Record and replay debugging in R

Kryštof Slavík, supervised by Petr Máj

R programming language
- dynamically typed interpreted language
- mainly used in statistics

Record and replay debugging
- helps to identify cause of non-deterministic bugs
- user repeatedly runs a program while recording each execution until the bug appears
- then replays the recorded run multiple times with the bug always present

Principle of the implemented solution
- all non-deterministic behaviour in R happens inside the interpreter's implementation or in an external C code
- **record**: capture calls of these C functions and record their return values to a trace
- **replay**: load the recorded values and return them instead of running the C functions
- handle remaining special cases where the simple solution does not work

Evaluation
- implementation was tested on 565 practical examples (vignettes) from various packages
- each vignette was run in three modes
  - plain (without the debugger)
  - record (new run with recording)
  - replay (replayed the recorded run)
- successful test = all three outputs are the same

Performance impact
- performance impact of the debugger is negligible

Simple example
# random numbers differ:
> runif(1)
[1] 0.9117631
> runif(1)
[1] 0.7701264
# replayed are the same:
> rec <- record(runif(1))
> replay(rec)
[1] 0.880779
> replay(rec)
[1] 0.880779

User interface
- native R debugging tool can be used together with the following four additional functions
  - **record(expression)** runs the expression while recording all necessary info and returns a **replay structure**
  - **replay(rep_str)** deterministically re-runs the recorded expression using the given **replay structure**
  - **recordFindBug(expr, detect_func)** keeps recording the expression until a bug occurs, then returns a **replay structure**
  - **recordTrace(...)** allows the user to insert arbitrary debugging code into the recorded expression