R.NET

Introduction

This page relates to R.NET 1.5.13. version 1.5.13 is functionally identical to 1.5.12, but is available as a package on NuGet.org.

R.NET enables the .NET Framework to interoperate with the R statistical language in the same process. R.NET requires .NET Framework 4 and the native R DLLs installed with the R environment. You can use R.NET from any language targetting .NET (it has been used at least from C#, F#, Vb.NET, IronPython). A couple of related works must be mentioned before you dive into this documentation. For F#, you probably should consider F# R Provider. One motivation for releasing 1.5.13 is for the RProvider to more easily manage dependency on R.NET.

Getting set up

There is a page gathering Software Prerequisites listing the platforms on which R.NET is known to run.
As of version 1.5.10, R.NET binaries are platform independent. You *might* need to set up a small add-on workaround on some Linux distributions (CentOS a known one), but you can just move and use the R.NET binaries across platforms.

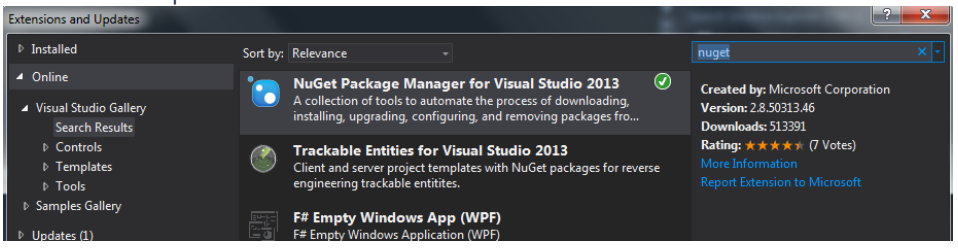
Visual Studio

If you are using the binaries from the zip file distribution, unzip the file and copy the content to a location of your choice. Add project references to RDotNet.dll and RDotNet.Native.dll the "usual" way.

NuGet is the preferred way to manage dependencies on R.NET.

If you are using the NuGet packages:

You first have to install, if you have not already, the NuGet package manager via Tools - Extension and Updates:



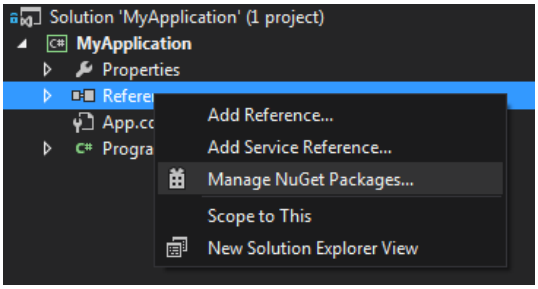
You can add the R.NET package as a dependency to one or more projects in your solution. For one project:

Search Wiki & Documentation

SYSTEM REQUIREMENTS

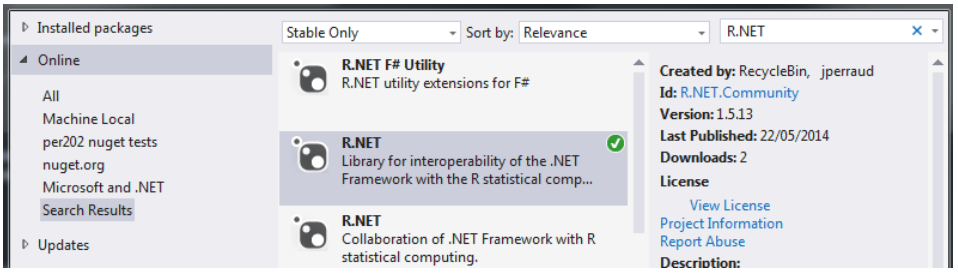
FOR USE

- .NET Framework 4 or above (Windows)
- Mono 3.2.3 or above (Mac or Linux) - may work with older versions
- R 3.0.2 (probably works with older versions)

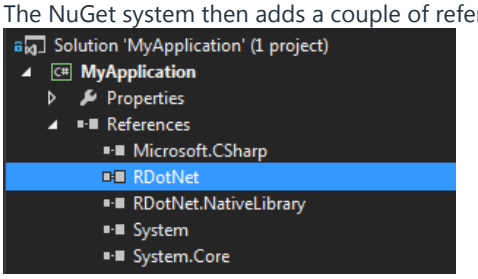


Note that you should probably uninstall packages dependencies or R.NET 1.5.5 or earlier, if pre-existing.

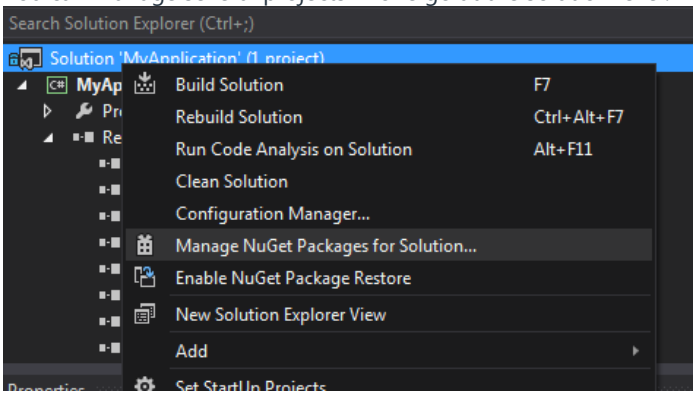
R.NET 1.5.13 uses a different package identifier: R.NET.Community. Be sure to use the most recent entry when searching for R.NET on Nuget:



The NuGet system then adds a couple of references.



You can manage several projects in one go at the solution level:



You can find more general information about NuGet at NuGet documentation

Xamarin Studio

This section is a placeholder as of 2014-04-20.

You may want to look at the page Setting up R.NET on Mac

Getting started with coding

R.NET 1.5.10 and subsequent versions include significant changes notably to alleviate two stumbling blocks often dealt with by users: paths to the R shared library, and preventing multiple engine initializations.

The following "Hello World" sample illustrates how the new API is simpler in 90% of use cases on Windows:

```
static void Main(string[] args)
{
        REngine.SetEnvironmentVariables(); // <-- May be omitted; the
next line would call it.
        REngine engine = REngine.GetInstance();
        // A somewhat contrived but customary Hello World:
        CharacterVector charVec = engine.CreateCharacterVector(new[] {
"Hello, R world!, .NET speaking" });
        engine.SetSymbol("greetings", charVec);
        engine.Evaluate("str(greetings)"); // print out in the console
        string[] a = engine.Evaluate("'Hi there .NET, from the R
engine'").AsCharacter().ToArray();
        Console.WriteLine("R answered: '{0}'", a[0]);
        Console.WriteLine("Press any key to exit the program");
        Console.ReadKey();
        engine.Dispose();
}
```

You retrieve a single REngine object instance, after setting the necessary environmental variables. Even the call to *SetEnvironmentVariables* can be omitted, though we'd advise you keep it explicit. *SetEnvironmentVariables*, on Windows, looks at the Registry settings set up by the R installer. If need be, you can override the behaviours setting the environment variables and engine initialization with your own steps, detailed in the Appendix.

On Linux/MacOS, the path to libR.so (for Linux) **must** be in the environment variable LD_LIBRARY_PATH **before** the process start, otherwise the R.NET engine will not properly initialize. If this is not set up, R.NET will throw an exception with a detailed message giving users hints as to what to do. Read the Appendix at the end of this page if R.NET complains about your LD_LIBRARY_PATH.

Sample code

You usually interact with the REngine object with the methods Evaluate, GetSymbol, and SetSymbol. To create R vector and matrices, the REngine object has extension methods such as CreateNumericVector, CreateCharacterMatrix, etc. Finally, you can invoke R functions in a variety of ways, using the method Evaluate of the REngine object, and also more directly.

Basic example with t-test statistic

It is available from the sample code 1 at https://github.com/jmp75/rdotnet-onboarding, as of 2014-04.

```
  static void Main(string[] args)
  {
        REngine.SetEnvironmentVariables();
        REngine engine = REngine.GetInstance();
        // REngine requires explicit initialization.
        // You can set some parameters.
        engine.Initialize();

        // .NET Framework array to R vector.
        NumericVector group1 = engine.CreateNumericVector(new
double[] { 30.02, 29.99, 30.11, 29.97, 30.01, 29.99 });
        engine.SetSymbol("group1", group1);
        // Direct parsing from R script.
```

```
        NumericVector group2 = engine.Evaluate("group2 <- c(29.89,
29.93, 29.72, 29.98, 30.02, 29.98)").AsNumeric();

        // Test difference of mean and get the P-value.
        GenericVector testResult = engine.Evaluate("t.test(group1,
group2)").AsList();
        double p = testResult["p.value"].AsNumeric().First();

        Console.WriteLine("Group1: [{0}]", string.Join(", ",
group1));
        Console.WriteLine("Group2: [{0}]", string.Join(", ",
group2));
        Console.WriteLine("P-value = {0:0.000}", p);

        // you should always dispose of the REngine properly.
        // After disposing of the engine, you cannot reinitialize nor
reuse it
        engine.Dispose();
  }
```

Numeric vectors

The following sample code illustrate the most used capabilities. It is extracted from the sample code 2 at https://github.com/jmp75/rdotnet-onboarding, as of 2014-04.

This illustrate basic operations with numeric vectors

```
var e = engine.Evaluate("x <- 3");
// You can now access x defined in the R environment
NumericVector x = engine.GetSymbol("x").AsNumeric();
engine.Evaluate("y <- 1:10");
NumericVector y = engine.GetSymbol("y").AsNumeric();
```

Calling R functions

While you may evaluate function calls by generating a string and call the Evaluate method, this can be unwieldy for cases where you pass large amounts of data. The following demonstrates how you may call a function, a bit like how you would invoke a function reflectively in .NET.

```
// Invoking functions; Previously you may have needed custom function
definitions
var myFunc = engine.Evaluate("function(x, y) { expand.grid(x=x, y=y)
}").AsFunction();
var v1 = engine.CreateIntegerVector(new[] { 1, 2, 3 });
var v2 = engine.CreateCharacterVector(new[] { "a", "b", "c" });
var df = myFunc.Invoke(new SymbolicExpression[] { v1, v2
}).AsDataFrame();
```

R.NET 1.5.10 includes many improvements to support function calls directly from C#, with less string manipulations and less calls to REngine.Evaluate.

```
// As of R.NET 1.5.10, more function call syntaxes are supported.
var expandGrid = engine.Evaluate("expand.grid").AsFunction();
var d = new Dictionary<string, SymbolicExpression>();
d["x"] = v1;
d["y"] = v2;
df = expandGrid.Invoke(d).AsDataFrame();
```

Data frame manipulations

Continuing with the results of our use of expand.grid, the following code illustrate that while R.NET tries to mimic the behavior of R with respect to data frames. Data frames are a central part of R data structures, so it is worth expanding with a few examples

```
engine.SetSymbol("cases", df);
// As of R.NET 1.5.10, factor to character expressions work
consistently with R
var letterCases =
engine.Evaluate("cases[,'y']").AsCharacter().ToArray();
// "a","a","a","b","b","b", etc. Same as as.character(cases[,'y']) in
R
// Note that this used to return  "1", "1", "1", "2", "2", etc. with
R.NET 1.5.5
```

There are other ways to extract columns from the data frame, without passing strings of R expressions:

```
// Equivalent:
letterCases = df[1].AsCharacter().ToArray();
letterCases = df["y"].AsCharacter().ToArray();
```

The behavior for what is returned by 2-dimensional indexing usually mirrors what is observed directly in R. One exception is when row names are missing; the R behavior is debatable, so R.NET prefers to be strict.

```
// Accessing items by two dimensional indexing
string s = (string)df[1, 1]; // "a"
s = (string)df[3, 1]; // "a"
s = (string)df[3, "y"]; // "b"
// s = (string)df["4", "y"]; // fails because there are no row names
df[3, "y"] = "a";
s = (string)df[3, "y"]; // "a"
df[3, "y"] = "d";
s = (string)df[3, "y"]; // null, because we have an <NA> string in R
```

Calling R scripts

To reuse whole scripts, the simplest method is to use the 'source' function in R

```
engine.Evaluate("source('c:/src/path/to/myscript.r')");
```

Missing values

Placeholder, showing what happens bidirectionally with NA values for the various vector types. See the Data Types section later in this page.

Further examples

Looking at the unit tests under the project RDotNet.Tests will provide further information on R.NET uses and programming idioms.
Illustrate the speed of data transfer

Runtime performance

Placeholder, showing best practices to maximise runtime speed

Other examples to document yet

Placeholder

- Handling date and time

Data Types

All expressions in R are represented as SymbolicExpression objects in R.NET. For data access, the following special classes are defined. Note that there is no direct equivalent in .NET for 'NA' values in R. Special values are used for some types but pay attention to the behaviour, so as not to risk incorrect calculations.

Table. Classes in R.NET bridges between R and .NET Framework.

R	R.NET	.NET Framework	Note
character vector	RDotNet.CharacterVector	System.String[]	
integer vector	RDotNet.IntegerVector	System.Int32[]	The minimum value in R is -2^31+1 while that of .NET Framework is -2^31. Missing values are int.MinValue
real vector	RDotNet.NumericVector	System.Double[]	Missing values are represented as double.NaN
complex vector	RDotNet.ComplexVector	System.Numerics.Complex[]	System.Numerics assembly is required for .NET Framework 4.
raw vector	RDotNet.RawVector	System.Byte[]	
logical vector	RDotNet.LogicalVector	System.Boolean[]	
character matrix	RDotNet.CharacterMatrix	System.String[,]	
integer matrix	RDotNet.IntegerMatrix	System.Int32[,]	The minimum value in R is -2^31+1 while that of .NET Framework is -2^31.
real matrix	RDotNet.NumericMatrix	System.Double[,]	
complex matrix	RDotNet.ComplexMatrix	System.Numerics.Complex[,]	Reference to System.Numerics assembly is required.
raw matrix	RDotNet.RawMatrix	System.Byte[,]	
logical matrix	RDotNet.LogicalMatrix	System.Boolean[,]	
list	RDotNet.GenericVector		**From version 1.1.**
data frame	RDotNet.GenericVector		**From version 1.1.** RDotNet.DataFrame class is also available (below).
data frame	RDotNet.DataFrame		**From version 1.3.** And from version 1.5.3, DataFrameRowAttribute and DataFrameColumnAttribute are available for data mapping.
function	RDotNet.Function		**From version 1.4.** Including closure, built-in function, and special function.

factor	RDotNet.Factor	System.Int32[]	**From version 1.5.2.**
S4	RDotNet.S4Object		Not Available Yet. See S4 branch in the source control.

Acknowledgements

Contributors, directly or indirectly, to the code for this release are jperraud, kos59125, evolvedmicrobe, Daniel Collins, gchapman, sukru, nakagawa_hiroyuki, JoeJoe

Appendices

Updating environment variables on Linux and MacOS

The path to libR.so (for Linux) **must** be in the environment variable LD_LIBRARY_PATH **before** the process start, otherwise the R.NET engine will not properly initialize. If this is not set up, R.NET will throw an exception with a detailed message.

For setting up on MacOS, you should read Evelyna Gabasova's Setting up R.NET on Mac

What you will need to do there depends on the Linux machine you are.
Let's say you needed to compile your own R from source, to get a shared R library:

```
LOCAL_DIR=/home/username/local
JAVAHOME=/apps/java/jdk1.7.0_25
cd ~src
cd R/
tar zxpvf R-3.0.2.tar.gz
cd R-3.0.2
./configure --prefix=$LOCAL_DIR --enable-R-shlib  CFLAGS="-g"
make
make install
```

Then prior to running a project with R.NET, you may need to update your LD_LIBRARY_PATH, and quite possibly PATH (though the latter can be done at runtime too).

```
LOCAL_DIR=/home/username/local
if [ "${LD_LIBRARY_PATH}" != "" ]
then
    export
LD_LIBRARY_PATH=$LOCAL_DIR/lib:$LOCAL_DIR/lib64/R/lib:/usr/local/lib64
:${LD_LIBRARY_PATH}
else
    export
LD_LIBRARY_PATH=$LOCAL_DIR/lib:$LOCAL_DIR/lib64/R/lib:/usr/local/lib64
fi
# You may as well update the PATH environment variable, though R.NET
does update it if need be.
export PATH=$LOCAL_DIR/bin:$LOCAL_DIR/lib64/R/lib:${PATH}
```

Workaround for dlerror: 'invalid caller' issue on some Linux boxes.

On at least one instance of one Linux flavour (CentOS), R.NET fails and 'dlerror' returns the

message 'invalid caller'.
Dowload and follow the instructions in the zip file "libdlwrap.zip" included in the this download page. If you use the source code, it is located under RDotNet.NativeLibrary/libdlwrap/
See https://rdotnet.codeplex.com/workitem/73 for detailed information about the issue.

Advanced options for the R engine initialization

This is a placeholder section.

- custom CharacterConsole
- Multiple app domains
- A nuget documentation page to set up a local feed.

Last edited Jun 17, 2014 at 3:20 PM by jperraud, version 27

COMMENTS

sehunley Oct 1, 2014 at 10:50 PM
I am currently working with the latest version of R.NET (version 1.5.14.13671) and when I use the onboarding examples described above, or whenever I call the engine from a WebAPI or web project I am getting an error where it can't find a function, usually the ones in the stats package. It fails even when I am trying to load the stats package.

I am running a pretty simple script:

```
# Goal: Show the efficiency of the mean when compared with the median
# using a large simulation where both estimators are applied on
# a sample of U(0,1) uniformly distributed random numbers.

one.simulation <- function(N=100) { # N defaults to 100 if not supplied
x <- runif(N)
return(c(mean(x), median(x)))
}
# Simulation --
results <- replicate(100000, one.simulation(20)) # Gives back a 2x100000 matrix

# Two kernel densities --
k1 <- density(results[1,]) # results[1,] is the 1st row
k2 <- density(results[2,])

# A pretty picture --
xrange <- range(k1$x, k2$x)
plot(k1$x, k1$y, xlim=xrange, type="l", xlab="Estimated value", ylab="")
grid()
lines(k2$x, k2$y, col="red")
abline(v=.5)
legend(x="topleft", bty="n",
lty=c(1,1),
col=c("black", "red"),
legend=c("Mean", "Median"))
```

The web example provided, if I can make it work, is PERFECT for what I'm try to do which is to execute an RScript
(maybe map some variables into and out of it) and then collect the results and even the images. But I can't seem
to find where the issue is happening to anyone else. I also verified the script runs in the R GUI. So if it's
running in a windows application or the R GUI, it's fine, but if it runs as a web service, web API or from a

Web Site - It's not recognizing the functions or packages correctly. It's as if it parses them incorrectly.

Any thoughts or ideas that anyone has would be greatly appreciated!

sehunley

touchbutton Oct 12, 2013 at 9:58 PM
I tested R.NET 1.5.5 with R 3.0.1 x64 on Windows 7, it worked perfectly well. I tested round-tripping dotnet IEnumerable<int>, <string>, <DateTime>, <decimal>, <double> with REngine numeric, POSIXCT, character vectors, creating data.frame in REngine by combinding vectors. It worked efficiently and I managed to create vectors and data.frame with length larger than 2 million quite quickly. I conclude that the library is very stable now. Thanks.

billybond Sep 24, 2013 at 7:18 PM
@jdizzy
what a load of utter nonsense. I have had great success with this project and find it far more useable that the COM method. You clearly need to get laid or have a drink or something.

BenWiseman Sep 20, 2013 at 7:13 PM
This set up guide doesn't work - even if you copy it directly :/

touchbutton Aug 16, 2013 at 11:16 AM
It's great to see new updates. This can potentially be an extremely useful library. Keep up with the work!

viprenkun Aug 13, 2013 at 4:18 PM
The new version here perfectly works with R 3.0.1 as I tested.

kulong995 Aug 10, 2013 at 4:28 PM
thanks for new version.!!

Gravitas Jan 6, 2013 at 3:48 AM
Has anyone managed to get it to work with VS2012?

jdizzy769 Aug 17, 2012 at 2:06 AM
I return hat in hand after my previous post. Although I have not yet tested the memory management and garbage collection, which was the cause of all of my frustrations last time, v1.5 is just plain AWESOME. I am giving it another shot. Thank you for the update!!

QuidProMS Mar 27, 2012 at 6:36 AM
In response to jdizzy769, i think you have it wrong. This is a great adult project, albeit a work in progress with some bugs, that works quite well if the programmer avoids the buggy usages, which seem to involve multiple R instances (see below). The whole idea of anything dotnet is NOT to have to *explicitly* use COM, e.g. R(D)-COM. R.Net seems to manage the memory involved across the managed/unmanaged boundaries quite well, but more testing on my part will tell, and that makes programming a lot easier. There are some features I of R.Net I don't understand, but the idea of getting delegate for an R.dll internal, var cos = engine.GetSymbol("cos").AsFunction(), is very powerful. In response to those who want multiple instances of the REngine, just glancing at the code, it seems like the problem is that the R.dll is LoadLibrary() loaded multiple times into the same process and upon calling yet again "setup_Rmainloop()", the application is locked. I don't know much about the R.dll code, but I can guess it is not threadsafe, and it is an error to call "setup_Rmainloop()" again before exiting the main loop. So the multiple instance issue seems to demand either one process per instance, with the concomitant added complexity to R.Net, or a revision of the R.dll to support multiple instances.

aoldevstat Oct 20, 2011 at 9:14 AM
I've quickly become a fan of the R.NET and started using it in real solutions even though it took some additional work. This is pretty fast, much more faster than RDCOM library.

Please don't leave it and develop it better and better. I agree that error handling still needs work. Why not through the ParseException? Using ICharacterDevice to do that isn't the best way in my opinion...

FrankyHollywood Oct 13, 2011 at 10:37 PM
The idea is very good, however it keeps crashing constantly. Once a crash has occured I have to restart VS2010 again before I can send any command to the engine before it responds without error (probably the com component which need te reset or something). I'm trying to build a webservice, which needs multiple R sessions. Maybe a winforms app with just 1 engine instance works better.

conclusion, nice project, pretty simple and straightforward interface, but session management and engine response/error handling need some work. That would make it production worthy :)

tylt Aug 19, 2011 at 3:03 PM
I think it's a worthwhile project, and I'm glad it exists. Definitely needs a lot more work, but I think it'll evolve to be more useful. COM is reliable, but the performance is pretty poor.

jdizzy769 Aug 18, 2011 at 10:31 PM
VERY buggy and terribly supported. This is merely a hobby project, and a juvenile one at that. Use R(D)-COM if you need it to actually work.

Sign in to add a comment