R Server Demo Instructions:

# Set up instructions:

1. Get the code from GitHub.
2. Install latest [R Client](http://aka.ms/rclient/download) in your computer with Windows 7 SP1 and above.
3. After installation, please follow [this guidance](https://msdn.microsoft.com/en-us/microsoft-r/r-client-get-started#step-2-configure-your-ide) to ensure your R IDE (RStudio or RTVS) configured correctly to use R Client as the R Engine.
4. Load scripts and set working directory
   1. Open “RServer\_FlightPredictionDemo.r” in your R IDE
   2. Configure the R IDE left pane to show script, right pane for R Interactive Window to show execution results.
   3. **Session--> Set working directory** to your local folder mentioned in step #1.

|  |
| --- |
| # Your local folder with demo script and .rds files. Replace “xx” with your folder path.  setwd("C:/xxx/xx/xxx”) |

1. Test if you can connect to the R Server (that Microsoft R Team is hosting in Azure).

* Please ensure you have VPN connection
* Please Add your server details and copy the following R scripts to test the connection;

|  |
| --- |
| remoteLoginAAD(  "YOUR AAD Server Address",  authuri = "Auth URI",  tenantid = "Tenant Id",  clientid = "Client Id",  resource = "Resource Id",  session = FALSE  ) |

*Please note: Please don’t share this internal server info with customers.*

1. This step is Optional. This is to set up the R Server in your local computer, in case there is no internet connection in your demo environment or cannot connect via VPN.

please follow [the guidance to set up R Server in your local machine](https://msdn.microsoft.com/en-us/microsoft-r/operationalize/configuration-initial#the-basic-one-box-configuration). After the set up and configuration is done, please use the following R scripts to test the connection. A pop up Window will ask you to enter user name and password. The user name is “**admin**” (case sensitive); The password is the one you just set during the configuration.

|  |
| --- |
| endpoint <- "http://localhost:12800"  remoteLogin(endpoint, session = FALSE, diff = FALSE) |

# Demo Instructions

The demo will use a deck to brief audience the value and the demo flow. It requires you to switch back and forth between deck and Visual Studio.

The deck is attached here.

FlightPredictionDemo.ppt

|  |  |
| --- | --- |
| Demo Screen / R Scripts | Narrative |
|  | I am very glad to demo the new deployment functionalities in Microsoft R Server V9.  This is the traditional deployment process: after the model is built, it took a long time to convert it to other programming languages and then integrate with business apps in production. It is an error prune process with slow innovation rates. |
|  | Microsoft R Server V9 introduces a new deployment solution. It exposes R models as web services so that Line of Business applications can integrate them without any translation, and developers don’t need to know R.  I will demo 3 experiences today.   * The first one, how to easily deploy the model we just built in HDI as a web service * The 2nd one, how to easily consume this web service inside R by another data scientist.   Thinking about the scenario that as a data scientist in QA team, after a model is deployed into production, I can validate and monitor its performance with the new production data, from R!   * The 3rd one, how to easily integrate this web service into an application written with C#.   Thinking about the scenario of integrating the prediction service with an online booking application. After I searched a flight, I can see the flight delay prediction. It will help me make better choice. |
|  | To run the demo, we need to set up an R server to host the web services. Due to the time limitations, I will not demo the set up and configuration. I will use a server already running in Azure. We have detail instructions in our online product documentation about the set up and configuration. |
| Switch to RTVS | Now, let’s start the demo. The first experience, deploy the model as a web service.  I’ve chosen to do my work in R Tools for Visual Studio. The demo will work with any R IDEs, such as RStudio. The R scripts are in the left pane and results will be shown in the right pane.  This is the model we just built in HDI in previous demo. We load it, associate with the training data’s column info, and wrap it with a scoring function. The function’s input data is a batch data set for scoring.  Now let’s test this function.  *<Action: highlight the Step 1 scripts and press “Ctrl+Enter” to run them>*  We can see the 900 predictions for each row of the data set. |
|  | The next step is to login to the remote R Server. As I mentioned before, we already set up a server in Azure.  *<Action: highlight the Step 2 scripts and press “Ctrl+Enter” to run them>*  We successfully connected to the remote R Server.  Tip: If you are going to run this demo with the R Server in your local computer, please replace these scripts with:   |  | | --- | | endpoint <- "http://localhost:12800"  remoteLogin(endpoint, session = FALSE, diff = FALSE) | |
|  | The last step, let’s deploy the model as a service. We can define the inputs and output for developer to easily consume it. We can define the version number for version control.  *<Action: highlight the Step 3 publishService lines and press “Ctrl+Enter” to run them>*  With this line of code, we got a service hosted in remote R server!  We can now verify it right away, without leaving R!  *<Action: highlight the rest codes in Step 3 and press “Ctrl+Enter” to run them>*  We saw the same 900+ prediction results.  Tips: Please change the version number in “publishService” function each time before you run the demo. If you use the same “service name” and same version, the publish will be fail.  Tips: Don’t be panic if you saw a failure. In most cases, you can change the version number to fix it. |
|  | We have completed the demo of the first experience. Now let’s demo the 2nd experience: how other data scientists can consume this web service.  Assume now I am another data scientist from QA team. I am going to verify the Flight Prediction web service just published by model team.  First, I use listServices function to list all available services.  *<Action: highlight ‘listServices’ code and press “Ctrl+Enter” to run them>*  I can explore and choose the one that I am interested. I find out the one just published. I choose it. Then I use getService function to consume the functions of that service.  *<Action: highlight ‘Get the services’ code and press “Ctrl+Enter” to run them>*  I can use the new flight data to verify and monitor the deployed models.  *<Action: highlight ‘Consume the Service’ code and press “Ctrl+Enter” to run them>*  You can see the 600 prediction results with the new data. The whole process is in R, so it is easy and nature for me.  Consuming these services in R is a unique capability of Microsoft R Sever. It enables a set of exciting scenarios. For example, data scientists can deploy not only models, but any arbitrary R scripts/functions, they can share the useful functions for other data scientist to quickly consume in R! |
| Switch to Slide | The last experience I am going to demo is how to easily integrate this web service with a C# app.  We leverage Swagger to achieve this goal. Swagger is the most popular open source API framework. It greatly simplifies the consumption of RESTful web services. By adding the Swagger description to our RESTful web services, developers can generate the codes easily.  It took 3 very easy steps to integrate the web service to an application.   * First, data scientist generates the swagger doc, with 2 line of code. * Second, developer uses open source tools such as AutoRest to generate the sample code for the web services. * Last step, developer writes a few line of C# codes to consume the service and use the autogenerated code. |
| Switch to RTVS | Now, let’s demo the first step, generating swagger doc.  After running these 2 line of code, data scientist will generate a swagger.json file in local machine. He/she can then share with the app developers to consume the web services. |
| Switch to the Command Line Windows | Now let’s demo the second step, using AutoRest tool to generate the sample code.  In the Command Line window, enter this one line of command, the **AutoRest tool will generate the C# code for the web services.**  *<Action: execute the command to generate the code>*  Tips:   1. Please bring up the command line window 2. Navigate to <the folder>/AutoRest mentioned in the Setup Instructions Step #1. 3. Copy/paste the one line of command from the PPT slide 5. 4. Get it ready before the demo will help you make the whole demo more smoothly. |
| Switch to Visual Studio | *<Action: load the ‘program.cs’ file into Visual Studio>*  Last step, developer writes a few lines of C# code to consume the service. They will use the autogenerated code.  You can integrate this code with the existing Flight Booking app to show the delay prediction.  You can see that during the whole process the developers don’t need to see R code at all.  That’s all for today’s demo. Easy deployment, easy consumption from R and easy integration with Apps. Thank you!  To Execute your code, You would need   * AutoRest Stubs that are available in AutoRest/Generated folder as part of this example * Newtonsoft.Json package * Microsoft.Rest.ClientRuntime package * Add you Authentication information.   + YOUR SERVER ADDRESS   + const string authority = "Authority server";   + const string clientId = "Client ID";   + const string clientKey = "Client Key"; |
|  |  |