What rmr aims to be:

1. For map-reduce programmers:
    - the easiest, most productive, most elegant way to write map reduce jobs. One-two orders of magnitude less code than Java. Readable, reusable, extensible.
    - A great prototyping, executable spec and research language.
    - General, and not a crippled language trap. Any map reduce algorithm can be implemented with this package. With Alan Kay: "Simple things should be simple, complex things should be possible."
2. For R programmers:
    - a way to access the Map Reduce programming paradigm.
    - a way to work on big data sets in a way that is “natural” or “R-like”.
    - a way to access massive, fault tolerant parallelism without mastering distributed programming.
    - just a library to use, no run-time, no R patches, nothing

What rmr is not trying to be:

1. rmr is not Hadoop Streaming. It uses streaming for its implementation but it doesn’t aim to support every single option that streaming has. Streaming is accessible from R with no additional packages because R can execute an external program and R scripts can read stdin and stdout. The point is to provide an abstraction over that. rmr is not syntactic sugar for Hadoop Streaming.

2. Map reduce programs written in rmr are not going to be the most efficient. While aiming to reduce the gap over time to extend its applicability, it is unlikely to ever be the most effective way to implement massive production jobs.

3. rmr does not provide a map reduce version of any of the more than 3000 packages available for R. It does not solve the problem of parallel programming. You still have to write parallel algorithms for any problem you need to solve, but you can focus only on the interesting aspects. Some problem are believed not to be amenable to a parallel solution and using the map reduce paradigm or rmr does not create an exception.