1. Importando dados iniciais

É possível importar os dados iniciais de uma atividade para as tabelas apropriadas no Sagitarii usando o aplicativo de processamento de atividades “Sagitarii Teapot”, desde que as tabelas que irão receber os dados possuam, no mínimo, os mesmos atributos (campos) que o arquivo CSV possui (colunas).

Clique no ícone “Download Teapot Cluster” no topo da tela. Após fazer o download do arquivo, descompacte-o em um diretório qualquer. O arquivo CSV que será importado deverá estar neste mesmo diretório. Crie uma pasta “outbox” logo abaixo deste diretório e grave ali todos os demais arquivos que serão importados juntamente com os dados CSV. Estes arquivos deverão estar referenciados em alguma coluna dentro do CSV. Edite o arquivo “config.xml”, colocando o endereço do servidor Sagitarii na tag “hostURL”.

Para importar os dados, execute o Teapot usando os parâmetros:

java -jar teapot-1.0.125.jar upload <nome_do_arquivo_csv> <nome_da_tabela_destino> <tag_do_experimento> <pasta_de_trabalho>

Exemplo:

java -jar teapot-1.0.125.jar upload mydata.csv mytable 2D7B8C77-8822-49B /home/user/insurance

Neste caso, mydata.csv está na pasta /home/user/insurance e os arquivos de dados em /home/user/insurance/outbox. É importante verificar o arquivo CSV e confirmar se as colunas da primeira linha do arquivo possuem os mesmos nomes dos atributos da tabela que receberá os dados do arquivo. É importante verificar também se o CSV está bem formatado no padrão RFC 4180, sendo obrigatório a presença dos nomes das colunas. A figura 9 ilustra um exemplo de arquivo CSV aceito pelo Sagitarii. Neste caso, a coluna “filename” representa um atributo tipo “File” em uma tabela no banco de dados do Sagitarii e os arquivos descritos por esta coluna estão na pasta “outbox”. O Teapot enviará este CSV e seus arquivos ao Sagitarii.

O Sagitarii gravará os arquivos na tabela de arquivos, receberá seus índices e trocará cada nome de arquivo por seu índice, já que o atributo “File” na tabela de dados é na verdade do tipo “Integer”, com chave estrangeira para a tabela de arquivos. A figura 10 ilustra a tabela que receberia os dados do CSV do exemplo.

```
1  filetype,filename
2  type1,teste.csv
3  type2,teste2.csv
4  type3,teste3.csv
5  type4,teste.csv
6  type5,teste4.csv
```

Figura 9. Exemplo de um arquivo CSV

```
CREATE TABLE mytable
(
  index_id serial NOT NULL,
  filetype character varying(250),
  filename integer,
  CONSTRAINT mytable_pkey PRIMARY KEY (index_id),
  CONSTRAINT mytable_fkac FOREIGN KEY (filename)
      REFERENCES files (id_file)
)
```

Figura 10. Esquema da tabela no banco de dados

Antes de enviar os dados, o experimento já deverá estar criado, pois cada envio de dados deve estar no contexto de um experimento.

2. Integração com o "R"

O Sagitarri oferece uma integração especial com o "R". Segundo definição do próprio site:

"R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control."

Naturalmente, a interação com o "R" poderia ser feita usando um wrapper, mas o Sagitarii possui um wrapper especial, que utiliza a biblioteca JRI instalada com o rJava.

3. Instalando o "R"

Após baixar e instalar o "R", é necessário configurar as variáveis de ambiente "PATH" e "R_HOME". A variável "PATH" precisa conter os caminhos para as pastas "bin\x64" e "library", ambas na pasta de instalação do "R". A variável "R_HOME" deve apontar para a pasta de instalação do "R".

Após instalar o "R", é necessário instalar o rJava, apenas para usar o JRI. Caso o usuário queira criar um wrapper que utilize o JRI, é necessário incluir a pasta das bibliotecas no "classpath":

java -Djava.library.path=<RJAVA_ROOT>/jri -jar <SEU_WRAPPER>.jar

IMPORTANTE: A pasta onde estão localizadas as bibliotecas do JRI deve ser

configurada no arquivo “config.xml” em todos os nós de execução (Teapot), na tag “rPath”.

Para testar a instalação do “R” e a integração com o Java, pode-se executar o wrapper criado para executar atividades “R” do Sagitarii:

java -Djava.library.path=/usr/lib64/R/library/rJava/jri/

-jar r-wrapper.jar /home/my/work/folder

Este wrapper, como todos os outros, só estarão disponíveis após o nó de execução Teapot ter sido executado pelo menos uma vez, pois são baixados do servidor Sagitarii pelo Teapot.

4. Criando executores para o “R”

O Sagitarii possui o executor “executor_r”, do tipo “RRUNNER”, que executa o wrapper “r-wrapper.jar”. Este wrapper é encarregado de executar os scripts R do usuário e já está presente na instalação padrão do Sagitarii, não podendo ser removido.

As atividades irão utilizar os executores do tipo “RSCRIPT” para executar scripts “R”. Quando o nó de execução Teapot é iniciado, ele irá fazer o download de todos os executores, incluindo os wrappers, os scripts “R” e bibliotecas. Uma atividade do tipo “RSCRIPT” usa um script “R” no lugar de um wrapper, então o wrapper “RRUNNER” (“r-wrapper.jar”) é executado, chamando o script em questão e passando os dados e arquivos recebidos. O wrapper “r-wrapper.jar” passará ao script “R” o caminho da pasta de trabalho da atividade que será executada usando o JRI. O script precisará desta informação para acessar o arquivo CSV com os dados necessários a sua execução, bem como os arquivos de dados enviados pelo servidor. O script terá esta informação disponível para acesso imediato na variável “sagitariiWorkFolder”. A tabela 3 possui os caminhos das pastas que deverão ser utilizadas pelo script e sua finalidade.

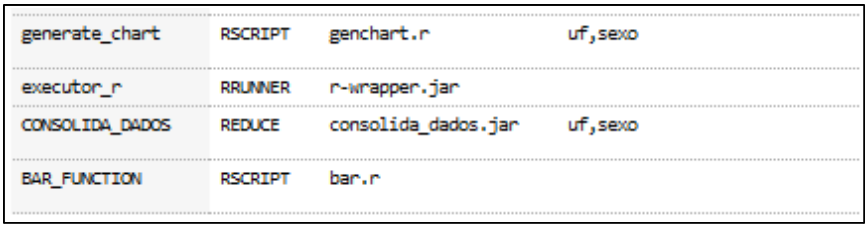
Atividades do tipo “RSCRIPT” podem ser associadas a todos os operadores, exceto “QUERY”, pois este é executado internamente pelo próprio servidor Sagitarii.

Tabela 3. Lista das pastas usadas pelo script “R”

sagitariiWorkFolder	Caminho completo da pasta de trabalho da instância da atividade que está executando o script. Esta pasta contém o arquivo “sagi_input.txt”, que é o arquivo CSV com os dados de execução da atividade. Nesta pasta o script DEVERÁ gravar o arquivo “sagi_output.txt” contendo os dados CSV de saída da atividade.
SagitariiWorkFolder / inbox	Pasta contendo os arquivos de dados do usuário enviados pelo servidor, que foram produzidos pela atividade anterior ou por carga inicial.
SagitariiWorkFolder	Pasta de saída dos arquivos produzidos pela atividade.

/ outbox	Qualquer arquivo que deva ser passado para a atividade seguinte deverá ser gravado nesta pasta e referenciado em uma das colunas do arquivo CSV “sagi_output.txt”.

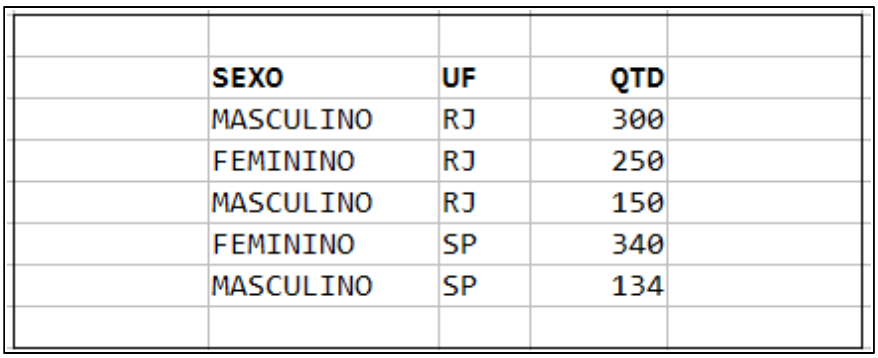
Para usar um executor “RSCRIPT” com uma atividade “REDUCE”, é necessário informar um critério de agrupamento, que são os campos da tabela de entrada, separados por vírgula (figura 16). O Sagitarii irá aplicar um “SELECT DISTINCT” na tabela de entrada usando estes atributos e gerar uma instância para cada grupo destes registros onde os demais atributos forem diferentes. A figura 16.a ilustra uma possível situação de agrupamento. Usando os dados da figura, o Sagitarii faria um “select distinct”, resultando em “MASCULINO,RJ”, “FEMININO,RJ”, MASCULINO,SP” e “FEMININO,SP”. Depois criaria instâncias onde os dados diferem destes resultados (figura 16.b).



generate_chart	RSCRIPT	genchart.r	uf,sexo
executor_r	RRUNNER	r-wrapper.jar	
CONSOLIDA_DADOS	REDUCE	consolida_dados.jar	uf,sexo
BAR_FUNCTION	RSCRIPT	bar.r	

Figura 16. Executor para script “R”

A figura 17 ilustra a tela de cadastro de executores com os dados do executor “generate_chart” do exemplo.



SEXO	UF	QTD
MASCULINO	RJ	300
FEMININO	RJ	250
MASCULINO	RJ	150
FEMININO	SP	340
MASCULINO	SP	134

Figura 16.a. Exemplo de dados a serem agrupados pelo “REDUCE”

Instance 1	MASCULINO	RJ	300
	MASCULINO	RJ	150
Instance 2	FEMININO	RJ	250
Instance 3	FEMININO	SP	340
Instance 4	MASCULINO	SP	134

Figura 16.b. Exemplo de instâncias geradas pelo "REDUCE" agrupando por SEXO,UF

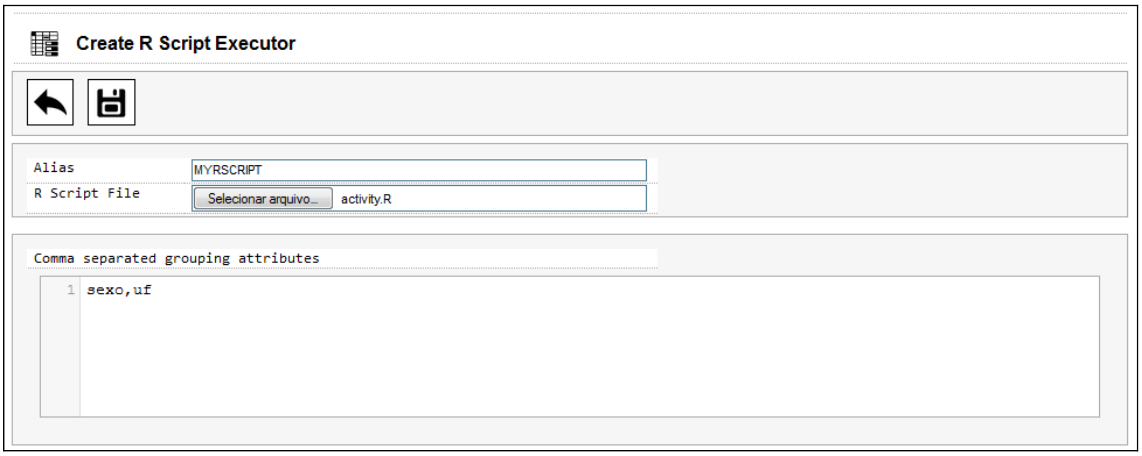


Figura 17. Criação de um executor de script "R"

O Teapot, ao se deparar com uma atividade do tipo "RSCRIPT", executa o wrapper "r-wrapper.jar", que usa o JRI para configurar a pasta de trabalho e executar o script "R", que deverá se encarregar de ler os arquivos necessários (sagi-input.txt e a pasta inbox) e gravar os arquivos produzidos (sagi_output.txt e pasta outbox, caso hajam). A figura 18 ilustra um script "R" usado como exemplo. Caso o script necessite de outros scripts (funções e bibliotecas), cadastre-os como "Biblioteca", assim o Teapot poderá fazer download de todos os scripts necessários. Este recurso foi usado no script de exemplo da figura 18, linha 15. A figura 18a exibe a tela de cadastro de bibliotecas de apoio para os wrappers (quaisquer arquivos que serão utilizados pelos wrappers para seu funcionamento – jar, lib, dat, etc...). Estes arquivos de apoio serão baixados pelo Teapot e colocados na mesma pasta que os wrappers.

```
1  #!/usr/bin/env Rscript
2
3  inputFileFolder <- paste( sagitariiWorkFolder, "inbox", sep = "/")
4  outputFileFolder <- paste( sagitariiWorkFolder, "outbox", sep = "/")
5
6  inputFile <- paste( sagitariiWorkFolder, "sagi_input.txt", sep = "/")
7  inputData <- paste( inputFileFolder, "wine.data", sep = "/")
8  outputFile <- paste( sagitariiWorkFolder, "sagi_output.txt", sep = "/")
9
10 param <- read.table( inputFile, header = TRUE, sep = ",")
11 wine <- read.table( inputData, header = TRUE, sep = ",")
12
13 library(nnet)
14
15 source("data-preprocessing.R")
```

Figura 18. Script "R" utilizando a variável de pasta de trabalho

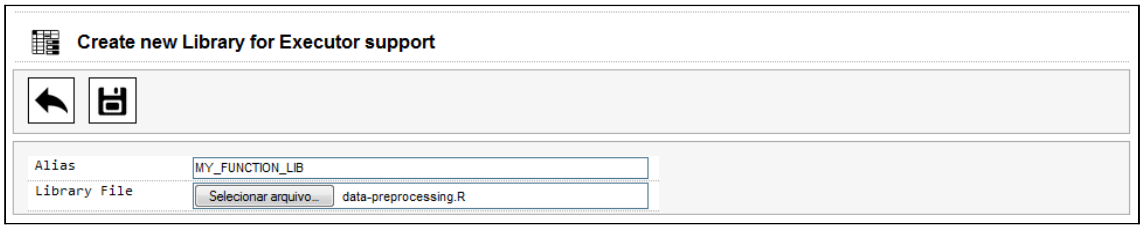


Figura 18a. Cadastro de uma biblioteca de funções para o script "R"

Na figura 18 pode-se perceber o uso da variável contendo a pasta de trabalho da atividade (linhas 3 até 8) na composição das pastas de trabalho, entrada e saída de dados do script (na linha 11, o script carrega um arquivo da dados enviado pelo Sagitarii, que está na pasta "inbox") . Estas linhas provavelmente estarão presentes em todos os scripts R usados pelo Sagitarii. A figura 19 ilustra uma atividade de exemplo que usa um executor R.

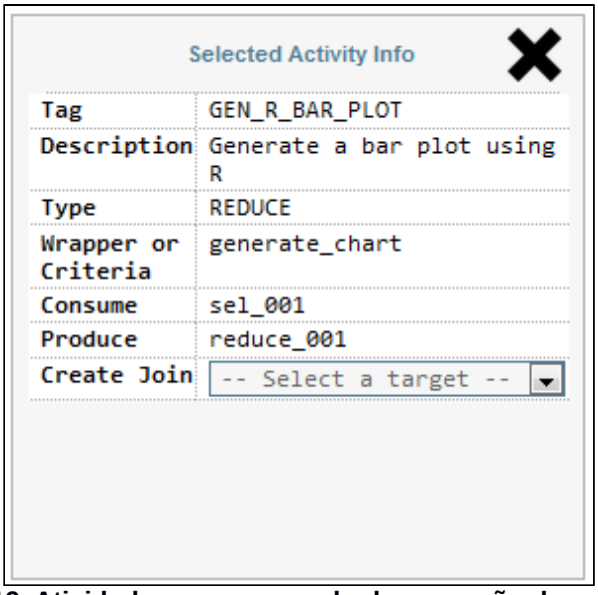


Figura 19. Atividade para o exemplo de execução de script "R"

O script “R” deverá se encarregar de gravar os dados de saída na pasta “outbox” e no arquivo “sagi-output.txt”. O arquivo “sagi_output.txt” deverá conter os dados que serão gravados na tabela de saída da atividade, no formato CSV, com os nomes das colunas coincidindo com os atributos daquela tabela. Qualquer coluna que contenha nomes de arquivos que existam na pasta “outbox” fará com que o Teapot envie estes arquivos para o Sagitarii. Esta coluna deverá estar identificada como tipo “File” na tabela de destino. A figura 20 ilustra um trecho de um script “R” gravando seus dados de saída.

Na linha 52, uma nova coluna “filename” é adicionada ao *array* de dados. Na linha 63 o nome do arquivo é criado. Na linha 64, o nome do arquivo é adicionado ao caminho completo da pasta de saída “outbox” (criado na linha 4 do script da figura 18, agregando a variável passada pelo Teapot com a pasta de trabalho “SagitariiWorkFolder” com o nome da pasta de saída “outbox”).

Na linha 65, o nome do arquivo é colocado no CSV de saída, abaixo da coluna “filename”, criada na linha 52. A linha 66 grava então este arquivo de dados na pasta de saída.

A linha 70 termina por gravar o CSV produzido (“sagi_output.txt”) na pasta de trabalho da atividade (não na pasta “outbox”), representada pela variável “outputFile”, criada na linha 8 do script da figura 18 agregando o nome do CSV de saída "sagi_output.txt" com a variável passada pelo Teapot contendo a pasta de trabalho da atividade (“SagitariiWorkFolder”). É importante que o CSV produzido seja bem formatado no padrão RFC 4180, sendo obrigatório a presença dos nomes das colunas.

```
51  param$resultado <- NULL
52  param$filename <- NULL
53  for (i in 1:nrow(param) ) {
54    tnet <- nnet(data.train, alvo.class.train, size=param[i,"size"],
55      decay=param[i,"decay"], maxit=param[i,"maxit"])
56    pnet <- predict(tnet, data.test, type="raw")
57    roc_data <- croc(pnet, alvo.class.test)
58    tx <- unlist(slot(roc_data, "y.values"))
59
60    newTx <- gsub(",", ".", tx[2])
61
62    param$resultado[i] <- newTx
63    filename <- paste(paste("result-",as.character(i), sep=""), ".csv", sep="")
64    fullFileName <-  paste(outputFileFolder, filename, sep="/")
65    param$filename[i] <- filename
66    write.table(pnet, file=fullFileName, row.names = FALSE,
67      col.names = TRUE, sep = ",", dec = ".", quote = TRUE)
68  }
69  param$dataset <- NULL
70  write.table(param, file=outputFile, row.names = FALSE,
71      col.names = TRUE, sep = ",", dec = ".", quote = FALSE)
```

Figura 20. Script “R” gravando os dados de saída

A tabela 4 resume os passos necessários para criar e executar um script R.

Tabela 4. Resumo de atividades para criar executar um script “R” no Sagitarii

Ao criar um script, utilize a variável fornecida pelo Teapot “sagitariiWorkFolder” para montar os nomes das pastas e arquivos necessários ao script.
Leia o arquivo “sagi_input.txt” que contém os dados CSV enviados pelo Sagitarii, necessários para a execução da atividade. Estes são os dados da tabela de entrada da atividade. Caso o Sagitarii tenha enviado algum arquivo de dados, eles estarão na pasta “inbox”.
Trabalhe com os dados em seu script normalmente.
Produza o CSV de saída, com os dados que deseja gravar na tabela de saída da atividade. As colunas deste CSV devem coincidir em nome e tipo com os atributos da tabela de saída existente no banco de dados (exceto no caso colunas com nomes de arquivos, que no banco será “Integer” (File) e a coluna do CSV será “String”, mas o Sagitarii sabe como contornar isso).
Caso seu script R produza arquivos adicionais (imagens, dados compactados, áudio, vídeo, etc...) estes arquivos devem ser gravados na pasta “outbox” e referenciados em colunas no CSV de saída, sendo que estas colunas devem coincidir com os atributos tipo “File” na tabela de saída da atividade.
Grave o CSV com os dados da tabela de saída com o nome “sagi_output.txt”, na pasta de trabalho da atividade (mesmo lugar onde está o arquivo “sagi_input.txt”.
O Teapot vai abrir o CSV de saída “sagi_output.txt” e verificar se precisa enviar os arquivos da pasta “outbox”.
Ao receber o arquivo CSV e os arquivos de dados, o Sagitarii grava os arquivos de dados no repositório de arquivos, recebe seu índice e troca o nome do arquivo pelo seu índice dentro do arquivo CSV, pois o atributo tipo “File” na tabela de dados é na realidade tipo “Integer” com chave estrangeira para a tabela de repositório de arquivos.
Após gravar todos os arquivos e preparar o CSV, o Sagitarii insere as linhas do CSV na tabela de saída da atividade. Colunas existentes no CSV sem atributos correspondentes não serão gravados e atributos existentes na tabela sem correspondentes no CSV ficarão como NULL.