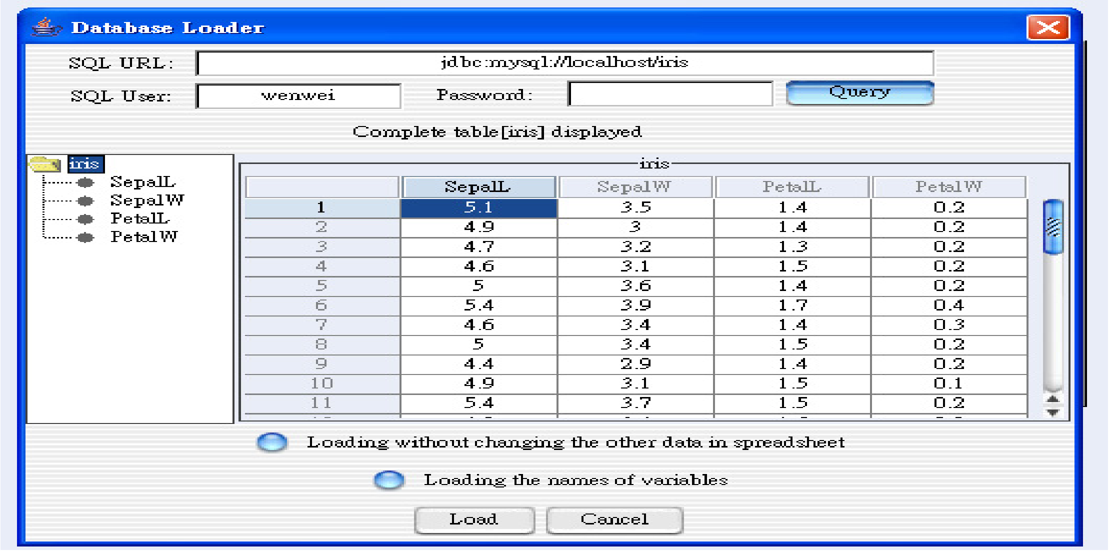
Files of Type: (*.txt *.xls)Filt Files

3. From database management system (DBMS)

(i)



(ii)



SQL URL: jdbc:mysql://localhost/iris

SQL User: wenwei

Password:

Complete table[iris] displayed

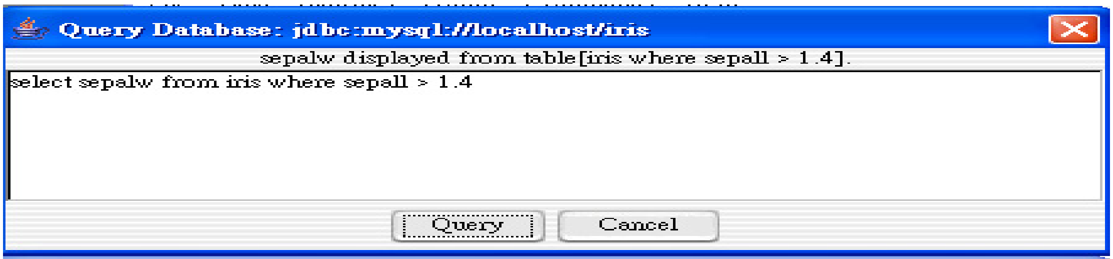
iris

	SepalL	SepalW	PetalL	PetalW
1	5.1	3.5	1.4	0.2
2	4.9	3	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1
11	5.4	3.7	1.5	0.2

○ Loading without changing the other data in spreadsheet

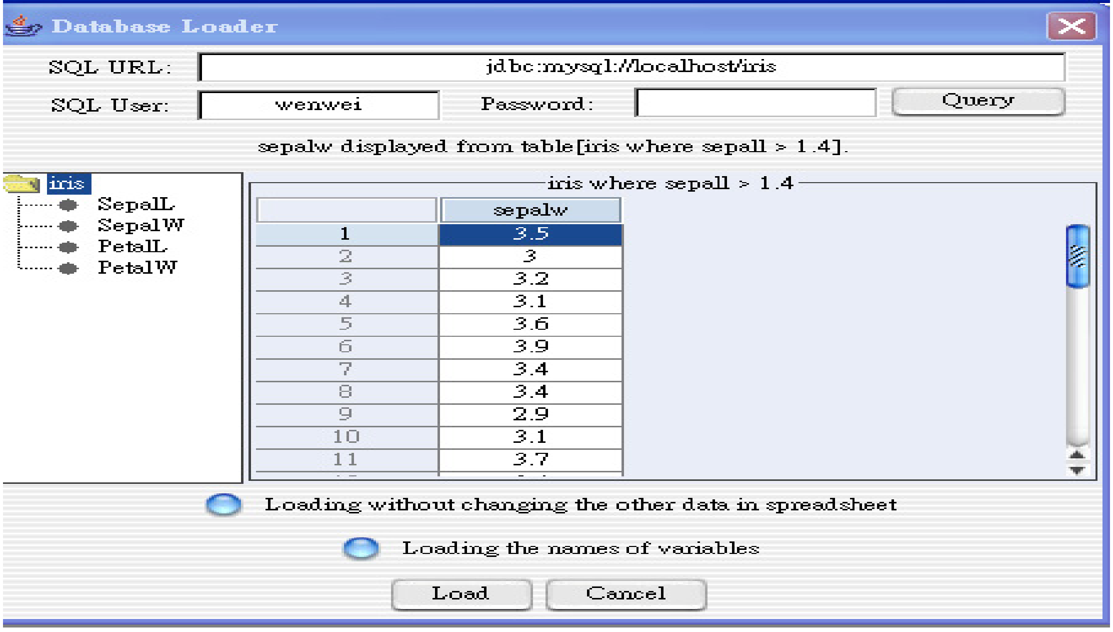
○ Loading the names of variables

Note: the user can enter the SQL queries by pushing Query button.



sepalw displayed from table[iris where sepall > 1.4].

```
select sepalw from iris where sepall > 1.4
```



SQL URL: jdbc:mysql://localhost/iris

SQL User: wenwei

Password:

sepalw displayed from table[iris where sepall > 1.4].

iris where sepall > 1.4

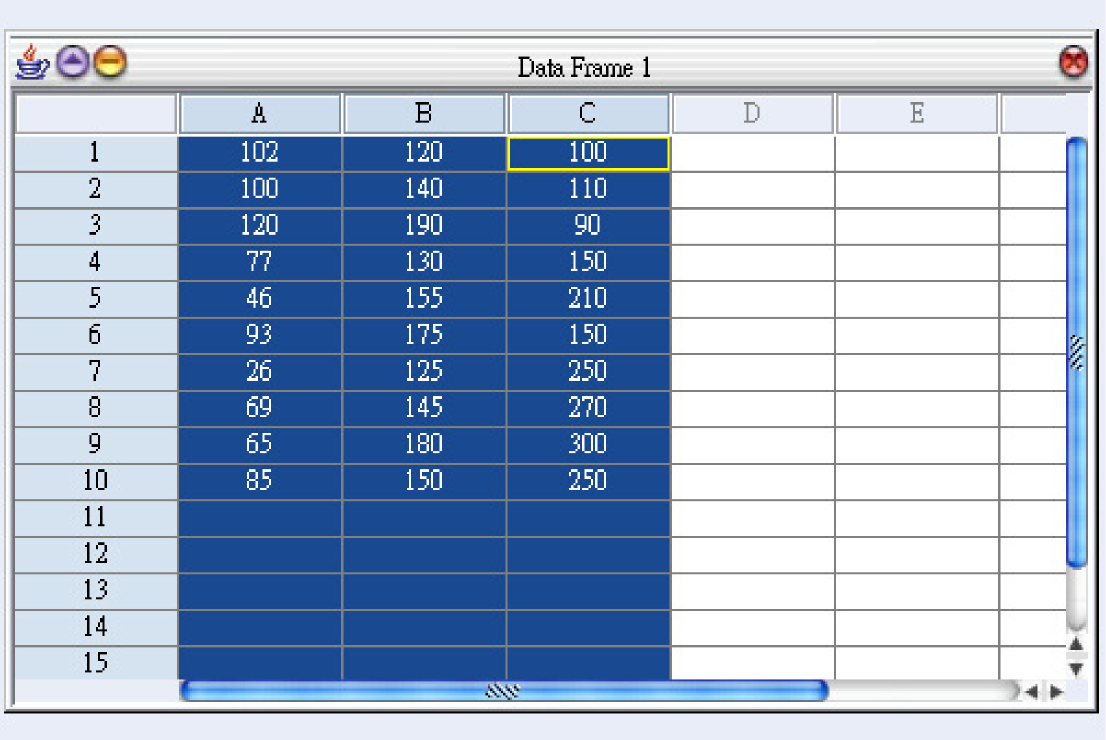
	sepalw
1	3.5
2	3
3	3.2
4	3.1
5	3.6
6	3.9
7	3.4
8	3.4
9	2.9
10	3.1
11	3.7

○ Loading without changing the other data in spreadsheet

○ Loading the names of variables

C. Data Selection: Column Selection and Row Selection

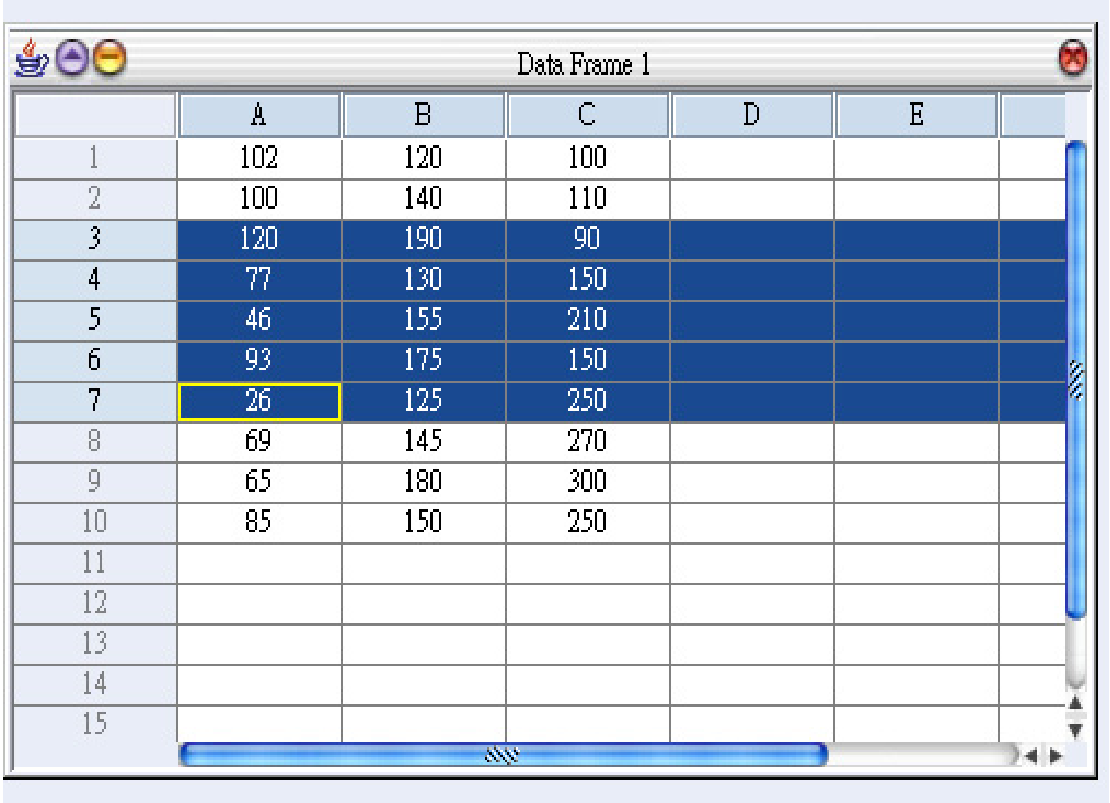
(i) Column selection: clicking on column buttons



Data Frame 1

	A	B	C	D	E
1	102	120	100		
2	100	140	110		
3	120	190	90		
4	77	130	150		
5	46	155	210		
6	93	175	150		
7	26	125	250		
8	69	145	270		
9	65	180	300		
10	85	150	250		
11					
12					
13					
14					
15					

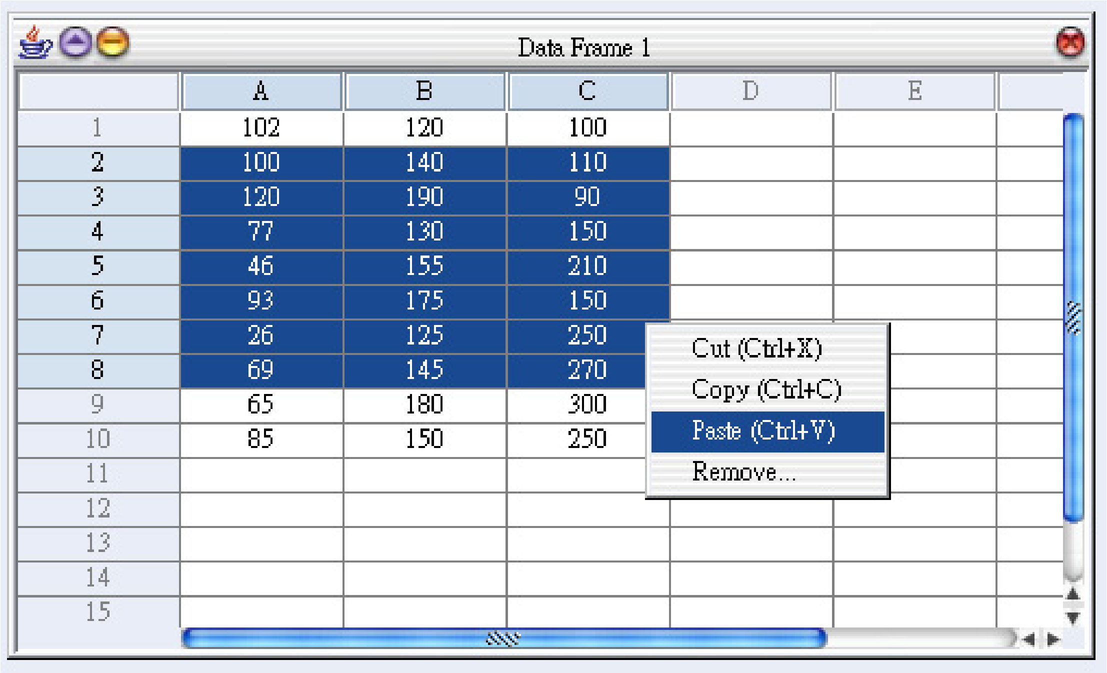
(ii) Row selection: clicking on row buttons



Data Frame 1

	A	B	C	D	E
1	102	120	100		
2	100	140	110		
3	120	190	90		
4	77	130	150		
5	46	155	210		
6	93	175	150		
7	26	125	250		
8	69	145	270		
9	65	180	300		
10	85	150	250		
11					
12					
13					
14					
15					

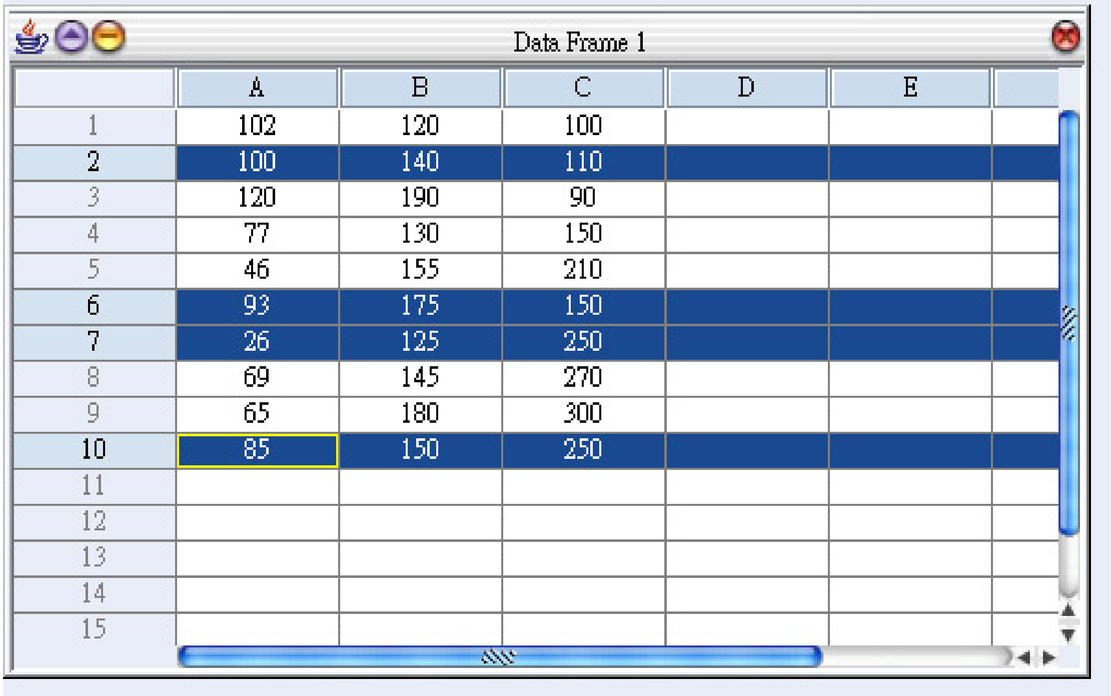
(iii) Selects cells within certain range: clicking on these cells



	A	B	C	D	E
1	102	120	100		
2	100	140	110		
3	120	190	90		
4	77	130	150		
5	46	155	210		
6	93	175	150		
7	26	125	250		
8	69	145	270		
9	65	180	300		
10	85	150	250		
11					
12					
13					
14					
15					

(iv) Selects inconsecutive rows or columns:

1. choosing cells of certain row or column,
2. pressing on *Ctrl* key to select the other rows or columns.



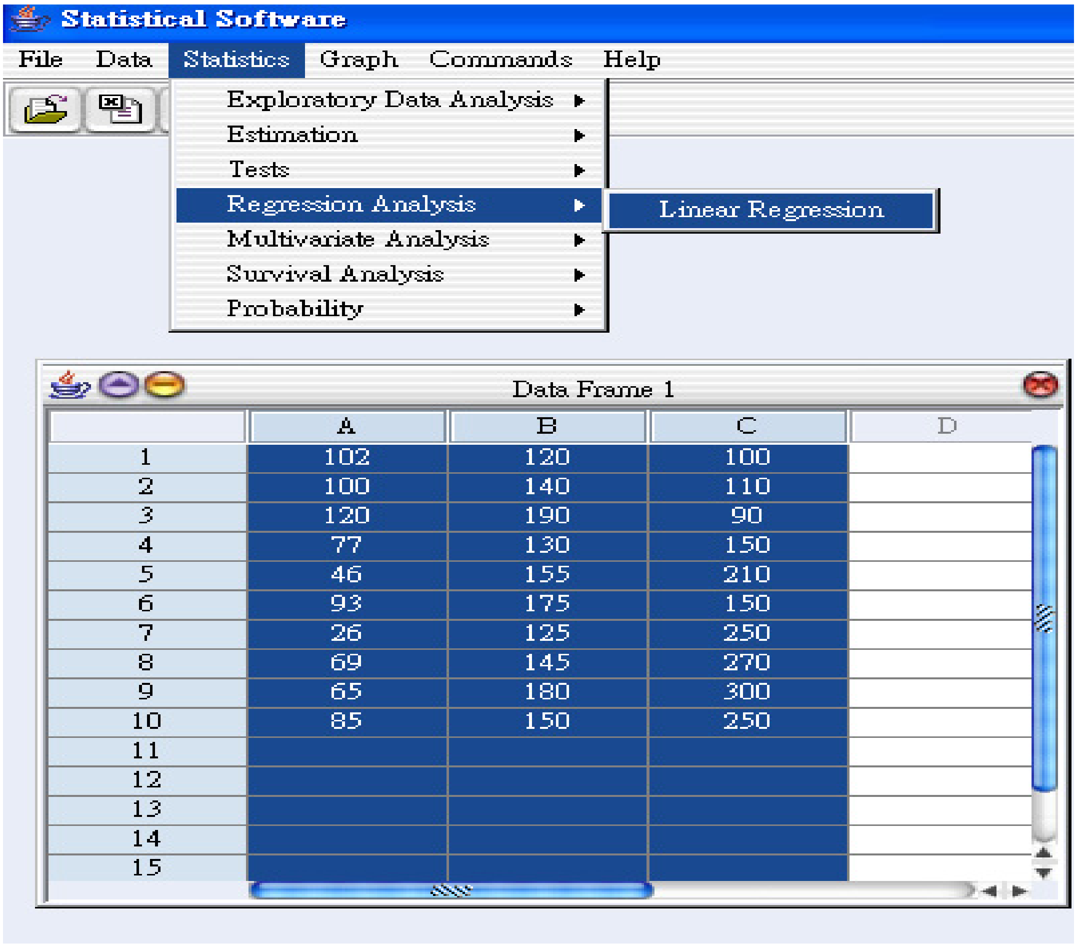
	A	B	C	D	E
1	102	120	100		
2	100	140	110		
3	120	190	90		
4	77	130	150		
5	46	155	210		
6	93	175	150		
7	26	125	250		
8	69	145	270		
9	65	180	300		
10	85	150	250		
11					
12					
13					
14					
15					

Note: by clicking on the blank button in the spreadsheet, all rows and columns will be selected.

Note: several functions, including pasting, coping, moving, and cutting the cells in the spreadsheet, are supported.

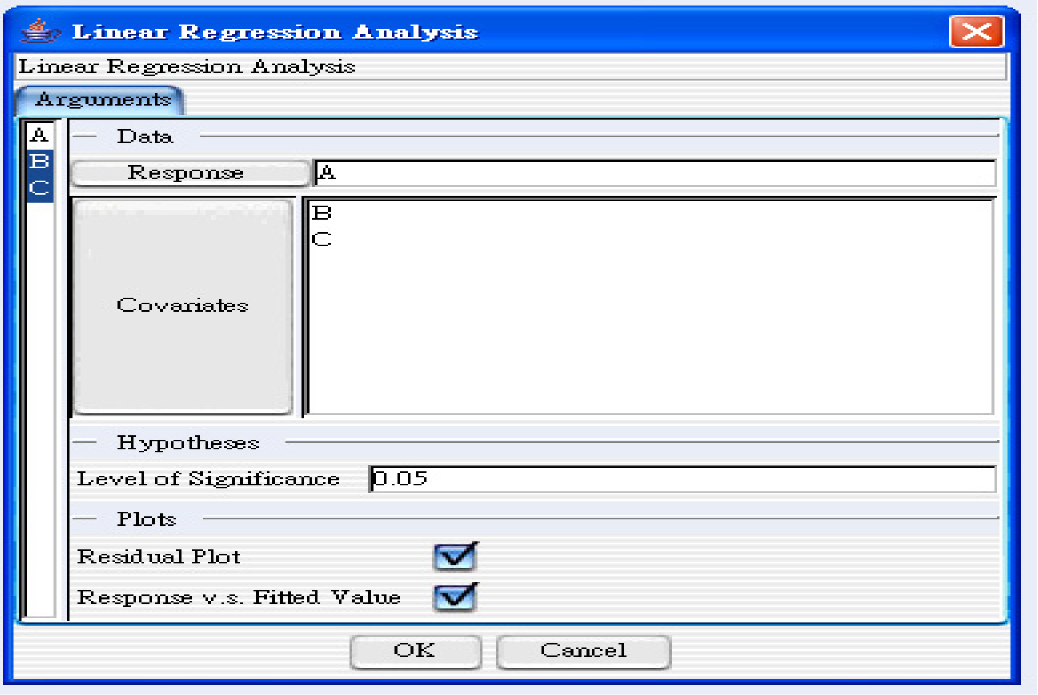
D. Statistical Analysis: Linear Regression Analysis

(i) Selects the menus



	A	B	C	D
1	102	120	100	
2	100	140	110	
3	120	190	90	
4	77	130	150	
5	46	155	210	
6	93	175	150	
7	26	125	250	
8	69	145	270	
9	65	180	300	
10	85	150	250	
11				
12				
13				
14				
15				

(ii) Specifies the arguments



Data

Response: A

Covariates: B C

Hypotheses

Level of Significance: 0.05

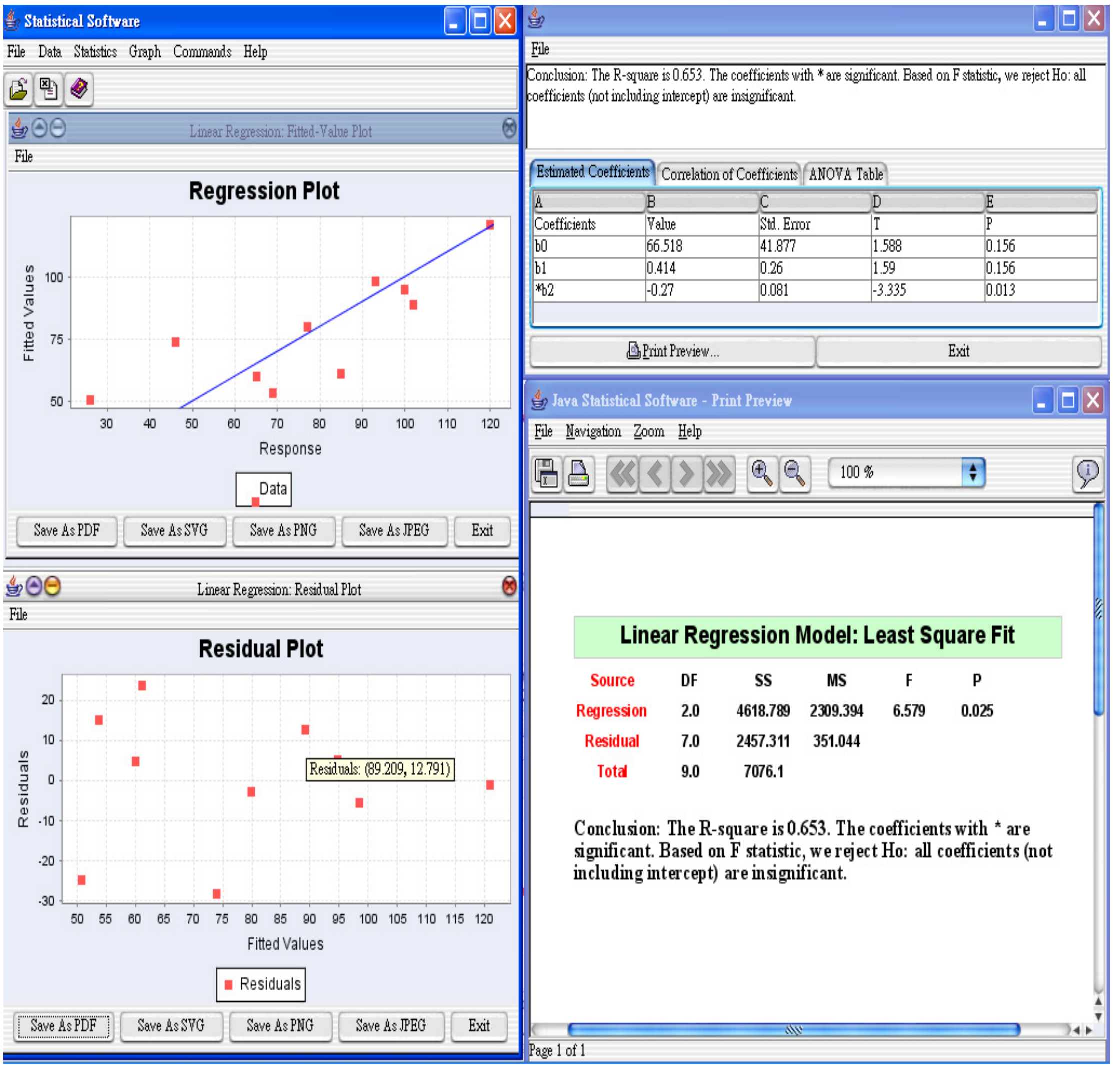
Plots

Residual Plot ☑

Response v.s. Fitted Value ☑

Note: uses the keys *Ctrl* or *Shift* to select multiple variables from the list in the dialog.

(iii) Output report, print preview, and graphical summary



Conclusion: The R-square is 0.653. The coefficients with * are significant. Based on F statistic, we reject Ho: all coefficients (not including intercept) are insignificant.

A	B	C	D	E
Coefficients	Value	Std. Error	T	P
b0	66.518	41.877	1.588	0.156
b1	0.414	0.26	1.59	0.156
*b2	-0.27	0.081	-3.335	0.013

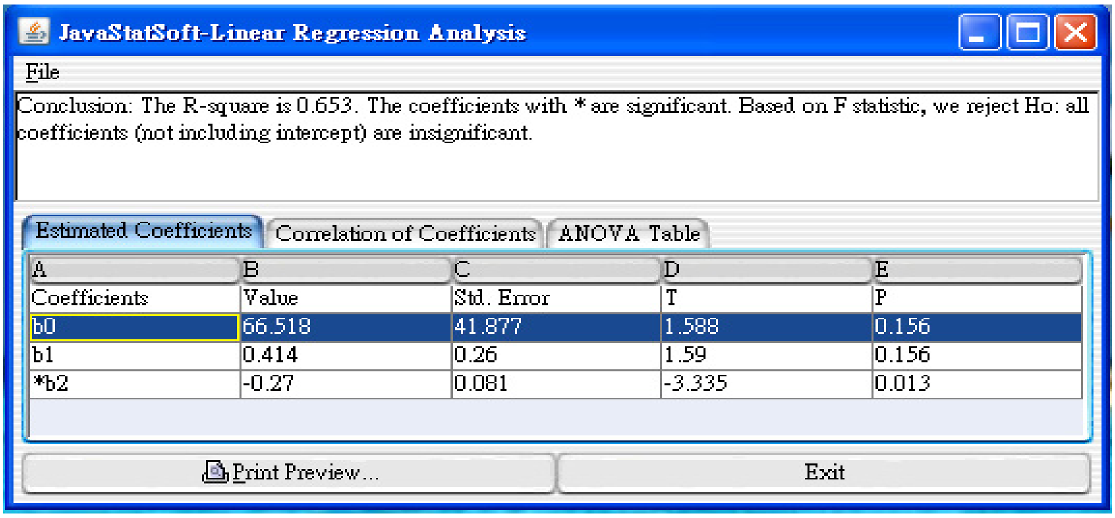
Linear Regression Model: Least Square Fit

Source	DF	SS	MS	F	P
Regression	2.0	4618.789	2309.394	6.579	0.025
Residual	7.0	2457.311	351.044		
Total	9.0	7076.1			

Conclusion: The R-square is 0.653. The coefficients with * are significant. Based on F statistic, we reject Ho: all coefficients (not including intercept) are insignificant.

(iv) Modifies output report

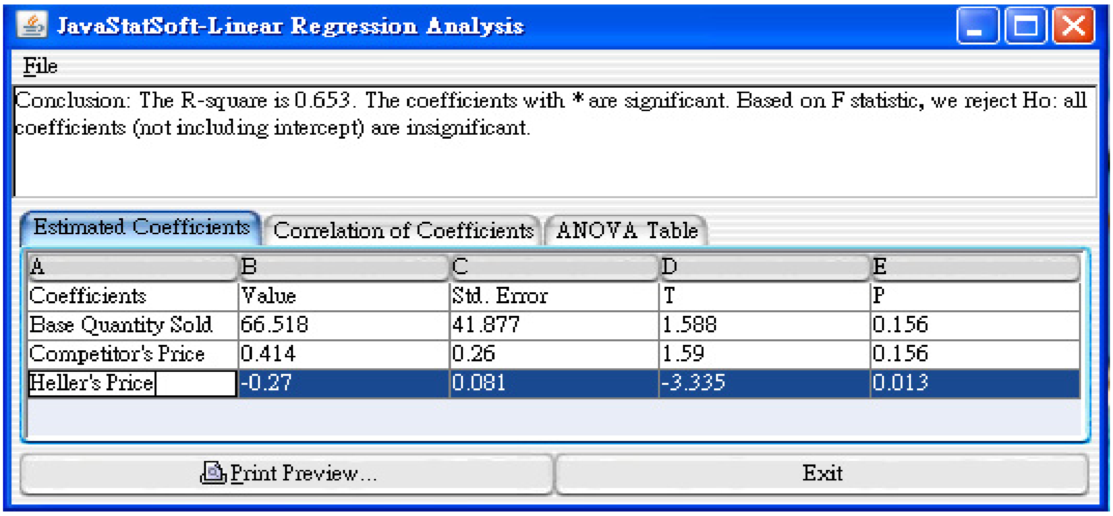
1. Original report


JavaStatSoft-Linear Regression Analysis
File
Estimated Coefficients
Correlation of Coefficients
ANOVA Table
Print Preview...
Exit

Conclusion: The R-square is 0.653. The coefficients with * are significant. Based on F statistic, we reject Ho: all coefficients (not including intercept) are insignificant.

A	B	C	D	E
Coefficients	Value	Std. Error	T	P
b0	66.518	41.877	1.588	0.156
b1	0.414	0.26	1.59	0.156
*b2	-0.27	0.081	-3.335	0.013

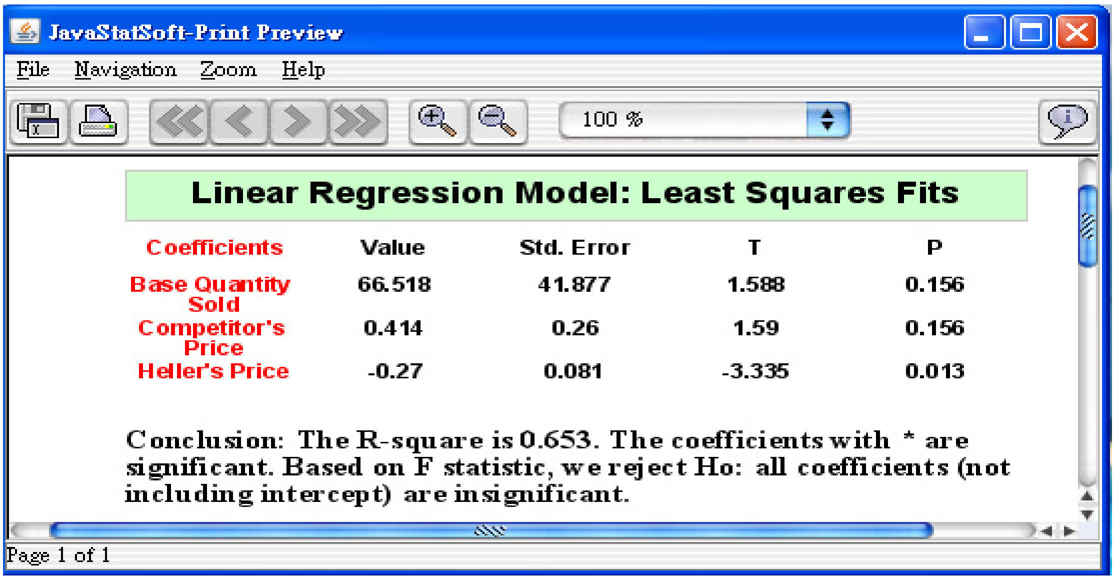
2. Enters the required texts


JavaStatSoft-Linear Regression Analysis
File
Estimated Coefficients
Correlation of Coefficients
ANOVA Table
Print Preview...
Exit

Conclusion: The R-square is 0.653. The coefficients with * are significant. Based on F statistic, we reject Ho: all coefficients (not including intercept) are insignificant.

A	B	C	D	E
Coefficients	Value	Std. Error	T	P
Base Quantity Sold	66.518	41.877	1.588	0.156
Competitor's Price	0.414	0.26	1.59	0.156
Heller's Price	-0.27	0.081	-3.335	0.013

3. Generates new print preview



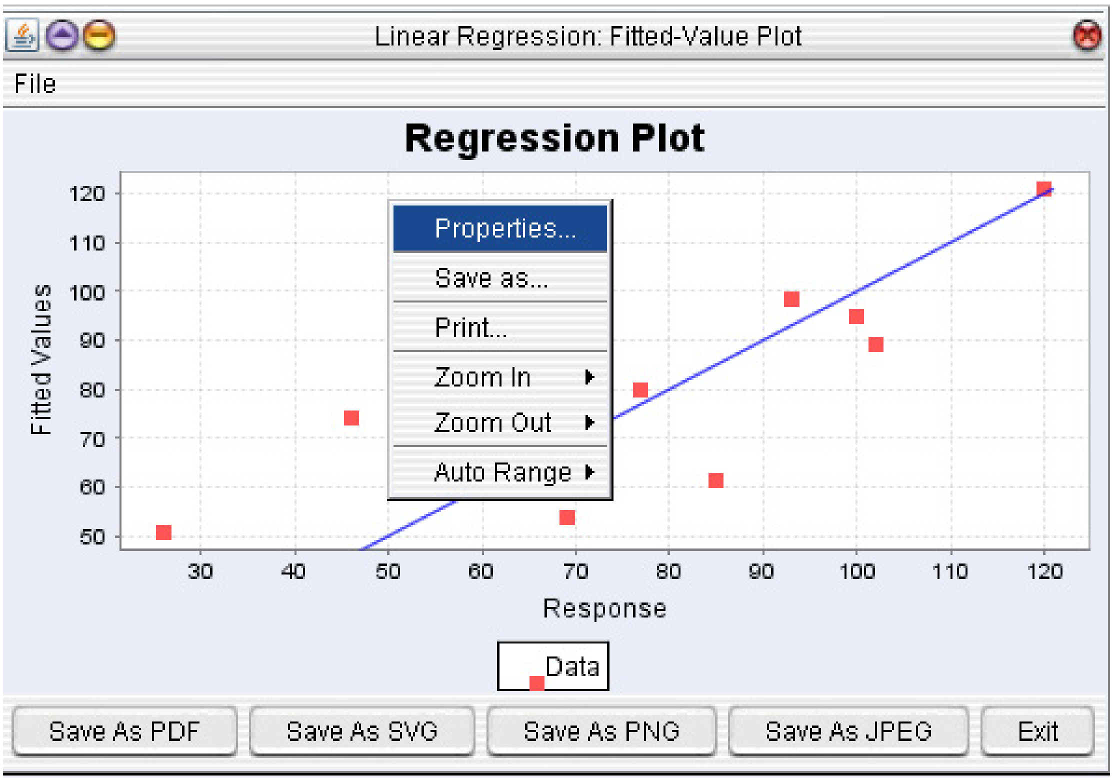
Linear Regression Model: Least Squares Fits

Coefficients	Value	Std. Error	T	P
Base Quantity Sold	66.518	41.877	1.588	0.156
Competitor's Price	0.414	0.26	1.59	0.156
Heller's Price	-0.27	0.081	-3.335	0.013

Conclusion: The R-square is 0.653. The coefficients with * are significant. Based on F statistic, we reject Ho: all coefficients (not including intercept) are insignificant.

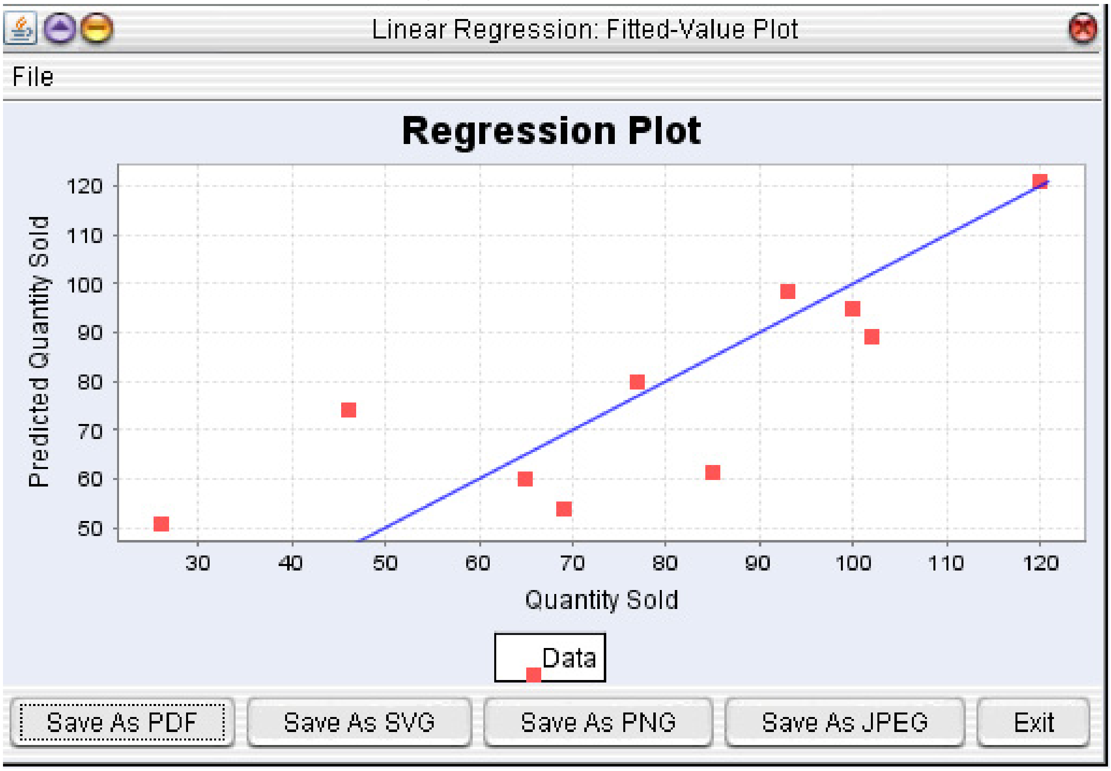
(v) Modifies graphical summary

1. **Clicks on the right button of the mouse and presses on "Properties.." item.**



2. Modifies the labels of x-axis and y-axis.

3. The labels of x-axis and y-axis have been modified.

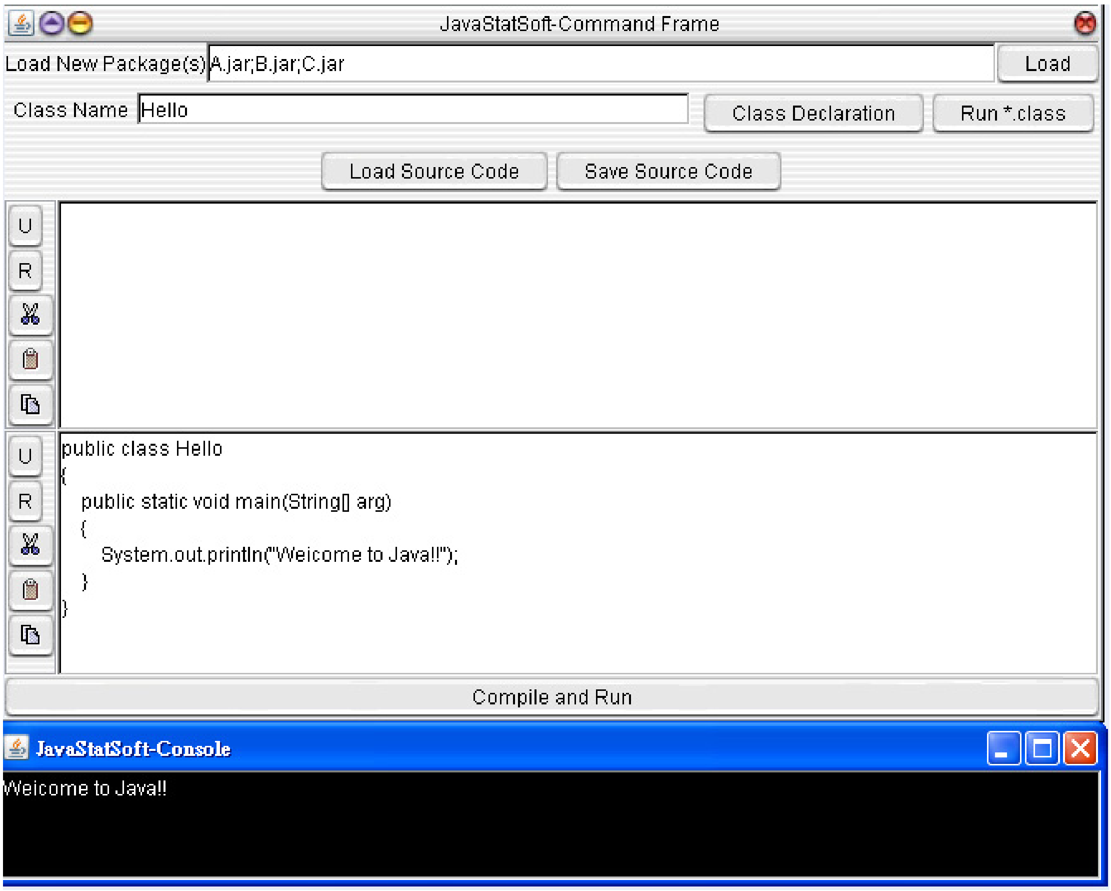


E. StatCompiler (Compile Java Source Code)

I. Two ways of compiling Java code: the compiled *.class file can be found in the directory "users".

(i) Enters the code directly

1. enters the class name first,
2. presses on "Class Declaration" button,
3. enters the code,
4. presses on "Compile and Run" button,
5. saves the code by pressing on "Save Source Code" button.



Load New Package(s) A.jar;B.jar;C.jar

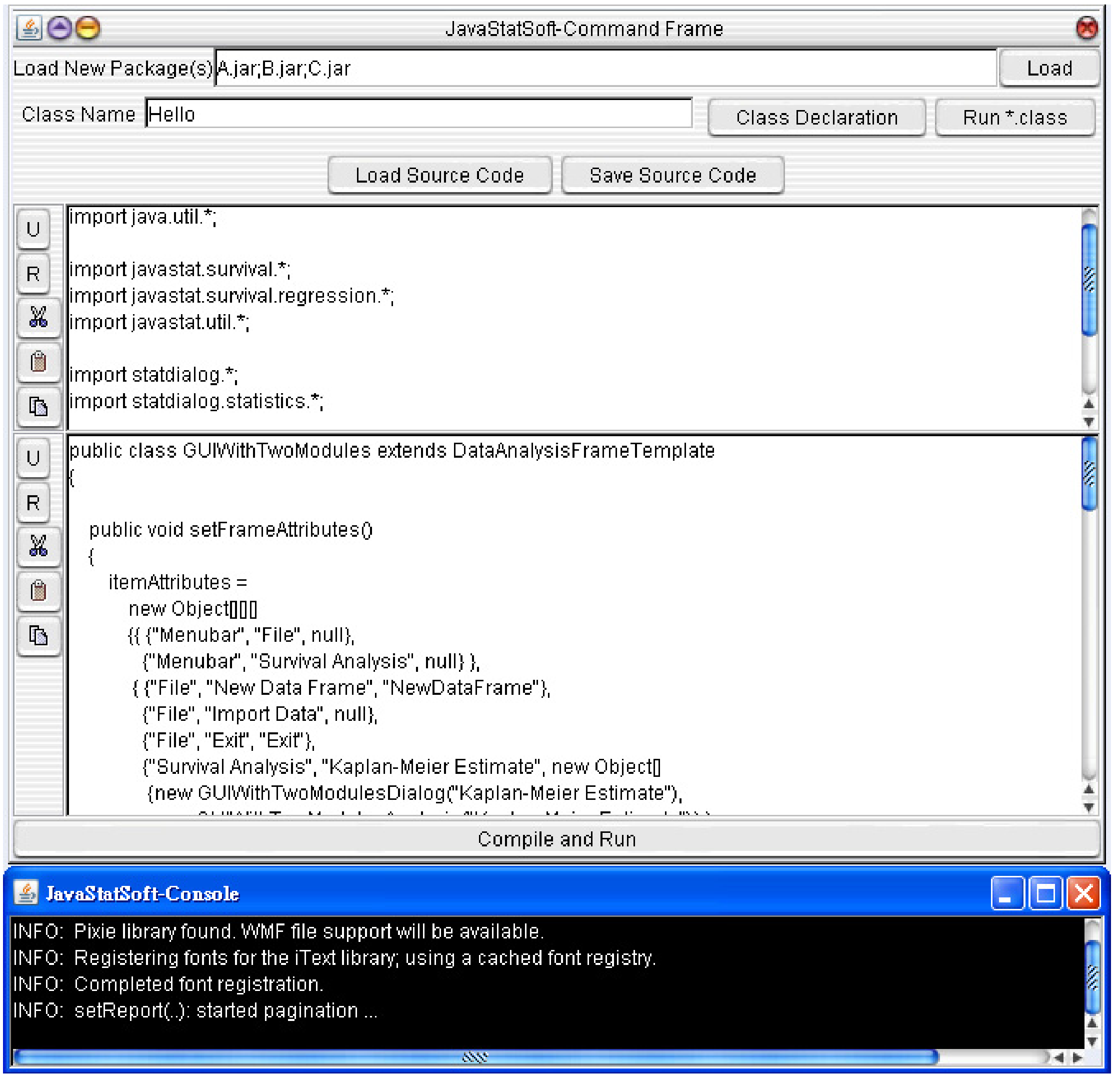
Class Name Hello

```
public class Hello
{
  public static void main(String[] arg)
  {
    System.out.println("Weicome to Java!!");
  }
}
```

```
Weicome to Java!!
```

(ii) Loads the source code

1. loads the code by pressing on "Load Source Code" button,
2. presses on "Compile and Run" button.



Load New Package(s) A.jar;B.jar;C.jar

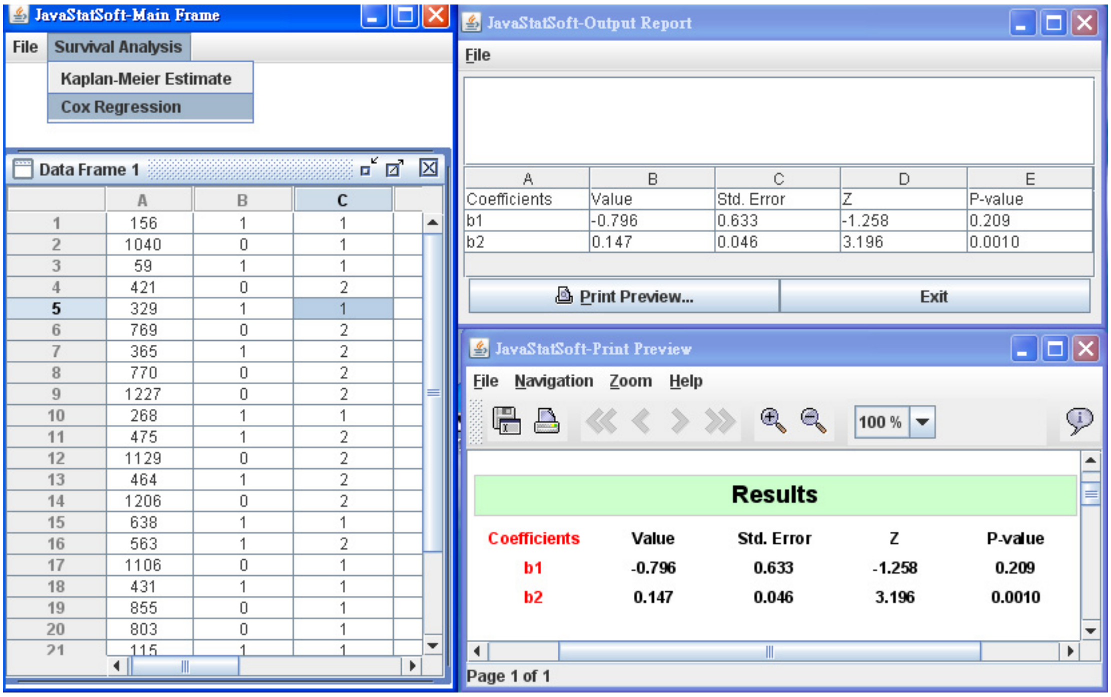
Class Name Hello

```
import java.util.*;

import javastat.survival.*;
import javastat.survival.regression.*;
import javastat.util.*;

import statdialog.*;
import statdialog.statistics.*;
```

```
public class GUIWithTwoModules extends DataAnalysisFrameTemplate
{

  public void setFrameAttributes()
  {
    itemAttributes =
      new Object[][][]
      {{ {"Menubar", "File", null},
         {"Menubar", "Survival Analysis", null} },
       { {"File", "New Data Frame", "NewDataFrame"},
         {"File", "Import Data", null},
         {"File", "Exit", "Exit"},
         {"Survival Analysis", "Kaplan-Meier Estimate", new Object[]
          {new GUIWithTwoModulesDialog("Kaplan-Meier Estimate"),
```

```
INFO:  Pixie library found. WMF file support will be available.
INFO:  Registering fonts for the iText library; using a cached font registry.
INFO:  Completed font registration.
INFO:  setReport(..): started pagination ...
```

The results for running the code are



	A	B	C
1	156	1	1
2	1040	0	1
3	59	1	1
4	421	0	2
5	329	1	1
6	769	0	2
7	365	1	2
8	770	0	2
9	1227	0	2
10	268	1	1
11	475	1	2
12	1129	0	2
13	464	1	2
14	1206	0	2
15	638	1	1
16	563	1	2
17	1106	0	1
18	431	1	1
19	855	0	1
20	803	0	1
21	115	1	1

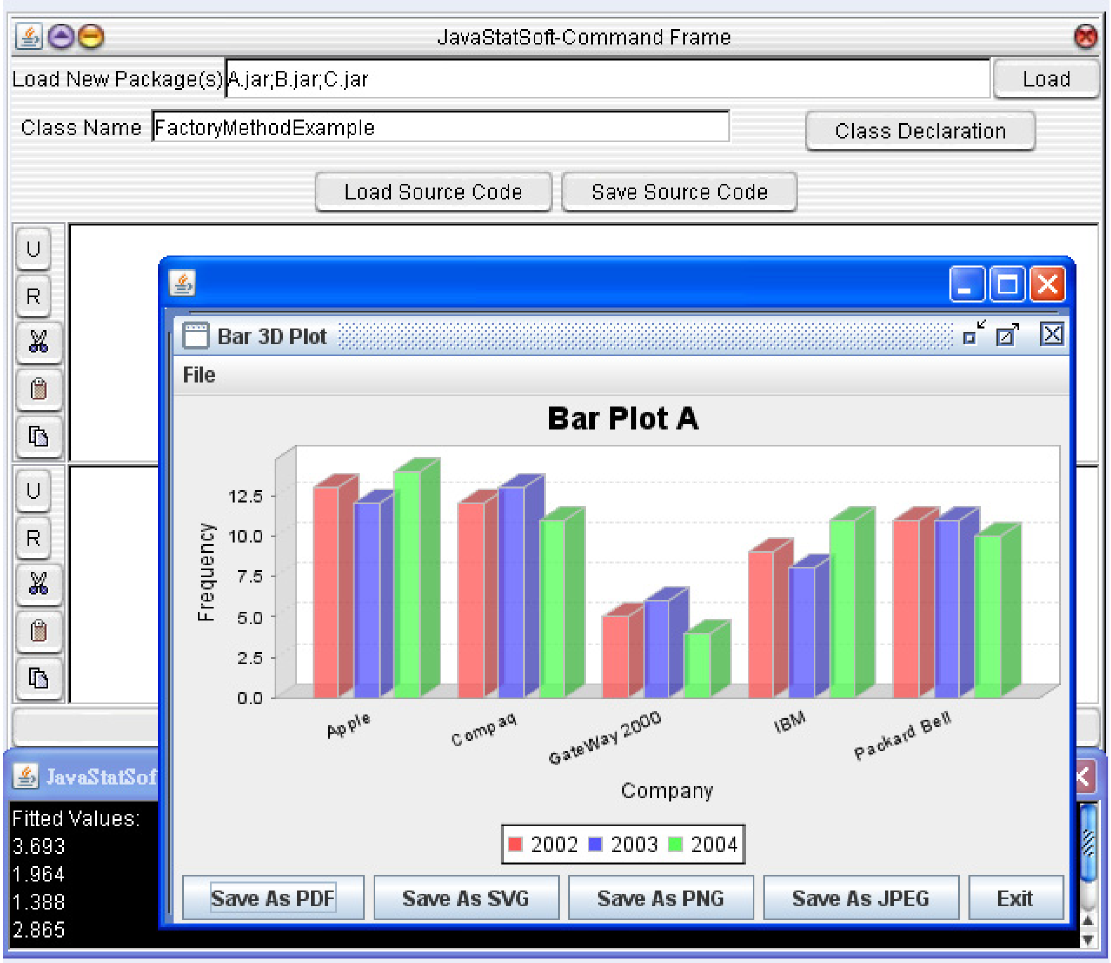
A	B	C	D	E
Coefficients	Value	Std. Error	Z	P-value
b1	-0.796	0.633	-1.258	0.209
b2	0.147	0.046	3.196	0.0010

Results

Coefficients	Value	Std. Error	Z	P-value
b1	-0.796	0.633	-1.258	0.209
b2	0.147	0.046	3.196	0.0010

II. Running the compiled *.class files in the directory "users" directly: the source code might be compiled by other IDEs and put in the directory.

1. **Enters the class name first, for example, the class name "FactoryMethodExample" in the figure,**
2. **presses on "Run *.class" button.**



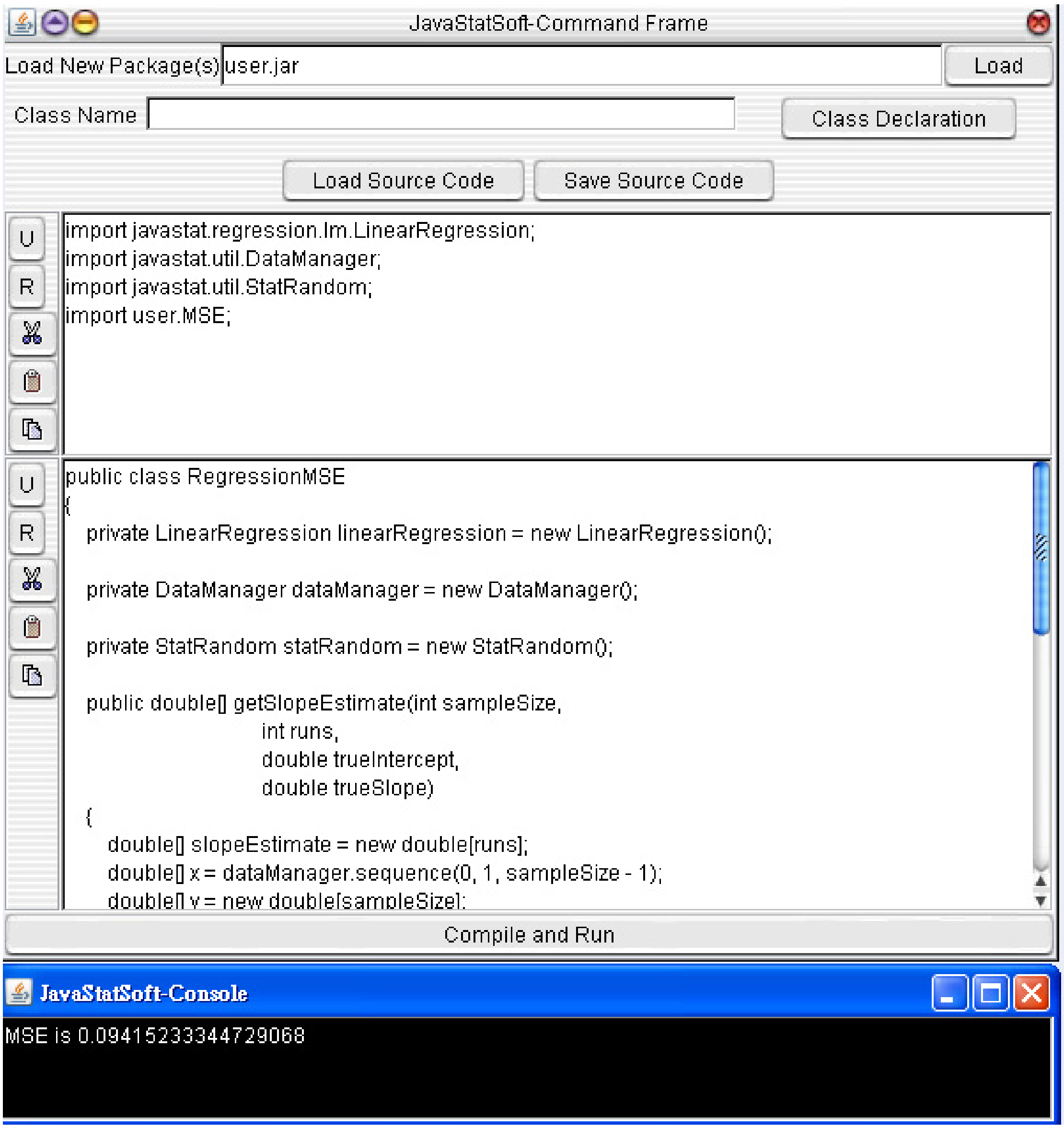
Load New Package(s) A.jar;B.jar;C.jar

Class Name FactoryMethodExample

```
Fitted Values:
3.693
1.964
1.388
2.865
```

III. Using JAR files

1. **Puts the *.jar files in the directory in which the jar file javastatsoft_beta1.3.jar was put,**
2. **enters the full jar names, for example, "user.jar" in the figure below,**
3. **presses on "Load" button and the classes in the loaded jar files can be used, for example, the class "MSE" contained in "user.jar" file being accessed by the class "RegressionMSE" in the figure below.**



Load New Package(s): user.jar

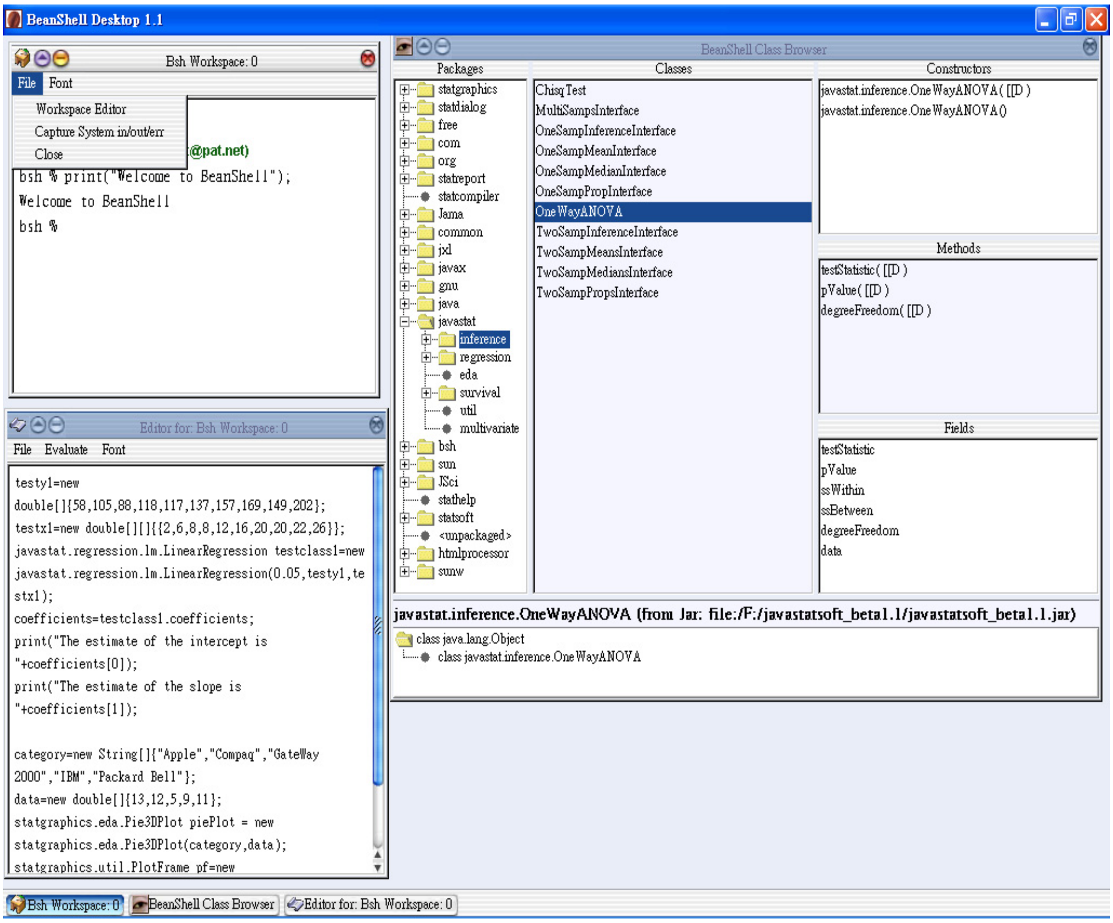
```
import javastat.regression.lm.LinearRegression;
import javastat.util.DataManager;
import javastat.util.StatRandom;
import user.MSE;
```

```
public class RegressionMSE
{
   private LinearRegression linearRegression = new LinearRegression();

   private DataManager dataManager = new DataManager();

   private StatRandom statRandom = new StatRandom();

   public double[] getSlopeEstimate(int sampleSize,
                                    int runs,
                                    double trueIntercept,
                                    double trueSlope)
   {
      double[] slopeEstimate = new double[runs];
      double[] x = dataManager.sequence(0, 1, sampleSize - 1);
      double[] y = new double[sampleSize];
```

```
MSE is 0.09415233344729068
```

F. BeanShell (Scripting Language)

Look and Feel



```
bsh % print("Welcome to BeanShell");
Welcome to BeanShell
bsh %
```

```
testy1=new
double[]{58,105,88,118,117,137,157,169,149,202};
testx1=new double[][]{{2,6,8,8,12,16,20,20,22,26}};
javastat.regression.lm.LinearRegression testclass1=new
javastat.regression.lm.LinearRegression(0.05,testy1,te
stx1);
coefficients=testclass1.coefficients;
print("The estimate of the intercept is
"+coefficients[0]);
print("The estimate of the slope is
"+coefficients[1]);

category=new String[]{"Apple","Compaq","GateWay
2000","IBM","Packard Bell"};
data=new double[]{13,12,5,9,11};
statgraphics.eda.Pie3DPlot piePlot = new
statgraphics.eda.Pie3DPlot(category,data);
statgraphics.util.PlotFrame pf=new
```

Writes scripts

Two ways of writing scripts:

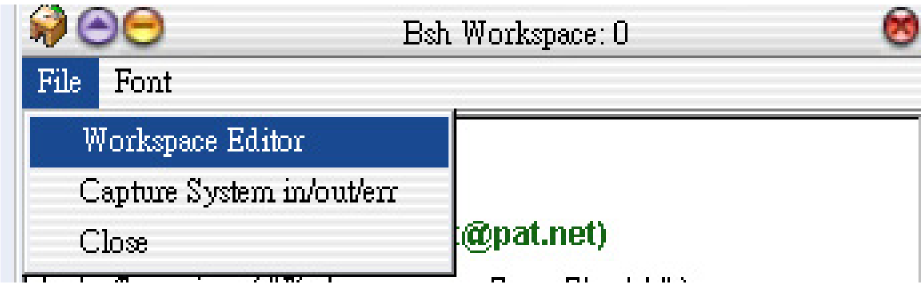
1. **enters scripts in "Bsh Workspace",**
2.

(i) **invokes the editor in BeanShell by selecting "File→Workspace Editor",**

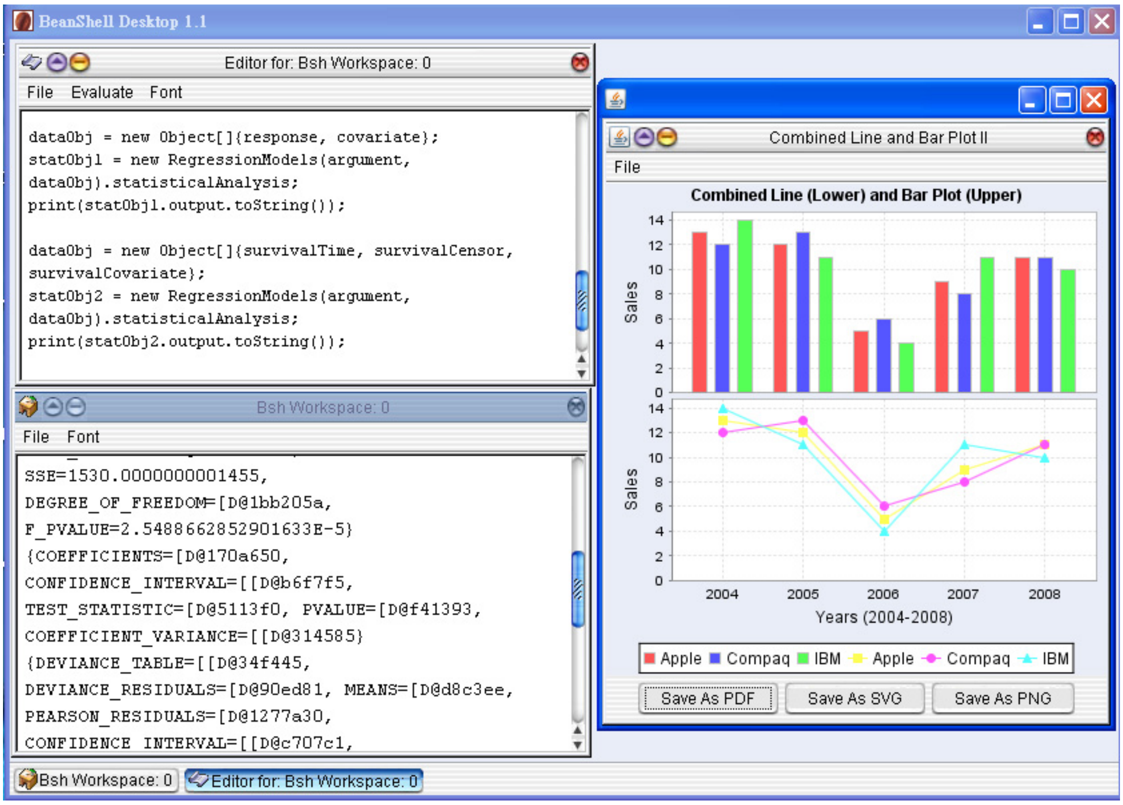
(ii) **enters the scripts,**

(iii) **evaluates by pressing on "Evaluate→Eval in Workspace".**

(i)



(ii), (iii)

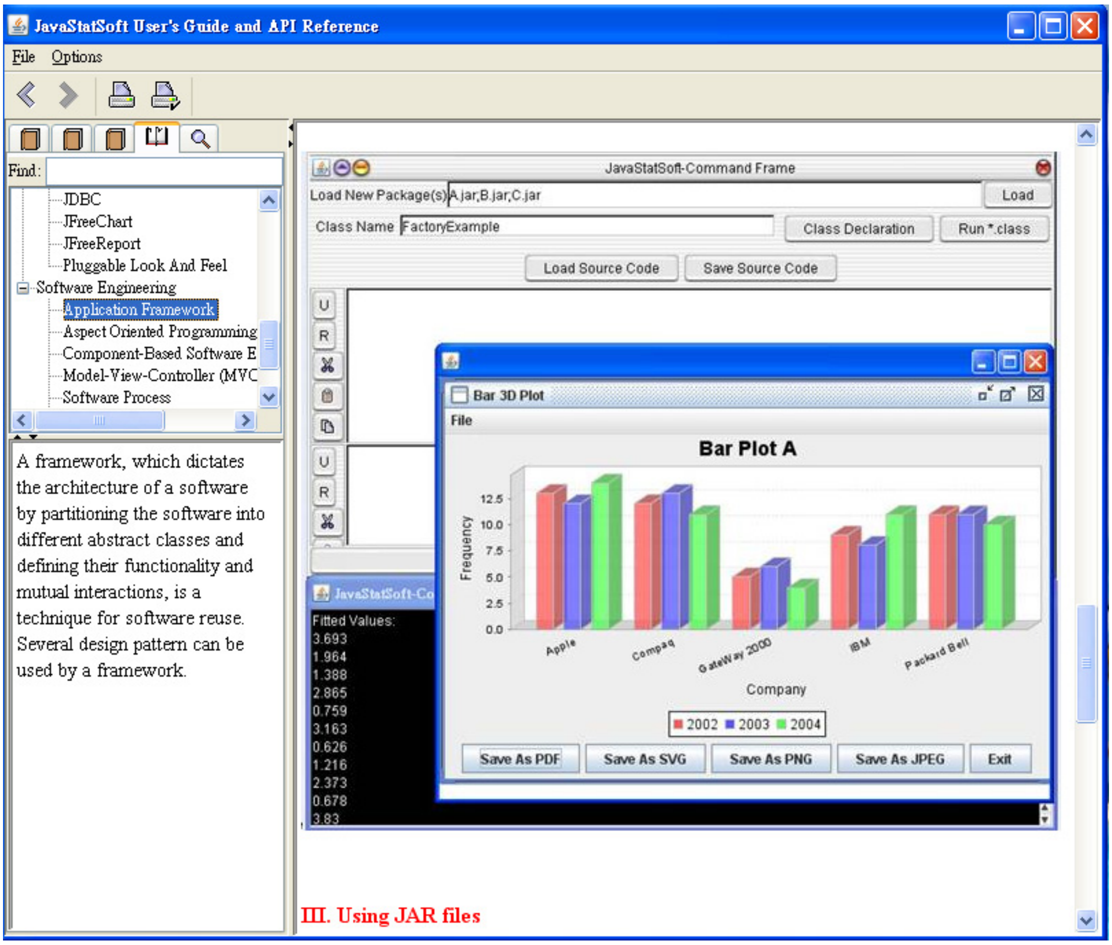


```
dataObj = new Object[]{response, covariate};
statObj1 = new RegressionModels(argument,
dataObj).statisticalAnalysis;
print(statObj1.output.toString());

dataObj = new Object[]{survivalTime, survivalCensor,
survivalCovariate};
statObj2 = new RegressionModels(argument,
dataObj).statisticalAnalysis;
print(statObj2.output.toString());
```

```
SSE=1530.0000000001455,
DEGREE_OF_FREEDOM=[D@1bb205a,
F_PVALUE=2.5488662852901633E-5}
{COEFFICIENTS=[D@170a650,
CONFIDENCE_INTERVAL=[[D@b6f7f5,
TEST_STATISTIC=[D@5113f0, PVALUE=[D@f41393,
COEFFICIENT_VARIANCE=[[D@314585}
{DEVIANCE_TABLE=[[D@34f445,
DEVIANCE_RESIDUALS=[D@90ed81, MEANS=[D@d8c3ee,
PEARSON_RESIDUALS=[D@1277a30,
CONFIDENCE_INTERVAL=[[D@c707c1,
```

G. Help System

1. Look and Feel:



A framework, which dictates the architecture of a software by partitioning the software into different abstract classes and defining their functionality and mutual interactions, is a technique for software reuse. Several design pattern can be used by a framework.

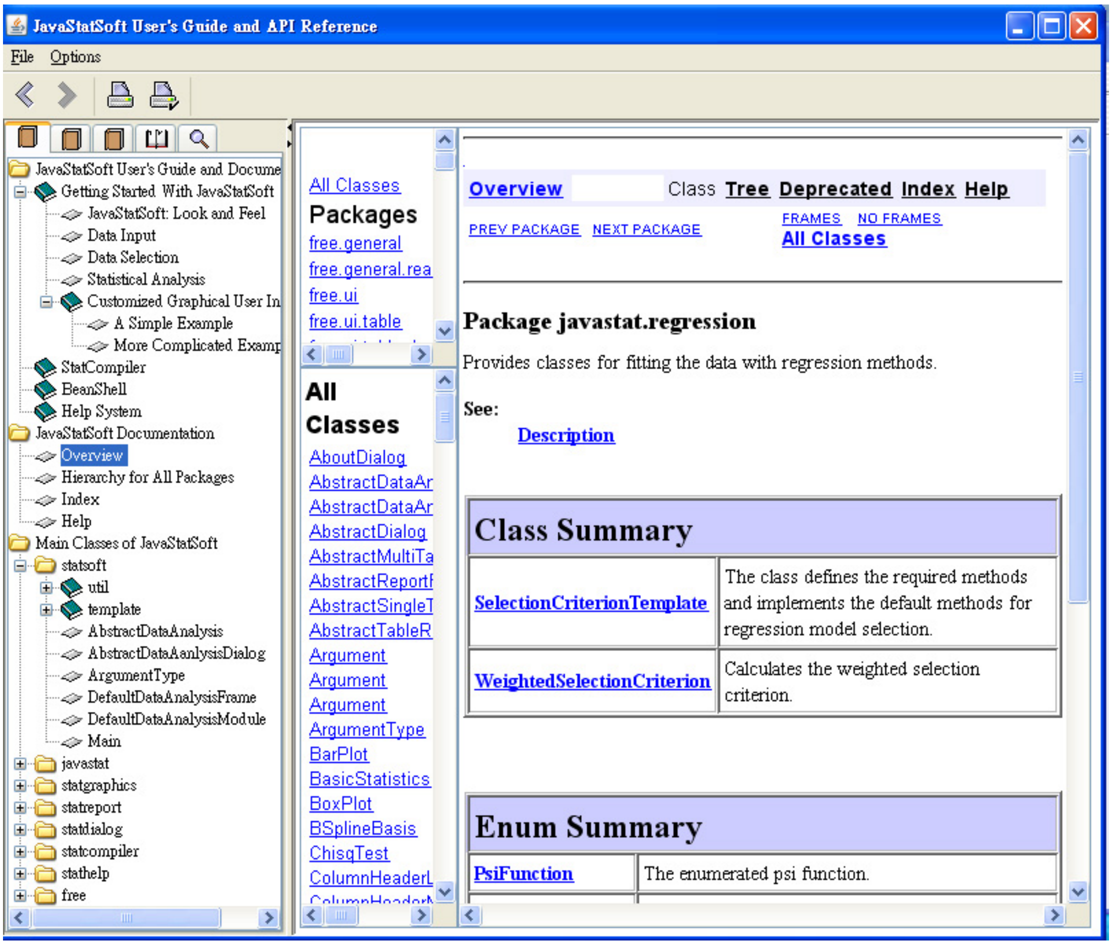
Load New Package(s) A.jar,B.jar,C.jar

Class Name FactoryExample

```
Fitted Values:
3.693
1.964
1.388
2.865
0.759
3.163
0.626
1.216
2.373
0.678
3.83
```

III. Using JAR files

2. User’s guide and API reference:



Package javastat.regression

Provides classes for fitting the data with regression methods.

See:

Description

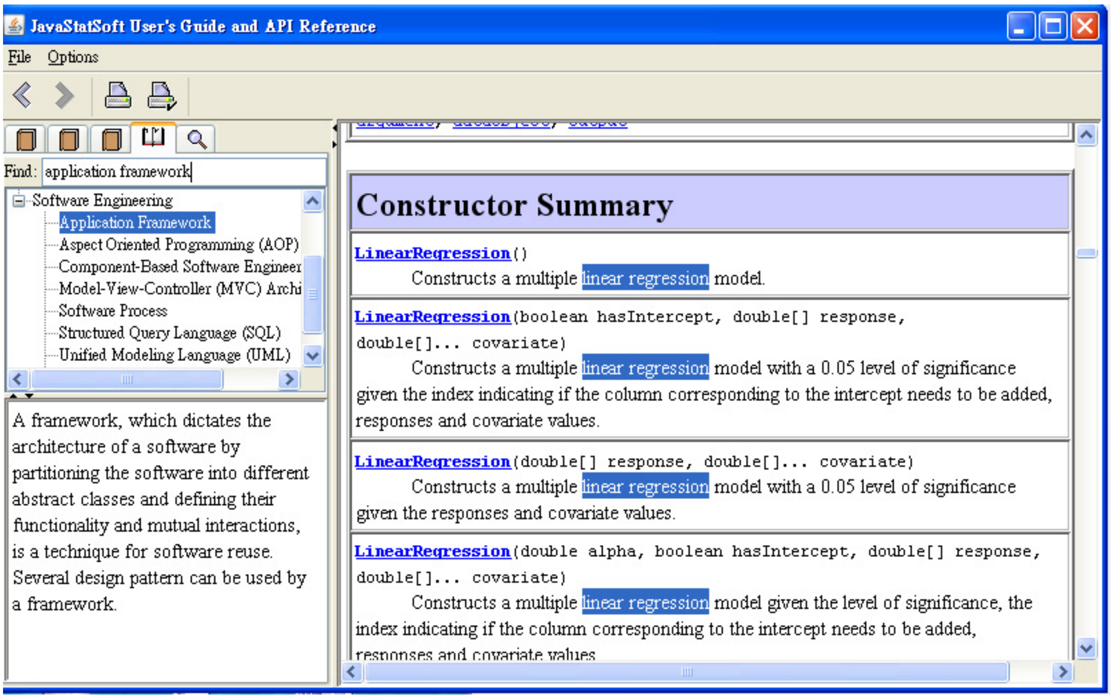
Class Summary

SelectionCriterionTemplate	The class defines the required methods and implements the default methods for regression model selection.
WeightedSelectionCriterion	Calculates the weighted selection criterion.

Enum Summary

PsiFunction	The enumerated psi function.

3. Glossary:


JavaStatSoft User's Guide and API Reference
File
Options
Software Engineering
Application Framework
Aspect Oriented Programming (AOP)
Component-Based Software Engineer
Model-View-Controller (MVC) Archi
Software Process
Structured Query Language (SQL)
Unified Modeling Language (UML)

Find: application framework

A framework, which dictates the architecture of a software by partitioning the software into different abstract classes and defining their functionality and mutual interactions, is a technique for software reuse. Several design pattern can be used by a framework.

Constructor Summary

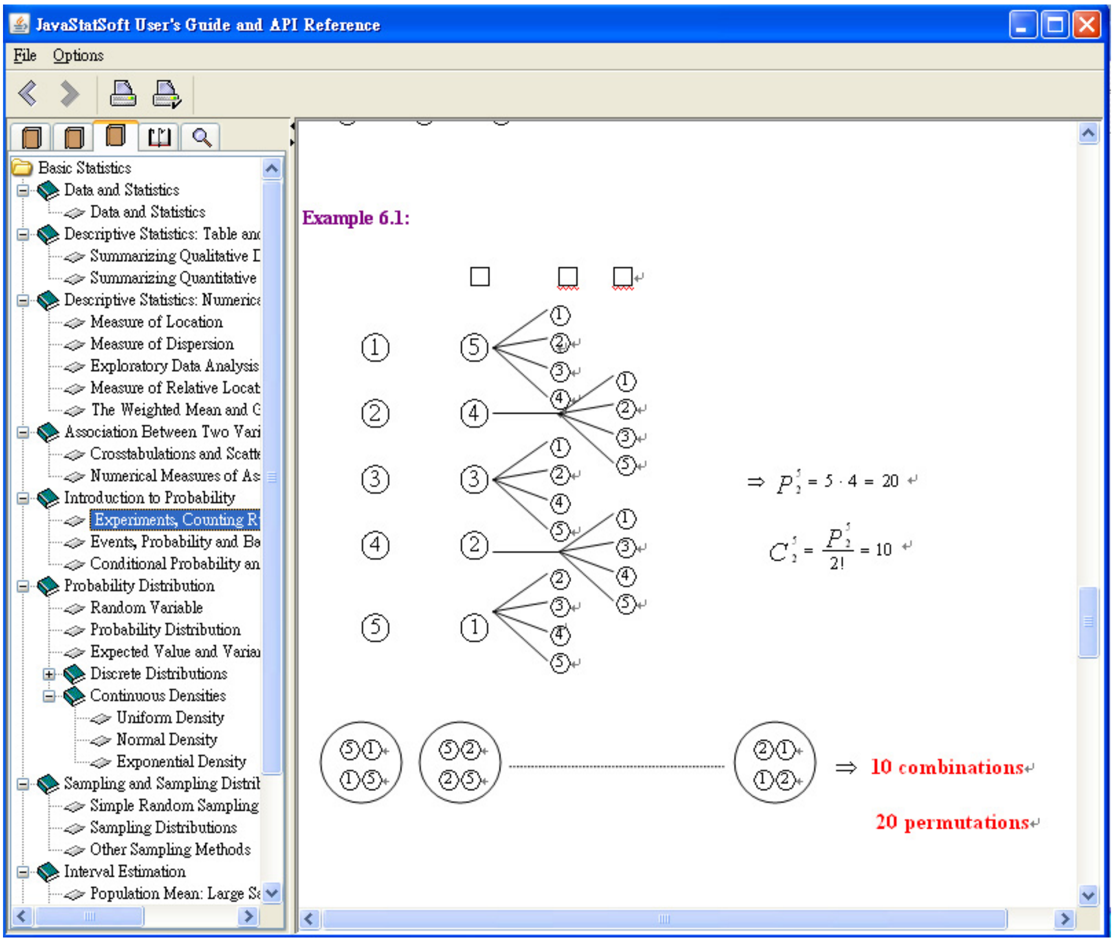
`LinearRegression()`
Constructs a multiple linear regression model.

`LinearRegression(boolean hasIntercept, double[] response, double[]... covariate)`
Constructs a multiple linear regression model with a 0.05 level of significance given the index indicating if the column corresponding to the intercept needs to be added, responses and covariate values.

`LinearRegression(double[] response, double[]... covariate)`
Constructs a multiple linear regression model with a 0.05 level of significance given the responses and covariate values.

`LinearRegression(double alpha, boolean hasIntercept, double[] response, double[]... covariate)`
Constructs a multiple linear regression model given the level of significance, the index indicating if the column corresponding to the intercept needs to be added, responses and covariate values.

4. Course notes:



Example 6.1:

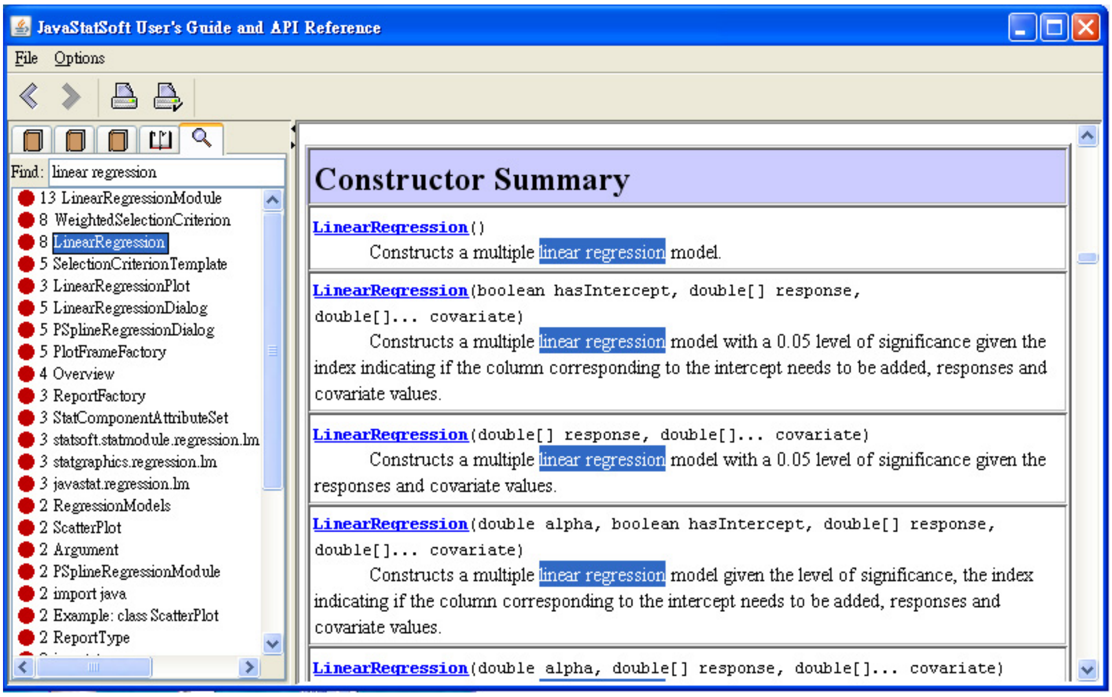
$\Rightarrow P^5_2 = 5 \cdot 4 = 20$

$C^5_2 = \frac{P^5_2}{2!} = 10$

$\Rightarrow$ 10 combinations

20 permutations

5. Full-text search:



- 13 LinearRegressionModule
- 8 WeightedSelectionCriterion
- 8 LinearRegression
- 5 SelectionCriterionTemplate
- 3 LinearRegressionPlot
- 5 LinearRegressionDialog
- 5 PSplineRegressionDialog
- 5 PlotFrameFactory
- 4 Overview
- 3 ReportFactory
- 3 StatComponentAttributeSet
- 3 statsoft.statmodule.regression.lm
- 3 statgraphics.regression.lm
- 3 javastat.regression.lm
- 2 RegressionModels
- 2 ScatterPlot
- 2 Argument
- 2 PSplineRegressionModule
- 2 import java
- 2 Example: class ScatterPlot
- 2 ReportType

Constructor Summary

`LinearRegression()`

Constructs a multiple linear regression model.

`LinearRegression(boolean hasIntercept, double[] response, double[]... covariate)`

Constructs a multiple linear regression model with a 0.05 level of significance given the index indicating if the column corresponding to the intercept needs to be added, responses and covariate values.

`LinearRegression(double[] response, double[]... covariate)`

Constructs a multiple linear regression model with a 0.05 level of significance given the responses and covariate values.

`LinearRegression(double alpha, boolean hasIntercept, double[] response, double[]... covariate)`

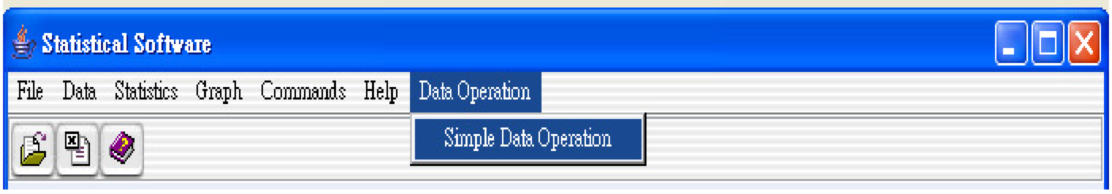
Constructs a multiple linear regression model given the level of significance, the index indicating if the column corresponding to the intercept needs to be added, responses and covariate values.

`LinearRegression(double alpha, double[] response, double[]... covariate)`

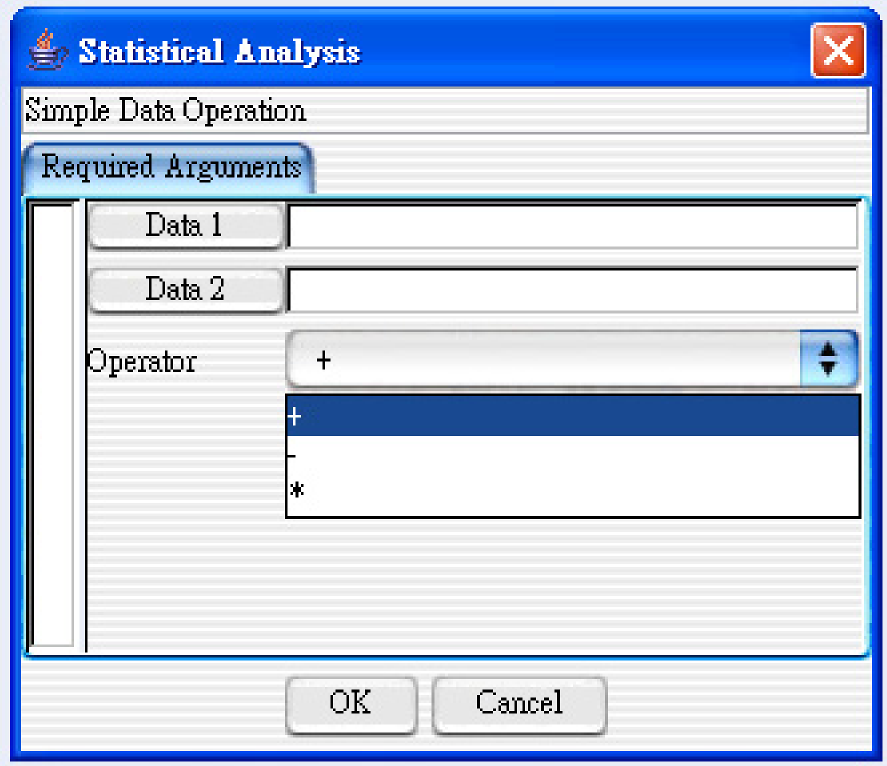
H. Customized Graphical User Interface : A Simple Example

Adds a menu:

Suppose a statistician wants to construct a module for computing the arithmetic operations of two input vectors and add a menu with a menu item for the module to the window of JavaStatSoft. The menu bar of JavaStatSoft then looks like



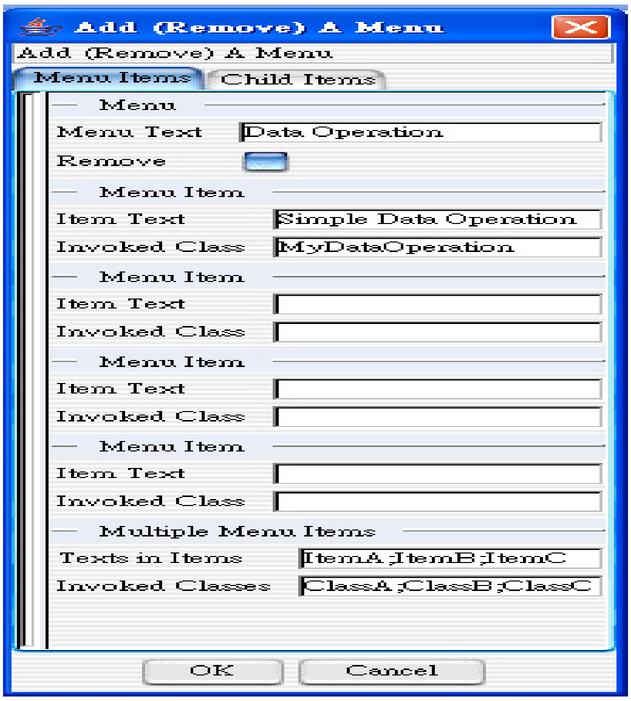
The dialog, including two pairs of (Button,Textfield) and one pair of (Label,Combobox), the statistician wants to create looks like:



Note that the types of the arguments for "Data 1" and "Data 2" are double arrays and the one for "Operator" is String. In JavaStatSoft, the statistician only needs to focus on how to obtain the results based on the arguments specified by the user, not for creating the GUI. The statistician can add a menu for the module to JavaStatSoft by the following steps.

1. Selects "File-->Add (Remove) User's Menu Items" and enters the

texts for the menu and menu item and the name of the only class "MyDataOperation" the statistician needs to construct.



Add (Remove) A Menu

Add (Remove) A Menu

Menu Items | Child Items

Menu

Menu Text: Data Operation

Remove

Menu Item

Item Text: Simple Data Operation

Invoked Class: MyDataOperation

Menu Item

Item Text:

Invoked Class:

Menu Item

Item Text:

Invoked Class:

Menu Item

Item Text:

Invoked Class:

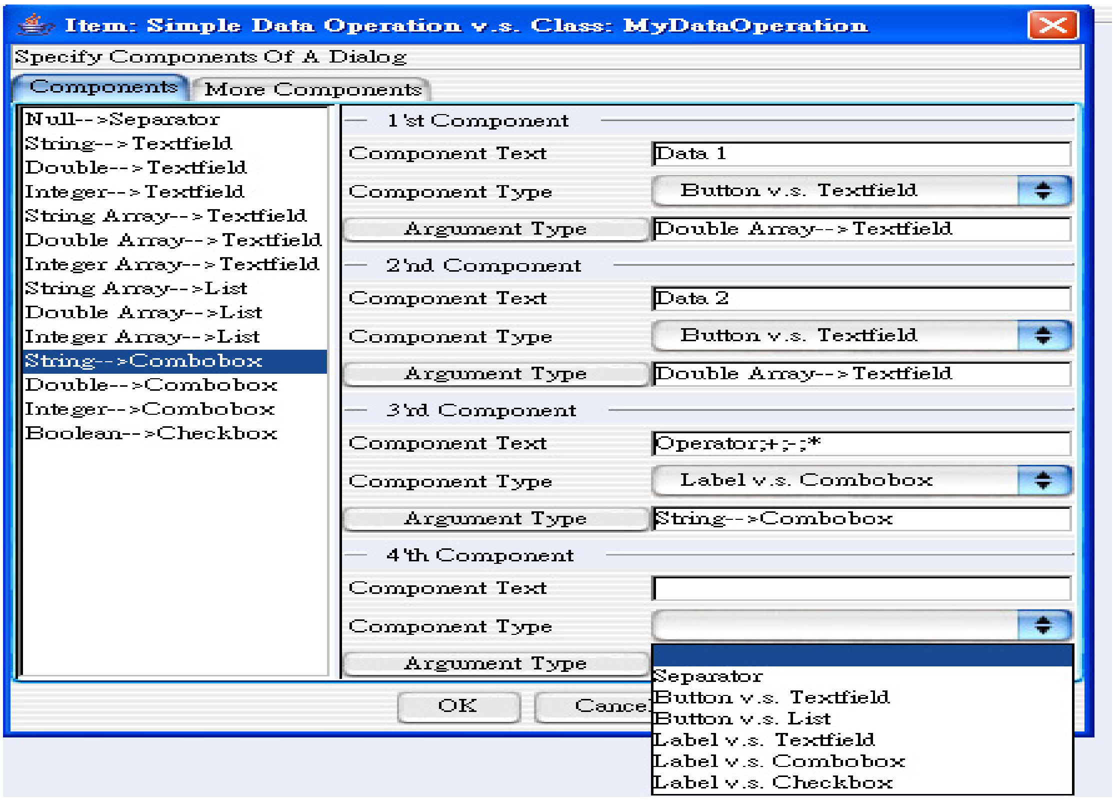
Multiple Menu Items

Texts in Items: ItemA;ItemB;ItemC

Invoked Classes: ClassA;ClassB;ClassC

OK | Cancel

2. A dialog for specifying the components of the dialog invoked as selecting the item "Simple Data Operation" will be brought up automatically. To create the dialog for the item "Simple Data Operation", the statistician needs to specify the components and associated texts.



Item: Simple Data Operation v.s. Class: MyDataOperation

Specify Components Of A Dialog

Components | More Components

Null-->Separator
String-->Textfield
Double-->Textfield
Integer-->Textfield
String Array-->Textfield
Double Array-->Textfield
Integer Array-->Textfield
String Array-->List
Double Array-->List
Integer Array-->List
String-->Combobox
Double-->Combobox
Integer-->Combobox
Boolean-->Checkbox

1'st Component

Component Text: Data 1

Component Type: Button v.s. Textfield

Argument Type: Double Array-->Textfield

2'nd Component

Component Text: Data 2

Component Type: Button v.s. Textfield

Argument Type: Double Array-->Textfield

3'rd Component

Component Text: Operator;+;-;*

Component Type: Label v.s. Combobox

Argument Type: String-->Combobox

4'th Component

Component Text:

Component Type:

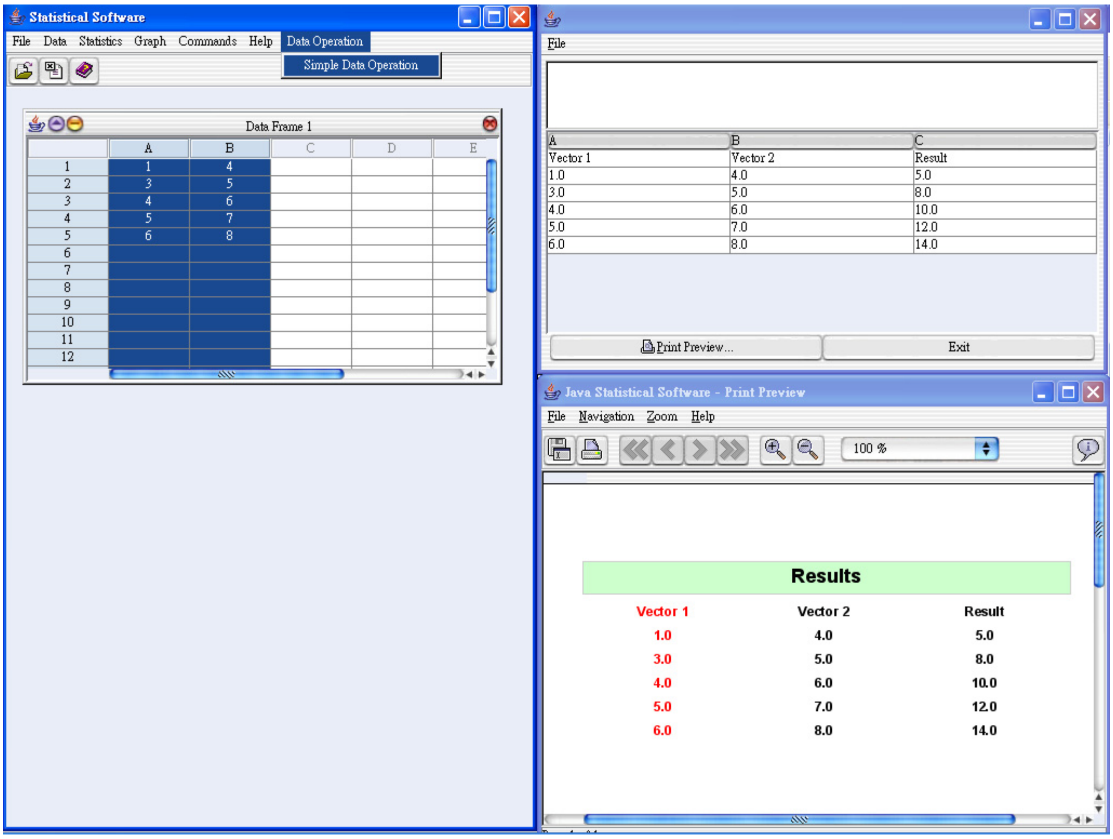
Argument Type:

Note: as the component is the pair of **(Label,Combobox)**, the text entered in the textfield of **"Component Text"** has the form

label text;item text; item text; item text

3. The menu and item will be added to the menu bar of JavaStatSoft. The last thing the statistician needs to do is to create the class **"MyDataOperation"**. The source code can be found in the directory **"examples\pluggable"or clicking here.**

   The above code mainly performs arithmetic operations of the input vectors which can be obtained from the arguments specified by the user. The following is the result as the user inputs two vectors and obtains the sum of the two vectors.



	A	B	C	D	E
1	1	4			
2	3	5			
3	4	6			
4	5	7			
5	6	8			
6					
7					
8					
9					
10					
11					
12					

A	B	C
Vector 1	Vector 2	Result
1.0	4.0	5.0
3.0	5.0	8.0
4.0	6.0	10.0
5.0	7.0	12.0
6.0	8.0	14.0

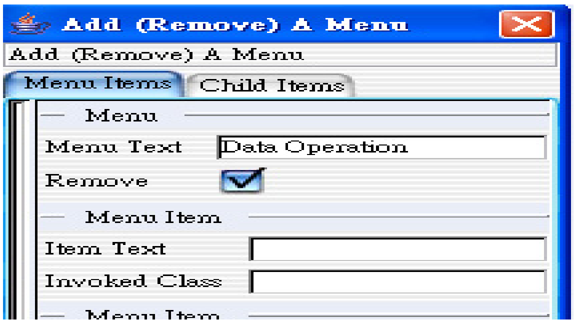
Results

Vector 1	Vector 2	Result
1.0	4.0	5.0
3.0	5.0	8.0
4.0	6.0	10.0
5.0	7.0	12.0
6.0	8.0	14.0

Note: the statistician can compile the source code by **StatCompiler** in JavaStatSoft. Alternative, the user can use any Java IDE (Integrated Development Environment) to compile the code, then put the ***.class** file in the directory **"users"**.

Removes a menu:

The statistician can remove the added menu by selecting “File-->Add (Remove) User’s Menu Items”, entering the texts of the menu to be removed and checking with the “Remove” checkbox.



Menu Text: Data Operation

Remove: ☑

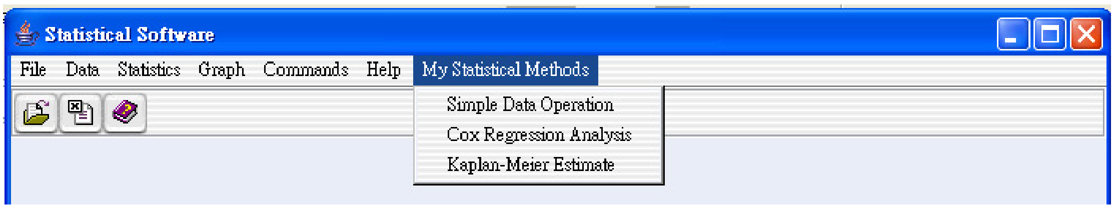
Item Text:

Invoked Class:

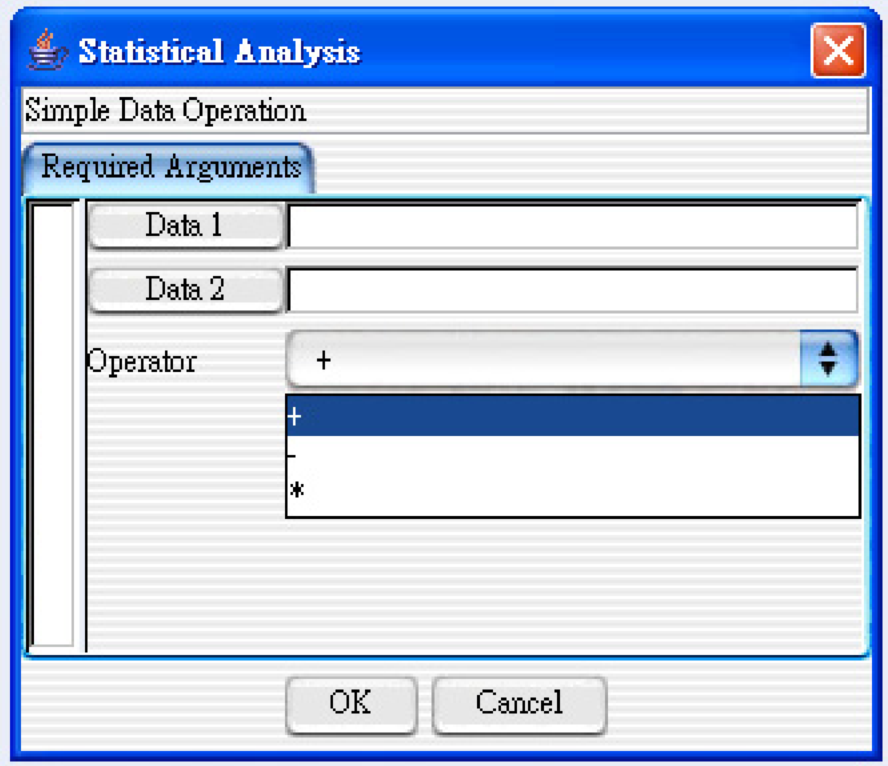
I. Customized GUI: More Complicated Examples

Adds a menu with several menu items:

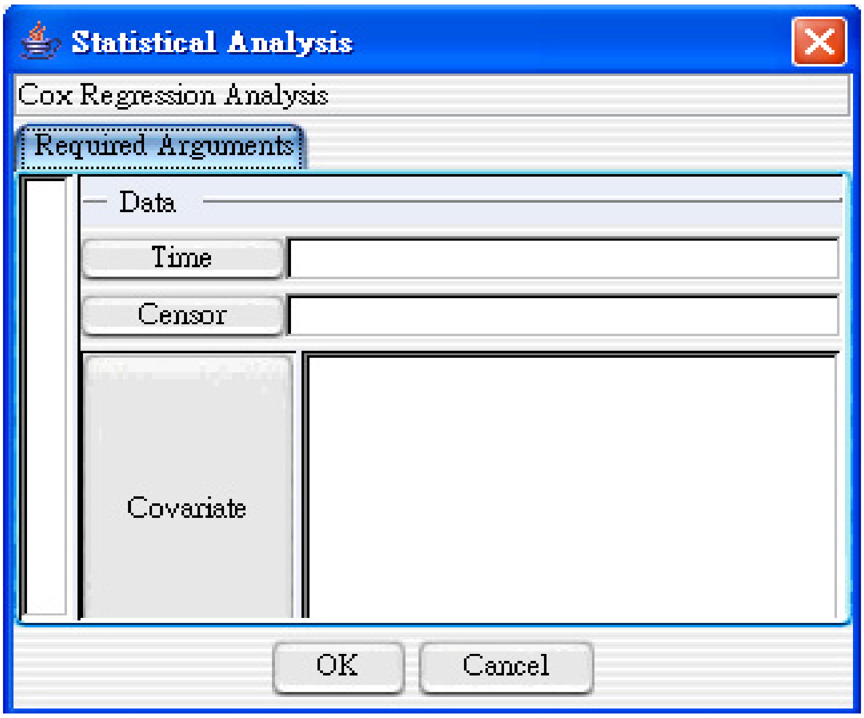
Suppose a statistician wants to add a menu with 3 menu items for 3 modules, one for simple arithmetic operations of two input vectors and the other two for fitting a Cox proportional hazards regression model and for calculating the Kaplan-Meier estimates of survival function, respectively. The menu bar of JavaStatSoft then looks like



The dialog, including two pairs of (Button,Textfield) and one pair of (Label,Combobox),   for the simple data operation looks like

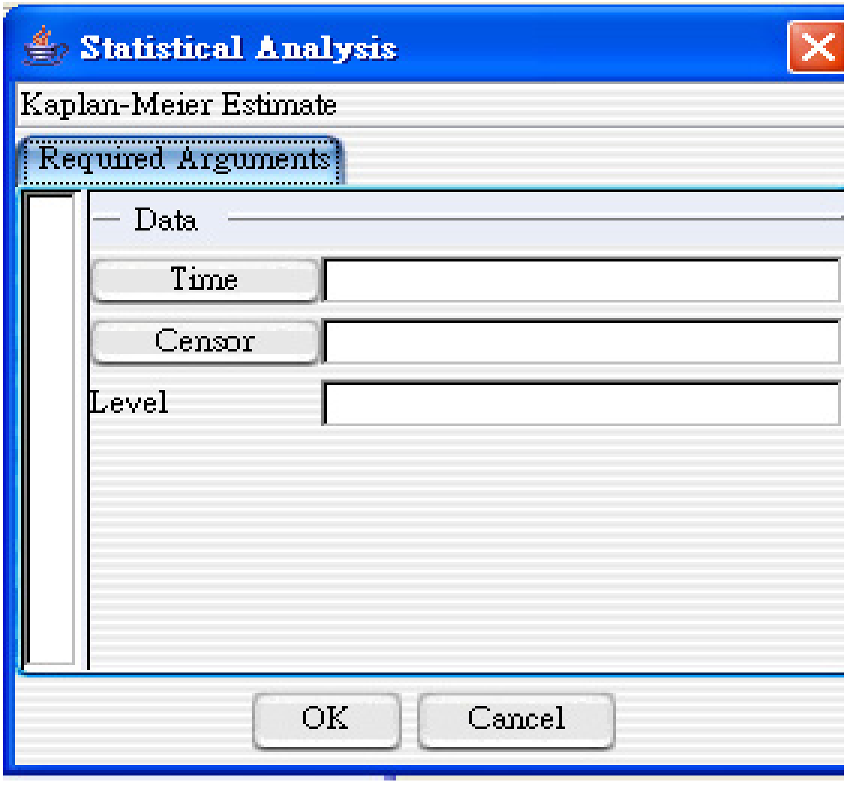


The dialog, including one separator, two pairs of (Button,Textfield) and one pair of (Button,List), for fitting a Cox proportional hazards regression model looks likes:



The types of the arguments for “Time”, “Censor” and “Covariate” are all double arrays.

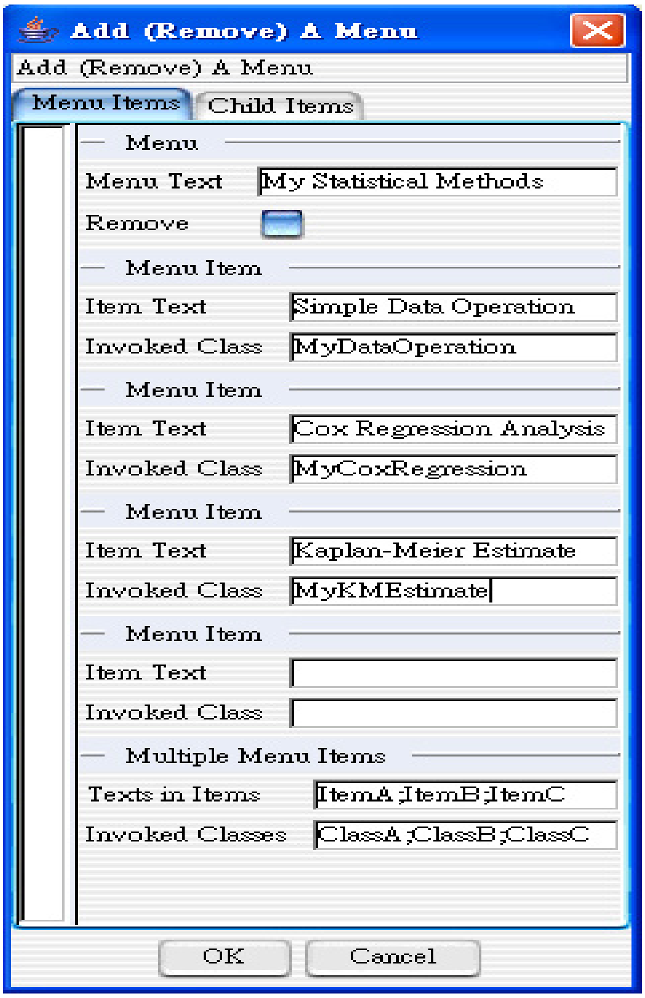
The dialog, including one separator, two pairs of (Button,Textfield) and one pair of (Label,Textfield), for calculating the Kaplan-Meier estimates of survival function looks like



The types of the arguments for “Time” and “Censor” are double arrays and the one for “Level” is double.

The statistician can add a menu for these modules to JavaStatSoft by the following steps.

1. Selects “File-->Add (Remove) User’s Menu Items” and enters the texts for the menu and menu items and the names of the classes “MyDataOperation”, “MyCoxRegression” and “MyKMEstiamte” the statistician needs to construct.



Menu

Menu Text: My Statistical Methods

Remove

Menu Item

Item Text: Simple Data Operation

Invoked Class: MyDataOperation

Menu Item

Item Text: Cox Regression Analysis

Invoked Class: MyCoxRegression

Menu Item

Item Text: Kaplan-Meier Estimate

Invoked Class: MyKMEstimate

Menu Item

Item Text:

Invoked Class:

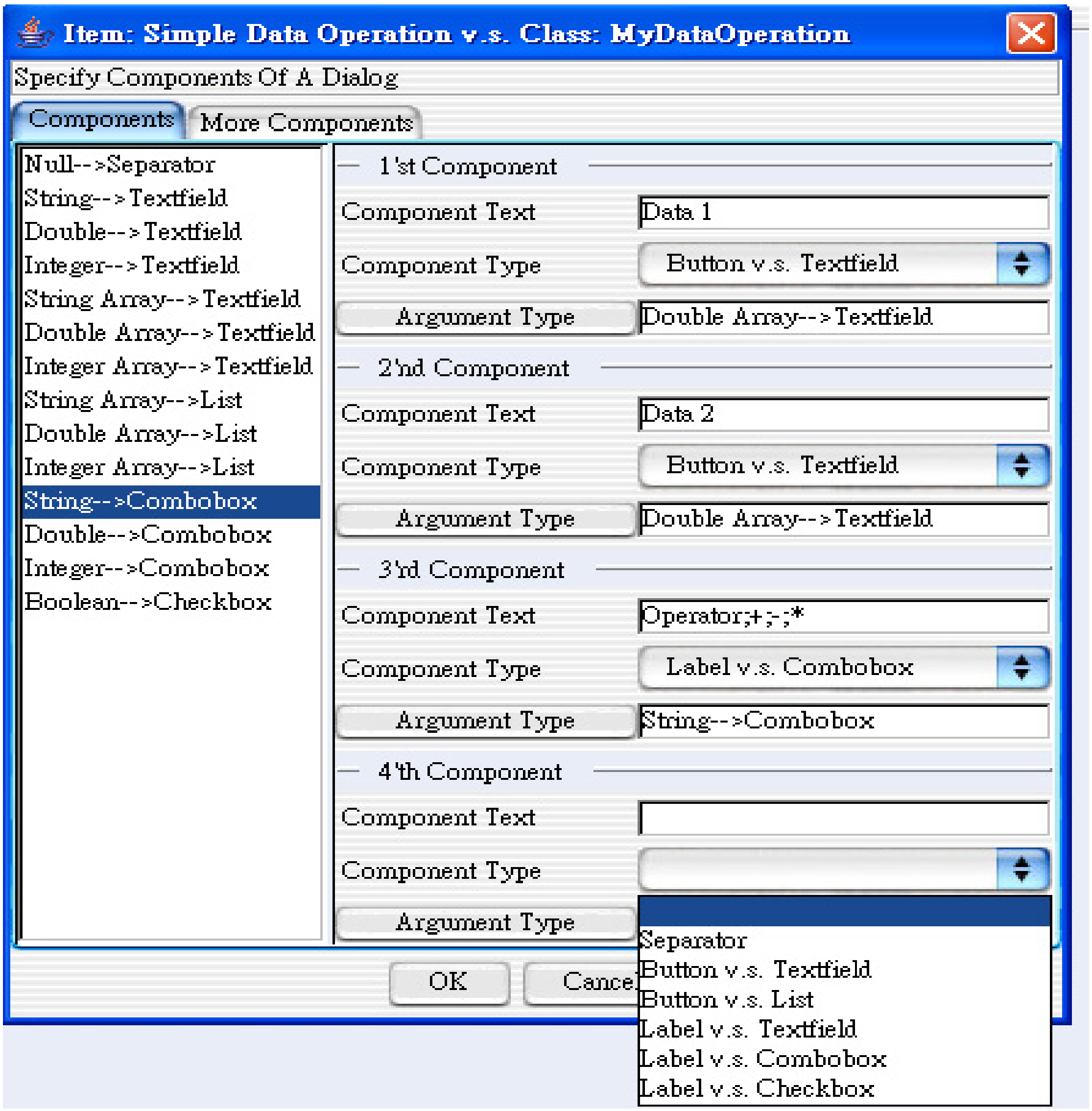
Multiple Menu Items

Texts in Items: ItemA;ItemB;ItemC

Invoked Classes: ClassA;ClassB;ClassC

2. **A dialog for specifying the components of the dialogs invoked as selecting the added items will be brought up automatically. To create the dialogs for these items, the statistician needs to specify the components and associated texts.**

For “Simple Data Operation”,



Item: Simple Data Operation v.s. Class: MyDataOperation

Specify Components Of A Dialog

1'st Component

Component Text: Data 1

Component Type: Button v.s. Textfield

Argument Type: Double Array-->Textfield

2'nd Component

Component Text: Data 2

Component Type: Button v.s. Textfield

Argument Type: Double Array-->Textfield

3'rd Component

Component Text: Operator;+;-;*

Component Type: Label v.s. Combobox

Argument Type: String-->Combobox

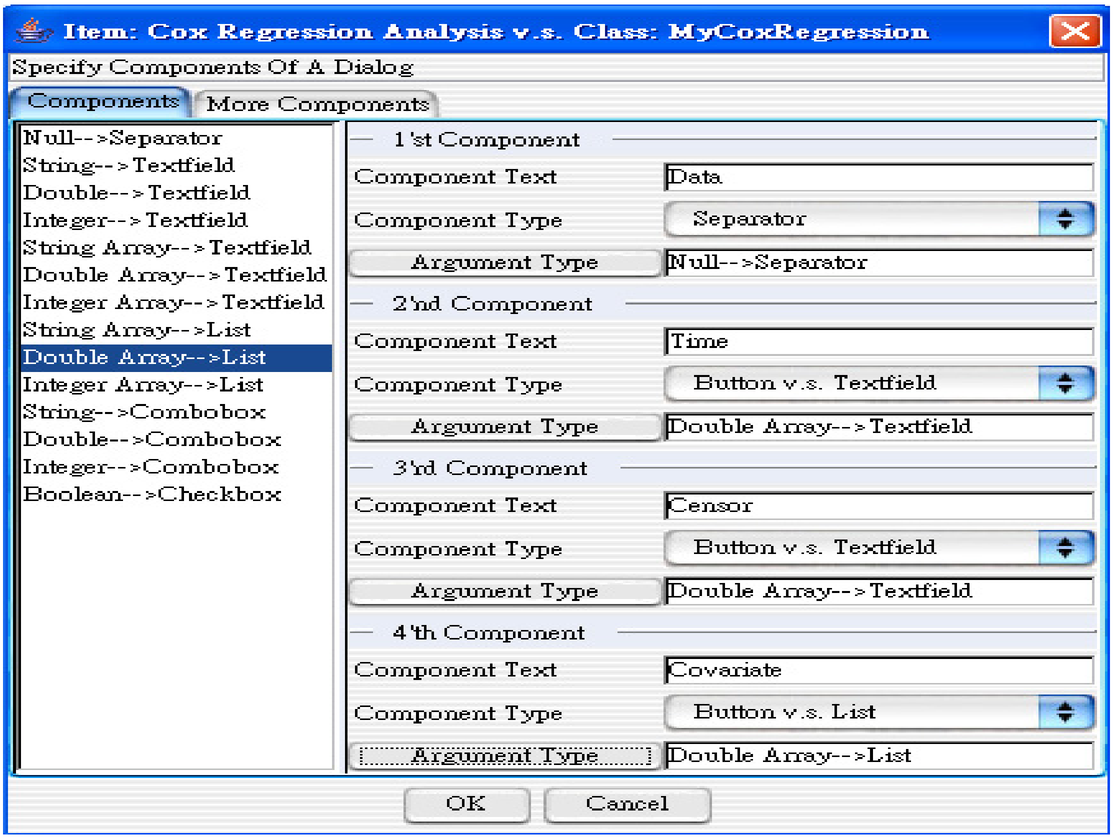
4'th Component

Component Text:

Component Type:

Argument Type

For “Cox Regression Analysis”,



1'st Component

Component Text	Data
Component Type	Separator
Argument Type	Null-->Separator

2'nd Component

Component Text	Time
Component Type	Button v.s. Textfield
Argument Type	Double Array-->Textfield

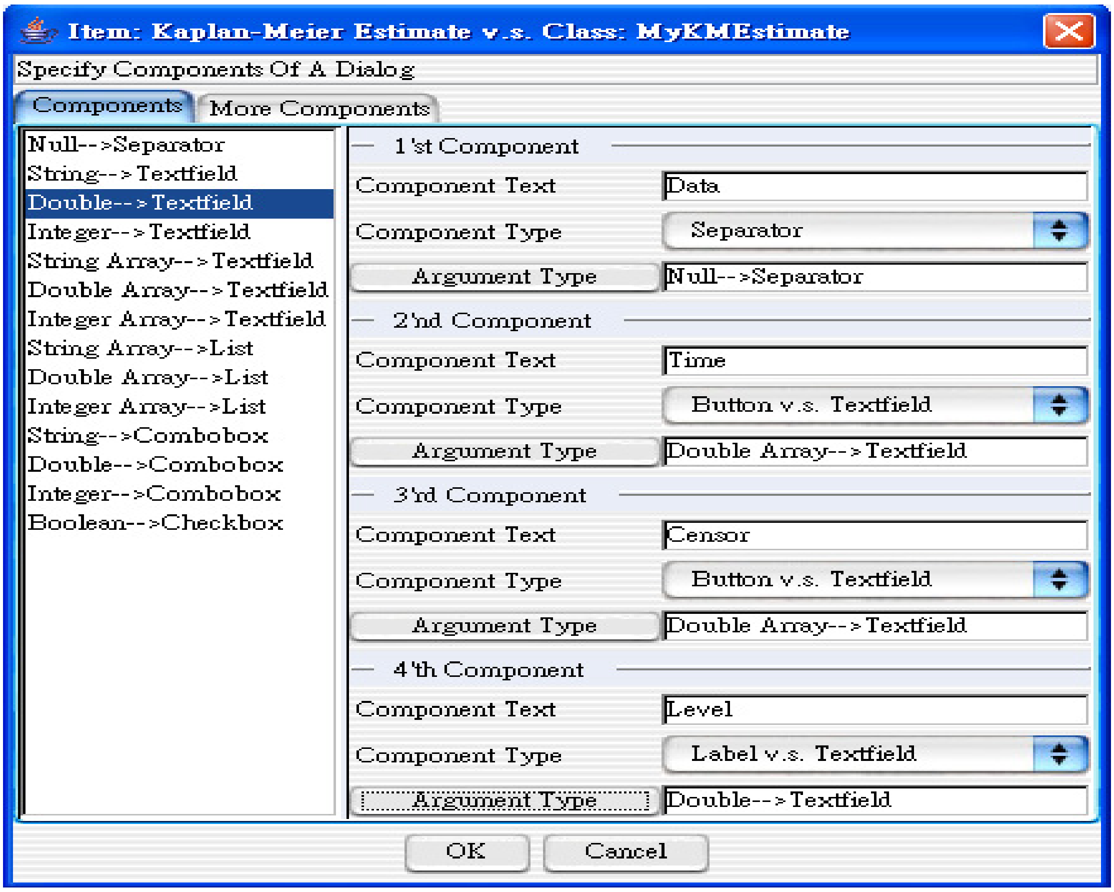
3'rd Component

Component Text	Censor
Component Type	Button v.s. Textfield
Argument Type	Double Array-->Textfield

4'th Component

Component Text	Covariate
Component Type	Button v.s. List
Argument Type	Double Array-->List

For “Kaplan-Meier Estimate”,



1'st Component

Component Text	Data
Component Type	Separator
Argument Type	Null-->Separator

2'nd Component

Component Text	Time
Component Type	Button v.s. Textfield
Argument Type	Double Array-->Textfield

3'rd Component

Component Text	Censor
Component Type	Button v.s. Textfield
Argument Type	Double Array-->Textfield

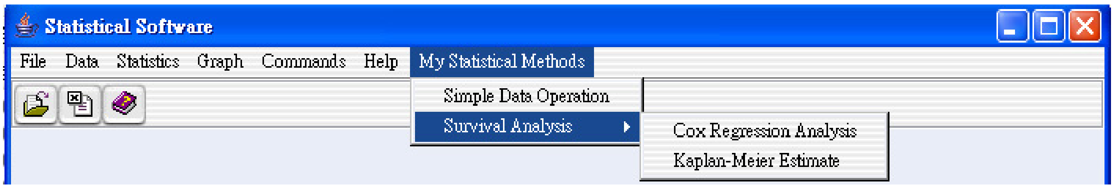
4'th Component

Component Text	Level
Component Type	Label v.s. Textfield
Argument Type	Double-->Textfield

3. The menu and items will be added to the menu bar of JavaStatSoft. The last thing the statistician needs to do is to create the classes “MyDataOperation”, “MyCoxRegression”, and "MyKMEstimate”. The source code can be found in the directory “examples\pluggable” or clicking (MyDataOperation,MyCoxRegression,MyKMEstimate).

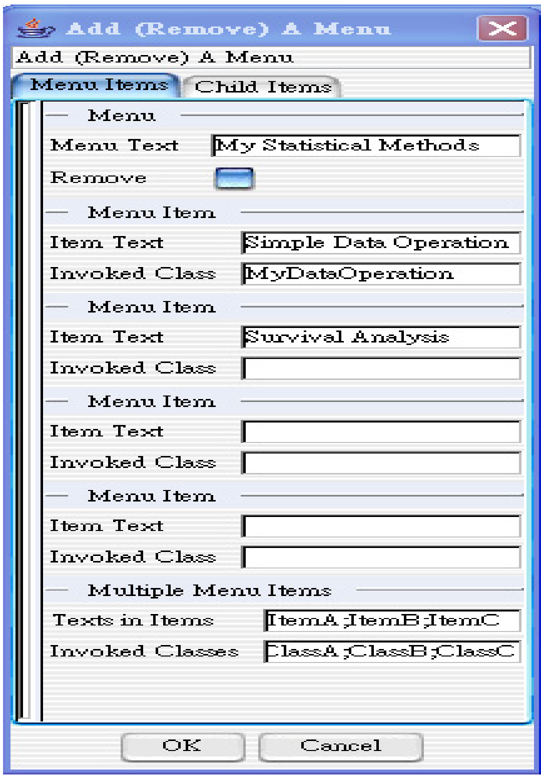
Adds a menu with a sub-menu and menu items:

Suppose a statistician wants to add a menu with 1 sub-menu and 3 menu items for the above modules. The menu bar of JavaStatSoft then looks like



The statistician can add these components by the following.

1. Selects “File-->Add (Remove) User’s Menu Items” and enters the texts for the menu and menu items and the names of the classes “MyDataOperation”, “MyCoxRegression” and “MyKMEstiamte” the statistician needs to construct.



Add (Remove) A Menu

Add (Remove) A Menu

Menu Items | Child Items

Menu

Menu Text: My Statistical Methods

Remove

Menu Item

Item Text: Simple Data Operation

Invoked Class: MyDataOperation

Menu Item

Item Text: Survival Analysis

Invoked Class:

Menu Item

Item Text:

Invoked Class:

Menu Item

Item Text:

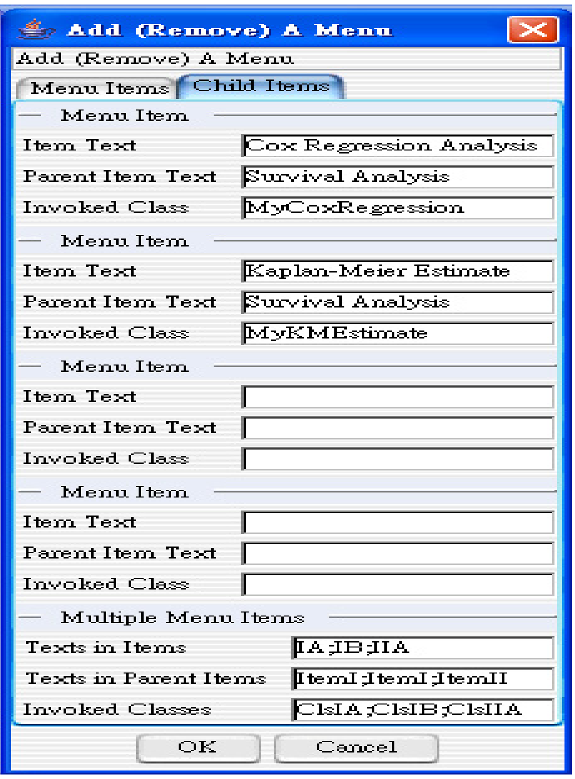
Invoked Class:

Multiple Menu Items

Texts in Items: ItemA;ItemB;ItemC

Invoked Classes: ClassA;ClassB;ClassC

OK | Cancel



Add (Remove) A Menu

Add (Remove) A Menu

Menu Items | Child Items

Menu Item

Item Text: Cox Regression Analysis

Parent Item Text: Survival Analysis

Invoked Class: MyCoxRegression

Menu Item

Item Text: Kaplan-Meier Estimate

Parent Item Text: Survival Analysis

Invoked Class: MyKMEstimate

Menu Item

Item Text:

Parent Item Text:

Invoked Class:

Menu Item

Item Text:

Parent Item Text:

Invoked Class:

Multiple Menu Items

Texts in Items: IA;IB;IIA

Texts in Parent Items: ItemI;ItemI;ItemII

Invoked Classes: ClsIA;ClsIB;ClsIIA

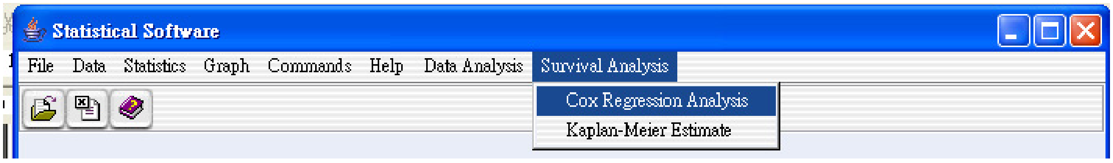
OK | Cancel

The blank textfield for "Invoked Class" associated with item text "Survival Analysis" indicates the component is a sub-menu. Further, in the "Child Items" tab, two menu items "Cox Regression Analysis" and "Kaplan-Meier Estimate" of the sub-menu "Survival Analysis" are specified as well as the associated invoked classes, "MyCoxRegression" and "MyKMEstimate".

The other two steps are similar to the above example.

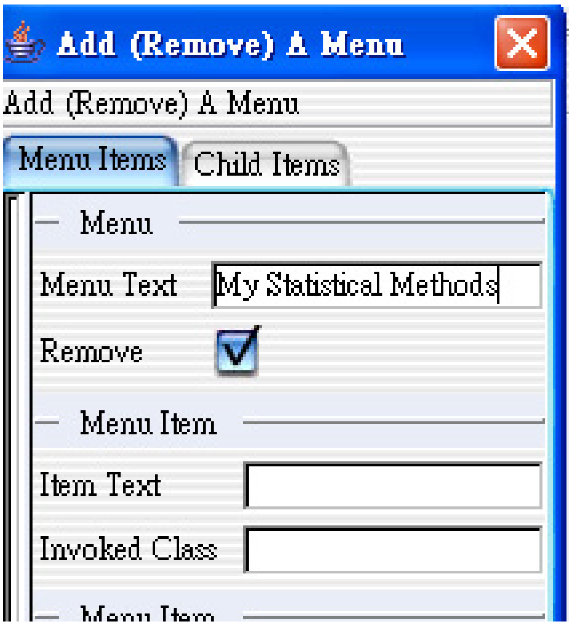
Note: adding multiple sub-menus with multiple items can be done similarly.

Note: multiple menus can be added one by one. The menu bar of JavaStatSoft with two added menus looks like



Removes a menu:

The statistician can remove the added menu by selecting "File-->Add (Remove) User's Menu Items", entering the texts of the menu to be removed and checking with the "Remove" checkbox.



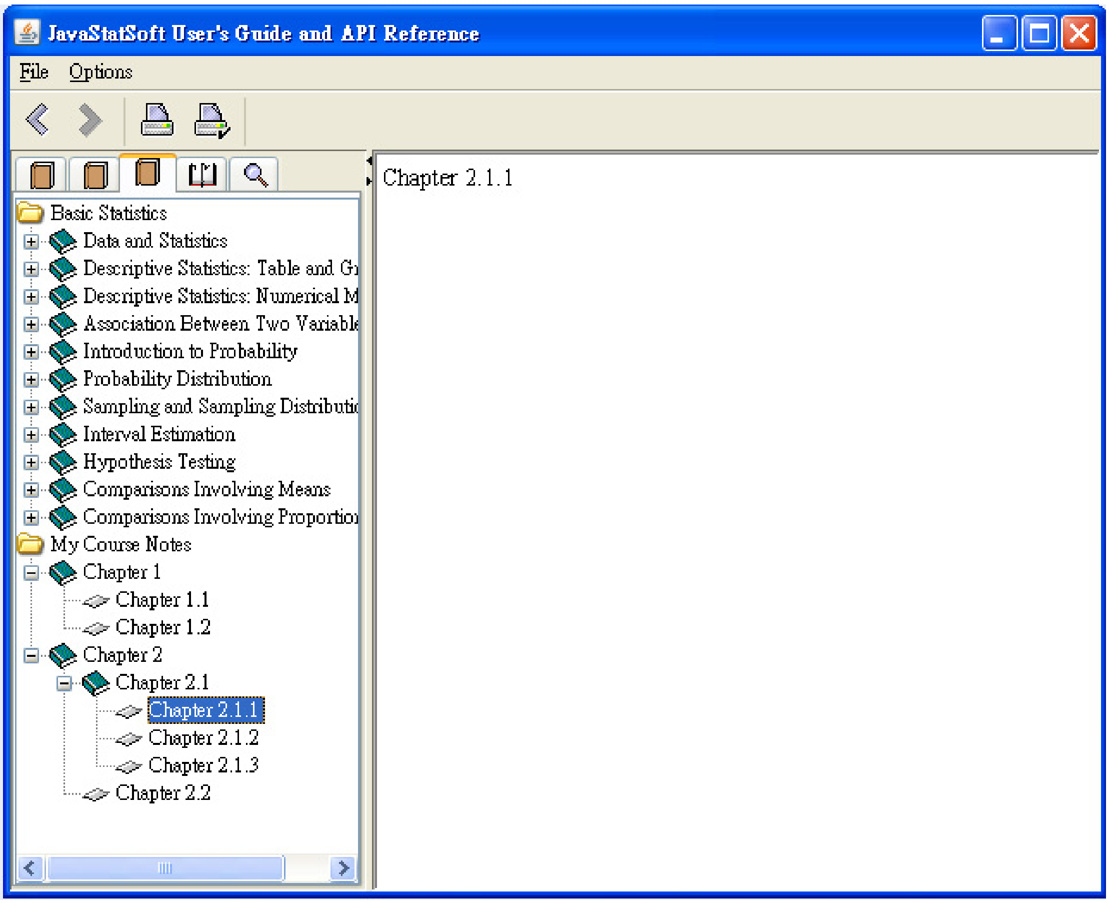
Menu Text: My Statistical Methods

Remove ☑

J. Customized Help System

Adds course notes:

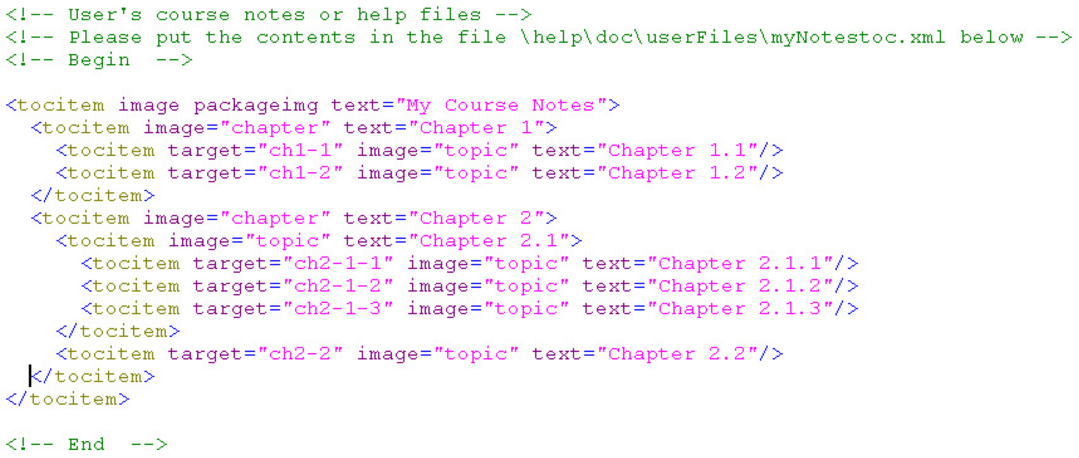
Suppose a statistician wants to add his (or her) course notes to the help system. The help system then looks like


JavaStatSoft User's Guide and API Reference
File
Options
Basic Statistics
Data and Statistics
Descriptive Statistics: Table and Gr
Descriptive Statistics: Numerical M
Association Between Two Variabl
Introduction to Probability
Probability Distribution
Sampling and Sampling Distributio
Interval Estimation
Hypothesis Testing
Comparisons Involving Means
Comparisons Involving Proportio
My Course Notes
Chapter 1
Chapter 1.1
Chapter 1.2
Chapter 2
Chapter 2.1
Chapter 2.1.1
Chapter 2.1.2
Chapter 2.1.3
Chapter 2.2
Chapter 2.1.1

The steps to add the course notes to the help system are as follows .

1. Puts all the *.html files for the course in the directory “\help\doc\userFiles”.
2. Modifies the XML file, “statisticsGuidetoc.xml” in the directory “\help\doc”, as shown below. The statistician can use the file “myNotestoc.xml” in the directory “\help\doc\userFiles” as a template file. The contents of the file “myNotestoc.xml” can be copied to the file “statisticsGuidetoc.xml”. Basically, the statistician mainly specifies the texts to be displayed in the help system and the id associated with URL. For example, the text is "Chapter 1.1" and the id associated with the HTML file for this course note is “ch1-1”, as specified by

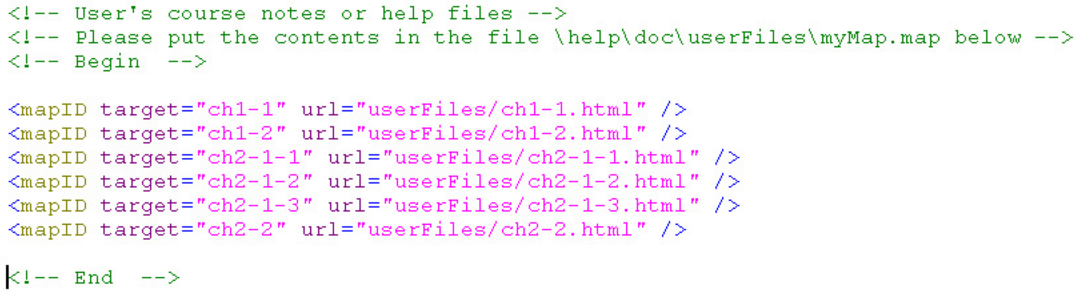
<tocitem target="ch1-1" image="topic" text="Chapter 1.1"/>



```
<!-- User's course notes or help files -->
<!-- Please put the contents in the file \help\doc\userFiles\myNotestoc.xml below -->
<!-- Begin  -->

<tocitem image packageimg text="My Course Notes">
  <tocitem image="chapter" text="Chapter 1">
    <tocitem target="ch1-1" image="topic" text="Chapter 1.1"/>
    <tocitem target="ch1-2" image="topic" text="Chapter 1.2"/>
  </tocitem>
  <tocitem image="chapter" text="Chapter 2">
    <tocitem image="topic" text="Chapter 2.1">
      <tocitem target="ch2-1-1" image="topic" text="Chapter 2.1.1"/>
      <tocitem target="ch2-1-2" image="topic" text="Chapter 2.1.2"/>
      <tocitem target="ch2-1-3" image="topic" text="Chapter 2.1.3"/>
    </tocitem>
    <tocitem target="ch2-2" image="topic" text="Chapter 2.2"/>
  </tocitem>
</tocitem>

<!-- End  -->
```

3. **Modifies the XML file, "Map.map" in the directory "\help\doc", as shown below. The statistician can use the file "myMap.map" in the directory "\help\doc\userFiles" as a template file. The contents of the file "myMap.map" can be copied to the file "Map.map". The statistician mainly specifies the id associated with URL in the map file. For example, the HTML file "ch1-1.html" in the directory "\help\doc\userFiles" is associated with id "ch1-1", as specified by**

   <mapID target="ch1-1" url="userFiles/ch1-1.html" />



```
<!-- User's course notes or help files -->
<!-- Please put the contents in the file \help\doc\userFiles\myMap.map below -->
<!-- Begin  -->

<mapID target="ch1-1" url="userFiles/ch1-1.html" />
<mapID target="ch1-2" url="userFiles/ch1-2.html" />
<mapID target="ch2-1-1" url="userFiles/ch2-1-1.html" />
<mapID target="ch2-1-2" url="userFiles/ch2-1-2.html" />
<mapID target="ch2-1-3" url="userFiles/ch2-1-3.html" />
<mapID target="ch2-2" url="userFiles/ch2-2.html" />

<!-- End  -->
```

```java
import java.util.Vector;

import statsoft.user.PluggableDataAnalysis;

public class MyDataOperation extends PluggableDataAnalysis
{
    public String [][] createReportData(Vector arguments)
    {
        double[] vectorOne = (double[]) arguments.get(0);
        double[] vectorTwo = (double[]) arguments.get(1);
        String operator = (String) arguments.get(2);
        int size=Math.max(vectorOne.length,vectorTwo.length);
        String[][] reportData=new String[size+1][];
        reportData[0]=new String[]{"Vector 1","Vector 2","Result"};
        if(operator.equalsIgnoreCase("+"))
            for(int i=1; i <= size; i++)
                reportData[i]=new String[]{
                              Double.toString(vectorOne[i-1]),
                              Double.toString(vectorTwo[i-1]),
                              Double.toString(vectorOne[i-1]+vectorTwo[i-1])};
        else if(operator.equalsIgnoreCase("-"))
            for(int i=1; i <= size; i++)
                reportData[i]=new String[]{
                              Double.toString(vectorOne[i-1]),
                              Double.toString(vectorTwo[i-1]),
                              Double.toString(vectorOne[i-1]-vectorTwo[i-1])};
        else
            for(int i=1; i <= size; i++)
                reportData[i]=new String[]{
                              Double.toString(vectorOne[i-1]),
                              Double.toString(vectorTwo[i-1]),
                              Double.toString(vectorOne[i-1]*vectorTwo[i-1])};

        return reportData;
    }

}
```

```
import java.util.Vector;


import statsoft.user.PluggableDataAnalysis;


import javastat.util.DataManager;
import javastat.survival.regression.CoxRegression;


public class MyCoxRegression extends PluggableDataAnalysis
{


    public String [][] createReportData(Vector arguments)
    {
        double[] time = (double[]) arguments.get(0);
        double[] censor = (double[]) arguments.get(1);
        DataManager dataManager=new DataManager();
        double [][] covariate=(double[][]) arguments.get(2);
        CoxRegression coxRegression=new CoxRegression(time,censor,covariate);
        String [][] reportData= new String[coxRegression.coefficients.length + 1][5];
        reportData[0] = new String[] {"Coefficients", "Value",
                                "Std. Error", "Z", "P-value"};
        for (int j = 0; j < coxRegression.coefficients.length; j++)
        {
            reportData[j + 1][0] = "b" + (j + 1);
            reportData[j + 1][1] = Double.toString(
                    dataManager.roundDigits(coxRegression.coefficients[j], 3.0));
            reportData[j + 1][2] = Double.toString(dataManager.roundDigits(
                    Math.pow(coxRegression.variance[j][j], 0.5), 3.0));
            reportData[j + 1][3] = Double.toString(
                    dataManager.roundDigits(coxRegression.testStatistic[j], 3.0));
            reportData[j + 1][4] = Double.toString(
                    dataManager.roundDigits(coxRegression.pValue[j], 3.0));
        }


        return reportData;
    }


}
```

```
import java.util.Vector;


import statsoft.user.PluggableDataAnalysis;


import javastat.util.DataManager;
import javastat.survival.KaplanMeierEstimate;


public class MyKMEstimate extends PluggableDataAnalysis
{
    public String [][] createReportData(Vector arguments)
    {
        double[] time = (double[]) arguments.get(0);
        double[] censor = (double[]) arguments.get(1);
        DataManager dataManager=new DataManager();
        double level = ((Double) arguments.get(2)).doubleValue();
        KaplanMeierEstimate kaplanMeierEstimate=new KaplanMeierEstimate(level,time,censor);
        int[] orderIndex = dataManager.orderIndex(kaplanMeierEstimate.time);
        String[][] reportData= new String[kaplanMeierEstimate.time.length + 1][4];
        reportData[0] = new String[] {"Time", "Estimate","Std. Error", "Interval"};
        for (int j = 0; j < kaplanMeierEstimate.time.length; j++)
        {
            reportData[j + 1][0] = Double.toString(
                        dataManager.roundDigits(kaplanMeierEstimate.time[orderIndex[j]], 3.0));
            reportData[j + 1][1] = Double.toString(
                        dataManager.roundDigits(kaplanMeierEstimate.estimate[orderIndex[j]],3.0));
            reportData[j + 1][2] = Double.toString(
            dataManager.roundDigits(Math.pow(kaplanMeierEstimate.variance[orderIndex[j]], 0.5),
                                    3.0));
            reportData[j + 1][3] = "[" + Double.toString(
                      dataManager.roundDigits(
                      kaplanMeierEstimate.confidenceInterval[orderIndex[j]][0], 3.0)) + "," +
                      Double.toString(dataManager.roundDigits(
                      kaplanMeierEstimate.confidenceInterval[orderIndex[j]][1], 3.0)) +"]";
        }


      return reportData;
    }
}
```