데이터 과학 심층 분석: SQL Server 에서 RevoScaleR 패키지 사용하기

원문:https://docs.microsoft.com/ko-kr/sql/advanced-analytics/tutorials/deepdive-data-science-deep-dive-using-the-revoscaler-packages

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

이 자습서에는 고성능 빅 데이터 분석을 위한 계산 컨텍스트로 서버를 사용함으로써 R Services(In-Database)에서 제공하는 향상된 R 패키지를 사용하여 SQL Server 데이터를 작업하고 확장 가능한 R 솔루션을 만드는 방법을 보여줍니다.

로컬 및 원격 계산 컨텍스트 간에 데이터를 이동 하는 원격 계산 컨텍스트를 만들고 원격 SQL Server 에서 R 코드를 실행하는 방법을 학습합니다. 또한 로컬 및 원격 서버에서 데이터를 분석 하고 표시하는 방법과 모델을 만들고 배포하는 방법을 학습합니다.

참고

이 자습서에서는 Windows 의 SQL Server 데이터를 사용하여 작업하고 데이터베이스 내 계산 컨텍스트를 사용합니다. Teradata, Linux, Hadoop 과 같은 다른 환경에서 R 를 사용하려면 다음 Microsoft R Server 자습서를 참조합니다.

- R 서버 sparklyr 와 함께 사용
- Explore R and ScaleR in 25 Functions(25 개 함수에서 R 및 ScaleR 알아보기)
- Hadoop MapReduce 에 ScaleR 시작

개요

RevoScaleR 패키지의 유연성과 처리 능력을 경험하기위해 이 자습서에서 데이터의 이동과 계산 컨텍스트 전환을 자주 합니다. 설명을 위해 이 자습서의 일부 작업은 다음과 같습니다:

- 데이터는 초기에 CSV 파일 또는 XDF 파일에서 가져온 것입니다. RevoScaleR 패키지에서 함수를 사용해서 SQL Server 로 데이터를 가져옵니다.
- SQL Server 계산 컨텍스트를 사용해서 모델 학습 및 채점을 수행합니다.
- RevoScaleR 함수를 사용해서 채점 결과를 저장하는 SQL Server 테이블을 만듭니다.
- 서버와 로컬 계산 컨텍스트에서 플롯을 만듭니다.
- SQL Server 인스턴스에서 R 를 실행하는, SQL Server 데이터베이스 내 데이터에 대한 모델을 학습시킵니다.
- 데이터의 하위 세트를 추출하고 로컬 워크스테이션에서 분석 작업에 재사용하도록 XDF 파일에 저장합니다.
- SQL Server 데이터베이스에 ODBC 연결을 열어 채점을 위하 새 데이터를 가져옵니다. 채점은 로컬 워크스테이션에서 수행합니다.
- 사용자 지정 R 함수를 만들고 서버 계산 컨텍스트에서 실행하여 시뮬레이션을 수행합니다.

필요한 시간 및 항목 목록

이 자습서는 설치를 제외하고 약 75 분 정도 소요됩니다.

1. R 을 사용하여 SQL Server 데이터 작업하기
2. RxSqlServerData 를 사용하여 SQL Server 데이터 개체 만들기
3. SQL Server 데이터 쿼리 및 수정하기
4. 계산 컨텍스트 정의 및 사용하기
5. R 스크립트 만들기 및 실행하기
6. R 을 사용하여 SQL Server 데이터 시각화하기

7. R 모델 만들기
8. 새 데이터 채점하기
9. R 을 사용하여 데이터 변환하기
10. rxImport 를 사용하여 메모리에 데이터 로드하기
11. rxDataStep 를 사용하여 새 SQL Server 테이블 만들기
12. rxDataStep 을 사용하여 청크 단위로 분석하기
13. 로컬 계산 컨텍스트에서 데이터 분석하기
14. XDF 파일을 사용하여 SQL Server 에서 데이터 이동하기
15. 간단한 시뮬레이션 만들기

대상 사용자

이 자습서는 모델 작성과 요약과 같은 데이터 과학 작업을 하는 데이터 과학자 또는 R 에 이미 어느 정도 친숙한 사람들을 대상으로 합니다. 그러나 모든 코드가 제공되므로 R 을 처음 사용하는 경우에도 필요한 서버 및 클라이언트 환경이 있다면 코드를 실행하고 따라할 수 있습니다.

또한 Transact-SQL 구문에 익숙하며 다음과 같은 도구를 사용해서 SQL Server 데이터베이스를 액세스하는 방법을 알고 있어야 합니다:

- SQL Server Management Studio
- Visual Studio 에서 데이터베이스 도구
- 무료 Visual Studio Code 용 mssql extension

팁

중단한 위치부터 다시 시작하기 쉽도록 다음 단원으로 넘어가기 전에 R 작업 영역을 저장하세요.

사전 요구 사항

- **R 을 지원하는 SQL Server**

  SQL Server 2016 을 설치하고 R Services (In-database)를 사용하도록 설정합니다. 또는 SQL Server 2017 을 설치하고 Machine Learning Services 를 사용하도록 설정하고 R 언어를 선택합니다.

- **데이터베이스 사용 권한**

  모델 학습에 사용되는 쿼리를 실행하려면 데이터가 저장된 데이터베이스에 대한 **db_datareader** 권한이 있어야 합니다. R 을 실행하려면 사용자에게 EXECUTE ANY EXTERNAL SCRIPT 권한이 있어야 합니다.

- **데이터 과학 개발 컴퓨터**

  R 개발 환경으로 사용되는 컴퓨터에는 RevoScaleR 패키지 및 관련된 공급자를 설치해야 합니다. 가장 쉬운 방법은 Microsoft R Client, Microsoft R Server (독립 실행형) 또는 Machine Learning Server (독립 실행형)를 설치하는 것입니다.

  참고

  다른 버전의 Revolution R Enterprise 또는 Revolution R Open 은 지원되지 않습니다.

  RevoScaleR 함수만 원격 계산 컨텍스트를 사용할 수 있으므로 이 자습서에서는 R 의 오픈 소스 배포를 사용할 수 없습니다.

- **추가 R 패키지**

  이 자습서에서는 다음 패키지를 설치합니다: **dplyr**, **ggplot2**, **ggthemes**, **reshape2**, 및 **e1071**. 설명서는 자습서의 일부로 제공됩니다.

두 위치에 모든 패키지를 설치해야 함: R 솔루션을 개발하는 컴퓨터와 R 스크립트를 실행하는 SQL Server 컴퓨터. 서버 컴퓨터에 패키지를 설치할 수 있는 권한이 없는 경우 관리자에게 요청합니다.

사용자 라이브러리에 패키지를 설치하지 마세요. 패키지는 SQL Server 인스턴스에서 사용되는 R 패키지 라이브러리에 설치되어야 합니다.

자세한 내용은 데이터 과학 연습에 대한 필수 구성 요소를 참조합니다.

다음 단계

R 을 사용하여 SQL Server 데이터 작업

R를 사용하여 SQL Server 데이터 작업(SQL과 R 심층 분석)

이 단원에서는 환경을 설정하고 모델 학습에 필요한 데이터를 추가하고 데이터에 대한 간략한 요약을 실행합니다. 절차의 일부로 다음 작업을 완료해야 합니다.

- 두 개의 R 모델을 학습하고 채점하기 위한 데이터를 저장할 새 데이터베이스를 만듭니다.
- 워크스테이션과 SQL Server 컴퓨터 간에 통신할 때 사용할 계정(Windows 사용자 또는 SQL 로그인)을 만듭니다.
- SQL Server 데이터와 데이터베이스 개체로 작업하기 위한 데이터 원본을 R 에서 만듭니다.
- R 데이터 원본을 사용하여 데이터를 SQL Server 에 로드합니다.
- R 을 사용하여 변수 목록을 가져오고 SQL Server 테이블의 메타데이터를 수정합니다.
- R 코드의 원격 실행이 가능하도록 계산 컨텍스트를 만듭니다.
- (선택 사항) 원격 계산 컨텍스트에서 추적을 사용합니다.

데이터베이스와 사용자 만들기

이 연습에서 SQL Server 2017 에 새 데이터베이스를 만들고, 데이터를 읽고 쓸 수 있으며 R 스크립트를 실행할 수 있는 사용 권한을 가진 SQL 로그인을 추가합니다.

참고

데이터 읽기만 한다면 R 스크립트를 실행하는 계정에는 지정된 데이터베이스에서 SELECT 사용 권한(**db_datareader** 역할)이 필요합니다. 그러나 이 자습서에서는 데이터베이스를 준비하고 채점 결과를 저장하기 위한 테이블을 만들려면 DDL 관리 권한이 있어야 합니다.

또한 데이터베이스 소유자가 아닌 경우 R 스크립트를 실행하려면 EXECUTE ANY EXTERNAL SCRIPT 사용 권한이 필요 합니다.

1. SQL Server Management Studio 에서 R Services(In-Database) 을 사용할 수 있는 인스턴스를 선택하고 **데이터베이스**를 마우스 오른쪽 단추로 클릭한 다음 **새 데이터베이스**를 선택합니다.
2. 새 데이터베이스의 이름을 입력합니다. 원하는 이름을 임의로 사용할 수 있지만 이 연습의 모든 Transact-SQL 스크립트 및 R 스크립트를 적절하게 편집해야 합니다.

   팁

   업데이트된 데이터베이스 이름을 보려면 **데이터베이스**를 마우스 오른쪽 단추로 클릭하고 **새로 고침**을 선택합니다.

3. **새 쿼리**를 클릭하고 데이터베이스 컨텍스트를 master 데이터베이스로 변경합니다.
4. 새 **쿼리** 창에서 다음 명령을 실행하여 사용자 계정을 만들고 이 자습서에 사용되는 데이터베이스에 할당합니다. 필요한 경우 데이터베이스 이름을 변경해야 합니다.

Windows 사용자

SQL 복사

```
 -- Create server user based on Windows account

USE master
```

```
GO

CREATE LOGIN [<DOMAIN>\<user_name>] FROM WINDOWS WITH
DEFAULT_DATABASE=[DeepDive]



 --Add the new user to tutorial database

USE [DeepDive]

GO

CREATE USER [<user_name>] FOR LOGIN [<DOMAIN>\<user_name>] WITH
DEFAULT_SCHEMA=[db_datareader]
```

SQL 로그인

SQL 복사

```
-- Create new SQL login

USE master

GO

CREATE LOGIN DDUser01 WITH PASSWORD='<type password here>',
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF;



-- Add the new SQL login to tutorial database

USE [DeepDive]

GO

CREATE USER [DDUser01] FOR LOGIN [DDUser01] WITH
DEFAULT_SCHEMA=[db_datareader]
```

1. 사용자가 만들어졌는지 확인하려면 새 데이터베이스를 선택하고 **보안** 및 **사용자**를 차례로 확장합니다.

문제 해결

이번 섹션에서는 데이터베이스를 설정하는 과정에서 발생할 수 있는 몇 가지 일반적인 문제를 보여 줍니다.

- **데이터베이스 연결을 확인하고 SQL 쿼리를 검사하려면 어떻게 하나요?**

  서버를 사용하여 R 코드를 실행하기 전에 R 개발 환경에서 데이터베이스에 연결할 수 있는지 확인하는 것이 좋습니다. Visual Studio 의 서버 탐색기 및 SQL Server Management Studio 는 둘 다 강력한 데이터베이스 연결 및 관리 기능을 갖춘 무료 도구입니다.

  추가 데이터베이스 관리 도구를 설치하지 않으려는 경우 제어판의 ODBC 데이터 원본 관리자를 사용하여 SQL Server 인스턴스에 대한 테스트 연결을 만들 수 있습니다. 데이터베이스가 올바르게 구성되고 올바른 사용자 이름 및 암호를 입력한 경우 방금 만든 데이터베이스가 표시되며 기본 데이터베이스로 선택할 수 있습니다.

  데이터베이스에 연결할 수 없는 경우 서버에 대해 원격 연결을 사용할 수 있고 명명된 파이프 프로토콜이 사용하도록 설정되었는지 확인합니다. 이 문서에서 추가 문제 해결 팁을 제공: SQL Server 데이터베이스 엔진에 연결 문제를 해결합니다.

- **내 테이블 이름 앞에 datareader 가 붙는 것은 무엇 때문인가요?**

  이 사용자에 대한 기본 스키마로 **db_datareader** 를 지정하는 경우, 이 사용자가 만든 모든 테이블 및 기타 새 개체들의 *스키마* 이름에 사용됩니다. 스키마는 개체를 구성하기 위해 데이터베이스에 추가할 수 있는 폴더와 비슷합니다. 또한 스키마는 데이터베이스 내에서 사용자 권한을 정의합니다.

스키마가 특정 사용자 이름과 연관되어 있으면 사용자는 *스키마 소유자*입니다. 개체를 만들 때, 개체를 다른 스키마에서 작성하도록 특별히 요구하지 않는 한, 항상 자신의 스키마에서 만듭니다.

예를 들어 이름 TestData 로 테이블을 만들 경우, 기본 스키마가 **db_datareader** 라면, 해당 테이블은 <database_name>.db_datareader. TestData 이름으로 만들어집니다.

이러한 이유로 테이블이 서로 다른 스키마에 속하기만 하면 한 데이터베이스에 같은 이름의 테이블이 여러 개 포함될 수 있습니다.

테이블을 찾을 때 스키마를 지정하지 않으면 데이터베이스 서버는 사용자가 소유한 스키마를 찾습니다. 따라서 로그인과 연관된 스키마의 테이블에 액세스할 때는 스키마 이름을 지정하지 않아도 됩니다.

- **DDL 권한이 없습니다. 그래도 자습서를 실행할 수 있나요?**

  예, 그러나 누군가에게 데이터를 SQL Server 테이블에 미리 로드하도록 요청하고 새 테이블을 만드는 섹션을 건너뛰어야 합니다. DDL 권한이 필요한 함수는 자습에서 호출됩니다.

  또한 EXECUTE ANY EXTERNAL SCRIPT 권한을 달라고 관리자에게 요청합니다. 원격 혹은 `sp_execute_external_script` 을 사용해서 R 스크립트를 실행하는데 필요합니다.

다음 단계

RxSqlServerData 를 사용하여 SQL Server 데이터 개체 만들기

개요

데이터 과학 심층 분석: RevoScaleR 패키지 사용

RxSqlServerData를 사용하여 SQL Server 데이터 개체 만들기(SQL과 R 심층 분석)

지금까지 SQL Server 데이터베이스를 만들고 데이터 작업에 필요한 필요한 권한을 얻었습니다. 이 단계에서는 R 에서 데이터 작업을 할 수 있도록 일부 개체를 만듭니다.

SQL Server 데이터 개체 만들기

이 단계에서는 **RevoScaleR** 패키지의 함수를 사용해서 두 개의 테이블을 만들고 데이터를 채웁니다. 한 테이블은 모델 학습에 사용하고 다른 테이블은 채점에 사용합니다. 두 테이블 모두 시뮬레이션된 신용 카드 사기 데이터를 포함합니다.

원격 SQL Server 컴퓨터에서 테이블을 만들기 위해 **RxSqlServerData** 함수를 호출합니다.

팁

R Tools for Visual Studio 를 사용하는 경우, 도구 모음에서 **R Tools** 를 선택하고 **Windows** 를 클릭하여 R 변수 디버깅 및 보기에 대한 옵션을 표시하세요.

학습 데이터 테이블 만들기

1. R 변수에 데이터베이스 연결 문자열을 저장합니다. 다음은 SQL Server 에 대한 유효한 ODBC 연결 문자열의 두 가지 예제: SQL 로그인을 사용하는 것 하나와 Windows 통합 인증 한 개입니다.

   SQL 로그인

   R 복사

   ```
   sqlConnString <- "Driver=SQL Server;Server=instance_name;
   Database=DeepDive;Uid=user_name;Pwd=password"
   ```

   Windows 인증

   R 복사

   ```
   sqlConnString <- "Driver=SQL
   Server;Server=instance_name;Database=DeepDive;Trusted_Connection=Tru
   e"
   ```

   인스턴스 이름, 데이터베이스 이름, 사용자 이름 및 암호를 적절하게 수정해야 합니다.

2. 만들려는 테이블의 이름을 지정하고 R 변수에 저장합니다.

   R 복사

   ```
   sqlFraudTable <- "ccFraudSmall"
   ```

   인스턴스 및 데이터베이스 이름이 연결 문자열의 일부로 이미 지정되었음으로 두 변수를 결합할 경우 새 테이블의 *정규화된* 이름은 *instance.database.schema.ccFraudSmall* 이 됩니다.

3. 데이터 원본 개체를 인스턴스화하기 전에 추가 매개 변수 *rowsPerRead* 를 지정하는 줄을 추가합니다. *rowsPerRead* 매개 변수는 각 배치에서 읽는 데이터 행 수를 제어합니다.

R 복사

```
sqlRowsPerRead = 5000
```

이 매개 변수는 선택 사항이지만 메모리 사용 및 효율적인 계산을 처리하는 데 중요합니다. R Services(In-Database)의 향상된 분석 기능은 대부분 데이터를 청크로 처리하고 중간 결과를 저장하며 모든 데이터를 읽은 후에 최종 계산을 반환합니다.

이 매개 변수의 값이 너무 크면 그만큼 큰 데이터 청크를 효율적으로 처리할 수 있는 메모리가 없어서 데이터 액세스 속도가 느려질 수 있습니다. 일부 시스템에서는 *rowsPerRead* 값이 너무 작아서 성능이 느려질 수 있습니다. 따라서 대량 데이터 세트로 작업하는 경우엔 시스템에서 이 설정값을 시험하는 것이 좋습니다.

이 연습에는 SQL Server 인스턴스에서 정의한 기본 일괄 처리 크기를 사용 하여 각 청크의 행 수를 제어합니다. 해당 값을 `sqlRowsPerRead` 변수에 저장 합니다.

4. 마지막으로, 새 데이터 원본 개체에 대한 변수를 정의하고 이전에 정의한 인수를 RxSqlServerData 생성자에 전달합니다. 이렇게 하면 데이터 원본 개체가 만들어지기만 하고 채워지지는 않습니다.

R 복사

```
sqlFraudDS <- RxSqlServerData(connectionString = sqlConnString,
  table = sqlFraudTable,
  rowsPerRead = sqlRowsPerRead)
```

채점 데이터 테이블을 만들려면

동일한 단계와 절차를 사용해서 채점 데이터를 보관하는 테이블을 만듭니다.

1. 새 R 변수 *sqlScoreTable* 을 만들어 채점에 사용되는 테이블의 이름을 저장합니다.

```
R 복사
```

```
sqlScoreTable <- "ccFraudScoreSmall"
```

2. 이 변수를 RxSqlServerData 함수의 인수로 제공해서 두 번째 데이터 원본 개체인 *sqlScoreDS* 를 정의합니다.

```
R 복사
```

```
sqlScoreDS \<- RxSqlServerData(connectionString = sqlConnString,
  table = sqlScoreTable, rowsPerRead = sqlRowsPerRead)
```

R 작업 영역에서 이미 연결 문자열과 기타 매개 변수를 변수로 정의했음으로 다른 테이블, 뷰 또는 쿼리에 대한 새 데이터 원본을 쉽게 만들 수 있습니다.

참고

이 함수는 쿼리를 기반으로 데이터 원본 용과 전체 테이블을 기반으로 데이터 원본을 정의하는데 있어서 서로 다른 인수를 사용합니다. 이는 SQL Server 데이터베이스 엔진이 쿼리를 다르게 준비해야하기 때문입니다. 이 자습서의 뒷부분에서 SQL 쿼리를 기반으로 데이터 원본 개체를 만드는 방법을 배웁니다.

R 을 사용하여 SQL 테이블에 데이터를 로드하기

이제 SQL Server 테이블을 만들었으므로 적절한 **Rx** 함수를 사용하여 해당 테이블에 데이터를 로드할 수 있습니다.

RevoScaleR 패키지에는 다양한 데이터 원본을 지원하는 함수가 있습니다. 텍스트 데이터의 경우 RxTextData 를 사용해서 데이터 원본 개체를 생성합니다. Hadoop 데이터, ODBC 데이터 등에서 데이터 원본 개체를 만들기 위한 추가 함수들이 있습니다.

참고

이 섹션을 수행하려면 데이터베이스에 **DDL 실행** 사용 권한이 있어야 합니다.

학습 테이블에 데이터 로드하기

1. R 변수 *ccFraudCsv* 를 만들고 샘플 데이터를 포함하는 CSV 파일의 경로를 변수에 할당합니다.

R 복사

```
ccFraudCsv <- file.path(rxGetOption("sampleDataDir"),
"ccFraudSmall.csv")
```

RevoScaleR 의 rxOptions 에 연관된 GET 메소드인 **rxGetOption** 에 대한 호출을 확인하십시오. 이 유틸리티를 사용하여 기본 공유 디렉터리 또는 계산에 사용할 프로세서(코어) 수와 같은 로컬 및 원격 계산 컨텍스트에 관련된 옵션을 설정하고 나열합니다.

이 특정 호출은 코드를 실행하는 위치에 관계없이 올바른 라이브러리에서 샘플을 가져옵니다. 예를 들어 SQL Server 및 개발 컴퓨터에서 이 함수를 실행해 보고 경로가 어떻게 다른지 확인하세요.

2. 새 데이터를 저장할 변수를 정의하고 **RxTextData** 함수를 사용하여 텍스트 데이터 원본을 지정합니다.

R 복사

```
inTextData <- RxTextData(file = ccFraudCsv,      colClasses = c(
    "custID" = "integer", "gender" = "integer", "state" = "integer",
    "cardholder" = "integer", "balance" = "integer",
    "numTrans" = "integer",
    "numIntlTrans" = "integer", "creditLine" = "integer",
    "fraudRisk" = "integer"))
```

colClasses 인수가 중요합니다. 이 인수를 사용하여 텍스트 파일에서 로드된 데이터의 각 열에 할당할 데이터 형식을 나타냅니다. 이 예제에서는 정수로 처리되는 명명된 열을 제외하고 모든 열이 텍스트로 처리됩니다.

3. 이 시점에서 잠시 멈추고 SQL Server Management Studio 에서 데이터베이스를 볼 수 있습니다. 데이터베이스의 테이블 목록을 새로 고칩니다.

   R 데이터 개체가 로컬 작업 공간에 생성되었지만 SQL Server 데이터베이스에 테이블이 생성되지 않았음을 알 수 있습니다. 또한 텍스트 파일에서 R 변수에 데이터가 로드 되지 않았습니다.

4. 이제 rxDataStep 함수를 호출하여 SQL Server 테이블에 데이터를 삽입합니다.

R 복사

```
rxDataStep(inData = inTextData, outFile = sqlFraudDS, overwrite =
TRUE)
```

연결 문자열에 문제가 없다고 가정하면 잠시 후 다음과 같은 결과가 표시됩니다.

Total Rows written: 10000, Total time: 0.466

Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.577 seconds

5. SQL Server Management Studio 를 사용하여 테이블 목록을 새로 고칩니다. 각 변수가 올바른 데이터 형식을 갖고 성공적으로 가져왔는지 확인하려면 SQL Server Management Studio 에서 테이블을 마우스 오른쪽 단추로 클릭하고 **상위 1000 개 행 선택**을 선택합니다.

채점 테이블에 데이터 로드하기

1. 채점용으로 사용되는 데이터 세트를 데이터베이스에 로드하기 위해 단계를 반복합니다.

   먼저 원본 파일의 경로를 제공합니다.

   R 복사

```
ccScoreCsv <- file.path(rxGetOption("sampleDataDir"),
"ccFraudScoreSmall.csv")
```

2. **RxTextData** 함수를 사용하여 데이터를 가져와 *inTextData* 변수에 저장합니다.

   R 복사

```
inTextData <- RxTextData(file = ccScoreCsv,      colClasses = c(
    "custID" = "integer", "gender" = "integer", "state" = "integer",
    "cardholder" = "integer", "balance" = "integer",
    "numTrans" = "integer",
    "numIntlTrans" = "integer", "creditLine" = "integer"))
```

3. **rxDataStep** 함수를 호출하여 현재 테이블을 새 스키마와 데이터로 덮어씁니다.

R 복사

```
rxDataStep(inData = inTextData, sqlScoreDS, overwrite = TRUE)
```

- *inData* 인수는 사용할 데이터 원본을 정의합니다.
- *outFile* 인수는 데이터를 저장할 SQL Server 의 테이블을 지정합니다.
- 테이블이 이미 존재하고 *overwrite* 옵션을 사용하지 않는 경우 결과는 잘림없이 입력됩니다.

연결에 성공하면 완료를 나타내는 메시지와 테이블에 데이터를 기록하는 데 필요한 시간이 표시됩니다.

Total Rows written: 10000, Total time: 0.384

Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.456 seconds

RxDataStep 에 대한 추가 정보

rxDataStep 은 R 데이터 프레임에서 여러 변환을 수행할 수 있는 강력한 함수입니다. 또한 rxDataStep 을 사용하여 대상에서 필요로 하는 표현으로 데이터를 변환할 수 있습니다: 이 경우 SQL Server.

선택적으로 **rxDataStep** 에 대한 인수에서 R 함수를 사용하여 데이터에 대해 변환을 지정할 수 있습니다. 이러한 작업의 예는 이 자습서의 뒷부분에서 제공됩니다.

다음 단계

SQL Server 데이터 쿼리 및 수정

이전 단계

R 을 사용하여 SQL Server 데이터 작업

SQL Server 데이터 쿼리 및 수정(SQL 과 R 심층 분석)

이제 데이터를 SQL Server 에 로드했음으로 앞서 만든 데이터 원본을 R Services(In-Database)의 R 함수에 대한 인수로 사용하여 변수에 대한 기본 정보를 가져오고 요약 및 히스토그램을 생성합니다.

이 단계에서는 몇 가지 빠른 분석을 수행한 다음 데이터를 강화 하도록 데이터 원본을 다시 사용 합니다.

데이터 쿼리

먼저, 열과 해당 데이터 형식 목록을 가져옵니다.

1. rxGetVarInfo 함수를 사용하여 분석하려는 데이터 원본을 지정합니다.

   RevoScaleR 의 버전을 따라 rxGetVarNames 를 사용할 수도 있습니다.

```
R 복사
```

```
rxGetVarInfo(data = sqlFraudDS)
```

결과

Var 1: custID, Type: integer

Var 2: gender, Type: integer

Var 3: state, Type: integer

Var 4: cardholder, Type: integer

Var 5: balance, Type: integer

Var 6: numTrans, Type: integer

Var 7: numIntlTrans, Type: integer

Var 8: creditLine, Type: integer

Var 9: fraudRisk, Type: integer

메타 데이터 수정

모든 변수가 정수로 저장되었지만 일부 변수는 범주형 데이터를 나타내며 R 에서는 *요인(factor) 변수*라고 합니다. 예를 들어 *state* 열에는 50 개 주와 콜롬비아 특별구에 대한 식별자로 사용되는 숫자가 포함되어 있습니다. 데이터를 더 쉽게 이해하기위해 숫자를 주 약어 목록으로 바꿉니다.

이 단계에서는 약어를 포함하는 문자열 벡터를 만들고 원래 정수 식별자에 이러한 범주 값을 매핑합니다. 그 다음 *colInfo* 인수에 새 변수를 사용하여 이 열을 요인으로 처리하도록 지정합니다. 데이터를 분석하거나 이동시킬 때마다 약어가 사용되며 열은 요인으로 처리됩니다.

열을 요인으로 사용하기 전에 약어에 매핑하면 실제로 성능 또한 향상됩니다. 자세한 내용은 R 및 데이터 최적화를 참조합니다.

1. 먼저 R 변수 *stateAbb* 를 만들고 다음과 같이 추가할 문자열 벡터를 정의합니다.

R 복사

```
stateAbb <- c("AK", "AL", "AR", "AZ", "CA", "CO", "CT", "DC",
    "DE", "FL", "GA", "HI","IA", "ID", "IL", "IN", "KS", "KY", "LA",
```

```
    "MA", "MD", "ME", "MI", "MN", "MO", "MS", "MT", "NB", "NC",
"ND",
    "NH", "NJ", "NM", "NV", "NY", "OH", "OK", "OR", "PA", "RI","SC",
    "SD", "TN", "TX", "UT", "VA", "VT", "WA", "WI", "WV", "WY")
```

2. 다음으로, 범주 수준(주 약어)과 기존 정수 값 간의 매핑을 지정하는 *ccColInfo* 라는 열 정보 개체를 만듭니다.

   이 문은 성별 및 카드 소유자에 대한 요인 변수도 만듭니다.

R 복사

```
ccColInfo <- list(
gender = list(
          type = "factor",
          levels = c("1", "2"),
          newLevels = c("Male", "Female")
          ),
cardholder = list(
              type = "factor",
              levels = c("1", "2"),
              newLevels = c("Principal", "Secondary")
               ),
state = list(
          type = "factor",
          levels = as.character(1:51),
          newLevels = stateAbb
          ),
balance = list(type = "numeric")
```

```
)
```

3. 업데이트된 데이터를 사용하는 SQL Server 데이터 원본을 만들려면 이전과 같이 **RxSqlServerData** 함수를 호출하되, *colInfo* 인수를 추가합니다.

R 복사

```
sqlFraudDS <- RxSqlServerData(connectionString = sqlConnString,
table = sqlFraudTable, colInfo = ccColInfo,
rowsPerRead = sqlRowsPerRead)
```

- *table* 매개변수에 대해 앞에서 만든 데이터 원본을 포함하는 *sqlFraudTable* 변수를 전달합니다.
- *colInfo* 매개변수에 대해 열 데이터 형식 및 요인 수준을 포함하는 *ccColInfo* 변수를 전달합니다.

4. 이제 **rxGetVarInfo** 함수를 사용하여 새 데이터 원본의 변수를 확인할 수 있습니다.

R 복사

```
rxGetVarInfo(data = sqlFraudDS)
```

결과

Var 1: custID, Type: integer

Var 2: gender 2 factor levels: Male Female

Var 3: state 51 factor levels: AK AL AR AZ CA ... VT WA WI WV WY

Var 4: cardholder 2 factor levels: Principal Secondary

Var 5: balance, Type: integer

Var 6: numTrans, Type: integer

Var 7: numIntlTrans, Type: integer

Var 8: creditLine, Type: integer

Var 9: fraudRisk, Type: integer

이제 지정한 변수 3 개(*gender*, *state*, *cardholder*)가 요인으로 처리됩니다.

다음 단계

계산 컨텍스트 정의 및 사용

이전 단계

RxSqlServerData 를 사용하여 SQL Server 데이터 개체 만들기

계산 컨텍스트 사용 및 정의 (SQL과 R 심층 분석)

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

이 단원에서는 RxInSqlServer 함수 SQL Server 에 대 한 계산 컨텍스트를 정의 하고 다음 로컬 컴퓨터 대신 서버에서 복잡 한 계산을 실행할 수 있습니다.

RevoScaleR 은 Hadoop, Spark, 또는 데이터베이스 내에서 R 코드를 실행할 수 있도록 여러 계산 컨텍스트를 지원 합니다. SQL Server 에 대 한 서버를 정의 하고 로컬 컴퓨터와 원격 실행 컨텍스트 간에 연결 및 전달 개체는 데이터베이스를 만드는 작업을 처리 하는 함수입니다.

SQL Server 를 만드는 함수를 사용 하 여 상황에 맞는 다음 정보를 계산 합니다.

- 에 대 한 연결 문자열은 SQL Server 인스턴스
- 출력 처리 방법 지정
- 추적을 사용 하거나 추적 수준을 지정 하는 선택적 인수
- 공유 데이터 디렉터리의 선택적 사양

만들기 및 계산 컨텍스트를 설정 합니다.

1. 계산이 수행 되는 인스턴스에 대 한 연결 문자열을 지정 합니다. 앞에서 만든 연결 문자열을 다시 사용할 수 있습니다. 다른 서버에는 계산을 이동 하거나 일부 작업을 수행 하는 다른 로그인을 사용 하려는 경우 다른 연결 문자열을 만들 수 있습니다.

SQL 로그인 사용

R 복사

```
sqlConnString <- "Driver=SQL Server;Server=<SQL Server instance name>; Database=<database name>;Uid=<SQL user name>;Pwd=<password>"
```

Windows 인증 사용

R 복사

```
sqlConnString <- "Driver=SQL Server;Server=instance_name;Database=DeepDive;Trusted_Connection=True"
```

2. 출력 처리 방법을 지정합니다. 다음 코드에서는 워크스테이션의 R 세션이 항상 R 작업 결과를 기다리지만, 원격 계산의 콘솔 출력을 반환하지 않도록 지정합니다.

R 복사

```
sqlWait <- TRUE

sqlConsoleOutput <- FALSE
```

RxInSqlServer 에 대한 **wait** 인수는 다음 옵션을 지원합니다.

- **TRUE**. 작업 차단으로 구성 되 고 완료 되었거나 실패 될 때까지 반환 하지 않습니다. 자세한 내용은 참조 분산 및 학습 서버 컴퓨터에에서 병렬 컴퓨팅합니다.
- **FALSE**. 작업은 비차단로 구성 및 즉시 반환 하므로 다른 R 코드 실행을 계속할 수 있습니다. 그러나 비차단 모드에서도 작업이 실행되는 동안 SQL Server 와의 클라이언트 연결을 유지해야 합니다.

3. 원격 및 로컬 R 세션에서 공유 사용을 위해 로컬 디렉터리의 위치를 지정할 수는 필요에 따라 SQL Server 컴퓨터 및 해당 계정.

R 복사

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

4. 공유에 대 한 특정 디렉터리를 수동으로 만들 하려는 경우에 다음과 같은 줄을 추가할 수 있습니다.

복사

```
dir.create(sqlShareDir, recursive = TRUE)
```

폴더를 공유 하는 데 현재 사용 중인을 확인 하려면 실행 `rxGetComputeContext()`, 세부 정보에 대 한 현재 계산 컨텍스트를 반환 하는 합니다. 자세한 내용은 ScaleR reference(ScaleR 참조)를 참조하세요.

5. 모든 변수를 준비 하지 제공 하 고에 대 한 인수는 **RxInSqlServer** 를 만드는 생성자는 *컨텍스트 개체를 계산*합니다.

R 복사

```
sqlCompute <- RxInSqlServer(
    connectionString = sqlConnString,
    wait = sqlWait,
    consoleOutput = sqlConsoleOutput)
```

에 대 한 구문 **RxInSqlServer** 의 거의 동일는 **RxSqlServerData** 데이터 원본을 정의 하려면 이전에 사용 하는 함수입니다. 하지만 다음과 같은 점에서 중요한 차이가 있습니다.

- RxSqlServerData 함수를 사용하여 정의한 데이터 원본 개체는 데이터가 저장되는 위치를 지정합니다.

- 함수를 사용 하 여 계산 컨텍스트를 정의 하는 반면, RxInSqlServer 집계 및 기타 계산이 수행 된 것을 표시 합니다.

  계산 컨텍스트 정의는 워크스테이션에서 수행할 수 있는 다른 모든 제네릭 R 계산에 영향을 주지 않으며 데이터의 원본을 변경하지 않습니다. 예를 들어 로컬 텍스트 파일을 데이터 원본으로 정의하지만 계산 컨텍스트를 SQL Server 로 변경하지 않고 SQL Server 컴퓨터에서 데이터에 대한 모든 읽기 및 요약을 수행할 수 있습니다.

계산 컨텍스트에서 추적을 사용 하도록 설정

때로는 작업이 로컬 컨텍스트에서 작동하지만 원격 계산 컨텍스트에서 실행될 때 문제가 발생합니다. 문제를 분석하거나 성능을 모니터링하려는 경우 계산 컨텍스트에서 추적을 사용하도록 설정하여 런타임 문제 해결을 지원할 수 있습니다.

1. 인수를 추가 하지만 동일한 연결 문자열을 사용 하 여 새 계산 컨텍스트를 만들어 *traceEnabled* 및 *traceLevel* 에 **RxInSqlServer** 생성자입니다.

   R 복사

```
sqlComputeTrace <- RxInSqlServer(
    connectionString = sqlConnString,
    #shareDir = sqlShareDir,
    wait = sqlWait,
    consoleOutput = sqlConsoleOutput,
    traceEnabled = TRUE,
    traceLevel = 7)
```

이 예제에서는 *traceLevel* 속성이 7 로 설정되었으며 이는 "모든 추적 정보 표시"를 의미합니다.

2. 계산 컨텍스트를 변경하려면 rxSetComputeContext 함수를 사용하고 이름으로 컨텍스트를 지정합니다.

```
R 복사
```

```
rxSetComputeContext( sqlComputeTrace)
```

참고

이 자습서에 대 한 추적을 설정 하지 않은 계산 컨텍스트를 사용 합니다.

그러나 추적을 사용 하려는 경우 유의 경험 네트워크 연결에 의해 영향을 받을 수 있습니다. 또한 유의 해야 하는 모든 작업에 대 한 추적이 설정 된 옵션에 대 한 성능 테스트 되지 않은 때문에 있습니다.

사용 하는 방법을 알아보려면 다음 단계에서는 서버에서 R 코드를 실행 하거나 로컬 컨텍스트를 계산 합니다.

다음 단계

R 스크립트 만들기 및 실행

이전 단계

SQL Server 데이터 쿼리 및 수정

만들기 및 R 스크립트 (SQL과 R 심층 분석)를 실행 합니다.

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

이제 데이터 원본을 설정하고 하나 이상의 계산 컨텍스트를 설정했으므로 SQL Server 를 사용하여 몇 가지 고성능 R 스크립트를 실행할 준비가 되었습니다. 이 단원 server 계산 컨텍스트를 사용 하 여 몇 가지 일반적인 시스템 작업 학습을 수행 하려면:

- 데이터 시각화 및 몇 가지 요약 통계 생성
- 선형 회귀 모델 만들기
- 로지스틱 회귀 모델 만들기
- 새 데이터 점수 매기기 및 점수 히스토그램 만들기

계산 서버에는 컨텍스트 변경

R 코드를 실행하기 전에 *현재* 또는 *활성* 계산 컨텍스트를 지정해야 합니다.

1. R 을 사용하여 이미 정의한 계산 컨텍스트를 활성화하려면 다음과 같이 **rxSetComputeContext** 함수를 사용합니다.

   R 복사

   ```
   rxSetComputeContext(sqlCompute)
   ```

이 문을 실행 하는 즉시 모든 이후 계산에 진행는 SQL Server 에 지정 된 컴퓨터는 *sqlCompute* 매개 변수입니다.

2. 워크스테이션에서 R 코드를 실행하려는 경우 **local** 키워드를 사용하여 계산 컨텍스트를 다시 로컬 컴퓨터로 전환할 수 있습니다.

```
R 복사
```

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

이 함수에서 지원하는 다른 키워드 목록을 보려면 R 명령줄에서 `help("rxSetComputeContext")` 를 입력합니다.

3. 계산 컨텍스트를 지정하면 변경할 때까지 활성 상태로 유지됩니다. 그러나 원격 서버 컨텍스트에서 실행할 수 *없는* 모든 R 스크립트는 로컬로 실행됩니다.

일부 요약 통계를 계산 합니다.

계산 컨텍스트 작동 방식을 보려면 사용 하 여 일부 요약 통계를 생성을 시도 `sqlFraudDS` 데이터 원본입니다. 기억, 데이터 원본 개체에는 방금; 사용 하는 데이터 정의 계산 컨텍스트를 바뀌지 않습니다.

- 로컬로 요약을 수행 하려면 **rxSetComputeContext** 지정는 *로컬* 키워드입니다.
- SQL Server 컴퓨터에서 같은 계산을 만들려면 앞에서 정의한 SQL 계산 컨텍스트로 전환합니다.

1. 호출 된 rxSummary 함수 및 수식 및 데이터 원본 등의 필수 인수를 전달 하 고 결과 변수에 할당 `sumOut` 합니다.

```
R 복사
```

```
sumOut <- rxSummary(formula = ~gender + balance + numTrans +
numIntlTrans + creditLine, data = sqlFraudDS)
```

R 언어 여러 요약 기능을 제공 하지만 **rxSummary** 다양 한 원격 계산 컨텍스트를 포함 하 여 실행을 지원 SQL Server 합니다. 유사한 기능에 대 한 정보를 참조 하십시오. RevoScaleR 을 사용 하 여 데이터 요약합니다.

2. 처리가 완료 되는 경우의 내용을 인쇄할 수 있습니다는 `sumOut` 콘솔에 변수입니다.

```
R 복사
```

```
sumOut
```

참고

SQL Server 컴퓨터에서 결과가 반환되기 전에 결과를 출력하려고 하지 마세요. 그러지 않으면 오류가 발생할 수 있습니다.

결과

Summary Statistics Results for: ~gender + balance + numTrans +

numIntlTrans + creditLine

Data: sqlFraudDS (RxSqlServerData Data Source)

Number of valid observations: 10000

Name Mean StdDev Min Max ValidObs MissingObs

balance 4075.0318 3926.558714 0 25626 100000

numTrans 29.1061 26.619923 0 100 10000 0 100000

numIntlTrans 4.0868 8.726757 0 60 10000 0 100000

creditLine 9.1856 9.870364 1 75 10000 0 100000

성별에 대 한 범주 수

Number of categories: 2

Number of valid observations: 10000

Number of missing observations: 0

gender Counts

Male 6154

Female 3846

최대 및 최소 값 추가

계산된 요약 통계에 따라 데이터에 대한 몇 가지 유용한 정보를 발견했으며 추가 계산에 사용하기 위해 이 정보를 데이터 원본에 저장하려고 합니다. 예를 들어 히스토그램을 계산 하는 최소 및 최대 값을 사용할 수 있습니다. 이러한 이유로 추가 하 고가 및 저가 값은 **RxSqlServerData** 데이터 원본입니다.

다행히 R Services(In-Database) 범주 비율 데이터에 정수 데이터를 효율적으로 변환할 수 있는 최적화 된 기능을 포함 합니다.

1. 먼저 몇 가지 임시 변수를 설정합니다.

   R 복사

   ```
   sumDF <- sumOut$sDataFrame

   var <- sumDF$Name
   ```

2. 앞에서 만든 `ccColInfo` 변수를 사용하여 데이터 원본의 열을 정의합니다.

또한 일부 새 계산 열을 추가 (`numTrans`, `numIntlTrans`, 및 `creditLine`)을 열 컬렉션입니다.

R 복사

```
ccColInfo <- list(
    gender = list(type = "factor",
      levels = c("1", "2"),
      newLevels = c("Male", "Female")),
    cardholder = list(type = "factor",
      levels = c("1", "2"),
      newLevels = c("Principal", "Secondary")),
    state = list(type = "factor",
      levels = as.character(1:51),
      newLevels = stateAbb),
    balance  = list(type = "numeric"),
    numTrans = list(type = "factor",
      levels = as.character(sumDF[var == "numTrans",
"Min"]:sumDF[var == "numTrans", "Max"])),
    numIntlTrans = list(type = "factor",
        levels = as.character(sumDF[var == "numIntlTrans",
"Min"]:sumDF[var =="numIntlTrans", "Max"])),
    creditLine = list(type = "numeric")
        )
```

3. 업데이트 된 버전을 만들려면 다음 문을 적용 열 컬렉션에 있는 업데이트 것은 SQL Server 앞에서 정의한 데이터 원본.

R 복사

```
sqlFraudDS <- RxSqlServerData(
    connectionString = sqlConnString,
    table = sqlFraudTable,
    colInfo = ccColInfo,
    rowsPerRead = sqlRowsPerRead)
```

`sqlFraudDS` 데이터 원본을 사용 하 여 추가 하는 새 열이 포함 이제 `ccColInfo` 합니다.

수정만 R;에서 데이터 원본 개체에 영향을 줄이 시점에서 새 데이터가 아직 데이터베이스 테이블에 기록 된 했습니다. 그러나에서 수집 된 데이터를 사용할 수는 `sumOut` 변수를 시각화 및 요약을 만듭니다. 다음 단계 계산 컨텍스트를 전환 하는 동안이 작업을 수행 하는 방법에 설명 합니다.

팁

사용 하는 계산 컨텍스트를 잊은 경우 실행 `rxGetComputeContext()`합니다. 반환 값이 "RxLocalSeq 계산 컨텍스트" 로컬 계산 컨텍스트에서 실행을 나타냅니다.

다음 단계

R 을 사용하여 SQL Server 데이터 시각화

이전 단계

계산 컨텍스트 정의 및 사용

R (SQL과 R 심층 분석)를 사용하여 SQL Server 데이터를 시각화 합니다.

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

R Services(In-Database) 의 향상된 패키지에는 확장성과 병렬 처리에 최적화된 여러 함수가 포함되어 있습니다. 일반적으로 이러한 함수 앞에는 **rx** 또는 **Rx** 가 붙습니다.

이 연습에서는 사용는 **rxHistogram** 함수에서 값의 분포를 볼 수는 *creditLine* 성별 열입니다.

RxHistogram 를 사용 하 여 데이터 시각화

1. 다음 R 코드를 사용하여 rxHistogram 함수를 호출하고 수식 및 데이터 원본을 전달합니다. 처음에 이 코드를 로컬로 실행하여 예상 결과 및 걸리는 시간을 확인할 수 있습니다.

R 복사

```
rxHistogram(~creditLine|gender, data = sqlFraudDS,  histType = "Percent")
```

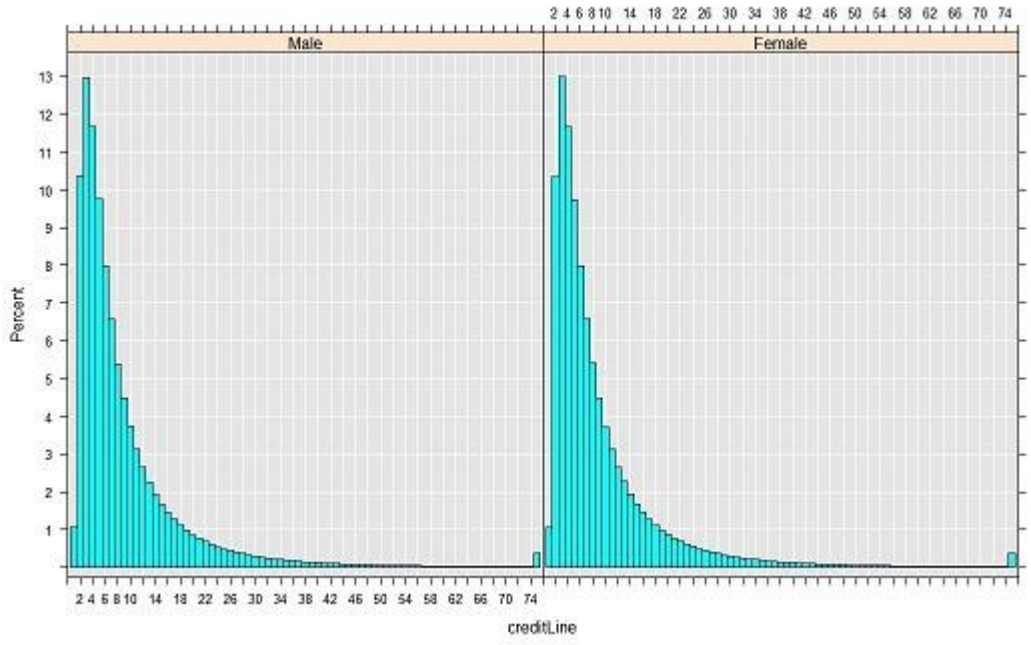
내부적으로 **rxHistogram** 은 RevoScaleR 패키지에 포함된 **rxCube** 함수를 호출합니다. **rxCube** 계산 열 수식에 지정 된 각 변수에 대 한 열이 포함 된 단일 목록 (또는 데이터 프레임)을 출력 합니다.

2. 이제 원격 SQL Server 컴퓨터에 계산 컨텍스트를 설정 하 고 실행 **rxHistogram** 다시 합니다.

R 복사

```
rxSetComputeContext(sqlCompute)
```

3. 같은 데이터 원본을 사용하므로 결과는 정확히 같지만, 계산은 SQL Server 컴퓨터에서 수행됩니다. 그런 다음 결과가 로컬 워크스테이션에 반환되어 그려집니다.



1. 호출할 수도 있습니다는 **rxCube** 함수 및 함수를 그래프에 표시 되는 R 에 결과 전달 합니다. 예를 들어 다음 예제에서는 **rxCube** 를 사용하여 *numTrans* 및 *numIntlTrans* 의 모든 조합에 대해 *fraudRisk* 의 평균을 계산합니다.

R 복사

```
cube1 <- rxCube(fraudRisk~F(numTrans):F(numIntlTrans),  data = sqlFraudDS)
```

그룹 평균을 계산하는 데 사용되는 그룹을 지정하려면 `F()` 표기법을 사용합니다. 이 예제에서는 `F(numTrans):F(numIntlTrans)` 나타냅니다 변수에 정수 `_numTrans` 및 `numIntlTrans` 각 정수 값에 대 한 수준으로 범주 변수로 처리 해야 합니다.

낮은 임계값과 높은 수준의 데이터 원본에 이미 추가 된 때문에 `sqlFraudDS` (사용 하는 `colInfo` 매개 변수), 수준 히스토그램에 자동으로 사용 됩니다.

2. 기본 반환 값의 **rxCube** 는 *rxCube 개체*, 교차 집계를 나타냅니다. 그러나 rxResultsDF 함수를 사용하여 R 의 표준 그리기 함수 중 하나에서 쉽게 사용할 수 있는 데이터 프레임으로 결과를 변환할 수 있습니다.

R 복사

```
cubePlot <- rxResultsDF(cube1)
```

rxCube 함수는 선택적 인수를 포함 *returnDataFrame* = **TRUE**, 하는 직접 결과 데이터 프레임으로 변환 하는 데 사용할 수 없습니다. 예를 들어 다음과 같이 사용할 수 있습니다.

```
print(rxCube(fraudRisk~F(numTrans):F(numIntlTrans), data = sqlFraudDS, returnDataFrame = TRUE))
```

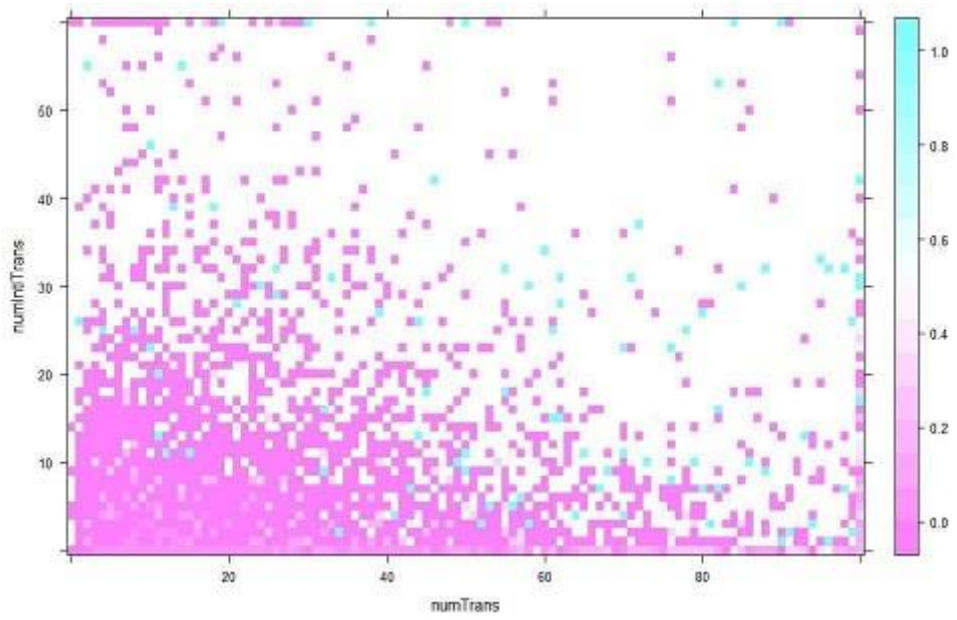
그러나 **rxResultsDF** 출력이 훨씬 더 명확하며 원본 열의 이름을 유지합니다.

3. 사용 하 여 열 지도 만들려면 다음 코드를 실행 하는 마지막으로 `levelplot` 에서 함수는 **격자** 모든 R 배포에 포함 되어 있는 패키지입니다.

R 복사

```
levelplot(fraudRisk~numTrans*numIntlTrans, data = cubePlot)
```

결과



이 빠른 분석에서도 트랜잭션 수와 국제 트랜잭션 수 둘 다에서 사기 위험이 증가하는 것을 확인할 수 있습니다.

에 대 한 자세한 내용은 **rxCube** 함수와 크로스탭 일반적으로 참조 RevoScaleR 을 사용 하 여 데이터 요약합니다.

다음 단계

SQL Server 데이터를 사용 하 여 R 모델 만들기

이전 단계

R 스크립트 만들기 및 실행

R 모델 (SQL 과 R 심층 분석)을 만들으십시오

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.
이제 교육 데이터를 보강했으므로 선형 회귀를 사용하여 데이터를 분석해야 합니다. 선형 모델은 예측 분석 분야의 중요한 도구이며, 의 RevoScaleR R Services(In-Database) 패키지에는 확장성 있는 고성능 알고리즘이 포함되어 있습니다.

선형 회귀 모델 만들기

이 단계에 있는 값의 독립 변수로 사용 하 여 고객에 대 한 신용 카드 잔액을 예측 하는 간단한 선형 모델을 만듭니다는 *성별* 및 *creditLine* 열입니다.
이 위해 사용 하 여는 rxLinMod 원격 계산 컨텍스트를 지 원하는 함수입니다.

1. 완료 된 저장 하는 R 변수를 만들 모델 및 호출 **rxLinMod**, 적절 한 수식 전달 합니다.

   R 복사

   ```
   linModObj <- rxLinMod(balance ~ gender + creditLine,  data = sqlFraudDS)
   ```

2. 표준 R 을 호출 하는 결과의 요약을 보려면 `summary` 모델 개체에 대해 함수입니다.

   R 복사

   ```
   summary(linModObj)
   ```

일반 R 함수 등의 특이 한 생각할 수 있습니다 `summary` 이전 단계에서 서버에 계산 컨텍스트를 설정 하므로 여기에서 작동할 것입니다. 그러나 **rxLinMod** 함수는 원격 계산 컨텍스트를 사용하여 모델을 만들더라도 로컬 워크스테이션에 대한 모델을 포함하고 이를 공유 디렉터리에 저장하는 개체도 반환합니다. 따라서 "로컬" 컨텍스트를 사용하여 만들어진 것처럼 모델에 대해 표준 R 명령을 실행할 수 있습니다.

결과

복사

```
Linear Regression Results for: balance ~ gender + creditLineData: sqlFraudDS
(RxSqlServerData Data Source)

Dependent variable(s): balance

Total independent variables: 4 (Including number dropped: 1)

Number of valid observations: 10000

Number of missing observations: 0

Coefficients: (1 not defined because of singularities)


Estimate Std. Error t value Pr(>|t|) (Intercept)

3253.575 71.194 45.700 2.22e-16

gender=Male -88.813 78.360 -1.133 0.257

gender=Female Dropped Dropped Dropped Dropped

creditLine 95.379 3.862 24.694 2.22e-16

Signif. codes: 0  0.001  0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 3812 on 9997 degrees of freedom

Multiple R-squared: 0.05765

Adjusted R-squared: 0.05746

F-statistic: 305.8 on 2 and 9997 DF, p-value: < 2.2e-16

Condition number: 1.0184
```

로지스틱 회귀 모델 만들기

다음으로, 특정 고객 사기 위험 인지 여부를 나타내는 로지스틱 회귀 모델을 만듭니다. 사용 하 여는 **RevoScaleR** rxLogit 계산 컨텍스트는 원하는 맞춤 원격에 로지스틱 회귀 모델의 함수입니다.

1. 계산 컨텍스트를 그대로 유지합니다. 또한 계속해서 같은 데이터 원본을 사용합니다.

2. **rxLogit** 함수를 호출하고 모델을 정의하는 데 필요한 수식을 전달합니다.

R 복사

```
logitObj <- rxLogit(fraudRisk ~ state + gender + cardholder + balance +
numTrans + numIntlTrans + creditLine, data = sqlFraudDS, dropFirst =
TRUE)
```

삭제된 더미 변수 3 개를 비롯하여 60 개의 독립 변수를 포함하는 큰 모델이기 때문에 계산 컨텍스트에서 개체가 반환될 때까지 잠시 기다려야 할 수도 있습니다.

모델이 이렇게 큰 이유는 R 및 **RevoScaleR** 패키지에서 모든 수준의 범주 요인 변수가 자동으로 별도의 더미 변수로 처리되기 때문입니다.

3. 반환 된 모델의 요약을 확인 하려면 R 을 호출 `summary` 함수입니다.

R 복사

```
summary(logitObj)
```

일부 결과

복사

```
Logistic Regression Results for: fraudRisk ~ state + gender + cardholder +
balance + numTrans + numIntlTrans + creditLine

Data: sqlFraudDS (RxSqlServerData Data Source)

Dependent variable(s): fraudRisk

Total independent variables: 60 (Including number dropped: 3)

Number of valid observations: 10000 -2



LogLikelihood: 2032.8699 (Residual deviance on 9943 degrees of freedom)



Coefficients:

Estimate Std. Error z value Pr(>|z|)    (Intercept)
```

```
-8.627e+00  1.319e+00  -6.538 6.22e-11

state=AK               Dropped    Dropped Dropped  Dropped

state=AL            -1.043e+00  1.383e+00  -0.754   0.4511


(other states omitted)


gender=Male            Dropped    Dropped Dropped  Dropped

gender=Female         7.226e-01  1.217e-01   5.936 2.92e-09

cardholder=Principal    Dropped    Dropped Dropped  Dropped

cardholder=Secondary  5.635e-01  3.403e-01   1.656   0.0977

balance              3.962e-04  1.564e-05  25.335 2.22e-16

numTrans             4.950e-02  2.202e-03  22.477 2.22e-16

numIntlTrans          3.414e-02  5.318e-03   6.420 1.36e-10

creditLine            1.042e-01  4.705e-03  22.153 2.22e-16


Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Condition number of final variance-covariance matrix: 3997.308

Number of iterations: 15
```

다음 단계

새 데이터 점수

새 데이터 (SQL 과 R 심층 분석) 점수를 매깁니다.

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

이 단계에서는 앞에서 만든 입력으로 동일한 독립 변수를 사용 하는 다른 데이터 세트에 대 한 점수를 작성 하는 로지스틱 회귀 모델을 사용 합니다.

참고

다음이 단계 중 일부에 대 한 DDL 관리자 권한이 필요합니다.

생성 하 고 점수를 저장 합니다.

1. 이전에 설정 하는 데이터 원본을 업데이트 `sqlScoreDS`, 필요한 열 정보를 추가 합니다.

R 복사

```
sqlScoreDS <- RxSqlServerData(
    connectionString = sqlConnString,
    table = sqlScoreTable,
    colInfo = ccColInfo,
    rowsPerRead = sqlRowsPerRead)
```

2. 결과 손실 하지 않으려면 하 게 하려면 새 데이터 원본 개체를 만듭니다. 그런 다음 새 데이터 원본 개체를 사용 하 여 새 테이블에 채우기는 SQL Server 데이터베이스입니다.

R 복사

```
sqlServerOutDS <- RxSqlServerData(table = "ccScoreOutput",
    connectionString = sqlConnString,
    rowsPerRead = sqlRowsPerRead )
```

이때 테이블은 만들어지지 않았습니다. 이 문은 데이터의 컨테이너를 정의할 뿐입니다.

3. 현재 계산 컨텍스트를 확인하고, 필요한 경우 계산 컨텍스트를 서버로 설정합니다.

R 복사

```
rxSetComputeContext(sqlCompute)
```

4. 결과를 생성하는 예측 함수를 실행하기 전에 기존 출력 테이블이 있는지 확인해야 합니다. 그렇지 않으면 새 테이블을 작성 하려고 할 때 오류가 발생할 것 있습니다.

이렇게 하려면 **rxSqlServerTableExists** 및 **rxSqlServerDropTable** 함수를 호출하여 테이블 이름을 입력으로 전달합니다.

R 복사

```
if (rxSqlServerTableExists("ccScoreOutput"))
rxSqlServerDropTable("ccScoreOutput")
```

- **rxSqlServerTableExists** 함수는 ODBC 드라이버를 쿼리하고, 테이블이 있으면 TRUE 를 반환하고 없으면 FALSE 를 반환합니다.
- 함수 **rxSqlServerDropTable** DDL 을 실행 하 고 TRUE 테이블이 있는 경우 성공적으로 삭제 FALSE 그렇지 않으면 반환 합니다.
- 두 함수를 볼 수 있습니다에 대 한 참조: rxSqlServerDropTable

5. 사용할 준비가 되었습니다. 이제는 rxPredict 함수를 점수를 생성 하 고 데이터 원본에 정의 된 새 테이블에 저장할 `sqlScoreDS` 합니다.

R 복사

```
rxPredict(modelObject = logitObj,
    data = sqlScoreDS,
    outData = sqlServerOutDS,
    predVarNames = "ccFraudLogitScore",
      type = "link",
    writeModelVars = TRUE,
    overwrite = TRUE)
```

rxPredict 함수는 원격 계산 컨텍스트에서의 실행을 지원하는 또 다른 함수입니다. **rxPredict** 함수를 사용하여 rxLinMod, rxLogit 또는 rxGlm 을 통해 만든 모델에서 점수를 만들 수 있습니다.

- 여기서는 *writeModelVars* 매개 변수가 **TRUE** 로 설정되었습니다. 이 경우 예측에 사용된 변수가 새 테이블에 포함됩니다.
- *predVarNames* 매개 변수는 결과를 저장할 변수를 지정합니다. 새 변수를 전달 하는 여기 `ccFraudLogitScore` 합니다.
- *rxPredict* 에 대한 **type** 매개 변수는 예측 계산 방법을 정의합니다. 키워드 지정 **응답** 응답 변수의 배율에 따라 점수를 생성 합니다. 키워드를 사용 또는 **링크** 는

쿼리에서 기본 링크 함수에 따라 점수를 생성할 예측 로지스틱 눈금을 사용 하여 생성 됩니다.

6. 잠시 후 Management Studio 에서 테이블 목록을 새로 고쳐 새 테이블 및 해당 데이터를 확인할 수 있습니다.
7. 출력 예측에 변수를 더 추가하려면 *extraVarsToWrite* 인수를 사용합니다. 예를 들어 다음 코드에서는 변수에서에서 `custID` 점수 매기기 데이터 테이블에서 예측의 출력 테이블에 추가 됩니다.

R 복사

```
rxPredict(modelObject = logitObj,
        data = sqlScoreDS,
        outData = sqlServerOutDS,
        predVarNames = "ccFraudLogitScore",
          type = "link",
        writeModelVars = TRUE,
        extraVarsToWrite = "custID",
        overwrite = TRUE)
```

히스토그램에 표시 점수

새 테이블을 만든 후 계산 표시 하는 10,000 예측 점수 막대 그래프. 낮은 임계값과 높은 값을 지정 하므로 데이터베이스에서 해당 권한을 받아야, 경우 작업 데이터에 추가할 계산이 빠릅니다.

1. 새 데이터 원본을 만들고 `sqlMinMax`, 낮은 임계값과 높은 값을 가져올 데이터베이스를 쿼리 하는 합니다.

R 복사

```
sqlMinMax <- RxSqlServerData(
    sqlQuery = paste("SELECT MIN(ccFraudLogitScore) AS minVal,",
    "MAX(ccFraudLogitScore) AS maxVal FROM ccScoreOutput"),
    connectionString = sqlConnString)
```

이 예제에서는 **RxSqlServerData** 데이터 원본 개체를 사용하여 SQL 쿼리, 함수 또는 저장 프로시저를 기반으로 임의 데이터 세트를 정의한 다음 이를 R 코드에서 사용하기가 얼마나 쉬운지를 확인할 수 있습니다. 변수는 실제 값을 저장하는 것이 아니라 데이터 원본 정의를 저장할 뿐입니다. 쿼리는 **rxImport** 와 같은 함수에서 변수를 사용할 때만 실행되어 값을 생성합니다.

2. 호출 된 rxImport 계산 컨텍스트 간에 공유할 수 있는 데이터 프레임의 끝에 있는 함수입니다.

R 복사

```
minMaxVals <- rxImport(sqlMinMax)
minMaxVals \<- as.vector(unlist(minMaxVals))
```

결과

> minMaxVals

[1] -23.970256 9.786345

3. 이제는 최대값과 최소값을 사용할 수 값을 사용 하 여 생성 된 점수에 대 한 다른 데이터 소스를 만듭니다.

R 복사

```
sqlOutScoreDS <- RxSqlServerData(sqlQuery = "SELECT
ccFraudLogitScore FROM ccScoreOutput",
    connectionString = sqlConnString,
```

```
    rowsPerRead = sqlRowsPerRead,
        colInfo = list(ccFraudLogitScore = list(
            low = floor(minMaxVals[1]),
                    high = ceiling(minMaxVals[2]) ) ) )
```

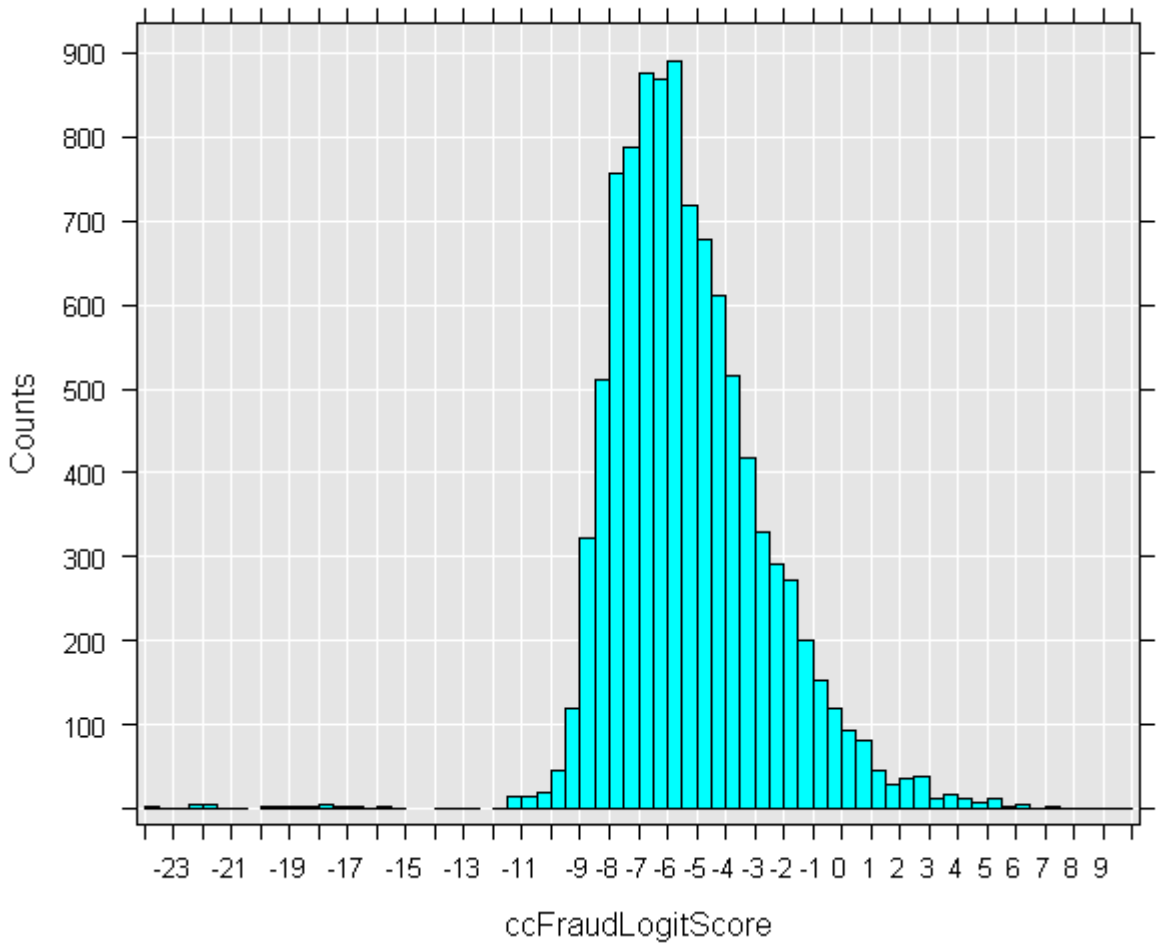
4. 데이터 원본 개체를 사용 하 여 `sqlOutScoreDS` 하는 점수를 얻을 계산 하 고 막대 그래프를 표시 합니다. 필요한 경우 계산 컨텍스트를 설정하는 코드를 추가합니다.

R 복사

```
# rxSetComputeContext(sqlCompute)

rxHistogram(~ccFraudLogitScore, data = sqlOutScoreDS)
```

결과



다음 단계

R 을 사용하여 데이터 변환

이전 단계

모델 만들기

R (SQL 과 R 심층 분석)를 사용하 여 데이터를 변환 합니다.

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

RevoScaleR 패키지는 다양한 분석 단계에서 데이터를 변환하기 위한 여러 함수를 제공합니다.

- **rxDataStep** 은 데이터 하위 세트를 만들고 변환하는 데 사용할 수 있습니다.
- **rxImport** 는 XDF 파일 또는 메모리 내 데이터 프레임으로 또는 그 반대로 데이터를 가져올 때 데이터 변환을 지원합니다.
- 특별히 데이터 이동에 사용되지는 않지만 **rxSummary**, **rxCube**, **rxLinMod** 및 **rxLogit** 함수도 모두 데이터 변환을 지원합니다.

이 섹션에서는 이러한 함수를 사용 하는 방법을 설명 합니다. 부터 시작 하겠습니다 rxDataStep 합니다.

RxDataStep 를 사용 하 여 변수를 변환

rxDataStep 함수는 하나의 데이터 원본에서 읽어 다른 데이터 원본에 쓰면서 한 번에 하나씩 데이터 청크를 처리합니다. 변환할 열, 로드할 변환 등을 지정할 수 있습니다.

효과적으로 만들기 위해이 예에서는, 데이터를 변형 다른 R 패키지의 함수를 사용 하겠습니다. **boot** 패키지는 "권장" 패키지 중 하나이므로, **boot** 는 R 의 모든 배포에 포함되지만 시작할 때 자동으로 로드되지 않습니다. 따라서 SQL Server 와

함께 사용 중인 R Services(In-Database)인스턴스에서 패키지를 이미 사용할 수 있어야 합니다.

부팅 함수를 사용 하 여, 패키지 `inv.logit` 는 logit 의 역 수를 계산 합니다. 즉, `inv.logit` 함수는 로짓을 다시 [0,1] 눈금의 확률로 변환합니다.

팁

이 규모에 맞게 예측을 가져오는 다른 방법을 설정 하는 것은 *형식* 매개 변수를 **응답** rxPredict 원래 호출에 합니다.

1. 테이블에 데이터를 저장할 데이터 소스를 만들어 시작 `ccScoreOutput` 합니다.

R 복사

```
sqlOutScoreDS <- RxSqlServerData( table =  "ccScoreOutput",
connectionString = sqlConnString, rowsPerRead = sqlRowsPerRead )
```

2. 다른 데이터 소스 테이블에 대 한 데이터를 저장할 추가 `ccScoreOutput2` 합니다.

R 복사

```
sqlOutScoreDS2 <- RxSqlServerData( table =  "ccScoreOutput2",
connectionString = sqlConnString, rowsPerRead = sqlRowsPerRead )
```

새 테이블에는 이전 모든 변수를 저장 `ccScoreOutput` 테이블 및 새로 만든된 변수에 합니다.

3. 계산 컨텍스트를 SQL Server 인스턴스로 설정합니다.

R 복사

```
rxSetComputeContext(sqlCompute)
```

4. 함수를 사용 하 여 **rxSqlServerTableExists** 확인 하려면 있는지 여부를 출력 테이블 `ccScoreOutput2` 이미; 있으며이 경우 함수를 사용 **rxSqlServerDropTable** 는 테이블을 삭제 합니다.

R 복사

```
if (rxSqlServerTableExists("ccScoreOutput2"))
rxSqlServerDropTable("ccScoreOutput2")
```

5. **rxDataStep** 함수를 호출하고 목록에서 원하는 변환을 지정합니다.

R 복사

```
rxDataStep(inData = sqlOutScoreDS,
    outFile = sqlOutScoreDS2,
    transforms = list(ccFraudProb = inv.logit(ccFraudLogitScore)),
    transformPackages = "boot",
    overwrite = TRUE)
```

각 열에 적용되는 변환을 정의할 때 변환에 필요한 추가 R 패키지를 지정할 수도 있습니다. 변환을 수행할 수 있는 형식에 대한 자세한 내용은 RevoScaleR 을 사용하여 데이터를 변환 및 부분 집합 방법을 참조합니다.

6. **rxGetVarInfo** 를 호출하여 새 데이터 세트의 변수 요약을 확인합니다.

R 복사

```
rxGetVarInfo(sqlOutScoreDS2)
```

결과

Var 1: ccFraudLogitScore, Type: numeric

Var 2: state, Type: character

Var 3: gender, Type: character

Var 4: cardholder, Type: character

Var 5: balance, Type: integer

Var 6: numTrans, Type: integer

Var 7: numIntlTrans, Type: integer

Var 8: creditLine, Type: integer

Var 9: ccFraudProb, Type: numeric

원래 로짓 점수는 유지되지만 새 열 *ccFraudProb* 가 추가되어 로짓 점수가 0 과 1 사이의 값으로 표시됩니다.

테이블에 기록 된 비율 변수는 `ccScoreOutput2` 문자 데이터로 합니다. 이후 분석에서 요인으로 사용하려면 *colInfo* 매개 변수를 사용하여 수준을 지정합니다.

다음 단계

rxImport 를 사용하여 메모리에 데이터 로드

이전 단계

R 스크립트 만들기 및 실행

RxImport (SQL 과 R 심층 분석)를 사용하여 메모리에 데이터 로드

이 문서는 데이터 과학 심층 분석 자습서를 사용하는 방법에 대한 일부 RevoScaleR SQL Server 와 함께 합니다.
rxImport 데이터 프레임에서 세션 메모리 나 디스크에 있는 XDF 파일에 데이터 원본에서 데이터를 이동하는 함수를 사용할 수 있습니다. 파일을 대상으로 지정하지 않으면 데이터는 메모리에 데이터 프레임으로 저장됩니다.
이 단계에서 데이터를 가져오는 방법을 배웁니다 SQL Server 를 사용하여는 **rxImport** 관심 있는 데이터는 로컬 파일에 추가하는 함수입니다. 이렇게 하면 데이터베이스를 다시 쿼리하지 않고도 로컬 계산 컨텍스트에서 반복하여 데이터를 분석할 수 있습니다.

SQL Server 에서 로컬 메모리에 데이터의 하위 세트를 추출

자세히 위험 수준이 높은 개인만 검사할 것인지 결정합니다. 원본 테이블에 SQL Server 크므로 방금 위험 수준이 높은 고객에 대한 정보를 가져오려는 합니다. 그런 다음 로컬 워크스테이션의 메모리에 데이터 프레임으로 해당 데이터를 로드합니다.

1. 계산 컨텍스트를 로컬 워크스테이션으로 다시 설정합니다.

   R 복사

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

2. *sqlQuery* 매개 변수에 유효한 SQL 문을 제공하는 새 SQL Server 데이터 원본 개체를 만듭니다. 이 예제에서는 위험 점수가 가장 높은 관찰의 하위 세트를 가져옵니다. 이렇게 하면 필요한 데이터만 로컬 메모리에 저장됩니다.

   R 복사

```
sqlServerProbDS \<- RxSqlServerData(

    sqlQuery = paste("SELECT * FROM ccScoreOutput2",

    "WHERE (ccFraudProb > .99)"),

    connectionString = sqlConnString)
```

3. 함수 호출 rxImport 로컬 R 세션에서 데이터 프레임으로 데이터를 읽을 수 있습니다.

R 복사

```
highRisk <- rxImport(sqlServerProbDS)
```

작업에 성공 하면 다음과 같은 상태 메시지가 나타납니다: "Rows Read: 35, 총 행 처리 됨: 35, 총 청크 시간: 0.036 시간 (초)"

4. 메모리 내 데이터 프레임에서 위험 수준이 높은 관찰 인 했으므로 데이터 프레임을 조작 하기 위한 다양 한 R 함수를 사용할 수 있습니다. 예를 들어 해당 위험 점수 고객 주문 수 있으며 가장 높은 위험 노출 하는 고객의 목록을 인쇄.

R 복사

```
orderedHighRisk <- highRisk[order(-highRisk$ccFraudProb),]

row.names(orderedHighRisk) <- NULL

head(orderedHighRisk)
```

결과

ccFraudLogitScore state gender cardholder balance numTrans numIntlTrans creditLine ccFraudProb1
9.786345 SD Male Principal 23456 25 5 75 0.99994382
9.433040 FL Female Principal 20629 24 28 75 0.99992003
8.556785 NY Female Principal 19064 82 53 43 0.99980784
8.188668 AZ Female Principal 19948 29 0 75 0.99972235
7.551699 NY Female Principal 11051 95 0 75 0.99947516
7.335080 NV Male Principal 21566 4 6 75 0.9993482

rxImport 대한 자세한 정보

rxImport 를 사용하여 데이터 이동은 물론 데이터를 읽는 동안 데이터를 변환할 수도 있습니다. 예를 들어 고정 너비 열에 대해 문자 수를 지정하고, 변수에 대한 설명을 제공하고, 요인 열에 대한 수준을 설정하고, 가져온 후 사용할 새로운 수준을 만들 수 있습니다.
rxImport 함수 가져오기 프로세스 동안 열에 변수 이름을 할당 하지만 사용 하여 새 변수 이름을 지정할 수 있습니다는 *colInfo* 매개 변수 또는 를사용하여변경데이터형식 *colClasses* 매개 변수입니다.
transforms 매개 변수에서 추가 작업을 지정하여 읽은 각 데이터 청크에 대한 기본 처리를 수행할 수 있습니다.

다음 단계

rxDataStep 을 사용하여 새 SQL Server 테이블 만들기

이전 단계

R 을 사용하여 데이터 변환

RxDataStep (SQL 과 R 심층 분석)를 사용하여 새 SQL Server 테이블 만들기

이 문서는 데이터 과학 심층 분석 자습서를 사용하는 방법에 대한 일부 RevoScaleR SQL Server 와 함께 합니다.

이 단원에서는 메모리 내 데이터 프레임 간에 데이터를 이동하는 방법을 배웁니다는 SQL Server 컨텍스트와 로컬 파일입니다.

참고

이 단원에서는 다른 데이터 세트를 사용합니다. 항공편 지연 데이터 세트는 머신 러닝 실험에 널리 사용되는 공용 데이터 세트입니다. 이 예에서 사용된 데이터 파일은 다른 제품 예제와 동일한 디렉터리에서 사용할 수 있습니다.

로컬 데이터에서 SQL Server 테이블 만들기

이 자습서의 앞부분에서 사용한는 **RxTextData** 텍스트 파일에서 R 로 데이터를 가져오려면 작동하고 다음 사용는 **RxDataStep** 함수에 데이터를 이동하려면 SQL Server 합니다.

이 단원에는 다른 방법을 사용하고 파일에서 사용하여 데이터에 저장된 XDF 형식합니다. 변환된 데이터 형식의 새 파일에 저장하면 XDF 파일을 사용

하 여 데이터에 대 한 몇 가지 간단한 변형 작업을 수행한 후 SQL Server 테이블입니다.

XDF 소식

XDF 형식이 고 차원 데이터에 대 한 개발 하는 XML 표준 및에서 사용 하는 네이티브 파일 형식 컴퓨터 학습 서버합니다. 또한 행 및 열 처리와 분석을 최적화하는 R 인터페이스가 포함된 이진 파일 형식입니다. XDF 형식을 사용하여 데이터를 이동하고 분석에 유용한 데이터의 하위 세트를 저장할 수 있습니다.

1. 계산 컨텍스트를 로컬 워크스테이션으로 설정합니다. **이 단계에 대 한 DDL 권한이 필요 합니다.**

복사

```R
rxSetComputeContext("local")
```

1. **RxXdfData** 함수를 사용하여 새 데이터 원본 개체를 정의합니다. XDF 데이터 소스를 정의 하려면 데이터 파일의 경로를 지정 합니다.

텍스트 변수를 사용 하 여 파일의 경로를 지정할 수 있습니다. 그러나,이 경우에 사용 하는 편리한 바로 가기는 **rxGetOption** 작동 하 고 샘플 데이터 디렉터리에서 파일 (AirlineDemoSmall.xdf)을 가져옵니다.

R 복사

```R
xdfAirDemo <- RxXdfData(file.path(rxGetOption("sampleDataDir"), "AirlineDemoSmall.xdf"))
```

2. 메모리 내 데이터에서 rxGetVarInfo 를 호출하여 데이터 세트 요약을 확인합니다.

R 복사

```
rxGetVarInfo(xdfAirDemo)
```

결과

Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)

Var 2: CRSDepTime, Type: numeric, Storage: float32, Low/High: (0.0167, 23.9833)

Var 3: DayOfWeek 7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday

참고

XDF 파일로 데이터를 로드하기 위해 다른 함수를 호출할 필요가 없으며 데이터에서 즉시 **rxGetVarInfo** 를 호출할 수 있다는 사실을 알아차리셨나요? XDF 가 RevoScaleR 에 대한 기본 임시 저장 방법이기 때문입니다. XDF 파일 외에 **rxGetVarInfo** 함수에는 이제 다양 한 원본 형식을 지원 합니다.

1. 이 데이터를 넣기는 SQL Server 테이블, 저장 *DayOfWeek* 으로 정수 1 ~ 7 사이의 값입니다.

   이렇게 하려면 먼저 SQL Server 데이터 원본을 정의합니다.

   R 복사

   ```
   sqlServerAirDemo <- RxSqlServerData(table = "AirDemoSmallTest",
   connectionString = sqlConnString)
   ```

2. 이름이 같은 테이블이 이미 있는지 확인하고, 있을 경우 테이블을 삭제합니다.

   R 복사

   ```
   if (rxSqlServerTableExists("AirDemoSmallTest",  connectionString =
   sqlConnString))  rxSqlServerDropTable("AirDemoSmallTest",
   connectionString = sqlConnString)
   ```

3. 테이블을 만들고 **rxDataStep** 에 대한 다양한 제품 샘플에서 사용되므로 테스트를 위해 보관해 두면 유용합니다. 이 함수는 두 데이터에서 이미 데이터 원본을 정의 하 고 전달 되는 도중 데이터를 선택적으로 변환할 수를 이동 합니다.

R 복사

```
rxDataStep(inData = xdfAirDemo, outFile = sqlServerAirDemo,
       transforms = list( DayOfWeek = as.integer(DayOfWeek),
       rowNum = .rxStartRow : (.rxStartRow + .rxNumRows - 1) ),
       overwrite = TRUE )
```

이 테이블은 매우 크기 때문에 테이블, 하므로이 이와 같은 최종 상태 메시지가 표시 될 때까지 기다렸다가: *Rows Read: 200000, 총 행이 처리: 600000* 합니다.

4. 계산 컨텍스트를 SQL Server 컴퓨터로 다시 설정합니다.

R 복사

```
rxSetComputeContext(sqlCompute)
```

5. 새 테이블에서 간단한 SQL 쿼리를 사용하여 새 SQL Server 데이터 원본을 만듭니다. 이 정의 대 한 계수 수준을 추가 *DayOfWeek* 열을 사용 하는 *colInfo* 인수를 **RxSqlServerData** 합니다.

R 복사

```
SqlServerAirDemo <- RxSqlServerData(
    sqlQuery = "SELECT * FROM AirDemoSmallTest",
    connectionString = sqlConnString,
    rowsPerRead = 50000,
    colInfo = list(DayOfWeek = list(type = "factor",  levels =
as.character(1:7))))
```

6. 호출 **rxSummary** 쿼리에서 데이터의 요약을 검토 하는 한 번 더 합니다.

R 복사

```
rxSummary(~., data = sqlServerAirDemo)
```

다음 단계

rxDataStep 을 사용하여 청크 분석 수행

이전 단계

rxImport 를 사용하여 메모리에 데이터 로드

RxDataStep (SQL 과 R 심층 분석)를 사용 하 여 청크 분석 수행

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

이 단원에서는 사용 하 여는 **rxDataStep** 전체 데이터 세트의 메모리에 로드 하 고 기존의 오른쪽에서와 같이 한 번에 처리 될 필요 하지 않고 데이터 청크를 처리 하려면 함수 **rxDataStep** 함수 읽는 데이터 청크를 차례로 데이터의 각 청크를 R 함수를 적용 하 고 다음에 공통 각 청크에 요약 결과 저장 SQL Server 데이터 원본입니다. 모든 데이터를 읽을 때 결과가 결합 됩니다.

팁

이 단원에서는 사용 하 여 대체 테이블을 계산에서 `table` R 에서 함수 이 예제에서는 지침만 제공 하기 위한 것입니다.

실제 데이터 세트를 표로 해야 할 경우 사용 하는 것이 좋습니다는 **rxCrossTabs** 또는 **rxCube** 함수가 **RevoScaleR**,이 대 한 일종의 최적화 작업입니다.

값으로 데이터 분할

1. R 을 호출 하는 사용자 지정 R 함수를 만들 `table` 데이터의 각 청크에서 작동 하 고 새 함수 이름을 `ProcessChunk` 합니다.

R 복사

```
ProcessChunk <- function( dataList) {

# Convert the input list to a data frame and compute contingency
table

chunkTable <- table(as.data.frame(dataList))


# Convert table output to a data frame with a single row

varNames <- names(chunkTable)

varValues <- as.vector(chunkTable)

dim(varValues) <- c(1, length(varNames))

chunkDF <- as.data.frame(varValues)

names(chunkDF) <- varNames


# Return the data frame, which has a single row

return( chunkDF )

}
```

2. 계산 컨텍스트를 서버로 설정합니다.

R 복사

```
rxSetComputeContext( sqlCompute )
```

3. 처리 중인 데이터를 보관할 SQL Server 데이터 원본을 정의 합니다. 먼저 SQL 쿼리를 변수에 할당합니다. 그런 다음에 해당 변수를 사용 하 여는 *sqlQuery* 새의 인수 SQL Server 데이터 원본입니다.

복사

```
```R
```
```

```
dayQuery <-  "SELECT DayOfWeek FROM AirDemoSmallTest"

inDataSource <- RxSqlServerData(sqlQuery = dayQuery,

    connectionString = sqlConnString,

    rowsPerRead = 50000,

    colInfo = list(DayOfWeek = list(type = "factor",

        levels = as.character(1:7))))
```

1. 실행할 수 있습니다 **rxGetVarInfo** 이 데이터 원본에 있습니다. 이 시점에서 단일 열을 포함: *Var 1: DayOfWeek, 유형:을 고려 하 고 사용 가능한 비율 수준이 없습니다.*
2. 이 요인 변수를 원본 데이터에 적용하기 전에 중간 결과를 저장할 별도의 테이블을 만듭니다. 다시 방금는 RxSqlServerData 함수를 사용 하면 makign 같은 이름의 기존 테이블을 삭제 하 시겠습니까 데이터를 정의 합니다.

R 복사

```
iroDataSource = RxSqlServerData(table = "iroResults",
connectionString = sqlConnString)

# Check whether the table already exists.

if (rxSqlServerTableExists(table = "iroResults",  connectionString =
sqlConnString))  { rxSqlServerDropTable( table = "iroResults",
connectionString = sqlConnString) }
```

3. 사용자 정의 함수를 호출 `ProcessChunk` 읽을 때로 사용 하 여 데이터를 변환 하는 *transformFunc* 인수에는 **rxDataStep** 함수입니다.

R 복사

```
rxDataStep( inData = inDataSource, outFile = iroDataSource,
transformFunc = ProcessChunk, overwrite = TRUE)
```

4. 중간 결과를 보려면 `ProcessChunk`, 결과를 할당 **rxImport** 변수에 한 다음 콘솔에 결과 출력 합니다.

R 복사

```
iroResults <- rxImport(iroDataSource)
iroResults
```

일부 결과

	1	2	3	4	5	6	7
1	8228	8924	6916	6932	6944	5602	6454
2	8321	5351	7329	7411	7409	6487	7692

1. 모든 청크의 최종 결과를 계산하려면 열의 합계를 구한 다음 결과를 콘솔에 표시합니다.

R 복사

```
finalResults <- colSums(iroResults)
finalResults
```

결과

1	2	3	4	5	6	7
97975	77725	78875	81304	82987	86159	94975

2. 호출 하는 중간 결과 테이블을 제거 하려면 **rxSqlServerDropTable** 합니다.

R 복사

```
rxSqlServerDropTable( table = "iroResults", connectionString =
sqlConnString)
```

다음 단계

로컬 계산 컨텍스트에서 데이터 분석

이전 단계

rxDataStep 을 사용하여 새 SQL Server 테이블 만들기

로컬 계산 컨텍스트 (SQL 과 R 심층 분석)에서 데이터 분석

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.
이 섹션에서는 로컬 계산 컨텍스트로 다시 전환 하 고 성능을 최적화 하기 위해 컨텍스트 간에 데이터를 이동 하는 방법을 설명 합니다.
I 수는 있지만 더 빠르게 서버 컨텍스트를 사용 하 여 복잡 한 R 코드를 실행 하려면, 때로는 것이 더 편리의 데이터를 가져오는 SQL Server 로컬 워크스테이션에서 분석 하 고 있습니다.

로컬 요약 만들기

1. 모든 작업을 로컬에서 수행하도록 계산 컨텍스트를 변경합니다.

   R 복사

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

2. SQL Server 에서 데이터를 추출하는 경우 각 읽기에서 추출되는 행 수를 늘리면 대체로 성능이 향상됩니다. 이렇게 하려면 데이터 원본에 대한 *rowsPerRead* 매개 변수의 값을 늘립니다. 이전에는 *rowsPerRead* 값이 5000 으로 설정되었습니다.

   R 복사

   ```
   sqlServerDS1 <- RxSqlServerData(
      connectionString = sqlConnString,
      table = sqlFraudTable,
      colInfo = ccColInfo,
      rowsPerRead = 10000)
   ```

3. 호출 **rxSummary** 새 데이터 원본에 있습니다.

R 복사

```
rxSummary(formula = ~gender + balance + numTrans + numIntlTrans +
creditLine, data = sqlServerDS1)
```

실제 결과는 **컴퓨터의 컨텍스트에서** rxSummary SQL Server 를 실행할 때와 같아야 합니다. 그러나 작업은 더 빠르거나 느릴 수 있습니다. 데이터가 분석을 위해 로컬 컴퓨터로 전송되므로 이러한 작업 속도는 데이터베이스에 대한 연결에 크게 좌우됩니다.

다음 단계

SQL Server 및 XDF 파일 간에 데이터를 이동

이전 단계

rxDataStep 을 사용하여 청크 분석 수행

SQL Server 및 XDF 파일 (SQL 과 R 심층 분석) 간 데이터 이동

이 문서는 데이터 과학 심층 분석 자습서를 사용 하는 방법에 대 한 일부 RevoScaleR SQL Server 와 함께 합니다.

이 단계에서는 파일을 사용 하는 XDF 원격 및 로컬 컴퓨팅 컨텍스트 간에 데이터를 전송할 방법을 배웁니다. 데이터 파일에에서 저장할 XDF 데이터에서 변환을 수행할 수 있습니다.

파일에 데이터를 사용 하 여 새을 만들고 완료 되 면 SQL Server 테이블입니다. 함수 rxDataStep 고.xdf 파일 및 데이터 프레임 간의 변환을 수행 하는 데이터에 변환을 적용할 수 있습니다.

XDF 파일에서 SQL Server 테이블 만들기

이 연습에서는 신용 카드 사기 데이터 다시 합니다. 이 시나리오에서는 California, Oregon 및 Washington 주의 사용자에 대한 몇 가지 추가 분석을 수행하라는 요청을 받았습니다. 보다 효율적으로 결정 로컬 컴퓨터에만 이러한 상태에 대 한 데이터를 저장 하 고 변수 성별, 카드 소유자, 상태 및 잔액으로 작업 합니다.

1. 다시 사용 된 `stateAbb` 변수를 포함 하 고, 새 변수를 쓸 수준 식별 이전 만든 `statesToKeep` 합니다.

R 복사

```
statesToKeep <- sapply(c("CA", "OR", "WA"), grep, stateAbb)

statesToKeep
```

결과

CA 또는 WA

5 38 48

2. SQL server 에서를 통해 해제 하려는 데이터 정의 사용 하 여는 Transact-SQL 쿼리 합니다. 나중에으로이 변수를 사용 하는 *inData* 에 대 한 인수 **rxImport** 합니다.

R 복사

```
importQuery <- paste("SELECT gender,cardholder,balance,state FROM",
sqlFraudTable,  "WHERE (state = 5 OR state = 38 OR state = 48)")
```

줄 바꿈 또는 탭 등의 숨겨진 문자가 쿼리에 없는지 확인합니다.

3. R 에서 데이터로 작업할 때 사용할 열을 다음으로 정의 예를 들어 더 작은 데이터 세트에 필요한 세 개의 요인 수준, 쿼리만 세 가지 상태에 대 한 데이터를 반환 하기 때문에 합니다. 적용 된 `statesToKeep` 변수를 포함 하도록 올바른 수준 식별 합니다.

R 복사

```
importColInfo <- list(

    gender = list( type = "factor",  levels = c("1", "2"), newLevels
= c("Male", "Female")),

    cardholder = list(  type = "factor",  levels = c("1", "2"),
newLevels = c("Principal", "Secondary")),
```

```
    state = list(   type = "factor",  levels =
as.character(statesToKeep), newLevels = names(statesToKeep))

        )
```

4. 계산 컨텍스트를 설정 **로컬**로컬 컴퓨터에서 사용할 수 있는 모든 데이터를 원하는 때문에 있습니다.

R 복사

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

rxImport 함수 로컬 XDF 파일에 지원 되는 데이터 소스에서 데이터를 가져올 수 있습니다. 하지만 동일한 쿼리를 반복 해 실행 되지 않도록 데이터에 대해 많은 다르게 분석 하려는 경우 데이터의 로컬 복사본을 사용 하는 것이 편리 합니다.

5. 이전에 인수로 정의 된 변수를 전달 하 여 데이터 원본 개체를 만들 **RxSqlServerData** 합니다.

R 복사

```
sqlServerImportDS <- RxSqlServerData(
    connectionString = sqlConnString,
    sqlQuery = importQuery,
    colInfo = importColInfo)
```

6. 호출 **rxImport** 라는 파일에 데이터를 쓸 `ccFraudSub.xdf`, 현재 작업 디렉터리에 있습니다.

R 복사

```
localDS <- rxImport(inData = sqlServerImportDS,
    outFile = "ccFraudSub.xdf",
    overwrite = TRUE)
```

`localDs` 에서 반환 된 개체는 **rxImport** 함수는는 간단한 **RxXdfData** 나타내는 데이터 원본 개체는 `ccFraud.xdf` 데이터 파일을 디스크에 로컬로 저장 합니다.

7. XDF 파일에 대해 rxGetVarInfo 를 호출하여 데이터 스키마가 같은지 확인합니다.

R 복사

```
rxGetVarInfo(data = localDS)
```

결과

rxGetVarInfo(data = localDS)

Var 1: gender, Type: factor, no factor levels available

Var 2: cardholder, Type: factor, no factor levels available

Var 3: balance, Type: integer, Low/High: (0, 22463)

Var 4: state, Type: factor, no factor levels available

8. 이제 SQL Server 의 원본 데이터와 마찬가지로 다양한 R 함수를 호출하여 `localDs` 개체를 분석할 수 있습니다. 예를 들어 성별 요약할 수 있습니다.

R 복사

```
rxSummary(~gender + cardholder + balance + state, data = localDS)
```

이제 계산 컨텍스트 사용 및 다양한 데이터 원본 작업 방법을 익혔으므로 재미있는 작업을 시도해 보겠습니다. 다음 및 최종 단원에서는 원격 서버의 사용자 지정 R 함수를 실행 하는 간단한 시뮬레이션을 만들 수 있습니다.

다음 단계

간단한 시뮬레이션 만들기

이전 단계

로컬 계산 컨텍스트에서 데이터 분석

간단한 시뮬레이션 (SQL 과 R 심층 분석) 만들기

이 문서를 사용 하는 방법에는 데이터 과학 심층 분석 자습서의 마지막 단계는 RevoScaleR SQL Server 와 함께 합니다.

지금까지 사용 중인 설계 된 R 기능 간의 데이터 이동에 맞게 SQL Server 로컬 계산 컨텍스트 및 합니다. 그러나 사용자 지정 R 함수를 작성하고 서버 컨텍스트에서 실행하려는 경우를 가정해 보세요.

SQL Server rxExec 함수를 사용하여 컴퓨터 컨텍스트에서 임의 함수를 호출할 수 있습니다. 사용할 수도 있습니다 **rxExec** 코어 단일 서버에서 작업을 명시적으로 배포 합니다.

이 단원에서는 간단한 시뮬레이션을 만들려면 원격 서버를 사용 합니다. 시뮬레이션에는 SQL Server 데이터가 필요하지 않습니다. 예제에서는 사용자 지정 함수를 디자인한 다음 **rxExec** 함수를 사용하여 호출하는 방법만 보여 줍니다.

사용 하 여 보다 복잡 한 예제를 보려면 **rxExec**,이 문서를 참조: foreach 및 rxExec 세분화 정교 하지 않은 병렬 처리

사용자 정의 함수 만들기

일반적인 카지노 게임은 다음과 같은 규칙에 따라 한 쌍의 주사위를 굴리는 동작으로 구성됩니다.

- 처음 굴릴 때 7 또는 11 이 나오면 이깁니다.

- 2, 3 또는 12 가 나오면 집니다.
- 4, 5, 6, 8, 9 또는 10 이 나오면 해당 숫자가 포인트가 되고, 다시 포인트를 얻거나(이기는 경우) 7 이 나올 때까지(지는 경우) 계속해서 주사위를 굴립니다.

R 에서 사용자 지정 함수를 만든 다음 여러 번 실행하면 게임을 쉽게 시뮬레이트할 수 있습니다.

1. 다음 R 코드를 사용하여 사용자 지정 함수를 만듭니다.

R 복사

```
rollDice <- function()
{
    result <- NULL
    point <- NULL
    count <- 1
        while (is.null(result))
        {
            roll <- sum(sample(6, 2, replace=TRUE))

            if (is.null(point))
            { point <- roll }
            if (count == 1 && (roll == 7 || roll == 11))
            {  result <- "Win" }
            else if (count == 1 && (roll == 2 || roll == 3 || roll ==
12))
            { result \<- "Loss" }
            else if (count > 1 && roll == 7 )
```

```
            { result \<- "Loss" }
            else if (count > 1 && point == roll)
            { result <- "Win" }
            else { count <- count + 1 }
        }
        result
}
```

2. 분할의 단일 게임을 시뮬레이트하기 위해 함수를 실행 합니다.

R 복사

```
rollDice()
```

이겼나요, 졌나요?

지금 사용 하는 방법을 살펴보겠습니다 **rxExec** 기능을 여러 번 실행 하려면, 달성의 확률을 확인 하는 데 도움이 되는 시뮬레이션을 만들려고 합니다.

시뮬레이션 만들기

SQL Server 컴퓨터 컨텍스트에서 임의 함수를 실행하려면 **rxExec** 함수를 호출합니다. 하지만 **rxExec** 코어가 여기 서버 컨텍스트에 SQL Server 컴퓨터에 사용자 지정 함수를 실행 또는 노드에 걸쳐 병렬로 함수의 분산된 실행을 지원 합니다.

1. 사용자 지정 함수에 인수로 호출 **rxExec** 시뮬레이션을 수정 하는 다른 매개 변수와 함께 합니다.

R 복사

```
sqlServerExec <- rxExec(rollDice, timesToRun=20, RNGseed="auto")
length(sqlServerExec)
```

- *timesToRun* 인수를 사용하여 함수를 실행해야 하는 횟수를 나타냅니다. 이 경우 주사위를 20 회 굴립니다.
- *RNGseed* 및 *RNGkind* 인수를 사용하여 난수 생성을 제어할 수 있습니다. *RNGseed* 를 **auto** 로 설정하면 병렬 난수 스트림이 각 작업자에 대해 초기화됩니다.

2. **rxExec** 함수는 각 실행에 대한 하나의 요인이 포함된 목록을 만듭니다. 그러나 목록이 완성될 때까지 큰 변화는 없습니다. 모든 반복이 완료되면 `length` 로 시작하는 줄에 값이 반환됩니다.

   그러면 다음 단계로 이동하여 승패 레코드 요약을 가져올 수 있습니다.

3. R 의 `unlist` 함수를 사용하여 반환된 목록을 벡터로 변환하고 `table` 컴퓨터 컨텍스트에서 임의 함수를 호출할 수 있습니다.

   R 복사

```
table(unlist(sqlServerExec))
```

   결과가 다음과 같이 표시됩니다.

   손실 Win 12 8

결론

이 자습서에서는 다음과 같은 태스크를 익혔습니다.

- 분석에 사용할 SQL Server 데이터 가져오기
- R 에서 데이터 원본 만들기 및 수정
- 워크스테이션과 SQL Server 서버 간에 모델, 데이터 및 그림 전달

데이터 파일은 Revolution analytics 웹 사이트에서 사용할 수 있는 10 백만 관찰의 더 큰 데이터 세트를 사용하여 이러한 기술은 시험 하려는 경우: 데이터 세트의 인덱스

이 연습에서는 더 큰 데이터 파일을 다시 사용 하려면 데이터를 다운로드 하 고 각 데이터 원본 다음과 같이 수정 합니다.

1. 변수 수정 `ccFraudCsv` 및 `ccScoreCsv` 새 데이터 파일을 가리키도록
2. 참조 하는 테이블의 이름을 변경 *sqlFraudTable* 를 `ccFraud10`
3. 참조 하는 테이블의 이름을 변경 *sqlScoreTable* 를 `ccFraudScore10`

추가 예제

계산 컨텍스트 및 RevoScaler 함수를 전달 하 고 데이터 변환의 사용, 마스터 한 했으므로이 자습서를 확인해 보십시오.

컴퓨터 학습 서비스에 대 한 R 자습서

이전 단계

SQL Server 와 XDF 파일 간에 데이터 이동