Plan

Overview

The R package *scR* translates R code to C. Whereby, *scR* is an abbreviation for static compiled R.

Types

The following types are avaialble in R. As in R no scalar character variables can be defined. Types are specified using the %type% infix operator. In case no type is defined the generic vector type is assumed. The generic type is a Union containing all types. Therefore, the generic type is the most flexible type but this comes at the cost of performance.

bool	int	num	chr	complex	generic
scalar	scalar	scalar	scalar	scalar	NA
vector	vector	vector	NA	vector	vector

from R to C

It is difficult to translate R code to C and keeping the structure of the original code. Within the R code variables are used. Instead of variables the C code uses a structure called *VectorManager*. This structure is used to store all variables independet of their type. This approach is required to free the memory of all variables in case an error is encountered.

Moreover, R is a vectorized language. This means that each operation is performed on the whole vector. To emulate this behavior in C for loops are used. To prevent naming conflicts with other variables and to ensure a good structure each expression is executed in a separate function.

Each of this function is created in R. However, the structure is always the same. The function takes as input a pointer to the *VectorManager*. Thereby, all variables can be accessed and modified.
Next, two int arrays are defined. The first array contains the indices of the variables that are used in the expression. The second array contains the types of the variables. Moreover, is the size of the two arrays saved in an int variable.
Using the two integer arrays and the size of them the length of the expression is calculated. Notably, the size corresponds to the largest variable contained in the right hand side.
Afterwards, a temporary variable is (re)allocated using the size as length. The temporary variable is used to store the result of the expression. Thereby, the variable on the left hand side is not altered as it potentially is used on the right hand side. Considering the expression `a[c(3, 2)] <- a[c(2, 3)] + 1` if one would directly assign the result to the left hand side than the value of `a[3]` would be altered before it is used on the right hand side.

Additionally an integer variable is defined containing the index value of the variable that is assigned.

Subsequently, the expression is evaluated. This is done by a for loop starting at 0 and ending at the size of the temporary object. Within the body of the loop the original structure of the expression is maintained. Each variable is replaced with the correct getter function for the corresponding type.

Finally, the result is assigned to the left hand side.