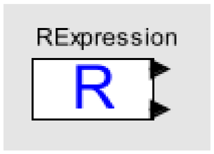
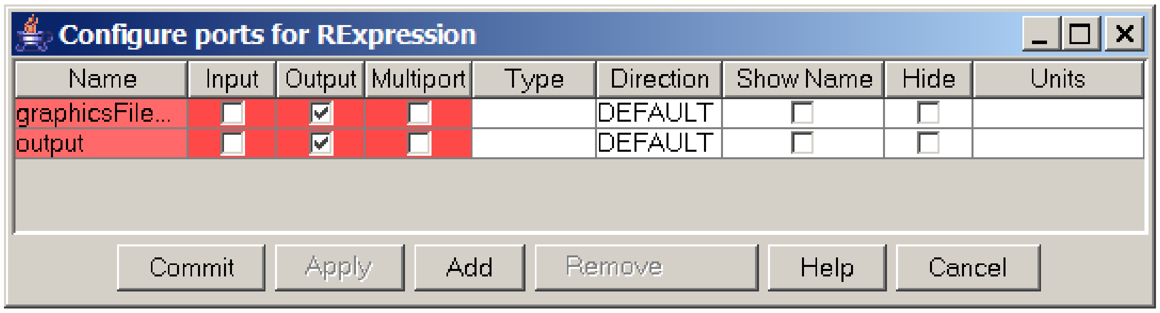
Dan Higgins – April 2005
DRAFT
RExpression Actor

Introduction

The ‘RExpression’ actor has been created for inserting R commands and scripts into Kepler workflows. This actor makes it easy to integrate the powerful data manipulation and statistical functions of R into a workflows. The actor can be found under the Mathematical Expressions node in the actor tree; its icon when dragged to the graph window is shown below. Note that initially there are only two ports, both outputs, called ‘graphicsFileName’ and ‘output’.



RExpression Actor



Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
graphicsFile...	☐	✓	☐		DEFAULT	☐	☐	
output	☐	✓	☐		DEFAULT	☐	☐	

Initial Port Configuration for RExpression actor

This actor is designed to work much like the Ptolemy/Kepler ‘Expression’ actor. With the Expression actor, one adds input ports and the names of these input ports can be used in a mathematical expression, the value of which is sent to the output port. One also adds input ports to the RExpression actor, and the name of those ports become named R objects which can be used in the R script.

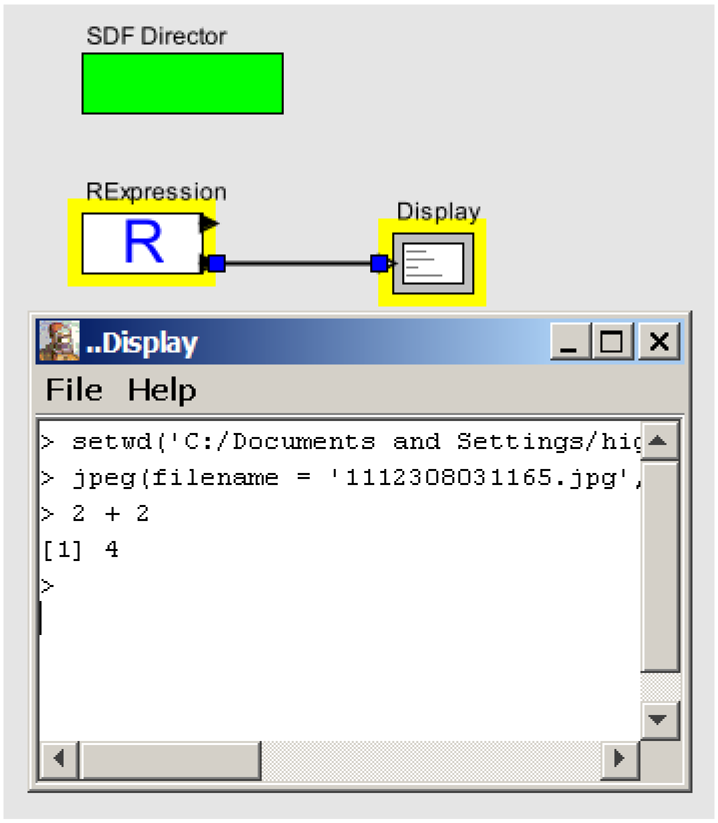
As an example, consider the simple workflow shown below where a SDF director is added and the RExpression actor’s output port has been connected to a Display actor. If you ‘Configure’ a RExpression actor, you can see that the default R script is just

```
2 + 2
```

Skipping the first few lines for the moment, one sees this command followed by the result; i.e.

```
> 2 + 2
[1] 4
```

This is exactly what would appear if one was running the R system from the command line.
[The first two lines, with ‘setwd…’ and ‘jpeg…’ are setup commands for R automatically added by the actor.]

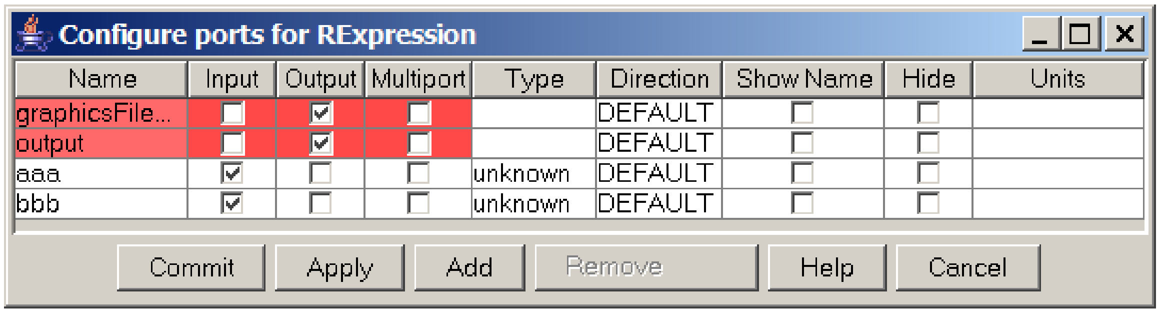


```
> setwd('C:/Documents and Settings/hig
> jpeg(filename = '1112308031165.jpg',
> 2 + 2
[1] 4
>
```

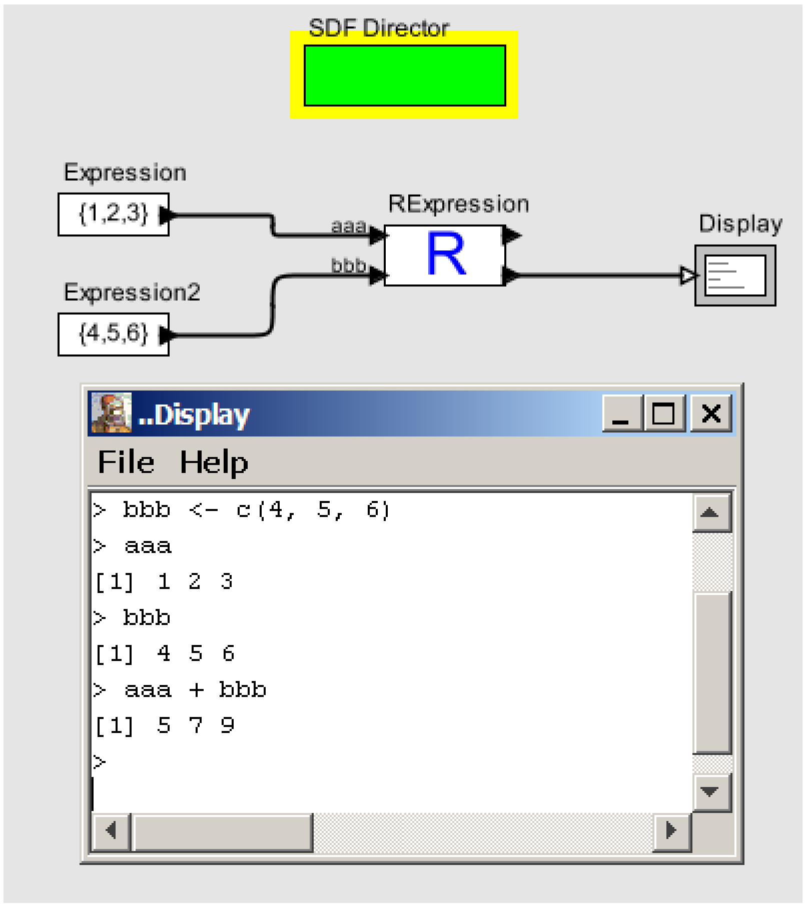
Now add 2 input ports as indicated below with the names ‘aaa’ and ‘bbb’ and set the R script to:

```
aaa
bbb
aaa + bbb
```

If `aaa` and `bbb` were simple scalars, we would have simply duplicated the functionality of the Expression actor. But the base data type of the R system is the ‘vector’ (which corresponds to a Kepler ‘array’). Thus consider the example workflow below where 2 expression actors have been connected to the two input ports, `aaa` and `bbb`. It can be seen the the result is an array with corresponding elements added together.



Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
graphicsFile...	☐	✓	☐		DEFAULT	☐	☐	
output	☐	✓	☐		DEFAULT	☐	☐	
aaa	✓	☐	☐	unknown	DEFAULT	☐	☐	
bbb	✓	☐	☐	unknown	DEFAULT	☐	☐	

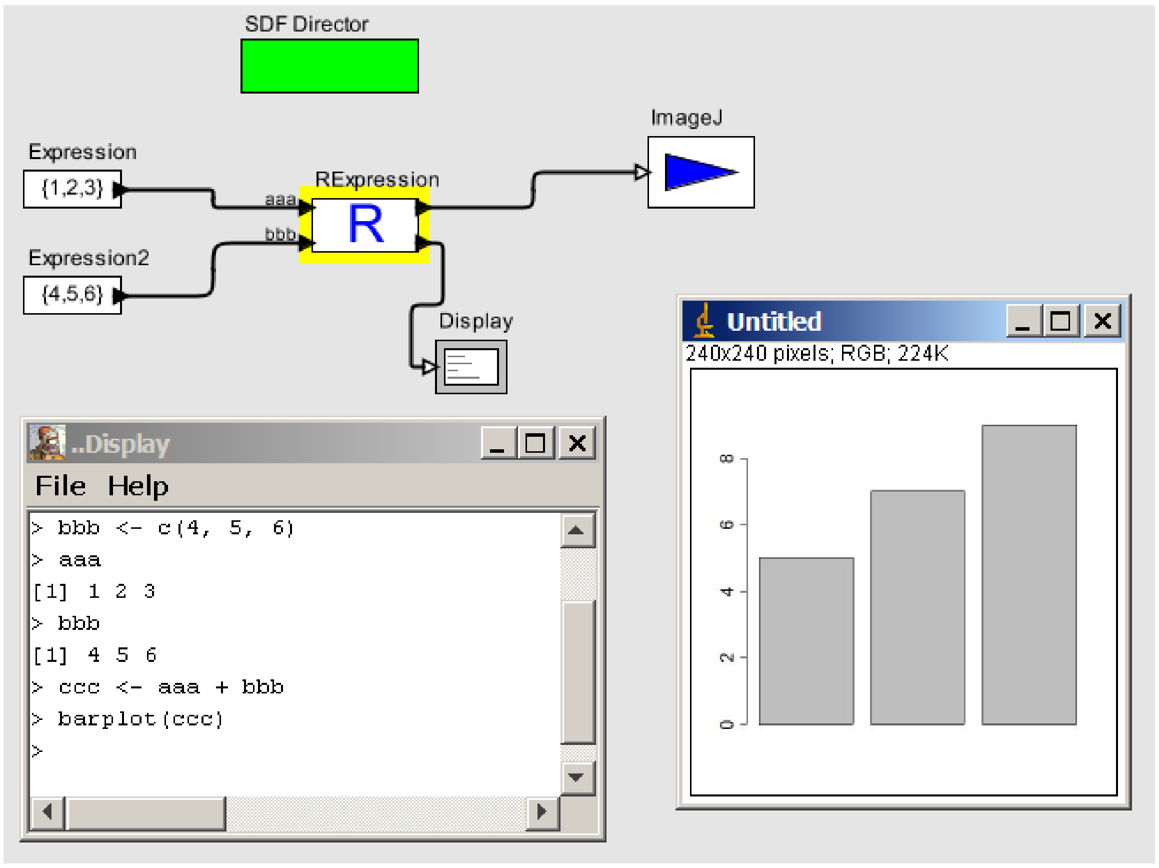


```
> bbb <- c(4, 5, 6)
> aaa
[1] 1 2 3
> bbb
[1] 4 5 6
> aaa + bbb
[1] 5 7 9
>
```

Now change the RScript to

```
aaa
bbb
ccc <- aaa + bbb
barplot(ccc)
```

and connect the graphicsOutputFileName port to an ImageJ graphics display and you can see a barplot created by R from the input data (see below).

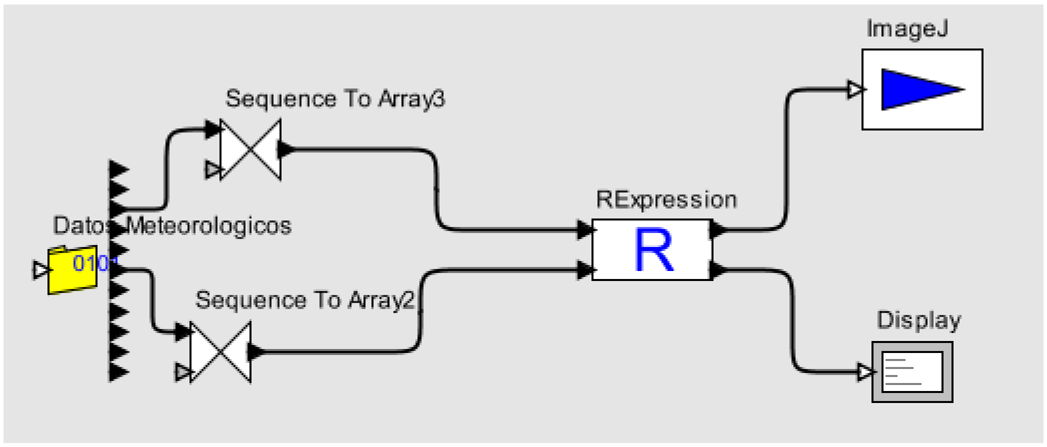


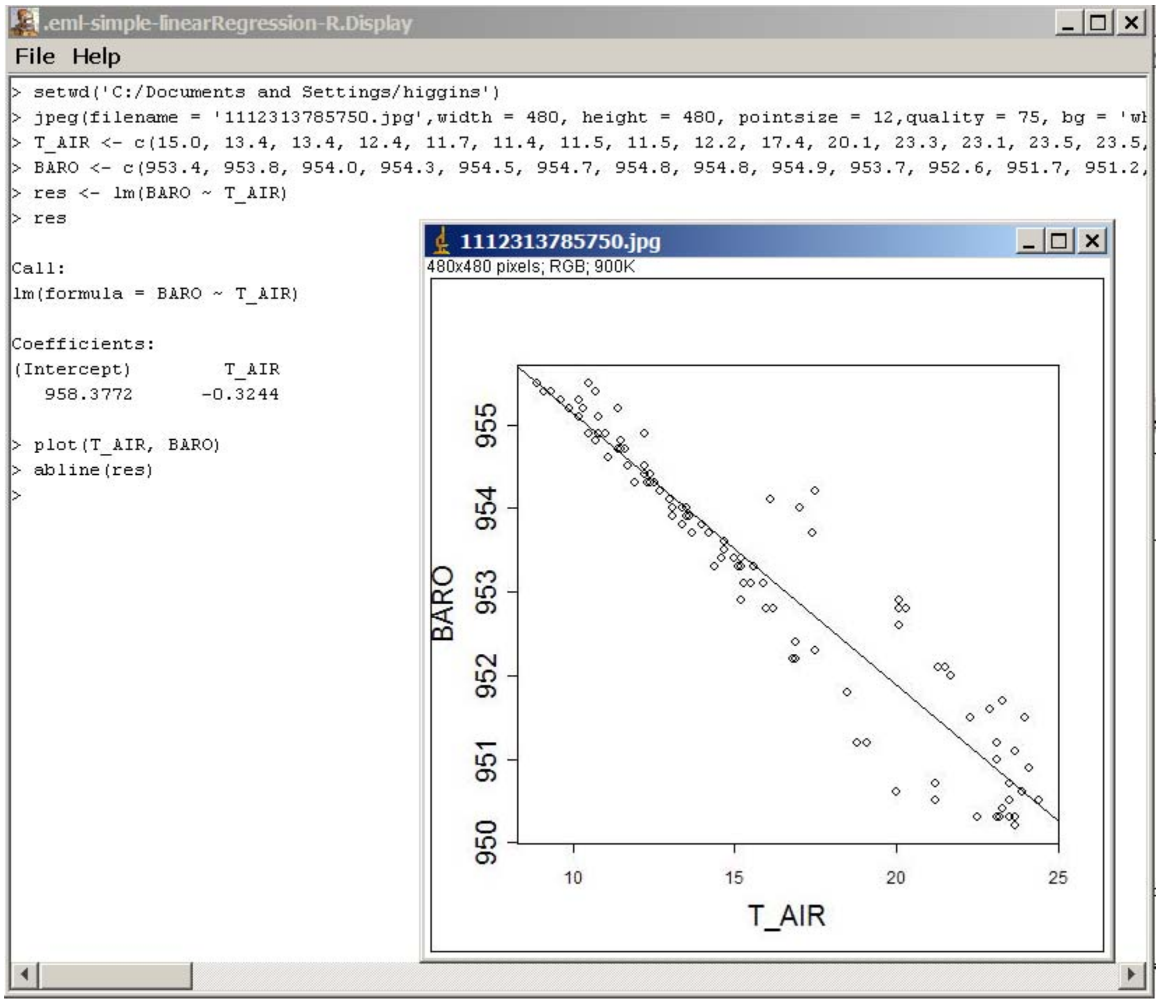
```
> bbb <- c(4, 5, 6)
> aaa
[1] 1 2 3
> bbb
[1] 4 5 6
> ccc <- aaa + bbb
> barplot(ccc)
>
```

A somewhat more interesting example is shown in the workflow below (a simple linear regression model). In this case, the input data comes from 2 output ports of the 'Datos Meterologicos' eml2DataSource actor. SequenceToArray actors are used to convert the sequence of values to arrays. [Note that these converters are used because otherwise one would need to tell the RExpression actor the length of each input sequence.] The RExpression script is

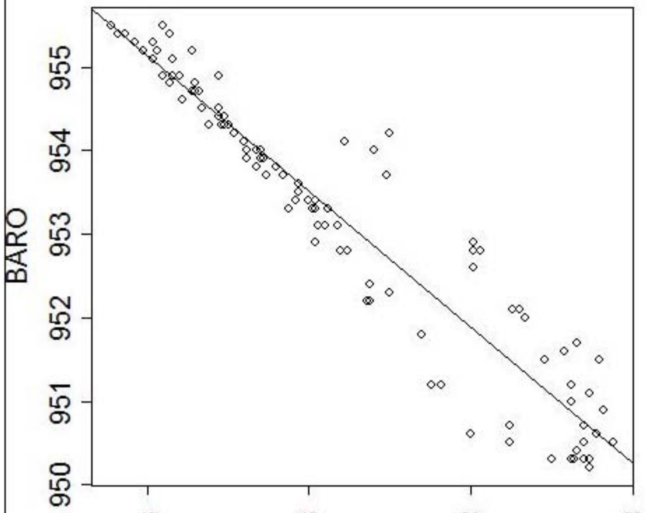
```
res <- lm(BARO ~ T_AIR)
res
plot(T_AIR, BARO)
abline(res)
```

The resulting text and graphic output are also shown below:




.eml-simple-linearRegression-R.Display
File Help
1112313785750.jpg
480x480 pixels; RGB; 900K
BARO
T_AIR

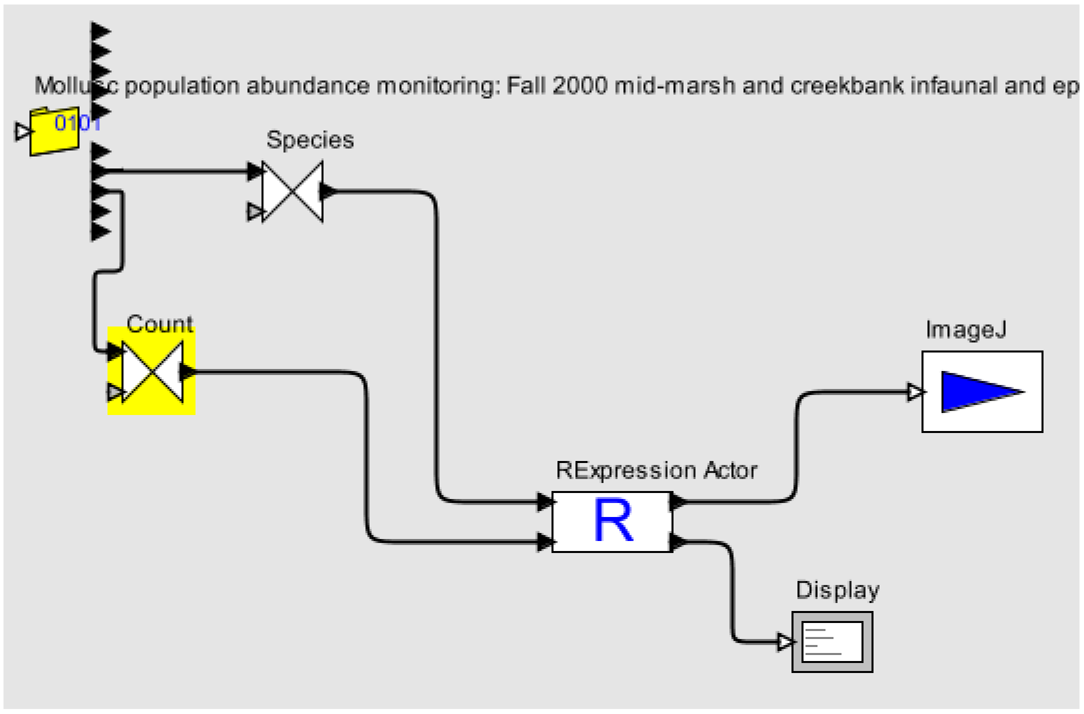
```
> setwd('C:/Documents and Settings/higgins')
> jpeg(filename = '1112313785750.jpg',width = 480, height = 480, pointsize = 12,quality = 75, bg = 'wh
> T_AIR <- c(15.0, 13.4, 13.4, 12.4, 11.7, 11.4, 11.5, 11.5, 12.2, 17.4, 20.1, 23.3, 23.1, 23.5, 23.5,
> BARO <- c(953.4, 953.8, 954.0, 954.3, 954.5, 954.7, 954.8, 954.8, 954.9, 953.7, 952.6, 951.7, 951.2,
> res <- lm(BARO ~ T_AIR)
> res

Call:
lm(formula = BARO ~ T_AIR)

Coefficients:
(Intercept)        T_AIR
   958.3772      -0.3244

> plot(T_AIR, BARO)
> abline(res)
>
```

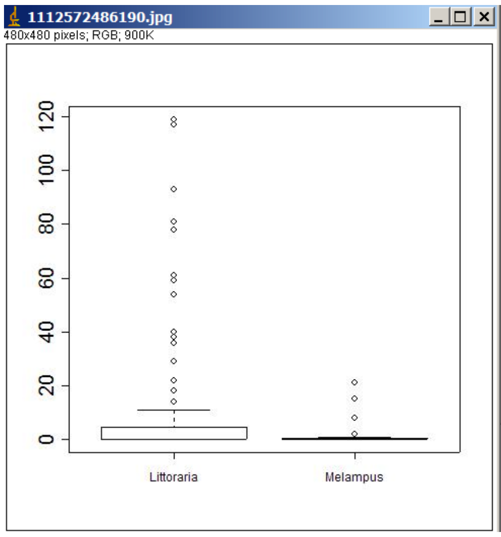


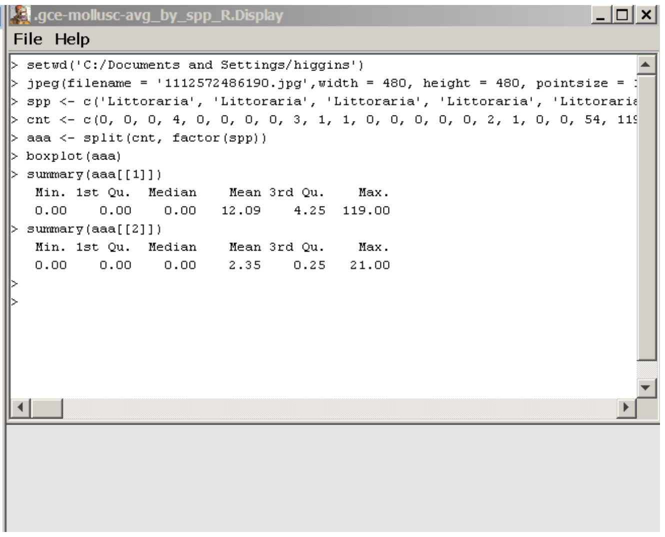
Previous examples have all used numerical vectors as input. One can also input string vectors as illustrated below where one input is a species list and the other is the count (i.e. the number of individuals of the species found. In this case, the R script is

```
aaa <- split(cnt, factor(spp))
boxplot(aaa)
summary(aaa[[1]])
summary(aaa[[2]])
```

Here the ‘factor’ and ‘split’ functions are used to summarize the results for each of the two species.


Mollusc population abundance monitoring: Fall 2000 mid-marsh and creekbank infaunal and ep
0101
Species
Count
ImageJ
RExpression Actor
R
Display



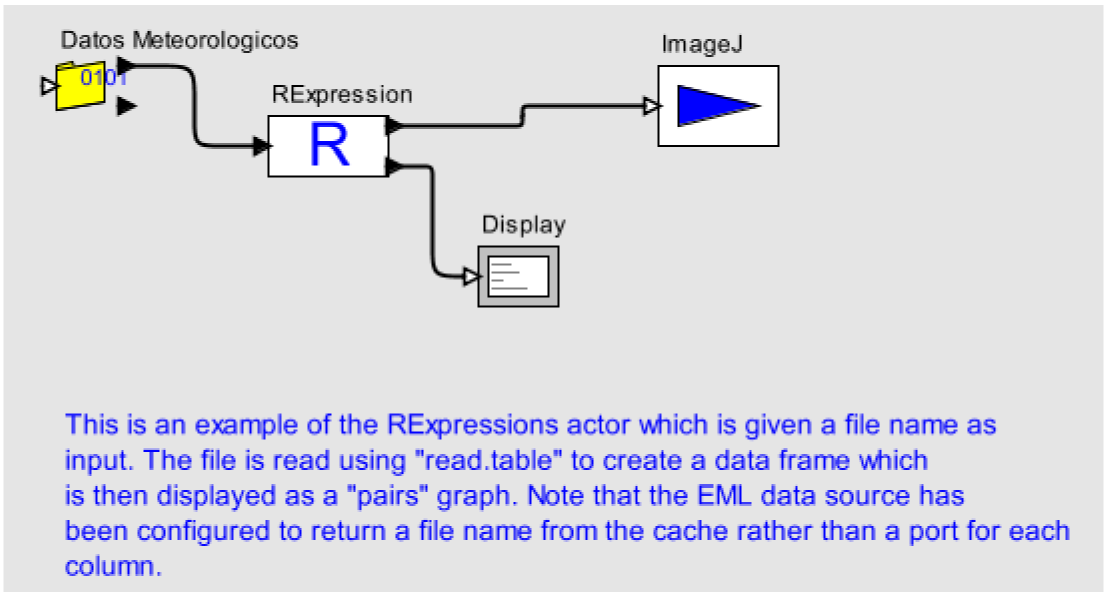


```
> setwd('C:/Documents and Settings/higgins')
> jpeg(filename = '1112572486190.jpg',width = 480, height = 480, pointsize = 
> spp <- c('Littoraria', 'Littoraria', 'Littoraria', 'Littoraria', 'Littorari
> cnt <- c(0, 0, 0, 4, 0, 0, 0, 0, 3, 1, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 54, 11
> aaa <- split(cnt, factor(spp))
> boxplot(aaa)
> summary(aaa[[1]])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    0.00   12.09    4.25  119.00
> summary(aaa[[2]])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    0.00    2.35    0.25   21.00
>
>
```

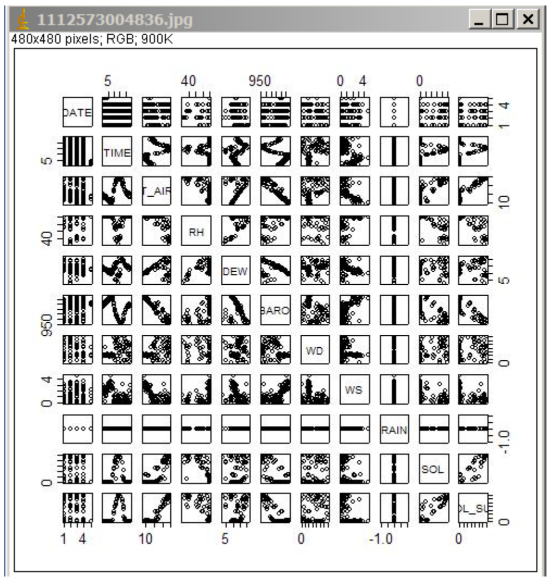
Inputs can also be simple strings or numbers. For example, one can reconfigure the EML2DataSource to give a data table file name rather than a set of ports that give sequences of column values. By doing this, the entire datafile can be passed to the R engine and read as an R data frame. In this case, the R script is

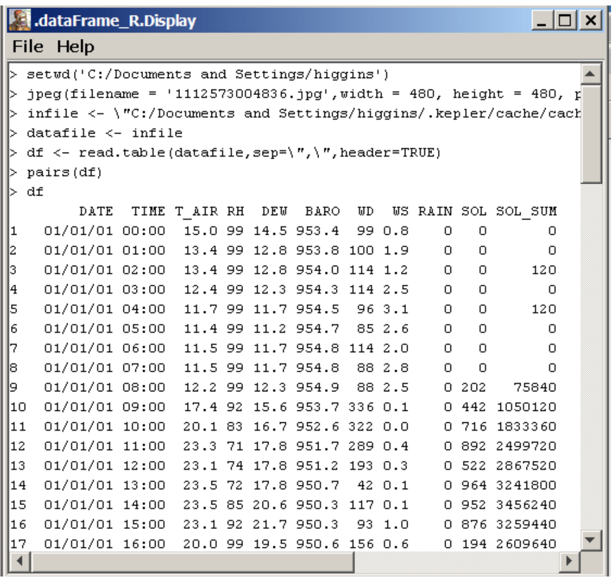
```
datafile <- infile
df <- read.table(datafile,sep=",",header=TRUE)
pairs(df)
df
```

where `infile` is the name of the single input port.



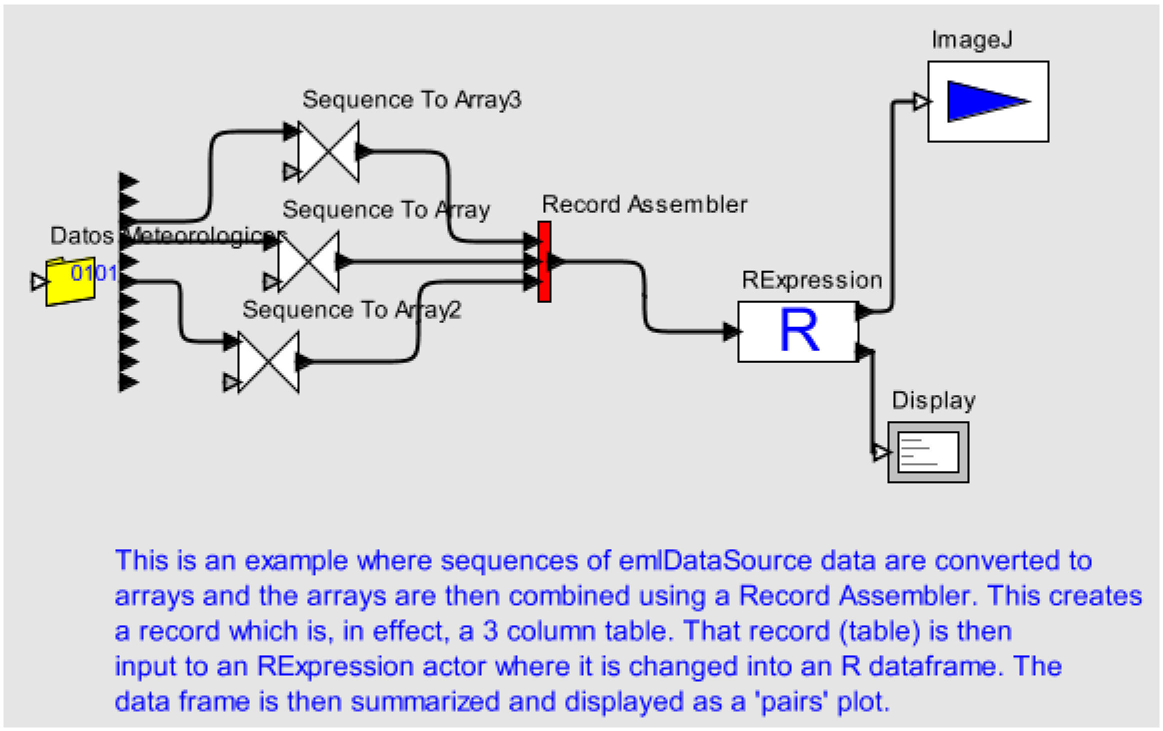
This is an example of the RExpressions actor which is given a file name as input. The file is read using "read.table" to create a data frame which is then displayed as a "pairs" graph. Note that the EML data source has been configured to return a file name from the cache rather than a port for each column.





```
> setwd('C:/Documents and Settings/higgins')
> jpeg(filename = '1112573004836.jpg',width = 480, height = 480, p
> infile <- \"C:/Documents and Settings/higgins/.kepler/cache/cach
> datafile <- infile
> df <- read.table(datafile,sep=\",\",header=TRUE)
> pairs(df)
> df
       DATE  TIME T_AIR RH  DEW  BARO  WD  WS RAIN SOL SOL_SUM
1  01/01/01 00:00  15.0 99 14.5 953.4  99 0.8    0   0       0
2  01/01/01 01:00  13.4 99 12.8 953.8 100 1.9    0   0       0
3  01/01/01 02:00  13.4 99 12.8 954.0 114 1.2    0   0     120
4  01/01/01 03:00  12.4 99 12.3 954.3 114 2.5    0   0       0
5  01/01/01 04:00  11.7 99 11.7 954.5  96 3.1    0   0     120
6  01/01/01 05:00  11.4 99 11.2 954.7  85 2.6    0   0       0
7  01/01/01 06:00  11.5 99 11.7 954.8 114 2.0    0   0       0
8  01/01/01 07:00  11.5 99 11.7 954.8  88 2.8    0   0       0
9  01/01/01 08:00  12.2 99 12.3 954.9  88 2.5    0 202   75840
10 01/01/01 09:00  17.4 92 15.6 953.7 336 0.1    0 442 1050120
11 01/01/01 10:00  20.1 83 16.7 952.6 322 0.0    0 716 1833360
12 01/01/01 11:00  23.3 71 17.8 951.7 289 0.4    0 892 2499720
13 01/01/01 12:00  23.1 74 17.8 951.2 193 0.3    0 522 2867520
14 01/01/01 13:00  23.5 72 17.8 950.7  42 0.1    0 964 3241800
15 01/01/01 14:00  23.5 85 20.6 950.3 117 0.1    0 952 3456240
16 01/01/01 15:00  23.1 92 21.7 950.3  93 1.0    0 876 3259440
17 01/01/01 16:00  20.0 99 19.5 950.6 156 0.6    0 194 2609640
```

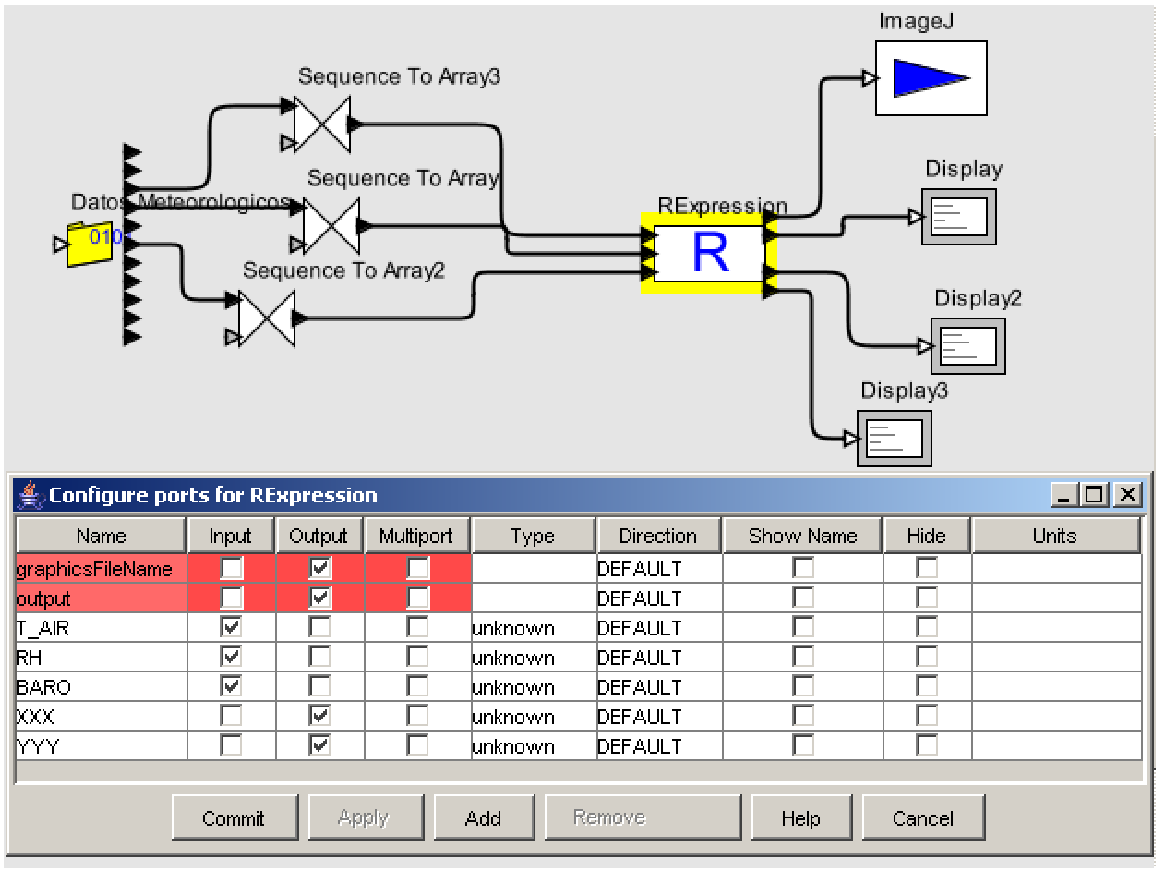
The RExpression actor can also be given a Kepler record token as an input as illustrated below. [In this case, a record is created using the Record Assembler actor from a set of three SequenceToArray actors. Ideally, the data source would create a Record directly.] It is assumed here that the record is a collection of named arrays representing the columns of a table. All the arrays need be the same length (but not the same data type). [If the arrays are not the same length, the input is ignored.]



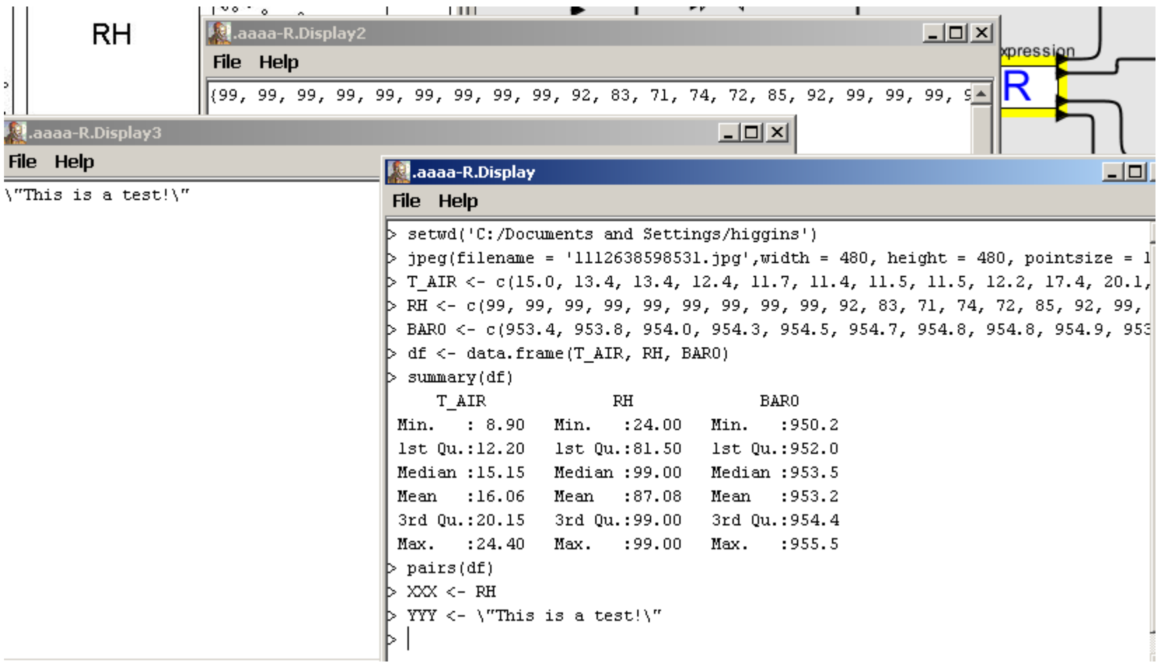
This is an example where sequences of emlDataSource data are converted to arrays and the arrays are then combined using a Record Assembler. This creates a record which is, in effect, a 3 column table. That record (table) is then input to an RExpression actor where it is changed into an R dataframe. The data frame is then summarized and displayed as a 'pairs' plot.

One can also add output ports to the Rexpression actor. Just give the output port a name that corresponds to an R variable. Consider the R script below and the workflow that follows. An array is output to the `XXX` port and a string to `YYY` port.

```
df <- data.frame(T_AIR, RH, BARO)
summary(df)
pairs(df)
XXX <- RH
YYY <- "This is a test!"
```



Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
graphicsFileName	☐	✓	☐		DEFAULT	☐	☐	
output	☐	✓	☐		DEFAULT	☐	☐	
T_AIR	✓	☐	☐	unknown	DEFAULT	☐	☐	
RH	✓	☐	☐	unknown	DEFAULT	☐	☐	
BARO	✓	☐	☐	unknown	DEFAULT	☐	☐	
XXX	☐	✓	☐	unknown	DEFAULT	☐	☐	
YYY	☐	✓	☐	unknown	DEFAULT	☐	☐	



```
{99, 99, 99, 99, 99, 99, 99, 99, 99, 92, 83, 71, 74, 72, 85, 92, 99, 99, 99, 9
```

```
\"This is a test!\"
```

```
> setwd('C:/Documents and Settings/higgins')
> jpeg(filename = '1112638598531.jpg',width = 480, height = 480, pointsize = 1
> T_AIR <- c(15.0, 13.4, 13.4, 12.4, 11.7, 11.4, 11.5, 11.5, 12.2, 17.4, 20.1,
> RH <- c(99, 99, 99, 99, 99, 99, 99, 99, 99, 92, 83, 71, 74, 72, 85, 92, 99,
> BARO <- c(953.4, 953.8, 954.0, 954.3, 954.5, 954.7, 954.8, 954.8, 954.9, 95
> df <- data.frame(T_AIR, RH, BARO)
> summary(df)
     T_AIR              RH              BARO
 Min.   : 8.90   Min.   :24.00   Min.   :950.2
 1st Qu.:12.20   1st Qu.:81.50   1st Qu.:952.0
 Median :15.15   Median :99.00   Median :953.5
 Mean   :16.06   Mean   :87.08   Mean   :953.2
 3rd Qu.:20.15   3rd Qu.:99.00   3rd Qu.:954.4
 Max.   :24.40   Max.   :99.00   Max.   :955.5
> pairs(df)
> XXX <- RH
> YYY <- \"This is a test!\"
> |
```

Some Technical Details

Note that when ports are added to the RExpression actor, there is no need to set the port types. Input tokens are examined by the actor to automatically determine the type (e.g. strings, doubles, integers, or arrays of strings, doubles, integers, or records).

R is called as a subprocess of Kepler with the input and output streams linked to the Kepler ports. Java code is used to convert the Kepler input tokens to text that is understandable by the R system (i.e R commands) and R objects are converted to text for output.

The RExpression actor generates some R code automatically. Some of this is inserted ahead of the script commands inserted by the user and some is inserted after the user commands.

Basically, the commands inserted before the user's R code just set up a jpeg graphics device so that images can be displayed and are saved as files. A unique file name, like `1112804966625.jpg` (based on the system time) is generated each time the actor is fired and saved in the user's home directory. The R '`setwd()`' function is also used to set the home directory as R's working directory so that R can use the file without having to include the entire file path.

Note that the user's R script can always override these settings since those commands are sent after these initial ones. If the user wants a different graphics device, he/she can simply add the function to the script. (And any subsequent graphics function always uses the last graphics device.) And R's working directory can also be reset. Data files can also be read from anywhere on the system by simply putting the absolute path into the script.

There are also some additional R commands added after the user's commands if additional output ports have been added. (These commands are trimmed from the R output and thus not displayed.) These commands use the R `dput()` function to output results for output ports that can be converted to Kepler tokens.