Package 'R.matlab'

October 12, 2022

Version 3.7.0

Depends R (>= 2.14.0)

Imports methods, utils, R.methodsS3 (>= 1.7.1), R.oo (>= 1.23.0), R.utils (>= 2.5.0)

Suggests Matrix, SparseM

Title Read and Write MAT Files and Call MATLAB from Within R

Author Henrik Bengtsson [aut, cre, cph], Andy Jacobson [ctb] (Internal MAT v4 reader), Jason Riedy [ctb] (Support for reading compressed files, sparse matrices and UTF-encoded strings.)

Maintainer Henrik Bengtsson <henrikb@braju.com>

Description Methods readMat() and writeMat() for reading and writing MAT files. For user with MATLAB v6 or newer installed (either locally or on a remote host), the package also provides methods for controlling MATLAB (trademark) via R and sending and retrieving data between R and MATLAB.

License LGPL (>= 2.1)

LazyLoad TRUE

ByteCompile TRUE

URL https://github.com/HenrikBengtsson/R.matlab

BugReports https://github.com/HenrikBengtsson/R.matlab/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-25 21:52:34 UTC

R topics documented:

R.matlab-package 2
1. The MATLAB server running in MATLAB 5
Matlab 16
readMat 23
writeMat 26

Index **29**

`R.matlab-package` *Package R.matlab*

Description

Methods readMat() and writeMat() for reading and writing MAT files. For user with MATLAB v6 or newer installed (either locally or on a remote host), the package also provides methods for controlling MATLAB (trademark) via R and sending and retrieving data between R and MATLAB.

In brief, this package provides a one-directional interface from R to MATLAB, with communication taking place via a TCP/IP connection and with data transferred either through another connection or via the file system. On the MATLAB side, the TCP/IP connection is handled by a small Java add-on.

The methods for reading and writing MAT files are stable. The R to MATLAB interface, that is the Matlab class, is less prioritized and should be considered a beta version.

For package history, see `showHistory(R.matlab)`.

Requirements

This is a cross-platform package implemented in plain R. This package depends on the **R.oo** package [1].

To use the Matlab class or requesting verbose output messages, the R.utils package is loaded when needed (and therefore required in those cases).

The `readMat()` and `writeMat()` methods do *not* require a MATLAB installation neither do they depend on the `Matlab` class.

To connect to MATLAB, MATLAB v6 or higher is required. It does *not* work with MATLAB v5 or before (because those versions do not support Java). For confirmed MATLAB versions, see the `Matlab` class.

Installation

To install this package do

```
install.packages("R.matlab")
```

To get started

To get started, see:

1. `readMat`() and `writeMat`() - For reading and writing MAT files (MATLAB is *not* needed).
2. `Matlab` - To start MATLAB and communicate with it from R.

Miscellaneous

A related initiative is *RMatlab* by Duncan Temple Lang and Omegahat. It provides a bi-directional interface between the R and MATLAB languages. For more details, see https://www.omegahat.net/RMatlab/. To call R from MATLAB on Windows (only), see *MATLAB R-link* by Robert Henson available at the MATLAB Central File Exchange (https://www.mathworks.com/matlabcentral/fileexchange/5051-matlab-r-link).

How to cite this package

Whenever using this package, please cite as

Troubleshooting

In general:
For trouble shooting in general, rerun erroneous function with verbose/debug messages turned on. For `readMat()` and `writeMat()` see their help. For communication with a MATLAB server, use

```
matlab <- Matlab()
setVerbose(matlab, threshold = -2)
```

The lower the threshold is the more information you will see.

Cannot connect to MATLAB:
If R fails to connect to MATLAB, make sure to try the example in help(Matlab) first. Make sure that the MATLAB server is running before trying to connect to it from R first. If MATLAB is running but `open()` times out, make sure MATLAB is listening to the same port that R is trying to connect to. If that does not help, try to increase the time-out limit, see help(open.Matlab).

Expected an 'answer' from MATLAB, but kept receiving nothing.:
When launching a really long MATLAB process by `evaluate()`, you may get the above error message.
Reason: This happens because `evaluate()` expect a reply from MATLAB as soon as MATLAB is done. The waiting should be "blocked", i.e. it should wait until it receives something. For unknown reasons, this is not always happening. The workaround we have implemented is to try `readResult/maxTries` waiting `readResult/interval` seconds in-between.
Solution: Increase the total waiting time by setting the above options, e.g.

```
setOption(matlab, "readResult/interval", 10)     # Default is 1 second
setOption(matlab, "readResult/maxTries", 30 * (60 / 10)) # ~30 minutes
```

Wishlist

Here is a list of features that would be useful, but which I have too little time to add myself. Contributions are appreciated.

- Add a function, say, `Matlab$createShortcut()` which creates a Windows shortcut to start the MATLAB server by double clicking it. It should be possible to create it in the current directory or to the Desktop. Maybe it is possible to do this upon installation and even to a Start -> All Programs -> R menu.
- To improve security, update the MatlabServer.m script to allow the user to specify a "password" to be send upon connection from R in order for MATLAB to accept the connection. This password should be possible to specify from the command line when starting MATLAB. If not given, no password is required.
- Add additional methods to the Matlab class. For instance, inline function in MATLAB could have its own method.
- Wrap up common MATLAB commands as methods of the Matlab class, e.g. `who(matlab)`, `clear(matlab)` etc. Can this be done automatically using "reflection", so that required arguments are automatically detected?
- Add access to MATLAB variables via `"$"` and `"$<-"`, e.g. `matlab$A` and `matlab$A <- 1234`. Is this wanted? Maybe the same for functions, e.g. `matlab$dice(1000)`. Is it possible to return multiple return values?

If you consider implement some of the above, make sure it is not already implemented by downloading the latest "devel" version!

Acknowledgments

Thanks to the following people who contributed with valuable feedback, suggestions, code and more:

- Patrick Drechsler, Biocenter, University of Wuerzburg.
- Spencer Graves.
- Andy Jacobson, Atmospheric and Oceanic Sciences Program, Princeton University.
- Jason Riedy, Computer Science Division, University of California, Berkeley.
- Chris Sims, Department of Economics, Princeton University.
- Frank Stephen, National Renewable Energy Laboratory.
- Yichun Wei, Department of Biological Sciences, University of Southern California.
- Wang Yu, ECE Department, Iowa State University.

License

The releases of this package is licensed under LGPL version 2.1 or newer.

The development code of the packages is under a private license (where applicable) and patches sent to the author fall under the latter license, but will be, if incorporated, released under the "release" license above.

References

1 H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. https://www.r-project.org/conferences/DSC-2003/Proceedings/

Author(s)

Henrik Bengtsson

---

1. The MATLAB server running in MATLAB

1. The MATLAB server running in MATLAB

Description

This section gives addition details on the MATLAB server. At the end, the MatlabServer.m script and the InputStreamByteWrapper.java code is shown.

Starting the MATLAB server on Windows

Note that you "cannot prevent MATLAB from creating a window when starting on Windows systems, but you can force the window to be hidden, by using " the option -minimize. See https://www.mathworks.com/matlabcentral/answers/102082 for more information.

MatlabServer.m script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MatlabServer
%
% This scripts starts a minimalistic MATLAB "server".
%
% When started, the server listens for connections at port 9999 or the
% port number specified by the environment variable 'MATLABSERVER_PORT'.
%
% Troubleshooting: If not working out of the box, add this will to the
% MATLAB path. Make sure InputStreamByteWrapper.class is in the same
% directory as this file!
%
% Requirements:
% This requires MATLAB with Java support, i.e. MATLAB v6 or higher.
%
% Author: Henrik Bengtsson, 2002-2016
%
% References:
```

```
% [1] http://www.mathworks.com/access/helpdesk/help/techdoc/
%                                      matlab_external/ch_jav34.shtml#49439
% [2] http://staff.science.uva.nl/~horus/dox/horus2.0/user/
%                                                  html/n_installUnix.html
% [3] http://www.mathworks.com/access/helpdesk/help/toolbox/
%                                                modelsim/a1057689278b4.html
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf(2, 'Running MatlabServer v3.5.9-9000\n');

%  addpath R/R_LIBS/linux/library/R.matlab/misc/

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% MATLAB version-dependent setup
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Identify major version of Matlab
MatlabServer_tmp_hasMajor = eval('length(regexp(version, ''^[0-9]'')) ~= 0', '0');
if (MatlabServer_tmp_hasMajor)
  MatlabServer_tmp_verParts = sscanf(version, '%d.');
  MatlabServer_tmp_verMajor = MatlabServer_tmp_verParts(1);
  clear MatlabServer_tmp_verParts;
else
  MatlabServer_tmp_verMajor = -1;
end
clear MatlabServer_tmp_hasMajor;

if (MatlabServer_tmp_verMajor < 6)
  % Java is not available/supported
  error('MATLAB v5.x and below is not supported.');
elseif (MatlabServer_tmp_verMajor == 6)
  fprintf(2, 'MATLAB v6.x detected.\n');
  % Default save option
  MatlabServer_saveOption = '';
  % In MATLAB v6 only the static Java CLASSPATH is supported. It is
  % specified by a 'classpath.txt' file. The default one can be found
  % by which('classpath.txt'). If a 'classpath.txt' exists in the
  % current(!) directory (that MATLAB is started from), it *replaces*
  % the global one. Thus, it is not possible to add additional paths;
  % the global ones has to be copied to the local 'classpath.txt' file.
  %
  % To do the above automatically from R, does not seem to be an option.
else
  fprintf(2, 'MATLAB v7.x or higher detected.\n');
  % MATLAB v7 and above saves compressed files, which is not recognized
  % by R.matlab's readMat(); force saving in old format.
  MatlabServer_saveOption = '-V6';
  fprintf(2, 'Saving with option -V6.\n');

  % In MATLAB v7 and above both static and dynamic Java CLASSPATH:s exist.
```

```
  % Using dynamic ones, it is possible to add the file
  % InputStreamByteWrapper.class to CLASSPATH, given it is
  % in the same directory as this script.
  javaaddpath({fileparts(which('MatlabServer'))});
  fprintf(2, 'Added InputStreamByteWrapper to dynamic Java CLASSPATH.\n');
end
clear MatlabServer_tmp_verMajor;


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Import Java classes
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
import java.io.*;
import java.net.*;


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% If an old MATLAB server is running, close it
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% If a server object exists from a previous run, close it.
if (exist('MatlabServer_server'))
  close(MatlabServer_server);
  clear MatlabServer_server;
end

% If an input stream exists from a previous run, close it.
if (exist('MatlabServer_is'))
  close(MatlabServer_is);
  clear MatlabServer_is;
end

% If an output stream exists from a previous run, close it.
if (exist('MatlabServer_os'))
  close(MatlabServer_os);
  clear MatlabServer_os;
end

fprintf(2, '----------------------\n');
fprintf(2, 'MATLAB server started!\n');
fprintf(2, '----------------------\n');

fprintf(2, 'MATLAB working directory: %s\n', pwd);


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Initiate server socket to which clients may connect
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
MatlabServer_port = getenv('MATLABSERVER_PORT');
```

```
if (length(MatlabServer_port) > 0)
  MatlabServer_port = str2num(MatlabServer_port);
else
  % Try to open a server socket on port 9999
  MatlabServer_port = 9999;
end

% Ports 1-1023 are reserved for the Internet Assigned Numbers Authority.
% Ports 49152-65535 are dynamic ports for the OS. [3]
if (MatlabServer_port < 1023 | MatlabServer_port > 65535)
 error('Cannot not open connection. Port (''MATLABSERVER_PORT'') is out of range [1023,65535]: %d', Mat
end

fprintf(2, 'Trying to open server socket (port %d)...', MatlabServer_port);
MatlabServer_server = java.net.ServerSocket(MatlabServer_port);
fprintf(2, 'done.\n');


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Wait for client to connect
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Create a socket object from the ServerSocket to listen and accept
% connections.
% Open input and output streams

% Wait for the client to connect
fprintf(2, 'Waiting for client to connect (port %d)...', MatlabServer_port);
MatlabServer_clientSocket = accept(MatlabServer_server);
fprintf(2, 'connected.\n');

% ...client connected.
MatlabServer_is = java.io.DataInputStream(getInputStream(MatlabServer_clientSocket));
MatlabServer_os = java.io.DataOutputStream(getOutputStream(MatlabServer_clientSocket));



% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% The MATLAB server state machine
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Commands
MatlabServer_commands = {'eval', 'send', 'receive', 'send-remote', 'receive-remote', 'echo', 'evalc'};

MatlabServer_lasterr = [];
MatlabServer_variables = [];

% As long as we receive data, echo that data back to the client.
MatlabServer_state = 0;
while (MatlabServer_state >= 0),
```

```
 if (MatlabServer_state == 0)
   MatlabServer_tmp_cmd = readByte(MatlabServer_is);
   fprintf(2, 'Received cmd: %d\n', MatlabServer_tmp_cmd);
  if (MatlabServer_tmp_cmd < -1 | MatlabServer_tmp_cmd > length(MatlabServer_commands))
     fprintf(2, 'Unknown command code: %d\n', MatlabServer_tmp_cmd);
   else
     MatlabServer_state = MatlabServer_tmp_cmd;
   end
   clear MatlabServer_tmp_cmd;

 %------------------
 % 'evalc'
 %------------------
elseif (MatlabServer_state == strmatch('evalc', MatlabServer_commands, 'exact'))
   MatlabServer_tmp_bfr = char(readUTF(MatlabServer_is));
   fprintf(2, '"evalc" string: "%s"\n', MatlabServer_tmp_bfr);
   try
     MatlabServer_tmp_bfr = sprintf(MatlabServer_tmp_bfr);
     MatlabServer_tmp_result = evalc(MatlabServer_tmp_bfr);
     writeByte(MatlabServer_os, 0);
     fprintf(2, 'Sent byte: %d\n', 0);
     writeUTF(MatlabServer_os, MatlabServer_tmp_result);
     fprintf(2, 'Sent UTF: %s\n', MatlabServer_tmp_result);
     flush(MatlabServer_os);
     clear MatlabServer_tmp_result;
   catch
   MatlabServer_lasterr = sprintf('Failed to evaluate expression ''%s''.', MatlabServer_tmp_bfr);
     fprintf(2, 'EvaluationException: %s\n', MatlabServer_lasterr);
     writeByte(MatlabServer_os, -1);
     fprintf(2, 'Sent byte: %d\n', -1);
     writeUTF(MatlabServer_os, MatlabServer_lasterr);
     fprintf(2, 'Sent UTF: %s\n', MatlabServer_lasterr);
     flush(MatlabServer_os);
   end
   flush(MatlabServer_os);
   MatlabServer_state = 0;
   clear MatlabServer_tmp_bfr;

 %------------------
 % 'eval'
 %------------------
elseif (MatlabServer_state == strmatch('eval', MatlabServer_commands, 'exact'))
   MatlabServer_tmp_bfr = char(readUTF(MatlabServer_is));
   fprintf(2, '"eval" string: "%s"\n', MatlabServer_tmp_bfr);
   try
     eval(MatlabServer_tmp_bfr);
     writeByte(MatlabServer_os, 0);
     fprintf(2, 'Sent byte: %d\n', 0);
```

```
    flush(MatlabServer_os);
  catch
  MatlabServer_lasterr = sprintf('Failed to evaluate expression ''%s''.', MatlabServer_tmp_bfr);
    fprintf(2, 'EvaluationException: %s\n', MatlabServer_lasterr);
    writeByte(MatlabServer_os, -1);
    fprintf(2, 'Sent byte: %d\n', -1);
    writeUTF(MatlabServer_os, MatlabServer_lasterr);
    fprintf(2, 'Sent UTF: %s\n', MatlabServer_lasterr);
    flush(MatlabServer_os);
  end
  flush(MatlabServer_os);
  MatlabServer_state = 0;
  clear MatlabServer_tmp_bfr;

%--------------------
% 'send'
%--------------------
elseif (MatlabServer_state == strmatch('send', MatlabServer_commands, 'exact'))
 MatlabServer_tmp_tmpname = sprintf('%s_%d.mat', tempname, MatlabServer_port);
MatlabServer_tmp_expr = sprintf('save(MatlabServer_tmp_tmpname, ''%s''', MatlabServer_saveOption);
  MatlabServer_tmp_ok = 1;
  for MatlabServer_tmp_k=1:length(MatlabServer_variables),
    MatlabServer_tmp_variable = MatlabServer_variables{MatlabServer_tmp_k};
    if (exist(MatlabServer_tmp_variable) ~= 1)
   MatlabServer_lasterr = sprintf('Variable ''%s'' not found.', MatlabServer_tmp_variable);
      fprintf(2, '%s\n', MatlabServer_lasterr);
      MatlabServer_tmp_ok = 0;
      break;
    end;
  MatlabServer_tmp_expr = sprintf('%s, ''%s''', MatlabServer_tmp_expr, MatlabServer_tmp_variable);
  end;

  MatlabServer_tmp_expr = sprintf('%s)', MatlabServer_tmp_expr);
  if (~MatlabServer_tmp_ok)
    writeInt(MatlabServer_os, -1);
    writeUTF(MatlabServer_os, MatlabServer_lasterr);
  else
    fprintf(2, '%s\n', MatlabServer_tmp_expr);
    eval(MatlabServer_tmp_expr);
    writeInt(MatlabServer_os, 0); % Here anything but -1 means "success"
    writeUTF(MatlabServer_os, MatlabServer_tmp_tmpname);
  end

  MatlabServer_tmp_answer = readByte(MatlabServer_is);
  fprintf(2, 'answer=%d\n', MatlabServer_tmp_answer);

  MatlabServer_state = 0;
 clear MatlabServer_tmp_name MatlabServer_tmp_expr MatlabServer_tmp_ok MatlabServer_tmp_answer;
```

```
%-------------------
% 'send-remote'
%-------------------
elseif (MatlabServer_state == strmatch('send-remote', MatlabServer_commands, 'exact'))
 MatlabServer_tmp_tmpname = sprintf('%s_%d.mat', tempname, MatlabServer_port);
 MatlabServer_tmp_expr = sprintf('save(MatlabServer_tmp_tmpname, ''%s''', MatlabServer_saveOption);
  MatlabServer_tmp_ok = 1;
  for MatlabServer_tmp_k=1:length(MatlabServer_variables),
    MatlabServer_tmp_variable = MatlabServer_variables{MatlabServer_tmp_k};
    if (exist(MatlabServer_tmp_variable) ~= 1)
     MatlabServer_lasterr = sprintf('Variable ''%s'' not found.', MatlabServer_tmp_variable);
      fprintf(2, '%s\n', MatlabServer_lasterr);
      MatlabServer_tmp_ok = 0;
      break;
    end;
  MatlabServer_tmp_expr = sprintf('%s, ''%s''', MatlabServer_tmp_expr, MatlabServer_tmp_variable);
  end;
  clear MatlabServer_tmp_k MatlabServer_tmp_variable;

  MatlabServer_tmp_expr = sprintf('%s)', MatlabServer_tmp_expr);
  if (~MatlabServer_tmp_ok)
    writeInt(MatlabServer_os, -1);
    writeUTF(MatlabServer_os, MatlabServer_lasterr);
  else
    fprintf(2, '%s\n', MatlabServer_tmp_expr);
    eval(MatlabServer_tmp_expr);
    MatlabServer_tmp_file = java.io.File(MatlabServer_tmp_tmpname);
    MatlabServer_tmp_maxLength = length(MatlabServer_tmp_file);
    clear MatlabServer_tmp_file;
  writeInt(MatlabServer_os, MatlabServer_tmp_maxLength); % Here anything but -1 means "success"
    fprintf(2, 'Send int: %d (maxLength)\n', MatlabServer_tmp_maxLength);
    MatlabServer_tmp_fid = fopen(MatlabServer_tmp_tmpname, 'r');
    MatlabServer_tmp_count = 1;
    while (MatlabServer_tmp_count ~= 0)
     [MatlabServer_tmp_bfr, MatlabServer_tmp_count] = fread(MatlabServer_tmp_fid, 65536, 'int8');
      if (MatlabServer_tmp_count > 0)
        write(MatlabServer_os, MatlabServer_tmp_bfr);
      end;
    end;
    fclose(MatlabServer_tmp_fid);
    fprintf(2, 'Send buffer: %d bytes.\n', MatlabServer_tmp_maxLength);
    delete(MatlabServer_tmp_tmpname);
  clear MatlabServer_tmp_bfr MatlabServer_tmp_count MatlabServer_tmp_maxLength MatlabServer_tmp_fi
  end
  flush(MatlabServer_os);

  MatlabServer_tmp_answer = readByte(MatlabServer_is);
```

```
  fprintf(2, 'answer=%d\n', MatlabServer_tmp_answer);

  MatlabServer_state = 0;
 clear MatlabServer_tmp_name MatlabServer_tmp_expr MatlabServer_tmp_ok MatlabServer_tmp_answer;

%------------------
% 'receive-remote'
%------------------
elseif (MatlabServer_state == strmatch('receive-remote', MatlabServer_commands, 'exact'))
  MatlabServer_tmp_len = readInt(MatlabServer_is);
 fprintf(2, 'Will read MAT file structure of length: %d bytes.\n', MatlabServer_tmp_len);

  MatlabServer_tmp_reader = InputStreamByteWrapper(4096);
  MatlabServer_tmp_bfr = [];
  MatlabServer_tmp_count = 1;
  while (MatlabServer_tmp_len > 0 & MatlabServer_tmp_count > 0)
  MatlabServer_tmp_count = MatlabServer_tmp_reader.read(MatlabServer_is, min(4096, MatlabServer_tmp
    if (MatlabServer_tmp_count > 0)
   MatlabServer_tmp_bfr = [MatlabServer_tmp_bfr; MatlabServer_tmp_reader.bfr(1:MatlabServer_tmp_co
      MatlabServer_tmp_len = MatlabServer_tmp_len - MatlabServer_tmp_count;
    end;
  end;

  clear MatlabServer_tmp_reader MatlabServer_tmp_count MatlabServer_tmp_len;

 MatlabServer_tmp_tmpfile = sprintf('%s_%d.mat', tempname, MatlabServer_port);
  MatlabServer_tmp_fh = fopen(MatlabServer_tmp_tmpfile, 'wb');
  fwrite(MatlabServer_tmp_fh, MatlabServer_tmp_bfr, 'int8');
  fclose(MatlabServer_tmp_fh);

  clear MatlabServer_tmp_fh MatlabServer_tmp_bfr;

  load(MatlabServer_tmp_tmpfile);

  delete(MatlabServer_tmp_tmpfile);
  clear MatlabServer_tmp_tmpfile;
  writeByte(MatlabServer_os, 0);

  MatlabServer_state = 0;

%------------------
% 'receive'
%------------------
elseif (MatlabServer_state == strmatch('receive', MatlabServer_commands, 'exact'))
  MatlabServer_tmp_filename = char(readUTF(MatlabServer_is));
  fprintf(2, 'Will read MAT file: "%s"\n', MatlabServer_tmp_filename);
  load(MatlabServer_tmp_filename);
  clear MatlabServer_tmp_filename;
```

```
    writeByte(MatlabServer_os, 0);
    MatlabServer_state = 0;
    clear MatlabServer_tmp_filename;
  end
end


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Shutting down the MATLAB server
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

fprintf(2, '-----------------------\n');
fprintf(2, 'MATLAB server shutdown!\n');
fprintf(2, '-----------------------\n');
writeByte(MatlabServer_os, 0);
close(MatlabServer_os);
close(MatlabServer_is);
close(MatlabServer_server);
quit;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HISTORY:
% 2016-04-05 [v3.6.0]
% o All verbose/debug messages are outputted to stream #2 ("stderr").
% 2015-09-11 [v3.3.0]
% o Now temporary files use format <tempname>_<port>.mat.
% o Add 'MatlabServer_' prefix to all variables.
% o Add 'evalc' command.  Thanks to Rohan Shah for this.
% 2015-01-08 [v3.1.2]
% o BUG FIX: Matlab$getVariable() for a non-existing variable would
%   crash the R-to-Matlab communication if remote=FALSE.
% 2014-06-23 [v3.0.2]
% o ROBUSTNESS: Variables 'lasterr' and 'variables' are now always
%   defined. Potential bug spotted by Steven Jaffe at Morgan Stanley.
% o Added more progress/verbose output, e.g. current working directory.
% 2014-01-21 [v2.2.0]
% o BUG FIX: The MatlabServer.m script would incorrectly consider
%   Matlab v8 and above as Matlab v6.  Thanks to Frank Stephen at NREL
%   for reporting on this and providing a patch.
% 2013-07-11 [v1.3.5]
% o Updated messages to use 'MATLAB' instead of 'Matlab'.
% 2010-10-25 [v1.3.4]
% o BUG FIX: The MatlabServer.m script incorrectly referred to the
%   InputStreamByteWrapper class as java.io.InputStreamByteWrapper.
%   Thanks Kenvor Cothey at GMO LCC for reporting on this.
% 2010-08-28
% o Now the MatlabServer script reports its version when started.
% 2010-08-27
```

```
% o BUG FIX: Now MatlabServer.m saves variables using the function form,
%   i.e. save().  This solves the problem of having single quotation marks
%   in the pathname. Thanks Michael Q. Fan at NC State University for
%   reporting this problem.
% 2009-08-25
% o BUG FIX: Started to get the error "Undefined function or method
%   'ServerSocket' for input arguments of type 'double'.".  It seems like
%   import java.net.* etc does not work. A workaround is to specify the
%   full path for all Java classes, e.g. java.net.ServerSocket.
%   Thanks Nicolas Stadler for reporting this issue.
% 2006-12-28
% o Extended the accepted range of ports from [1023,49151] to [1023,66535].
% 2006-05-08
% o BUG FIX: The error message string for reporting port out of range
%   was invalid and gave the error '... Line: 109 Column: 45 ")" expected,
%   "identifier" found.'.  Thanks Alexander Nervedi for reporting this.
% 2006-01-21
% o Now an error is thrown if port number is out of (safe) range.
% o Added option to specify the port number via the system environment
%   variable MATLABSERVER_PORT, after request by Wang Yu, Iowa State Univ.
% 2005-03-08
% o BUG FIX: substring() is not recognized by MATLAB v7. Using regexp()
%   which works in MATLAB 6.5 and 7. Workaround eval('try', 'catch').
%   Thanks Patrick Drechsler, University of Wuerzburg for the bug report.
% 2005-02-24
% o Now the dynamic Java classpath is set for MATLAB v7 or higher. This
%   will simplify life for MATLAB v7 users.
% 2005-02-22
% o Added javaaddpath() to include InputStreamByteWrapper.class.
%   Thanks Yichun Wei for feedback and great suggestions.
% 2005-02-11
% o If MATLAB v7 or higher is detected, all MAT structures are saved with
%   option '-V6' so readMat() in R.matlab can read them.
% 2002-09-02 [or maybe a little bit earlier]
% o Created.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

InputStreamByteWrapper.(class|java) script

The Java class InputStreamByteWrapper is needed in order for MATLAB to *receive data* via a data stream. R sends data via a data stream if, and only if, the connection was setup for "remote" communication, that is, with argument `remote = TRUE`).

```
import java.io.*;

/**************************************************************************
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/506033241015010112