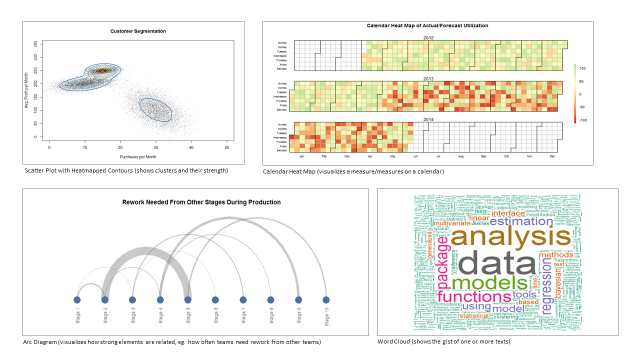
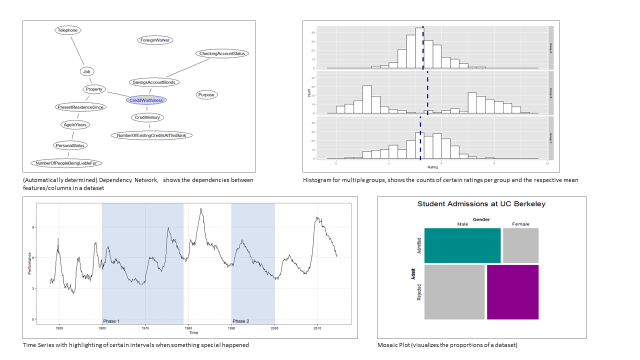
|  |  |  |  |
| --- | --- | --- | --- |
| Summary and Notes This document describes the R Graphics Device for Reporting Services and contains the steps required for installation.  The R Graphics Device for Reporting Services has been developed by Timo Klimmer, Microsoft, and is not an official Microsoft product. The solution provided is provided on an "AS IS" basis without any warranty or technical support. The solution may not have all desired features and may not have been tested extensively. Thus feel free to further extend this solution and please share your enhancements publicly. Do not use this component in a production environment without sufficient testing.  The R Graphics Device works with Reporting Services both in SharePoint Mode as well as in Native Mode (although the assembly that is compiled with the provided source code needs to be compiled differently for each mode and version of SQL Server/SharePoint).  Version: 1.0, 2014-10-23.  Contents  [Summary and Notes 1](#_Toc397378660)  [What is it, the "R Graphics Device for Reporting Services"? 2](#_Toc397378661)  [Installation 4](#_Toc397378662)  [Usage 7](#_Toc397378663)  [Known Limitations/Issues 8](#_Toc397378664) |  |  |  |

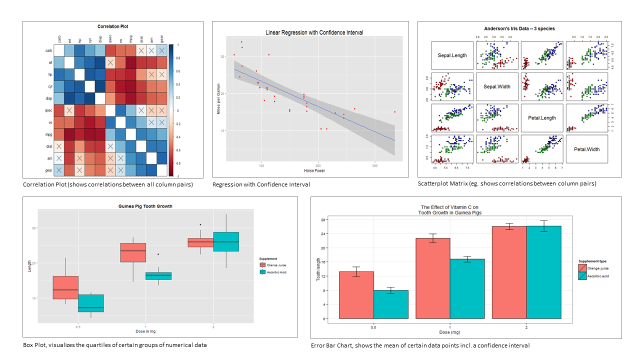
# What is it, the "R Graphics Device for Reporting Services"?

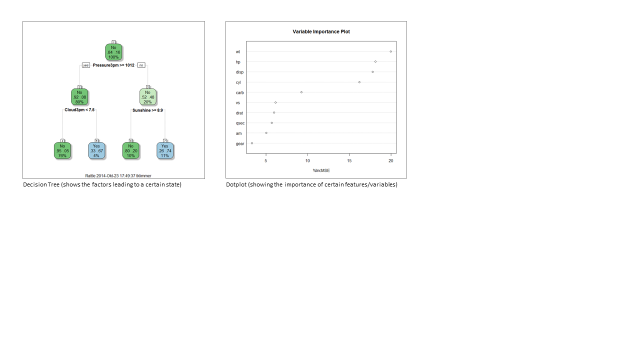
R is a programming language which has its focus on statistics, data mining and machine learning. One of the very valuable features of R is its wide range of different chart types it can plot, and R can also be extended fairly easily to plot even more chart types.

Only a few examples:









The "R Graphics Device for Reporting Services" is a Custom Report Item in Reporting Services that renders the graphics output of arbitrary R code in Reporting Services. Thereby, it is possible to do the following:

* Use all the SRSS report management and subscription infrastructure
* Plot R data visualizations with SSRS with or without passing SSRS parameters
* Leverage R's advanced data mining and machine learning computation algorithms

Through the R integration given by the R Graphics Device, very powerful reports can be created that go beyond Reporting Services' out-of-the-box functionality.

# Installation

The R Graphics Device for Reporting Services needs to be installed on the Reporting Services server as well as in Visual Studio. As there is no automated installation routine yet, configuration and compilation steps need to be done manually.

The following steps are not exactly straightforward, and it is "easy" to make mistakes or maybe even break Reporting Services. In case you encounter an error and don't know how to fix the issue, a good starting point is always the log files or Windows' Event Viewer. Besides, it is always a good idea to make a copy of the config files to be modified before modification so that you can always return to a "clean" configuration.

*Prerequisites*

* Decide about the R distribution that shall be used. The free R distribution is available at <http://r-project.org>. In case you need support and/or more advanced features, a commercial distribution such as the one provided by <http://www.revolutionanalytics.com> may suit better to your requirements, although code modifications may be required.
* Ensure approval to give the SSRS.CustomReportItems assembly FullTrust permissions in the used Report Server instance. Note: FullTrust is required because we are invoking R when the report is rendered. In case of doubts, it is always possible to review the code, compile the code again and use this reviewed compilation.
* Ensure that you have proper permissions on the computers you will work on. "Proper permissions" means administrative privileges. (It may also work with fewer permissions but I have never tested the exactly required permissions.)

*Reporting Services Server*

1. Download and install R.

Note: When installing, select the option in the installer to set the R version in the registry.

1. Check that R basically works on the machine.
2. Locate the *Microsoft.ReportingServices.ProcessingCore.dll* assembly of the Reporting Services instance you want to extend.

Example for Native Mode / SQL Server 2014:

C:\Program Files\Microsoft SQL Server\MSRS12.MSSQLSERVER\Reporting Services\ReportServer\bin

Example for SharePoint Mode / SQL Server 2012, SharePoint 2013

C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\WebServices\Reporting\bin

1. Open the provided Visual Studio solution and ensure that the CustomReportItems project references exactly that Microsoft.ReportingServices.ProcessingCore.dll assembly (an exact copy of the file is ok but not the assembly of another Reporting Services version).
2. Check that the target platform in Visual Studio is set right. Use MSIL for SharePoint Reporting Services and as well as for Native Mode.
3. Also, the target .NET framework should be the same version as the version referenced by the Microsoft.ReportingServices.ProcessingCore.dll mentioned above. SQL 2014 Native uses .NET 3.5.
4. Recompile the project and put the just compiled SSRS.CustomReportItems.dll file into the same folder where the Microsoft.ReportingServices.ProcessingCore.dll file resides.
5. Go in the folder above and edit the rsreportserver.config file, eg. C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\WebServices\Reporting\rsreportserver.config.

Ensure that the following ReportItem element is specified under Configuration/Extensions/ReportItems.

<ReportItem

Name="RGraphicsDevice" Type="SSRS.CustomReportItems.RGraphicsDeviceReportItem,SSRS.CustomReportItems" />

1. Edit the rssrvpolicy.config file in the same folder and ensure that the following CodeGroup element is added as sibling to the other inner CodeGroup elements.

<CodeGroup class="UnionCodeGroup" version="1" PermissionSetName="FullTrust"

Description="This code group grants SSRS.CustomReportItems.dll FullTrust permission.">

<IMembershipCondition class="UrlMembershipCondition" version="1"

Url="C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\WebServices\Reporting\bin\SSRS.CustomReportItems.dll" />

</CodeGroup>

Notes: You may adjust the path highlighted in yellow. If the path in the code below does not match, adjust it.

1. For SharePoint 2013/SQL Server 2012: Run the following Powershell code in SharePoint 2013 Management Shell. Also adjust paths if required.

$codeGroup = @"

<CodeGroup class="UnionCodeGroup" version="1" PermissionSetName="FullTrust"

Description="This code group grants SSRS.CustomReportItems.dll FullTrust permission.">

<IMembershipCondition class="UrlMembershipCondition" version="1"

Url="C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\WebServices\Reporting\bin\SSRS.CustomReportItems.dll" />

</CodeGroup>

";

New-SPRSExtension -Identity (Get-SPRSServiceApplication) -ExtensionType 'ReportItems' -Name 'RGraphicsDevice' -TypeName 'SSRS.CustomReportItems.RGraphicsDeviceReportItem,SSRS.CustomReportItems' -CodeGroup $codeGroup

New-SPRSExtension -Identity (Get-SPRSServiceApplication) -ExtensionType 'ReportItemDesigner' -Name 'RGraphicsDevice' -TypeName 'SSRS.CustomReportItems.Designer.RGraphicsDeviceDesigner, SSRS.CustomReportItems.Designer'

*Visual Studio*

1. Download and install R.

Note: When installing, select the option in the installer to set the R version in the registry.

1. Check that R basically works on the machine.
2. Edit the RSReportDesigner.config file in folder C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\PrivateAssemblies and add the two following XML elements under the Extensions element.

<ReportItems>

<ReportItem Name="RGraphicsDevice" Type="SSRS.CustomReportItems.RGraphicsDeviceReportItem,SSRS.CustomReportItems" />

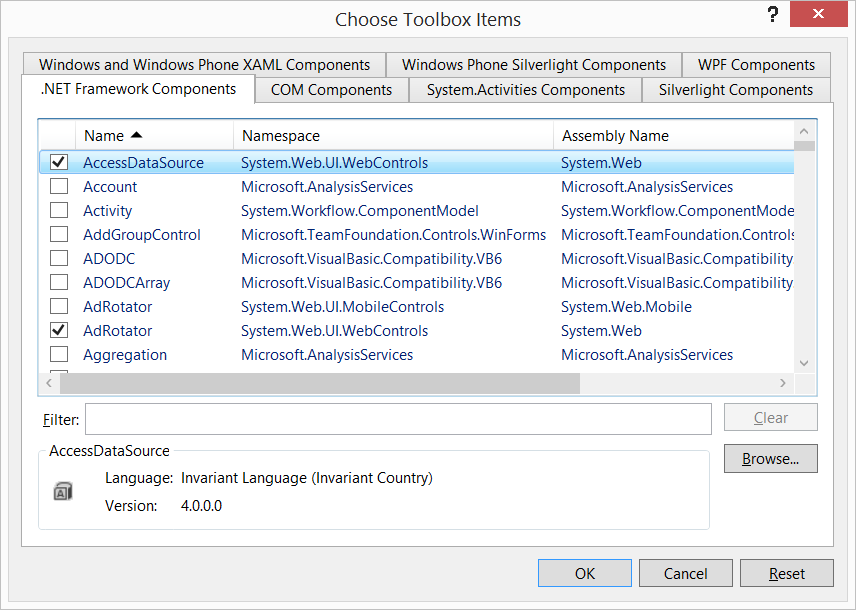
</ReportItems>

<ReportItemDesigner>

<ReportItem Name="RGraphicsDevice" Type="SSRS.CustomReportItems.Designer.RGraphicsDeviceDesigner,SSRS.CustomReportItems.Designer" />

</ReportItemDesigner>

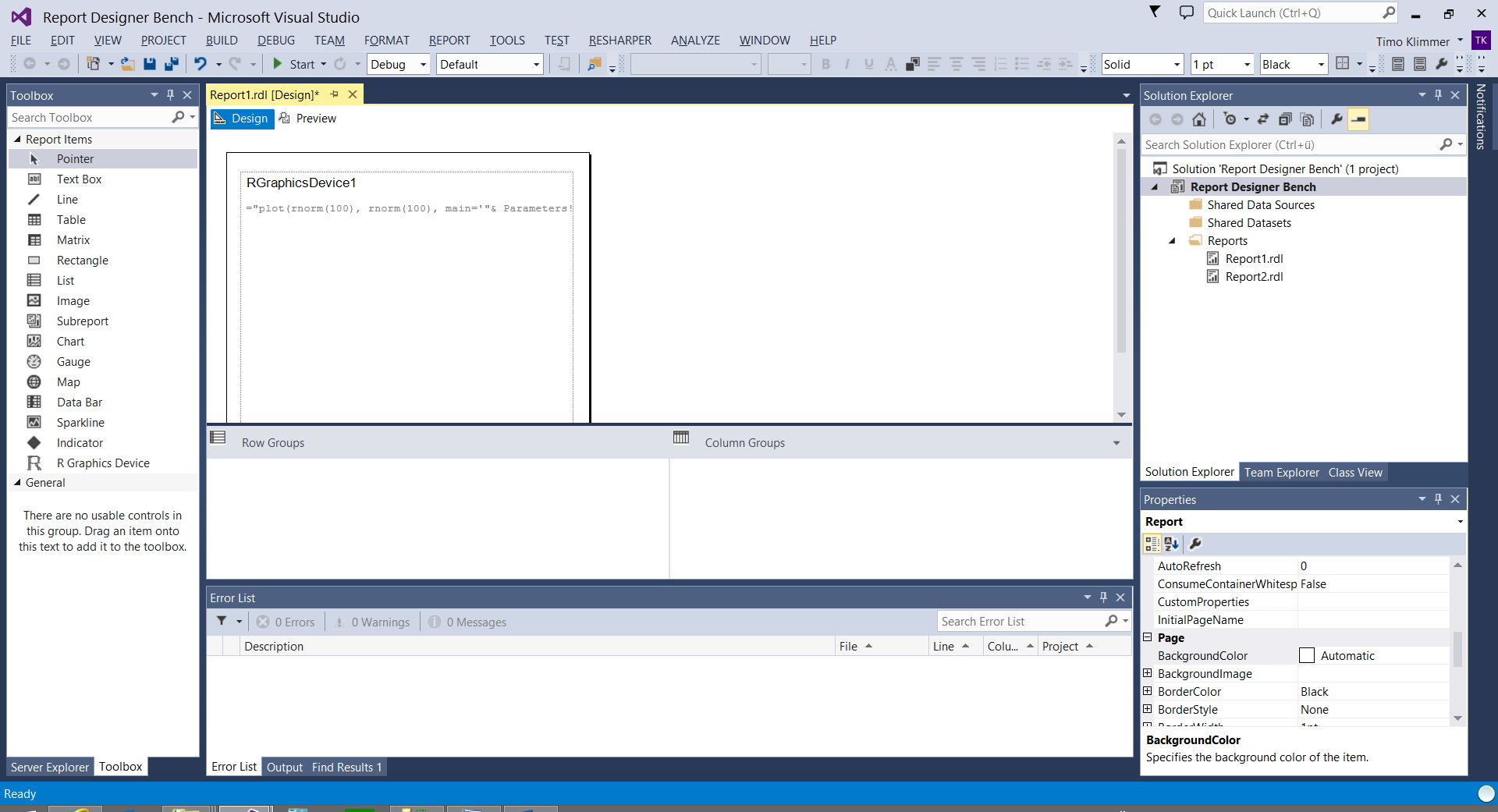
1. Open a Reporting Service project in Visual Studio and show the Toolbox window.
2. Click the right button in the Toolbox window to open the context menu / Choose Items...
3. Once the dialog has opened, click on Browse...



1. Select the SSRS.CustomReportItems.Designer.dll file in folder C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\PrivateAssemblies.

# Usage

1. Once added, the toolbox will show a new report item, the "R Graphics Device". You can drag and drop this item onto the report page as you would do with any other item.



1. Once added, you can change the Code property of the RGraphics Device item to denote the R code whose output you want to render. The Code property supports expressions, thus you can also use parameterized R code.
2. Besides, you can change the DPI property which will change the resolution of the image produced by R.

# Debugging

When a report uses custom report items, the Preview pane in Visual Studio/Business Intelligence Development Studio/Reporting Services will open a separate process once the report is previewed (will be visible by showing a black commandline window).

Now to debug the report item code, you can place the debug information (.pdb file) that is compiled with the custom report item assembly into the PrivateAssemblies folder of your Visual Studio installation. If the debug information is in sync with the compiled assembly containining our custom report item (ie. they stem from the same build), you can attach a new Visual Studio instance with the custom report item code loaded to the separate process mentioned earlier.

Once attached, we can set breakpoints in our code etc. and debug the code as usual. Note that the debugging feature will not work if the same Visual Studio instance as the instance in which the Preview is done is used. Therefore, you will need two instances of Visual Studio.

# Known Limitations/Issues

* To show the output of R in Reporting Services, Reporting Services' custom report item feature is used. Custom report items can render only images. It is not possible to use interactivity features as eg. sliders provided by the manipulate package.
* Due to the architecture of Reporting Services' custom report item feature, a custom report item cannot know about the resolution in which the report output is rendered. The default resolution is 150dpi. If you want to change this, eg. because the charts look nicer in PDF when they are 300dpi, you can modify the Dpi property of the R Graphics Device report item. It is even possible to use expressions to determine the right DPI. However, due to SSRS' architecture, it is not possible to use the render format in the expression. It seems that SSRS does not even know itself at the time when the graphic of the custom report item is generated.
* Depending which Reporting Services flavor is used, the assemblies compiled from the provided code and/or the assemblies referenced by the assembly to be compiled may not match what Reporting Services expects. Ensure that the right assemblies (eg. those from SharePoint) are referenced when compiling, and that the assemblies are compiled for the right target platform (x64, x86 or MSIL). Also ensure that the right .NET framework is referenced by the C# solutions. In case you encounter a BadImageFormatException in your logs, this is a good hint that the assemblies you are using have not been compiled properly. If you do not know the target platform etc., use a decompiler to check the situation of similar assemblies and compile our assemblies likewise.
* Report Builder does not support Custom Report Items. Thus, the R Graphics Device for Reporting Services can also not be used in Report Builder.
* There is no feature that limits the functionality of R. As consequence, people might get more permissions on the server than they should. It should be possible to extend the code such that R is called in a specific user context with limited permissions. However, this is not implemented yet.
* Depending on the target platform configured when compiling, the component will either use the 32 or 64 bit version of R (x86 and MSIL lead to 32 bit R, x64 to 64 bit R). Be sure that the packages you want to use are installed in the right version of R. A future version of the R Graphics Device component may give better control about the R version used.