

Azure Machine Learning 之 R 程式設計語言的快速入門教學課程

透過 [Larry Franks](#)

更新日期：07/12/2016

本文內容：

簡介

與 [Machine Learning Studio](#) 中的 R 語言互動

將資料輸入執行 R 指令碼模組及從此模組輸出

資料篩選和轉換

時間序列物件和相互關聯分析

時間序列範例：季節性預測

附錄 A：RStudio 指南

附錄 B：進階閱讀

Stephen F Elston 博士

簡介

本快速入門教學課程將協助您使用 R 程式設計語言來快速開始擴充 Azure Machine Learning。請跟著 R 程式設計教學課程來於 Azure Machine Learning 中建立、測試及執行 R 程式碼。在您隨著此教學課程進行的過程中，您將在 Azure Machine Learning 中使用 R 語言來建立一個完整的預測解決方案。

Microsoft Azure Machine Learning 包含許多功能強大的機器學習和資料操作模組。功能強大的 R 語言被描述為分析通用語言。好消息是，在 Azure Machine Learning 中，分析和資料操作皆可藉由使用 R 來加以擴充。這個組合利用 R 的彈性和深入分析，讓 Azure Machine Learning 更具延展性且更易於部署。

免費試用 Azure Machine Learning

不需要信用卡或 Azure 訂用帳戶。[立即開始 >](#)

預測和資料集

預測是一個獲得廣泛採用且相當實用的分析方法。常見的用法範圍可從預測季節性項目的銷售額、判斷最佳的庫存量，一直到預測總體經濟變數。進行預測時通常是搭配時間序列模型。

時間序列資料係指其當中的值具有時間索引的資料。時間索引可以具規則性 (例如每個月或每分鐘) 或不具規則性。時間序列模型會根據時間序列資料。R 程式設計語言包含時間序列資料的彈性架構和廣泛分析。

在本快速入門指南中，我們將使用加州乳製品產量和訂價資料。此資料包含數項乳製品之產量及奶油 (基準商品) 價格的每月相關資訊。

本文中使用的資料以及 R 指令碼可[在此下載](#)。此資料原先是綜合自威斯康辛大學，網址為 <http://future.aae.wisc.edu/tab/production.html>。

組織

我們將循序進行數個步驟，讓您了解如何在 Azure Machine Learning 環境中建立、測試及執行分析和資料操作 R 程式碼。

- 首先，我們將探討在 Azure Machine Learning Studio 環境中使用 R 語言的基本概念。

- 然後，我們將接著討論 Azure Machine Learning 環境中資料 I/O、R 程式碼及圖形的各個方面。

- 再接著，我們會藉由建立可清理和轉換資料的程式碼，建構預測解決方案的第一個部分。
- 在備妥資料後，我們將執行資料集內數個變數之間的相互關聯分析。
- 最後，我們將針對牛奶產量建立季節性的時間序列預測模型。

與 Machine Learning Studio 中的 R 語言互動

本節將帶領您了解在 Machine Learning Studio 環境中與 R 程式設計語言互動的一些基本概念。R 語言提供一個功能強大的工具，可在 Azure Machine Learning 環境內建立自訂的分析和資料操作模組。

我將使用 RStudio 來進行小規模的 R 程式碼開發、測試及偵錯。然後，將此程式碼剪下並貼到 Machine Learning Studio 中已準備好要執行的**執行 R 指令碼**模組中。

執行 R 指令碼模組

在 Machine Learning Studio 中，R 指令碼是在**執行 R 指令碼**模組內執行。圖 1 顯示 Machine Learning Studio 中**執行 R 指令碼**模組的範例。

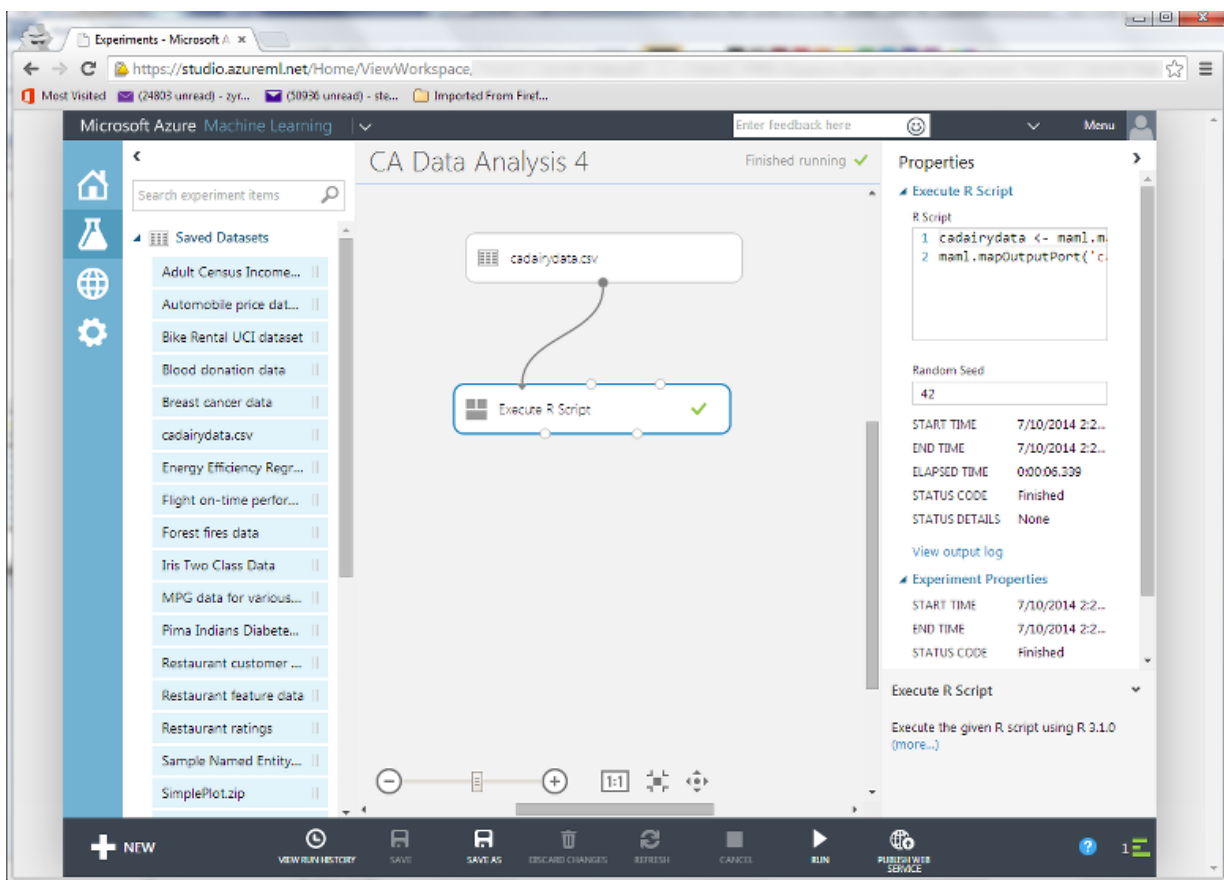


圖 1.顯示已選取 [執行 R 指令碼] 模組的 Machine Learning Studio 環境。

參照圖 1，讓我們看看 ML Studio 環境的一些與**執行 R 指令碼**模組搭配運作的主要部分。

- 實驗中的模組會顯示在中間的窗格。
- 右窗格的上半部包含一個可檢視和編輯 R 指令碼的視窗。
- 右窗格的下半部顯示**執行 R 指令碼**的一些屬性。您可以按一下此窗格上適當的點來檢視錯誤和輸出記錄檔。

當然，我們將會在這份文件的其餘部分更詳細地討論**執行 R 指令碼**。

使用複雜的 R 函式時，建議您在 RStudio 中進行編輯、測試及偵錯。與進行任何軟體開發相同，請以累加方式擴充您的程式碼，並在小型的簡單測試案例上進行測試。然後，將您的函式剪下並貼到**執行 R 指令碼**模組的 [R 指令碼] 視窗中。這個方法既可讓您控制 RStudio 整合式開發環境 (IDE)，也可讓您控制 Azure Machine Learning 的強大功能。

執行 R 程式碼

當您按一下 [執行] 按鈕來執行實驗時，將會執行**執行 R 指令碼**模組中的所有 R 程式碼。當執行完成時，**執行 R 指令碼**圖示上將會出現打勾記號。

Azure Machine Learning 的防禦型 R 編碼

如果您正在使用 Azure Machine Learning 為 Web 服務開發 R 程式碼，您應該明確地規劃程式碼將如何處理非預期的資料輸入和例外狀況。為了清楚起見，在所示範的大多數程式碼中，並未包含太多有關檢查或例外狀況處理的部分。不過，隨著我們繼續進行，我將會提供您幾個使用 R 例外狀況處理功能的函式範例。

如果您需要更完整的 R 例外狀況處理方式，建議您閱讀[附錄 B - 進階閱讀](#)列出之 Wickham 所著書籍中適用的小節。

在 Machine Learning Studio 中進行 R 程式碼偵錯和測試

再次提醒您，建議您在 RStudio 中進行小規模的 R 程式碼測試和偵錯。不過，會有一些您將必須探究[執行 R 指令碼](#)本身 R 程式碼問題的情況。此外，在 Machine Learning Studio 中檢查結果也是相當好的做法。

R 程式碼的執行及在 Azure Machine Learning 平台上的執行所產生的輸出主要都在 output.log 中。有些其他資訊會顯示在 error.log 中。

如果在執行 R 程式碼時，Machine Learning Studio 中發生錯誤，您的第一個行動方針應該是查看 error.log。此檔案可能包含可協助您了解並更正錯誤的實用錯誤訊息。若要檢視 error.log，請針對包含錯誤的[執行 R 指令碼](#)，按一下 [屬性] 窗格上的 [檢視錯誤記錄檔]。

例如，我執行了[執行 R 指令碼](#)模組中含有未定義之變數 y 的下列 R 程式碼：

```
x <- 1.0
z <- x + y
```

Copy

此程式碼無法執行，導致發生錯誤狀況。按一下 [屬性] 窗格上的 [檢視錯誤記錄檔]，便會產生如圖 2 所示的內容：

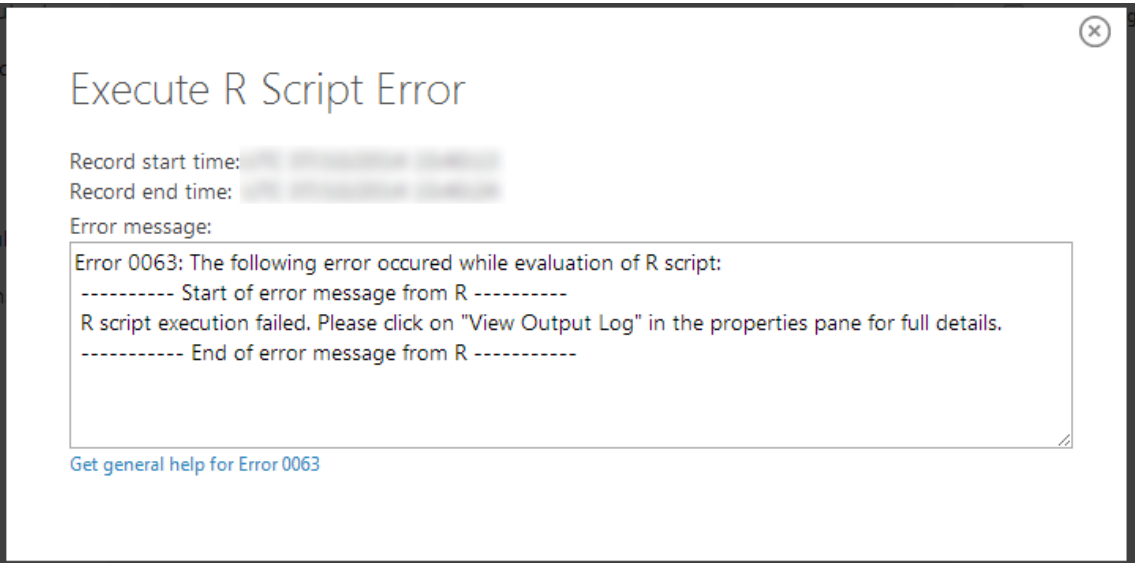


圖 2.錯誤訊息快顯。

看來我們必須查看 output.log 來找出 R 錯誤訊息。按一下[執行 R 指令碼](#)，然後按一下右邊 [屬性] 窗格上的 [檢視 output.log] 項目。新的瀏覽器視窗隨即開啟，我看到下列訊息。

```
[Critical] Error: Error 0063: The following error occurred during evaluation of R script:
----- Start of error message from R -----
object 'y' not found

object 'y' not found
----- End of error message from R -----
```

Copy

此錯誤訊息沒有任何出人意料的內容，並且清楚地指出問題所在。

若要檢查 R 中任何物件的值，您可以將這些值列印至 output.log 檔案中。檢查物件值的規則基本上與在互動式 R 工作階段中相同。例如，如果您在一行輸入變數名稱，物件的值就會列印至 output.log 檔案中。

Azure Machine Learning 附有超過 350 個預先安裝的 R 語言封裝。您可以使用**執行 R 指令碼**模組中的下列程式碼，來擷取預先安裝的封裝清單。

```
data.set <- data.frame(installed.packages())  
maml.mapOutputPort("data.set")
```

[Copy](#)

如果您目前對此程式碼的最後一行還不了解，請往下閱讀。在本文件的其餘部分，我們將大量討論如何在 Azure Machine Learning 環境中使用 R。

RStudio 簡介

RStudio 是一個廣泛使用、適用於 R 的 IDE。我將使用 RStudio 對本快速入門指南中所用的 R 程式碼進行編輯、測試及偵錯。測試並備妥 R 程式碼之後，您只要從 RStudio 編輯器剪下並貼到 Machine Learning Studio 的**執行 R 指令碼**模組中即可。

如果您的桌上型電腦上並未安裝 R 程式設計語言，建議您現在安裝。您可以從 Comprehensive R Archive Network (或稱為 CRAN) 免費下載開放原始碼 R 語言，網址為 <http://www.r-project.org/>。有提供適用於 Windows、MacOS 及 Linux/UNIX 的下載項目。請選擇附近的鏡像，然後依照下載指示進行。此外，CRAN 也包含大量實用的分析和資料操作封裝。

如果您是 RStudio 新手，您應該下載並安裝桌上型電腦版本。您可以在 <http://www.rstudio.com/products/RStudio/> 找到適用於 Windows、MacOS 及 Linux/UNIX 的 RStudio 下載項目。請依照提供的指示，在您的桌上型電腦上安裝 RStudio。

RStudio 的教學課程介紹位於 <https://support.rstudio.com/hc/sections/200107586-Using-RStudio>。

我在[附錄 A](#) 中提供了一些使用 RStudio 的其他相關資訊。

將資料輸入執行 R 指令碼模組及從此模組輸出

在本節中，我們將討論如何將資料輸入**執行 R 指令碼**模組及從此模組輸出。我們將回顧如何處理讀入**執行 R 指令碼**模組及從此模組讀出的各種資料類型。

您稍早下載的 Zip 檔案中有本節的完整程式碼。

在 Machine Learning Studio 中載入和檢查資料

載入資料集

我們將從把 **csdairydata.csv** 檔案載入到 Azure Learning Studio 中開始。

- 啟動 Azure Machine Learning Studio 環境。
- 按一下您畫面左下方的 [+ 新增]，然後選取 [資料集]。
- 選取 [從本機檔案]，然後按一下 [瀏覽] 以選取檔案。
- 請確定您已選取**含標頭的一般 CSV 檔案 (.csv)** 做為資料集類型。
- 按一下核取記號。
- 上傳資料集後，按一下 [資料集] 索引標籤，您應該會看到新的資料集。

建立實驗

既然我們在 Machine Learning Studio 中已經有一些資料，我們需要建立一個實驗來執行分析。

- 按一下左下方的 [+ 新增] 並選取 [實驗]，然後選取 [空白實驗]。
- 您可以選取和修改頁面頂端的**實驗建立目的**標題，為您的實驗命名。例如，將它變更為「加州乳製品分析」。
- 在實驗頁面左側展開 [儲存的資料集]，然後選取 [我的資料集]。您應該會看到先前上傳的 **csdairydata.csv** 檔案。
- 將 [csdairydata.csv 資料集] 拖放到實驗上。
- 在左窗格頂端的 [搜尋實驗項目] 方塊中，輸入**執行 R 指令碼**。您會看到該模組出現在搜尋清單中。
- 將**執行 R 指令碼**模組拖放到您的選盤上。
- 將 [csdairydata.csv 資料集] 的輸出連接到**執行 R 指令碼**最左邊的輸入 (**資料集1**)。

- 別忘了按一下 [儲存]！

此時，您的實驗應該會看起來像圖 3。

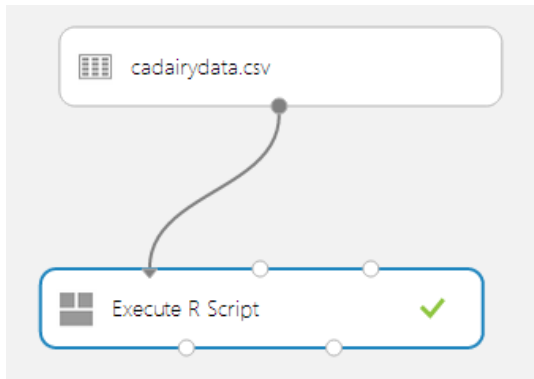


圖 3.含有資料集和 [執行 R 指令碼] 模組的「加州乳製品分析實驗」。

檢查資料

讓我們看看已載入到實驗中的資料。在此實驗中，按一下 [cadairydata.csv 資料集] 的輸出，然後選取 [視覺化]。您應該會看到類似圖 4 的內容。

228 rows	9 columns								
view as									
	Column 0	Year.Month	Month.Number	Year	Month	Cottagecheese.Prod	Icecream.Prod	Milk.Prod	N.CA.Fat.Price
Mean	114.5	2004.065	6.5	2004		2.8451	71.2222	2.9471	1.5058
Median	114.5	2004.065	6.5	2004		2.736	72.454	3.0705	1.4343
Min	1	1995.01	1	1995		1.865	47.127	1.932	0.8924
Max	228	2013.12	12	2013		4.538	93.01	3.804	2.5285
Standard Deviation	65.9621	5.4894	3.4596	5.4893		0.5004	11.1361	0.4773	0.396
Unique Values	228	228	12	19	14	203	228	216	198
Missing Values	0	0	0	0	0	0	0	0	0
Feature Type	Numeric	Numeric	Numeric	Numeric	String	Numeric	Numeric	Numeric	Numeric
	1	1995.01	1	1995	Jan	4.37	51.595	2.112	0.9803
	2	1995.02	2	1995	Feb	3.695	56.086	1.932	0.8924
	3	1995.03	3	1995	Mar	4.538	68.453	2.162	0.8924

圖 4：cadairydata.csv 資料集的摘要。

在這個檢視中，我們會看到許多有用的資訊。我們可以看到該資料集的前幾列。如果我們選取資料行，[統計資料] 區段會顯示有關資料行的詳細資訊。例如，[功能類型] 資料列顯示 Azure Machine Learning Studio 指派給各資料行的資料類型。擁有一個類似這樣的快速檢視，對於開始執行任何正式工作來說，是相當好的執行前例行性檢查。

第一個 R 指令碼

讓我們建立第一個要在 Azure Machine Learning Studio 中實驗的簡單 R 指令碼。我已在 RStudio 中建立並測試下列指令碼。

```
## Only one of the following two lines should be used
## If running in Machine Learning Studio, use the first line with maml.mapInputPort()
## If in RStudio, use the second line with read.csv()
cadairydata <- maml.mapInputPort(1)
# cadairydata <- read.csv("cadairydata.csv", header = TRUE, stringsAsFactors = FALSE)
str(cadairydata)
pairs(~ Cottagecheese.Prod + Icecream.Prod + Milk.Prod + N.CA.Fat.Price, data = cadairydata)
## The following line should be executed only when running in
## Azure Machine Learning Studio
maml.mapOutputPort('cadairydata')
```

Copy

現在，我需要將此指令碼轉移到 Azure Machine Learning Studio。我可以只利用剪下並貼上。不過，在此案例中，我將透過 Zip 檔案轉移我的 R 指令碼。

執行 R 指令碼模組的資料輸入

讓我們看看執行 R 指令碼模組的輸入。在此範例中，我們將會把加州乳製品資料讀入執行 R 指令碼模組中。

執行 R 指令碼模組有三個可能的輸入。視您的應用方式而定，您可以使用這當中的任何一個或所有輸入。使用不接受任何輸入的 R 指令碼也是十分合理的。

讓我們從左到右看看這當中的每一個輸入。您可以將游標放到輸入上方並閱讀工具提示，來查看每個輸入的名稱。

指令碼組合

「指令碼組合」輸入可讓您將 Zip 檔案的內容傳入到執行 R 指令碼模組中。您可以使用下列其中一個命令將 Zip 檔案的內容讀入到 R 程式碼中。

```
source("src/yourfile.R") # Reads a zipped R script
load("src/yourData.rdata") # Reads a zipped R data file
```

[Copy](#)

注意：

Azure Machine Learning 會將 Zip 中的檔案視為在 src/ 目錄中，因此您必須在您的檔案名稱前面加上此目錄名稱。例如，如果 Zip 在其根目錄中包含檔案 `yourfile.R` 和 `yourData.rdata`，使用 `source` 和 `load` 時，您會將這些處理為 `src/yourfile.R` 和 `src/yourData.rdata`。

我們已經在[載入資料集](#)中討論過如何載入資料集在您建立並測試上一節中所示的 R 指令碼之後，請執行下列作業：

1. 將 R 指令碼儲存成 .R 檔案。我將我的指令碼檔案稱為 "simpleplot.R"。內容如下。

```
## Only one of the following two lines should be used
## If running in Machine Learning Studio, use the first line with maml.mapInputPort()
## If in RStudio, use the second line with read.csv()
cadairydata <- maml.mapInputPort(1)
# cadairydata <- read.csv("cadairydata.csv", header = TRUE, stringsAsFactors = FALSE)
str(cadairydata)
pairs(~ Cotagecheese.Prod + Icecream.Prod + Milk.Prod + N.CA.Fat.Price, data = cadairydata)
## The following line should be executed only when running in
## Azure Machine Learning Studio
maml.mapOutputPort('cadairydata')
```

[Copy](#)

2. 建立一個 Zip 檔案，然後將您的指令碼複製到此 Zip 檔案。在 Windows 中，以滑鼠右鍵按一下檔案，選取 [傳送到]、[壓縮的 (zipped) 資料夾]。這會建立包含 "simpleplot.R" 檔案的新 Zip 檔案。
3. 將您的檔案新增到 Machine Learning Studio 中的 [資料集]，將類型指定為 **zip**。您現在應該會在您的資料集內看到的此 Zip 檔案。
4. 將此 Zip 檔案從 [資料集] 拖放到 [ML Studio 畫布] 上。
5. 將 [Zip 資料] 圖示的輸出連接到執行 R 指令碼模組的 [指令碼組合] 輸入。
6. 在執行 R 指令碼模組的程式碼視窗中，輸入含有您 Zip 檔案名稱的 `source()` 函式。在我的案例中，我鍵入了 `source("src/simpleplot.R")`。
7. 確定按一下 [儲存]。

完成這些步驟之後，執行 R 指令碼模組就會在實驗執行時，執行 Zip 檔案中的 R 指令碼。此時，您的實驗應該會看起來像圖 5。

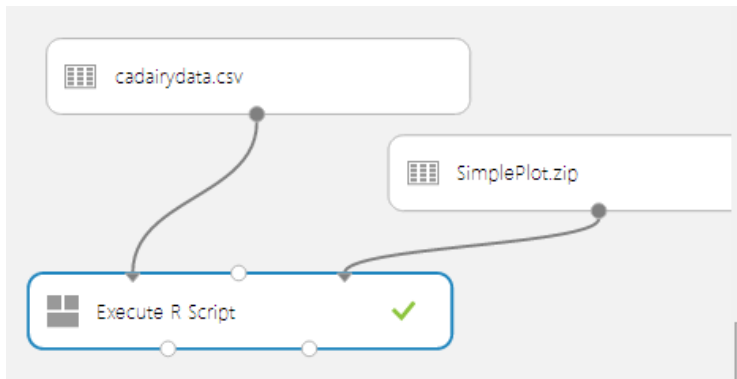


圖 5.使用已壓縮之 R 指令碼的實驗。

資料集1

您可以使用 [資料集1] 輸入將矩形資料表傳遞給您的 R 程式碼。在我們的簡單指令碼中，`maml.mapInputPort(1)` 函式會從連接埠 1 讀取資料。此資料會接著被指派給您程式碼中的資料框架變數名稱。在我們的簡單指令碼中，第一行程式碼會執行這項指派。

```
cadairydata <- maml.mapInputPort(1)
```

Copy

請按一下 [執行] 按鈕來執行您的實驗。執行完成時，請按一下執行 R 指令碼模組，然後按一下 [屬性] 窗格上的 [檢視輸出記錄檔]。您的瀏覽器中應該會出現一個新頁面，當中顯示 output.log 檔案的內容。當您向下捲動時，您應該會看到類似下列的內容。

```
[ModuleOutput] InputDataStructure
[ModuleOutput]
[ModuleOutput] {
[ModuleOutput]   "InputName":Dataset1
[ModuleOutput]   "Rows":228
[ModuleOutput]   "Cols":9
[ModuleOutput]   "ColumnTypes":System.Int32,3,System.Double,5,System.String,1
[ModuleOutput] }
```

Copy

頁面更下方有更詳細的資料行資訊，看起來與下列類似。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame':  228 obs. of  9 variables:
[ModuleOutput]
[ModuleOutput] $ Column 0      : int  1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year.Month    : num  1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month.Number  : int  1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year          : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month         : chr  "Jan" "Feb" "Mar" "Apr" ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num  4.37 3.69 4.54 4.28 4.47 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod   : num  51.6 56.1 68.5 65.7 73.7 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod       : num  2.11 1.93 2.16 2.13 2.23 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price  : num  0.98 0.892 0.892 0.897 0.897 ...
```

Copy

這些結果大致上如預期，資料框架中有 228 個觀察值和 9 個資料行。我們可以看到資料行名稱、R 資料類型及每個資料行的範例。

注意：

從執行 R 指令碼模組的 [R 裝置] 輸出可以便利地取得這個相同的列印輸出。我們將在下一節中討論執行 R 指令碼模組的輸出。

資料集2

[資料集2] 輸入的行為與 [資料集1] 的行為相同。您可以使用此輸入將第二個矩形資料表傳遞給您的 R 程式碼。含有引數 2 的函式 `maml.mapInputPort(2)` 可用來傳遞此資料。

執行 R 指令碼輸出

輸出資料框架

您可以使用 `maml.mapOutputPort()` 函式，透過 [結果資料集1] 連接埠將 R 資料框架的內容輸出成矩形資料表。在我們的簡單 R 指令碼中，會由下列程式碼行執行此動作。

maml.mapOutputPort('cadairydata')

Copy

執行實驗之後，請按一下 [結果資料集1] 輸出連接埠，然後按一下 [視覺化]。您應該會看到類似圖 6 的內容。

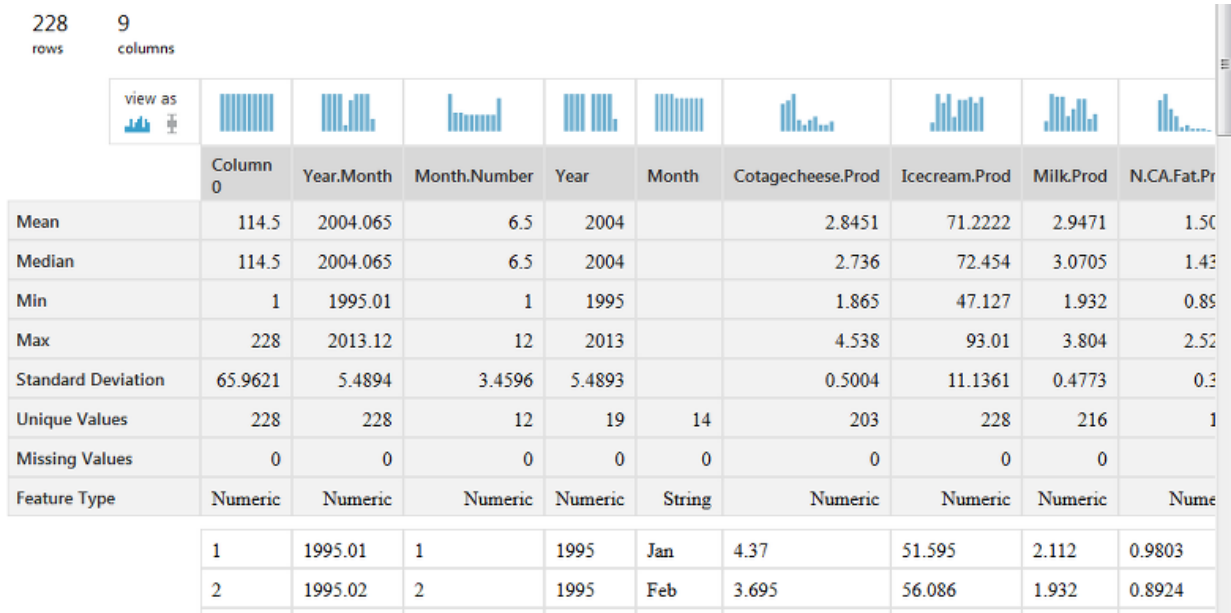


圖 6.加州乳製品資料的輸出視覺化。

此輸出看起來與輸入相同，完全如我們的預期。

R 裝置輸出

執行 R 指令碼模組的 [裝置] 輸出包含訊息和圖形輸出。來自 R 的標準輸出和標準錯誤訊息都會傳送到 [R 裝置] 輸出連接埠。

若要檢視 [R 裝置] 輸出，請按一下該連接埠，然後按一下 [視覺化]。我們會看到如圖 7 中來自 R 指令碼的標準輸出和標準錯誤。

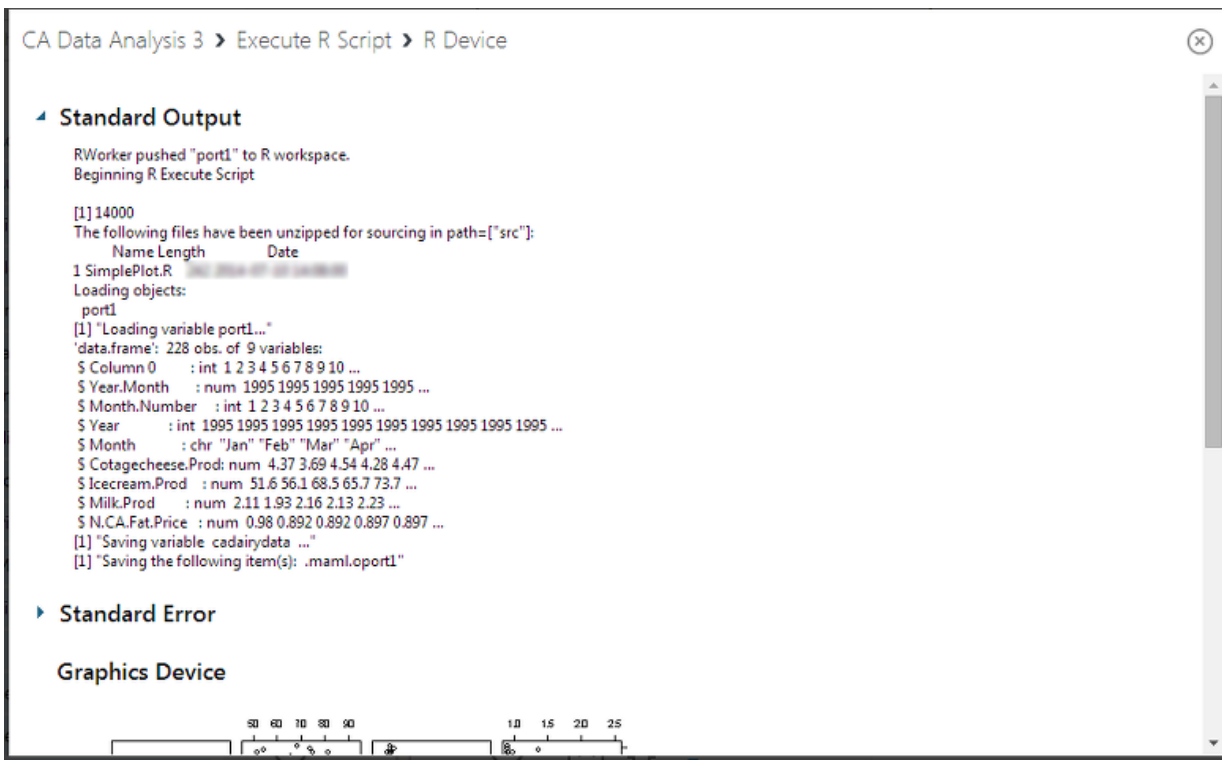


圖 7.來自 [R 裝置] 連接埠的標準輸出和標準錯誤。

向下捲動之後，我們會看到如圖 8 中來自 R 指令碼的圖形輸出。

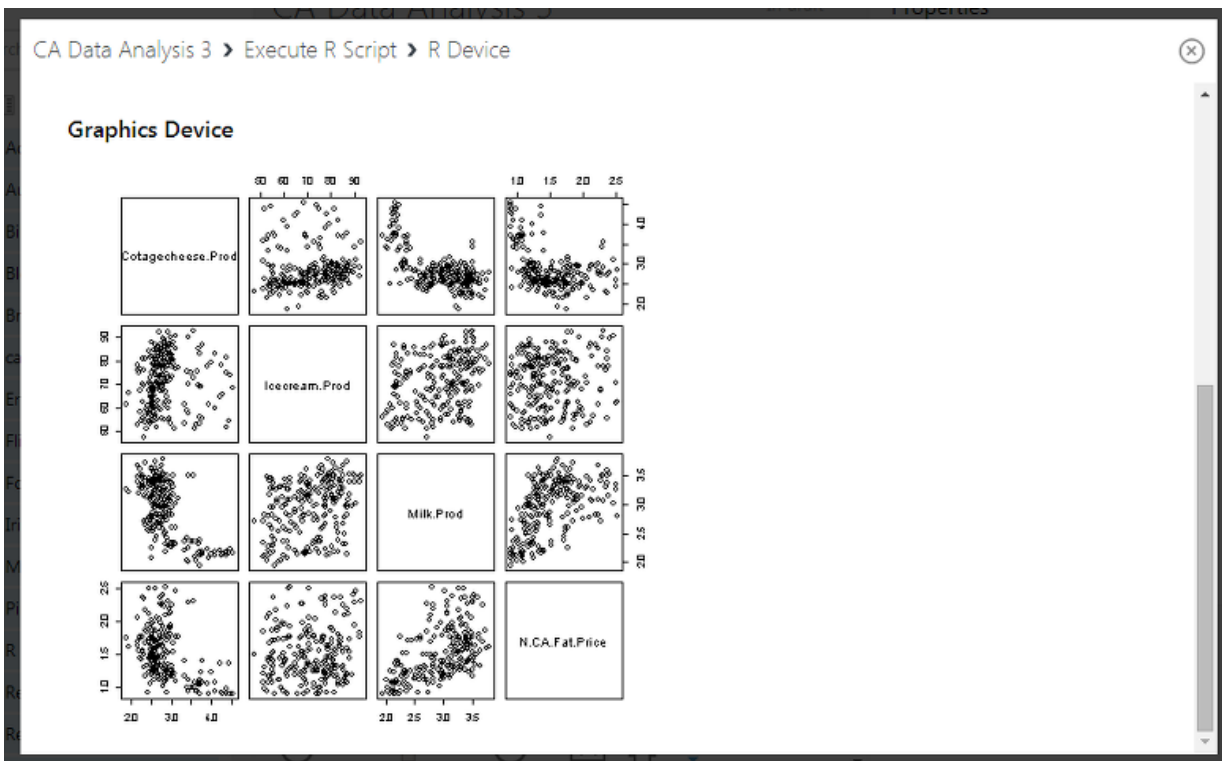


圖 8.來自 [R 裝置] 連接埠的圖形輸出。

資料篩選和轉換

在本節中，我們將對加州乳製品資料執行一些基本的資料篩選和轉換操作。在本節的結尾，我們將會以適用於建置分析模型的格式備妥資料。

更具體來說，在本節中，我們將會執行數個常見的資料清除和轉換工作：類型轉換、依據資料框架進行篩選、新增新的計算資料行，以及值轉換。此背景應該可以協助您處理在真實世界問題中遇到的許多變化。

您稍早下載的 Zip 檔案中有本節的完整 R 程式碼。

既然我們可以將加州乳製品資料讀入到[執行 R 指令碼](#)模組的 R 程式碼中，我們需要確保資料行中的資料具有預期的類型和格式。

R 是動態指定類型的語言，這表示會視需要強制將資料類型從一種類型轉換成另一種類型。R 中不可部分完成的資料類型包括數值、邏輯及字元。因素類型可用來簡潔地儲存分類資料。您可以在[附錄 B - 進階閱讀](#)的參考內容中，找到更多有關資料類型的資訊。

將表格資料從外部來源讀入到 R 中時，最好一律檢查資料行中產生的類型。您可能想要字元類型的資料行，但在許多情況下這會顯示為因素類型，反之亦然。在其他情況下，則是會以字元資料代表您認為應該是數值的資料行，例如 '1.23' 而非浮點數形式的 1.23。

幸運的是，只要能夠對應，將一個類型轉換成另一個類型相當容易。例如，您無法將 'Nevada' 轉換成數值，但是可以將它轉換成因素(類別變數)。另舉一例，您可以將數值 1 轉換成字元 '1' 或因素。

任何這些轉換的語法都很簡單：[as.datatype\(\)](#)。這些類型轉換函式包括：

- [as.numeric\(\)](#)
- [as.character\(\)](#)
- [as.logical\(\)](#)
- [as.factor\(\)](#)

看看我們在上一節中輸入之資料行的資料類型：除了標示為 'Month' 的資料行為字元類型之外，所有資料行的類型都是數值。讓我們將其轉換成因素，然後測試結果。

我已經刪除建立散佈圖矩陣的程式碼行，並新增將 'Month' 資料行轉換成因素的程式碼行。在我的實驗中，我將只是把 R 程式碼剪下並貼到[執行 R 指令碼](#) 模組的程式碼視窗中。您也可以更新 Zip 檔案，然後將它上傳到 Azure Machine Learning Studio，但這需要數個步驟。

```
## Only one of the following two lines should be used
## If running in Machine Learning Studio, use the first line with maml.mapInputPort()
## If in RStudio, use the second line with read.csv()
cadairydata <- maml.mapInputPort(1)
# cadairydata <- read.csv("cadairydata.csv", header = TRUE, stringsAsFactors = FALSE)
## Ensure the coding is consistent and convert column to a factor
cadairydata$Month <- as.factor(cadairydata$Month)
str(cadairydata) # Check the result
## The following line should be executed only when running in
## Azure Machine Learning Studio
maml.mapOutputPort('cadairydata')
```

Copy

讓我們執行這個程式碼並查看 R 指令碼的輸出記錄檔。圖 9 顯示來自記錄檔的相關資料。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame': 228 obs. of 9 variables:
[ModuleOutput]
[ModuleOutput] $ Column 0 : int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year.Month : num 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month.Number : int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year : int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month : Factor w/ 14 levels "Apr","April",...: 6 5 9 1 11 8 7 3 14 13 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num 4.37 3.69 4.54 4.28 4.47 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod : num 51.6 56.1 68.5 65.7 73.7 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod : num 2.11 1.93 2.16 2.13 2.23 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price : num 0.98 0.892 0.892 0.897 0.897 ...
[ModuleOutput]
```

Copy

```
[ModuleOutput] [1] "Saving variable  cadairydata ..."
[ModuleOutput]
[ModuleOutput] [1] "Saving the following item(s): .maml.oport1"
```

圖 9.含有因素變數之資料框架的摘要。

Month 的類型現在應該會表示為 'Factor w/ 14 levels'。這會發生問題，因為一年只有 12 個月。您也可以檢查看看 [結果資料集] 連接埠之 [視覺化] 中的類型是否為 'Categorical'。

問題在於 'Month' 資料行的程式碼並非以有系統的方式撰寫。在某些情況下，某個月份稱為 April，在其他情況下則會縮寫成 Apr。我們可以將字串修剪成 3 個字元來解決這個問題。這行程式碼現在看起來如下：

```
## Ensure the coding is consistent and convert column to a factor
cadairydata$Month <- as.factor(substr(cadairydata$Month, 1, 3))
```

Copy

重新執行實驗和檢視輸出記錄檔。圖 10 顯示預期的結果。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame':  228 obs. of  9 variables:
[ModuleOutput]
[ModuleOutput] $ Column 0      : int  1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year.Month    : num  1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month.Number  : int  1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year          : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month         : Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num  4.37 3.69 4.54 4.28 4.47 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod   : num  51.6 56.1 68.5 65.7 73.7 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod       : num  2.11 1.93 2.16 2.13 2.23 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price  : num  0.98 0.892 0.892 0.897 0.897 ...
[ModuleOutput]
[ModuleOutput] [1] "Saving variable  cadairydata ..."
[ModuleOutput]
[ModuleOutput] [1] "Saving the following item(s): .maml.oport1"
```

Copy

圖 10.因素層級數目正確之資料框架的摘要。

我們的因素變數現在具有所需的 12 個層級。

基本資料框架篩選

R 資料框架支援強大的篩選功能。藉由在資料列或資料行使用邏輯篩選，可將資料集再細分成子集。在許多情況下，將會需要複雜的篩選條件。[附錄 B - 進階閱讀](#)中的參考包含大量的篩選資料框架範例。

有一些篩選是我們應該在資料集上執行的。如果您看一下 cadariydata 資料框架中的資料行，您會看到兩個不必要的資料行。第一個資料行只存放了資料列編號，這不是很有用。第二個資料行 Year.Month 包含重複的資訊。我們可以使用下列 R 程式碼輕鬆地排除這些資料行。

注意：

從現在起，在本節中，我將只會示範要在執行 R 指令碼模組中新增的額外程式碼。我會在 `str()` 函式之前新增每個新程式碼行。我會使用此函式在 Azure Machine Learning Studio 中確認我的結果。

我在執行 R 指令碼模組的 R 程式碼中新增下列程式碼行。

```
# Remove two columns we do not need
cadairydata <- cadairydata[, c(-1, -2)]
```

Copy

在您的實驗中執行此程式碼，並檢查輸出記錄檔的結果。這些結果如圖 11 所示。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame': 228 obs. of 7 variables:
[ModuleOutput]
[ModuleOutput] $ Month.Number :int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year :int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month : Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num 4.37 3.69 4.54 4.28 4.47 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod : num 51.6 56.1 68.5 65.7 73.7 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod : num 2.11 1.93 2.16 2.13 2.23 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price : num 0.98 0.892 0.892 0.897 0.897 ...
[ModuleOutput]
[ModuleOutput] [1] "Saving variable cadairydata ..."
[ModuleOutput]
[ModuleOutput] [1] "Saving the following item(s): .maml.oport1"
```

Copy

圖 11.已移除兩個資料行之資料框架的摘要。

好消息！我們得到預期的結果。

加入新的資料行

若要建立時間序列模型，讓資料行包含自時間序列開始後的月份將會比較方便。我們將會建立新資料行 'Month.Count'。

為了協助組織程式碼，我們將建立我們的第一個簡單函式 `num.month()`。然後，我們會套用此函式在資料框架中建立新資料行。新程式碼如下所示。

```
## Create a new column with the month count
## Function to find the number of months from the first
## month of the time series
num.month <- function(Year, Month) {
  ## Find the starting year
  min.year <- min(Year)

  ## Compute the number of months from the start of the time series
  12 * (Year - min.year) + Month - 1
}

## Compute the new column for the dataframe
cadairydata$Month.Count <- num.month(cadairydata$Year, cadairydata$Month.Number)
```

Copy

現在，執行更新的實驗並使用輸出記錄檔來檢視結果。這些結果如圖 12 所示。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame': 228 obs. of 8 variables:
[ModuleOutput]
[ModuleOutput] $ Month.Number : int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year : int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month : Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num 4.37 3.69 4.54 4.28 4.47 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod : num 51.6 56.1 68.5 65.7 73.7 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod : num 2.11 1.93 2.16 2.13 2.23 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price : num 0.98 0.892 0.892 0.897 0.897 ...
[ModuleOutput]
[ModuleOutput] $ Month.Count : num 0 1 2 3 4 5 6 7 8 9 ...
[ModuleOutput]
[ModuleOutput] [1] "Saving variable cadairydata ..."
[ModuleOutput]
[ModuleOutput] [1] "Saving the following item(s): .maml.oport1"
```

圖 12.含有其他資料行之資料框架的摘要。

看來一切運作正常。我們的資料框架中有了包含預期值的新資料行。

值轉換

在本節中，我們將對資料框架之某些資料行中的值執行一些簡單的轉換。R 語言幾乎支援任何一種值轉換。[附錄 B - 進階閱讀](#)中的參考包含大量範例。

如果您看看我們資料框架摘要中的值，您應該會發現此處有點奇怪。加州生產的冰淇淋比牛奶多？否，當然不是，因為這樣並不合理，雖然這對我們當中的一些冰淇淋愛好者來說是個令人悲傷的事實。其單位並不相同。計價單位為美制磅，牛奶是以 100 萬美制磅為單位、冰淇淋是以 1,000 美制加侖為單位，而卡達乾酪則是以 1,000 美制磅為單位。假設冰淇淋每加侖重約 6.5 磅，我們便可輕鬆地進行乘法運算來轉換這些值，讓它們都同樣以 1000 磅為單位。

針對我們的預測模型，我們使用乘法模型來進行此資料的趨勢和季節性調整。對數轉換可讓我們使用線性模型，以簡化此程序。我們可以在套用乘數的相同函式中套用對數轉換。

在下列程式碼中，我定義了新函式 `log.transform()`，並將它套用至包含數值的資料列。R `Map()` 函式用於將 `log.transform()` 函式套用至資料框架中選取的資料行。`Map()` 與 `apply()` 類似，但可允函式有多個引數清單。請注意，乘數清單會提供 `log.transform()` 函式的第二個引數。`na.omit()` 函式是用來進行一點清除，以確保我們在資料框架中沒有遺失或未定義的值。

```
log.transform <- function(invec, multiplier = 1) {
  ## Function for the transformation, which is the log
  ## of the input value times a multiplier

  warningmessages <- c("ERROR: Non-numeric argument encountered in function log.transform",
    "ERROR: Arguments to function log.transform must be greate than zero",
    "ERROR: Aggurment multiplier to fuctionion log.transform must be a scaler",
    "ERROR: Invalid time seies value encountered in function log.transform"
  )

  ## Check the input arguments
  if(!is.numeric(invec) | !is.numeric(multiplier)) {warning(warningmessages[1]); return(NA)}
  if(any(invec < 0.0) | any(multiplier < 0.0)) {warning(warningmessages[2]); return(NA)}
  if(length(multiplier) != 1) {{warning(warningmessages[3]); return(NA)}}

  r ## Wrap the multiplication in tryCatch
```

```
## If there is an exception, print the warningmessage to
## standard error and return NA
tryCatch(log(multiplier * invvec),
         error = function(e){warning(warningmessages[4]); NA})
}

## Apply the transformation function to the 4 columns
## of the dataframe with production data
multipliers <- list(1.0, 6.5, 1000.0, 1000.0)
cadairydata[, 4:7] <- Map(log.transform, cadairydata[, 4:7], multipliers)

## Get rid of any rows with NA values
cadairydata <- na.omit(cadairydata)
```

`log.transform()` 函式中進行的事情不少。此程式碼中大部分會檢查引數的潛在問題，或是處理仍可能在計算期間發生的例外狀況。此程式碼中實際上只有幾行在進行計算。

防禦型程式設計的目標，是要防止因單一函式失敗而導致無法繼續處理的情況。執行很久的分析如果突然失敗，可能會讓使用者深感挫折。為了避免這種情況，必須選擇預設的傳回值，以將損害限制在下游處理。系統也會產生訊息來警示使用者發生錯誤。

如果您不熟悉使用 R 進行防禦型程式設計，此程式碼的一切可能會讓您不知所措。我將引導您完成主要的步驟：

1. 定義包含四個訊息的向量。這些訊息用來傳達一些可能在此程式碼發生的錯誤和例外狀況的相關資訊。
2. 我會針對每個案例傳回一個 NA 值。有許多其他可能會有較少副作用的可能性。例如，我可以傳回零向量或原始輸入向量。
3. 對函式的引數執行檢查。在每個案例中，如果偵測到錯誤，便會傳回預設值，並且由 `warning()` 函式產生訊息。我使用的是 `warning()` 而非 `stop()`，因為後者會終止執行，而這正是我試著要避免的情況。請注意，我是以程序性風格撰寫此程式碼，因此在此情況下，函式型方法看似既複雜又難以理解。
4. 對數計算被包裝在 `tryCatch()` 中，如此例外狀況就不會導致處理突然停止。如果沒有 `tryCatch()`，R 函式所引發的大多數錯誤都會導致產生停止訊號，而此訊號所執行的就是停止。

請在您的實驗中執行此 R 程式碼，然後看看 `output.log` 檔案中的列印輸出。您現在會在記錄中看到四個資料行的已轉換值，如圖 13 所示。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame': 228 obs. of 8 variables:
[ModuleOutput]
[ModuleOutput] $ Month.Number :int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year :int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month :Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num 1.47 1.31 1.51 1.45 1.5 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod :num 5.82 5.9 6.1 6.06 6.17 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod :num 7.66 7.57 7.68 7.66 7.71 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price :num 6.89 6.79 6.79 6.8 6.8 ...
[ModuleOutput]
[ModuleOutput] $ Month.Count :num 0 1 2 3 4 5 6 7 8 9 ...
[ModuleOutput]
[ModuleOutput] [1] "Saving variable cadairydata ..."
[ModuleOutput]
[ModuleOutput] [1] "Saving the following item(s): .maml.oport1"
```

Copy

我們會看到值已經轉換。現在，牛奶產量大幅超過所有其他乳製品產量，還記得我們現在看的是對數刻度。

此時會清除我們的資料，而我們已經準備好進行一些建立模型工作。看看[執行 R 指令碼](#)模組之 [結果資料集] 輸出的視覺化摘要，您會看到 'Month' 資料行是 'Categorical' 並具有 12 個唯一值，這又再次是我們所想要的。

時間序列物件和相互關聯分析

在本節中，我們將探討一些基本的 R 時間序列物件，並分析一些變數之間的相互關聯。我們的目標是要輸出資料框架，此框架中包含數段延隔時間的成對相互關聯資訊。

您稍早下載的 Zip 檔中有本節的完整 R 程式碼。

R 中的時間序列物件

如已經提過的，時間序列是一系列依時間編制索引的資料值。R 時間序列物件可用來建立和管理時間索引。使用時間序列物件有數個優點。時間序列物件可讓您不必理會管理封裝在物件中之時間序列索引值的許多細節。此外，時間序列物件還可讓您使用許多時間序列方法來繪製、列印、建立模型等。

POSIXct 時間序列類別是常用且相對簡單的類別。此時間序列類別是從 1970 年 1 月 1 日開始計量時間。在此範例中，我們將使用 POSIXct 時間序列物件。其他廣泛使用的 R 時間序列物件類別包括 zoo 和 xts (可延伸時間序列)。

時間序列物件範例

讓我們開始進行我們的範例。請將新的[執行 R 指令碼](#)模組拖放到您的實驗中。將現有[執行 R 指令碼](#)模組的 [結果資料集 1] 輸出連接埠連接到新的[執行 R 指令碼](#)模組的 [資料集 1] 輸入連接埠。

如同我為前幾個範例所做的，隨著我們循序進行此範例，在某些點，我將只會示範在每個步驟累加的額外 R 程式碼行。

讀取資料框架

第一步是先讀入一個資料框架，然後確定得到預期的結果。下列程式碼應該能執行此作業。

```
# Comment the following if using RStudio
cadairydata <- maml.mapInputPort(1)
str(cadairydata) # Check the results
```

[Copy](#)

現在，請執行實驗。新的執行 R 指令碼圖形的記錄檔看起來應該像圖 14。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame':  228 obs. of  8 variables:
[ModuleOutput]
[ModuleOutput] $ Month.Number   :int  1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year           :int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month          :Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num  1.47 1.31 1.51 1.45 1.5 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod   : num  5.82 5.9 6.1 6.06 6.17 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod       : num  7.66 7.57 7.68 7.66 7.71 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price  : num  6.89 6.79 6.79 6.8 6.8 ...
[ModuleOutput]
[ModuleOutput] $ Month.Count     : num  0 1 2 3 4 5 6 7 8 9 ...
```

[Copy](#)

- 14.[執行 R 指令碼] 模組中資料框架的摘要。*

[†] 此資料的類型和格式皆如預期。請注意，'Month' 資料行的類型是因素，並且具有預期的層級數目。

建立時間序列物件

我們需要將時間序列物件新增到我們的資料框架中。請以下列程式碼取代目前的程式碼，這會新增新的 POSIXct 類別資料行。

```
# Comment the following if using RStudio
```

Copy

```
cadairydata <- maml.mapInputPort(1)
```

```
## Create a new column as a POSIXct object
```

```
Sys.setenv(TZ = "PST8PDT")
```

```
cadairydata$Time <- as.POSIXct(strptime(paste(as.character(cadairydata$Year), "-", as.character(cadairydata$Month.Number), "-01 00:00:00", s
```

```
str(cadairydata) # Check the results
```

現在，請檢查記錄檔。這應該會看起來像圖 15。

```
[ModuleOutput] [1] "Loading variable port1..."
```

Copy

```
[ModuleOutput]
```

```
[ModuleOutput] 'data.frame': 228 obs. of 9 variables:
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Month.Number :int 1 2 3 4 5 6 7 8 9 10 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Year :int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Month : Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Cotagecheese.Prod: num 1.47 1.31 1.51 1.45 1.5 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Icecream.Prod : num 5.82 5.9 6.1 6.06 6.17 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Milk.Prod : num 7.66 7.57 7.68 7.66 7.71 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ N.CA.Fat.Price : num 6.89 6.79 6.79 6.8 6.8 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Month.Count : num 0 1 2 3 4 5 6 7 8 9 ...
```

```
[ModuleOutput]
```

```
[ModuleOutput] $ Time : POSIXct, format: "1995-01-01" "1995-02-01" ...
```

- 15.含有時間序列物件之資料框架的摘要。*

我們可以從摘要看出，新資料行的類別事實上是 POSIXct。

探索和轉換資料

讓我們探索此資料集內的一些變數。散佈圖矩陣是產生快速檢視的好方法。我將以下列程式碼行取代前一個 R 程式碼中的 `str()` 函式：

```
pairs(~ Cotagecheese.Prod + Icecream.Prod + Milk.Prod + N.CA.Fat.Price, data = cadairydata, main = "Pairwise Scatterplots of dairy time series")
```

請執行此程式碼，然後看看結果如何。在 [R 裝置] 連接埠產生的圖應該看起來像圖 16。

Graphics Device

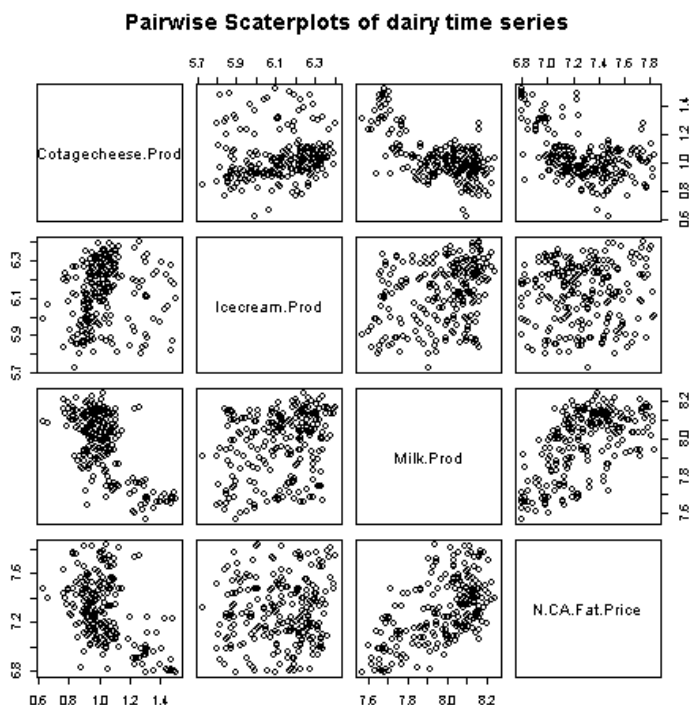


圖 16.所選變數的散佈圖矩陣。

這些變數之間的關係中有看似奇怪的結構。這可能是產生自資料中的趨勢，以及產生自我們未將變數標準化的事實。

相互關聯分析

若要執行相互關聯分析，我們必須將變數去除趨勢並標準化。我們可以就使用既可將變數置中又可縮放變數的 R `scale()` 函式。此函式可能也執行得更快。不過，我想要示範以 R 撰寫的防禦型程式設計範例。

以下所示的 `ts.detrend()` 函式即可執行這兩種作業。下列兩行程式碼會將資料去除趨勢，然後將值標準化。

```
ts.detrend <- function(ts, Time, min.length = 3){
  ## Function to de-trend and standardize a time series

  ## Define some messages if they are NULL
  messages <- c('ERROR: ts.detrend requires arguments ts and Time to have the same length',
    'ERROR: ts.detrend requires argument ts to be of type numeric',
    paste('WARNING: ts.detrend has encountered a time series with length less than', as.character(min.length)),
    'ERROR: ts.detrend has encountered a Time argument not of class POSIXct',
    'ERROR: Detrend regression has failed in ts.detrend',
    'ERROR: Exception occurred in ts.detrend while standardizing time series in function ts.detrend'
  )

  # Create a vector of zeros to return as a default in some cases
  zerovec <- rep(length(ts), 0.0)

  # The input arguments are not of the same length, return ts and quit
  if(length(Time) != length(ts)) {warning(messages[1]); return(ts)}

  # If the ts is not numeric, just return a zero vector and quit
  if(!is.numeric(ts)) {warning(messages[2]); return(zerovec)}

  # If the ts is too short, just return it and quit
  if((ts.length <- length(ts)) < min.length) {warning(messages[3]); return(ts)}

  ## Check that the Time variable is of class POSIXct
  if(class(cadairydata$Time)[1] != "POSIXct") {warning(messages[4]); return(ts)}

  ## De-trend the time series by using a linear model
  ts.frame <- data.frame(ts = ts, Time = Time)
```

Copy

```

tryCatch({ts <- ts - fitted(lm(ts ~ Time, data = ts.frame))},
  error = function(e){warning(messages[5]); zerovec})

tryCatch( {stdev <- sqrt(sum((ts - mean(ts))^2)/(ts.length - 1)
  ts <- ts/stdev},
  error = function(e){warning(messages[6]); zerovec})

ts
}
## Apply the detrend.ts function to the variables of interest
df.detrend <- data.frame(lapply(cadairydata[, 4:7], ts.detrend, cadairydata$Time))

## Plot the results to look at the relationships
pairs(~ Cotagecheese.Prod + Icecream.Prod + Milk.Prod + N.CA.Fat.Price, data = df.detrend, main = "Pairwise Scatterplots of detrended standi

```

`ts.detrend()` 函式中進行的事情不少。此程式碼中大部分會檢查引數的潛在問題，或是處理仍可能在計算期間發生的例外狀況。此程式碼中實際上只有幾行在進行計算。

我們已經在 [值轉換](#) 中討論過防禦型程式設計的範例兩個計算區塊皆包裝在 `tryCatch()` 中。就某些錯誤而言，傳回原始輸入向量相當合理，而在其他情況下，我會傳回零向量。

請注意，用於去除趨勢的線性迴歸是時間序列迴歸。預測工具變數是時間序列物件。

定義 `ts.detrend()` 之後，我們會將它套用到資料框架中感興趣的變數。我們必須使用 `as.data.frame()` 將 `lapply()` 所建立的結果清單強制轉換成資料框架。由於 `ts.detrend()` 的防禦性層面緣故，因此即使無法處理其中一個變數，也不會導致無法處理其他變數。

最後一行程式碼會建立成對的散佈圖。執行 R 程式碼之後，散佈圖的結果會如圖 17 所示。

Pairwise Scatterplots of detrended standardized time series

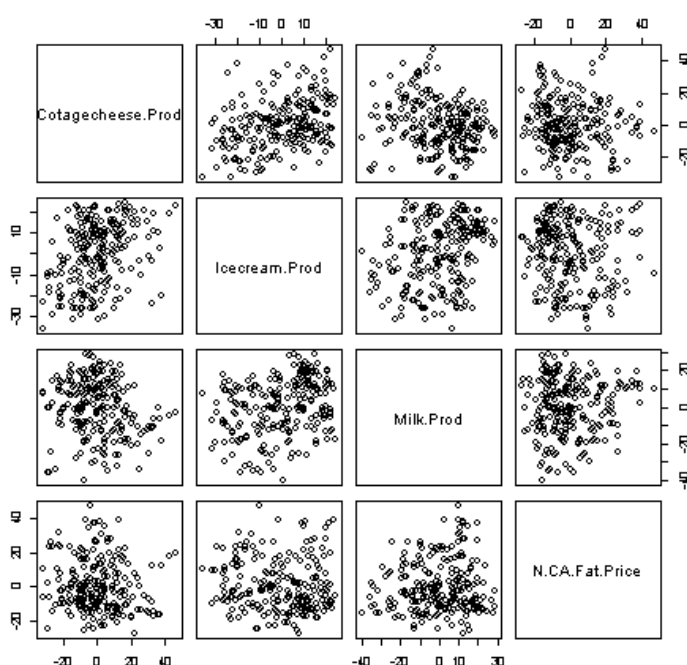


圖 17.已去除趨勢並已標準化之時間序列的成對散佈圖。

您可以將這些結果與圖 17 所示的結果做比較。在已移除趨勢並將變數標準化的情況下，我們會看到這些變數之間的關係少了許多結構。

以 R `ccf` 物件方式計算相互關聯的程式碼如下所示。

```

## A function to compute pairwise correlations from a
## list of time series value vectors
pair.cor <- function(pair.ind, ts.list, lag.max = 1, plot = FALSE){
  ccf(ts.list[[pair.ind[1]]], ts.list[[pair.ind[2]]], lag.max = lag.max, plot = plot)
}

```

```
## A list of the pairwise indices
corpairs <- list(c(1,2), c(1,3), c(1,4), c(2,3), c(2,4), c(3,4))

## Compute the list of ccf objects
cadairy correlations <- lapply(corpairs, pair.cor, df.detrend)

cadairy correlations
```

執行此程式碼會產生如圖 18 所示的記錄。

```
[ModuleOutput] Loading objects:
[ModuleOutput] port1
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput] [[1]]
[ModuleOutput]
[ModuleOutput] Autocorrelations of series 'X', by lag
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] -1 0 1
[ModuleOutput] 0.148 0.358 0.317
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] [[2]]
[ModuleOutput]
[ModuleOutput] Autocorrelations of series 'X', by lag
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] -1 0 1
[ModuleOutput] -0.395 -0.186 -0.238
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] [[3]]
[ModuleOutput]
[ModuleOutput] Autocorrelations of series 'X', by lag
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] -1 0 1
[ModuleOutput] -0.059 -0.089 -0.127
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] [[4]]
[ModuleOutput]
[ModuleOutput] Autocorrelations of series 'X', by lag
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] -1 0 1
[ModuleOutput] 0.140 0.294 0.293
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] [[5]]
[ModuleOutput]
[ModuleOutput] Autocorrelations of series 'X', by lag
[ModuleOutput]
[ModuleOutput]
[ModuleOutput] -1 0 1
[ModuleOutput] -0.002 -0.074 -0.124
```

Copy

圖 18.來自成對相互關聯分析的 ccf 物件清單。

每段延隔時間都有一個相互關聯值。這些相互關聯值都沒有大到足以具有重要性。因此，可以推斷出我們可以獨立建立每個變數的模型。

輸出資料框架

我們已經以 R ccf 物件清單方式計算成對的相互關聯。這樣會有一點問題，因為 [結果資料集] 輸出連接埠確實需要資料框架。此外，ccf 物件本身是清單，而我們只想要此清單第一個元素中的值，亦即各段延隔時間的相互關聯。

下列程式碼會從 ccf 物件 (本身是清單) 的清單中擷取延隔時間值。

```
df.correlations <- data.frame(do.call(rbind, lapply(cadairycorrelations, '[', 1)))
```

Copy

```
c.names <- c("correlation pair", "-1 lag", "0 lag", "+1 lag")
r.names <- c("Corr Cot Cheese - Ice Cream",
            "Corr Cot Cheese - Milk Prod",
            "Corr Cot Cheese - Fat Price",
            "Corr Ice Cream - Mik Prod",
            "Corr Ice Cream - Fat Price",
            "Corr Milk Prod - Fat Price")
```

```
## Build a dataframe with the row names column and the
## correlation data frame and assign the column names
outframe <- cbind(r.names, df.correlations)
colnames(outframe) <- c.names
outframe
```





```
## WARNING!
## The following line works only in Azure Machine Learning
## When running in RStudio, this code will result in an error
#maml.mapOutputPort('outframe')
```

第一行程式碼是需要一點技巧，一些說明可以幫助您了解它。由內而外可分為下列項目：

1. 含有引數 '[' 的 '1' 運算子會從 ccf 物件清單的第一個元素選取各段延隔時間的相互關聯向量。
2. `do.call()` 函式會在 `lapply()` 所傳回之清單的項目上套用 `rbind()` 函式。
3. `data.frame()` 函式會強制將 `do.call()` 產生的結果轉換成資料框架。

請注意，資料列名稱會在資料框架的資料行中。這麼做可在從執行 R 指令碼輸出資料列名稱時，保留這些資料列名稱。

執行此程式碼時，會在我將 [結果資料集] 連接埠的輸出 [視覺化] 時，產生如圖 19 所示的輸出。資料列名稱如預期般在第一個資料行中。

view as				
	RowNames	-1 lag	0 lag	+1 lag
Mean		-0.0227	0.0633	0.0287
Median		0.0155	0.001	-0.0362
Min		-0.3954	-0.1858	-0.2385
Max		0.1478	0.3581	0.317
Standard Deviation		0.1996	0.221	0.2336
Unique Values	6	6	6	6
Missing Values	0	0	0	0
Feature Type	Categorical	Numeric	Numeric	Numeric

Corr Cot Cheese - Ice Cream	0.147775	0.358106	0.317
Corr Cot Cheese - Milk Prod	-0.395444	-0.185777	-0.238486
Corr Cot Cheese - Fat Price	-0.059431	-0.0886404	-0.127372
Corr Ice Cream - Milk Prod	0.139825	0.294231	0.293429
Corr Ice Cream - Fat Price	-0.00187804	-0.0735102	-0.124443
Corr Milk Prod - Fat Price	0.0329361	0.0754193	0.0520239

圖 19.來自相互關聯分析的結果輸出。

時間序列範例：季節性預測

我們的資料現在是適用於分析的形式，而我們已判斷出變數之間沒有重大的相互關聯。讓我們繼續來建立時間序列預測模型。我們將使用此模型預測 2013 年 12 個月的加州牛奶產量。

我們的預測模型將會有兩個元件，亦即趨勢元件和季節性元件。這兩個元件的乘積即是完整預測。這種模型稱為乘法模型。替代模型是加法模型。我們已經將對數轉換套用到感興趣的變數，以便控制這項分析。

您稍早下載的 Zip 檔中有本節的完整 R 程式碼。

建立用於分析的資料框架

首先，請將新的執行 R 指令碼模組新增到您的實驗中。將現有執行 R 指令碼模組的 [結果資料集] 輸出連接到新模組的 [資料集1] 輸入。結果應該會看起來像圖 20。

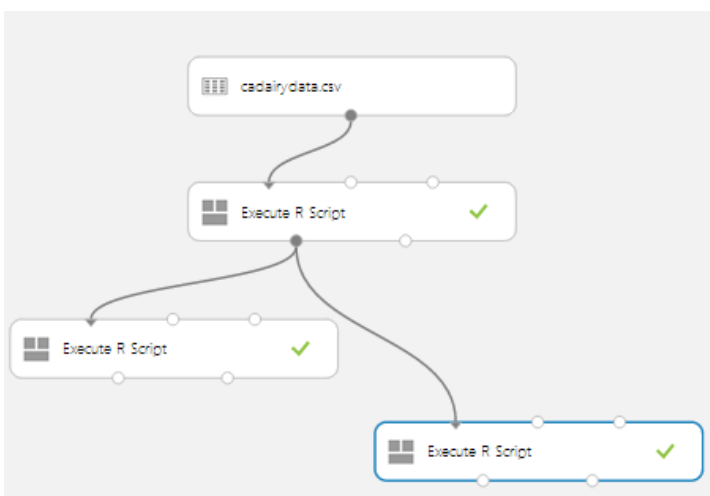


圖 20.新增了 [執行 R 指令碼] 模組的實驗。

[†] 與我們剛剛完成的相互關聯分析相同，我們需要新增一個含有 POSIXct 時間序列物件的資料行。下列程式碼將執行的就是這個動作。

If running in Machine Learning Studio, uncomment the first line with maml.mapInputPort()

Copy

```
cadairydata <- maml.mapInputPort(1)
```

Create a new column as a POSIXct object

```
Sys.setenv(TZ = "PST8PDT")
```

```
cadairydata$Time <- as.POSIXct(strptime(paste(as.character(cadairydata$Year), "-", as.character(cadairydata$Month.Number), "-01 00:00:00", s
```

```
str(cadairydata)
```

執行此程式碼並查看記錄檔。結果應該會看起來像圖 21。

```
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput]
[ModuleOutput] 'data.frame': 228 obs. of 9 variables:
[ModuleOutput]
[ModuleOutput] $ Month.Number : int 1 2 3 4 5 6 7 8 9 10 ...
[ModuleOutput]
[ModuleOutput] $ Year : int 1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
[ModuleOutput]
[ModuleOutput] $ Month : Factor w/ 12 levels "Apr","Aug","Dec",...: 5 4 8 1 9 7 6 2 12 11 ...
[ModuleOutput]
[ModuleOutput] $ Cotagecheese.Prod: num 1.47 1.31 1.51 1.45 1.5 ...
[ModuleOutput]
[ModuleOutput] $ Icecream.Prod : num 5.82 5.9 6.1 6.06 6.17 ...
[ModuleOutput]
[ModuleOutput] $ Milk.Prod : num 7.66 7.57 7.68 7.66 7.71 ...
[ModuleOutput]
[ModuleOutput] $ N.CA.Fat.Price : num 6.89 6.79 6.79 6.8 6.8 ...
[ModuleOutput]
[ModuleOutput] $ Month.Count : num 0 1 2 3 4 5 6 7 8 9 ...
[ModuleOutput]
[ModuleOutput] $ Time : POSIXct, format: "1995-01-01" "1995-02-01" ...
```

Copy

圖 21.資料框架的摘要。

有了這個結果之後，我們便已準備好開始進行分析。

建立訓練資料集

建構資料框架之後，我們需要建立訓練資料集。此資料將包含所有觀察值，但 2013 年最後一個的 12 除外，這是我們的測試資料集。下列程式碼會將資料框架細分成子集，並繪製乳製品產量和價格變數的圖。然後，我會繪製四個產量和價格變數的圖。匿名函式可用來定義一些用於繪圖的引數，然後藉由 `Map()` 逐一查看其他兩個引數的清單。如果您正在想著可以在這裡使用 `for` 迴圈，的確沒錯。但是，由於 R 是函式型語言，因此我示範給您的是函式型方法。

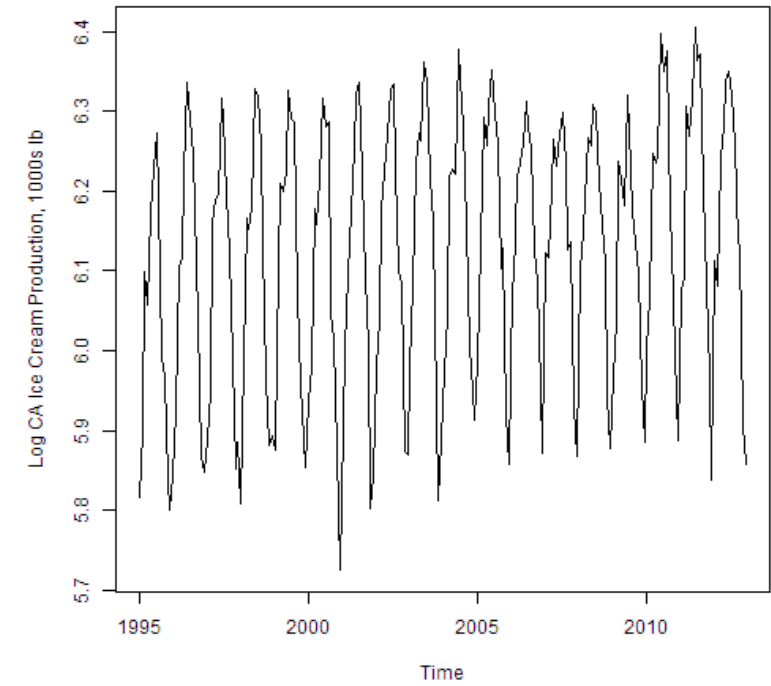
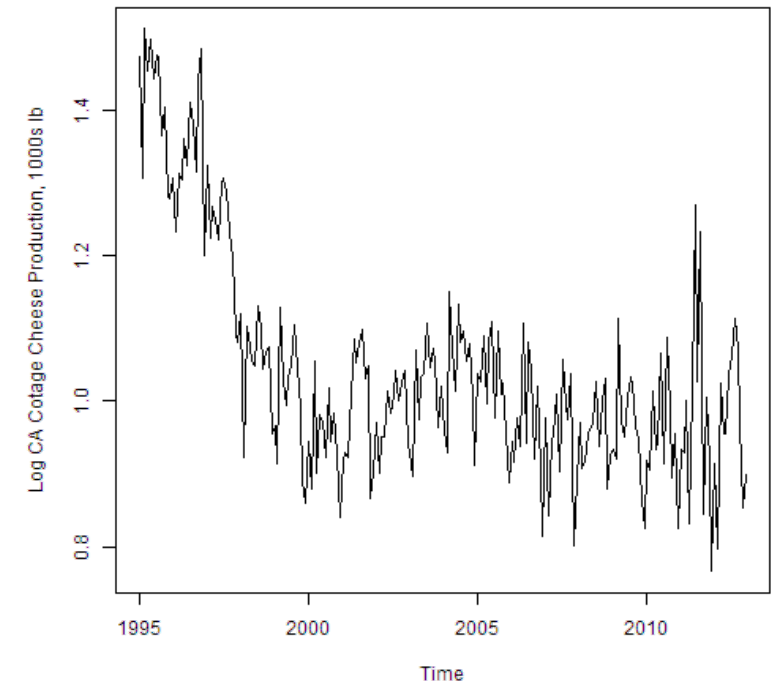
```
cadairytrain <- cadairydata[1:216,]
```

Copy

```
Ylabs <- list("Log CA Cotage Cheese Production, 1000s lb",
             "Log CA Ice Cream Production, 1000s lb",
             "Log CA Milk Production 1000s lb",
             "Log North CA Milk Milk Fat Price per 1000 lb")
```

```
Map(function(y, Ylabs){plot(cadairytrain$Time, y, xlab = "Time", ylab = Ylabs, type = "l"), cadairytrain[, 4:7], Ylabs)
```

執行此程式碼會從 [R 裝置] 輸出產生一系列時間序列圖，如圖 22 所示。請注意，時間軸的單位是日期，這是時間序列圖方法的一個極佳優點。



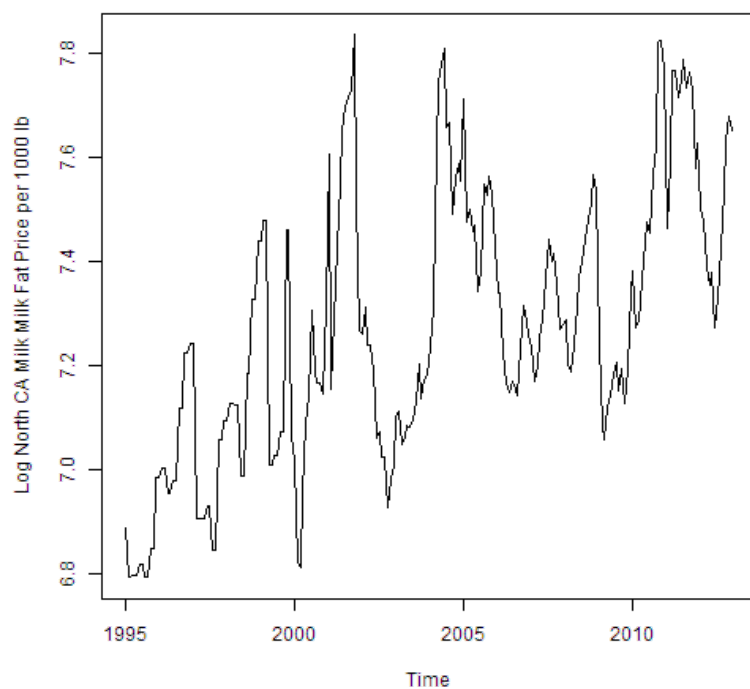
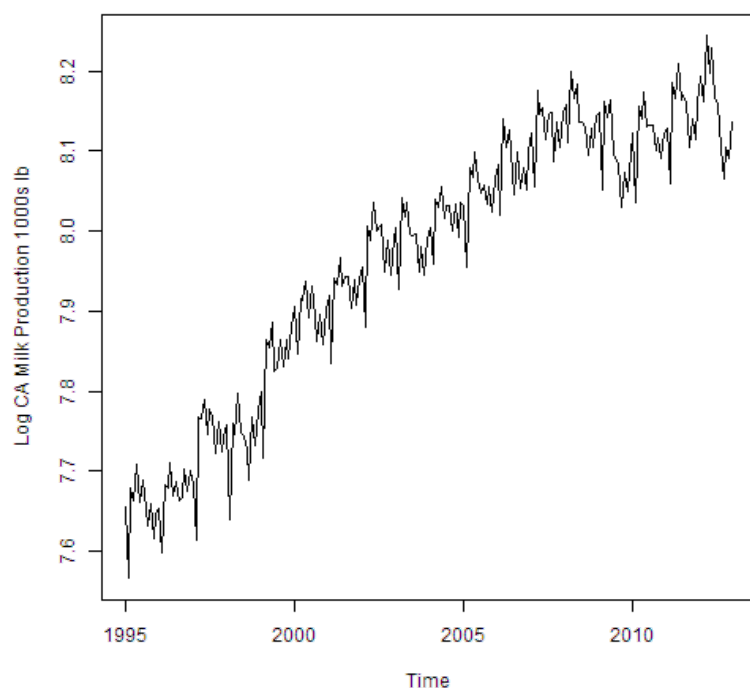


圖 22.加州乳製品產量和價格資料的時間序列圖。

趨勢模型

建立時間序列物件並查看過資料之後，讓我們開始建構加州牛奶產量資料的趨勢模型。我們可以使用時間序列迴歸來進行這項操作。不過，從圖中可以清楚看出，若要精確地為在訓練資料中所觀察到的趨勢建立模型，我們所需要的將不只是一個斜率和截距。

在資料規模較小的情況下，我會在 RStudio 中為趨勢建置模型，再將產生的模型剪下並貼到 Azure Machine Learning 中。RStudio 針對這種互動式分析提供了互動式環境。

在第一個嘗試中，我會試試最多 3 次方的多項式迴歸。這些種類的模型實際蘊藏過度配適的危險。因此，最好避免高階項。**I()** 函式禁止解譯內容 (會「依照原狀」解譯內容)，並且允許您在迴歸方程式中撰寫逐字解譯的函式。

```
r milk.lm <- lm(Milk.Prod ~ Time + I(Month.Count^2) + I(Month.Count^3), data = cadairytrain)
summary(milk.lm)
```

Copy

這會產生下列程式碼。

```
##
## Call:
## lm(formula = Milk.Prod ~ Time + I(Month.Count^2) + I(Month.Count^3),
##   data = cadairytrain)
##
## Residuals:
##   Min     1Q   Median     3Q      Max
## -0.12667 -0.02730  0.00236  0.02943  0.10586
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.33e+00  1.45e-01  43.60  <2e-16 ***
## Time          1.63e-09  1.72e-10   9.47  <2e-16 ***
## I(Month.Count^2) -1.71e-06  4.89e-06  -0.35   0.726
## I(Month.Count^3) -3.24e-08  1.49e-08  -2.17   0.031 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0418 on 212 degrees of freedom
## Multiple R-squared:  0.941, Adjusted R-squared:  0.94
## F-statistic: 1.12e+03 on 3 and 212 DF, p-value: <2e-16
```

Copy

從這個輸出的 P 值 ($\text{Pr}(>|t|)$)，我們可以看出平方項可能沒有意義。我將使用 `update()` 函式來卸除平方項，以修改此模型。

```
milk.lm <- update(milk.lm, . ~ . - I(Month.Count^2))
summary(milk.lm)
```

Copy

這會產生下列程式碼。

```
##
## Call:
## lm(formula = Milk.Prod ~ Time + I(Month.Count^3), data = cadairytrain)
##
## Residuals:
##   Min     1Q   Median     3Q      Max
## -0.12597 -0.02659  0.00185  0.02963  0.10696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.38e+00  4.07e-02  156.6  <2e-16 ***
## Time          1.57e-09  4.32e-11   36.3  <2e-16 ***
## I(Month.Count^3) -3.76e-08  2.50e-09  -15.1  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0417 on 213 degrees of freedom
## Multiple R-squared:  0.941, Adjusted R-squared:  0.94
## F-statistic: 1.69e+03 on 2 and 213 DF, p-value: <2e-16
```

Copy

這樣看起來較好。所有的項都變得有意義。不過， $2e-16$ 值是預設值，因此不應該太認真看待。

讓我們繪製顯示趨勢曲線的加州乳製品產量資料時間序列圖，來做為例行性測試。我已經在 Azure Machine Learning 執行 R 指令碼模型 (非 RStudio) 中新增下列程式碼，以建立模型並繪圖。結果顯示在「圖 23」中。

```
milk.lm <- lm(Milk.Prod ~ Time + I(Month.Count^3), data = cadairytrain)
```

Copy

```
plot(cadairytrain$Time, cadairytrain$Milk.Prod, xlab = "Time", ylab = "Log CA Milk Production 1000s lb", type = "l")
lines(cadairytrain$Time, predict(milk.lm, cadairytrain), lty = 2, col = 2)
```

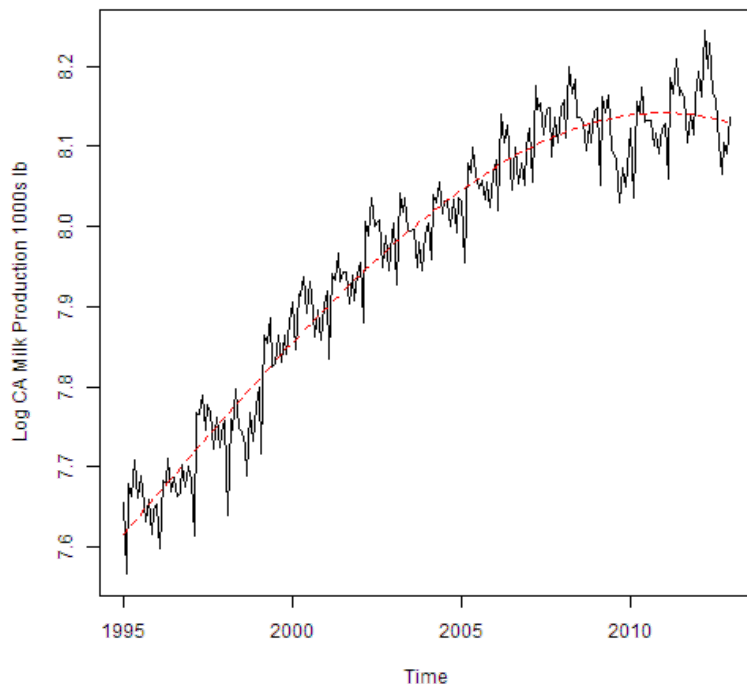


圖 23.顯示趨勢模型的加州牛奶產量資料。

看起來趨勢模型與資料非常相符。此外，看來似乎也沒有過度配適的跡象，例如模型曲線中有奇怪的擺動。

季節性模型

有了趨勢模型之後，我們還需要繼續進行來納入季節性效果。我們將使用年中月份做為線性模型中的虛擬變數，以擷取逐月的效果。請注意，當您將因素變數導入到模型中時，必須不計算截距。如果不這麼做，便會過度指定該公式，R 將會卸除其中一個想要的因素，而保留截距項。

既然我們有了令人滿意的趨勢模型，我們可以使用 `update()` 函式將新項新增到現有的模型。更新公式中的-1 會卸除截距項。目前先繼續在 RStudio 中進行：

```
milk.lm2 <- update(milk.lm, . ~ . + Month - 1)
summary(milk.lm2)
```

Copy

這會產生下列程式碼。

```
##
## Call:
## lm(formula = Milk.Prod ~ Time + I(Month.Count^3) + Month - 1,
## data = cadairytrain)
##
## Residuals:
## Min 1Q Median 3Q Max
## -0.06879 -0.01693 0.00346 0.01543 0.08726
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## Time 1.57e-09 2.72e-11 57.7 <2e-16 ***
## I(Month.Count^3) -3.74e-08 1.57e-09 -23.8 <2e-16 ***
## MonthApr 6.40e+00 2.63e-02 243.3 <2e-16 ***
## MonthAug 6.38e+00 2.63e-02 242.2 <2e-16 ***
## MonthDec 6.38e+00 2.64e-02 241.9 <2e-16 ***
```

Copy


```
## MonthFeb      6.31e+00  2.63e-02  240.1  <2e-16 ***
## MonthJan      6.39e+00  2.63e-02  243.1  <2e-16 ***
## MonthJul      6.39e+00  2.63e-02  242.6  <2e-16 ***
## MonthJun      6.38e+00  2.63e-02  242.4  <2e-16 ***
## MonthMar      6.42e+00  2.63e-02  244.2  <2e-16 ***
## MonthMay      6.43e+00  2.63e-02  244.3  <2e-16 ***
## MonthNov      6.34e+00  2.63e-02  240.6  <2e-16 ***
## MonthOct      6.37e+00  2.63e-02  241.8  <2e-16 ***
## MonthSep      6.34e+00  2.63e-02  240.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0263 on 202 degrees of freedom
## Multiple R-squared:  1, Adjusted R-squared:  1
## F-statistic: 1.42e+06 on 14 and 202 DF, p-value: <2e-16
```

我們會看到模型不再具有截距項，並且擁有 12 個重要的月份因素。這就是我們想要看到的。

讓我們繪製另一張加州乳製品產量資料的時間序列圖，看看季節性模型運作得如何。我已經在 Azure Machine Learning [執行 R 指令碼](#) 中新增下列程式碼，以建立模型並繪圖。

```
milk.lm2 <- lm(Milk.Prod ~ Time + I(Month.Count^3) + Month - 1, data = cadairytrain)
```

[Copy](#)

```
plot(cadairytrain$Time, cadairytrain$Milk.Prod, xlab = "Time", ylab = "Log CA Milk Production 1000s lb", type = "l")
lines(cadairytrain$Time, predict(milk.lm2, cadairytrain), lty = 2, col = 2)
```

在 Azure Machine Learning 中執行此程式碼會產生如圖 24 所示的圖。

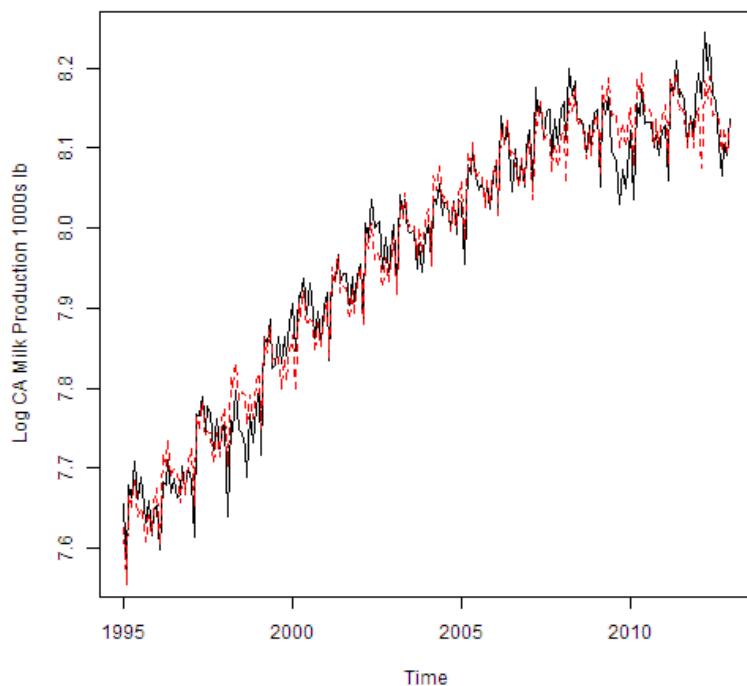


圖 24.模型包含季節性效果的加州牛奶產量。

圖 24 所示的資料相符程度相當令人振奮。趨勢和季節性效果 (每月變化) 看起來都相當合理。

接下來，對模型進行另一項檢查，讓我們看看殘差。下列程式碼會從我們的兩個模型計算預測值、計算季節性模型的殘差，然後繪製這些殘差的圖以供訓練資料使用。

```
## Compute predictions from our models
predict1 <- predict(milk.lm, cadairydata)
predict2 <- predict(milk.lm2, cadairydata)
```

[Copy](#)

```
## Compute and plot the residuals
residuals <- cadairydata$Milk.Prod - predict2
plot(cadairytrain$Time, residuals[1:216], xlab = "Time", ylab = "Residuals of Seasonal Model")
```

殘差圖如圖 25 所示。

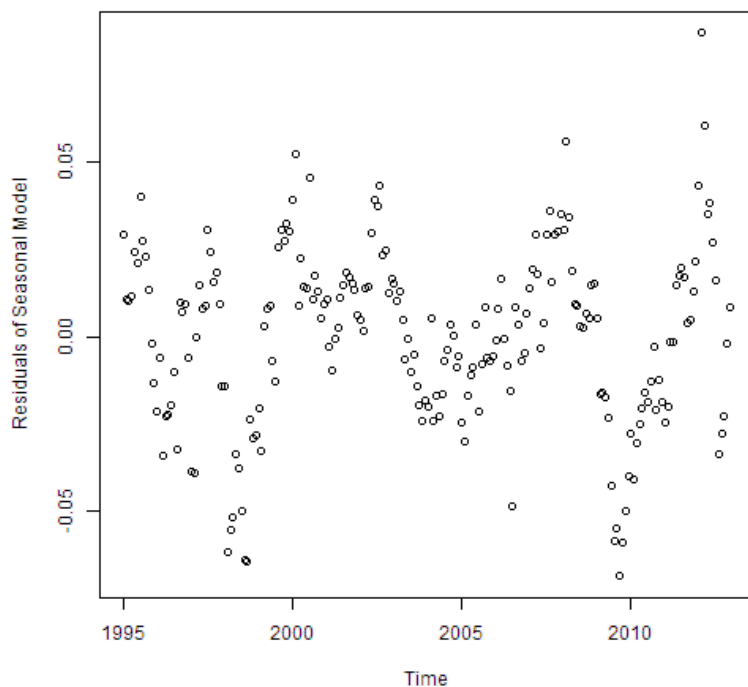


圖 25.訓練資料之季節性模型的殘差。

這些殘差看起來相當合理。除了我們模型無法很好地解釋的 2008-2009 年衰退現象之外，並沒有特定的結構。

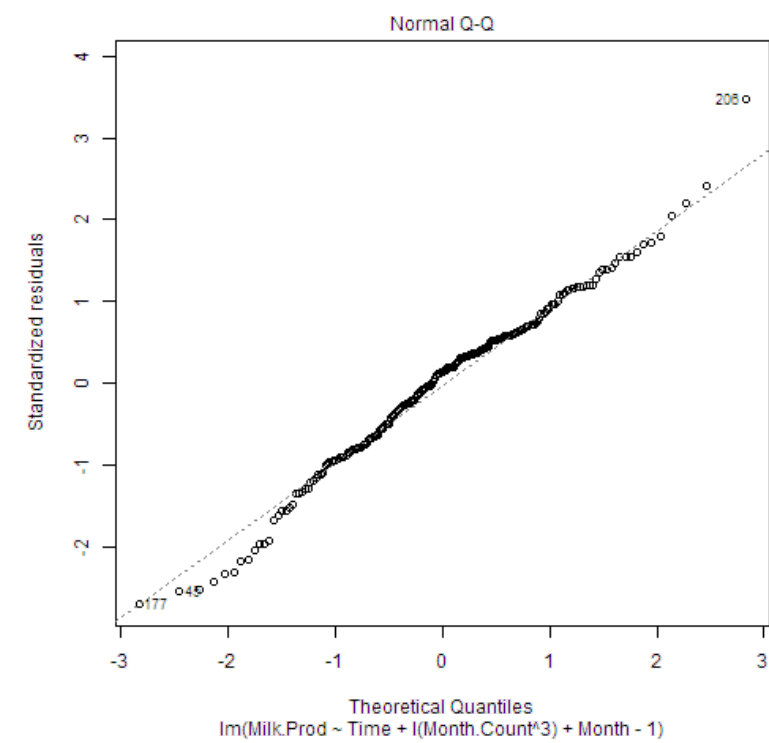
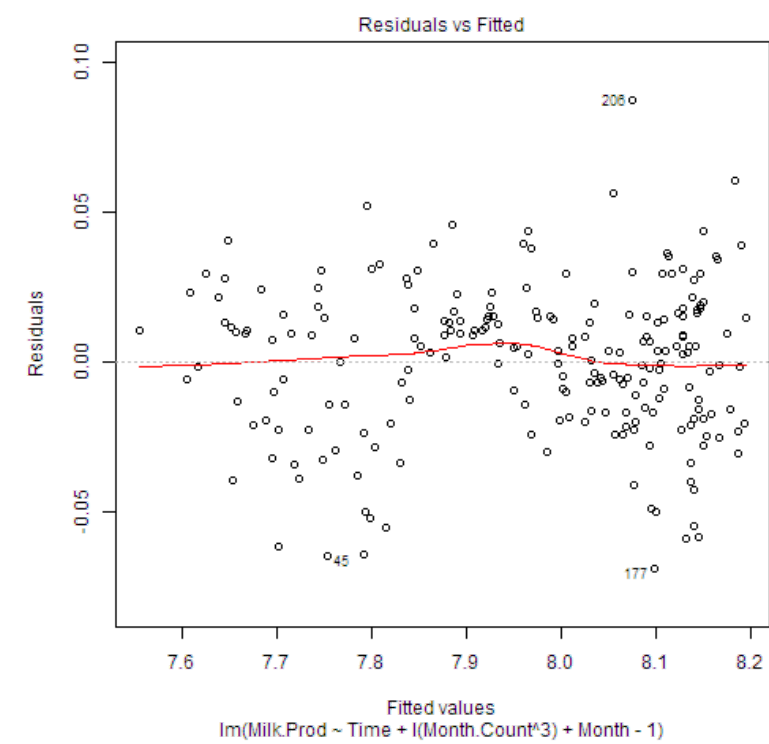
圖 25 所示的圖對於偵測殘差中任何時間相依模式來說，相當有用。我用來計算殘差及繪製殘差圖的明確方法，會讓殘差在圖上依時間排序。如果在另一方面，我已經繪製 `milk.lm$residuals` 的圖，此圖就不會依時間排序。

您也可以使用 `plot.lm()` 來產生一系列診斷圖：

```
## Show the diagnostic plots for the model
plot(milk.lm2, ask = FALSE)
```

Copy

此程式碼會產生一系列診斷圖，如圖 26 所示。



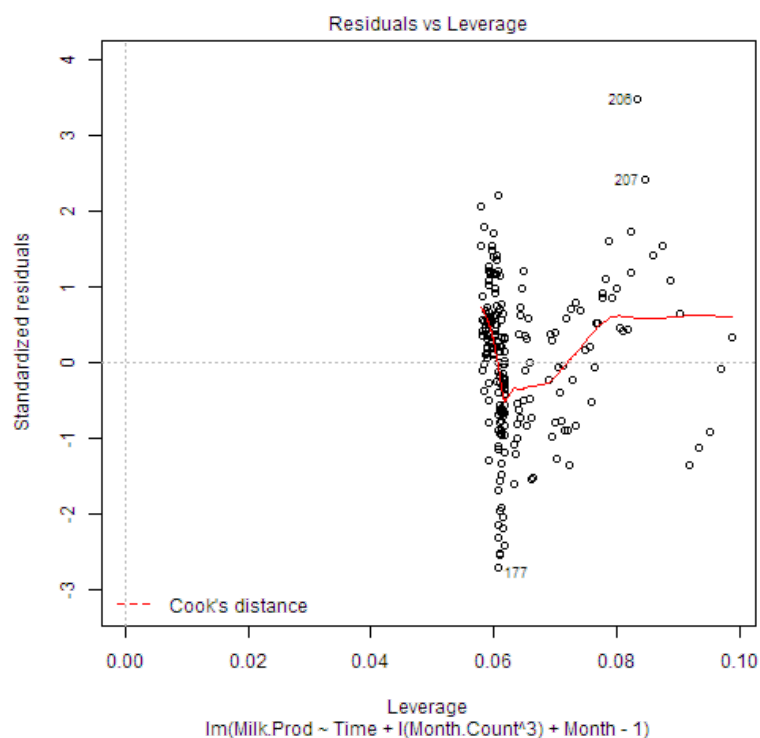
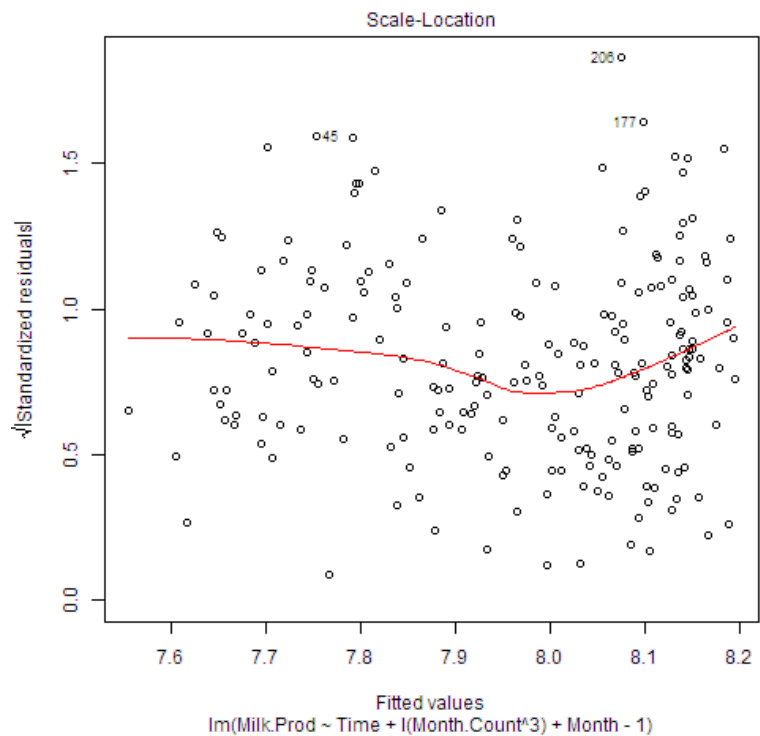


圖 26.季節性模型的診斷圖。

這些圖中有指出一些高影響力的點，但是不需要太過關注。此外，我們可以從常態分佈 Q-Q 圖看出殘差接近常態分佈，這是線性模型的重要認定依據。

預測和模型評估

距離完成我們的範例，只剩一件事情要做。我們需要計算預測，並對照實際資料來測量誤差。我們的預測將會針對 2013 年的 12 個月。我們可以針對這項不屬於我們訓練資料集之實際資料的預測，計算誤差衡量值。此外，我們可以將 18 年訓練資料的相關表現與 12 個月的測試資料做比較。

有一些衡量標準可用來衡量時間序列模型的表現。在我們的案例中，我們將使用均方根 (RMS) 誤差。下列函式會計算兩個數列間的 RMS 誤差。

```

RMS.error <- function(series1, series2, is.log = TRUE, min.length = 2){
  ## Function to compute the RMS error or difference between two
  ## series or vectors

  messages <- c("ERROR: Input arguments to function RMS.error of wrong type encountered",
    "ERROR: Input vector to function RMS.error is too short",
    "ERROR: Input vectors to function RMS.error must be of same length",
    "WARNING: Funtion rms.error has received invalid input time series.")

  ## Check the arguments
  if(!is.numeric(series1) | !is.numeric(series2) | !is.logical(is.log) | !is.numeric(min.length)) {
    warning(messages[1])
    return(NA)}

  if(length(series1) < min.length) {
    warning(messages[2])
    return(NA)}

  if((length(series1) != length(series2))) {
    warning(messages[3])
    return(NA)}

  ## If is.log is TRUE exponentiate the values, else just copy
  if(is.log) {
    tryCatch( {
      temp1 <- exp(series1)
      temp2 <- exp(series2) },
      error = function(e){warning(messages[4]); NA}
    )
  } else {
    temp1 <- series1
    temp2 <- series2
  }

  ## Compute predictions from our models
  predict1 <- predict(milk.lm, cadairydata)
  predict2 <- predict(milk.lm2, cadairydata)

  ## Compute the RMS error in a dataframe
  tryCatch( {
    sqrt(sum((temp1 - temp2)^2) / length(temp1)),
    error = function(e){warning(messages[4]); NA}
  }
)

```

與我們在 < 值轉換 > 一節中討論的 `log.transform()` 函式相同，此函式中也有許多錯誤檢查和例外狀況復原程式碼。所採用的原則相同。工作會由包裝在 `tryCatch()` 中的兩個地方完成。首先，由於我們一直以來都是使用值的對數，因此會將時間序列乘冪。其次，則是會計算實際的 RMS 誤差。

在具備測量 RMS 誤差的函式之後，讓我們建置並輸出包含 RMS 誤差的資料框架。我們將會包含只針對趨勢模型的各個項，以及針對含有季節性因素之完整模型的各個項。下列程式碼會使用我們已建構的兩個線性模型來進行此作業。

```

## Compute the RMS error in a dataframe
## Include the row names in the first column so they will
## appear in the output of the Execute R Script
RMS.df <- data.frame(
  rowNames = c("Trend Model", "Seasonal Model"),
  Traing = c(
    RMS.error(predict1[1:216], cadairydata$Milk.Prod[1:216]),
    RMS.error(predict2[1:216], cadairydata$Milk.Prod[1:216])),
  Forecast = c(
    RMS.error(predict1[217:228], cadairydata$Milk.Prod[217:228]),

```

```
RMS.error(predict2[217:228], cadairydata$Milk.Prod[217:228]))
)
RMS.df

## The following line should be executed only when running in
## Azure Machine Learning Studio
maml.mapOutputPort('RMS.df')
```

執行此程式碼會在 [結果資料集] 輸出連接埠產生如圖 27 所示的輸出。

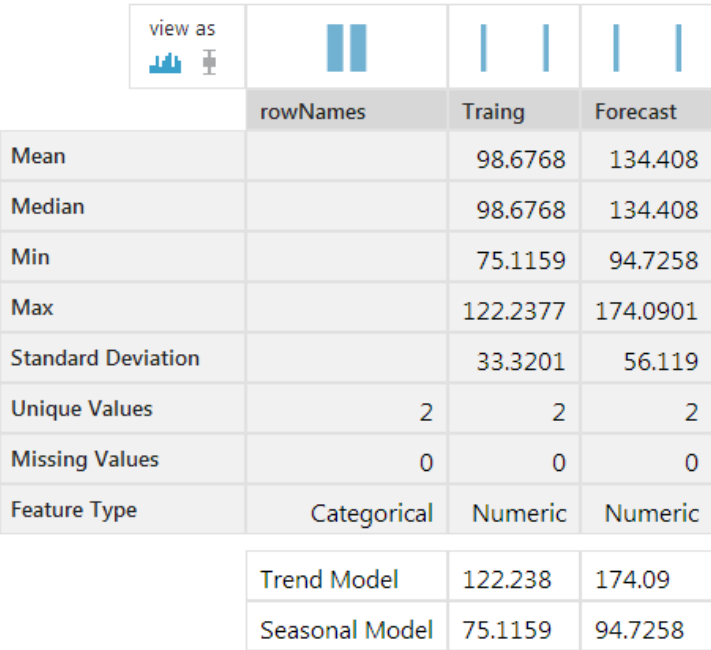


圖 27.模型的 RMS 誤差比較。

我們可以從這些結果看出，將季節性因素新增到模型中可大幅降低 RMS 誤差。不出所料，訓練資料的 RMS 誤差比預測的 RMS 誤差小一些。

附錄 A：RStudio 指南

RStudio 已經有相當充分的說明，因此在本附錄中，我將提供一些 RStudio 文件中重要小節的連結，讓您能夠輕鬆上手。

1. 建立專案

您可以使用 RStudio，以專案方式組織和管理您的 R 程式碼。下列網址提供使用專案的相關文件：
<https://support.rstudio.com/hc/articles/200526207-Using-Projects>。

建議您依照這些指示，為本文件中的 R 程式碼範例建立專案。

2. 編輯和執行 R 程式碼

RStudio 提供一個可編輯和執行 R 程式碼的整合式環境。您可以在下列網址找到相關文件：
<https://support.rstudio.com/hc/articles/200484448-Editing-and-Executing-Code>。

3. Debugging

RStudio 包含強大的偵錯功能。下列網址提供這些功能的相關文件：
<https://support.rstudio.com/hc/articles/200713843-Debugging-with-RStudio>。

下列網址提供中斷點疑難排解功能的相關文件：
<https://support.rstudio.com/hc/articles/200534337-Breakpoint-Troubleshooting>。

附錄 B：進階閱讀

此 R 程式設計教學課程涵蓋您搭配 Azure Machine Learning Studio 使用 R 語言時所需的基本知識。如果您不熟悉 R，CRAN 有提供兩本簡介：

- Emmanuel Paradis 所著的《R for Beginners》是初學者的首選，網址為 http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf。
- W. N. Venables et. al. 所著的《An Introduction to R》提供略為深入的探討，網址為 <http://cran.r-project.org/doc/manuals/R-intro.html>。

有許多 R 的相關書籍可以協助您輕鬆上手。以下是一些我認為實用的書籍：

- Norman Matloff 所著的《The Art of R Programming: A Tour of Statistical Software Design》是一本使用 R 進行程式設計的出色簡介。
- Paul Teetor 所著的《R Cookbook》提供關於使用 R 的問題與解決方法。
- Robert Kabacoff 所著的《R in Action》是另一本實用的簡介書籍。'Quick R' 網站是相當實用的資源，網址為 <http://www.statmethods.net/>。
- Patrick Burns 所著的《R Inferno》是一本令人出乎意料的幽默書籍，當中處理一些使用 R 進行程式設計時，可能遇到的較具技巧性和困難度的主題。這本書提供免費下載，網址為 <http://www.burns-stat.com/documents/books/the-r-inferno/>。
- 如果您想要深入探索 R 中的進階主題，可以看看 Hadley Wickham 所著的《Advanced R》。這本書的線上版本提供免費下載，網址為 <http://adv-r.had.co.nz/>。

您可以在「CRAN 工作檢視：時間序列分析」找到 R 時間序列封裝目錄：<http://cran.r-project.org/web/views/TimeSeries.html>。如需特定時間序列物件封裝的資訊，您應該參考該封裝的相關文件。

Paul Cowpertwait 與 Andrew Metcalfe 所著的《Introductory Time Series with R》介紹如何使用 R 進行時間序列分析。許多理論文本皆有提供 R 範例。

一些絕佳的網際網路資源：

- DataCamp：DataCamp 利用視訊課程和程式碼撰寫練習在瀏覽器中輕鬆教導 R。最新 R 技巧和封裝均有互動式教學課程。取得免費的互動式 R 教學課程，網址為 <https://www.datacamp.com/courses/introduction-to-r>
- Clarkson 大學 Kelly Black 提供的快速 R 教學課程：<http://www.cyclismo.org/tutorial/R/>
- 60+ R 資源詳列於 <http://www.computerworld.com/article/2497464/business-intelligence-60-r-resources-to-improve-your-data-skills.html>