(Just a couple of notes — not a complete doc by any means)

1. Explicit transfer from a to HDFS with `from.dfs` and `to.dfs` as opposed to implicit.
2. Impossible for mapreduce to make an educated guess on what the input means: is a list of character strings a list of HDFS paths or literally the input
3. It's going to be difficult to decide automatically whether to keep results in HDFS or bring them into memory and it's very unlikely we can make it transparent to the programmer, so `from.dfs` has to be a deliberate decision
4. We could have added options to `mapreduce` to achieve this, but they would have been extremely complicated. Imagine a multiple input job where some inputs come from memory and some from HDFS: not a common use case but a consequence of supporting both types of input, multuple inputs and having a goal of orthogonality

5. Finally we opted for a separation of concerns, `mapreduce` inputs and outputs are on HDFS; `from.dfs` and `to.dfs` do the transfer. This may change a bit with alternative backends (meaning that we could use a different file system) but the idea should stay the same.

6. Not a direct equivalent to lapply and tapply

7. We can easily simulate them but if we teach the user to use them as the basic primitives we end up with more jobs that are map-only or reduce-only and could be easily combined into simpler jobs. If we do that we also need to offer a "planner" feature, like Cascading, Hive etc have to merge some of the map only and reduce only jobs into a smaller number of more complicated jobs. This is a higer level interface and should be left to a separate package built on top of `rmr`, if there is demand for it

8. What to do with tuning parameters. Hadoop exposes lots of tuning parameters, too many. That means that it is fast unless you get the configuration wrong, which happens to many a times. Even worse, the appropriate tuning may depend on the specific algorithm, if it is more IO or CPU or memory bound we will want progressively fewer concurrent tasks on each node. We don't want to expose all that complexity in mapreduce to keep it simple and focused and with alternative backends it makes even less sense. Also rember Josh Block of netbeans fame warning on tuning parameters ("be suspicious!"). Right now there are no tuning parameters and one has to rely on global configuration. That's right for a 1.0 but unlikely to cut it in the long run and we need to take a hard look at this issue.