C++ is a compiled language, which means that the code is run directly by the CPU instead of carrying out checkings at run-time. This gives to C++ the property to be much faster than R, as it can be seen bellow, in the results of the present project.

In order to convert R code in C++, it is used the Rcpp function which allows to extend this language using the C++.

The main important thing when you convert code from R to C++ language is define the type of all variables at the beginning of the line. For instance, in order to define a variable with a decimal type, we should write: ***double*** *x = 5.3* in C++, instead of only: *x = 5.3*, in R.

Also, some commands might change like the way to defined the range in a "for" loop, or the definition of a position in a matrix.

On the other hand, this exercise has been done with a ***vectorized sugar function*** which allows to simplify the code by removing a "for" loop and operating directly over the vectors of the rows.

In addition, in order to include K=2 neighborhoods, it has been added another "for" loop which checks a second distance and returns the mean of these two values as the associated neighborhood.

Then, the step above has been done again, converting the code from R to C++, in order to compare the compilated time between both languages.

Finally, it has been repeated the coding in C++ for two neighborhoods including the inverse of the Euclidean distance which is equivalent to the next expression:

$$Inverse\ Euclidean\ distance = \frac{\frac{\text{value1}}{\text{distance 1}} + \frac{\text{value2}}{\text{distance2}}}{\frac{1}{\text{distance1}} + \frac{1}{\text{distance2}}}$$

By evaluating through the Benchmark library the functions created in R, and comparing them with the *Knn.reg function* and the C++ functions, which have been compilated using the sourceCpp function, we have determined that the *C++ code* is faster than the *R code* and the *Knn.reg function*.

The obtained results are gathered in the following table:

Numbers of Ks	Function	Time (mean)	Value
1	Knn_R	4.781,58	17,5000
	Knn.reg	563,21	17,5000
	Knn_C++	28,77	17,5000
2	Knn_R	5.193,14	17,8500
	Knn.reg	523,16	17,8500
	Knn_C++	47,43	17,8500
2	Knn_C++_inverse_Euclidean		17,7094

As it can be observed, all the values obtained for each "K" are the same, this has sense because the kind of language doesn't influence in the result of the coding. The times to execute the code are different and differ in a significant value that will increased with the complexity of the function, being C++ the fastest way, followed by the knn.reg function and then, by the R language which is the slowest one.