Publishing Predictive Web Services with Microsoft R Server

Colby Ford

If the predictive analytics team of your organization is already using Microsoft R Server, you may have missed an exciting feature that helps with the operationalization of your models: **Web Services**. A story we often hear from our clients is that once a data scientist creates a satisfactory model, it is often difficult to productionalize the model for use across the organization. In other words, packaging the model to be accessible - or callable - by other individuals and multiple systems can be challenging. Luckily, with Microsoft R Server, this process is simple. Your entire data science team can publish models such that anyone you wish may consume them through a web service.

Enhance Your Analytical Workflow

In many organizations, struggling to operationalize the work of the data science team is a common problem. Despite the team building exceptional models with great business impact, it often takes far too long to implement their solutions.

R, while a great language for machine learning and statistical analyses, does not have an innate ability to productionalize a predictive model beyond saving the model objects to the user's local machine. Traditionally, it has been a very manual and labor-intensive task to put a predictive model into production for continual use. I've even heard of companies having to hard-code linear regression coefficients into SQL Server to calculate predictions. In some cases, models have to be recoded into another language (.NET, for example) to work with other systems. This does not have to be this difficult.

This is where Microsoft R Server comes into play. Microsoft R Server makes publishing your model as a callable web service simple; breaking the barrier from model testing to its use in production. Using only a few lines of code, a published model can be called by practically anything using an application program interface (API).

With Microsoft R Server as your deployment engine, your data science team will require far less attention from the IT infrastructure team (besides the initial server setup, of course).

For more information about the features of operationalization, follow this link.

Requirements

Web services are created using the `mrsdeploy` package, which is included in Microsoft R Server 9.x and Microsoft R Client 3.x.

For remote execution of a model, participating machines (nodes) can have one of the following configurations:

- Two machines running the same version of R Server 9.x. (They may even be on different supported platforms, such as one Linux and one Windows.)
- One machine running R Client 3.x and one machine running R Server 9.x, where the R Client user remotely logs into the R Server instance. (Execution is always on the R Server side.)

The requirements for remote execution include:

- An R Integrated Development Environment (IDE) such as RStudio or Visual Studio with R Tools.
- Authenticated access to an instance of Microsoft R Server.

Configuration and Security

R Server offers two types of configuration for operationalizing analytics and remote execution:

- **One-box configuration**: One web node and one compute node run on a single machine. Set-up is a breeze. This configuration is useful when you want to explore what it is to operationalize R analytics using R Server. It is perfect for testing, proof-of-concepts, and small-scale prototyping, but might not be appropriate for production usage.

- **Enterprise configuration**: Multiple nodes are configured on multiple machines along with other enterprise features. This configuration can be scaled up or down by adding or removing nodes. This is likely the more permanent implementation of R Server for your organization as it utilizes enterprise-grade security while allowing for scalability above that of the one-box configuration.

For more information about enterprise configuration, follow this link.

Security should always be a top priority when it comes to your data. Luckily, R Server's operationalization feature offers seamless integration with popular enterprise security solutions like Active Directory LDAP or Azure Active Directory. You can configure R Server to authenticate using these methods to establish a trust relationship between your user community and the operationalization engine for R Server.

Your users can then supply simple `username` and `password` credentials in order to verify their identity. User access to the R Server and the operationalization services offered on its API are entirely under your control as the server administrator.

In addition to authentication, you can add other enterprise security around Microsoft R Server such as:

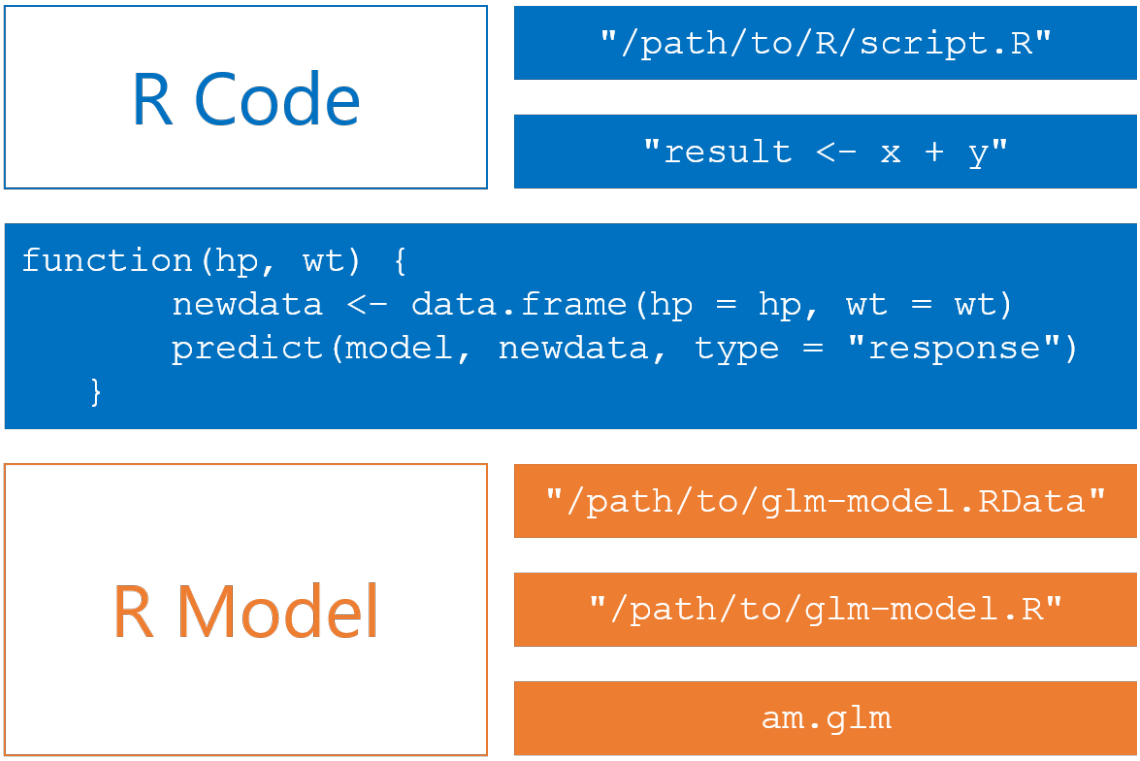
- Secured connections using SSL/TLS 1.2. (Strongly recommended for all production environments.)
- Cross-Origin Resource Sharing to allow restricted resources on a web page to be requested from another domain outside the originating domain.
- Role-based access control over web services in R Server.

For further security documentation, follow this link.

Create a Web Service

Step 1 - Ready your model

Web services offer fast execution and scoring of arbitrary R code and R models. They can contain R code, models, and model assets.



Step 2 - Authenticate with R Server

Before you can publish to the server, you must first login. Depending on your authentication method, you will either use `remoteLogin` or `remoteLoginAAD` to do so.

Use the `remoteLogin` function in these scenarios:

- Authenticating using Active Directory server on your network
- Using the default administrator account for an on-premises instance of R Server

```
> remoteLogin("http://localhost:12800",
username = "admin",
password = "",
diff = TRUE,
session = TRUE,
commandline = TRUE)
```

Use the `remoteLoginAAD` function if you are authenticating using Azure Active Directory in the cloud.

```
> remoteLoginAAD(
endpoint,
authuri = https://login.windows.net,
tenantid = "<AAD_DOMAIN>",
clientid = "<NATIVE_APP_CLIENT_ID>",
resource = "<WEB_APP_CLIENT_ID>",
session = TRUE,
diff = TRUE,
commandline = TRUE
)
```

Step 3 - Publish your model to the server as a web service

After you've authenticated, use the `publishService` function in the `mrsdeploy` package to publish a web service.

```
# Publish a standard service 'mtService' version 'v1.0.0'
# Assign service to 'api' variable
api <- publishService(
"mtService",
code = manualTransmission,
model = carsModel,
inputs = list(hp = "numeric", wt = "numeric"),
outputs = list(answer = "numeric"),
v = "v1.0.0"
)
```

Step 4a - Call your model through the web service with R

Once your model has been published to the server as a web service, you (and others) may now consume it. Using the getService function, you can retrieve any services with which you are allowed to interact.

```
# Get service using getService() function from `mrsdeploy`.
# Assign service to the variable `api`
api <- getService("mtService", "v1.0.0")

# Start interacting with the service by calling it with the
# generic name `consume` based on I/O schema
result <- api$consume(120, 2.8)

# Or, start interacting with the service using the alias argument
# that was defined at publication time.
result <- api$manualTransmission(120, 2.8)
```

For more code examples, follow this link.

Step 4b - Get your Swagger on. (Using RESTful APIs with .JSON)

In addition to calling the web service from within R, you can also access these RESTful APIs by using a Swagger .JSON file. Use a Swagger code tool to generate an API client library that can be used in any programming language, such as .NET, C#, Java, JavaScript, Python, or Node.js. The API client simplifies the making of calls, encoding of data, and markup response handling on the API.

R Server provides several Swagger templates each defining the list of resources that are available in the REST API and the operations that can be called on those resources. Additionally, another unique Swagger-based .JSON file is also generated for each and every web service version that is published.

For more information on Microsoft R Server web services for use in app development, follow this link.

As you can see, in just a few lines of code, your model transforms from just an object on one person's machine to a callable service for everyone. This capability helps your predictive analytics team productionalize their work and cause a greater impact in the organization.

If you are a data scientist looking to take your work to the next level and put your models in motion, a great place to start is with Microsoft R Server. You can read more about it on Microsoft's post, Get Started for Data Scientists. Still have questions? Contact us!