# R 및 SQL Server 용 데이터 과학 전체 과정 연습

**원문**:https://docs.microsoft.com/ko-kr/sql/advanced-analytics/tutorials/walkthrough-data-science-end-to-end-walkthrough?view=sql-server-2016

**검토 및 편집**: 김정선(jskim@sqlroad.com), Microsoft Data Platform MVP

이 연습에서는 SQL Server 2016 또는 SQL Server 2017 에서 Microsoft R 기반 예측 모델링을 위한 전체 솔루션을 개발합니다.

이 연습은 공용 데이터 중 많이 사용되는 뉴욕 시 택시 데이터 세트를 기반으로 합니다. R 코드, SQL Server 데이터, 사용자 지정 SQL 함수를 사용하여 분류 모델을 작성하고 이를 통해 택시 운전사가 팁을 얻을 수 있는지 그 가능성을 예측합니다. 또한 R 모델을 SQL Server 에 배포하고 서버 데이터를 사용해서 해당 모델에 기반한 점수(score)를 생성 합니다.

이 예제는 영업 캠페인에 대한 고객 반응 예측, 특정 행사의 비용 지출이나 참석 예측과 같은 모든 유형의 실제 문제로 확장될 수 있습니다. 저장 프로시저를 통해 예측 모델을 호출할 수 있으므로 응용 프로그램에 쉽게 내장할 수 있습니다.

이 연습은 R 개발자들에게 R Services(In-Database)를 소개하기 위해 설계되었음으로 R 이 가능한 모든 곳에 사용됩니다. 그러나 이것이 R 이 항상 모든 작업에서 가장 좋은 도구임을 의미하지 않습니다. 데이터 집계나 특성(feature) 엔지니어링같은 특정 작업에서는 대부분 SQL Server 가 더 나은 성능을 제공할 것입니다. 특히 메모리 최적화 columnstore 인덱스와 같은 SQL Server 2017 의 새로운 기능들이 도움이 될 수 있습니다. 과정을 거치면서 가용한 최적화 방안들을 언급할 것입니다.

참고

이 연습은 원래 SQL Server 2016 용으로 개발하고 시험했습니다만, 스크린 샷과 프로시저들은 SQL Server 2017 에서도 작동할 수 있도록 최신 버전의 SQL Server Management Studio 를 사용해서 업데이트 했습니다.

## 개요

아래 예상 시간에 설치 작업은 포함되지 않습니다. 자세한 내용은 연습에 대 한 필수 구성 요소를 참조하세요.

| 항목 목록 | 예상 시간 |
| --- | --- |
| R 연습 데이터 준비<br>모델 작성에 사용되는 데이터를 가져옵니다. 공용 데이터 세트를 다운로드하고 SQL Server 데이터베이스에 로드합니다. | 30 분 |
| SQL 을 사용한 데이터 탐색<br>SQL 도구와 요약을 사용하여 데이터를 이해합니다. | 10 분 |
| R 을 사용하여 데이터 요약<br>R 을 사용하여 데이터를 탐색하고 요약을 생성합니다. | 10 분 |
| SQL Server 에서 R 을 사용하여 plot 만들기<br>R 과 SQL 을 혼합하여 로컬 및 원격 계산 컨텍스트에서 plot 을 만듭니다. | 10 분 |
| R 과 T-SQL 을 사용하여 데이터 기능을 만들기 | 10 분 |

| 항목 목록 | 예상 시간 |
| --- | --- |
| R 과 Transact-SQL 에서 사용자 지정 함수를 사용하여 특성 공학(feature engineering)을 수행합니다. 특성 공학 작업에서의 R 및 T-SQL 의 성능을 비교하세요. | |
| R 모델을 작성하고 SQL Server 에 저장<br>예측 모델 학습과 튜닝. 모델 성능을 평가. 이 연습에서는 분류 모델을 만듭니다. R 을 사용하여 모델의 정확도를 그립니다. | 15 분 |
| SQL Server 를 사용하여 R 모델 배포<br>모델을 SQL Server 데이터베이스에 저장하여 프로덕션에 모델을 배포합니다. 저장 프로시저에서 모델을 호출하여 예측을 생성합니다. | 10 분 |

## 대상 독자

이 연습은 R 또는 SQL 개발자를 위한 것입니다. R Services(In-Database)를 사용하여 엔터프라이즈 워크플로에 R 을 통합하는 방법을 소개합니다. 데이터베이스와 테이블 만들기, 데이터 가져오기, 쿼리 실행 같은 데이터베이스 작업에 익숙해야 합니다.

- 모든 SQL 과 R 스크립트가 포함됩니다.
- 사용자 환경에서 실행하려면 스크립트내 문자열을 수정할 필요도 있습니다. Visual Studio Code 같은 코드 편집기로 이러한 작업을 수행할 수 있습니다.

## 필요 구성 요소

- SQL Server 2016 의 인스턴스 또는 SQL Server 2017 의 평가 버전에 액세스할 수 있어야 합니다.

- SQL Server 인스턴스에 최소 하나의 R Services(In-Database)가 설치되어 있어야 합니다.
- 랩톱이나 다른 네트워크 컴퓨터 같은 원격 컴퓨터에서 R 명령을 실행하려면 Microsoft R Open 라이브러리를 설치해야 합니다. Microsoft R 클라이언트 또는 Microsoft R Server 를 설치할 수 있습니다. 원격 컴퓨터는 SQL Server 인스턴스에 연결할 수 있어야 합니다.
- 동일한 컴퓨터에 클라이언트와 서버를 둘 경우 "원격" 클라이언트에서 R 스크립트를 보내는데 사용할 별도의 Microsoft R 라이브러리 세트를 설치하세요. SQL Server 인스턴스용으로 설치된 R 라이브러리를 이 목적으로 사용하지 마십시오.

  서버 및 클라이언트 환경을 설정하는 자세한 방법은 SQL Server 및 R 용 데이터 과학 연습을 위한 필수 구성 요소를 참조하세요.

# SQL Server 및 R 용 데이터 과학 연습을 위한 필수 구성 요소

랩톱 또는 Microsoft R 라이브러리가 설치된 다른 컴퓨터에서 이 연습을 수행하길 권장합니다. 동일한 네트워크에서 Machine Learning 서비스와 R 언어를 사용하는 SQL Server 컴퓨터에 연결할 수 있어야 합니다.

SQL Server 와 R 개발 환경을 모두 가진 컴퓨터에서 이 연습을 실행할 수 있지만 프로덕션 환경에서는 그러한 구성을 권장하지 않습니다.

## SQL Server 에 Machine Learning 설치

설치된 R 을 지원하는 SQL Server 의 인스턴스에 접근할 수 있어야 합니다. 이 연습은 원래 SQL Server 2016 용으로 개발되고 2017 에서 시험했으므로 다음 SQL Server 버전 중 하나를 사용할 수 있어야 합니다. (각 릴리스에서 RevoScaleR 기능에 약간의 차이가 있습니다)

- SQL Server 2017 용 Machin Learning Services (In-Database)
- SQL Server 2016 R Services

자세한 내용은 SQL Server R Services(In-Database) 설정을 참조하세요.

중요

SQL Server 2016 이전 버전은 R 과의 통합을 지원하지 않지만 ODBC 데이터 원본으로서 SQL 데이터베이스를 사용할 수 있습니다.

## R 개발 환경 설치

이 연습을 위해 R 개발 환경을 사용하길 권장합니다. 아래 몇 가지 제안이 있습니다.

- **R Tools for Visual Studio** (RTVS)는 Microsoft R 지원, 인텔리센스, 디버깅을 제공하는 무료 플러그 인입니다. R 서버와 SQL Server Machine Learning 서비스 모두에 사용할 수 있습니다. 다운로드하려면 R Tools for Visual Studio 를 참조하세요.
- **Microsoft R Client** 는 RevoScaleR 패키지를 사용하여 R 개발을 지원하는 경량 개발 도구입니다. 사용하려면 Microsoft R Client 시작하기를 참조하세요.
- **RStudio** 는 R 개발용으로 보다 대중적인 환경 중 하나입니다. 자세한 내용은 https://www.rstudio.com/products/RStudio/를 참조하세요.

  RStudio 의 일반 설치나 다른 환경을 사용해서 이 자습서를 완료할 수 없습니다. Microsoft R Open 용 R 패키지와 연결 라이브러리도 설치해야 합니다. 자세한 내용은 데이터 과학 클라이언트 설정을 참조하세요.

- SQL Server R 이나 R 클라이언트를 설치할 때 기본적인 R 도구 (R.exe, RTerm.exe, RScripts.exe) 또한 설치됩니다. IDE 설치를 원하지 않는다면 이러한 도구를 사용할 수 있습니다.

## SQL Server 인스턴스 및 데이터베이스에 대한 사용 권한 가져오기

스크립트를 실행하고 데이터를 업로드하기 위해서 SQL Server 인스턴스에 연결하려면, 데이터베이스 서버에 유효한 로그인이 있어야 합니다. SQL 로그인 또는 Windows 통합 인증을 사용할 수 있습니다. R 을 사용할 데이터베이스 계정에 다음 권한을 구성하도록 데이터베이스 관리자에게 요청하세요.

- 데이터베이스, 테이블, 함수 및 저장 프로시저 만들기

- 테이블에 데이터 쓰기
- R 스크립트 실행 권한 (`GRANT EXECUTE ANY EXTERNAL SCRIPT to <user>`)

  이 연습에서는 SQL 로그인 **RTestUser** 를 사용했습니다. 일반적으로 Windows 통합 인증을 사용하길 권장하지만 일부 데모 목적으로 SQL 로그인을 사용하는 것이 더 간단합니다.

## 변경 목록

- 이 예제는 SQL Server 2016 R Services 를 사용 하 여 개발 원래 되었습니다. 그런데 2016 SP1 용 Microsoft R 구성 요소에 변경 있었습니다. 특히 *varsToDrop* 및 *varsToKeep* 매개변수가 SQL Server 데이터 원본용으로 더 이상 지원되지 않습니다. 따라서 SP1 이전 자습서 버전을 다운로드한 경우 SP1 이후 빌드에서는 작동하지 않습니다.
- 현재 버전의 예제는 SQL Server 2017 Machine Learning Services (RC1 과 RC2)의 사전-릴리스 빌드를 사용하여 시험했습니다. 일반적으로 거의 모든 단계가 2016 SP1 에서 2017 사이에서 수정없이 동작합니다.

## 다음 단원

PowerShell 을 사용하여 데이터 준비

# PowerShell 를 사용한 데이터 준비(연습)

이 시점까지 다음 중 하나를 설치해야합니다:

- SQL Server 2016 R Services
- SQL Server 2017 Machine Learning Services, R 언어 사용

이 단원에서는 Github 리포지토리로부터 연습에 사용되는 데이터, R 패키지 및 R 스크립트를 다운로드합니다. 편의를 위해 PowerShell 스크립트를 사용하여 전체를 다운로드할 수 있습니다.

또한 서버와 R 워크스테이션 모두에 몇 가지 추가 R 패키지를 설치해야 합니다. 단계가 설명됩니다.

그런 다음 두 번째 PowerShell 스크립트인 RunSQL_R_Walkthrought.ps1 을 사용해서 모델링과 채점에 사용되는 데이터베이스를 구성합니다. 이 스크립트는 지정한 데이터베이스로 데이터를 대량 로드한 다음 데이터 과학 작업을 단순화하기 위한 몇 가지 SQL 함수와 저장 프로시저를 생성합니다.

이제 시작하겠습니다.

## 1. 데이터와 스크립트 다운로드

필요한 모든 코드가 GitHub 리포지토리에서 제공됩니다. PowerShell 스크립트를 사용하여 파일의 로컬 복사본을 만들 수 있습니다.

1. 데이터 과학 클라이언트 컴퓨터에서 관리자 권한으로 Windows PowerShell 명령 프롬프트를 엽니다.
2. 다운로드 스크립트를 오류 없이 실행하기위해 아래 명령을 실행합니다. 시스템 기본값을 변경하지 않고 일시적으로 스크립트를 허용합니다.

복사

```
Set-ExecutionPolicy Unrestricted -Scope Process -Force
```

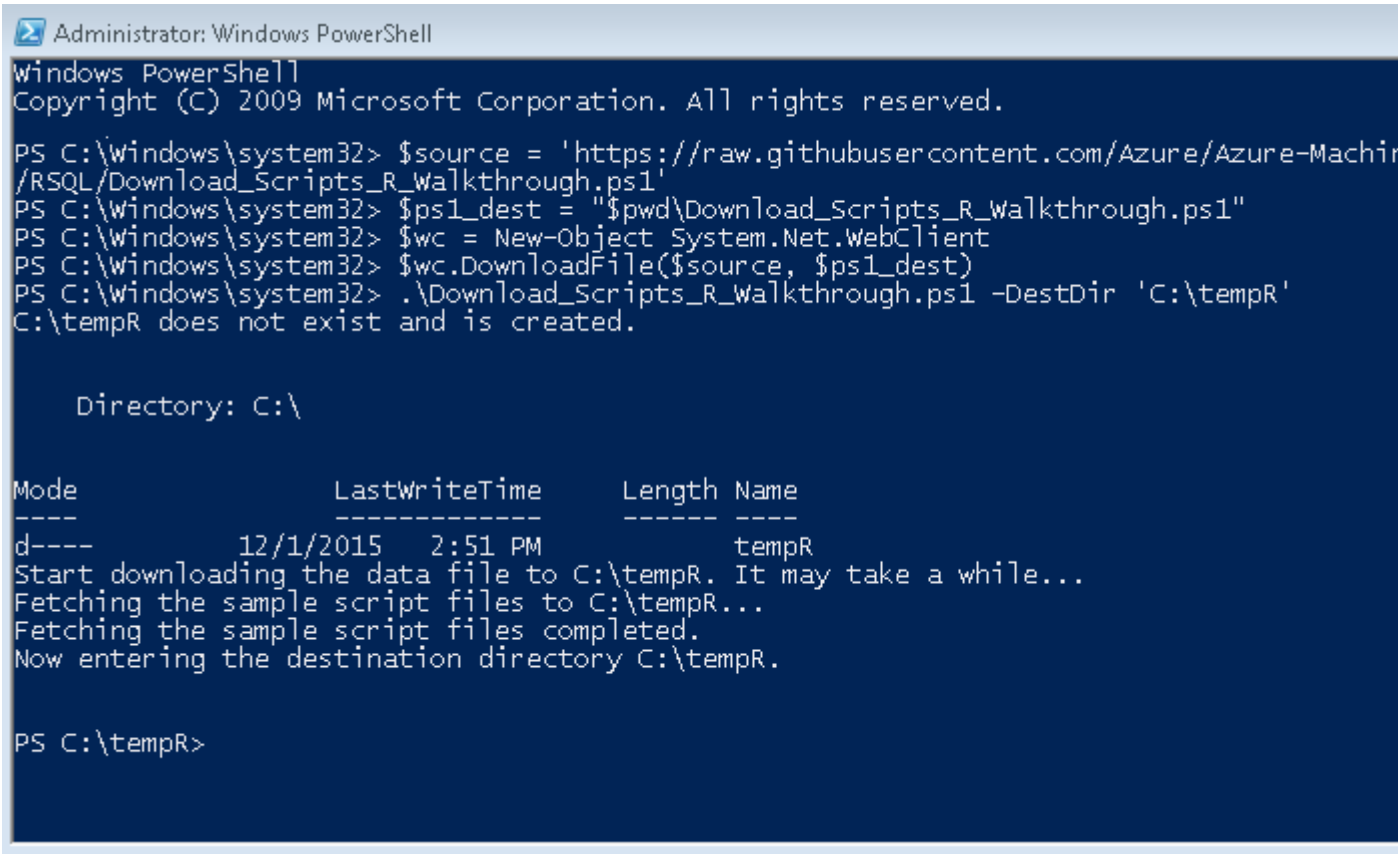
3. 다음 Powershell 명령을 실행하여 스크립트 파일을 로컬 디렉터리로 다운로드합니다. 디렉터리를 다르게 지정하지 않으면 기본적으로 `C:\tempR 폴더가` 만들어지고 모든 파일이 저장됩니다.

복사

```
$source = 'https://raw.githubusercontent.com/Azure/Azure-
MachineLearning-
DataScience/master/Misc/RSQL/Download_Scripts_R_Walkthrough.ps1'

$ps1_dest = "$pwd\Download_Scripts_R_Walkthrough.ps1"

$wc = New-Object System.Net.WebClient

$wc.DownloadFile($source, $ps1_dest)

.\Download_Scripts_R_Walkthrough.ps1 –DestDir 'C:\tempR'
```

다른 디렉터리에 파일을 저장하려면 *DestDir* 매개 변수의 값을 편집해서 컴퓨터의 다른 폴더를 지정하세요. 존재하지 않는 폴더 이름을 입력하면 PowerShell 스크립트가 해당 폴더를 만듭니다.

4. 다운로드는 다소 시간이 걸릴 수 있습니다. 다운로드가 완료된 후 Windows PowerShell 명령 콘솔은 아래와 같이 표시됩니다.




5. PowerShell 콘솔에서 `ls` 명령을 실행하여 *DestDir* 에 다운로드된 파일 목록을 확인할 수 있습니다. 파일에 대한 설명은 샘플에 포함된 내용을 참조합니다.

## 2. 필수 R 패키지 설치

이 연습을 수행하려면 R Services(In-Database)의 일부로 기본 설치되지 않는 몇 가지 R 라이브러리가 필요합니다. 솔루션을 개발하는 클라이언트와 솔루션을 배포하는 SQL Server 컴퓨터 양쪽 모두에 패키지를 설치해야 합니다.

### 클라이언트에 필수 패키지 설치

다운로드한 R 스크립트에는 이러한 패키지를 다운로드하고 설치하는 명령이 포함되어 있습니다.

1. R 환경에서 스크립트 파일 RSQL_R_Walkthrough.R 을 엽니다.

2. 아래 행을 강조 표시하고 실행합니다.

복사

```
# Install required R libraries, if they are not already installed.

if (!('ggmap' %in%
rownames(installed.packages()))){install.packages('ggmap')}

if (!('mapproj' %in%
rownames(installed.packages()))){install.packages('mapproj')}

if (!('ROCR' %in%
rownames(installed.packages()))){install.packages('ROCR')}

if (!('RODBC' %in%
rownames(installed.packages()))){install.packages('RODBC')}
```

일부 패키지는 필요한 패키지도 설치합니다. 모두 약 32 개의 패키지가 필요합니다.

### 서버에 필수 패키지 설치

SQL Server 에 패키지를 설치할 수 있는 여러가지 방법이 있습니다. 예를 들어 SQL Server 는 데이터베이스 관리자가 패키지 리포지토리를 만들고 사용자가 자신의 패키지를 설치할 수 있도록 권한을 할당할 수 있는 패키지 관리 기능을 제공합니다. 그런데 컴퓨터 관리자의 경우 R 을 사용해서 새로운 패키지를 설치할 수도 있습니다.

### 참고

서버에서 메시지가 표시되더라도 사용자 라이브러리에 설치하지 마십시오. 사용자 라이브러리에 설치하면 SQL Server 인스턴스가 해당 패키지를 찾거나 실행할 수 없습니다. 자세한 내용은 SQL Server 에 새 R 패키지 설치를 참조하세요.

1. SQL Server 컴퓨터에서 RGui.exe 를 **관리자 권한으로** 엽니다. 기본값으로 SQL Server R Services 를 설치한 경우 RGui.exe 는 C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\R_SERVICES\bin\x64)에 있습니다.
2. R 프롬프트에서 다음 R 명령을 실행합니다.

복사

```
install.packages("ggmap", lib=grep("Program Files", .libPaths(), value=TRUE)[1])

install.packages("mapproj", lib=grep("Program Files", .libPaths(), value=TRUE)[1])

install.packages("ROCR", lib=grep("Program Files", .libPaths(), value=TRUE)[1])

install.packages("RODBC", lib=grep("Program Files", .libPaths(), value=TRUE)[1])
```

- 이 예제에서는 R grep 함수로 사용 가능한 경로의 벡터를 검색하고 그 중 "Program Files"를 포함하는 경로를 찾습니다. 자세한 내용은 http://www.rdocumentation.org/packages/base/functions/grep 을 참조하세요.
- 패키지가 이미 설치되었다고 생각되면 `installed.packages()`를 실행하여 설치된 패키지 목록을 확인하세요.

## 3. RunSQL_R_Walkthrough.ps1 으로 환경 준비

다운로드는 데이터 파일, R 스크립트, T-SQL 스크립트와 더불어 PowerShell 스크립트 `RunSQL_R_Walkthrough.ps1`를 포함합니다. 이 스크립트는 다음 작업을 수행합니다.

- SQL Server 용 SQL Native Client 와 명령줄 유틸리티가 설치되어 있는지 확인합니다. 명령줄 도구는 bcp 유틸리티를 실행하기 위해 필요합니다. 이 유틸리티는 SQL 테이블에 데이터를 빠르게 대량로드하는데 사용됩니다.

- 지정된 SQL Server 인스턴스에 연결하고 Transact-SQL 스크립트를 실행해서 데이터베이스를 구성하고 모델과 데이터를 위한 테이블을 만듭니다.
- SQL 스크립트를 실행하여 여러 개의 저장 프로시저를 만듭니다.
- 이전에 다운로드한 데이터를 `nyctaxi_sample` 테이블에 로드합니다.
- 지정한 데이터베이스 이름을 사용하도록 R 스크립트 파일의 인수를 다시 씁니다.

  솔루션을 구성하는 컴퓨터에서 이 스크립트를 실행하세요: 예를 들어 R 코드를 개발하고 검증하는 랩톱. 데이터 과학 클라이언트로 부르게 될 이 컴퓨터는 명명된 파이프 프로토콜을 사용하는 SQL Server 컴퓨터에 연결할 수 있어야 합니다.

1. PowerShell 명령줄을 **관리자** 권한으로 엽니다.
2. 스크립트를 다운로드한 폴더로 이동하고 그림과 같이 스크립트 이름을 입력하세요. Enter 키를 누릅니다.

   복사

   ```
   .\RunSQL_R_Walkthrough.ps1
   ```

3. 다음 각 매개변수마다 프롬프트가 표시 됩니다.

   **데이터베이스 서버 이름**: Machine Learning 서비스 또는 R Services 가 설치된 SQL Server 인스턴스의 이름입니다.

   네트워크 요구 사항에 따라 인스턴스 이름에 하나 이상의 서브넷 이름이 필요할 수도 있습니다. 예를 들어 MYSERVER 가 작동하지 않으면 myserver.subnet.mycompany.com 을 시도해 보세요.

   **만들려는 데이터베이스의 이름**: 예를 들어 **Tutorial** 또는 **Taxi** 를 입력할 수 있습니다.

**사용자 이름**: 데이터베이스 액세스 권한이 있는 계정을 제공합니다. 두 가지 옵션이 있습니다:

- CREATE DATABASE 권한이 있는 SQL 로그인의 이름을 입력하고 이어진 프롬프트에서 SQL 암호를 제공합니다.
- Windows 인증을 사용하려면 이름 입력없이 ENTER 키를 누른 뒤 보안 프롬프트에서 Windows 암호를 입력합니다. PowerShell 에서 다른 Windows 사용자 이름을 입력할 수는 없습니다.
- 유효한 사용자를 지정하지 않으면 스크립트 기본적으로 Windows 통합 인증을 사용합니다.

경고

PowerShell 스크립트의 프롬프트를 사용하여 자격 증명을 제공하면 업데이트된 스크립트 파일에 일반 텍스트로 암호가 기록됩니다. 필요한 R 개체를 만든 직 후 파일을 편집해서 자격 증명을 제거하십시오.

**csv 파일 경로**: 데이터 파일의 전체 경로를 제공합니다. 기본 경로와 파일명은 `C:\tempR\nyctaxi1pct.csv` 입니다.

4. Enter 키를 눌러 스크립트를 실행합니다.

스크립트에서 파일을 다운로드하고 데이터베이스에 데이터를 자동으로 로드합니다. 이 작업은 다소 시간이 걸릴 수 있습니다. PowerShell 창에서 상태 메시지를 확인합니다.

대량 가져오기 또는 다른 단계가 실패하면 문제 해결 절에 설명된대로 수동으로 데이터를 로드할 수 있습니다.

**결과(성공적으로 완료)**

```
복사
```

```
Execution successful

Completed registering all stored procedures used in this
walkthrough.

This step (registering all stored procedures) takes 0.39 seconds.

Plug in the database server name, database name, user name and
password into the R script file

This step (plugging in database information) takes 0.48 seconds.
```

다음 단원으로 이동하려면 이 링크 클릭합니다: SQL 을 사용 하여 데이터를 보고 탐색하기

## 문제 해결

PowerShell 스크립트에 문제가 있다면 수동으로 전체 혹은 일부를 실행할 수 있습니다. 이번 절에서 몇 가지 공통적인 문제와 해결 방법을 나열합니다.

### PowerShell 스크립트가 데이터를 다운로드하지 않음

데이터를 수동으로 다운로드하려면 다음 링크를 마우스 오른쪽 단추로 클릭하고 **다른 이름으로 대상 저장**을 선택합니다.

http://getgoing.blob.core.windows.net/public/nyctaxi1pct.csv

다운로드한 데이터 파일의 경로와 데이터가 저장된 파일 이름을 적어둡니다. **bcp** 를 사용하여 테이블에 데이터를 로드하려면 전체 경로가 필요합니다.

### 데이터를 다운로드할 수 없음

데이터 파일이 큽니다. 인터넷 연결 상태가 비교적 좋은 컴퓨터를 사용하지 않으면 다운로드가 시간 초과 될 수 있습니다.

### 연결할 수 없거나 스크립트에 실패

스크립트 중 하나에서 이 오류가 발생할 수 있습니다: *SQL Server 에 연결을 설정하는 동안에 네트워크 또는 인스턴스 관련 오류가 발생했습니다.*

- 인스턴스 이름의 철자를 확인합니다.
- 전체 연결 문자열을 확인합니다.
- 네트워크 요구 사항에 따라 하나 이상의 서브넷 이름을 사용하여 인스턴스 이름을 한정해야 할 수도 있습니다. 예를 들어 MYSERVER 가 작동하지 않는 경우 myserver.subnet.mycompany.com 을 사용해 보세요.
- Windows 방화벽이 SQL Server 연결을 허용하는지 확인합니다.
- 서버를 등록하고 원격 연결을 허용하는지 확인합니다.
- 명명된 인스턴스를 사용한다면 SQL Browser 서비스를 켜서 연결을 보다 쉽게 합니다.

## 네트워크 오류 또는 프로토콜 찾을 수 없음

- 인스턴스가 원격 연결을 지원하는지 확인합니다.
- 지정된 SQL 사용자가 원격으로 데이터베이스에 연결할 수 있는지 확인합니다.
- 인스턴스에서 명명된 파이프를 사용하도록 설정합니다.
- 계정의 사용 권한을 확인합니다. 지정한 계정이 새 데이터베이스를 만들고 데이터를 업로드할 수 있는 권한이 없을 수 있습니다.

## bcp 를 실행하지 못함

- 컴퓨터에서 bcp 유틸리티 를 사용할 수 있는지 확인합니다. PowerShell 창 또는 Windows 명령 프롬프트에서 **bcp** 를 실행할 수 있습니다.
- 오류가 발생하며 **bcp** 유틸리티의 위치를 PATH 시스템 환경 변수에 추가한 후 다시 시도합니다.

## 테이블 스키마가 생성되었지만 테이블에 데이터가 없음

스크립트의 나머지 부분이 문제 없이 실행된 경우 다음과 같이 명령 줄에서 **bcp** 를 호출하여 수동으로 테이블에 데이터를 업로드할 수 있습니다.

**SQL 로그인 사용**

복사

```
bcp TutorialDB.dbo.nyctaxi_sample in c:\tempR\nyctaxi1pct.csv -t ',' -S rtestserver.contoso.com -f C:\tempR\taxiimportfmt.xml -F 2 -C "RAW" -b 200000 -U <SQL login> -P <password>
```

**Windows 인증 사용**

복사

```
bcp TutorialDB.dbo.nyctaxi_sample in c:\tempR\nyctaxi1pct.csv -t ',' -S rtestserver.contoso.com -f C:\tempR\taxiimportfmt.xml -F 2 -C "RAW" -b 200000 -T
```

- `in` 키워드는 데이터 이동 방향을 지정합니다.
- **-f** 인수를 사용하려면 서식 파일의 전체 경로를 지정해야 합니다. **in** 옵션을 사용하는 경우 서식 파일이 필요합니다.
- SQL 로그인으로 bcp 를 실행하는 경우 **-U** 와 **-P** 인수를 사용합니다.
- Windows 통합 인증을 사용하는 경우 **-T** 인수를 사용합니다.

스크립트가 데이터를 로드하지 않으면 구문을 확인하고 서버 이름이 네트워크에 대해 올바르게 지정되었는지 확인합니다. 예를 들어 서브넷을 포함하고 명명된 인스턴스에 연결할 경우 컴퓨터 이름을 포함시킵니다. 대량 업로드를 수행 하는 로그인에 권한이 있는지 확인합니다.

## 프롬프트없이 스크립트를 실행하고 싶다

다음 템플릿을 사용해서 단일 명령 줄에서 필요한 모든 매개변수를 지정할 수 있습니다.

복사

```
.\RunSQL_R_Walkthrough.ps1 -server <server address> -dbname <new db
name> -u <user name> -p <password> -csvfilepath <path to csv file>
```

다음 예제는 SQL 로그인을 사용하여 스크립트를 실행합니다.

복사

```
.\RunSQL_R_Walkthrough.ps1 -server MyServer.subnet.domain.com -
dbname MyDB –u SqlUserName –p SqlUsersPassword -csvfilepath
C:\temp\nyctaxi1pct.csv
```

- *SqlUserName* 의 자격 증명을 사용하여 지정된 인스턴스 및 데이터베이스에 연결합니다.
- *C:\temp\nyctaxi1pct.csv* 파일에서 데이터를 가져옵니다.
- *MyServer* 라는 SQL Server *인스턴스의* MyDB 데이터베이스에 있는 *dbo.nyctaxi_sample* 테이블에 데이터를 로드합니다.

### 로드된 데이터에 중복이 있음

데이터베이스에 동일한 이름과 동일한 스키마의 테이블이 존재하는 경우 **bcp** 는 기존 데이터를 덮어쓰지 않고 새로운 데이터 사본을 삽입합니다.

중복 데이터를 방지 하려면 스크립트를 다시 실행 하기 전에 기존 테이블을 자릅니다.

## 샘플에 포함된 내용

GitHub 리포지토리에서 파일을 다운로드 하면 다음과 같습니다:

- CSV 형식의 데이터; 자세한 내용은 학습 및 데이터 채점 을 참조합니다.
- 환경 준비를 위한 PowerShell 스크립트
- bcp 를 사용하여 SQL Server 에 데이터를 가져오기 위한 XML 서식 파일

- 여러 T-SQL 스크립트
- 이 연습을 실행하는 데 필요한 모든 R 코드

### 데이터 학습 및 채점

데이터는 뉴욕 시 택시 데이터 세트의 견본 샘플로, 2017 년에 각 승차에서 지불한 요금과 팁을 포함해 1 억 7300 만 건의 개별 승차 기록을 포함합니다. 데이터를 보다 쉽게 사용하기 위해 Microsoft 데이터 과학 팀이 다운샘플링을 수행하여 데이터의 1%만 얻었습니다. 이 데이터는 Azure 의 공용 Blob 저장소 컨테이너에 .CSV 형식으로 공유됩니다. 원본 데이터에는 350MB 이하의 압축되지 않은 파일입니다.

- 공용 데이터 세트: NYC Taxi and Limousine Commission
- [뉴욕 택시 데이터 세트에서 Azure ML 모델 만들기](https://blogs.technet.microsoft.com/machinelearning/2015/04/02/building-azure-ml-models-on-the-nyc-taxi-dataset/

### PowerShell 및 R 스크립트 파일

- **RunSQL_R_Walkthrough.ps1** PowerShell 을 사용해서 이 스크립트를 먼저 실행합니다. SQL 스크립트로 데이터베이스에 데이터를 로드합니다.
- **taxiimportfmt.xml** BCP 유틸리티에서 데이터베이스에 데이터를 로드하는 데 사용하는 형식 정의 파일입니다.
- **RSQL_R_Walkthrough.R** 데이터 분석 및 모델링을 위해 단원의 나머지 부분에서 사용 되는 핵심 R 스크립트입니다. SQL Server 데이터를 탐색하고, 분류 모델을 작성하고, 플롯을 만드는 데 필요한 모든 R 코드를 제공합니다.

### T-SQL 스크립트 파일

PowerShell 스크립트는 SQL Server 인스턴스에서 여러 Transact-SQL 스크립트를 실행합니다. 다음 표는 Transact-SQL 스크립트와 용도입니다.

| **SQL 스크립트 파일 이름** | **Description** |
|---|---|
| create-db-tb-upload-data.sql | 데이터베이스와 두 개의 테이블을 만듭니다.<br><br>*nyctaxi_sample*: NYC 택시 데이터 세트의 1% 샘플인 학습용 데이터를 저장하는 테이블입니다. 저장소와 쿼리 성능 향상을 위해 클러스터형 columnstore 인덱스가 테이블에 추가됩니다.<br><br>*nyc_taxi_models*: 학습된 모델을 이진 형식으로 저장하는데 사용되는 테이블입니다. |
| PredictTipBatchMode.sql | 새로운 관측(observations)에 대한 레이블(labels)을 예측하기위해 학습된 모델을 호출하는 저장 프로시저를 만듭니다. 입력 매개 변수로 쿼리를 받습니다. |
| PredictTipSingleMode.sql | 새로운 관측에 대한 레이블을 예측하기위해 학습된 분류 모델을 호출하는 저장 프로시저를 만듭니다. 새로운 관측의 변수가 인라인 매개 변수로 전달됩니다. |
| PersistModel.sql | 데이터베이스의 테이블에 분류 모델의 이진 표현을 저장하는 데 도움이 되는 저장 프로시저를 만듭니다. |
| fnCalculateDistance.sql | 승하차 위치 간의 직접 거리를 계산하는 SQL 스칼라 반환 함수를 만듭니다. |

| SQL 스크립트 파일 이름 | Description |
| --- | --- |
| fnEngineerFeatures.sql | 분류 모델 학습을 위해 특성(feature)를 만드는 SQL 테이블 반환 함수를 만듭니다. |

이 연습에서 사용된 T-SQL 은 검증을 거쳤으며 R 코드에서 그대로 실행될 수 있습니다. 그러나 추가 실험을 원하거나 자체 솔루션을 개발하는 경우엔 전용 SQL 개발 환경에서 쿼리를 검증하고 튜닝한 뒤 R 코드에 추가하는 것을 권장합니다.

- Visual Studio Code 용 mssql 확장은 대부분의 데이터베이스 개발 작업을 지원하는 쿼리를 실행할 수 있는 무료 경량 환경입니다.
- SQL Server Management Studio 는 SQL Server 데이터베이스 개발 및 관리를 위해 제공되는 강력한 무료 도구입니다.

## 다음 단원

SQL 를 사용한 데이터 관찰 및 탐색.

## 이전 단원

R 및 SQL Server 용 데이터 과학 전체 과정 연습

데이터 과학 연습을 위한 필수 구성 요소

# SQL 을 사용한 데이터 관찰 및 탐색(연습)

SQL Server 를 사용하여 다운로드한 데이터 확인

다음 단원

이전 단원

데이터 탐색은 데이터 모델링의 중요한 한 부분이며 분석에 사용되는 데이터의 요약 정보를 검토하고 데이터를 시각화하는 작업을 포함합니다. 이 단원에서는 Transact-SQL 과 R Services(In-Database)에 포함된 R 함수 모두를 사용해서 데이터 개체를 탐색하고 플롯을 생성합니다.

그런 다음 R Services(In-Database)로 설치된 패키지에서 제공되는 새로운 함수를 사용하여 데이터를 시각화하는 플롯을 생성합니다.

팁

이미 R 을 잘 안다면?

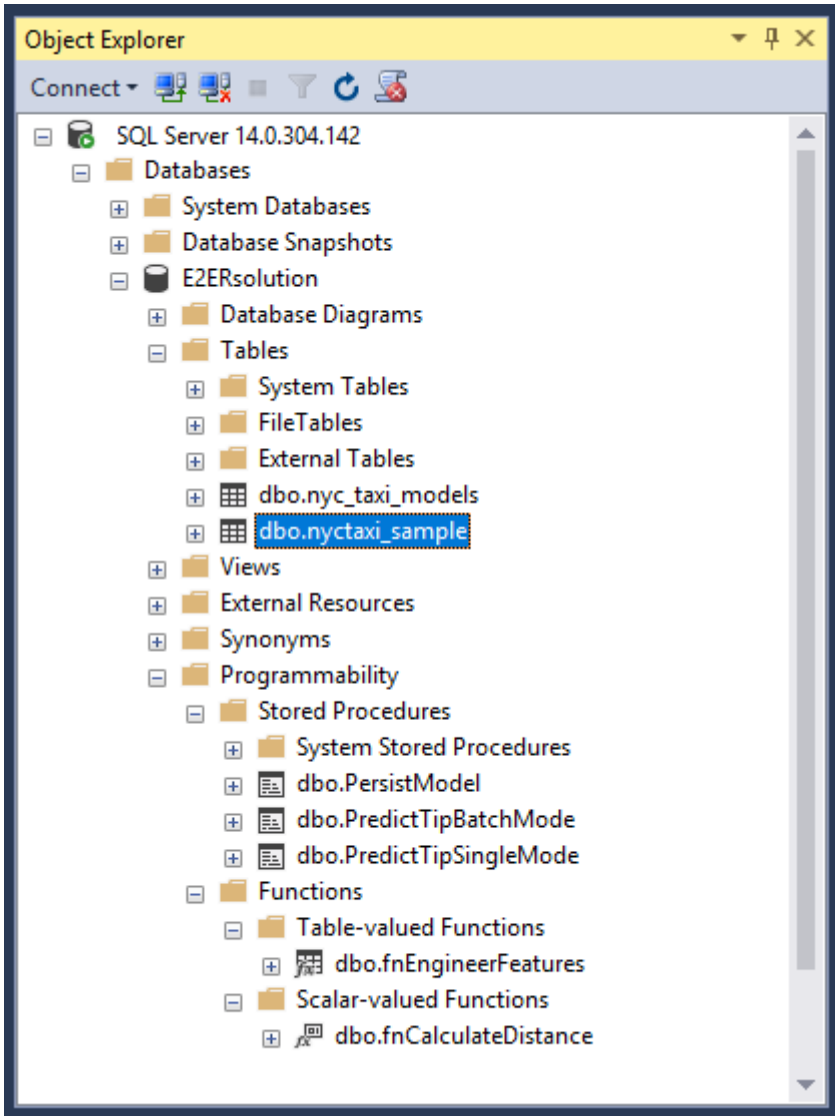
모든 데이터를 다운로드하고 필요한 환경이 준비되었으므로 RStudio 나 다른 환경에서 전체 R 스크립트를 실행하고 직접 그 기능을 살펴볼 수 있습니다. RSQL_Walkthrough.R 파일을 열고 각 줄을 실행하거나 전체 스크립트를 데모로 실행합니다.

RevoScaleR 함수에 대한 추가 설명과 R 에서 SQL Server 데이터 작업에 대한 팁을 얻으려면 자습서를 계속 진행합니다. 정확히 동일한 스크립트를 사용합니다.

## SQL Server 를 사용하여 다운로드한 데이터 확인

먼저 데이터가 올바르게 로드되었는지 확인합니다.

1. SQL Server, Visual Studio 의 서버 탐색기, Visual Studio Code 와 같이 선호하는 데이터베이스 관리 도구를 사용해서 SQL Server 인스턴스에 연결합니다.
2. 새로 만든 데이터베이스를 선택하고 확장해서 새 데이터베이스, 테이블, 함수를 확인합니다.




3. 데이터가 올바르게 로드되었는지 확인하기위해 테이블을 마우스 오른쪽 클릭하고 **상위 1000 개 행 선택**을 **선택**합니다. 아래 쿼리가 실행됩니다:

```
SQL 복사
```

```
SELECT TOP 1000 * FROM [dbo].[nyctaxi_sample]
```

테이블의 데이터가 보이지 않는 경우 이전 항목의 문제 해결 절을 참조합니다.

4. 이 데이터 테이블은 columnstore 인덱스를 추가해서 세트 기반 계산용으로 최적화되었습니다. 아래 문을 실행해서 테이블에 대한 빠른 요약을 생성합니다.

SQL 복사

```
SELECT DISTINCT [passenger_count]
    , ROUND (SUM ([fare_amount]),0) as TotalFares
    , ROUND (AVG ([fare_amount]),0) as AvgFares
FROM [dbo].[nyctaxi_sample]
GROUP BY [passenger_count]
ORDER BY  AvgFares DESC
```

다음 단원에서는 R 을 사용하여 좀 더 복잡한 요약을 생성합니다.

## 다음 단원

R 을 사용한 데이터 요약

## 이전 단원

PowerShell 을 사용한 데이터 준비

# R 를 사용한 데이터 관찰 및 요약

SQL Server 계산 컨텍스트 정의

RxSqlServer 를 사용한 데이터 원본 생성

R 요약에서 Server 데이터 사용

다음 단원

이전 단원

이제 R 코드를 사용하여 동일한 데이터를 작업해보겠습니다. 이 단원에서는 **RevoScaleR** 패키지에 포함된 함수를 사용하는 방법을 배웁니다.

이번 연습에서 제공되는 R 스크립트에는 데이터 개체를 만들고 요약을 생성하고 모델을 작성하는데 필요한 모든 코드를 포함하고 있습니다. R 스크립트 파일 **RSQL_RWalkthrough.R** 은 스크립트 파일을 설치한 곳에서 찾을 수 있습니다.

- R 에 익숙하다면 스크립트를 한꺼번에 실행할 수 있습니다.
- RevoScaleR 사용법을 배우는 사람을 위해 이 자습서에서는 스크립트를 한 줄씩 살펴봅니다.
- 스크립트에서 개별 줄을 실행하려면 파일에서 한 줄 혹은 여러 줄을 강조 표시하고 Ctrl + ENTER 를 누릅니다.

팁

연습의 나머지 부분을 나중에 완료하려는 경우 R 작업 영역을 저장해 두세요. 그렇게 하면 데이터 개체와 다른 변수들을 다시 사용할 수 있습니다.

역주. 이전 연습 'TSQL 에서 R 코드 사용'을 하지 않고 바로 이 연습을 시작하는 경우 Lauchpad 서비스 시작 여부, 'external scripts enabled' 구성 옵션 설정 등에

따라 동작에 문제가 발생할 수 있으므로 사전에 필요 사항들을 확인하시길 바랍니다.

## SQL Server 계산 컨텍스트 정의

Microsoft R 을 사용하면 R 코드에서 SQL Server 데이터를 손쉽게 가져올 수 있습니다. 프로세스는 다음과 같습니다.

- SQL Server 인스턴스에 대한 연결 만들기
- 필요한 데이터를 가지는 쿼리를 정의하거나 테이블 또는 뷰를 지정하기
- R 코드를 실행할 때 사용할 하나 이상의 계산 컨텍스트 정의하기
- 선택적으로, 데이터를 읽는 동안 원본에 적용할 변환을 정의

다음 단계는 모두 R 코드의 일부이므로 R 환경에서 실행합니다. Microsoft R 클라이언트를 사용한 이유는 모든 RevoScaleR 패키지를 포함하고 있으면 기본적이고 가벼운 R 도구세트이기 때문입니다.

1. **RevoScaleR** 패키지가 로드 되지 않은 경우, 아래 R 코드를 실행합니다.

   R 복사

   ```
   library("RevoScaleR")
   ```

   인용 부호는 선택 사항이나 이 경우에는 권장됩니다.

   오류가 발생한다면 R 개발 환경이 RevoScaleR 패키지를 포함한 라이브러리를 사용하는지 확인합니다. 현재 라이브러리 경로를 보려면 `.libPaths()` 같은 명령을 사용합니다.

2. SQL Server 에 대한 연결 문자열을 만들고 R 변수 *connStr_*에 저장합니다.

R 복사

```
connStr <- "Driver=SQL
Server;Server=your_server_name;Database=Your_Database_Name;Uid=Your_
User_Name;Pwd=Your_Password"
```

서버 이름으로 인스턴스 이름만 사용할 수도 있고 네트워크에 따라 전체 식별자 이름이 필요할 수도 있습니다.

Windows 인증의 경우 구문이 조금 다릅니다:

R 복사

```
connStr <- "Driver=SQL
Server;Server=SQL_instance_name;Database=database_name;Trusted_Conne
ction=Yes"
```

다운로드 가능한 R 스크립트는 SQL 로그인만 사용합니다. 일반적으로 R 코드에 암호가 저장되는 것을 피하기위해 가능한 Windows 인증을 사용하길 권장합니다. 그러나 이 자습서의 코드가 Github 에서 다운로드한 코드와 일치하도록 나머지 연습에서는 SQL 로그인을 사용할 것입니다. (역주. 명명된 인스턴스에 연결하는 경우 구분 문자 \ 를 두 번 지정합니다.)

3. 새로운 *계산 컨텍스트*를 만들 때 사용할 변수를 정의합니다. 계산 컨텍스트 개체를 만든 후에 이를 사용해서 SQL Server 인스턴스에 R 코드를 실행할 수 있습니다.

R 복사

```
sqlShareDir <- paste("C:\\AllShare\\",Sys.getenv("USERNAME"),sep="")

sqlWait <- TRUE

sqlConsoleOutput <- FALSE
```

- R 은 워크스테이션과 SQL Server 컴퓨터 간에 R 개체를 직렬화할 때 임시 디렉터리를 사용합니다. *sqlShareDir* 로 사용되는 로컬 디렉터리를 지정하거나 기본값을 사용할 수 있습니다.
- *sqlWait* 을 사용하여 R 이 서버의 결과를 기다릴지 여부를 지정합니다. 대기 vs. 비대기 작업에 대한 논의는 Microsoft R 에서 ScaleR 을 사용한 분산과 병렬 컴퓨팅을 참조합니다.
- *sqlConsoleOutput* 인수를 사용해서 R 콘솔에 출력되지 않도록 지정합니다.

4. RxInSqlServer 생성자를 호출하여 이전에 정의된 변수와 연결 문자열로 계산 컨텍스트 개체를 만들고 R 변수 *sqlcc* 에 새로운 개체로 저장합니다.

```
R 복사
```

```
sqlcc <- RxInSqlServer(connectionString = connStr, shareDir =
sqlShareDir, wait = sqlWait, consoleOutput = sqlConsoleOutput)
```

5. 기본적으로 계산 컨텍스트는 로컬이므로 명시적으로 *활성* 계산 컨텍스트를 설정해야 합니다.

```
R 복사
```

```
rxSetComputeContext(sqlcc)
```

- `rxSetComputeContext` 는 이전의 활성 계산 컨텍스트를 사용할 수 있도록 보이지 않게 반환합니다.
- `rxGetComputeContext` 는 활성 계산 컨텍스트를 반환합니다.

계산 컨텍스트 설정은 **RevoScaleR** 패키지의 함수를 사용하는 연산에만 영향을 줍니다; 오픈 소스 R 연산이 수행되는 방식에는 영향을 미치지 않습니다.

## RxSqlServer 를 사용한 데이터 원본 생성

Microsoft R 에서 *데이터 소스*는 RevoScaleR 함수를 사용하여 만드는 개체입니다. 데이터 원본 개체는 모델 학습 또는 특성 추출과 같은 작업에 사용할 데이터 세트를 지정 합니다. 다양한 원본으로부터 데이터를 얻을 수 있습니다. 현재 지원되는 소스 목록은 RxDataSource 를 참조합니다.

이전에 우리는 연결 문자열을 정의하고 R 변수에 그 정보를 저장했습니다. 해당 연결 정보를 재사용해서 가져올 데이터를 지정할 수 있습니다.

1. SQL 쿼리를 문자열 변수로 저장합니다. 쿼리는 모델 학습용 데이터를 정의합니다.

R 복사

```
sampleDataQuery <- "SELECT 1000 tipped, fare_amount,
passenger_count,trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime, pickup_longitude, pickup_latitude,
dropoff_longitude, dropoff_latitude FROM nyctaxi_sample"
```

더 빠르게 실행하도록 TOP 절을 사용했었지만 그로 인해 쿼리에서 반환 되는 실제 행은 순서에 따라 다양할 수 있습니다. 따라서 아래에 나열된 요약 결과 또한 다른 수 있습니다. 편하게 TOP 절을 제거하세요 (역주. 원문에 TOP 절이 생략되어 있어 추가함.)

2. 쿼리 정의를 인수로 RxSqlServerData 함수에 전달합니다.

R 복사

```
inDataSource <- RxSqlServerData(
  sqlQuery = sampleDataQuery,
  connectionString = connStr,
  colClasses = c(pickup_longitude = "numeric", pickup_latitude =
"numeric",
  dropoff_longitude = "numeric", dropoff_latitude = "numeric"),
```

```
rowsPerRead=500
)
```

- *colClasses* 인수는 SQL Server 와 R 간에 데이터를 이동할 때 사용할 열 형식을 지정합니다. SQL Server 가 R 과는 다른 혹은 더 많은 데이터 형식을 사용하므로 이 중요한 부분입니다. 자세한 내용은 R 라이브러리와 데이터 형식을 참조합니다.
- *rowsPerRead* 인수는 메모리 사용량 관리와 효율적인 계산을 위해 중요한 부분입니다. R Services(In-Database) 의 대부분의 향상된 분석 함수들은 데이터를 청크(chunk)로 처리하고 중간 결과를 누적하면서 모든 데이터를 읽은 후에 최종 계산을 반환합니다. *rowsPerRead* 매개 변수를 추가하여 각 청크에서 처리할 데이터 행 수를 조절할 수 있습니다. 이 매개 변수의 값이 너무 크면 그만큼 큰 데이터 청크를 효율적으로 처리할 수 있는 메모리가 부족해서 데이터 접근 속도가 느려질 수 있습니다. 일부 시스템에서는 *rowsPerRead* 를 지나치게 작은 값으로 설정하는 것도 성능을 저하시킬 수 있습니다.

3. 이 시점에서 *inDataSource* 개체를 만들었지만 데이터는 포함되지 않았습니다. rxImport 또는 rxSummary 같은 함수를 실행하기 전까지 SQL 쿼리로부터 로컬 환경으로 데이터를 가져오지 않습니다. 하지만 이제 데이터 개체를 정의했으므로 다른 함수에 인수로 사용할 수 있습니다.

## R 요약에서 SQL Server 데이터 사용

이번 섹션에서 원격 계산 컨텍스트를 R Services(In-Database)의 몇 가지 함수를 시도해 보겠습니다. 데이터 소스에 R 함수를 적용하면 SQL Server 데이터를 탐색, 요약, 차트로 작성할 수 있습니다.

1. rxGetVarInfo 함수를 호출하면 데이터 원본에 있는 변수들의 목록과 해당 데이터 형식을 얻을 수 있습니다.

**rxGetVarInfo** 는 편리한 함수입니다; 데이터 프레임 또는 원격 데이터 개체 내의 데이터 세트에 대해 호출하면 최대값과 최소값, 그 데이터 유형, 인자(factor) 열이 몇 개의 수준(level)으로 되어 있는지 등의 정보를 얻을 수 있습니다.

데이터 입력, 특성 변환 또는 특성 추출(feature engineering)후에 이 함수를 실행하는 것을 고려하세요. 그렇게 하면 모델에서 사용하려는 모든 특성의 예상 데이터 유형을 확인하고 오류를 피할 수 있습니다.

R 복사

```
rxGetVarInfo(data = inDataSource)
```

**결과**

복사

```
Var 1: tipped, Type: integer
Var 2: fare_amount, Type: numeric
Var 3: passenger_count, Type: integer
Var 4: trip_time_in_secs, Type: numeric, Storage: int64
Var 5: trip_distance, Type: numeric
Var 6: pickup_datetime, Type: character
Var 7: dropoff_datetime, Type: character
Var 8: pickup_longitude, Type: numeric
Var 9: pickup_latitude, Type: numeric
Var 10: dropoff_longitude, Type: numeric
```

2. 이제 RevoScaleR 함수 rxSummary 를 호출해서 개별 변수에 대한 자세한 통계를 얻습니다.

rxSummary 는 R 요약 함수에 기반하지만 몇 가지 추가 기능과 이점을 가집니다. rxSummary 는 다중 계산 컨텍스트로 작동하고 청킹(chunking)를 지원합니다. 또한 값을 변환하거나 인자 수준 기반의 요약에도 rxSummary 를 사용할 수 있습니다.

이 예제에서는 승객 수에 따른 요금을 요약합니다.

R 복사

```
start.time <- proc.time()

rxSummary(~fare_amount:F(passenger_count,1,6), data = inDataSource)

used.time <- proc.time() - start.time

print(paste("It takes CPU Time=",
round(used.time[1]+used.time[2],2)," seconds,

  Elapsed Time=", round(used.time[3],2),

  " seconds to summarize the inDataSource.", sep=""))
```

- rxSummary 의 첫 번째 인수는 요약할 수식이나 용어를 지정합니다. 여기서 F() 함수는 요약 전에 *passenger_count* 값을 인자(factor)로 변환하는데 사용됩니다. 또한 *passenger_count* 인자 변수의 최소값(1)과 최대값(6) 또한 지정해야 합니다.
- 출력할 통계를 지정하지 않으면 기본적으로 rxSummary 가 Mean, StDev, Min, Max 그리고 유효 관측 수와 누락 관측 수를 출력합니다. (역주. 각각 ValidObs, MissingObs 라는 항목으로 출력됨)
- 이 예제에는 함수의 시작과 완료 시간을 추적해서 성능을 비교할 수 있는 코드도 포함하고 있습니다.

**결과**

rxSummary 함수가 성공적으로 실행되면 다음과 같은 범주별 통계 목록을 보여줍니다.

복사

```
rxSummary(formula = ~fare_amount:F(passenger_count, 1,6), data =
inDataSource)

Data: inDataSource (RxSqlServerData Data Source)

Number of valid observations: 1000
```

## 빅 데이터 보너스 연습

모든 행을 사용하는 새로운 쿼리 문자열을 정의하세요. 이 실험을 위해 새 데이터 원본 개체를 설정하는 것을 권장합니다. 처리량에 미치는 영향을 보기 위해 *rowsToRead* 매개변수를 바꾸어볼 수도 있습니다.

R 복사

```
bigDataQuery  <- "SELECT tipped, fare_amount,
passenger_count,trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime, pickup_longitude, pickup_latitude,
dropoff_longitude, dropoff_latitude FROM nyctaxi_sample"



bigDataSource <- RxSqlServerData(

     sqlQuery = bigDataQuery,

     connectionString = connStr,

     colClasses = c(pickup_longitude = "numeric", pickup_latitude =
"numeric",

     dropoff_longitude = "numeric", dropoff_latitude = "numeric"),

     rowsPerRead=500

     )



start.time <- proc.time()

rxSummary(~fare_amount:F(passenger_count,1,6), data = bigDataSource)

used.time <- proc.time() - start.time
```

```
print(paste("It takes CPU Time=",
round(used.time[1]+used.time[2],2)," seconds,

  Elapsed Time=", round(used.time[3],2),

  " seconds to summarize the inDataSource.", sep=""))
```

팁

코드를 실행하는 동안 Process Explorer 또는 SQL Profiler 같은 도구를 사용해서 SQL Server 서비스에 어떻게 연결되고 R code 가 실행되는지 확인할 수 있습니다.

SQL Server 에서 실행되는 R 작업을 모니터링하는 또 다른 옵션은 사용자 지정 보고서입니다.

## 다음 단원

SQL 및 R 을 사용하여 그래프와 플롯 만들기

## 이전 단원

SQL 을 사용한 데이터 탐색

# SQL 및 R을 사용한 그래프와 플롯(plot) 만들기(연습)

이 연습에서는 SQL Server 데이터와 R을 사용 하여 플롯과 맵을 생성하는 기술을 배웁니다. 연습을 위해 간단한 히스토그램을 만들어 본 뒤 더 복잡한 지도 플롯을 개발합니다.

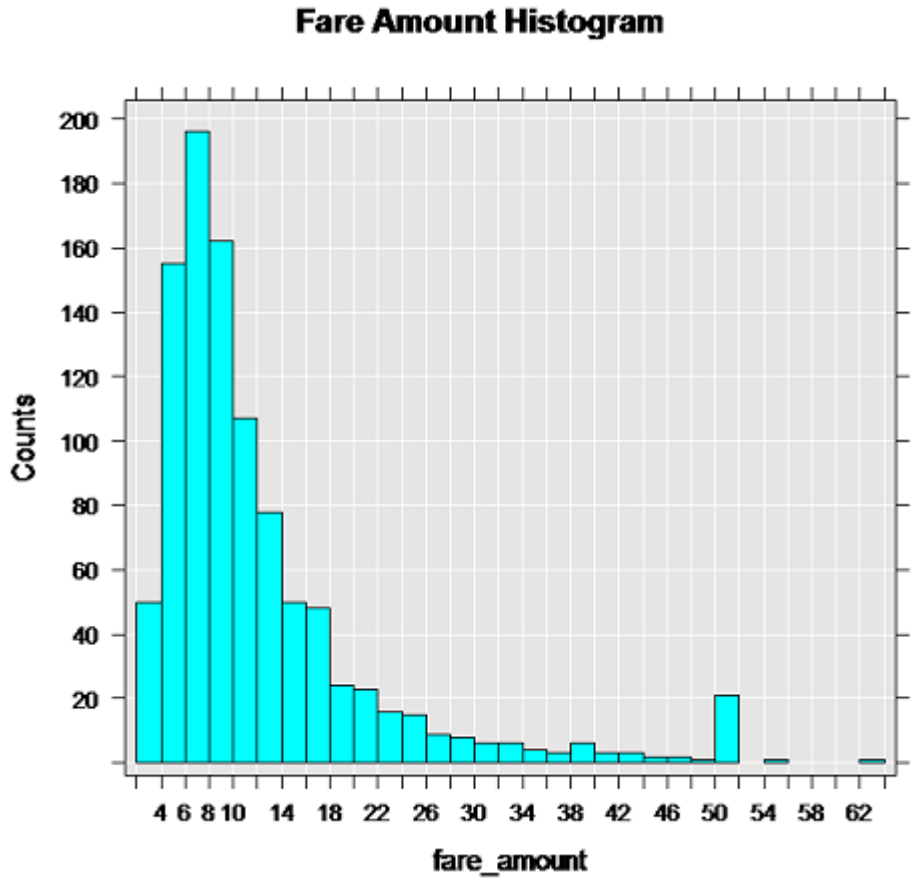
## 히스토그램 만들기

1. rxHistogram 함수를 사용하여 첫 번째 플롯을 생성합니다. RxHistogram 함수는 오픈 소스 R 패키지에 있는 것과 유사한 기능을 제공하지만 원격 실행 컨텍스트에서 실행할 수 있습니다.

R 복사

```
# Plot fare amount on SQL Server and return the plot

start.time <- proc.time()

rxHistogram(~fare_amount, data = inDataSource, title = "Fare Amount Histogram")

used.time <- proc.time() - start.time

print(paste("It takes CPU Time=",
round(used.time[1]+used.time[2],2), " seconds, Elapsed Time=",
round(used.time[3],2), " seconds to generate plot.", sep=""))
```

2. 이미지는 개발 환경을 위해 R 그래픽 장치로 반환됩니다. 예를 들어 RStudio 에서 Plot 창을 클릭합니다. R Tools for Visual Studio 에서는 별도 그래픽 창이 열립니다.




참고

그래프가 다르게 보입니까?

*inDataSource* 가 상위 1000 개 행만 사용하기 때문입니다. ORDER BY 절 없는 TOP 의 행 순서는 비 결정적이므로(역주. 정렬 순서가 보장되지 않는) 예상 데이터와 결과 그래프가 다를 수 있습니다. 이 이미지는 약 10,000 개의 데이터 행으로 생성되었습니다. 서로 다른 행 수를 사용해서 다른 그래프를 생성하고 그 결과를 반환하는데 걸리는 시간을 확인해 보기를 권장합니다.

## 지도 플롯 만들기

일반적으로 데이터베이스 서버는 인터넷 접근을 차단합니다. 이는 지도나 혹은 플롯을 생성하는 다른 이미지를 다운로드해야 하는 R 패키지를 사용할 때

불편할 수 있습니다. 그러나 자신의 응용 프로그램을 개발할 때 유용할 수도 있는 해결 방법이 있습니다. 기본적으로 클라이언트에서 지도 표현을 생성한 다음 SQL Server 테이블에 속성으로 저장된 점을 지도에 오버레이합니다.

1. R 플롯 개체를 만드는 함수를 정의합니다. 사용자 지정 함수 *mapPlot* 은 택시 승차 위치를 사용하는 산점도(scatter plot)를 생성하고 각 위치에서 시작한 탑승 수를 표시합니다. 이미 설치되고 로드된 **ggplot2** 와 **ggmap** 패키지를 사용합니다.

R 복사

```
mapPlot <- function(inDataSource, googMap){
    library(ggmap)
    library(mapproj)
    ds <- rxImport(inDataSource)
    p <- ggmap(googMap)+
    geom_point(aes(x = pickup_longitude, y =pickup_latitude ),
data=ds, alpha =.5,
color="darkred", size = 1.5)
    return(list(myplot=p))
}
```

- *mapPlot* 함수는 두 개의 인수를 가집니다: RxSqlServerData 를 사용하여 이전에 정의한 기존 데이터 개체, 그리고 클라이언트로 보낸 지도 표현입니다.
- *ds* 변수로 시작하는 줄에서 rxImport 는 이전에 만든 데이터 원본 *inDataSource* 의 데이터를 메모리로 로드하는데 사용됩니다. (데이터 원본은 1000 개 행만 포함합니다; 더 많은 데이터 포인트를 가진 지도를 만들고 싶은 경우 다른 데이터 원본을 대신 사용할 수 있습니다.)
- **오픈 소스** R 함수를 사용할 때마다 데이터는 로컬 메모리내 데이터 프레임으로 저장되어야 합니다. 그러나 rxImport 함수 호출을 통해서 원격 계산 컨텍스트의 메모리에서 실행할 수 있습니다.

2. 계산 컨텍스트를 로컬로 변경하고 지도를 만드는 데 필요한 라이브러리를 로드합니다.

R 복사

```
rxSetComputeContext("local")

library(ggmap)

library(mapproj)

gc <- geocode("Times Square", source = "google")

googMap <- get_googlemap(center = as.numeric(gc), zoom = 12, maptype = 'roadmap', color = 'color');
```

- `gc` 변수는 뉴욕 타임스퀘어의 좌표 세트를 저장합니다.
- `googmap`으로 시작하는 줄은 지정된 좌표가 중심인 지도를 생성합니다.

3. SQL Server 계산 컨텍스트로 전환하고 rxExec 를 plot 함수로 래핑하여 결과를 렌더링합니다. rxExec 함수는 **RevoScaleR** 패키지의 일부이며 원격 계산 컨텍스트에서 임의의 R 함수 실행을 지원합니다.

R 복사

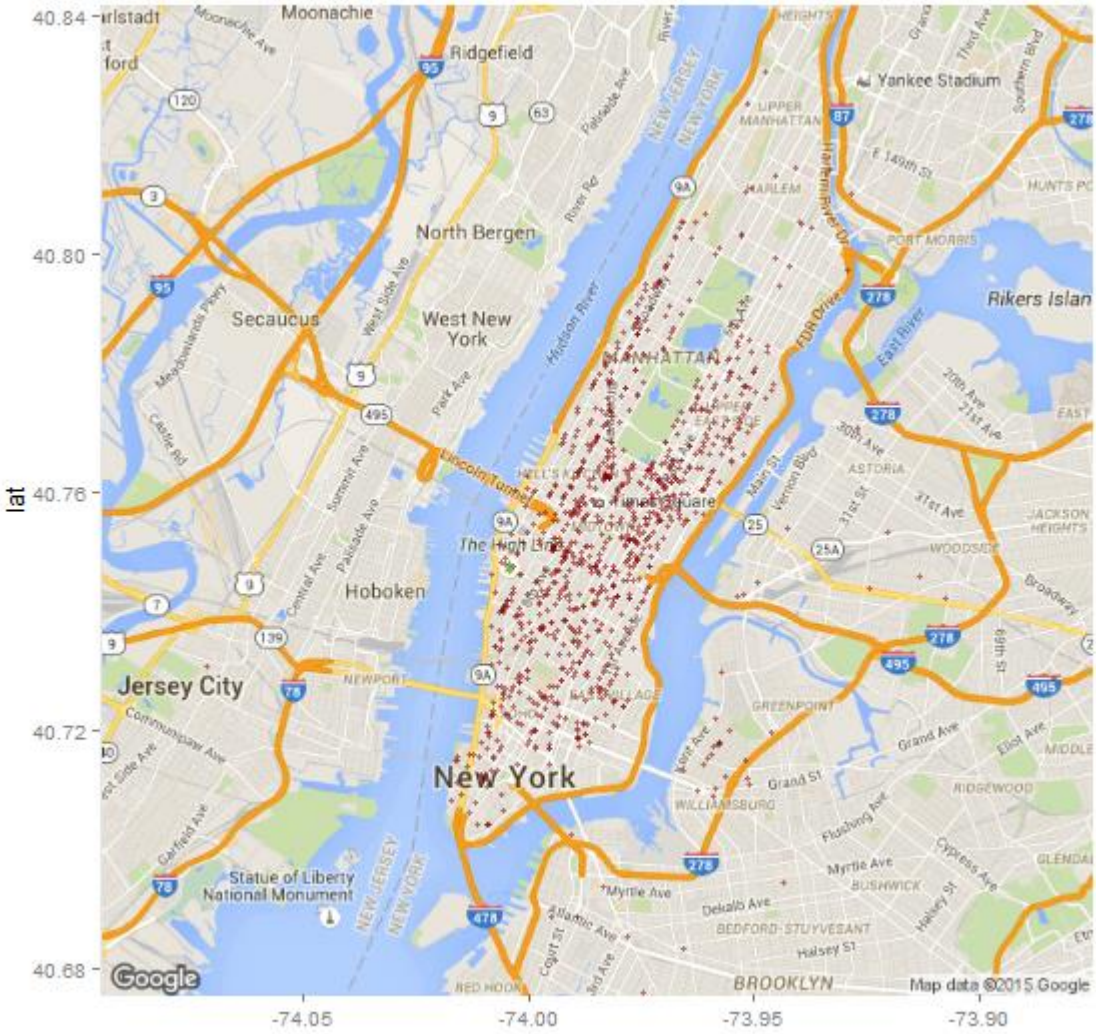
```
rxSetComputeContext(sqlcc)

myplots <- rxExec(mapPlot, inDataSource, googMap, timesToRun = 1)

plot(myplots[[1]][["myplot"]]);
```

- `googMap 의` 지도 데이터는 원격 실행 함수 *mapPlot* 에 인수로 전달됩니다. 지도가 로컬 환경에서 생성되었기 때문에 SQL Server 의 컨텍스트에서 플롯을 만들기 위해 함수로 전달되어야합니다.
- `plot`으로 시작하는 행이 실행되면 렌더링 데이터가 다시 로컬 R 환경으로 직렬화되어 R 클라이언트에서 볼 수 있게 됩니다.

참고

Azure 가상 머신에서 SQL Server 를 사용한다면 이 지점에서 오류가 발생할 수 있습니다. Azure 의 기본 방화벽 규칙이 R 코드에서의 네트워크 접근을 차단하기 때문입니다. 이 오류의 해결 방법은 Azure VM 에서 R Services 설치를 참조하세요.

4. 다음 그림은 출력 플롯을 보여줍니다. 택시 승차 위치가 지도에 빨간색 점으로 추가되었습니다. 사용한 데이터 원본의 위치 수에 따라 이미지가 다르게 보일 수 있습니다.



## 다음 단원

R 및 SQL 을 사용하여 데이터 특성 만들기

## 이전 단원

R 을 사용한 데이터 요약

# R 및 SQL 를 사용하여 데이터 특성 만들기(연습)

이 문서의 내용



데이터 엔지니어링은 머신 러닝(Machine Learning)의 중요한 부분입니다. 데이터는 가끔 예측 모델링에 사용하기 위해 사전이 변환이 필요하기도 합니다. 기존 데이터에서 필요한 특성이 없는 경우 기존 값에서 가공해낼 수도 있습니다.

이번 모델링 작업에서는 승차와 하차 위치에 대한 위도와 경도 값을 사용하는 대신 두 위치 간의 거리(마일)를 사용하는 것이 더 좋을 수 있습니다. 이 특성을 만들려면 haversine 공식을 사용하여 두 점 간의 직선 거리를 계산합니다.

이 단계에서는 데이터로부터 특성을 만들기 위한 두 가지 다른 방법을 비교합니다.

- 사용자 지정 R 함수 사용하기
- 사용자 지정 T-SQL 함수 사용하기

목표는 원본 열과 더불어 새로운 numeric 특성인 *direct_distance* 을 포함하는 새로운 SQL Server 데이터 세트를 만드는 것입니다.

## R 을 사용한 새 특성 추가(Featurization)

R 언어는 풍부하고 다양한 통계 라이브러리로 잘 알려져 있지만 여전히 사용자 정의 데이터 변환을 만들어야 할 수도 있습니다.

우선, R 사용자가 익숙한 방식으로 해 봅니다: 데이터를 랩톱에 가져온 다음 위도와 경로 값으로 지정된 두 점 간의 직선 거리를 계산하는 사용자 지정 R 함수 *ComputeDist* 를 실행합니다.

1. 이전에 만든 데이터 원본 개체는 상위 1000 개 행만 가져옵니다. 그래서 모든 데이터를 가져오는 쿼리를 정의해 보겠습니다.

R 복사

```
bigQuery <- "SELECT tipped, fare_amount,
passenger_count,trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime,  pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude FROM nyctaxi_sample";
```

2. 쿼리를 사용하여 새 SQL Server 데이터 원본을 만듭니다.

R 복사

```
featureDataSource <- RxSqlServerData(sqlQuery = bigQuery,colClasses
= c(pickup_longitude = "numeric", pickup_latitude = "numeric",
dropoff_longitude = "numeric", dropoff_latitude = "numeric",
passenger_count  = "numeric", trip_distance  = "numeric",
trip_time_in_secs  = "numeric", direct_distance  = "numeric"),
connectionString = connStr);
```

- RxSqlServerData 는 *sqlQuery* 매개변수의 인수로 제공되는 유효한 SELECT 쿼리 혹은 *table* 매개변수로 제공되는 테이블 개체 이름으로 구성할 수 있습니다.
- 테이블에서 데이터를 샘플링하고 싶은 경우 *sqlQuery* 매개변수를 사용하고, T-SQL TABLESAMPLE 절을 사용해서 샘플 매개변수를 정의하며, *rowBuffering* 인수를 FALSE 로 설정해야 합니다.

3. 다음 코드를 실행해서 사용자 지정 R 함수를 만듭니다. ComputeDist 는 두 쌍의 위도와 경도 값을 받아서 직선 거리를 계산하고 마일 단위 거리를 반환합니다.

R 복사

```
env <- new.env();

env$ComputeDist <- function(pickup_long, pickup_lat, dropoff_long,
dropoff_lat){

  R <- 6371/1.609344 #radius in mile

  delta_lat <- dropoff_lat - pickup_lat

  delta_long <- dropoff_long - pickup_long

  degrees_to_radians = pi/180.0

  a1 <- sin(delta_lat/2*degrees_to_radians)

  a2 <- as.numeric(a1)^2

  a3 <- cos(pickup_lat*degrees_to_radians)

  a4 <- cos(dropoff_lat*degrees_to_radians)

  a5 <- sin(delta_long/2*degrees_to_radians)

  a6 <- as.numeric(a5)^2

  a <- a2+a3*a4*a6
```

```
  c <- 2*atan2(sqrt(a),sqrt(1-a))

  d <- R*c

  return (d)

}
```

- 첫 번째 줄은 새 환경을 정의합니다. R 에서는 패키지와 같은 이름 공간을 캡슐화하는데 환경을 사용할 수 있습니다. `search()` 함수를 사용하여 작업 영역의 환경을 볼 수 있습니다. 특정 환경의 개체를 보려면 `ls(<envname>)`를 입력합니다.
- `$env.ComputeDistance` 로 시작하는 줄에는 haversine 공식을 정의하는 코드가 포함되어 있으며, 구(sphere)의 두 점간 *거리(great-circle distance)*를 계산합니다.

4. 함수를 정의한 후 새로운 특성 열인 *direct_distance* 를 생성하기 위해 함수를 데이터에 적용합니다. 변환을 실행하기전에 계산 컨텍스트를 로컬로 변경합니다.

R 복사

```
rxSetComputeContext("local");
```

5. rxDataStep 함수를 호출하여 특성 엔지니어링 데이터를 가져오고 `env$ComputeDist` 함수를 메모리의 데이터에 적용합니다.

R 복사

```
start.time <- proc.time();



changed_ds <- rxDataStep(inData = featureEngineeringQuery,

transforms =
list(direct_distance=ComputeDist(pickup_longitude,pickup_latitude,
dropoff_longitude, dropoff_latitude),

tipped = "tipped", fare_amount = "fare_amount", passenger_count =
"passenger_count",
```

```
trip_time_in_secs = "trip_time_in_secs",
trip_distance="trip_distance",

pickup_datetime = "pickup_datetime",  dropoff_datetime =
"dropoff_datetime"),

transformEnvir = env,

rowsPerRead=500,

reportProgress = 3);



used.time <- proc.time() - start.time;

print(paste("It takes CPU Time=",
round(used.time[1]+used.time[2],2)," seconds, Elapsed Time=",
round(used.time[3],2), " seconds to generate features.", sep=""));
```

- RxDataStep 함수는 데이터를 현재 위치에서 수정하는 다양한 방법을 지원합니다. 자세한 내용은 이 문서를 참조하세요: Microsft R 에서 데이터 변환 방법과 부분 집합

  그러나 rxDataStep 과 관련된 몇 가지 주목할만한 포인트가 있습니다:

  다른 데이터 원본에서는 *varsToKeep* 과 *varsToDrop* 인수를 사용할 수 있지만 SQL Server 데이터 원본에는 지원되지 않습니다. 따라서 이 예제에서는 *transform* 인수를 사용해서 통과(pass-through) 열과 변환 열 모두를 지정했습니다. 또한 SQL Server 계산 컨텍스트에서 실행할 때 *inData* 인수에는 SQL Server 데이터 원본만 사용할 수 있습니다.

  이전 코드는 큰 데이터 세트에서 실행할 때 경고 메시지를 생성할 수도 있습니다. 행과 열의 수가 설정된 값(기본 3,000,000)을 초과하는 경우 rsDataStep 은 경고를 반환하고 결과 데이터 프레임의 행 수가 잘립니다. 경고를 제거하려면 rxDataStep 함수의 *maxRowsByCols* 인수를 수정할 수 있습니다. 그러나 *maxRowsByCols* 가 너무 큰 경우 데이터 프레임을 메모리로 로드할 때 문제가 발생할 수 있습니다.

6. 선택적으로 변환된 데이터 원본의 스키마를 검사하기 위해 rxGetVarInfo 를 호출할 수 있습니다.

R 복사

```
rxGetVarInfo(data = changed_ds);
```

## Transact-SQL 을 사용한 새 특성 추가(featurzation)

이제 사용자 지정 R 함수와 동일한 작업을 수행할 SQL 함수 *ComputeDist* 를 만듭니다.

1. 새로운 사용자 지정 SQL 함수 *fnCalculateDistance* 를 정의합니다. 이 사용자 정의 SQL 함수에 대한 코드는 데이터베이스를 만들고 구성하기 위해 실행하는 PowerShell 스크립트의 일부로 제공됩니다. 함수는 데이터베이스에 미리 생성해야 합니다.

   존재하지 않으면 SQL Server Management Studio 를 사용하여 택시 데이터가 저장된 동일한 데이터베이스에 함수를 생성합니다. (역주. 아래 코드에서 RETURNS 문의 반환 데이터 형식이 생략되어 추가로 지정했습니다)

SQL 복사

```
CREATE FUNCTION [dbo].[fnCalculateDistance] (@Lat1 float, @Long1
float, @Lat2 float, @Long2 float)

-- User-defined function calculates the direct distance between two
geographical coordinates.

RETURNS decimal(28, 10)

AS

BEGIN

  DECLARE @distance decimal(28, 10)
```

```
  -- Convert to radians

  SET @Lat1 = @Lat1 / 57.2958

  SET @Long1 = @Long1 / 57.2958

  SET @Lat2 = @Lat2 / 57.2958

  SET @Long2 = @Long2 / 57.2958

  -- Calculate distance

  SET @distance = (SIN(@Lat1) * SIN(@Lat2)) + (COS(@Lat1) *
COS(@Lat2) * COS(@Long2 - @Long1))

  --Convert to miles

  IF @distance <> 0

  BEGIN

    SET @distance = 3958.75 * ATAN(SQRT(1 - POWER(@distance, 2)) /
@distance);

  END

  RETURN @distance

END
```

2. 함수가 제대로 작동하는지 보기 위해 Transact-SQL 을 지원하는 응용 프로그램에서 다음 Transact-SQL 문을 실행합니다. (역주. 함수 동작 검증이 목적이므로 TOP 절을 추가해서 일부 데이터만 시험하길 권장합니다)

SQL 복사

```
SELECT tipped, fare_amount,
passenger_count,trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime,

dbo.fnCalculateDistance(pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude) as direct_distance,

pickup_latitude, pickup_longitude,  dropoff_latitude,
dropoff_longitude

FROM nyctaxi_sample
```

3. 함수를 정의하고 나면 SQL 을 사용해서 원하는 특성을 만들고 그 값을 새로운 테이블에 직접 입력하기가 쉬워집니다.

복사

```
SELECT tipped, fare_amount,
passenger_count,trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime,

dbo.fnCalculateDistance(pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude) as direct_distance,

pickup_latitude, pickup_longitude,  dropoff_latitude,
dropoff_longitude

INTO NewFeatureTable

FROM nyctaxi_sample
```

4. 그렇지만 R 코드에서 사용자 지정 SQL 함수를 호출하는 방법을 살펴보겠습니다. 먼저 R 변수에 SQL 특성 추가 쿼리를 저장합니다.

R 복사

```
featureEngineeringQuery = "SELECT tipped, fare_amount,
passenger_count,

    trip_time_in_secs,trip_distance, pickup_datetime,
dropoff_datetime,

    dbo.fnCalculateDistance(pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude) as direct_distance,

    pickup_latitude, pickup_longitude,  dropoff_latitude,
dropoff_longitude

    FROM nyctaxi_sample

    tablesample (1 percent) repeatable (98052)"
```

팁

위 쿼리는 연습을 더 빨리하기 위해 더 적은 샘플 데이터를 얻도록 수정되었습니다. 모든 데이터를 가져오려면 TABLESAMPLE 절을 제거할 수 있습니다. 그러나 환경에 따라서는 전체 데이터를 R 에 로딩할 수 없어서 오류가 발생할 수 있습니다.

5. 다음 코드 줄을 사용하여 R 환경에서 Transact-SQL 함수를 호출하고 이를 *featureEngineeringQuery* 에 정의된 데이터에 적용합니다.

R 복사

```
featureDataSource = RxSqlServerData(sqlQuery =
featureEngineeringQuery,

  colClasses = c(pickup_longitude = "numeric", pickup_latitude =
"numeric",

         dropoff_longitude = "numeric", dropoff_latitude = "numeric",

         passenger_count  = "numeric", trip_distance  = "numeric",

          trip_time_in_secs  = "numeric", direct_distance  =
"numeric"),

  connectionString = connStr)
```

6. 이제 새 특성이 만들어졌으므로 **rxGetVarsInfo** 를 호출하여 특성 테이블에 데이터 요약을 만듭니다.

R 복사

```
rxGetVarInfo(data = featureDataSource)
```

*결과*

복사

```
Var 1: tipped, Type: integer

Var 2: fare_amount, Type: numeric
```

```
Var 3: passenger_count, Type: numeric

Var 4: trip_time_in_secs, Type: numeric

Var 5: trip_distance, Type: numeric

Var 6: pickup_datetime, Type: character

Var 7: dropoff_datetime, Type: character

Var 8: direct_distance, Type: numeric

Var 9: pickup_latitude, Type: numeric

Var 10: pickup_longitude, Type: numeric

Var 11: dropoff_latitude, Type: numeric

Var 12: dropoff_longitude, Type: numeric
```

참고

일부의 경우 다음과 같은 오류가 발생할 수 있습니다: *'fnCalculateDistance' 개체에 대해 EXECUTE 권한이 거부 되었습니다* 만일 그렇다면 사용하는 로그인이 스크립트를 실행하고 데이터베이스 개체를 만드는 권한이 있는지 확인하세요. FnCalculateDistance 개체에 대한 스키마를 확인합니다. 만일 개체가 데이터베이스 소유자에 의해 생성되었고, 로그인이 db_datareader 역할에 속한다면 스크립트를 실행하기 위해서 명시적으로 사용 권한을 부여해야 합니다.

# R 함수와 SQL 함수 비교

R 코드의 시간을 재는데 사용된 코드 조각을 기억하십니까?

R 복사

```
start.time <- proc.time()

<your code here>

used.time <- proc.time() - start.time
```

```
print(paste("It takes CPU Time=",
round(used.time[1]+used.time[2],2)," seconds, Elapsed Time=",
round(used.time[3],2), " seconds to generate features.", sep=""))
```

이것을 SQL 사용자 지정 함수 예제와 함께 사용해서 SQL 함수 호출 시 데이터 변환 작업이 얼마나 오래 걸리는지 확인할 수 있습니다. 또한 rxSetComputeContext 로 계산 컨텍스트를 전환하고 타이밍을 비교해보세요.

네트워크 속도와 하드웨어 구성 등에 따라서 시간이 크게 차이날 수 있습니다. 우리가 시험한 구성에서는 Transact-SQL 함수 방식이 사용자 지정 R 함수보다 더 빨랐음으로 이후 단계에서는 Transact-SQL 함수를 사용합니다.

팁

Transact-SQL 을 사용한 특성 엔지니어링이 R 의 경우보다 더 빠를 것입니다. 예를 들어 T-SQL 에는 이동 평균과 *n-tile* 같은 공통적인 데이터 과학 계산에 적용할 수 있는 빠른 윈도우 및 순위 함수를 포함하고 있습니다. 실제 데이터와 작업에 기반해서 가장 효율적인 방법을 선택하세요.

## 다음 단원

R 모델을 작성하고 SQL Server 에 저장하기

## 이전 단원

SQL 및 R 를 사용한 그래프와 플롯 만들기

# R 모델을 작성하고 SQL Server 에 저장하기



이 단계에서는 머신 러닝 모델을 작성하고 SQL Server 에 그 모델을 저장하는 방법을 학습합니다.

## RxLogit 을 사용하여 분류 모델 만들기

작성하는 모델은 택시 운전사가 특정 승차에 대해 팁을 받을 수 있는지를 예측하는 이진 분류기입니다. 이전 단원에서 만든 데이터 원본을 사용하여 팁 분류기를 학습하며 로지스틱 회귀를 사용합니다.

1. RevoScaleR 패키지에 포함된 **rxLogit** 함수를 호출하여 로지스틱 회귀 모델을 만듭니다.

R 복사

```
system.time(logitObj <- rxLogit(tipped ~ passenger_count +
trip_distance + trip_time_in_secs + direct_distance, data =
sql_feature_ds));
```

모델을 작성하는 호출은 system.time 함수로 묶여 있습니다. 이것으로 모델을 작성하는 데 필요한 시간을 구할 수 있습니다.

2. 모델을 작성한 후에 `summary` 함수를 사용하여 모델을 조사하고 계수를 관찰할 수 있습니다.

R 복사

```
summary(logitObj);
```

*결과*

*Logistic Regression Results for: tipped ~ passenger_count + trip_distance +*
*trip_time_in_secs +*
*direct_distance*
*Data: featureDataSource (RxSqlServerData Data Source)*
*Dependent variable(s): tipped*
*Total independent variables: 5*
*Number of valid observations: 17068*
*Number of missing observations: 0*
*-2*LogLikelihood: 23540.0602 (Residual deviance on 17063 degrees of freedom)*
*Coefficients:*
*Estimate Std. Error z value Pr(>|z|)*
*(Intercept) -2.509e-03 3.223e-02 -0.078 0.93793*
*passenger_count -5.753e-02 1.088e-02 -5.289 1.23e-07 ****
*trip_distance -3.896e-02 1.466e-02 -2.658 0.00786 ***
*trip_time_in_secs 2.115e-04 4.336e-05 4.878 1.07e-06 ****
*direct_distance 6.156e-02 2.076e-02 2.966 0.00302 ***
---
*Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1*
*Condition number of final variance-covariance matrix: 48.3933*
*Number of iterations: 4*

## 로지스틱 회귀 모델을 사용하여 채점하기

이제 모델이 작성되었으므로 운전사가 특정 드라이브에서 팁을 받을 가능성이 있는지 여부를 예측할 수 있습니다.

1. 우선 RxSqlServerData 함수를 사용해서 채점 결과를 저장하기 위한 데이터 원본 개체를 정의합니다. (역주. 원본에 “scoring resul” 이라는 단어로 끝나는 문장을 가지고 있으며 이를 역자는 “scoring result” 라고 판단해 해석에 적용했습니다.)

R 복사

```
scoredOutput <- RxSqlServerData(
  connectionString = connStr,
  table = "taxiScoreOutput"  )
```

   - 예제를 더 간단하게 만들기 위해 로지스틱 회귀 모델의 입력은 모델 학습에 사용했던 같은 특성 데이터 원본(`sql_feature_ds`)입니다. 보다 일반적으로는 채점을 위해 새로운 데이터를 가지고 있거나 혹은 검증용(test)과 학습용(training)으로 일부 데이터를 따로 준비했을 것입니다.
   - 예측 결과는 *taxiscoreOutput* 테이블에 저장 됩니다. 이 테이블에 스키마는 rxSqlServerData 를 사용할 때 정의되지 않습니다. rxPredict 출력에서 스키마를 가져옵니다.
   - 예측된 값을 저장하는 테이블을 만들려면 rxSqlServer 데이터 함수를 실행하는 SQL 로그인에 데이터베이스의 DDL 권한이 있어야 합니다. 로그인이 테이블을 만들 수 없는 경우 문이 실패합니다.
2. rxPredict 함수를 호출하여 결과를 생성합니다.

R 복사

```
rxPredict(modelObject = logitObj,
    data = sql_feature_ds,
    outData = scoredOutput,
```

```
    predVarNames = "Score",
    type = "response",
    writeModelVars = TRUE, overwrite = TRUE)
```

문이 성공하면 실행하는데 약간의 시간이 걸립니다. 완료되면 SQL Server Management Studio 을 열어서 해당 테이블의 생성 여부 그리고 Score 열과 기타 예상되는 결과가 포함되어 있는지 확인합니다.

## 모델 정확도 그리기

모델의 정확도에 대한 아이디어를 얻으려면 Receiver Operating Curve 를 그리기 위해 rxRoc 함수를 사용할 수 있습니다. rxRoc 는 원격 계산 컨텍스트를 지원하는 RevoScaleR 패키지가 제공하는 새로운 기능 중 하나이므로 원격 다음 두 가지 옵션이 있습니다.

- rxRoc 함수를 사용하여 원격 계산 컨텍스트에서 플롯을 실행한 다음 로컬 클라이언트에 플롯을 반환합니다.
- 데이터를 R 클라이언트 컴퓨터로 가져온 후 다른 R 플로팅 함수를 사용해서 성능 그래프를 만듭니다.

이번 절에서는 두 가지 방법을 모두 시험합니다.

### 원격(SQL Server) 계산 컨텍스트에서 플롯 실행

1. rxRoc 함수를 호출하고 앞에서 정의한 데이터를 입력으로 제공합니다.

R 복사

```
scoredOutput = rxImport(scoredOutput);

rxRoc(actualVarName= "tipped", predVarNames = "Score",
scoredOutput);
```

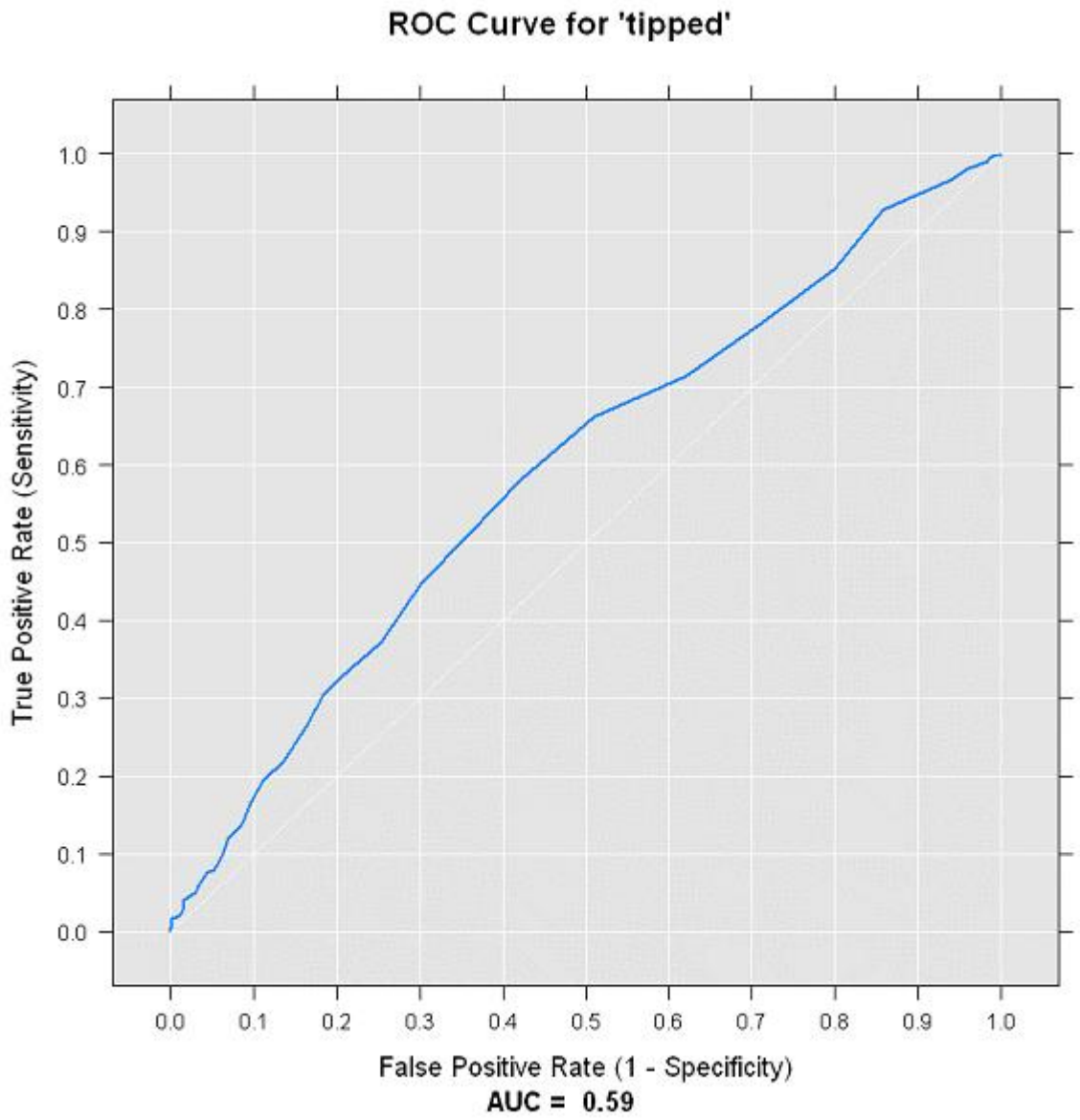
이 호출은 ROC 차트를 계산할 때 사용되는 값을 반환합니다. 라벨(Label) 열은 *tipped* 이고 예측하려는 실제 결과를 가지고 있으며 *Score* 열에는 예측이 들어 있습니다.

2. 실제로 차트를 그리기 위해서는 ROC 개체를 저장한 다음에 `plot` 함수로 그릴 수 있습니다. 그래프는 원격 계산 컨텍스트에 생성되고 R 환경에 반환됩니다.

R 복사

```
scoredOutput = rxImport(scoredOutput);

rocObjectOut <- rxRoc(actualVarName= "tipped", predVarNames =
"Score", scoredOutput);

plot(rocObjectOut);
```

R 그래픽 장치를 열거나 RStudio 의 **Plot** 창을 클릭해서 그래프를 봅니다.




**SQL Server** 의 데이터를 사용하여 로컬 계산 컨텍스트에서 플롯 만들기

1. 로컬 계산 컨텍스트에 대해서도 절차는 거의 동일합니다. rxImport 함수를 사용해서 로컬 R 환경에 지정된 데이터를 가져옵니다.

```
R 복사
```

```
scoredOutput = rxImport(scoredOutput)
```

2. 로컬 메모리 내 데이터를 사용해서 **ROCR** 패키지를 로드하고 예측 함수를 사용해서 몇가지 새로운 예측을 만듭니다.

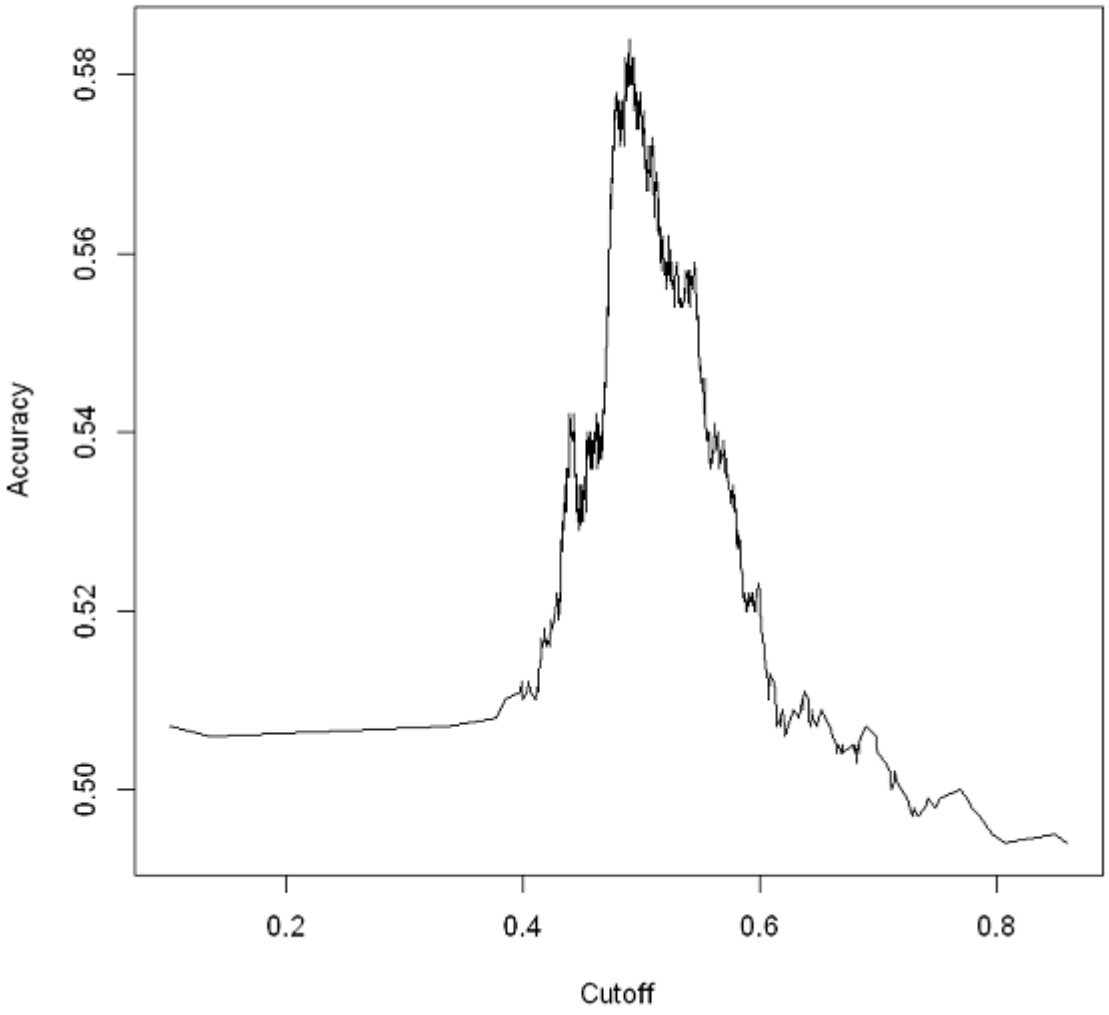
R 복사

```
library('ROCR');

pred <- prediction(scoredOutput$Score, scoredOutput$tipped);
```

3. 출력 변수 `pred`에 저장된 값을 기반으로 로컬 플롯을 생성합니다.

R 복사

```
acc.perf = performance(pred, measure = 'acc');

plot(acc.perf);

ind = which.max( slot(acc.perf, 'y.values')[[1]] );

acc = slot(acc.perf, 'y.values')[[1]][ind];

cutoff = slot(acc.perf, 'x.values')[[1]][ind];
```




참고

차트는 사용한 데이터 포인트의 수에 따라 다르게 보일 수 있습니다.

## 모델 배포

모델을 작성하고 잘 수행되는 것을 확인한 후 그 모델을 사용하거나 혹은 정기적으로 재교육과 재조정하는 조직의 사용자나 사람들이 있는 사이트로 배포하길 원할 것입니다. 이러한 절차를 때때로 *operationalizing a model* 이라고 합니다.

R Services(In-Database) 사용하면 Transact-SQL 저장 프로시저를 사용해서 R 모델을 호출할 수 있으므로 클라이언트 응용 프로그램에서 R 을 쉽게 사용할 수 있습니다.

그러나 외부 응용 프로그램에서 모델을 호출하려면 프로덕션에서 사용되는 데이터베이스에 모델을 저장해야 합니다. R Services(In-Database)에서는 학습된 모델이 **varbinary(max)**형식의 단일 열에 이진 형식으로 저장됩니다.

따라서 학습된 모델을 R 에서 SQL Server 로 이동하는 단계는 다음과 같습니다:

- 모델을 16 진수 문자열로 직렬화
- 직렬화된 개체를 데이터베이스로 전송
- varbinary(max) 열에 모델 저장

이번 절에서는 모델을 유지하는 방법과 예측을 수행하기 위해 모델을 호출하는 방법을 배웁니다.

1. 로컬 R 환경으로 다시 전환하고(사용하지 않고 있다면) 모델을 직렬화하고 변수에 저장합니다.

R 복사

```
rxSetComputeContext("local");
modelbin <- serialize(logitObj, NULL);
modelbinstr=paste(modelbin, collapse="");
```

2. **RODBC 를** 사용하여 ODBC 연결을 엽니다.

R 복사

```
library(RODBC);
conn <- odbcDriverConnect(connStr);
```

패키지를 이미 로드한 경우 RODBC 호출을 생략할 수 있습니다.

3. PowerShell 스크립트로 만든 저장된 프로시저를 호출하여 모델의 이진 표현을 데이터베이스의 열에 저장합니다.

R 복사

```
q <- paste("EXEC PersistModel @m='", modelbinstr,"'", sep="");
sqlQuery (conn, q);
```

테이블에 모델을 저장하는데 INSERT 문만 있으면 되지만 *PersistModel* 처럼 저장 프로시저로 구성하는 것이 더 용이합니다.

참고

만일 " PersistModel 개체에 EXECUTE 권한이 거부되었습니다" 같은 오류가 발생하는 경우 로그인에 권한이 있는지 확인하세요. `GRANT EXECUTE ON [dbo].[PersistModel] TO <user_name>` 같은 T-SQL 문을 실행하여 저장 프로시저에 명시적으로 권한을 줄 수 있습니다.

4. 모델을 만들고 데이터베이스에 저장한 후에는 sp_execute_external_script 시스템 저장 프로시저를 사용해서 Transact-SQL 코드로부터 모델을 직접 호출할 수 있습니다.

그런데 자주 사용하는 모델이라면 사용자 저장 프로시저 내에서 입력 쿼리와 모델 호출을 다른 매개변수와 함께 구성하는 것 더 용이합니다.

다음은 그러한 저장 프로시저의 전체 코드입니다. SQL Server 에서 R 모델을 보다 쉽게 관리하고 업데이트할 수 있도록 이와 같은 저장된 프로시저를 만드는 것이 좋습니다.

tsql 복사

```
CREATE PROCEDURE [dbo].[PersistModel]  @m nvarchar(max)

AS

BEGIN

  SET NOCOUNT ON;

  INSERT INTO nyc_taxi_models (model) values
(convert(varbinary(max),@m,2))

END
```

참고

**SET NOCOUNT ON** 절을 사용하여 SELECT 문의 추가 결과 집합을 방지합니다. (역주. SELECT 결과 외 추가로 반환되는 메시지를 제거하는 목적임)

다음 및 최종 단원에서 Transact-SQL 을 사용하여 저장된 모델을 대상으로 채점을 수행하는 방법을 배웁니다.

## 다음 단원

R 모델을 배포하고 SQL 에서 사용하기

## 이전 단원

R 및 SQL 을 사용하여 데이터 특성 만들기

# R 모델을 배포하고 SQL 에서 사용하기

이 문서의 내용



이 단원에서는 저장된 프로시저에서 학습된 모델을 호출하여 프로덕션 환경에서 R 모델을 사용합니다. 그런 다음 R 또는 Transact-SQL 을 지원하는 응용 프로그램 언어(C#, Java, Python, 등)에서 저장 프로시저를 호출하고 해당 모델을 사용해서 새 관찰에 대한 예측을 만듭니다.

이 샘플은 채점에서 모델을 사용하는 가장 일반적인 방법 두 가지를 보여줍니다.

- **일괄 처리 채점 모드** SQL 쿼리나 테이블을 입력으로 전달해서 여러 예측을 매우 빠르게 만들 필요가 있을 때 사용됩니다. 결과 테이블이 반환되면 다른 테이블에 직접 입력하거나 파일에 쓸 수 있습니다.
- **개별 채점 모드** 한 번에 하나씩 예측을 만들 때 사용 됩니다. 개별 값 집합을 저장 프로시저에 전달합니다. 그 값들은 모델에서 예측을 만들기 위해 사용하는 특성이나 또 다른 결과를 생성하기 위한 확률 값 같은 것에 해당합니다. 그런 다음 그 값을 응용 프로그램이나 사용자에게 반환할 수 있습니다.

## 일괄 처리 채점

초기 PowerShell 스크립트를 실행했을 때 일괄 처리 채점용 저장 프로시저 *PredictTipBatchMode* 가 생성되었으며 다음을 수행합니다.

- 입력 데이터 세트를 SQL 쿼리로 가져오기
- 이전 단원에서 저장한 학습된 로지스틱 회귀 모델 호출
- 운전자가 팁을 받을 확률 예측.

1. 잠시 시간을 내어 저장 프로시저 *PredictTipBatchMode* 를 살펴보세요. R Services(In-Database)을 사용하여 모델을 운영하는 방법에 관한 여러 측면을 보여 줍니다.

tsql 복사

```
CREATE PROCEDURE [dbo].[PredictTipBatchMode]
@input nvarchar(max)
AS
BEGIN
  DECLARE @lmodel2 varbinary(max) = (SELECT TOP 1 model  FROM
nyc_taxi_models);
  EXEC sp_execute_external_script @language = N'R',
    @script = N'
      mod <- unserialize(as.raw(model));
      print(summary(mod))
      OutputDataSet<-rxPredict(modelObject = mod,
        data = InputDataSet,
        outData = NULL,
        predVarNames = "Score", type = "response",
        writeModelVars = FALSE, overwrite = TRUE);
      str(OutputDataSet)
```

```
      print(OutputDataSet)',
  @input_data_1 = @input,
  @params = N'@model varbinary(max)',
  @model = @lmodel2
  WITH RESULT SETS ((Score float));
END
```

- SELECT 문을 사용하여 SQL 테이블에 저장된 모델을 호출합니다. 모델은 **varbinary (max)** 데이터로 SQL 변수 *@lmodel2* 에 저장되며 시스템 저장 프로시저 sp_execute_external_script 에 매개변수 *@model* 에 전달됩니다.
- 채점을 위한 입력용 데이터는 SQL 쿼리로 정의되고 SQL 변수 *@input* 에 문자열로 저장됩니다. 데이터가 데이터베이스로부터 검색되면 *InputDataSet* 이라는 데이터 프레임에 저장됩니다. 이 데이터 프레임은 sp_execute_external_script 프로시저에 입력 데이터용 기본 이름입니다. 다른 변수 이름이 필요하다면 *@input_data_1_name* 매개 변수를 사용할 수 있습니다.
- 채점을 위해 저장 프로시저에서 RevoScaleR 라이브러리에 `rxPredict` 함수를 호출합니다.
- 반환 값 *Score* 는 주어진 모델에서 운전자가 팁을 얻는 확률입니다. 선택 사항으로 반환 값을 “팁”과 “팁 없음” 그룹으로 분류하도록 일종의 필터를 쉽게 적용할 수 있습니다. 예를 들어 0.5 미만의 확률은 팁이 거의 없음을 의미합니다.

2. 일괄 처리 모드로 저장 프로시저를 호출하기 위해 저장 프로시저 입력으로 필요한 쿼리를 아래와 같이 정의합니다. 동작 여부를 확인하기 위해 SSMS 에서 실행할 수 있습니다.

SQL 복사

```
SELECT TOP 10
  a.passenger_count AS passenger_count,
  a.trip_time_in_secs AS trip_time_in_secs,
```

```
  a.trip_distance AS trip_distance,

  a.dropoff_datetime AS dropoff_datetime,

  dbo.fnCalculateDistance( pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude) AS direct_distance

  FROM

    (SELECT medallion, hack_license, pickup_datetime,
passenger_count,trip_time_in_secs,trip_distance, dropoff_datetime,
pickup_latitude, pickup_longitude, dropoff_latitude,
dropoff_longitude

    FROM nyctaxi_sample)a

  LEFT OUTER JOIN

  ( SELECT medallion, hack_license, pickup_datetime

  FROM nyctaxi_sample  tablesample (1 percent) repeatable
(98052)  )b

  ON a.medallion=b.medallion

  AND a.hack_license=b.hack_license

  AND a.pickup_datetime=b.pickup_datetime

  WHERE b.medallion is null
```

3. SQL 쿼리로 프로시저의 입력 문자열을 만들기 위해 다음 R 코드를 사용합니다.

R 복사

```
input <- "N'SELECT TOP 10 a.passenger_count AS passenger_count,
a.trip_time_in_secs AS trip_time_in_secs, a.trip_distance AS
trip_distance, a.dropoff_datetime AS dropoff_datetime,
dbo.fnCalculateDistance(pickup_latitude, pickup_longitude,
dropoff_latitude, dropoff_longitude) AS direct_distance FROM (SELECT
medallion, hack_license, pickup_datetime,
passenger_count,trip_time_in_secs,trip_distance, dropoff_datetime,
pickup_latitude, pickup_longitude, dropoff_latitude,
dropoff_longitude FROM nyctaxi_sample)a LEFT OUTER JOIN ( SELECT
medallion, hack_license, pickup_datetime FROM nyctaxi_sample
tablesample (1 percent) repeatable (98052)  )b ON
a.medallion=b.medallion AND a.hack_license=b.hack_license AND
a.pickup_datetime=b.pickup_datetime WHERE b.medallion is null'";
```

```
q <- paste("EXEC PredictTipBatchMode @inquery = ", input, sep="");
```

4. R 에서 저장 프로시저를 실행하기 위해 **RODBC** 패키지의 **sqlQuery** 메서드를 호출하며 이전의 정의한 SQL 연결 `conn` 을 사용합니다:

R 복사

```
sqlQuery (conn, q);
```

ODBC 오류가 나면 쿼리 구문에서 인용 부호 개수가 맞는지 확인합니다.

사용 권한 오류가 난다면 해당 로그인이 저장 프로시저를 실행할 수 있는 권한이 있는지 확인합니다.

## 단일 행 채점

행별로 예측 모델을 호출할 때 각 개별 사례의 특성을 나타내는 값 세트를 전달합니다. 그런 다음 저장 프로시저에서 단일 예측이나 확률을 반환합니다.

저장 프로시저 *PredictTipSingleMode* 가 이러한 접근 방법을 보여줍니다. 여러 개의 입력 매개변수로 특성 값(예: 승객 수와 이동 거리)을 받아서 저장된 R 모델을 사용하여 그러한 특성을 채점하고 팁 확률을 반환합니다.

1. 초기 PowerShell 스크립트에서 저장 프로시저 *PredictTipSingleMode* 가 생성되지 않은 경우 다음 Transact-SQL 문을 실행해서 지금 만들 수 있습니다.

tsql 복사

```
CREATE PROCEDURE [dbo].[PredictTipSingleMode] @passenger_count int =
0,

@trip_distance float = 0,

@trip_time_in_secs int = 0,
```

```
@pickup_latitude float = 0,

@pickup_longitude float = 0,

@dropoff_latitude float = 0,

@dropoff_longitude float = 0

AS

BEGIN

  DECLARE @inquery nvarchar(max) = N'

    SELECT * FROM [dbo].[fnEngineerFeatures](@passenger_count,
@trip_distance, @trip_time_in_secs, @pickup_latitude,
@pickup_longitude, @dropoff_latitude, @dropoff_longitude)'

  DECLARE @lmodel2 varbinary(max) = (SELECT TOP 1 model FROM
nyc_taxi_models);



  EXEC sp_execute_external_script @language = N'R',  @script = N'

        mod <- unserialize(as.raw(model));

        print(summary(mod))

        OutputDataSet<-rxPredict(

          modelObject = mod,

          data = InputDataSet,

          outData = NULL,

          predVarNames = "Score",

          type = "response",

          writeModelVars = FALSE,

          overwrite = TRUE);

        str(OutputDataSet)

        print(OutputDataSet)

        ',
```

```
  @input_data_1 = @inquery,
  @params = N'
  -- passthrough columns
  @model varbinary(max) ,
  @passenger_count int ,
  @trip_distance float ,
  @trip_time_in_secs int ,
  @pickup_latitude float ,
  @pickup_longitude float ,
  @dropoff_latitude float ,
  @dropoff_longitude float',
  -- mapped variables
  @model = @lmodel2 ,
  @passenger_count =@passenger_count ,
  @trip_distance=@trip_distance ,
  @trip_time_in_secs=@trip_time_in_secs ,
  @pickup_latitude=@pickup_latitude ,
  @pickup_longitude=@pickup_longitude ,
  @dropoff_latitude=@dropoff_latitude ,
  @dropoff_longitude=@dropoff_longitude
  WITH RESULT SETS ((Score float));
END
```

2. SQL Server Management Studio 에서 Transact-SQL **EXEC** 프로시저 (또는 **EXECUTE**)를 사용하여 저장 프로시저를 호출하고 필요한 입력 값을 전달할 수 있습니다. 예를 들어 Management Studio 에서 다음 문을 실행해 봅니다.

SQL 복사

```
EXEC [dbo].[PredictTipSingleMode] 1, 2.5, 631, 40.763958,-73.973373,
40.782139,-73.977303
```

여기에 전달 되는 값은 각각 변수 *passenger_count*, *trip_distance*, *trip_time_in_secs*, *pickup_latitude*, *pickup_longitude*, *dropoff_latitude*, *dropoff_longitude* 입니다.

3. 동일한 호출을 R 코드에서 실행하려면 다음과 같이 전체 저장 프로시저 호출을 포함하는 R 변수를 정의하면 됩니다.

R 복사

```
q2 = "EXEC PredictTipSingleMode 1, 2.5, 631, 40.763958,-73.973373,
40.782139,-73.977303 ";
```

여기에 전달 되는 값은 각각 변수 *passenger_count*, *trip_distance*, *trip_time_in_secs*, *pickup_latitude*, *pickup_longitude*, *dropoff_latitude*, *dropoff_longitude* 입니다.

4. `sqlQuery` (**RODBC** 패키지에서)를 호출하고 연결 문자열과 함께 저장 프로시저 호출을 포함한 문자열 변수를 전달합니다.

R 복사

```
# predict with stored procedure in single mode

sqlQuery (conn, q2);
```

팁

R Tools for Visual Studio (RTVS)는 SQL Server 와 R 모두를 위한 뛰어난 통합을 제공합니다. SQL Server 연결에서 RODBC 에 관한 더 많은 예제 다음 기사를 참조합니다: SQL Server 와 R 작업

## 요약

이제 SQL Server 데이터로 작업하는 방법과 학습된 R 모델을 SQL Server 에 저장하는 방법을 배웠음으로 이러한 데이터 세트에 기반해 새로운 모델을 만드는 것은 상대적으로 쉬울 것입니다. 예를 들어, 다음과 같은 추가 모델을 만들어볼 수 있습니다.

- 팁 금액을 예측하는 회귀 모델
- 팁이 많은지 보통인지 적은지 예측하는 다중 클래스 분류 모델

다음 추가 샘플과 리소스 또한 확인하기를 권장합니다.

- 데이터 과학 시나리오와 솔루션 템플릿
- 데이터베이스 내 고급 분석
- Microsoft R - 데이터 분석 살펴보기
- 추가 리소스

## 이전 단원

R 모델을 작성하고 SQL Server 에 저장하기

## 다음 단계

SQL Server R 자습서

sqlrutils 를 사용하여 저장 프로시저 만드는 방법