Using R in Taverna

Aleksandra Pawlik
myGrid Team
University of Manchester

VLIZ, 2014-10-06 / 2014-10-08
http://www.taverna.org.uk/

Introduction

- R is a popular scripting language oriented towards statistical computing
- There are a large number of modules that add functionality to R such as BioConductor and rCDK
- The Rshell service in Taverna allows workflows to include services that run R scripts on an installation of R
- R can be located on the same machine as you use to run the workflow, or on a different machine
- To allow Taverna to talk to the R installation, Rserve must also be running on the same machine as R

R Pages on the Taverna Wiki

- Installation of a local R Server may be too much for today's tutorial
- In which case just read through the slides to see if Installing R is something you need/want
- More information on Taverna and R can be found at:
  - http://dev.mygrid.org.uk/wiki/display/tav250/Rshell

Installation of R

- Documentation available from:
  - http://cran.r-project.org/doc/manuals/R-admin.html
- Windows
  - Download executable file
  - http://cran.r-project.org/bin/windows/base/
- Linux
  - Depends on the version of linux
  - http://cran.r-project.org/
- Mac
  - Download pkg file
  - http://cran.r-project.org/bin/macosx/

Installation of Rserve

- After installing R, the easiest way to install Rserve is to install it from CRAN. Simply use in R:
  - install.packages("Rserve”);
- Since Rserve comes as an R package, you can start Rserve within R by typing:
  - library(Rserve);
  - Rserve();
- Please note that if you get an error (*Fatal error: you must specify '--save', '--no-save' or '--vanilla'*), then start Rserve with the following command
  - Rserve(args="--no-save")

Configuration of Rserve

- Rserve is configured by the configuration file located at /etc/Rserv.conf
- Configuration of Rserve on your R installation has already done using a Rserv.conf file
- Documentation on configuring Rserve
  - http://www.rforge.net/Rserve/doc.html#conf

Checking R installation

- Download the Increment_an_integer workflow from the myExperiment group
    - See: http://www.myexperiment.org/workflows/4488
- Open the workflow in Taverna and run it
- If you use the default input value, then the result will be *8*

An example statistical analysis

- To illustrate how to use the Rshell service, we will carry out a simple statistical analysis on a small hypothetical set of species incidence data from 4 species measured from 6 sites:

Species	*N1*	*N2*	*A1*	*A2*	*B1*	*B2*
Species_A	90	110	190	210	290	310
Species_B	190	210	390	410	590	610
Species_C	90	110	110	90	120	80
Species_D	200	100	400	90	600	200

An example statistical analysis

- This data set can be found in a comma-separated file named biodiv_R_testdata.csv in the myExperiment group under “Biodiversity Test Data for Rservice Tutorial”.
- See: http://www.myexperiment.org/files/1078.html
- Download the file.
- We will read from the file during the workflow run

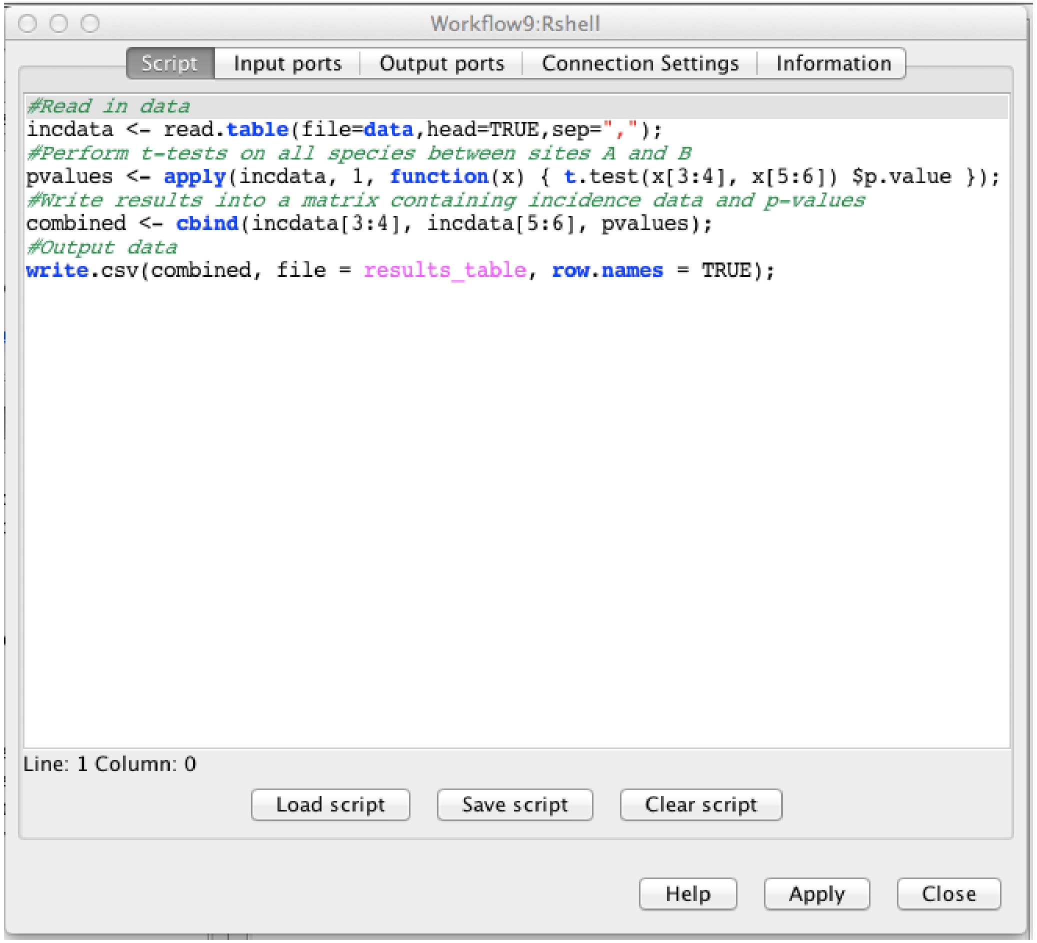
Adding an Rshell service

- Add an Rshell service to a workflow by locating it under **Service templates** in the Service panel and dragging it onto the workflow diagram
- A window will appear to configure the use of the Rshell service

Configuring a Rshell service

- The configuration of the Rshell service is split into several tabs
- Each tab has Apply and Close buttons at the bottom. Apply saves the configuration as shown in the tabs, and Close closes the configuration dialog

Configuring a Rshell service



```
#Read in data
incdata <- read.table(file=data,head=TRUE,sep=",");
#Perform t-tests on all species between sites A and B
pvalues <- apply(incdata, 1, function(x) { t.test(x[3:4], x[5:6]) $p.value });
#Write results into a matrix containing incidence data and p-values
combined <- cbind(incdata[3:4], incdata[5:6], pvalues);
#Output data
write.csv(combined, file = results_table, row.names = TRUE);
```

Configuring a Rshell service

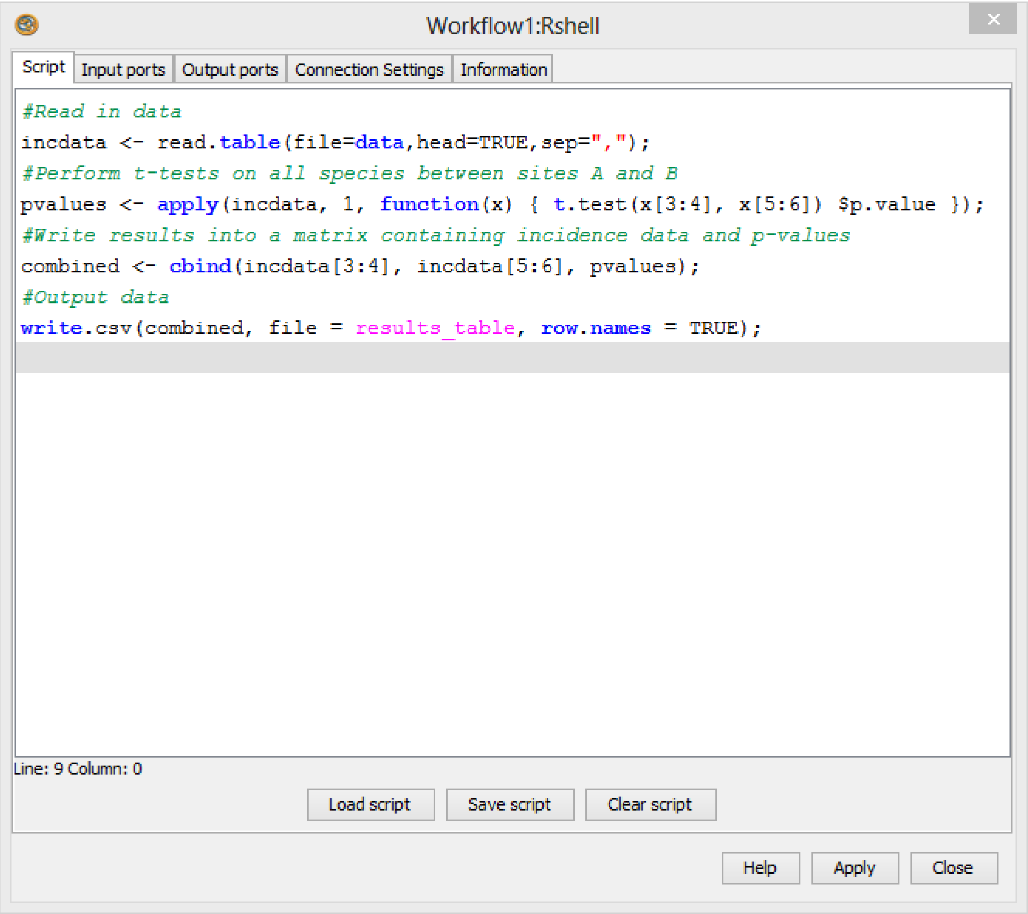
- The first tab of the Rshell configuration is used to enter the R script that will be executed
- We will use an R script that will perform a series of *t*-tests to see if species incidence differs significantly between site A and site B.
- You should be careful about performing a *t*-test on as little as 2 replicates - this example is just for illustrative purposes

R script

```
#Read in data
incdata <- read.table(file=data,head=TRUE,sep=",");
#Perform t-tests on all species between sites A and B
pvalues <- apply(incdata, 1, function(x) { t.test(x[3:4], x[5:6]) $p.value });
#Write results into a matrix containing incidence data and p-values
combined <- cbind(incdata[3:4], incdata[5:6], pvalues);
#Output data
write.csv(combined, file = results_table, row.names = TRUE);
```

- Copy and paste the above script into the Script tab of the Rshell configuration box

R script



```
#Read in data
incdata <- read.table(file=data,head=TRUE,sep=",");
#Perform t-tests on all species between sites A and B
pvalues <- apply(incdata, 1, function(x) { t.test(x[3:4], x[5:6]) $p.value });
#Write results into a matrix containing incidence data and p-values
combined <- cbind(incdata[3:4], incdata[5:6], pvalues);
#Output data
write.csv(combined, file = results_table, row.names = TRUE);
```

Line: 9 Column: 0

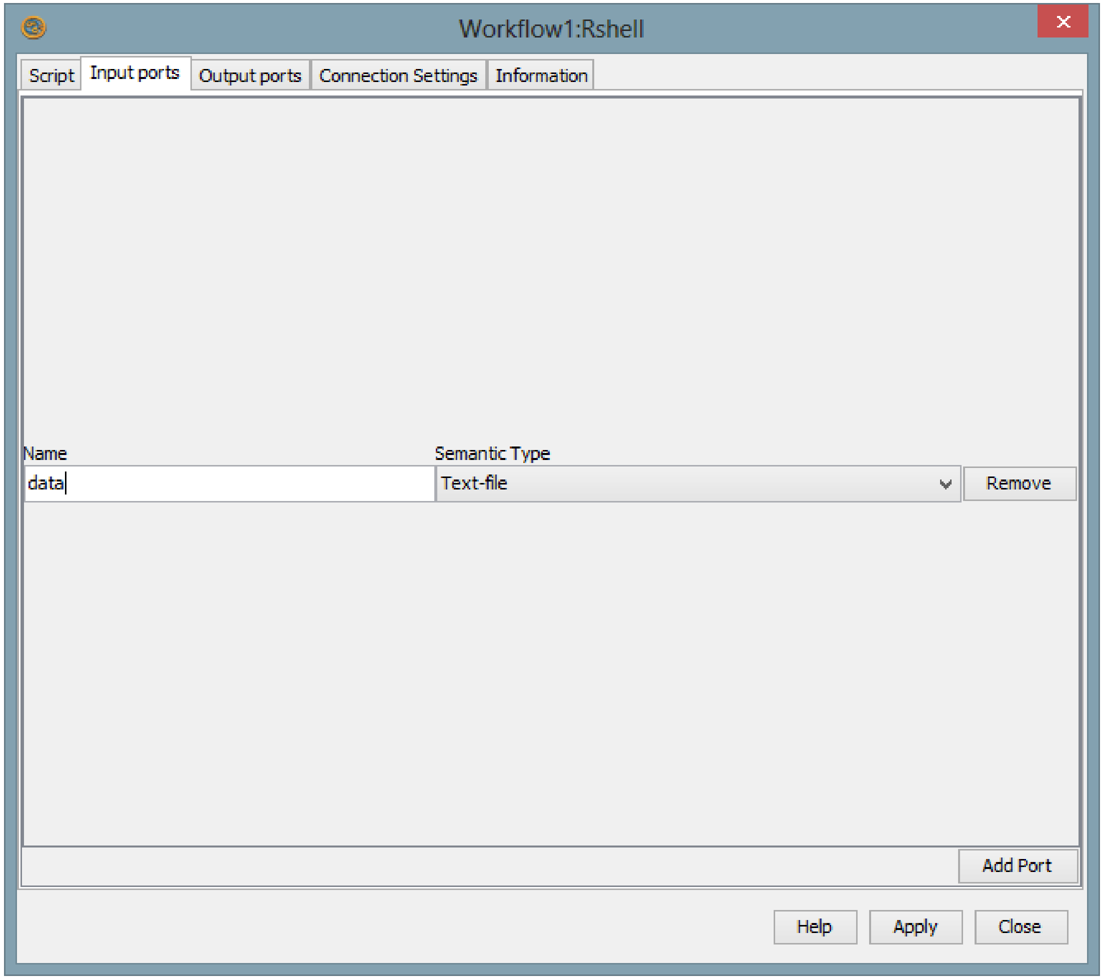
Rshell input and output ports

- Input and output ports are the connection points between the rest of the workflow and the Rshell service
- Rshell makes input ports available as variables in the script named after the port.
- Output ports read their named variable after executing the script. The last assigned value to the variable will be the one returned from the service via the output port.

Rshell input and output ports

- To add an input port:
  - Select the **Input ports** tab from the Rshell configuration dialog
  - Click **Add Port** button
  - Enter the name of the input port, for this example use ‘data’
  - Specify the input port type, for this example use ‘Text-file’

Rshell input and output ports



Name: data

Semantic Type: Text-file

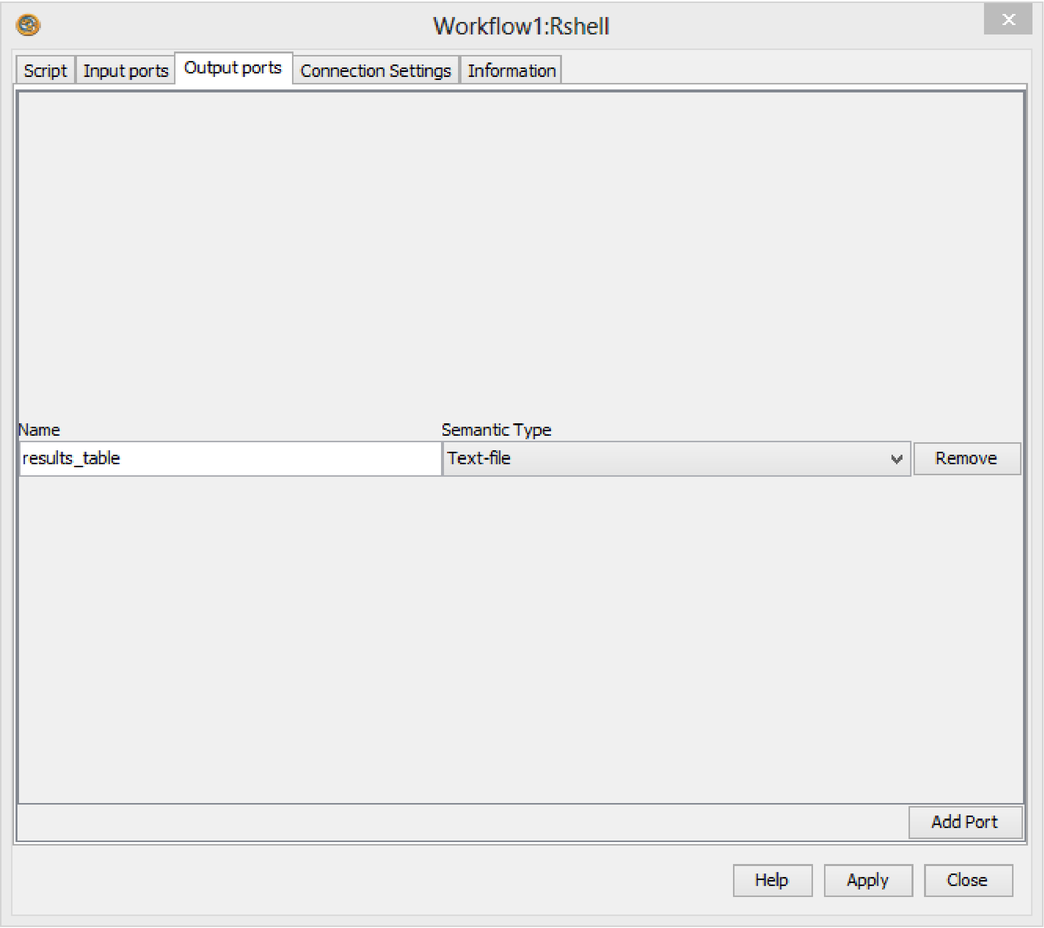
Rshell input and output ports

- The input port type indicates the data type this variable will have within the R-script. The possible types for R input ports are:
    - Logical
    - Numeric
    - Integer
    - String
    - Logical vector
    - Numeric vector
    - Integer vector
    - String vector
    - Text-file

Rshell input and output ports

- An output port can be added in a similar way:
  - Select the **Output ports** tab from the Rshell configuration dialog
  - Click **Add Port** button
  - Enter the name of the output port, for this example use ‘results_table’
  - Specify the output port type, for this example use ‘Text-file’

Rshell input and output ports



Name: results_table

Semantic Type: Text-file

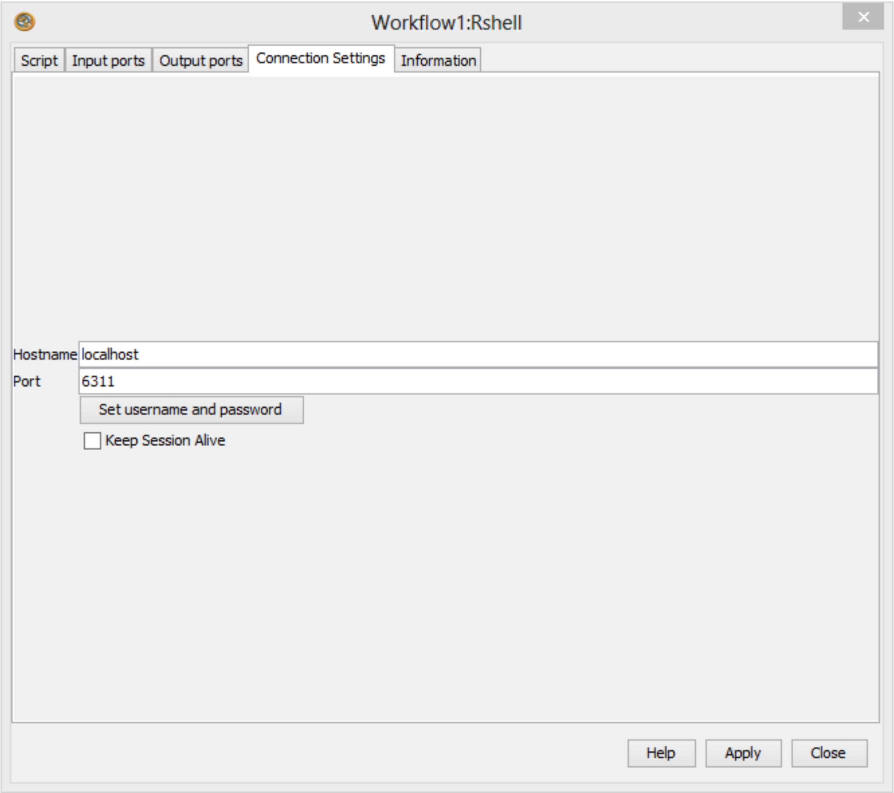
Rshell input and output ports

- The output port type indicates the type this variable has within the R-script. The possible types for R output ports are:
    - Numeric
    - Integer
    - String
    - Logical vector
    - Numeric vector
    - Integer vector
    - String vector
    - Text-file

Rshell connection settings

- Configuration of the connection parameters for Rserve is done using the Connection settings tab. This tab can be used to:
  - Configure the Rshell to use an Rserve installation on a different machine to where you run the Taverna workbench
  - Configure the access of Rserve on a different port
  - Provide authentication details for accessing Rserve in the form of a username and password
- If you are using Rserve on the same machine that you are running Taverna on then you probably do not need to change the connection settings

Rshell connection settings



Workflow1:Rshell

Hostname localhost

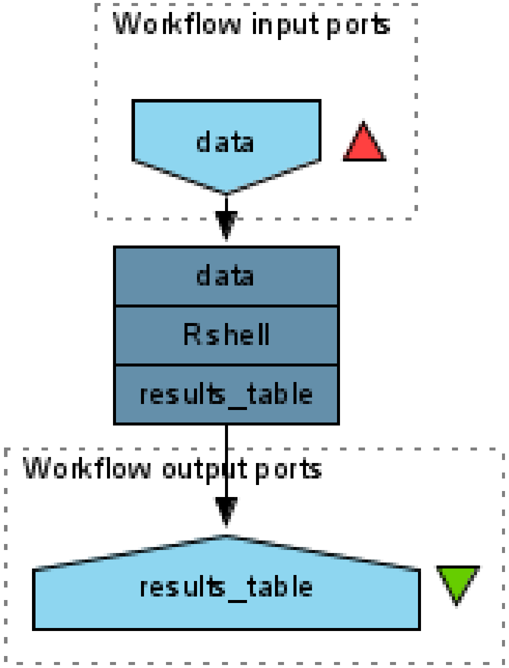
Port 6311

☐ Keep Session Alive

Completing the workflow

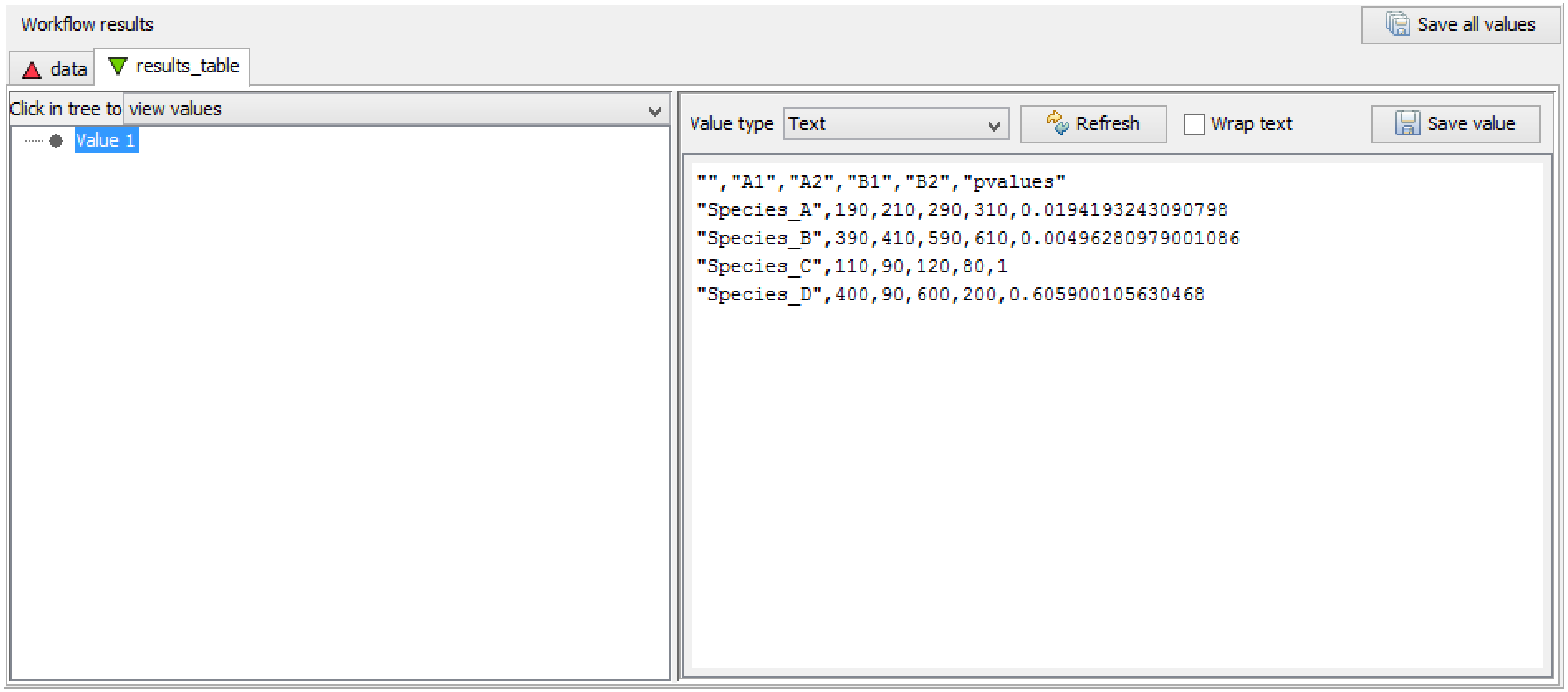
- To complete the workflow:
  - Create a workflow input "data" and connect it to the data input port of the Rshell service
  - Create a workflow output from the results_table output port of the Rshell service
- Your workflow should now look as follows:

Completing the workflow



Results

When you run the workflow, you should get the following results:



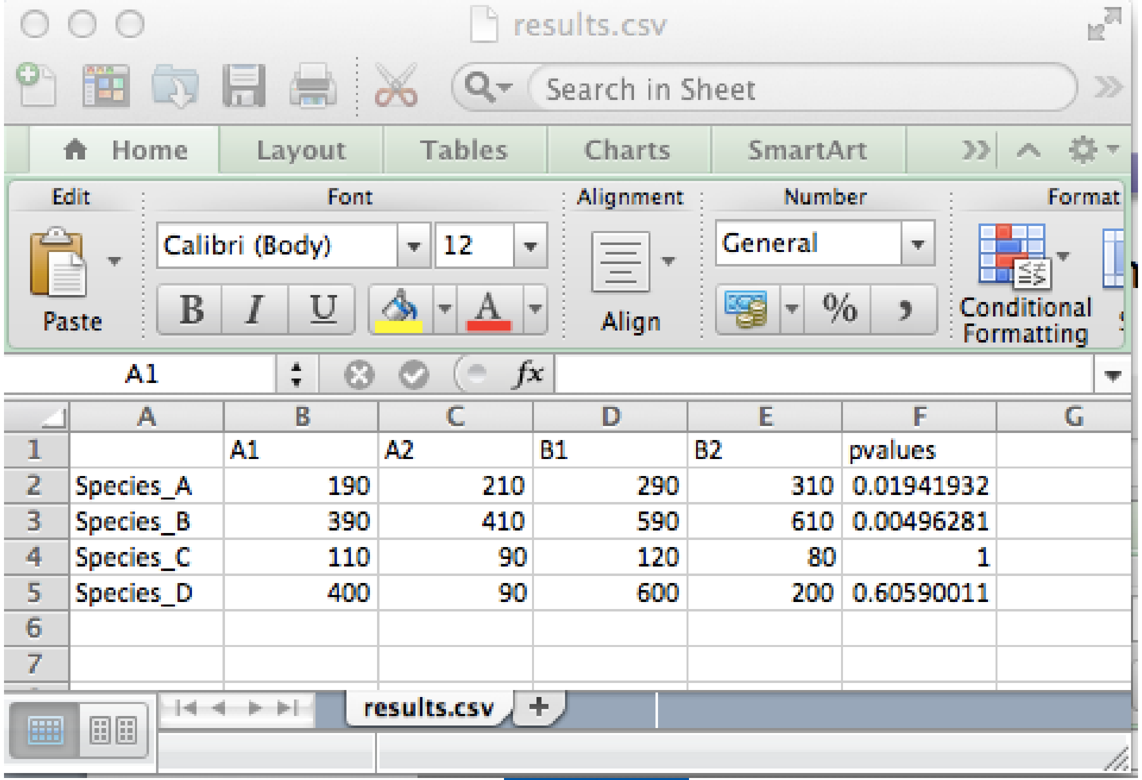
Value type: Text

☐ Wrap text

```
"","A1","A2","B1","B2","pvalues"
"Species_A",190,210,290,310,0.0194193243090798
"Species_B",390,410,590,610,0.00496280979001086
"Species_C",110,90,120,80,1
"Species_D",400,90,600,200,0.605900105630468
```

Results

Try saving the results as a csv file and opening the file in Excel. You see something as follows:



	A	B	C	D	E	F
1		A1	A2	B1	B2	pvalues
2	Species_A	190	210	290	310	0.01941932
3	Species_B	390	410	590	610	0.00496281
4	Species_C	110	90	120	80	1
5	Species_D	400	90	600	200	0.60590011

Results

- The results show that:
    - The incidence of species *b* is significantly different at 0.01 level
    - The incidence of species *a* is not significant at 0.01 level
    - The incidence of Species *c* and *d* have no significant difference. For species *d* that is because even though an increasing trend is observed, the variation within each category is too high to allow any conclusions.

Output of images from Rshell

- The results from R scripts may be in the form of images such as a plot or a graph
- These images can be output from an Rshell service

Output of images from Rshell

- Add another Rshell processor onto the current workflow from the service template folder
- Provide the processor with the R script below:

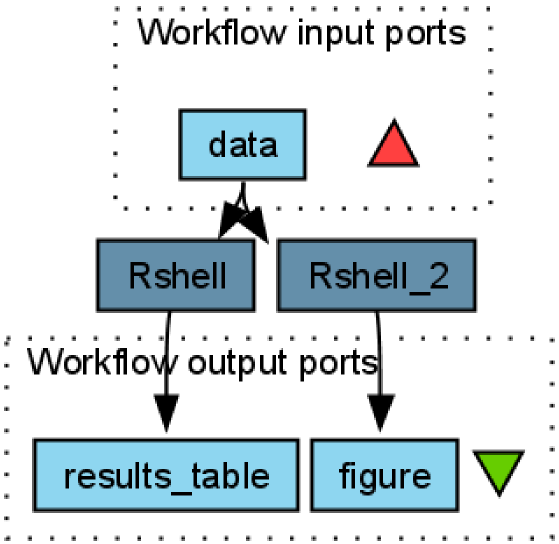
Output of images from Rshell

```
#Read in data
incdata <- read.table(file="data",head=TRUE,sep=",");
#Transpose
t <- t(incdata);
#Calculate means
mean_a <- mean(t[,"Species_A"]);
mean_b <- mean(t[,"Species_B"]);
mean_c <- mean(t[,"Species_C"]);
mean_d <- mean(t[,"Species_D"]);
#Combine data
means <- c(mean_a, mean_b, mean_c, mean_d);
#Transform to data frame
means <- data.frame(means, row.names = c("Species_A", "Species_B",
"Species_C", "Species_D"));
png(filename=figure, height=400, width=400, bg="white");
#Plot
barplot(t(means[1]), main = "mean species incidence levels", xlab =
"Species");
dev.off();
```

Output of images from Rshell

- Complete the configuration for this Rshell processor by:
  - Creating an input port called **data** and associating it with a Text-file data type
  - Creating an output port called **figure** and associating it with a png-file data type
- Also, finish building the workflow by connecting a workflow output to the figure output port of the Rshell processor
- Your workflow will now look as follows:

Output of images from Rshell



Output of images from Rshell

- Now run the workflow. You should get the following results:

- The result is an image showing a bar plot of the mean incidence levels of the four species