PI Machine Learning How To

earn how to actually use PI Machine Learning within Celonis.

Overview over R-Support

R can currently be used to calculate additional columns for use in table and plotting components. There are two different functions for interacting with R: **RCALL** and **RAGG**.

RCALL executes a user defined R-function for a given set of columns (which can be pre-aggregated by the user using PQL commands). RCALL should return exactly one value for each line of input.

In contrast, **RAGG** operates on groups and will return one value for each group of data. **RAGG** operates similar to PQL aggregation commands like AVG.

RCALL

The syntax of an RCALL statement is simply:

```
RCALL(COLUMN1 AS VAR1, COLUMN2 AS VAR2, ..., 'StringWithRCommand')
```

so for example:

```
RCALL("MYTABLE"."MYCOLUMN" AS SOMEDATA, "MYTABLE2"."MYCOLUMN7" AS MOREDATA, 'f <- SOMEDATA + MOREDATA')
```

RCALL can take an arbitrary number of arguments. The first n-1 arguments to RCALL are named columns from your data model.
The last argument is an R script, that has access to the previously defined columns via their names defined in the AS part. The syntax of an RAGG call is similar.

You can use PQL queries inside of RCALL statements. The data you will recieve will depend on which columns you define (similar to TABLE statements).

For example:

```
RCALL("MYTABLE"."MYCOLUMN" AS GROUP, SUM("MYTABLE2"."MYCOLUMN7") AS MOREDATA, SUM("MYTABLE2"."
MYCOLUMN7") AS SOMEDATA, 'f <- SOMEDATA + MOREDATA')
```

With this example you should recieve aggregated values for MOREDATA and SOMEDATA.

You can load arbitrary libraries within your R program, as long as they are installed on the server hosting Rserve.
Simply preface your script with regular library(PACKAGENAME) calls.

Difference between RCALL and RAGG

In the most simple example of an RCALL statement the provided column is just returned immediately:

```
TABLE("EKPO"."BELNR", RCALL("EKPO"."BELNR" AS MYFANCYVAR, 'rvar <- MYFANCYVAR'))
```

Here 'rvar <- MYFANCYVAR' is a very simple R program, in which the contents of the column "EKPO"."BELNR" are assigned to the variable rvar and the contents of rvar are then returned.
In this case, switching from RCALL to RAGG would deliver the same result. However, using RAGG here will result in a significant performance penality, because it would be executed for each entry in BELNR (each entry being treated as its own group).

The classic example for RAGG would be a call to the mean function:

```
TABLE("skewness.csv"."Group", RAGG("skewness.csv"."Value" AS VAL, 'mean(VAL)'))
```

Here we calculate the mean value of the column "skewness.csv"."Value" for each distinct "skewness.csv"."Group" entry.

RCALL examples

K-means clustering

A very simple example of clustering is shown in the app "Kmeans". In the kmeans dataset there are 4 obvious clusters in the data.
These clusters are easy to find visually, but in larger, multi-dimensional datasets an automatic method becomes necessary. K-means is one of the most straightforward methods for determining clusters.

In the Kmeans column we show an example R Call to perform Kmeans clustering. The syntax is:

```
RCALL("Kmeans.csv"."x" AS "X", "Kmeans.csv"."y" AS "Y", 'df <- data.frame(v1=X, v2=Y); k <- kmeans(df, 4);
k$cluster')
```

Here k$cluster is the vector with group numbers that is being returned.

Note that in the Kmeans method starts from different starting point each time it is invoked. Hence you might end up with a different cluster numbering each time the method is invoked.

Regression

Real data is often noisy, but we might be able to describe it by some simple theoretical model. For example, we might want to ascertain the trend in some dataset by fitting some known function to it.

A very simple example of such a fit is shown in the app "Regression". The underlying dataset was generated by adding gaussian noise to a simple polynomial.

```
RCALL("Regression.csv"."x" AS X, "Regression.csv"."y" AS "Y", ' model <- lm(Y ~ X + I(X^2) + I(X^3)); fitted
(model)')
```

RAGG examples

Skewness

Skewness is a measure of the assymmetry of a distribution. Positive skewness indicates that the distribution is skewed towards the right, negative skewness that it is skewed towards the left.

We can calculate the Skewness using the RAGG aggregation opertor:

```
RAGG("skewness.csv"."Value" AS VAL, 'library(e1071); skewness(VAL)')
```

Note that we load a library here, which needs to be installed on the server hosting Rserve.

Further Informations

Writing efficient R-Code

It's almost never a good idea to write for loops in R. Most functions work on vectors and are significantly faster that way.

File Access

If the user running the R interpreter has file access somewhere, you can save and load R-objects from that directory. This is especially useful if you have long-running scripts that you only need to perform once.
In such cases it is often better to store the result on disk after performing the calculation once, so that it can simply be loaded back in subsequent calls.

```
x <- c(1:10)
save(x, file="someFileName")
```

Here we saved the vector containing the numbers from 1 to 10 in a file called "someFileName". We can later load the vector in again simply via:

```
load("someFileName")
```

Note any other assignment to the variable x will be overwritten by the load statement.