CTA Algorithm 2, R Examples

Last Updated: November 23, 2016; First Released: January 09, 2015

Author: Kevin Boyle, President, DevTreks

Version: DevTreks 2.0.6

A. Overview

The sibling reference, Conservation Technology Assessment (CTA), introduces the background numerical techniques for completing CTAs. This reference introduces examples of CTAs completed using Algorithm 2, R Algorithms.

This algorithm relies on the R Project programming language to run R Project scripts. The R Project is an open source programming language that contains mathematical, statistical, and graphing libraries that support the analysis of scientific data. The following URLs demonstrate using the R Project statistical library to carry out typical statistical analyses.

http://astrostatistics.psu.edu/datasets/R/html/stats/html/lm.html

http://azure.microsoft.com/en-us/documentation/articles/machine-learning-r-quickstart/

Although the URLs demonstrate using the R Project directly, this algorithm programmatically runs these types of R Project scripts for Data URL datasets and then summarizes the statistical results in the Indicator.MathResult property. The scripts and datasets must be developed and tested in the R Project by algorithm authors prior to using this algorithm. Datasets and scripts must be cleaned up and tested prior to using them with any CTA algorithm. The same versions of the statistical packages documented in **Section B** must be used.

In the case of Resource Stock calculations, this algorithm requires that paths to R script files, *with ".txt" extensions*, be added to the Joint Data URL property. Monitoring and Evaluation (M&E) calculators use the Score.URL property. The Python-compatible datasets are referenced

using the Data URL property of both tools. This algorithm does not require that datasets follow the standard explained in the *Resource Stock and M&E Calculation* references. Instead, the dataset should follow R Project conventions for csv files (i.e. a header row followed by the raw data). The Data URL dataset must correspond to one and only one Indicator but multiple datasets can be used by following the semicolon delimiter convention. The dataset will be matched with that Indicator based on the dataset column names along with the standard Ix.Qx.ColName convention used in the Indicator.MathExpression.

The current version runs R Project scripts using the following techniques.

1. **Microsoft R Open 3.3.0 with Intel Math Kernel Library (MKL) (1*):** R Open is Microsoft's open source version of R. Documentation for Microsoft R Open can be found in the References. Advantages over regular R include multithreaded math operations, a date-specific mirrored CRAN repository for R Packages, and direct access to MKL. R Open is installed on appropriate web servers or cloud virtual machines. R packages needed by specific scripts are also installed. R scripts are run by executing Rscript.exe through a programmatic command line. The path to Rscript.exe is set in the web project's appsettings.json file. If for some reason ROpen can't be used, the regular R statistical package can also be used –just make sure the path to RScript.exe is accurate. Example 1 in **Appendix A** demonstrates this technique.
2. **Statistical Virtual Machine (2*)**: Azure cloud computing centers have preconfigured virtual machines that include R Server and other machine learning packages (i.e. Azure Data Sciences Virtual Machine or ADSVM**)**. Item 5 discusses R Server further. Fees are charged only for the virtual machine, not for the software on the machine. Examples 2 and 3 in **Appendix A** demonstrates using respective ASP.NET Core 1 WebApis to run R scripts using this technique.
3. **Azure Machine Learning (AML) web service with R algorithms**: The sibling reference, CTA 4 AML Algorithms, has examples demonstrating how to use R with an AML web service.

4. **R Studio and Shiny:** R Studio includes R Server, which is Microsoft's server version of R. Shiny is a web browser UI for R Studio. Both "free" and commercial licenses of R Studio are available. R Server runs on Item 2's virtual machine. Standalone examples that use both technologies are not available for this release.

The current default configuration uses Technique 1 for web server deployments and Techniques 2 and 3, Statistical Virtual Machines, for cloud sites. Future releases will investigate using more installation techniques as well as supporting additional R features (i.e. custom packages, chained scripts and datasets, graphical displays).

The goal of most scripts will be to produce confidence intervals for an Indicator's QTM, QTL, and QTU properties. Most subalgorithms generate those properties from the last csv row of the results.

B. Subalgorithms and Examples

Appendix A has examples demonstrating each of the following subalgorithms.

subalgorithm1: Scripts: Runs generic R scripts where R Open or R "standard" has been installed on the same server as the web app.

subalgorithm2: WebApi Scripts, Small Jobs: Runs generic R scripts where R has been installed on another server where an ASP.NET Core 1 WebApi web app, DevTreksStatsApi, has been deployed. DevTreks is the client that consumes the response from the WebApi. This technique is appropriate for RScripts and datasets that return results to the client quickly. This technique is cross platform –the WebApi server can be Windows, Linux, or Mac.

subalgorithm3: WebApi Scripts, Large Jobs: [Uses the same WebApi as subalgorithm2 but instead of responding immediately to the client with the statistical results, returns a message confirming receipt of the request. The R scripts use RServer to run R on very large datasets. Example needed.]

subalgorithm4: Custom Packages: [Custom R package example needed].

subalgorithm5: R Mathematical Libraries: [Intel Math Kernel example needed].

Summary

This reference demonstrates how to use R algorithms to complete CTAs. CTAs may help people to reach decisions that improve their lives and livelihoods.

Footnotes

1. The security implications of allowing users to run generic R scripts has not been fully investigated yet.
2. The following URL demonstrates that inexpensive Linux web servers can also be used to host the Statistical Virtual Machine. Although any statistical package can be used on the virtual machine, the current source only supports R. Python, Julia, and AML. http://www.hanselman.com/blog/PublishingAnASPNETCoreWebsiteToACheapLinuxVMHost.aspx

References (URLs were last accessed September, 2016)

R Project. Version 3.3.0.

http://www.r-project.org/

Microsoft R Open 3.3.0

https://msdn.microsoft.com/en-us/microsoft-r/r-open

https://mran.revolutionanalytics.com/open

Microsoft R Server.

https://msdn.microsoft.com/en-us/microsoft-r/rserver

Azure Data Sciences Virtual Machine:

https://azure.microsoft.com/en-us/documentation/articles/machine-learning-data-science-provision-vm/

https://www.visualstudio.com/en-us/docs/release/examples/azure/net-to-azure-vm

WebApi web app:

https://docs.asp.net/en/latest/tutorials/first-web-api.html

http://www.asp.net/web-api

http://www.asp.net/web-api/overview/advanced/calling-a-web-api-from-a-net-client

References Note

We try to use references that are open access or that do not charge fees.

Improvements, Errors, and New Features

Please notify DevTreks (devtrekkers@gmail.com) if you find errors in these references. Also please let us know about suggested improvements or recommended new features.

A video tutorial explaining this reference can be found at:

https://www.devtreks.org/commontreks/preview/commons/resourcepack/Technology Assessment 1/1526/none

Appendix A. Algorithm 2 Examples.

Example 1. Algorithm 2. SubAlgorithm1: R Project Scripts

URLs:

These datasets are owned by the Natural Resource Stock club in the GreenTreks network group. If testing on localhost, switch to the Carbon Emission Reduction club.

https://www.devtreks.org/greentreks/preview/carbon/input/R OLS 1/2147397537/none

R project Data URL dataset
https://devtreks1.blob.core.windows.net/resources/network_carbon/resourcepack_1534/resource_7969/Regress1.csv

R project Stock.JointDataURL or M&E.Score.URL script files
https://devtreks1.blob.core.windows.net/resources/network_carbon/resourcepack_1534/resource_7963/R1Web.txt

Media URL graphical plots
https://devtreks1.blob.core.windows.net/resources/network_carbon/resourcepack_1534/resource_7965/Ex8Plot1.PNG

The following URL contains both Resource Stock and M&E calculators that run this example. The URL demonstrates that, outside of the Stock.JointDataURL, and M&E.ScoreURL properties, no changes were needed to run both calculators.

http://localhost:5000/greentreks/preview/carbon/input/Example 2, R Web Regression/2147409824/none

The first example is for the same linear regression that is used in Example 1h in the CTA 1 reference. The dependent variable in the following regression is *household energy consumption* and the independent variables are *household size* and *household size squared*.

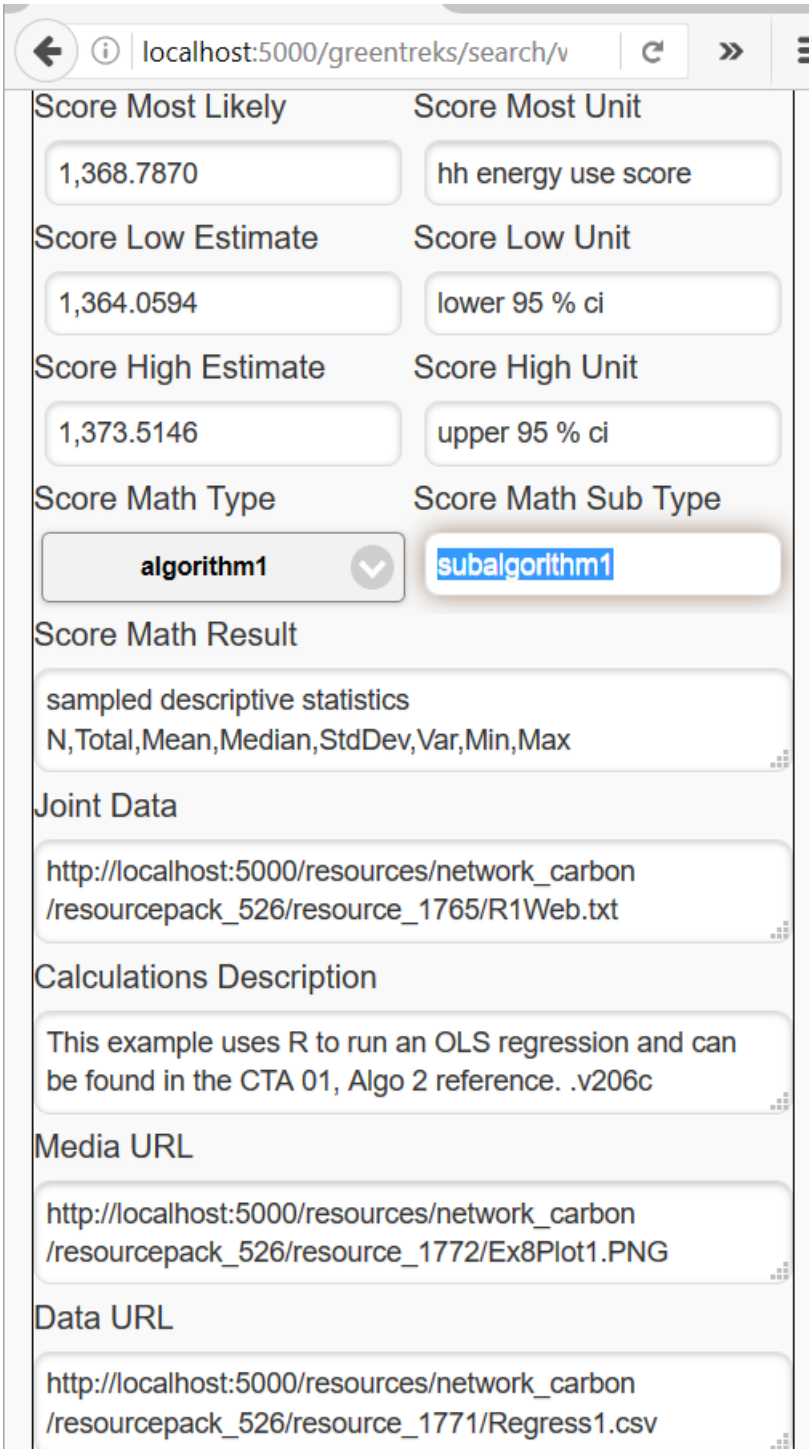
Abstract equation: y = B0 + B1x1 + B2x1^2 + e
Math Expression: I1.Q1.housesize1 + I1.Q2.housesize2

This script uses Technique 1 to run R directly on web servers. The author admits to being rusty with R scripts. The script is passed a Data URL in the first argument and produces the statistical results as a formatted string. The last 3 rows of data are not used in the regression, but are used to carry out a small sensitivity analysis of the confidence interval for the Qx variables (see Example1h in the CTA 1 reference). The final row is used to fill in QTM, QTL, and QTU.

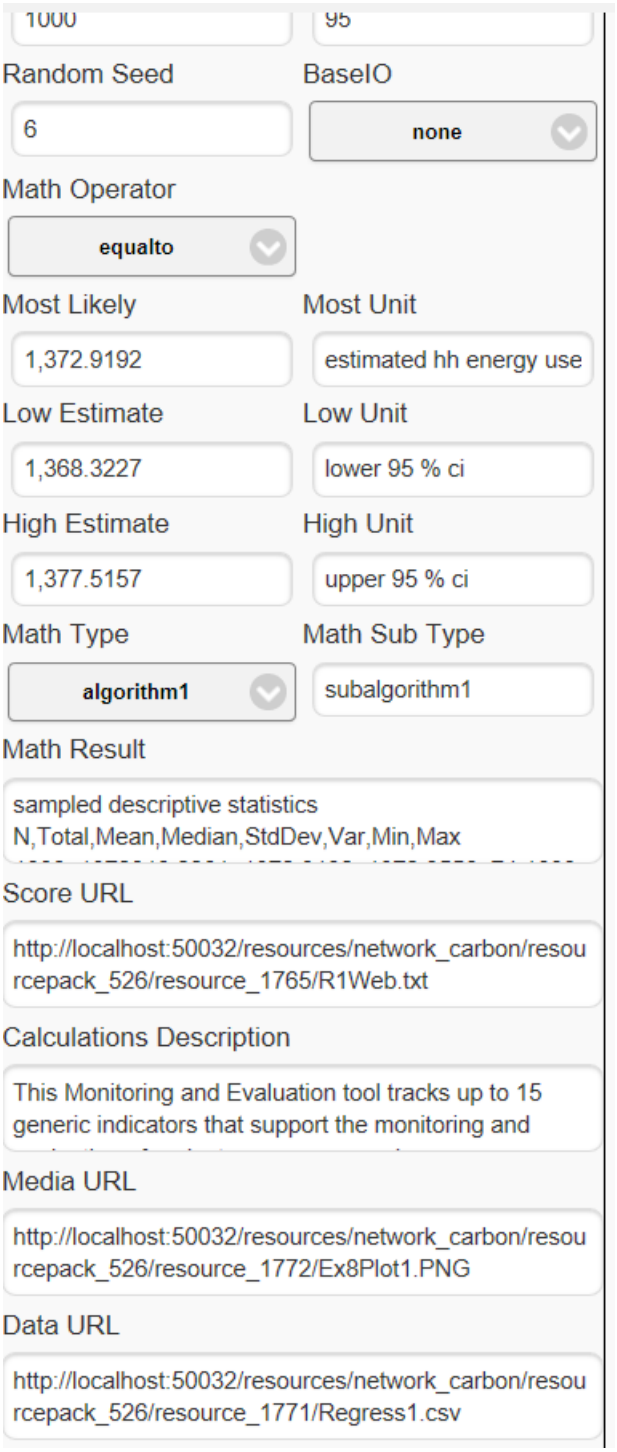
```
args <- commandArgs(TRUE)
url <- args[1]
print(url)
dataset1 <- read.table(url, header=TRUE, sep=",")
#dataset2 is dataset1 minus the last 3 lines of data
dataset2 <- head(dataset1, n=-3)
#dataset3 is the last 3 lines of data used for ci
dataset3 <- tail(dataset1, n=3)
model <- lm(energyuse ~ housesize1 + housesize2, data=dataset2)
f1 <- summary(model)
print(f1)
ci <- predict(model, dataset3, interval='confidence')
print(ci)
```

The following image shows the correct file path conventions to follow using Technique 1 with a localhost Intranet deployment in a Resource Stock Input calculator. The Score.JointDataURL property is used to hold the script file. The script file URL ends with a ".txt" extension and is

stored in a Resource. The dataset URL ends with a ".csv" extension and is stored in a Resource. No connection to the Internet is needed (i.e. because localhost URLs can be used).



The following image shows the correct file path conventions to follow using Technique 1 with a localhost Intranet deployment in an M&E Input calculator (localhost:50032 is the development server). The Score.URL property is used to hold the script file.



The only properties that are used differently than Example 1h in the CTA 1 reference are as follows:

- **QTM, QTL, and QTU**: All R scripts must include confidence intervals for these properties in the last row of data. That csv-delimited, or space-delimited, string is parsed by the algorithm to generate the Indicator's QTM, QTL, and QTU properties. The scripts above demonstrate one way to generate the interval. Alternatively, any Indicator or Score property can be manually set.
- **Math Type: algorithm2 (rproject) and Sub Math Type: subalgorithm1 (scripts):** Run generic R scripts.
- **Math Expression**: Follows the exact same format as Example 1h in the CTA 1 reference. The Math Expression is coupled with the Data URL dataset column names to determine which Indicator to update with the calculations. At least one Indicator.MathExpression must contain all of the dataset independent variable column names. The dependent variable is not included in Math Expressions.

  I1.Q1.housesize1 + I1.Q2.housesize2
- **Math Result**: The R project dataframe output will be saved as a csv TEXT file. If the initial Math Result has a URL to a Resource TEXT file, the csv TEXT file will be stored in the URL. If a URL is not found, the TEXT file will be added to the Math Result property and formatted into columns. The last line of csv text is used to complete QTM, QTL, and QTU. Large mathematical results should always be stored in Math Result URLs.

The following properties show how Score properties are set.

- **Data URL**: The dataset used in Example 1h in the CTA 1 reference has been converted to an R project-compatible csv dataset and referenced using this property. The independent variable column names must match the column names found in one Indicator.MathExpression. Each dataset corresponds to one and only one Indicator (unlike standard datasets). The last 3 rows of data in the dataset are used to score the statistical model that is trained using the remaining rows of data. These 3 rows will be used to conduct a sensitivity analysis of the confidence interval for the estimated dependent variable.

- **Stock.JointData URL or M&E.Score.URL**: This property stores URLs to one or more R script files that are saved as plain text file with ".txt" extensions in Resource elements. Some scripts may contain characters that translate badly to source code strings (i.e. double quotes around strings). Although error messages use terms like "bad request", any error should first assume that a script contains characters that cause the error. The cloud script must produce an R dataframe as the result. The web server script must produce a formatted string as the result.
- **Media URL**: These media files include images of graphical plots of the analysis which are generated in R Studio, uploaded to Resource elements, and then referenced here.
- **Remaining Score Properties:** Scores derive from uncertain Indicators and are themselves uncertain. For illustrative purposes, these properties were set to return a similar confidence interval to the sole Indicator being scored. Real scores derive from multiple Indicators and can have probability distributions that are different than the normal distributions used by regression Indicators. The M&E calculators support setting scores using separate datasets. The Stock calculators do not currently support using datasets to set Scores therefore their uncertainty must be analyzed using similar techniques (i.e. algorithm1, subalgorithms 1 to 4).

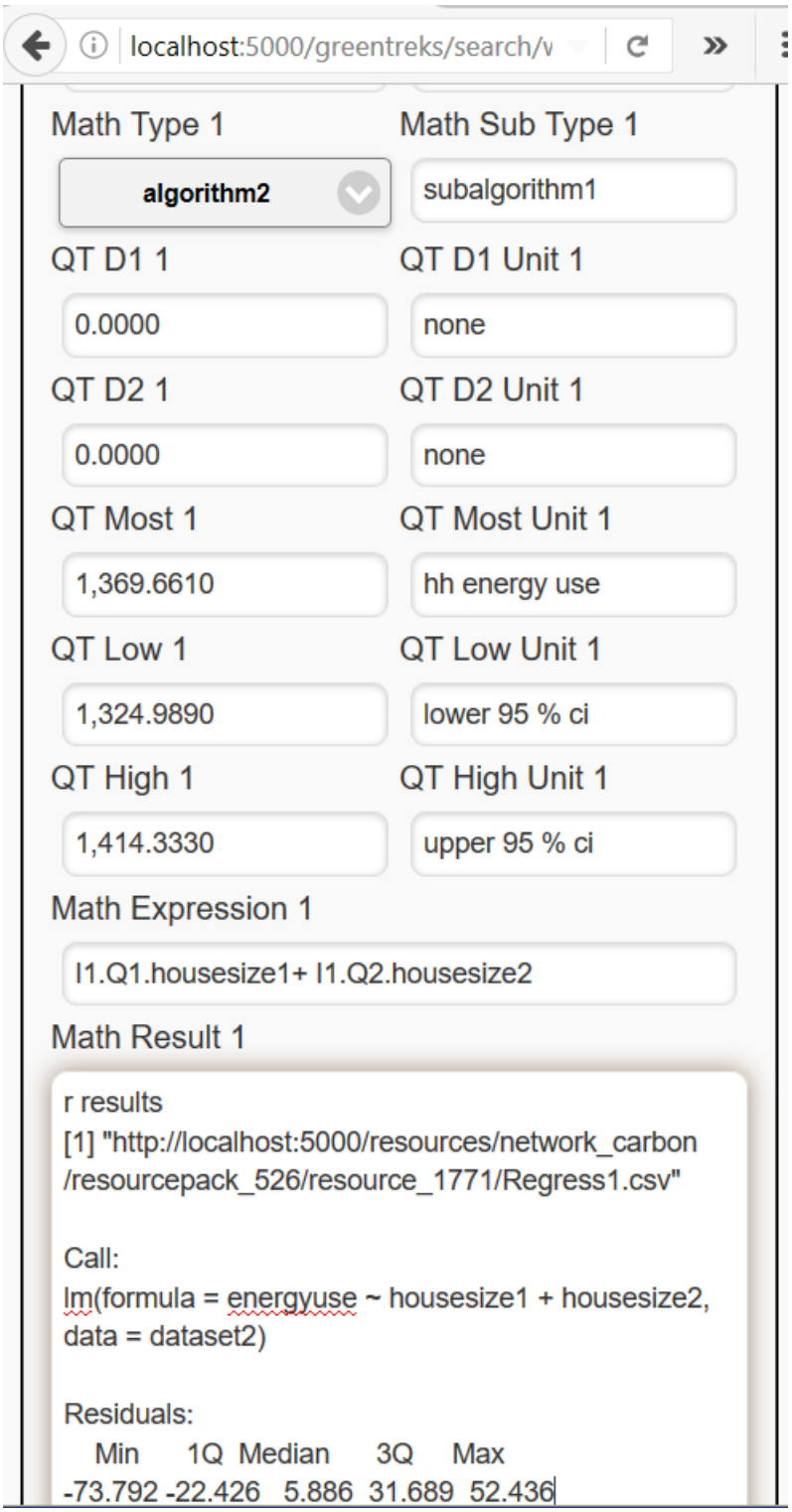
The calculator uses the following steps:

- **Step 1**. Run an asynchronous loop that simultaneously iterates through each dataset referenced in Data URL. Pass the URL to the algorithm and run the corresponding R Project script for the Indicator corresponding to the iteration loop. The cloud site can be slower than the web server executed script, so be patient. Errors with calculations will be added to the Math Result property of each Indicator. To the extent possible, algorithms with multiple datasets run their calculations asynchronously and simultaneously.
- **Step 2**. On cloud sites, the R project script dataframe output will be added to a csv output file. That file will be parsed and displayed in the Indicator.MathResult. Web server sites display the string generated by the R project script.

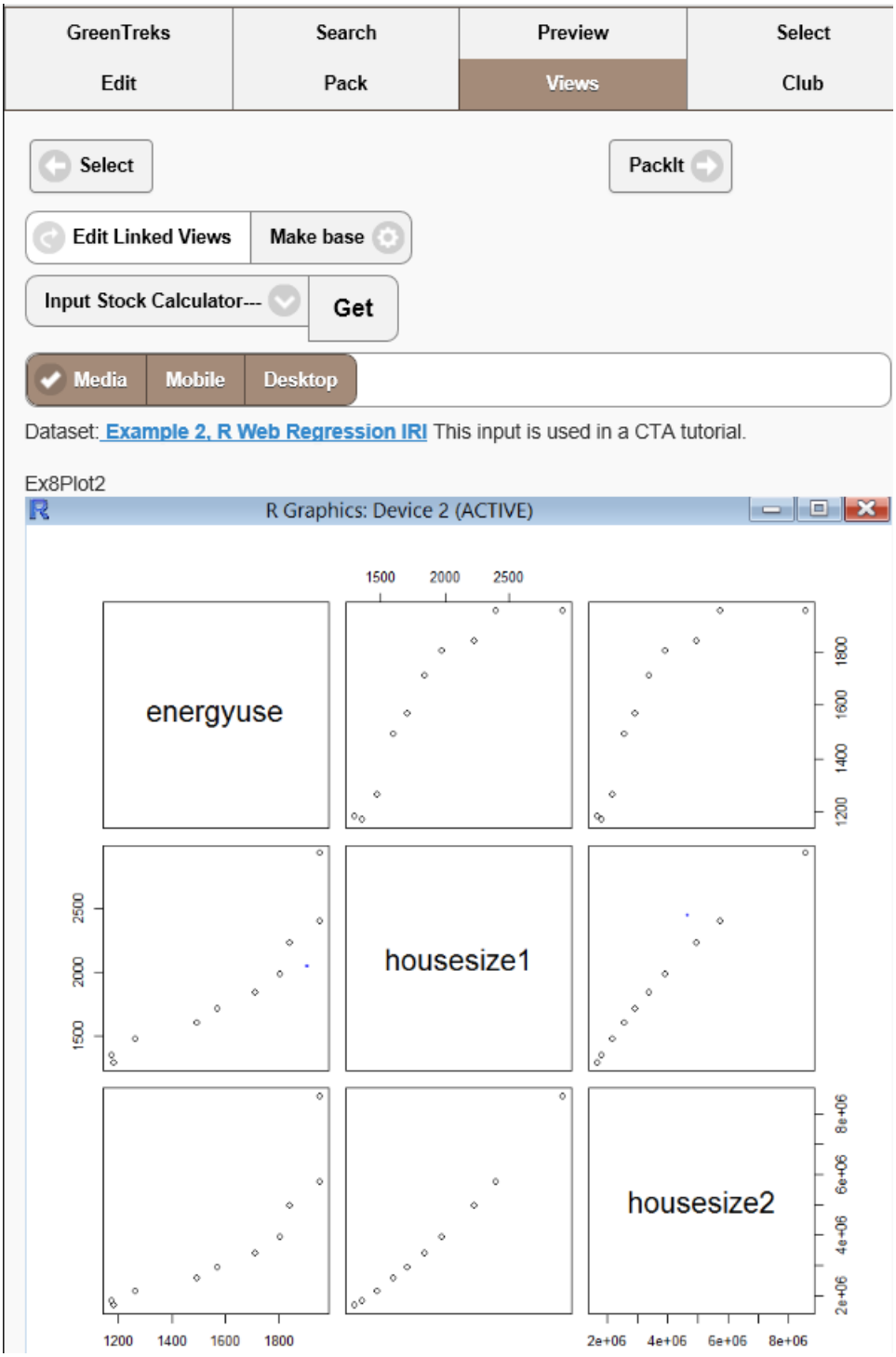
- **Step 3**. Add the results of the last line of the output file to each Indicator's QTM, QTL, and QTU properties. Cloud sites parse a csv string and web sites parse a csv or space-delimited string for this purpose.
- **Step 4**. Set the ScoreM property from the result of the Score.MathExpression.

The following results are similar to the results in Example 1h in the CTA 1 reference. The images demonstrates that numbers should be formatted correctly inside the R script because the algorithm does not format any of the statistical results. The Technology Assessment 2 tutorial explains that large mathematical results should always be stored in Math Result URLs.

Web server results (run using Version 2.0.6):


localhost:5000/greentreks/search/v
Math Type 1
algorithm2
Math Sub Type 1
subalgorithm1
QT D1 1
0.0000
QT D1 Unit 1
none
QT D2 1
0.0000
QT D2 Unit 1
none
QT Most 1
1,369.6610
QT Most Unit 1
hh energy use
QT Low 1
1,324.9890
QT Low Unit 1
lower 95 % ci
QT High 1
1,414.3330
QT High Unit 1
upper 95 % ci
Math Expression 1
I1.Q1.housesize1+ I1.Q2.housesize2
Math Result 1
r results
[1] "http://localhost:5000/resources/network_carbon/resourcepack_526/resource_1771/Regress1.csv"
Call:
lm(formula = energyuse ~ housesize1 + housesize2, data = dataset2)
Residuals:
Min 1Q Median 3Q Max
-73.792 -22.426 5.886 31.689 52.436

The following image demonstrates that graphical explanations of analytic results can be produced directly in R and then displayed using the Media View.



Example 2. Algorithm 2. SubAlgorithm2: WebApi R Scripts, Small Jobs

URLs:

The URLs from Example 1 are also used to test this algorithm. The only difference is to change the Indicator.MathType to algorithm2 and Indicator.SubMathType to subalgorithm2.

This subalgorithm is appropriate when the R script and associated dataset run relatively fast –the results return to the client relatively quickly. The term "relatively quickly" should be defined by the developer in an exact way.

A cross-platform, ASP.Net Core 1 WebApi web app, DevTeksStatsApi, is deployed to another web server, including Windows, Linux or Mac web servers. The Web API implements a REST method that accepts http POST requests that contain a JSON object. The JSON object is serialized into a regular POCO object, StatScript, with properties that include a script file URL, a data file URL, and an output file URL. Paths to the web server's Rscript.exe has been added to StatScript using dependency injection which has been configured in the Startup.cs class and appsettings.json.

The server processes the http POST by running the statistical scripts. The statistical results are saved in the output csv URL. The client, DevTreks, waits for a successful response from the WebApi call. When received, DevTreks deserializes the json response into a StatScript POCO object and fills in an Indicator's QT properties from DataScript.StatisticalResults. If the MathResult contains a legitimate Resource URL, the results will be saved in the URL. If not, the results will be stored in the MathResult itself. The recommended way of storing statistical results is in MathResult URLs.

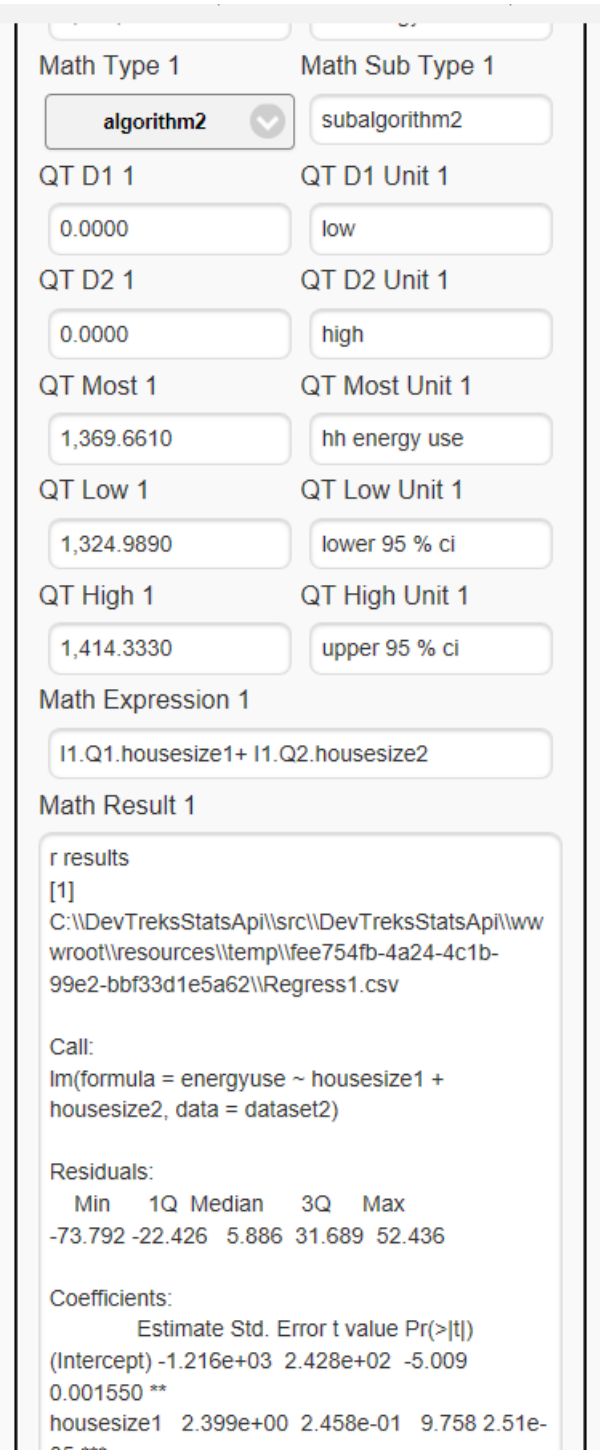
The WebApi host stores the data URL, script URL, and output URL, files in the standard "wwwroot\resources\temp" directory of the WebApi site. The WebApi folder and files must be maintained by DevTreks adminstrators by periodically deleting old directories. A future release may automate this task (i.e. see the Controller Action that runs DELETE http commands).

The following results show that this subalgorithm produces the same results as Example 1.

Version 2.0.6 discovered that this subalgorithm can't be fully debugged using localhost if both DevTreks and DevTreksStatsApi use localhost:5000 –only 1 application at a time can be run on a localhost server. In addition, files stored on localhost:50032 presents challenges in how files are loaded by the webapi. Previous releases used the cloud's App Service sites for debugging.

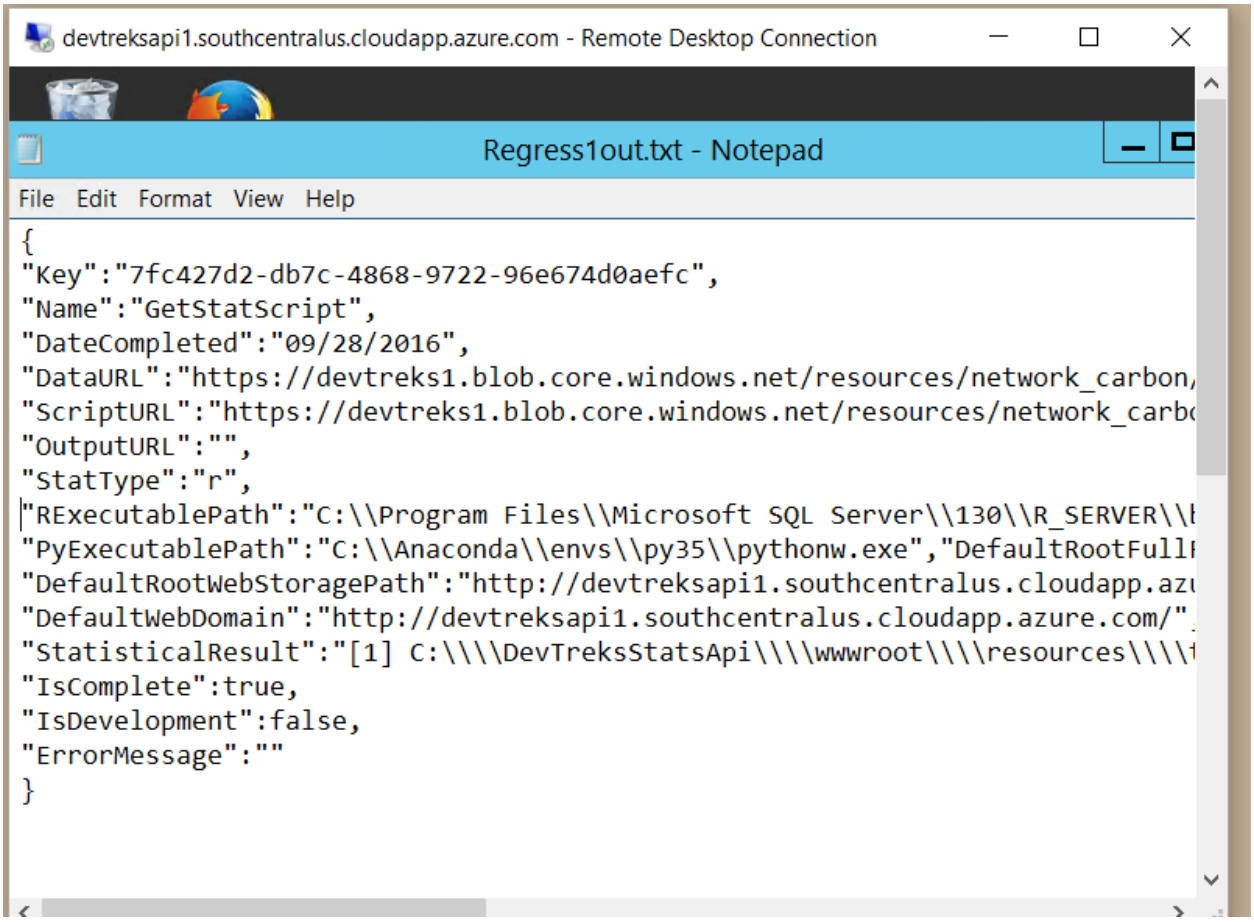
Further 2.0.6 testing also discovered that the algorithm doesn't run successfully on the cloud site. The WebApi virtual machine was "turned off" to save money after the Version 2.0.2 tests. Not enough is understood about the server/api calls yet to understand what happened. One explanation is that the WebApi app has to be reinstalled.

Given the 2.0.6 focus on the M&E tools rather than the CTA tools, a future release will address improvements in localhost debugging and provide additional documentation about how to run WebApi apps on virtual machines.



Cloud server json results:

The following image displays the json response generated from the WebApi deployed on a Windows Data Sciences Virtual Machine. This response is deserialized by a DevTreks client to produce the previous image's results. Linux will be tested in a future release.



Example 3. Algorithm 2. SubAlgorithm3: WebApi R Scripts, Large Jobs

URLs:

Example needed

This subalgorithm is appropriate when the R script and associated dataset take a long time to process. It uses the same DevTreksStatsApi WebApi as Example 2, but it uses RServer to run large datasets that require a large amount of time to process. The following URL gives examples of R scripts and datasets that would be appropriate to use with this subalgorithm.

https://msdn.microsoft.com/en-us/microsoft-r/scaler-getting-started

This example may be put on hold. It's not clear that a small ngo should be demonstrating machine learning techniques that investors are currently pouring large sums of money into. On the other hand.

Example 4. Algorithm 2. SubAlgorithm4: R Project Custom Packages

URLs:

Example needed

Example 5. Algorithm 2. SubAlgorithm5: R Mathematical Libraries

URLs:

Example needed