

Evince Script

User Guide

Version 1.04

Copyright UmBio AB, 2008-03-01

UmBio AB
Box 7980
907 19, Umeå
Sweden

+46 (0)90 15 48 20

Contents

Description	3
Symboling	3
Getting started	4
Available functions	5
General	5
load	5
show/hide	5
createPlot	6
createTable	6
loadModel	6
getConstant	6
Matrix	8
save (not finished)	8
getValue	8
Dataset	9
include	9
exclude	9
setTrain	9
setTest	9
setX	10
setY	10
createModification	10
createModel	10
cloneDataSet	11
get	11
setName	11
Model	12
createPrediction	12
get	12
save	12
setName	12
getComponents	12
Plot	14
addLayer	14
setTitle	14
save	14
addTextArea	14
Plot Layer settings	15
setDataRange	15
setColor	15
setShape	16
setSize	16
setLabel	17
setLine	17
setComment	17
setLegend	18
Plot Window	19
setBackgroundColor	19

setBorder 19

Description

The Evince scripting tool is used for creating customized work flows with an easy-to-use script. The available scripting functions and how to use them is described in the coming sections.

Symboling

Each function must end with the “;” character
dataset = load(“data file”);

Comments are made in the script as followed:
/* comment */

A range is set as followed:
[5,9] /* denotes the range 5 to 9 */

is a separator between parameters
0,1,2,3,4,5,10,14,#,2,3,7

@ is a symbol for adding text strings
“start” @ “end” -> “startend”

\n within quotation mark means end if line
“line1\nline2”

Getting started

A script can either be executed directly from Evince when importing a new project or from a web homepage using script and Evince web start.

The easiest way to create a new script is to export the script as a template from Evince history panel. The script is based on the actions that the user has been made in Evince and can directly be used to repeat the actions on a new data file.

Evince import template

Evince has a default template directory under .Evince in the user home folder. The directory is called Script and will contain all user defined templates. These templates are loaded in the last step of the import wizard and can be chosen from the drop down component.

Evince web start

Adapted script can be transferred to Evince using Evince web start. The script text is defined as an argument in the jnlp file called script. Example:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- JNLP File for Evince Application -->
<jnlp spec="1.0" codebase="http://" href="evince_script.jnlp">
  <information>
    <title>Evince Script, Web Start</title>
    <vendor>UmBio AB</vendor>
    <description>Evince Web Start</description>
  </information>
  <resources>
    <j2se version="1.5+" href="http://java.sun.com/products/autodl/j2se" initial-heap-size="64M" max-heap-size="1024M" />
    <jar href="Evince.jar"/>
    <extension name="libraries.jnlp" />
    <property name="webstart" value="true"/>
    <property name="script" value="[SCRIPT TEXT]"/>
  </resources>
  <application-desc main-class="map.GUI"/>
</jnlp>
```

See “Deploying Software with JNLP and Java Web Start” for more information.

Available functions

General

load

Description: Loads a file holding data that will be predicted. The prediction dataset is specified in the index page.

Syntax: *load(filename)*
load(filename, format, delimiter)
load(filename, obs-description index, var-description index, format, delimiter)
load(filename, obs-description index, var-description index)

Format parameters:

- txt
- csv
- excel
- mat
- envi

Delimiter parameters: (only for txt and csv)

- tab
- space
- comma
- line
- colon
- semicomma

Example: predictiondataset = load("filename",[1,5], 7,#,1, 4, "txt", "comma"); / loads prediction file and sets columns 1 to 5 to observation identifiers and row 1 to variable identifier */*

show/hide

Description: Shows or hides an Evince workspace area.

Syntax *show(workspace1, ... , workspace3)*
hide(workspace1, ... , workspace3)

Workspace parameters:

- tablearea
- plotarea
- modelarea
- historyarea
- datatree
- settings

Example: show("plotarea", "tablearea");
hide("datatree", "settings");

createPlot

Description: Creates a plot.

Syntax: `createPlot(plot type, data)`

Plot type parameters:

- scatter2d
- scatter2ddensity
- scatter3d
- scatter3ddensity
- histogram
- dendrogram
- column
- line
- contour2d
- contour3d
- piechart
- score2d
- score3d
- loading2d
- loading3d

Example: `Score_plot = createPlot("score2d, calibrationmodel1);`
`Score_plot2 = createplot("scatter2d", T);`

createTable

Description: Creates a table.

Syntax: `createTable(table type, data)`

Table type parameters:

- viewdataset
- datatable

Example: `table1 = createTable("datatable", dataset);`

loadModel

Description: Loads a calibration model. Calibration models 1 to n are specified in the index page.

Syntax: `loadModel(calibration model)`

Example: `model1 = loadModel($calibrationmodel1);`
`model2 = loadModel($calibrationmodel2);`

getConstant

Description: Get a constant from Evince.

Syntax: `getConstant(constant type)`

Constant parameters:

- `filename` (Get project file name)
- `version` (Get Evince version)

Matrix

save (not finished)

Description: Save the current matrix to disc

Syntax: `save(filepath, format)`

Format parameters:

- `csv`

Example: `matrix.save("C:\matrix.txt", "csv"); /* Saves matrix to the file matrix.txt */`

getValue

Description: Get a specified value from the given matrix. The value is represented as a string value. If no position is specified the last value will be return

Syntax: `getValue(x pos, y pos)`
 `getValue()`

Example: `getValue(5, 10) /* Returns the value on column 5 and row 10 */`

Dataset

include

Description: Includes observations or variables.

Syntax: *include(observations / variables, index)*
include(observations / variables, class name, sub class name)
include(observations / variables, class name, sub class index)

Example: *dataset.include("obs",3, [5,8]); /* includes observation 3 and observation 5 to 8 */*
dataset.include("var",3); / includes variable 3 */*

exclude

Description: Excludes observations or variables.

Syntax: *exclude(observations / variables, index)*
exclude(observations / variables, class name, sub class name)
exclude(observations / variables, class name, sub class index)

Example: *dataset.exclude("obs", 1,[2,4]); /* excludes observation 1 and observation 2 to 4 */*
dataset.exclude("var", 1,[2,4]); / excludes variable 1 and variable 2 to 4 */*

setTrain

Description: Sets observations to or removes observations from the training set.

Syntax: *setTrain(true / false, index)*
setTrain(true / false) - All indices will be set to either true or false
setTrain(true / false, class name, sub class name)
setTrain(true / false, class name, sub class index)

Example: *dataset.setTrain(true,[1,5]); /* Sets observations 1 to 5 to the training set */*
dataset.setTrain(false,[1,5]); / Removes observations 1 to 5 from the training set */*

setTest

Description: Sets observations to or removes observations from the test set.

Syntax: *setTest(true / false, index)*
setTest(true / false) - All indices will be set to either true or false
setTest(true / false, class name, sub class name)
setTest(true / false, class name, sub class index)

Example: *dataset.setTest(true,[1,5]); /* Sets observations 1 to 5 to a test set */*
dataset.setTest(false,[1,5]); / Removes observations 1 to 5 from the test set */*

setX

Description: Sets variables to X.

Syntax: *setX(index)*
 setX(class name)

Example: *dataset.setX(1);*

setY

Description: Sets variables to Y.

Syntax: *setY(index)*
 setY(class name)

Example: *dataset.setY(2); /* Sets variable 2 as a Y-variable */*

createModification

Description: Creates a modification for observations or variables-

Syntax: *createModification(direction, modification type)*
 createModification(direction, modification type, index)

Direction parameters:

- *obs* - *Observation data*
- *obsid* - *Observation description*
- *var* - *Variable data*
- *varid* - *Variable description*

Modification type parameters:

- *center*
- *uvscale*
- *logarithm*
- *class*
- *pareto*
- *quadterm*
- *interactionterm*

Example: *dataset.createModification("var", "center"); /* applies variable mean centering to the dataset */*
 dataset.createModification("obs", "class", 10); / creates an observation class using column 10 */*

createModel

Description: Creates a model from the specified dataset.

Syntax: *createModel(model type, components)*

Model type parameters:

- pca
- pls

Example: model = dataset.createModel("pca", 3); / creates a three-component pca-model from the dataset "dataset" */*

cloneDataSet

Description: Makes a copy of the current dataset

Syntax: cloneDataSet()

Example: preddataset2 = preddataset1.cloneDataSet() / makes a copy of preddataset1 into preddataset2 */*

get

Description: Gets a data matrix from the current dataset

Syntax: get(data matrix)

Common data matrix names:

- *originalmatrix*
- *case*
- *xtrain*
- *ytrain*
- *xtest*
- *ytest*

Example: originalMatrix = preddataset1.get ("originalmatrix") / set original matrix into variable originalMatrix */*

setName

Description: Set name of the current dataset

Syntax: setName(name)

Example: preddataset1.setName ("pred dataset 1") / set preddataset1 name to "pred dataset 1" */*

Model

createPrediction

Description: Returns the prediction statistics for the prediction dataset from the specified calibration model.

Syntax: `createPrediction(dataset)`

Example: `result = calibrationmodel.createPrediction(predictiondataset);`

get

Description: Returns a matrix from the data tree or from the prediction results.

Syntax: `get(data matrix)`

Common data matrix name:

- `xtr`
- `t`
- `p`
- `tpred`

Example: `T1 = calibrationmodel1.get("t"); /* gets T from model "calibrationmodel1" */`
`Tpred = result1.get("Tpred"); /* gets Tpred from prediction results "result1" */`

save

Description: Save the current model to disc

Syntax: `save(filepath)`

Example: `calibrationmodel1.save("C:\model.mdl"); /* Saves calibrationmodel1 to the file model.mdl */`

setName

Description: Set name of the current model

Syntax: `setName(name)`

Example: `calibrationmodel1.setName ("calibration model 1") /* set calibrationmodel1 name to "calibration model 1" */`

getComponents

Description: Get the number of components for the specified model. The returned value is represented as a string.

Syntax: `getComponents()`

Example: calibrationmodell.getComponents() / get the number of components for
calibrationmodell */*

Plot

addLayer

Description: Adds a layer to a plot.

Syntax: `addLayer(data)`

Example: `layer1 = score_plot1.addLayer(Tpred1); /* Adds Tpred1 as layer1 to the plot "score_plot1" */`

setTitle

Description: Adds a title to a plot.

Syntax: `setTitle(name)`

Example: `scatter2d_plot1.setTitle("Score plot 1"); /*Adds the title "Score plot 1" to the plot`

save

Description: Saves plot to specified filename

Syntax: `save(filename, format)`
`save(filename, format, X-resolution, Y-resolution)`

Format parameters:

- `png`
- `tiff`
- `jpg`
- `gif`

Example: `scatter2d_plot1.save("filename", "png", 500, 500); /* Saves the scatter2d_plot1 to "filename" with format "png". X-resolution is 500 and Y-resolution is 500`

addTextArea

Description: Add a text area into the given plot.

Syntax: `addTextArea(text, layout)`

Layout parameters:

- `top`
- `bottom`
- `left`
- `right`
- `center`
- `free`

Example: `scatter2d_plot1.addTextArea("Text", "bottom"); /*Adds a text area on the bottom of plot scatter2d_plot1 with the text "Text" */`

Plot Layer settings

setDataRange

Description: Sets which vector to plot for a given axis.

Syntax: `setDataRange(axis type, type, index)`

Axis type parameters:

- `x`
- `y`
- `z`

Type parameters

- `index`
- `value`

Example: `plot1.setDataRange("x", "value", 3); /* Plots the vector of index 3 on the X-axis. */`

setColor

Description: Sets the colours for the objects of a plot layer.

Syntax: `setColor(color name)`
`setColor(type, index[1..n])`

Description: Sets the colour of the objects in a plot.

Color name parameters:

- `red`
- `green`
- `blue`
- `yellow`
- `orange`
- `white`
- `black`
- `cyan`
- `darkgray`
- `gray`
- `lightgray`
- `magenta`
- `pink`

Type parameters:

- `fixed`
- `class`
- `index`
- `value`

- density
- altitude

Example: `score_plot.setColor(red)`
 `score_plot.setColor("class", 1); /* colours the objects in score_plot according`
 `to the class of index 1 */`

setShape

Description: Sets the shapes for the objects of a plot layer.

Syntax: `setColor(Shape name)`
 `setColor(type, option)`

Shape names:

- circlesquare
- uppertriangle
- lowertriangle
- diamond
- hollowcircle
- hollowsquare
- hollowuppertriangle
- hollowlowertriangle
- hollowdiamond
- plus
- cross
- crossandline
- line

Type parameters:

- fixed
- class

Example: `layer1.setShape("square");`
 `layer1.setShape(class,1); /* sets the shapes for the objects of layer1 according`
 `to the class of index 1 */`

setSize

Description: Sets the sizes for the objects of a plot layer.

Syntax: `setSize(size)`
 `setSize(type, index[1..n], minimum size, maximum size)`

Type parameters:

- fixed
- class
- index
- value
- perspective (3D scatter plots)

Example: `layer1.setSize(index,10,14); /* sets the objects in layer1 to sizes between 10 and 14 according to the index */`
`layer1.setSize(value,10,14,2) /* sets the objects in layer1 to sizes between 10 and 14 according the values of the vector of index 2 */`

setLabel

Description: Adds labels for the objects of a plot layer.

Syntax: `setLabel(type, index[1..n], size)`

Type parameters:

- fixed
- class
- index
- value
- identifier

Example: `layer1.setLabel("identifier", 2, 12); /* adds the second identifier to the objects of layer1 with a font size of 12 */`

setLine

Description: Adds lines between the objects of a plot layer.

Syntax: `setLine(type)`

Type parameters:

- index
- value

Example: `layer1.setLine("index"); /* sets lines between the objects of layer1 according to the index */`

setComment

Description: Adds comment to a plot layer. The comment will appear then the mouse is hovered over a object in the plot.

Syntax: `setComment(type)`

Type parameters:

- index
- value

Example: `layer1.setComment("index"); /* sets comment to objects of layer1 according to the index */`

setLegend

Description: Adds a legend to a plot layer.

Syntax: *setLegend(layer, type, layout)*
setLegend(type, layout)
setLegend(type)

Type parameters:

- color
- class

Layout parameters:

- top
- bottom
- left
- right - default
- center
- free

Example: *layer1.setLegend("color", "right"); /* adds legend from the color of the objects in layer1. The legend will be on the right side of the plot */*

Plot Window

setBackgroundColor

Description: Sets the background for the plot window

Syntax: `setBackgroundColor(color name)`

Color name parameters:

- red
- green
- blue
- yellow
- orange
- white
- black
- cyan
- darkgray
- gray
- lightgray
- magenta
- pink

Example: `score_plot.setBackgroundColor("red"); /* colours the background in score_plot to red I*/`

setBorder

Description: Sets the border for the plot window

Syntax: `setBorder(shape name, color name)`
 `setBorder(shape name)`

Share name parameters:

- none
- rectangle
- ellipse

Color name parameters:

- red
- green
- blue
- yellow
- orange
- white
- black
- cyan
- darkgray
- gray

- lightgray
- magenta
- pink

Example: score_plot.setBorder("ellipse", "red"); / set ellipse border with red color on
score_plot */*