Add a new distribution to the GAS package

This short note illustrates how to add a new distribution in the **GAS** package. Changes have to be made in the `.R` files inside the R folder and in the `.cpp` and `.h` files inside the src folder of the package tarball, respectively.
The following guidelines apply when:

1 You want to add an univariate distribution.

2 Your distribution has maximum 5 parameters.

If you want to add a multivariate distribution, most of the following points apply as well. However, additional changes have to be included in order to have a proper implementation. You can email the package maintainer for possible inclusion of your multivariate distribution in the package.
Changes with regards to R files are:

1 `Distribution.R` : `DistLabels()`, `DistName()`, `DistNote()`, `DistReference()`, `DistParameters()`, `DistType()`, `DistScalingType()`.

2 `ParameterConstraints.R`: include the possible constraints to the parameters dynamic. Look at the function `GetFixedPar_Uni()` and `GetFixedPar_Multi()` for univariate and multivariate distributions, respectively.

3 `ParNames.R` : `FullNamesUni()`. Choose the labels for the parameters of the distribution between : `"location"`, `"scale"`, `"skewness"`, `"shape"` and `"shape2"`. These are only names, *i.e.*, the label `"location"` does not necessarily represent a location parameter.

4 `StartingValue.R` : `StaticStarting_Uni()`. Use Method of Moments when possible. If you want to add a multivariate distribution, you need to create a new function `StartingValues_dist()` and link to the `MultiGAS_Starting()` function. Note that, in the multivariate case, the output of the new `StartingValues_dist()`, needs to be consistent with that of, for example, of `StartingValues_mvt()`.

Assuming that the label of the new distribution is `"poi"`. Changes with regards to the C++ files are:

1 Create `poi.cpp` with the functions:

- `double dPOI()` , probability density function : it accepts the additional `boolean` argument: `bLog`, by default `bLog = false`.

- `double pPOI()`, cumulative density function.
- `double qPOI()`, quantile function.
- `double rPOI()`, random generator.
- `arma::vec mPOI()`, it returns an `arma::vec` with 4 entries containing: the mean, the variance, the skewness coefficient, the kurtosis coefficient (not excess of kurtosis).
- `arma::vec poi_Score()`, score function.
- `arma::mat poi_IM()`, information matrix. If the information matrix is not available, then the output of `arma::mat poi_IM()` is the identity matrix of dimension equal to the number of parameters.

All the function of poi.cpp accept `double` arguments, expect `poi_Score()` and `poi_IM()` that accept an `arma::vec` argument. See, for example, the file `std.cpp` .

2 Create the header `poi.h` with all the functions in `poi.cpp` .

3 Modify `DistWrap.cpp` to include `poi.h`, add `if (Dist == "poi")` to all the functions.

4 Modify `IMWrap.cpp` to include `poi.h`, add `if (Dist == "poi")` to `arma::mat IM_univ()`.

5 Modify `Mapping.cpp`, the functions : `MapParameters_univ()`/`MapParameters_multi()`, `UnmapParameters_univ()`/`UnmapParameters_multi()`,`MapParametersJacobian_univ()`/ `MapParametersJacobian_multi()`. Use the same structure of the other distributions. If necessary, create global variables as `const double dLowerShape` and `const double dUpperShape` to determine the numerical upper and lower bounds for your parameters. You can use the modified logistic mapping function transformation (`double Map()`), its inverse (`double Unmap()`) and its derivative (`double MapDeriv()`).

6 Modify `ScoreWrap.cpp` to include `poi.h` : add `if (Dist == "poi")` to `arma::vec Score_univ()`/`arma::vec Score_multi()`.

7 Modify `Utils.cpp`, functions : `int NumberParameters()` which returns an integer indicating the number of parameters of your distribution.

If everything have been done correctly you can now compile the package through `R CMD BUILD` and use all the package functionalities.