

# Transact-SQL 에서 R 코드 사용하기 (R in SQL 빠른 시작)

원문:<https://docs.microsoft.com/ko-kr/sql/advanced-analytics/tutorials/rtsql-using-r-code-in-transact-sql-quickstart?view=sql-server-2016>

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

이 자습서에서는 T-SQL 저장 프로시저에서 R 스크립트를 호출하는 기본 메커니즘을 단계별로 안내합니다.

## 학습 내용

- T-SQL 함수에 R 을 포함하는 방법
- R 및 SQL 데이터 형식과 데이터 개체로 작업하기 위한 몇 가지 팁
- 간단한 모델을 만들고 SQL Server 에 저장하는 방법
- 예측 및 모델을 사용하는 R 플롯을 만드는 방법

## 예상 시간

30 분, 설치 제외

## 사전 요구 사항

다음 중 하나가 설치된 상태로 SQL Server 의 인스턴스에 액세스할 수 있어야 합니다.

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

Azure 가상 컴퓨터 또는 온-프레미스 SQL Server 인스턴스가 될 수 있습니다. 외부 스크립팅 기능은 기본적으로 비활성화되어 있으므로 작동하기 위해서는 몇 가지 추가 단계를 수행해야 합니다.

R 스크립트를 포함한 SQL 쿼리를 실행하려면 데이터베이스에 연결하고 T-SQL 코드를 실행할 수 있는 다른 응용 프로그램을 사용할 수 있습니다. SQL 전문가라면 SQL Server Management Studio (SSMS) 또는 Visual Studio 를 사용할 수 있습니다.

이 자습서에서는 SQL Server 내에서 R 을 실행하는 것이 얼마나 쉬운지 보여주기 위해 새로운 **Visual Studio Code 용 mssql 확장**을 사용했습니다. VS Code 는 Windows, Linux, macOS 에서 실행할 수 있는 무료 개발 환경입니다. **mssql** 확장은 SQL 쿼리를 실행하기 위한 가벼운 확장입니다. 이 확장을 설치하려면 [Use the mssql extension for Visual Studio Code](#)(Visual Studio Code 용 mssql 확장 사용) 문서를 참조하세요.

## 데이터베이스에 연결하고 Hello World 테스트 스크립트 실행

1. Visual Studio Code 에서 새 텍스트 파일을 만들고 이름을 BasicRSQL.sql 로 지정합니다.
2. 이 파일이 열리면 CTRL+SHIFT+P(macOS 의 경우 COMMAND + P)를 누르고, **sql** 을 입력하여 SQL 명령을 나열하고, **CONNECT** 를 선택합니다. Visual Studio 코드를 사용하면 특정 데이터베이스에 연결할 때 사용할 프로필을 만들 것인지 묻습니다. 이것은 선택사항이지만 데이터베이스와 로그인 간에 전환을 쉽게할 수 있습니다.
  - SQL Server R 이 설치된 서버 또는 인스턴스를 선택합니다.
  - 새 데이터베이스를 만들 권한이 있는 계정을 사용하여 SELECT 문을 실행하고 테이블 정의를 확인합니다.

3. 연결에 성공하면 상태 표시줄에 서버 및 데이터베이스 이름과 현재 자격 증명이 표시되어야 합니다. 연결에 실패하면 컴퓨터 이름과 서버 이름이 정확한지 확인하세요.
4. 이 문을 붙여 넣고 실행합니다.

SQL 복사

```
EXEC sp_execute_external_script

@language =N'R',

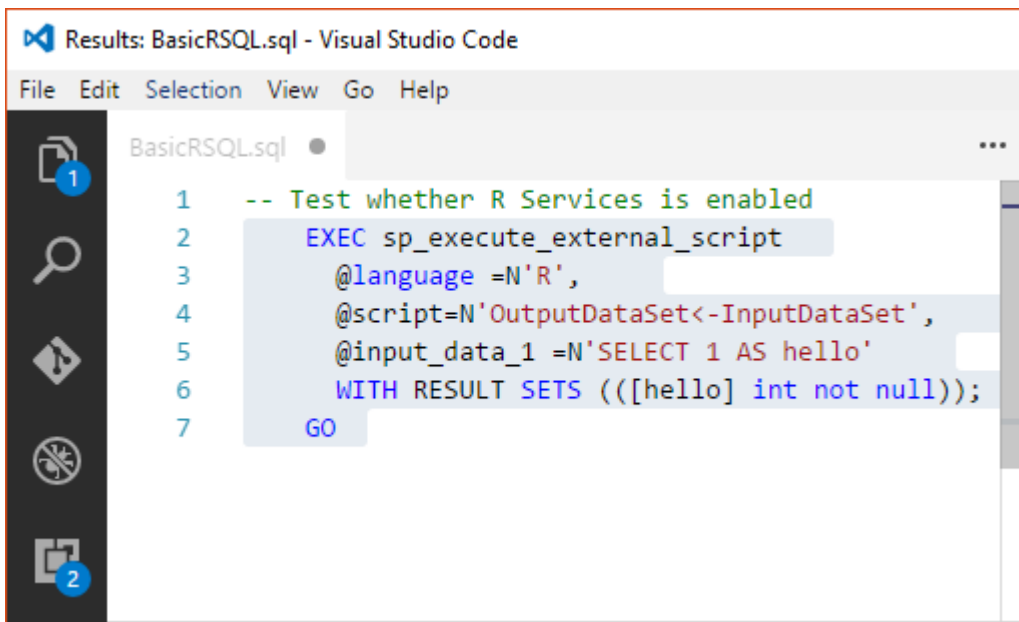
@script=N'OutputDataSet<-InputDataSet',

@input_data_1 =N'SELECT 1 AS hello'

WITH RESULT SETS ([[hello] int not null));

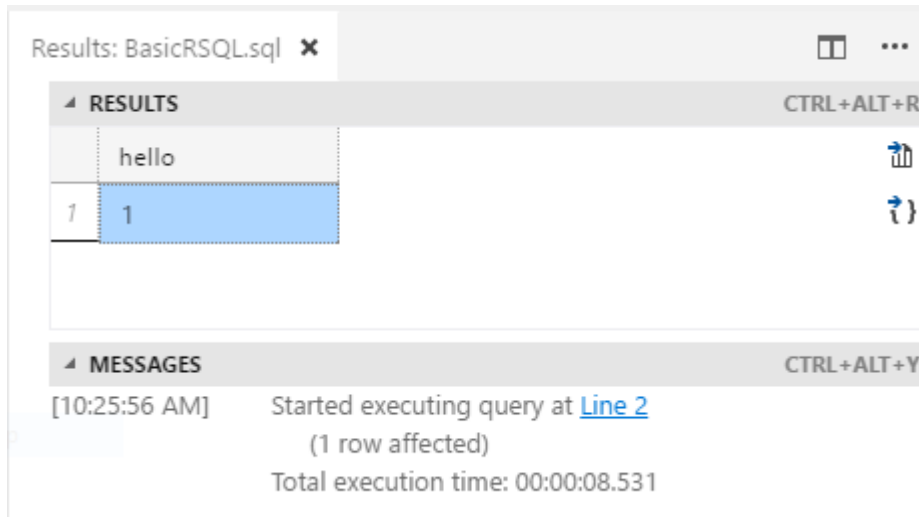
GO
```

Visual Studio Code 에서 실행할 코드를 선택하고 CTRL+SHIFT+E 를 누르면 됩니다. 바로 가기 키를 기억하기 어려우면 변경할 수 있습니다. [Customize the shortcut key bindings](#)(바로 가기 키 바인딩 사용자 지정)를 참조하세요.



```
Results: BasicRSQL.sql - Visual Studio Code
File Edit Selection View Go Help
BasicRSQL.sql
1 -- Test whether R Services is enabled
2 EXEC sp_execute_external_script
3 @language =N'R',
4 @script=N'OutputDataSet<-InputDataSet',
5 @input_data_1 =N'SELECT 1 AS hello'
6 WITH RESULT SETS ([[hello] int not null));
7 GO
```

결과



## 문제 해결

- 이 쿼리에서 오류가 발생한다면 설치가 완료되지 않았을 수 있습니다. SQL Server 설치 마법사를 사용하여 기능을 추가한 후 외부 코드 라이브러리를 사용할 수 있도록 몇 가지 추가 단계를 수행해야 합니다. [SQL Server R Services 설치](#)를 참조하세요.
- Launchpad 서비스가 실행 중인지 확인합니다. 환경에 따라 SQL Server 에 연결할 R 작업자 계정을 사용하도록 설정하거나, 추가 네트워크 라이브러리를 설치하거나, 원격 코드 실행을 사용하도록 설정하거나, 구성이 완료된 후 인스턴스를 다시 시작해야 할 수 있습니다. [R Services 설치 및 업그레이드 FAQ](#) 를 참조하세요.
- Visual Studio Code 를 다운로드하려면 [Download and install Visual Studio Code](#)(Visual Studio Code 다운로드 및 설치)를 참조하세요.

## 다음 단원

인스턴스가 R 로 작업할 준비가 되었으므로 이제 시작하겠습니다.

1 단원: [입 / 출력 작업](#)

2 단원: R 및 SQL 데이터 형식과 데이터 개체

3 단원: R 함수를 SQL Server 데이터와 함께 사용하기

4 단원: 예측 모델 만들기

5 단원: 모델에서 예측과 플롯

# 입력 및 출력 작업 (R in SQL 빠른 시작)

SQL Server 에서 R 코드를 실행하려는 경우 R 스크립트를 시스템 저장 프로시저 `sp_execute_external_script` 내부에 지정합니다. 이 저장 프로시저는 SQL Server 의 컨텍스트에서 R 런타임 시작하고, 데이터를 R 에 전달하고, R 사용자 세션을 안전하게 관리하며 결과를 클라이언트로 반환하는데 사용됩니다.

## 몇 가지 간단한 테스트 데이터 만들기

다음 T-SQL 문을 실행하여 테스트 데이터로 구성된 작은 테이블을 만듭니다.

SQL 복사

```
CREATE TABLE RTestData ([col1] int not null) ON [PRIMARY]
INSERT INTO RTestData VALUES (1);
INSERT INTO RTestData VALUES (10);
INSERT INTO RTestData VALUES (100) ;
GO
```

테이블을 만들었으면 다음 문을 사용하여 테이블을 쿼리합니다.

SQL 복사

```
SELECT * FROM RTestData
```

결과

**col1**

1.

col1

10

100

## R 스크립트를 사용하여 동일한 데이터 가져오기

테이블을 만든 후 다음 문을 실행합니다.

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' OutputDataSet <- InputDataSet;'
    , @input_data_1 = N' SELECT * FROM RTestData;'
    WITH RESULT SETS (([NewColName] int NOT NULL));
```

테이블에서 데이터를 가져옵니다, R 런타임을 왕복한 뒤 *NewColName* 열 이름으로 값을 반환합니다.

결과

Results: BasicRSQL.sql x [Icons]

RESULTS		CTRL+ALT+R
	NewColName	[Icons]
1	1	[Icons]
2	10	
3	100	

---

MESSAGES CTRL+ALT+Y

[11:07:06 AM] Started executing query at [Line 18](#)  
 (3 rows affected)  
 Total execution time: 00:00:03.093

## 주석

- Management Studio에서는 테이블 형식 결과가 **결과** 탭에 반환됩니다. R 런타임에서 반환된 메시지는 **메시지** 탭에 제공됩니다. (역주: 원문은 Values pane에 결과가 반환된다고 언급하고 있어서 “결과” 탭으로 수정했습니다)
- `@language` 매개 변수는 호출할 언어 확장(이 경우 R)을 정의합니다.
- `@script` 매개 변수에서 R 런타임에 전달할 명령을 정의합니다. 전체 R 스크립트는 이 인수에 유니코드 텍스트로 포함되어야 합니다. `nvarchar` 형식의 변수에 텍스트를 추가한 다음 변수를 호출할 수도 있습니다.
- 쿼리에서 반환된 데이터는 R 런타임에 전달되고 R 런타임은 데이터를 SQL Server에 데이터 프레임으로 전달합니다.
- WITH RESULT SETS 절은 SQL Server에 반환된 데이터 테이블의 스키마를 정의합니다.

## 입력 또는 출력 변수 변경



이전 예제에서는 기본 입력 및 출력 변수 이름으로 *InputDataSet* 및 *OutputDataSet\_*을 사용했습니다. *InputDataSet\_*과 연결된 입력 데이터를 정의하려면 *@input\_data\_1* 변수를 사용합니다.

이번 예제에서는 저장 프로시저의 출력 및 입력 변수 이름을 *SQL\_Out* 및 *SQL\_In*으로 변경했습니다.

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' SQL_out <- SQL_in;'
    , @input_data_1 = N' SELECT 12 as Col;'
    , @input_data_1_name = N'SQL_In'
    , @output_data_1_name = N'SQL_Out'
    WITH RESULT SETS (([NewColumnName] int NOT NULL));
```

(역주. 이후 설명과 실제 오류 내용이 맞지 않아 내용을 편집했습니다. 궁금하신 분들은 원문을 참조하세요)

“sp\_execute\_external\_script'를 실행하는 동안 'R' 스크립트 오류가 발생했습니다...” 라는 오류가 발생합니다. 원인은 R 이 대/소문자를 구분하기 때문입니다. R 스크립트에서는 변수명으로 *SQL\_in* 및 *SQL\_out*을 사용하지만 저장 프로시저의 매개 변수는 *SQL\_In* 및 *SQL\_Out*으로 정의되었기 때문입니다.

이제 *SQL\_In* 변수를 대소문자 동일하게 맞춘 뒤 프로시저를 다시 실행합니다.

또 다른 오류가 발생합니다:

Error 복사

EXECUTE 문의 WITH RESULT SETS 절에 1 개의 결과 세트가 지정되었지만 런타임에 0 개의 결과 세트만 보냈으므로 문의 실패했습니다.

새로운 R 코드를 테스트할 때 자주 만날 수 있는 오류입니다. R 스크립트는 성공적으로 실행했지만 SQL Server 가 수신한 데이터가 없거나 혹은 잘못되거나 예기치 않은 데이터를 받았음을 의미합니다.

위의 경우 출력 스키마(WITH 로 시작하는 줄)는 정수 데이터 열 하나를 반환되도록 지정했지만, R 에서 다른 변수에 데이터를 넣기 때문에 SQL Server 에 아무것도 반환되지 않으므로 오류가 발생합니다. 오류를 해결하려면 두 번째 변수 이름을 수정하십시오.

다음 요구 사항을 기억하세요!

- 변수 이름은 유효한 SQL 식별자 규칙을 따라야 합니다.
- 매개 변수의 순서가 중요합니다. 선택적 매개 변수 `@input_data_1_name` 및 `@output_data_1_name` 을 사용하기 위해 먼저 필수 매개 변수 `@input_data_1` 및 `@output_data_1` 을 지정해야 합니다.
- 하나의 입력 데이터 세트만 매개 변수로 전달할 수 있으며 하나의 데이터 세트만 반환할 수 있습니다. 그러나 R 코드 내에서 다른 데이터 세트를 호출할 수 있으며 데이터 세트 외에 다른 유형의 출력을 반환할 수 있습니다. 또한 모든 매개 변수에 OUTPUT 키워드를 추가하여 결과와 함께 매개 변수를 반환할 수도 있습니다. 이 자습서의 뒷부분에 간단한 예제가 있습니다.
- **WITH RESULT SETS** 문은 SQL Server 를 위해 데이터에 대한 스키마를 정의합니다. R 에서 반환하는 각 열에 대한 SQL 호환 데이터 형식을 제공해야 합니다. 스키마 정의를 사용하여 새 열 이름도 제공할 수 있고 R data.frame 의 열 이름을 사용할 필요가 없습니다. 경우에 따라 이 절은 생략 가능합니다. 생략하면 어떻게 되는지 확인해 보세요.

## R 을 사용하여 결과 생성

R 스크립트를 사용하여 값을 생성하고 `@input_data_1_` 의 입력 쿼리 문자열은 비워 둘 수도 있습니다. 또는 유효한 SQL SELECT 문을 자리 표시자로 사용하고 R 스크립트에는 SQL 결과를 사용하지 마세요.

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' mytextvariable <- c("hello", " ", "world");
        OutputDataSet <- as.data.frame(mytextvariable);'
    , @input_data_1 = N' SELECT 1 as Temp1'
    WITH RESULT SETS (([Co11] char(20) NOT NULL));
```

결과

*Co11*

*hello*

*world*

## 다음 단원

R 과 SQL 간 테이블 형식 데이터의 암시적 변환 및 차이점을 포함하여 R 과 SQL Server 간에 데이터를 전달할 때 발생할 수 있는 몇 가지 문제를 살펴보겠습니다.

[R 및 SQL 데이터 형식 및 데이터 개체](#)

# R 및 SQL 데이터 형식과 데이터 개체 (R in SQL 빠른 시작)

이 단계에서는 R 과 SQL Server 간에 데이터를 이동 시 발생하는 몇 가지 일반적인 문제에 관해 배웁니다.

- 데이터 형식이 일치하지 않음
- 암시적 변환이 일어날 수 있음
- CAST 및 Convert 연산이 필요함
- R 및 SQL 이 서로 다른 데이터 개체를 사용함

## R 데이터는 항상 데이터 프레임으로 반환

스크립트가 R 에서 SQL Server 로 결과를 반환할 경우 데이터를 **data.frame** 으로 반환해야 합니다. 스크립트에서 생성하는 다른 형식의 개체는 리스트(list), 팩터(factor), 벡터(vector) 또는 이진 데이터이든 관계없이 저장 프로시저 결과의 일부로 출력하려면 데이터 프레임으로 변환되어야 합니다. 다행히도 기타 개체를 데이터 프레임으로 변경하는 기능을 지원하는 여러 R 함수가 있습니다. 이진 모델 또한 직렬화해서 데이터 프레임으로 반환할 수 있으며, 이 자습서의 뒷부분에서 수행할 것입니다.

우선, 벡터, 행렬 및 리스트와 같은 몇 가지 기본 R 개체를 실험한 뒤 데이터 프레임으로 변환하면 SQL Server 로 전달되는 출력이 어떻게 변경되는지 확인합니다.

R 에서 다음 두 "Hello World" 스크립트를 비교하세요. 스크립트는 거의 동일 하지만 첫 번째는 세 개의 값을 단일 열로 반환하는 반면, 두 번째는 각각 단일 값을 가진 3 개의 열을 반환합니다.

### 예제 1

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' mytextvariable <- c("hello", " ", "world");
    OutputDataSet <- as.data.frame(mytextvariable);'
    , @input_data_1 = N' ';
```

### 예제 2

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' OutputDataSet<- data.frame(c("hello"), " ",
    c("world"));'
    , @input_data_1 = N' ';
```

## R 데이터의 스키마 및 데이터 형식 식별

왜 결과가 다를까요?

일반적으로 R `str()` 명령을 사용하여 그 대답을 찾을 수 있습니다. R 스크립트 임의의 위치에 `str(object_name)` 함수를 추가하여 지정된 R 개체의 데이터

스키마를 정보성 메시지로 반환합니다. 메시지를 보려면 Visual Studio Code 의 **Message** 창 또는 SSMS 의 **메시지** 탭에서 확인합니다.

예제 1 과 예제 2 의 결과가 다른 이유를 확인하려면 다음과 같이 각 문에서 `@script` 변수 정의의 끝 부분에 `str(OutputDataSet)` 줄을 삽입합니다.

### 예제 1 str 함수 추가

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' mytextvariable <- c("hello", " ", "world");
    str(OutputDataSet);'
    , @input_data_1 = N' '
;
```

### 예제 2 str 함수 추가

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R',
    @script = N' OutputDataSet <- data.frame(c("hello"), " ",
    c("world"));
    str(OutputDataSet)' ,
    @input_data_1 = N' ';
```

이제 **Message** 에 텍스트를 검토하여 출력이 다른 이유를 확인합니다. (역주. 결과 - 예제 1 은 실제론 다르게 출력됩니다)

### 결과 - 예제 1

복사

```
STDOUT message(s) from external script:  
'data.frame':  3 obs. of  1 variable:  
$ mytextvariable: Factor w/ 3 levels " ", "hello", "world": 2 1 3
```

결과 - 예제 2

복사

```
STDOUT message(s) from external script:  
'data.frame':  1 obs. of  3 variables:  
$ c..hello...: Factor w/ 1 level "hello": 1  
$ X...      : Factor w/ 1 level " ": 1  
$ c..world...: Factor w/ 1 level "world": 1
```

확인한 것처럼 R 구문을 약간 변경한 것이 결과 스키마에 큰 영향을 미쳤습니다. R 데이터 형식의 차이점은 Hadley Wickham 의 글 [R 데이터 구조](#)에서 보다 자세하게 설명하므로 여기서는 생략합니다.

단지 지금은 R 개체를 데이터 프레임으로 강제 변환할 때 예상되는 결과를 확인할 필요가 있음을 숙지하십시오.

팁

`is.matrix`, `is.vector` 등과 같은 R 함수를 사용할 수도 있습니다.

## 데이터 개체의 암시적 변환

각 R 데이터 개체에는 두 개의 데이터 개체가 같은 수의 차원을 가지거나 데이터 개체에 다른 데이터 형식이 포함될 경우 다른 데이터 개체와 결합될 때 값을 처리하는 방법에 대한 자체 규칙이 있습니다.

예를 들어 다음 문을 실행하여 R 을 사용한 행렬 곱셈을 수행한다고 가정합니다. 3 개 값이 포함된 단일 열 행렬에 4 개 값이 포함된 배열을 곱하고 결과로 4x3 행렬을 예상합니다.

SQL 복사

```
EXECUTE sp_execute_external_script

  @language = N'R'

  , @script = N'

      x <- as.matrix(InputDataSet);

      y <- array(12:15);

      OutputDataSet <- as.data.frame(x %*% y);'

  , @input_data_1 = N' SELECT [Col1] from RTestData;'

  WITH RESULT SETS (([Col1] int, [Col2] int, [Col3] int, Col4
int));
```

내부적으로 3 개의 값으로 구성된 단일 열이 단일-열 행렬로 변환됩니다. 행렬은 R 에서 단지 배열의 특수한 형태이므로, 배열 `y` 는 두 인수가 일치하도록 단일 열 행렬로 강제 변환됩니다.

## 결과

**Col1 Col2 Col3 Col4**

12 13 14 15

120 130 140 150

1200 1300 1400 1500

하지만 배열 크기를 변경하면 어떻게 되는지 주목하세요.

SQL 복사



```

execute sp_execute_external_script
    @language = N'R'
    , @script = N'
        x <- as.matrix(InputDataSet);
        y <- array(12:14);
        OutputDataSet <- as.data.frame(y %*% x);'
    , @input_data_1 = N' SELECT [Col1] from RTestData;'
    WITH RESULT SETS (([Col1] int ));

```

이제 R 은 단일 값을 반환합니다.

## 결과

**Col1**

1542

이유가 무엇일까요? 이 경우 두 인수는 동일한 길이의 벡터로 처리될 수 있으므로 R 이 내적(inner product)을 행렬로 반환합니다. 이것은 선형 대수 규칙에 따라 예상된 동작입니다. 하지만 다운스트림 응용 프로그램에서 출력 스키마가 절대 변경되지 않을 것으로 예상할 경우에는 이 동작으로 인해 문제가 발생할 수 있습니다.

## 팁

오류가 발생 했습니까? 이 예제는 **RTestData** 테이블을 필요로 합니다. 테스트 데이터 테이블을 만들지 않은 경우 [입 / 출력 작업](#) 주제로 다시 이동하세요.

테이블이 만들어져도 여전히 오류가 발생한다면, 해당 테이블을 포함한 데이터베이스 내에서 저장 프로시저를 실행하고 있는지 **master** 혹은 다른 데이터베이스가 아닌지 확인해 보세요.

또한 예제에서 임시 테이블을 사용하지 않는 것이 좋습니다. 일부 R 클라이언트는 일괄 처리 사이의 연결을 종료하고 임시 테이블을 삭제합니다.

## 다른 길이의 열 병합 또는 곱하기

R은 다양한 크기의 벡터로 작업하고 이러한 열과 같은 구조를 데이터 프레임에 결합하는데 뛰어난 유연성을 제공합니다. 벡터 리스트는 테이블처럼 보일 수 있지만 데이터베이스 테이블을 관리하는 모든 규칙을 따르지 않습니다.

예를 들어 다음 스크립트는 길이가 6인 숫자 배열을 정의하고 R 변수 `df1`에 저장합니다. 그런 다음 숫자 배열에는 3개의 값이 포함된 `RTestData` 테이블의 정수와 결합하여 세 데이터 프레임 `df2`를 만듭니다.

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N'
        df1 <- as.data.frame( array(1:6) );
        df2 <- as.data.frame( c( InputDataSet , df1 ) );
        OutputDataSet <- df2'
    , @input_data_1 = N' SELECT [Col1] from RTestData;'
    WITH RESULT SETS (( [Col2] int not null, [Col3] int not null ));
```

R은 데이터 프레임을 채우기 위해 `RTestData`에서 검색된 요소를 `df1` 배열의 요소 수와 일치하도록 필요한 횟수만큼 반복합니다.

결과

*Col2 Col3*

1 1

10 2

100 3

1 4

10 5

100 6

데이터 프레임이 테이블처럼 보이지만 실제로는 벡터의 리스트라는 것을 기억하세요.

## SQL Server 데이터 Cast 또는 convert

R 과 SQL Server 는 같은 데이터 형식을 사용하지 않으므로, SQL Server 에서 쿼리를 실행하여 데이터를 가져오고 이를 R 런타임에 전달할 경우 일부 유형에서 종종 암시적 변환이 이루어집니다. R 에서 SQL Server 로 데이터를 반환할 경우에는 또 다른 일련의 변환이 이루어집니다.

- SQL Server 는 쿼리의 데이터를 Launchpad 서비스가 관리하는 R 프로세스에 넣고 효율성을 높이기 위한 내부 표현으로 변환합니다.
- R 런타임이 데이터를 data.frame 변수에 로드하고 데이터에서 고유한 작업을 수행합니다.
- 데이터베이스 엔진은 보안 설정된 내부 연결을 통해 데이터를 SQL Server 로 반환하고 SQL Server 데이터 형식에 따라 데이터를 제공합니다.
- SQL 쿼리를 실행하고 테이블 형식 데이터 세트를 처리할 수 있는 클라이언트 또는 네트워크 라이브러리를 사용하여 SQL Server 에 연결하고 데이터를 가져옵니다. 이 클라이언트 응용 프로그램은 다른 방식으로 데이터에 영향을 미칠 수 있습니다.

이 응용 프로그램이 작동하는 방식을 확인하려면 AdventureWorksDW 데이터 웨어하우스에서 이와 같은 쿼리를 실행합니다. 이 뷰는 예측 생성에 사용된 매출 데이터를 반환합니다.

SQL 복사

```
SELECT ReportingDate
       , CAST(ModelRegion as varchar(50)) as ProductSeries
       , Amount
FROM [AdventureWorksDW2014].[dbo].[vTimeSeries]
WHERE [ModelRegion] = 'M200 Europe'
ORDER BY ReportingDate ASC
```

참고

어떠한 버전의 AdventureWorks 도 사용할 수 있으며, 자신의 데이터베이스를 사용해 다른 쿼리를 만들 수 있습니다. 요점은 텍스트, 날짜/시간 및 숫자 값이 포함된 일부 데이터를 처리하는 것입니다.

이제, 저장 프로시저에 입력으로 이 쿼리를 붙여 넣으세요.

SQL 복사

```
EXECUTE sp_execute_external_script
    @language = N'R'
    , @script = N' str(InputDataSet);
    OutputDataSet <- InputDataSet;'
    , @input_data_1 = N'
        SELECT ReportingDate
           , CAST(ModelRegion as varchar(50)) as ProductSeries
```

```

, Amount
FROM [AdventureWorksDW2014].[dbo].[vTimeSeries]
WHERE [ModelRegion] = 'M200 Europe'
ORDER BY ReportingDate ASC ;'
WITH RESULT SETS undefined;

```

오류가 발생하면 쿼리 텍스트를 약간 편집해야 할 수 있습니다. 예를 들어 WHERE 절의 문자열 조건자는 두 개의 작은따옴표 집합으로 묶어야 합니다.

쿼리를 실행한 후 `str` 함수의 결과를 검토하여 R에서 입력 데이터가 어떻게 처리되는지 확인합니다.

## 결과

복사

```

STDOUT message(s) from external script: 'data.frame':  37 obs. of
3 variables:

STDOUT message(s) from external script: $ ReportingDate: POSIXct,
format: "2010-12-24 23:00:00" "2010-12-24 23:00:00"

STDOUT message(s) from external script: $ ProductSeries: Factor w/ 1
levels "M200 Europe",...: 1 1 1 1 1 1 1 1 1 1

STDOUT message(s) from external script: $ Amount      : num  3400
16925 20350 16950 16950

```

- datetime 열은 R 데이터 형식 **POSIXct** 를 사용하여 처리되었습니다.
- 텍스트 "ProductSeries"는 **팩터**로 인식되었으며 이는 범주 변수임을 의미합니다. 문자열 값은 기본적으로 팩터로 처리됩니다. R에 문자열을 전달하면 문자열은 내부 사용을 위해 정수로 변환되고 다시 문자열에 매핑되어서 출력됩니다.

요약

간단한 예제를 통해 입력으로 SQL 쿼리를 전달할 때 데이터 변환의 결과 확인할 필요가 있음을 볼 수 있습니다. 일부 SQL Server 데이터 형식은 R에서 지원되지 않으므로, 오류를 방지하려면 다음과 같은 방법을 고려합니다.

- 데이터를 미리 테스트해서 R 코드에 전달할 때 문제가 될 수 있는 스키마의 열 혹은 값을 검사합니다.
- `SELECT *`를 사용하지 않고 입력 데이터 원본에서 열을 개별적으로 지정하고 각 열이 어떻게 처리되는지 알아봅니다.
- 문제를 방지하려면 입력 데이터를 준비할 때 필요에 따라 명시적 Cast를 수행합니다.
- 오류를 유발하거나 모델링에 유용하지 않은 데이터 열(예: GUID 또는 rowguids)은 전달하지 않습니다.

지원되는 혹은 지원되지 않는 데이터 형식에 대한 추가 정보는 [R 라이브러리 및 데이터 형식](#)을 참조하십시오.

런타임에 문자열에서 숫자 팩터로의 변환이 성능에 미치는 영향에 대한 정보는 [SQL Server R Services 성능 튜닝](#)을 참조하십시오.

## 다음 단원

다음 단계에서는 SQL Server 데이터에 R 함수를 적용하는 방법을 알아봅니다.

[R 함수를 SQL Server 데이터와 함께 사용하기](#)

# R 함수를 SQL Server 데이터와 함께 사용하기 (R in SQL 빠른 시작)

이제 기본 작업에 익숙해졌으므로 R의 재미있는 점을 살펴보겠습니다. 예를 들어 T-SQL을 사용하면 복잡할 수도 있지만 R 코드에서 한 줄이면 충분한 여러가지 고급 통계 함수가 있습니다. R Services를 통해 R 유틸리티 스크립트를 저장 프로시저에 쉽게 포함할 수 있습니다.

이 예제에서는 R의 수학 및 유틸리티 함수를 SQL Server 저장 프로시저에 포함시킬 것입니다.

## 난수를 생성하는 저장 프로시저 만들기

단순하게 사용하기 위해, R Services에서 기본적으로 설치 및 로드되는 R stats 패키지를 사용합니다. 이 패키지에는 일반 통계 작업용으로 수백 개의 함수가 포함되어 있으며, 그 중 `rnorm` 함수는 주어진 표준 편차와 평균으로 정규 분포를 사용한 지정된 수의 난수를 생성합니다.

예를 들어 이 R 코드는 평균 50, 표준 편차 3에 기반한 100개의 숫자를 반환합니다.

R 복사

```
as.data.frame(rnorm(100, mean = 50, sd = 3));
```

이 R 라인을 T-SQL에서 호출하려면 `sp_execute_external_script`를 실행하고 다음과 같이 R 스크립트 매개 변수에 R 함수를 추가합니다.

SQL 복사

```
EXEC sp_execute_external_script
    @language = N'R'
    , @script = N'
        OutputDataSet <- as.data.frame(rnorm(100, mean = 50, sd
=3));'
    , @input_data_1 = N'    ;'
    WITH RESULT SETS (([Density] float NOT NULL));
```

다른 난수 세트 생성을 쉽게 하려면 어떻게 하면 될까요?

SQL Server 랑 함께 사용하면 쉽습니다: 사용자로부터 인수를 받는 저장 프로시저를 정의합니다. 그 다음 그 인수들 R 스크립트에 변수로 넘겨줍니다.

SQL 복사

```
CREATE PROCEDURE MyRNorm (@param1 int, @param2 int, @param3 int)
AS
EXEC sp_execute_external_script
    @language = N'R'
    , @script = N'
        OutputDataSet <- as.data.frame(rnorm(mynumbers, mymean,
mysd));'
    , @input_data_1 = N'    ;'
    , @params = N' @mynumbers int, @mymean int, @mysd int'
    , @mynumbers = @param1
    , @mymean = @param2
    , @mysd = @param3
    WITH RESULT SETS (([Density] float NOT NULL));
```



- 첫 번째 줄은 저장 프로시저가 실행될 때 필요한 SQL 입력 매개 변수를 정의합니다.
- `@params`로 시작하는 줄은 R 코드에서 사용되는 변수들과 해당하는 SQL 데이터 형식을 정의합니다.
- 바로 뒤에 오는 줄은 SQL 매개 변수 이름을 대응하는 R 변수 이름에 지정합니다.

이제 저장 프로시저에서 R 함수를 래핑했으므로 다음과 같이 쉽게 함수를 호출하고 다른 값을 전달 할 수 있습니다.

SQL 복사

```
EXEC MyRNorm @param1 = 100,@param2 = 50, @param3 = 3
```

## 관련 리소스

- 더 많은 고급 통계 함수를 사용하기 위해 추가 R 패키지를 설치하시겠습니까? [R 패키지 설치 및 관리](#)를 참조합니다.
- Microsoft R 팀에서 새로운 R 패키지인 **sqlutils** 를 제공합니다, 이는 R 코드를 SQL Server 저장 프로시저에서 쉽게 사용할 수 있는 매개변수로 형태로 변환해 줍니다. 자세한 내용은 [sqlutils 를 사용하여 저장된 프로시저를 만드는 방법을](#) 참조합니다.

## 문제 해결에 R 유틸리티 함수 사용

기본적으로 R 설치에는 `utils` 패키지가 포함되며 현재 R 환경을 조사하기 위한 다양한 유틸리티 함수를 제공합니다. 이 패키지는 R 코드가 SQL Server 내부와 외부 환경에서 실행되는 방식에 불일치 찾는 경우 유용할 수 있습니다.

예를 들어, R의 `memory.limit()` 함수를 사용하여 현재 R 환경에 사용하는 메모리를 알 수 있습니다. `utils` 패키지는 기본적으로 로드되지 않으므로 먼저 `library()` 함수를 사용하여 로드해야 합니다.

SQL 복사

```
EXECUTE sp_execute_external_script

    @language = N'R'

, @script = N'

    library(utils);

    mymemory <- memory.limit();

    OutputDataSet <- as.data.frame(mymemory);'

, @input_data_1 = N' ;'

WITH RESULT SETS (([Col1] int not null));
```

많은 사용자가 성능 문제 분석을 위해 R 프로세스가 사용하는 시간을 캡처해주는 `system.time` 및 `proc.time` 같은 시스템 타이밍 함수를 사용합니다.

예제를 보려면 [3 단원: 데이터 특성 만들기\(R 및 SQL Server 용 데이터 과학 전체 과정 연습\)](#) 자습서를 참조하세요. 이 연습에서는 데이터로부터 특성을 만들기 위한 두 가지 방법(R 함수 vs. T-SQL 함수)의 성능을 비교하는 해결 방법에서 R 타이밍 함수가 포함됩니다. (역주: `proc_time()`이 사용됨)

## 다음 단원

다음으로 SQL Server 에서 R 을 사용하여 예측 모델을 빌드합니다.

[예측 모델 만들기](#)

# 예측 모델 만들기 (R in SQL 빠른 시작)

이 단계에서는 R 을 사용하여 모델을 학습한 다음 SQL Server 의 테이블에 모델을 저장하는 방법을 알아봅니다. 모델은 자동차의 속도에 따른 정지 거리를 예측하는 간단한 회귀 모델입니다. R 에 내장된 `cars` 데이터 세트는 작고 이해하기 쉬우므로 이를 사용할 것입니다.

## 원본 데이터 만들기

먼저 학습할 데이터를 저장할 테이블을 만듭니다.

SQL 복사

```
CREATE TABLE CarSpeed ([speed] int not null, [distance] int not null)

INSERT INTO CarSpeed

EXEC sp_execute_external_script

    @language = N'R'

    , @script = N'car_speed <- cars;'

    , @input_data_1 = N''

    , @output_data_1_name = N'car_speed'
```

- 임시 테이블 사용을 좋아하는 사람도 있겠지만, 일부 R 클라이언트는 배치 처리 간의 세션 연결을 끊다는 점을 유의하세요.
- R 런타임에는 작거나 데이터 세트가 많이 포함되어 있습니다. R 과 함께 설치된 데이터 세트 목록을 가져오려면 R 명령 프롬프트에서 `library(help="datasets")`를 입력합니다.

## 회귀 모델 만들기

자동차 속도 데이터에는 둘 다 숫자인 `dist` 및 `speed`의 두 열이 포함되어 있습니다. 어떤 속도는 여러 개의 관찰 값이 있습니다. 이 데이터에서 자동차 속도와 자동차를 정지하는 데 필요한 거리 간의 몇 가지 관계를 설명하는 선형 회귀 모델을 만들게 됩니다.

선형 모델의 요구 사항은 간단합니다.

- 독립 변수 `speed`와 종속 변수 `distance` 간의 관계를 설명하는 수식 정의 (역주. 원문에는 두 가지 변수의 순서가 거꾸로 되어 있어서 수정함)
- 모델 학습용 입력 데이터 제공

팁

선형 모델에 재교육(refresher)이 필요하다면 `RxLinMod`를 사용하여 모델을 적합시키는 과정을 설명하는 이 자습서를 권장합니다: [선형 모델 맞춤](#)

실제로 모델을 빌드하려면 R 코드 내에서 수식을 정의하고 데이터를 입력 매개 변수로 전달합니다.

SQL 복사

```
DROP PROCEDURE IF EXISTS generate_linear_model;

GO

CREATE PROCEDURE generate_linear_model

AS

BEGIN

    EXEC sp_execute_external_script

        @language = N'R'

        , @script = N'lmodel <- rxLinMod(formula = distance ~ speed,

data = CarsData);
```

```

    trained_model <- data.frame(payload =
as.raw(serialize(lrmodel, connection=NULL)));
    , @input_data_1 = N'SELECT [speed], [distance] FROM CarSpeed'
    , @input_data_1_name = N'CarsData'
    , @output_data_1_name = N'trained_model'
    WITH RESULT SETS ((model varbinary(max)));
END;
GO

```

- rxLinMod 에 대한 첫 번째 인수는 거리를 속도에 종속된 것으로 정의하는 *formula* 매개 변수입니다.
- 입력 데이터는 SQL 쿼리로 데이터를 채운 `CarsData` 변수에 저장됩니다. 입력 데이터에 특정 이름을 할당하지 않는 경우 기본 변수 이름은 *InputDataSet* 입니다.

## 모델을 저장하기 위한 테이블 만들기

다음으로, 다시 학습시키거나 예측에 사용할 수 있도록 모델을 저장 합니다. 모델을 만드는 R 패키지의 출력은 일반적으로 **이진 개체**입니다. 따라서 모델을 저장하는 테이블은 **varbinary** 형식의 열을 제공해야 합니다.

SQL 복사

```

CREATE TABLE stopping_distance_models (
    model_name varchar(30) not null default('default model') primary
key,
    model varbinary(max) not null);

```

## 모델 저장

모델을 저장하려면 다음 Transact-SQL 문을 실행하여 저장 프로시저를 호출하고 모델을 생성한 다음 테이블에 저장합니다.

SQL 복사

```
INSERT INTO stopping_distance_models (model)
EXEC generate_linear_model;
```

이 코드를 두 번 실행하면 다음 오류가 발생합니다:

복사

```
Violation of PRIMARY KEY constraint...Cannot insert duplicate key in
object dbo.stopping_distance_models
```

이 오류를 방지하는 한 가지 옵션은 각 새 모델의 이름을 업데이트하는 것입니다. 예를 들어 모델의 유형, 생성한 날짜 등을 포함해서 좀 더 서술적인 이름으로 변경할 수 있습니다.

SQL 복사

```
UPDATE stopping_distance_models
SET model_name = 'rxLinMod ' + format(getdate(), 'yyyy.MM.HH.mm',
'en-gb')
WHERE model_name = 'default model'
```

## 추가 변수 출력

일반적으로 저장 프로시저 `sp_execute_external_script`의 R 출력은 단일 데이터 프레임으로 제한됩니다. 이 제한은 나중에 제거될 수도 있습니다.

그러나 데이터 프레임 외에 스칼라 같은 다른 유형의 출력을 반환할 수 있습니다.

예를 들어, 모델을 학습하지만 그 모델의 계수(coefficients) 표를 즉시 보려고 한다고 가정합니다. 계수 표는 기본 결과 세트로 만들고 학습된 모델은 SQL 변수에 출력할 수 있습니다. 변수를 호출하여 모델을 재사용하거나 다음과 같이 테이블에 저장할 수 있습니다.

SQL 복사

```
DECLARE @model varbinary(max), @modelname varchar(30)
EXEC sp_execute_external_script
    @language = N'R'
    , @script = N'
        speedmodel <- rxLinMod(distance ~ speed, CarsData)
        modelbin <- serialize(speedmodel, NULL)
        OutputDataSet <- data.frame(coefficients(speedmodel));'
    , @input_data_1 = N'SELECT [speed], [distance] FROM CarSpeed'
    , @input_data_1_name = N'CarsData'
    , @params = N'@modelbin varbinary(max) OUTPUT'
    , @modelbin = @model OUTPUT
    WITH RESULT SETS (([Coefficient] float not null));

-- Save the generated model
INSERT INTO [dbo].[stopping_distance_models] (model_name, model)
VALUES ('latest model', @model)
```

결과

RESULTS	
	Coefficient
1	-17.579094890...
2	3.93240875912...

요약

`sp_execute_external_script` 에서 SQL 매개변수와 R 변수를 함께 작업할 때

기억할 규칙:

- R 스크립트에 매핑된 모든 SQL 매개 변수는 `@params` 인수에 이름이 나열 되어야 합니다.
- 이러한 매개 변수 중 하나를 출력하려면 `@params` 목록에 `OUTPUT` 키워드를 추가합니다.
- 매핑된 매개 변수를 나열한 후 `@params` 목록 바로 뒤의 R 변수에 SQL 매개 변수의 매핑을 줄 단위로 제공합니다.

## 다음 단원

이제 모델을 만들었으므로 최종 단계에서 모델로부터의 예측을 생성하고 결과를 그림으로 나타내는 방법을 배웁니다.

[모델에서 예측 및 플롯](#)



# 모델에서 예측 및 플롯 (R in SQL 빠른 시작)

새로운 데이터를 사용한 *채점(scoring)*을 위해서 테이블로부터 학습된 모델 중 하나를 가져온 다음 예측을 기반으로 새로운 데이터 세트를 호출합니다.

채점이란 학습된 모델에 주어진 새로운 데이터에 기반한 예측, 확률, 혹은 기타 값을 생성함을 의미하는 데이터 과학에서 종종 사용되는 용어입니다.

## 새로운 속도에 대한 테이블 만들기

원래 학습 데이터가 25mph(마일/시간)의 속도에서 중지된 것을 알아차리셨나요? 이는 원래 데이터가 1920 년의 실험을 기준으로 하기 때문입니다.

자동차가 60mph 또는 100mph 의 속도로 빠르게 달릴 수 있다고 가정하면 1920 년대의 자동차가 중지하는 데 얼마나 오랜 시간이 걸릴지 궁금할 것입니다. 이 질문에 답하기 위해 몇 가지 새로운 속도 값을 제공해야 합니다.

SQL 복사

```
CREATE TABLE [dbo].[NewCarSpeed]([speed] [int] NOT NULL,  
    [distance] [int] NULL) ON [PRIMARY]  
GO  
INSERT [dbo].[NewCarSpeed] (speed)  
VALUES (40), (50), (60), (70), (80), (90), (100)
```

## 중지 거리 예측

지금까지 테이블에는 여러 R 모델이 포함되고 모든 모델은 서로 다른 매개 변수 또는 알고리즘을 사용하여 빌드되거나 다양한 데이터 하위 세트를 기반으로 학습됩니다.

RESULTS		
	model_name	model
1	default model	0x580A000000...
2	rxLinMod 2017.02.12.48	0x580A000000...
3	rxLinMod 2017.02.12.50	0x580A000000...
4	rxLinMod 2017.02.12.51	0x580A000000...
5	rxLinMod 2017.02.14.57	0x580A000000...

MESSAGES	
[3:07:47 PM]	Started executing query at <a href="#">Line 67</a> (5 rows affected) Total execution time: 00:00:00.016

하나의 특정 모델을 기반으로 예측을 얻으려면 다음 작업을 수행하는 SQL 스크립트를 작성합니다.

1. 원하는 모델을 가져옵니다.
2. 새 입력 데이터를 가져옵니다.
3. 해당 모델과 호환되는 R 예측 함수를 호출합니다.

이 예제에서는 **RevoScaleR** 패키지의 일부로 제공되는 **rxLinMod** 알고리즘에 기반한 모델이므로 일반 R의 `predict` 함수 대신 `rxPredict` 함수를 호출합니다.

SQL 복사

```
DECLARE @speedmodel varbinary(max) = (SELECT model FROM
[dbo].[stopping_distance_models] WHERE model_name = 'latest model');
EXEC sp_execute_external_script
```

```

@language = N'R'
, @script = N'
    current_model <- unserialize(as.raw(speedmodel));
    new <- data.frame(NewCarData);
    predicted.distance <- rxPredict(current_model, new);
    str(predicted.distance);
    OutputDataSet <- cbind(new, ceiling(predicted.distance));
    '
, @input_data_1 = N' SELECT speed FROM [dbo].[NewCarSpeed] '
, @input_data_1_name = N'NewCarData'
, @params = N'@speedmodel varbinary(max)'
, @speedmodel = @speedmodel
WITH RESULT SETS (([new_speed] INT, [predicted_distance] INT))

```

- SELECT 문을 사용하여 테이블에서 단일 모델을 가져오고 입력 매개 변수로 전달합니다.
- 테이블에서 모델을 검색한 후 모델에 `unserialize` 함수를 호출합니다.

팁

실시간 점수 매기기를 지원하는 RevoScaleR의 `serialization` 함수도 확인해 보세요.

- 모델에 필요한 인수들로 `rxPredict` 함수에 적용하고 새 입력 데이터를 제공합니다.
- 예제에서 추가된 `str` 함수는 테스트 단계에 R에서 반환되는 데이터의 스키마를 확인하기 위한 용도입니다. 나중에 제거할 수 있습니다.
- R 스크립트에 사용된 열 이름은 저장 프로시저 출력으로 반드시 전달되지는 않습니다. 새 열 이름을 정의하는 WITH RESULTS 절을 사용했습니다.

결과

RESULTS		
	new_speed	predicted_dista...
1	40	140
2	50	180
3	60	219
4	70	258
5	80	298
6	90	337
7	100	376

MESSAGES	
[3:21:38 PM]	Started executing query at <a href="#">Line 71</a> (7 rows affected) STDOUT message(s) from external script: Rows Read: 7, Total Rows Processed: 7, Total Chunk Time: 0.001 seconds 'data.frame': 7 obs. of 1 variable: \$ distance_Pred: num 140 179 218 258 297 ... Total execution time: 00:00:00.475

## 병렬로 평가 수행

작은 데이터 세트에서는 예측이 상당히 빠르게 반환됩니다. 하지만 많은 예측을 매우 빠르게 해야한다고 가정해 보겠습니다. SQL Server 에서 처리 속도를 높이는 방법은 여러가지가 있으며, 작업을 병렬로 처리할 수 있다면 더 많이 있습니다. 특히 채점(scoring)을 하기 위해서는 `sp_execute_external_script`에 `@parallel` 매개변수를 추가하고 값을 **1**로 설정합니다.

수십만 개의 값을 들어있는 자동차 속도에 대한 훨씬 더 큰 테이블을 가지고 있다고 가정해 보겠습니다. 숫자 테이블을 생성하도록 도와주는 커뮤니티의 많은 샘플 T-SQL 스크립트가 있으므로 여기서는 그러한 스크립트를 재현하지

않겠습니다. 많은 정수가 포함된 열이 있고 모델에 `speed`에 대한 입력으로 이 열을 사용하는 경우를 고려해 보겠습니다.

이렇게 하려면 동일한 예측 쿼리를 실행하되 큰 데이터 세트로 대체하고 `parallel = 1` 인수를 추가합니다.

SQL 복사

```
DECLARE @speedmodel varbinary(max) = (select model from
[dbo].[stopping_distance_models] where model_name = 'default
model');

EXEC sp_execute_external_script

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

        current_model <- unserialize(as.raw(speedmodel));
        new <- data.frame(NewCarData);
        predicted.distance <- rxPredict(current_model, new);
        OutputDataSet <- cbind(new, ceiling(predicted.distance));
        '

    , @input_data_1 = N' SELECT [speed] FROM
[dbo].[HugeTableofCarSpeeds] '
    , @input_data_1_name = N'NewCarData'
    , @parallel = 1
    , @params = N'@speedmodel varbinary(max)'
    , @speedmodel = @speedmodel

WITH RESULT SETS (([new_speed] INT, [predicted_distance] INT))
```

- 병렬 실행은 일반적으로 매우 큰 데이터로 작업할 때만 이득을 제공합니다. SQL 데이터베이스 엔진이 병렬 실행이 불 필요하다고 판단할 수도 있습니다. 게다가 데이터를 가져오는 SQL 쿼리가 병렬 쿼리 계획을 생성할 수 있어야 합니다.

- 병렬 실행을 위해 옵션을 사용할 때 반드시 WITH RESULT SETS 절을 사용해서 먼저 출력 결과 스키마를 지정해야 합니다. 이렇게 함으로써 SQL Server 에서 알 수 없는 스키마가 포함될 수 있는 다중 병렬 데이터 세트의 결과를 집계할 수 있도록 허용합니다.
- 채점(scoring) 대신 학습을 시키는 경우엔 이 매개변수가 효과가 없을 수도 있습니다. 모델 유형에 따라 모델 생성 시 요약을 만들기 전에 모든 행을 읽어야 할 수 있습니다.
- 모델을 학습할 때 병렬 처리 이득을 취하기 위해서 **RevoScaleR** 알고리즘 중의 하나를 사용하기를 권장합니다. 이 알고리즘은 `sp_execute_external_script` 에 `@parallel =1` 을 지정하지 않더라도 자동으로 분산 처리하도록 설계되었습니다. RevoScaleR 알고리즘의 최적 성능을 얻기 위한 방법에 관한 지침으로 [분산 및 Microsoft R 에서 ScaleR 와 병렬 컴퓨팅](#)을 참조하십시오.

## 모델의 R 플롯 만들기

SQL Server Management Studio 를 비롯한 많은 클라이언트는

`sp_execute_external_script` 를 사용하여 생성된 그림을 직접 표시할 수 없습니다.

대신, R 플롯을 생성하기 위한 일반적인 프로세스는 R 코드의 일부로 플롯을 생성한 다음 이미지를 파일에 작성하는 것입니다.

또는 이미지를 표시할 수 있는 응용 프로그램에 직렬화된 이진 plot 개체를 반환할 수 있습니다.

다음 예제에서는 R 에 기본적으로 포함된 플로팅 함수를 사용하여 간단한 그래픽을 만드는 방법을 보여 줍니다. 이미지는 지정된 파일로 출력되며 더불어 저장 프로시저를 통한 SQL 변수의 출력됩니다.

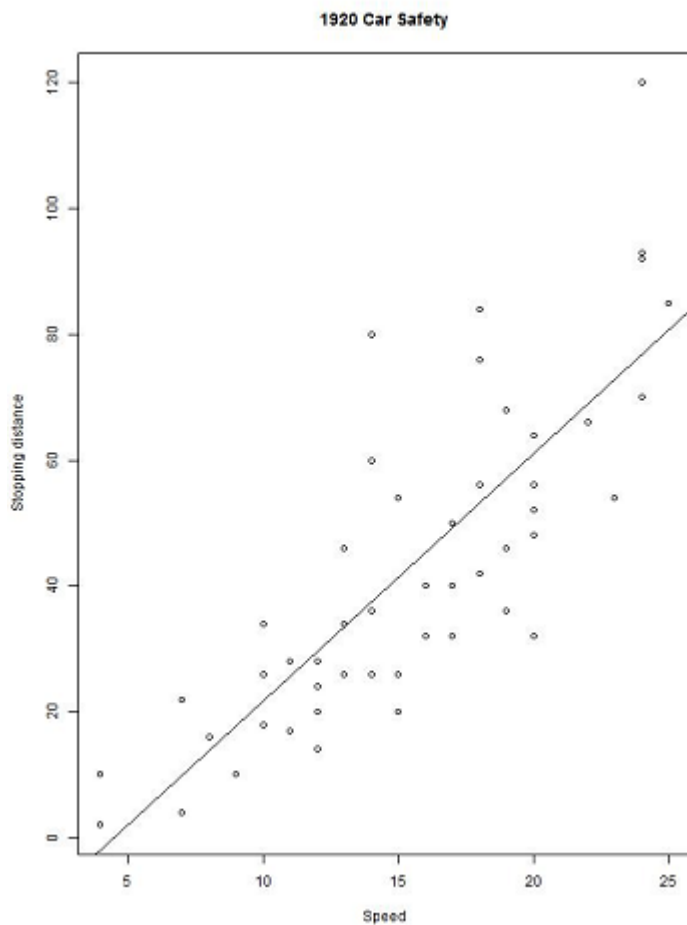
```

EXECUTE sp_execute_external_script
@language = N'R'
, @script = N'
    imageDir <- 'C:\\temp\\plots';
    image_filename = tempfile(pattern = "plot_", tmpdir = imageDir,
fileext = ".jpg")
    print(image_filename);
    jpeg(filename=image_filename, width=600, height = 800);
    print(plot(distance~speed, data=InputDataSet, xlab="Speed",
ylab="Stopping distance", main = "1920 Car Safety"));
    abline(lm(distance~speed, data = InputDataSet));
    dev.off();
    OutputDataSet <- data.frame(data=readBin(file(image_filename,
"rb"), what=raw(), n=1e6));
    '
, @input_data_1 = N'SELECT speed, distance from [dbo].[CarSpeed]'
WITH RESULT SETS ((plot varbinary(max)));

```

- `tempfile` 함수는 파일 이름으로 사용할 수 있는 문자열을 반환합니다, 파일은 아직 생성되지 않습니다.
- `tempfile` 에 인수로 접두사 및 파일 확장명과 디렉터리를 지정할 수 있습니다. 완성된 파일 이름과 경로 확인하기 위해 `str()`를 사용해서 메시지로 출력합니다.
- `jpeg` 함수는 지정된 매개 변수로 R 장치를 만듭니다.
- 플롯을 만든 후에 더 많은 시각적 기능을 추가할 수 있습니다. 이번 경우엔 `abline` 을 사용해서 회귀선을 추가합니다.
- 플롯 기능 추가를 완료하면 `dev.off()` 함수를 사용하여 그래픽 장치를 닫아야 합니다.
- `readBin` 함수는 읽을 파일, 파일 형식 사양 및 레코드 수를 가집니다. `rb***` 키워드는 파일이 텍스트가 아닌 이진임을 나타냅니다.

## 결과



R 용의 몇 가지 좋은 그래픽 패키지를 사용하여 더 정교한 플롯을 수행하고 싶다면 다음 기사들을 권장합니다. 둘 모두 인기있는 **ggplot2** 패키지가 필요합니다.

- [Loan Classification using SQL Server 2016 R Services](#)(SQL Server 2016 R Services 를 사용한 대출 분류): 보험 데이터를 기반으로 하는 전체 과정 연습 시나리오입니다. **reshape** 패키지가 필요합니다.
- [R 을 사용하여 그래프 및 플롯 만들기](#)

## 결론



R 과 SQL Server 를 통합하면 고성능 데이터 처리 및 빠른 R 분석을 위해 R 및 관계형 데이터베이스의 최고 기능을 활용하여 R 솔루션을 대규모로 더 쉽게 배포할 수 있습니다.

더 많은 R 예제 용 추가 리소스:

- [SQL Server R 자습서](#)

Microsoft 데이터 과학 및 R 서비스 개발 팀에서 만든 전체 과정 연습 시나리오를 통해 SQL Server 와 R 을 사용한 솔루션에 대해서 계속 학습합니다.

- [SQL Server Python 자습서](#)

SQL Server 2017 은 Python 언어와 함께 원격 계산 컨텍스트 및 확장 가능한 알고리즘을 사용합니다.

- [Microsoft R 용 자습서 및 예제 데이터](#)

새 RevoScaleR 패키지를 사용하여 모델을 만들고 데이터를 변환하는 방법을 배웁니다.

- [MicrosoftML 시작](#)

Microsoft Research 로부터 신속하고 확장 가능한 Machine Learning 알고리즘을 배웁니다.