

# Description of rbiouml package

Ivan Yevshin and Tagir Valeev

September 13, 2013

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Getting started</b>	<b>1</b>
2.1	Connecting to BioUML server . . . . .	1
2.2	Querying BioUML repository . . . . .	2
2.3	Using BioUML analyses . . . . .	3
2.4	Importing files to and from BioUML . . . . .	4
2.5	Disconnecting from BioUML server . . . . .	5

## 1 Introduction

The *rbiouml* package provides functions to interact with BioUML server from within R. BioUML is an open source integrated Java platform for analysis of data from omics sciences research and other advanced computational biology, for building the virtual cell and the virtual physiological human. It spans a comprehensive range of capabilities, including access to databases with experimental data, tools for formalized description of biological systems structure and functioning, as well as tools for their visualization, simulation, parameters fitting and analyses.

## 2 Getting started

### 2.1 Connecting to BioUML server

The first thing you need to do is load the package and login to the BioUML server. As an example we will connect to the free public BioUML server at <http://ie.biouml.org>. The function `biouml.login` connects to BioUML server and perform authentication.

```
> library(rbiouml)
> biouml.login("ie.biouml.org")
```

The `biouml.login` also accepts user and password, but we leave them empty in the example above for anonymous login. Alternatively you can install BioUML on your local computer and connect to it in the same way.

```
> biouml.login("localhost:8080")
```

See [http://wiki.biouml.org/index.php/BioUML\\_server\\_installation](http://wiki.biouml.org/index.php/BioUML_server_installation) for details on BioUML server installation.

## 2.2 Querying BioUML repository

The BioUML repository (or simply repository) is the central BioUML data storage place. Basically, all the data you work with in BioUML is stored in the repository. The repository has a hierarchical structure similar to file systems. On the top level the repository consists of several root folders. The most common ones are:

**databases** contains preinstalled or user-defined modules.

**data** contains user projects and public examples.

The `biouml.ls` function lists the contents of given folder in repository. The list of databases available in BioUML server:

```
> biouml.ls("databases")
```

```
[1] "Apoptosis model"
[2] "Biomodels"
[3] "Biopath"
[4] "DSMTS"
[5] "EHMN"
[6] "Ensembl"
[7] "EnsemblHuman52"
[8] "EnsemblMouse"
[9] "EnsemblRat"
[10] "GO"
[11] "GeneWays"
[12] "KEGG"
[13] "Reactome"
[14] "SBML tests"
[15] "The composite model of CD95 and NF-kB signaling"
[16] "Utils"
[17] "Virtual Human"
```

The list of data elements available in BioUML examples folder:

```
> biouml.ls("data/Examples/Optimization/Data/Experiments")
```

```
[1] "exp_data_1" "exp_data_2" "exp_data_3"
```

The `biouml.get` fetches a table from BioUML repository as R data.frame:

```
> x <- biouml.get("data/Examples/Optimization/Data/Experiments/exp_data_1")
> class(x)
```

```
[1] "data.frame"
```

```
> head(x)
```

	time	p43p41	pro8	casp8
0	0	0.05772537	59.96316	0.00000000
1	10	0.26814367	57.56464	0.04107502
2	20	4.76048117	58.58981	0.31611658
3	30	8.25193519	59.42156	1.39735609
4	45	16.14448337	48.18975	3.52037089
5	60	17.02060557	38.95027	3.94722894

This function allows to fetch not only true BioUML tables, but any data elements which have tabular representation, including profiles, user uploaded tracks and so on.

To store data.frame as a table into BioUML repository use `biouml.put` function:

```
> x[,5] <- x[,3] + x[,4]
> biouml.put("data/Collaboration/Demo/Data/Rtest/exp_data_1_sum", x)
> biouml.ls("data/Collaboration/Demo/Data/Rtest")
```

```
[1] "exp_data_1_sum"
```

## 2.3 Using BioUML analyses

BioUML provides a set of analyses organized in groups. The list of analyses available in the current server can be fetched with `biouml.analysis.list` function.

```
> summary( biouml.analysis.list() )
```

	Group	Name	
BSA	:50	Adaptive simulated annealing	: 1
Import	:37	Add expression values	: 1
Binding regions	:22	Affymetrix normalization	: 1
Data	:20	Agilent normalization	: 1
GTRD	:12	Analysis of Cis-Regulatory Modules	: 1
Molecular networks	:10	Annotate table	: 1
(Other)	:31	(Other)	:176

Each biouml analysis has a set of parameters, `biouml.analysis.parameters` returns a `data.frame` with row names corresponding to parameter names and one column 'description'.

```
> biouml.analysis.parameters("Filter table")
```

	description
inputPath	Table to filter
filterExpression	Expression in JavaScript like 'ColumnName1 > 5 && ColumnName2 < 0'
filteringMode	Which rows to select
outputPath	Path to the filtered table

The `biouml.analysis` launches analysis with given parameters.

```
> biouml.analysis("Filter table", list(
+   inputPath="data/Examples/Optimization/Data/Experiments/exp_data_1",
+   filterExpression="time < 40",
+   outputPath="data/Collaboration/Demo/Data/Rtest/exp_data_1 filtered"
+ ))
```

```
100 %
INFO - Analysis 'Filter table' added to queue
INFO - Analysis 'Filter table' started
INFO - Filtering...
INFO - Writing result...
INFO - Analysis 'Filter table' finished (0.244 s)
INFO - Completed
[1] "RJ0B65342751"
```

## 2.4 Importing files to and from BioUML

As described previously, `data.frames` can be fetched from and stored to BioUML repository using `biouml.get` and `biouml.put` functions. In addition, data can be imported from files and exported to files in various formats. The list of importers can be obtained with `biouml.importers` function.

```
> head( biouml.importers() )
```

```
[1] "SDF file"
[2] "SIF"
[3] "TRANSFAC matrices format (matrix*.dat)"
[4] "TRANSFAC profile (*.prf)"
[5] "ZIP-archive (*.zip)"
[6] "Generic file"
```

As an example we will import fasta file to BioUML.

```
> hiv.genome <- system.file("extdata","HIV-1.fa", package="rbiouml")
> output.folder <- "data/Collaboration/Demo/Data/Rtest"
> biouml.import(hiv.genome, output.folder, importer="Fasta format (*.fasta)")

100 %
data/Collaboration/Demo/Data/Rtest/HIV-1[1] "data/Collaboration/Demo/Data/Rtest/HIV-1"

> biouml.ls(output.folder)

[1] "exp_data_1 filtered" "exp_data_1_sum"      "HIV-1"
```

Similarly, we can use `biouml.export*` functions to export data from BioUML repository.

```
> head( biouml.exporters() )

[1] "Interval format (*.interval)"
[2] "Archive containing exported elements (*.zip)"
[3] "HTML document (*.html)"
[4] "Tab-separated text (*.txt)"
[5] "Scalable Vector Graphics(*.svg)"
[6] "BioUML format(*.dml)"

> biouml.export("data/Collaboration/Demo/Data/Rtest/HIV-1",
+   exporter="Fasta format (*.fasta)", target.file="HIV-1.fa")
> file.exists("HIV-1.fa")

[1] TRUE
```

## 2.5 Disconnecting from BioUML server

When you have finished with the BioUML server is recommended to do logout.

```
> biouml.logout()

NULL
```