Data Science and Machine Learning Essentials

Lab 2B – Transforming Data with Scripts

By Graeme Malcolm and Stephen Elston

Overview

In this lab, you will learn how to use Python or R to manipulate and analyze data in Azure ML. If you intend to work with Python, complete the *Transforming Data with Python* exercise. If you plan to work with R, complete the *Transforming Data with R* exercise. Unless you need to work in both languages, you do not need to try both exercises.

Note: This lab builds on knowledge and skills developed in the preceding labs in this course. If you have little experience with Azure ML, and you did not complete the previous labs, you are advised to do so before attempting this lab.

What You'll Need

To complete this lab, you will need the following:

- An Azure ML account
- A web browser and Internet connection
- The lab files for this lab
- Python Anaconda or R and RStudio
- A **CA Dairy Data** dataset (see *Prepare the Data* steps below)

Note: To set up the required environment for the lab, follow the instructions in the **Setup** document for this course. Then download and extract the lab files for this lab.

Prepare the Data

1. Open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new dataset named **CA Dairy Data** by uploading the **cadairydata.csv** file from the folder where you extracted the lab files.

Transforming Data with R

R is a programming language that is commonly used to perform statistical transformations and visualizations of data. In this exercise, you will use R to transform dairy production data from the state of California.

Note: If you prefer to work with Python, complete the next exercise, *Transforming Data with Python* and skip this exercise.

Create an R Script

In this procedure, you will create R code to manipulate columns in a dataset.

1. Start RStudio, and create a new empty project in a new directory named **Dairy** in the folder where you extracted the lab files for this course.
2. In the **Console** pane, enter the following command to install the **dplyr** library. If you are prompted to use a personal library, click **Yes**.

```
install.packages('dplyr', dep = TRUE)
```

3. When the dplyr package has been installed, add a new R script file to the project, and save it as **TransformDairyData.R**. Then add the following code to the script file (replacing ***C:/DAT203xLabfiles*** with the path to the folder where you extracted the lab files). You can find this code in **TransformDairyDataR.txt** in the folder where you extracted the lab files.

```
## Set a flag to define the environment
Azure = FALSE

if(Azure){
  ## If in Azure, read the input data table into a data frame
  frame1 <- maml.mapInputPort(1)
} else {
  ## If in RStudio read the local .csv file
  dirName <- "C:/DAT203xLabfiles"
  fileName <- "cadairydata.csv"
  infile <- file.path(dirName, fileName)
  frame1 <- read.csv( infile, header = TRUE, stringsAsFactors = FALSE)
}

## Select a subset of columns
library(dplyr)
frame1 <- select(frame1, Year, Month, Cottagecheese.Prod, Icecream.Prod,
Milk.Prod)

# chain verbs to show totals for August
frame1 <- frame1 %>%
  filter(Month == 'Aug') %>%
  mutate(Total.Prod = Cottagecheese.Prod + Icecream.Prod + Milk.Prod)

## If in Azure output the data frame.
if(Azure) maml.mapOutputPort('frame1')
```

Note that this code is designed to be tested in a local RStudio environment, and when ready, it can be copied to an **Execute R Script** module in Azure ML and run successfully there simply by changing the **Azure** Boolean variable from **FALSE** to **TRUE**. This is a commonly used technique when developing and testing R code for use in Azure ML.

The code uses the dplyr **select** method to retrieve a subset of columns from the dairy production dataset, and then chains the **filter** and **mutate** methods to generate a dataset that shows the total production figures for August each year.

4. Select all of the code in the **TransformDairyData.R** pane and click **Run** to run this code. You can ignore any warnings about some objects being masked. Then in the **Console** pane, enter the following command to test the script and display the first five rows of the data frame it generates:

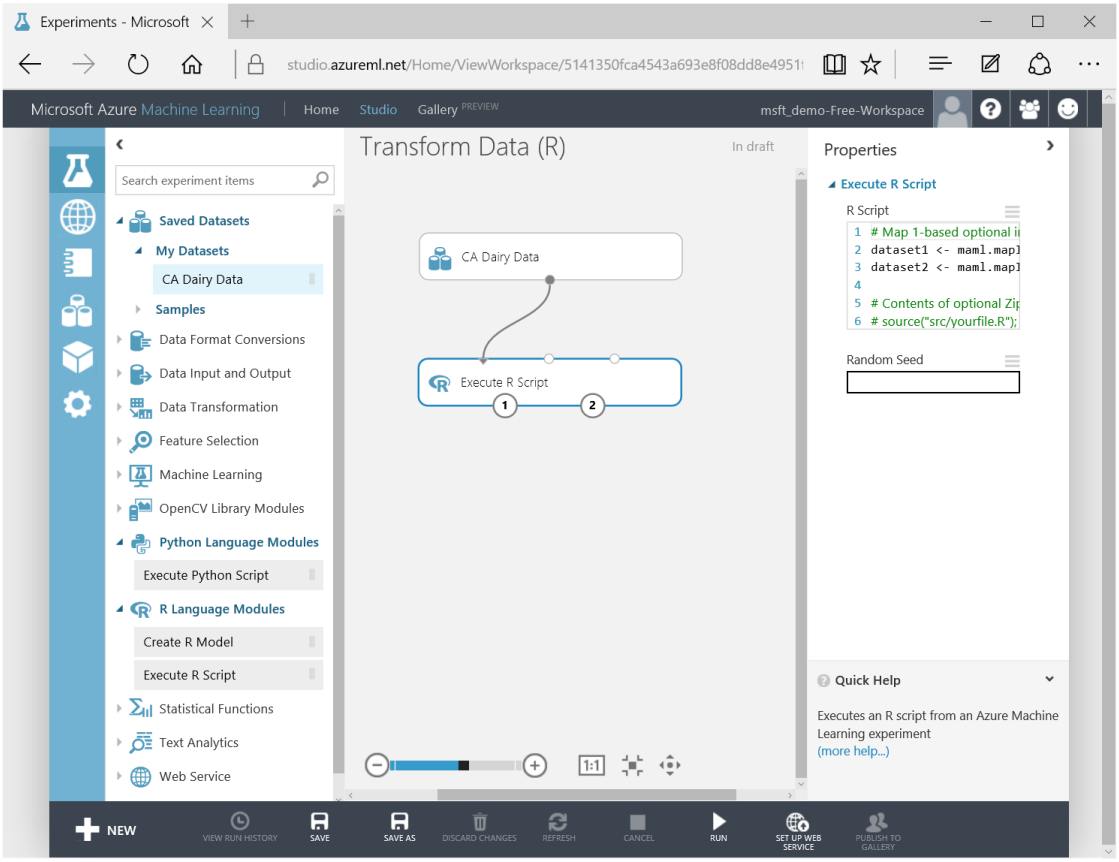
```
frame1[1:5,]
```

5. View the results generated by the script, which should include the **Year**, **Month**, **Cottagecheese.Prod**, **Icecream.Prod**, **Milk.Prod**, and **Total.Prod** values for August each year, similar to this:

```
  Year Month Cottagecheese.Prod Icecream.Prod Milk.Prod Total.Prod
1 1995   Aug               4.368        74.981     2.152     81.501
2 1996   Aug               3.979        79.260     2.129     85.368
3 1997   Aug               3.633        75.258     2.366     81.257
4 1998   Aug               3.007        80.306     2.271     85.584
5 1999   Aug               3.018        82.679     2.602     88.299
```

Use the Script in an Azure Machine Learning Experiment

1. If you have not already done so, open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Transform Dairy Data (R)**.
3. Add your **CA Dairy Data** saved dataset to the experiment canvas.
4. Find the **Execute R Script** module, and drag it to the experiment canvas under the **CA Dairy Data** dataset. Then connect the output port from the **CA Dairy Data** dataset to the first input port of the **Execute R Script** module as shown in the following image:



5. Select the **Execute R Script** module, and in the **Properties** pane, replace the default R code with the code from the **TransformDairyData.R** script you created in RStudio.

> **Tip**: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the code from the clipboard, replacing the existing code.

6. Edit the R code in the **Properties** pane to change the statement `Azure = FALSE` to `Azure = TRUE`. This is required to use the data from the dataset instead of loading it from a local file.
7. Save and run the experiment.
8. When the experiment has finished running, visualize the output from the **Results dataset** output port of the **Execute R Script** module to verify that only the **Year**, **Month**, **Cottagecheese.Prod**, **Icecream.Prod**, **Milk.Prod**, and **Total.Prod** values for August each year are displayed. Then, close the results dataset.

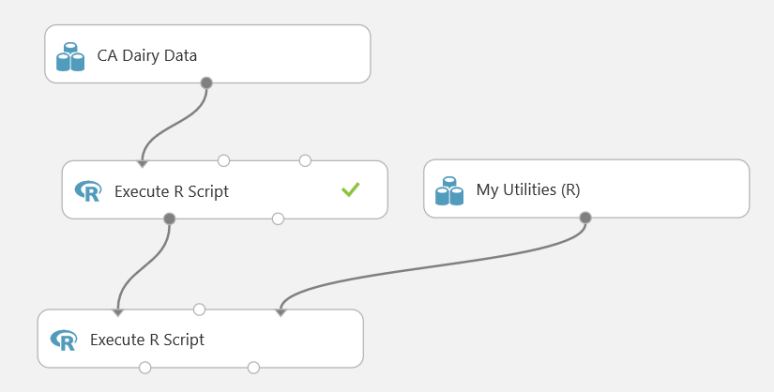
Create a Custom R Script Dataset

You can upload a zip file containing R code to create a custom dataset in Azure ML. This can be a useful technique when you may need to re-use the same script from multiple **Execute R Script** modules in the same experiment.

1. In RStudio, open the **myutilities.R** script in the lab files folder, and view the code it contains. The code contains a function named **round2** that rounds a numerical parameter to two decimal places, as shown here:

```
round2 <- function(x){
  ## Round values to 2 decimal places
  round(x, 2)
}
```

2. Close RStudio. Then, add the **myutilities.R** script file to a zip archive. If you are using Windows, you can do this by right-clicking the file in Explorer, clicking **Send to**, and clicking **Compressed (zipped) folder**. If you are using another operating system, you may need to install a compression utility.
3. In your browser, in Azure ML Studio, click **New**, and then click **Dataset**. Then click **From Local File** and upload the zipped archive containing the **myutilities.R** script file, assigning it the name **My Utilities (R)**.
4. In the **Transform Dairy Data (R)** experiment, drag a second **Execute R Script** module to the canvas and place it below the first **Execute R Script** module. Then connect the output dataset from the first **Execute R Script** module to the first input port of the second **Execute R Script** module.
5. Search for the **My Utilities (R)** dataset you added and drag it to the canvas next to the first **Execute R Script** module. Then connect the output port from the **My Utilities (R)** dataset to the **Script bundle (Zip)** input port of the second **Execute R Script** module as shown in the following image.



6. Select the second **Execute R Script** module, and in the **Properties** pane, replace the existing R script with the following code (which you can copy from **UseMyUtilities_R.txt** in the lab files folder). This code calls the **round2** function in the zipped **myutilities.R** script you uploaded.

```
frame1 <- maml.mapInputPort(1)
source("src/myutilities.R")

numCols <- c("Cottagecheese.Prod", "Icecream.Prod", "Milk.Prod",
"Total.Prod")
## Call the round2 function in the myutilities.R script
frame1[, numCols] <- lapply(frame1[, numCols], round2)
maml.mapOutputPort('frame1')
```

Note: This code uses the R **lapply** operator to call the function for each member of a list named **numcols**, which contains the numeric columns in the dataset. For more information about using lapply, enter the command **? lapply** in the console window in RStudio.

7. Save and run the experiment.
8. When the experiment has finished running, visualize the **Results dataset** output port of the **Execute R Script** module to view the results, and verify that the numeric values in the dataset have been rounded to two decimal places. Then close the results dataset.

Transforming Data with Python

Python is a versatile programming language that is commonly used to manipulate and transform data as part of a data science process. In this exercise, you will use Python to transform dairy production data from the state of California.

Note: If you prefer to work with R, skip this exercise and complete the previous exercise, *Transforming Data with R*.

Create a Python Script

In this procedure, you will create Python code to manipulate columns in a dataset.

1. Start Spyder. Then, if the Spyder interactive development environment (IDE) does not include an **IPython console** tab, on the **Consoles** menu, click **Open an IPython Console**.
2. Create a new code file, and save it as **ProcessDairyData.py** in the folder where you extracted the lab files.
3. In the code pane for the **TransformDairyData.py** script, enter the following code (replacing ***C:\\DAT203xLabfiles*** with the path to the folder where you extracted the lab files). You can find

this code in **TransformDairyDataPy.txt** in the lab files folder. The code defines a function named **azureml_main**.

```
def azureml_main(frame1):
  import pandas as pd
  import os.path

## Set a flag to define the environment
  Azure = False

## If in Azure, the data frame is passed to the function,
## If running in the IDE, load it from a local file
  if(Azure == False):
    pathName = "C://DAT203xLabfiles"
    fileName = "cadairydata.csv"
    filePath = os.path.join(pathName, fileName)
    frame1 = pd.read_csv(filePath)

## Select a subset of columns
  frame1 = frame1[["Year", "Month", "Cottagecheese.Prod", "Icecream.Prod",
"Milk.Prod"]]

## Filter and add a column to show totals for August
  frame1 = frame1[frame1['Month']=='Aug']
  frame1["Total.Prod"] = frame1["Cottagecheese.Prod"] + frame1["Icecream.Prod"]
+ frame1["Milk.Prod"]
  return frame1
```

Note that this code is designed to be tested in a local Python development environment, and when ready, it can be copied to an **Execute Python Script** module in Azure ML and run successfully there simply by changing the **Azure** Boolean variable from **False** to **True**. This is a commonly used technique when developing and testing Python code for use in Azure ML.

The code uses pandas operations to retrieve a subset of columns from the dairy production dataset, filter the data to include only records for August, and add a column containing the total production figure for each August.

Test the Script on the Local Computer

1. Select all of the code *except for the first and last lines* (which declare the **azureml_main** function and return the dataset). Then, on the toolbar, click **Run current cell** to run the selected code.
2. In the IPython console pane, enter the following code to return the first five rows of the **frame1** data frame generated by your code:

```
frame1[:5]
```

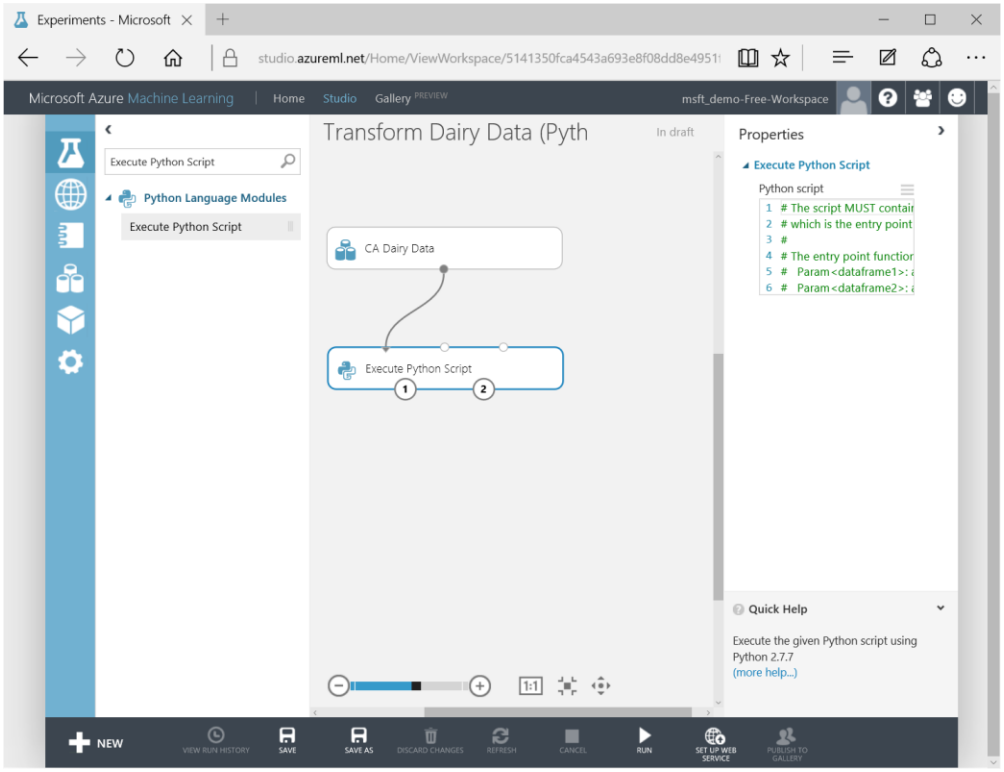
3. View the results generated by the script, which should include the **Year**, **Month**, **Cottagecheese.Prod**, **Icecream.Prod**, **Milk.Prod**, and **Total.Prod** values for August each year, similar to this:

```
    Year Month  Cottagecheese.Prod  Icecream.Prod  Milk.Prod  Total.Prod
7   1995   Aug               4.368         74.981      2.152      81.501
19  1996   Aug               3.979         79.260      2.129      85.368
31  1997   Aug               3.633         75.258      2.366      81.257
43  1998   Aug               3.007         80.306      2.271      85.584
55  1999   Aug               3.018         82.679      2.602      88.299
```

Use the Script in an Azure Machine Learning Experiment

1. If you have not already done so, open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Transform Dairy Data (Python)**.

3. Add your **CA Dairy Data** saved dataset to the experiment canvas.
4. Find the **Execute Python Script** module, and drag it to the experiment canvas under the **CA Dairy Data** dataset. Then connect the output port from the **CA Dairy Data** dataset to the first input port of the **Execute Python Script** module as shown in the following image:



5. Select the **Execute Python Script** module, and in the **Properties** pane, replace the default Python code with the **TransformDairyData.py** script you created in Spyder.

> **Tip**: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the code from the clipboard, replacing the existing code.

6. Edit the Python code in the **Properties** pane to change the statement `Azure = False` to `Azure = True`. This is required to use the data from the dataset instead of loading it from a local file.
7. Save and run the experiment.
8. When the experiment has finished running, visualize the output from the **Results dataset** output port of the **Execute Python Script** module to verify that only the **Year**, **Month**, **Cottagecheese.Prod**, **Icecream.Prod**, **Milk.Prod**, and **Total.Prod** values for August each year are displayed. Then, close the results dataset.

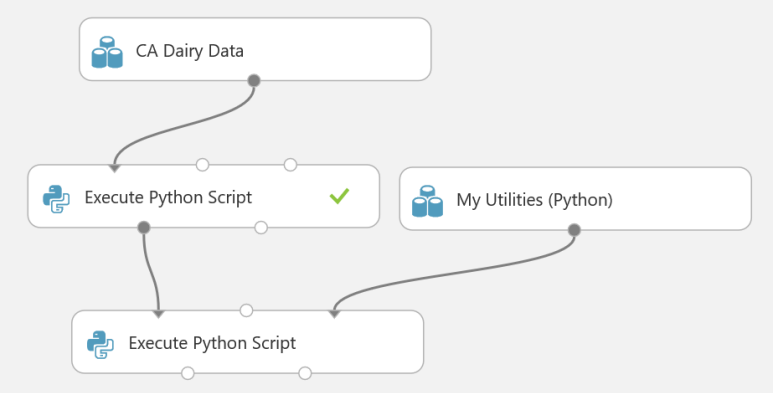
Create a Custom Python Script Dataset

You can upload a zip file containing Python code to create a custom dataset in Azure ML. This can be a useful technique when you may need to re-use the same function from multiple **Execute Python Script** modules in the same experiment.

1. In Spyder, open the **myutilities.py** in the folder where you extracted the lab files, and review the code it contains. contains a function named **round2** that rounds a numerical parameter to two decimal places, as shown here:

```
def round2(df):
## Round values to 2 decimal places
  import numpy as np
  return df.apply(lambda x: np.round(x, 2))
```

2. Close Spyder, and then add the **myutilities.py** script file to a zip archive. If you are using Windows, you can do this by right-clicking the file in Explorer, clicking **Send to**, and clicking **Compressed (zipped) folder**. If you are using another operating system, you may need to install a compression utility.
3. In your browser, in Azure ML Studio, click **New**, and then click **Dataset**. Then click **From Local File** and upload the zipped archive containing the **myutilities.py** script file, assigning it the name **My Utilities (Python)**.
4. In the **Transform Dairy Data (Python)** experiment, drag a second **Execute Python Script** module to the canvas and place it below the first **Execute Python Script** module. Then connect the output dataset from the first **Execute Python Script** module to the first input port of the second **Execute Python Script** module.
5. Search for the **My Utilities (Python)** dataset you added and drag it to the canvas next to the first **Execute Python Script** module. Then connect the output port from the **My Utilities (R)** dataset to the **Script bundle (Zip)** input port of the second **Execute Python Script** module as shown in the following image.



6. Select the second **Execute Python Script** module, and in the **Properties** pane, replace the existing Python script with the following code (which you can copy from **UseMyUtilitiesPy.txt** in the lab files folder). This code calls the **round2** function in the zipped **myutilities.py** script you uploaded.

```
def azureml_main(frame1):
  import myutilities as mu

  numCols = ["Cottagecheese.Prod", "Icecream.Prod", "Milk.Prod",
"Total.Prod"]
  ## Call the round2 function in the myutilities script
  frame1[numCols] = frame1[numCols].apply(mu.round2)
  return frame1
```

7. Save and run the experiment.

8. When the experiment has finished running, visualize the **Results dataset** output port of the **Execute Python Script** module to view the results, and verify that the numeric values in the dataset have been rounded to two decimal places. Then close the results dataset.

Summary

In this lab, you used a locally installed integrated development environment (IDE) to test R or Python code before deploying it to an Azure ML experiment. You then created a custom dataset from a zip file containing a script file, and used it from an **Execute Script** task.

Note: The experiment created in this lab is available in the Cortana Analytics library at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.