

ITWorld How To

숨은 알짜 함수부터 파이썬과의 협업까지 R 중급자를 위한 고급 팁과 가이드

페타바이트급 데이터를 능숙하게 다루는 사람을 보고 있으면 마치 숫자로 이뤄진 세계에 지적 생명체가 존재하는 것 같은 느낌이 든다. 통계학자, 빅데이터 전문가, 소셜 과학자, 데이터 과학자 등으로 불리는 이들 '데이터 교수'는 방대한 데이터 속에서 새로운 인사이트를 찾아내고야 한다. 이들 데이터 교수의 무기 중 하나가 바로 R이다. 그러나 아무리 교수라도 R의 세계를 속속들이 알지는 못 한다. 교수도 종종 놓치는 유용한 R 함수와 R에서 지도를 만드는 '덜 번거로운' 방법을 살펴본다. 데이터 교수의 최고 무기인 파이썬과 R도 자세히 분석해 보자.

- ❖ 중급자도 잘 모르는 유용한 R 함수들
- ❖ R에서 지도를 만드는 쉬운 방법 10단계
- ❖ '파이썬 vs. R' 데이터 과학자의 마음을 훔쳐라



무단 전재
재배포 금지

본 PDF 문서는 IDG Korea의 자산으로, 저작권법의 보호를 받습니다.
IDG Korea의 허락 없이 PDF 문서를 온라인 사이트 등에 무단 게재, 전재하거나 유포할 수 없습니다.

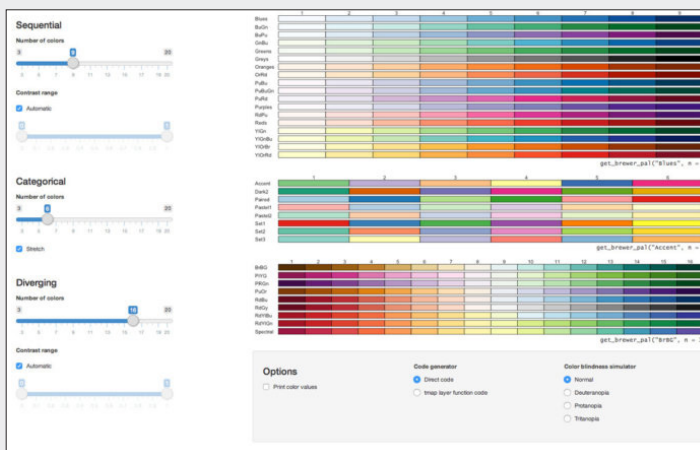
중급자도 잘 모르는 유용한 R 함수들

Sharon Machils | Computerworld

R 사용자라면 dplyr과 ggplot2 같은 유명 패키지를 모르는 사람이 없을 것이다. 그러나 CRAN에만 1만 개 이상의 패키지가 있고 깃허브에는 더 많은 패키지가 있다. 이 중에서 뛰어난 R 함수를 가진 라이브러리를 일일이 찾기는 쉽지 않다. 대신 멋진 R 코드를 찾는 가장 좋은 방법은 다른 R 사용자가 쓰는 것을 보는 것이다. 필자가 이 확인 작업을 대신했다.

인터랙티브 앱에서 ColorBrewer 팔레트 선택. 지도나 앱에서 색채 배합(Color Scheme)이 필요한가? 컬러브루어(ColorBrewer, <http://bit.ly/2ECn2oQ>)는 사전 구성된 팔레트 소스로 유명한데, R컬러브루어 패키지는 팔레트를 R로 가져온다. 그렇지만 사용할 수 있는 팔레트를 기억하는 것도 쉬운 일은 아니다. tmaptools 패키지의 palette_explorer는 사용 가능한 것을 보여주는 인터랙티브 애플리케이션을 생성한다.

먼저, `install.packages("tmaptools")` 명령으로 tmaptools를 설치하고, `library("tmaptools")`로 tmaptools를 로드한 후 `palette_explorer()`를 실행하라. 혹은, tmaptools를 로드하지 말고 `tmaptools::palette_explorer()`를 실행해도 된다. <화면 1>처럼 색상 옵션을 조정하기 위한 슬라이더뿐 아니라 사용할 수 있는 팔레트도 볼 수 있다. 각 팔레트 그룹 아래에는 색채 배합을 사용하기 위한 기본 구문에 대한 정보도 있다. 참고로 `palette_explorer()`가 인터랙티브 앱을 생성하려면 샤이니(shiny)와 샤이니js(shinyjs) 패키지를 미리 설치해야 한다.



화면 1 | tmaptools R 패키지의 palette_explorer 함수

인용부호 없이 문자 벡터 생성하기. Firefox, Chrome, Edge, Safari, Internet Explorer, Opera 등의 텍스트를 R에서 문자열 벡터로 사용하려면 `c("Firefox", "Chrome", "Edge", "Safari", "Internet-Explorer", "Opera")` 포맷으로 수작업으로 바꿔야 한다. 맞다. 조금 번거로운 일이다. Hmisc 패키지의 Cs 함수는 바로 이런 작업을 하기 위해 만들어졌다. 먼저 다음과 같이 Hmisc 패키지를 로드하자.

Cs(Firefox, Chrome, Edge, Safari, InternetExplorer, Opera)

이는 다음과 같은 의미이다.

c("Firefox", "Chrome", "Edge", "Safari", "InternetExplorer", "Opera")

긴 단어 줄에 인용부호를 수동으로 추가하느라 고생한 경험이 있다면 이 함수의 '우아함'에 감동할 것이다. 단, InternetExpolrer에 공백이 없음에 유의해야 한다. 공백이 있으면 Cs 함수에서 오류가 발생한다.

R스튜디오를 사용한다면 깔끔한 벡터 문자열을 만드는 다른 방법이 있다. 보안 전문가인 밥 루디스(Bob Rudis)가 심표로 구분된 텍스트를 받아 필요한 인용부호와 c()를 추가하는 R스튜디오 추가 기능(<http://bit.ly/2GNR7IO>)을 만들었다. 게다가 공백도 처리할 수 있다. `devtools::install_github("hrbrmstr/hrbraddins")` 명령으로 (이는 devtools 패키지도 필요하다는 의미다) 설치하면, RStudio Tools > Addins 메뉴에서 옵션으로 Bare Combine을 선택할 수 있다.

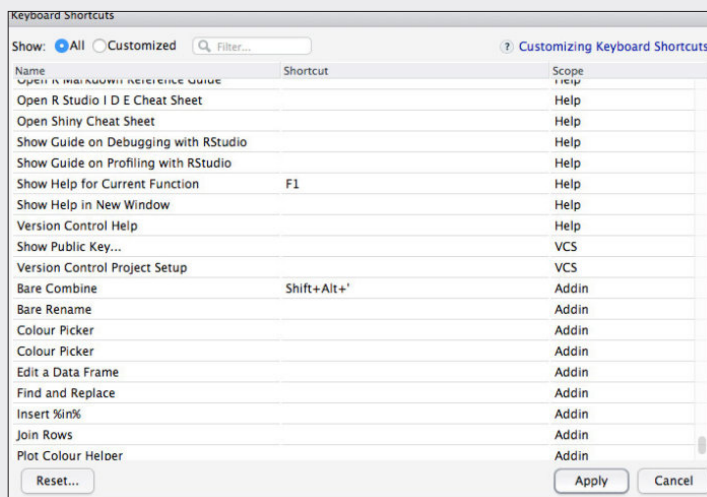
자, 이제 실행하는 문제가 남았다. 이 Addin 메뉴에서 실행할 수도 있지만, 텍스트를 선택한 다음 Tools > Addins 메뉴에서 Bare Combine을 선택하기 위해 코딩 창을 떠나야 한다면 별로 나아진 것이 없다. 이때는 추가 기능에 대한 사용자 정의 단축키를 만드는 것이 훨씬 더 낫다.

단축키를 만드는 메뉴는 Tools > Modify Keyboard Shortcuts에 있다. Addins 섹션에서 Bare Combine이 보일 때까지 스크롤다운 하라. 아니면 필터 상자에서 Bare Combine을 검색해도 된다. 단축키 선택영역에서 더블 클릭하고 해당 확장 기능에 할당하고 싶은 키를 입력하면 된다. 이제, 심표로 구분된 일반 텍스트를 문자열로 구성된 R 벡터로 변환하고 싶을 때마다 텍스트를 선택하고 단축 키를 사용하면 된다.

끝으로, datapasta 패키지의 `vector_paste()`는 또 다른 색다른 대안을 제시한다. Firefox, Chrome, Edge, Safari, Internet Explorer, Opera 같은 문자열을 클립보드에 복사한 후

`vector_paste()`를 실행하면 된다. 그렇다면 `vector_paste()`만 있으면 된다. 그렇게 하면, 이 함수는 클립보드의 내용을 `c("Firefox", "Chrome", "Edge", "Safari", "Internet Explorer", "Opera")`처럼 R 코드로 변환한다. 이는 심표로 구분하거나 한 줄에 한 단어가 있는 경우뿐 아니라 탭으로 구분된 내용에 대해서도 동작한다.

명령어에 데이터를 포함하려면, `c("Firefox", "Chrome", "Safari", "Edge")` 같은 코드를 생성하기 위해 `vector_`



화면 2 | R스튜디오에서 단축키 설정하기

Show 25 entries

	respondent_id	ck_weather	weather_source	weather_source_site	ck_weather_watch	age	female	hhold_income	
6	3886937140	true	A specific website or app (please provide the answer)	AccuWeather App	Somewhat likely	18 - 29	false	\$100,000 to \$124,999	W S C
7	3886923931	true	The Weather Channel		Very unlikely	30 - 44	false	\$25,000 to \$49,999	W S C
8	3886913587	true							
9	3886889048	true	The Weather Channel		Very likely	30 - 44	false	Prefer not to answer	Pi
10	3886848806	true	The default weather app on your phone		Very likely	30 - 44	false	\$150,000 to \$174,999	W N C
11	3886782609	false	Internet search		Somewhat unlikely	18 - 29	false	\$0 to \$9,999	Pi
12	3886416355	false	The default weather app on your phone		Somewhat likely	30 - 44	false	\$100,000 to \$124,999	Ei N C
13	3886356706	false							
14	3886328736	true	Local TV News		Very likely	30 - 44	false	\$0 to \$9,999	M A
15	3886328736	true	The Weather			18 - 29	false	\$10,000 to \$24,999	M A

Showing 1 to 25 of 928 entries

화면 3 | R DT 패키지를 사용해 생성한 HTML 테이블

(Wrangling) 할 수 있게 해준다.

코드 한 줄로 인터랙티브 테이블 만들기. 명령어 줄을 선호하고 아무리 자주 사용하더라도, 검색, 분류 그리고 필터를 위해 스프레드시트 형태의 데이터를 보는 것이 여전히 유용할 때가 있다. R스튜디오도 이런 기본 뷰를 제공했다. 그렇지만, 필자는 커다란 데이터 세트에 대해서는 DataTables 자바스크립트 래퍼(wrapper)인 R스튜디오의 DT 패키지를 선호한다. `DT::datatable(mydf)`은 인터랙티브 HTML 테이블을 생성한다. `DT::datatable(mydf, filter = "top")`은 각 행 위에 필터 박스를 추가한다.

편리한 파일 변환. rio는 필자가 가장 좋아하는 R 패키지 중 하나다. 특정 유형의 파일을 가져오기 위해 어떤 함수를 사용해야 할지 (`read.csv?` `read.table?` `read_excel?`)를 기억해야 하는 대신, rio는 수십 가지 파일 포맷에 대해 하나의 `import` 함수로 프로세스를 크게 단순화한다. 파일 확장자가 rio가 인식할 수 있는 형식이면, `.csv`, `.json`, `.xlsx`, 그리고 `.html` (테이블) 같은 파일에서 알맞게 가져온다. 특정 파일 포맷을 저장하고 싶다면 rio의 `export`도 마찬가지로 사용하면 된다.

그런데 rio에는 이외에 다른 주요 기능이 있다. 예를 들어 한 번에 가져오기와 내보내기를 하는 `convert`. CSV로 저장할 필요가 있는 수백만 행의 엑셀 파일이 있거나, JSON으로 저장하고 싶은 HTML 테이블이 있다면 rio가 안성맞춤이다. `convert("myfile.xlsx", "myfile.csv")` 구문을 사용하면 된다. 첫 번째 인수는 기존 파일이고 두 번째 인수는 원하는 확장자의 대상 파일이다.

R에서 클립보드로 복사해 붙여넣기. rio 관련된 추가 보너스다. rio를 사용해 클립보드와 R 간에 복사할 수 있다. `export(myRobject, "clipboard")`를 이용하면 된다.

커다란 파일 빠르게 가져오고 공간도 절약하기. 최근 필자는 용량이 큰 스프레드시트를 읽을 때 거의 30초가 걸린 적이 있다. 한번은 할 수 있었지만, 여러 번 액세스해야 하는 상황이 되자 매우 번거로웠다. 이럴 때는 시간뿐 아니라 공간을 절약하기 위해 성능이 뛰어나고 압축도 해주는 **fst 패키지**(<http://bit.ly/2ECcb9D>)가 유용하다. 필자가 최대 압축하도록 `write`.

`paste("Firefox, Chrome, Safari, Edge")`

같은 구문으로 `vector_paste()`를 사용할 수 있다. datapasta에는 웹, 엑셀 또는 다른 소스에서 클립보드로 복사한 테이블을 데이터 프레임 생성용 코드로 변환하는 `df_paste()`를 포함해 몇 가지 다른 깔끔한 기능도 포함돼 있다.

인용부호를 추가하는 `Cs()` 함수에 대응해 인용부호를 떼어내는 `nonquote()` 함수도 유용하다. 특정 유형의 데이터를 R로 가져올 때 특히 쓸모가 많다. `nonquote()`는 기본 R 함수로 변수를 더 쉽게 랭글링

`fst(mydf, "myfile.fst", 100)`라고 입력하자 속도가 매우 빨랐고 fst 파일 크기는 원래 스프레드시트의 1/3로 줄었다.

숫자로 구성된 데이터 프레임은 백분율로 변환하기. 한 열은 여러 가지 범주 그리고 나머지는 숫자로 된 데이터 프레임을 가정해 보자. 후보별 그리고 지역별 선거 결과를 보여주는 데이터 프레임 같은 것이다. 이때는 janitor 패키지의 `ns_to_percents()`로 모든 백분율을 계산할 수 있다. 각 백분율에 대한 분모를 "행", "열" 또는 "전체"로 합산할지를 사용자가 선택할 수 있다. 또한, 이 함수는 첫 번째 행에 범주 정보가 있다고 자동으로 가정해 해당 정보를 건너뛰기 때문에 사용자가 숫자가 아닌 열을 수작업으로 처리할 필요가 없다.

janitor는 다른 몇 가지 유용한 기능도 갖고 있다. `adorn_totals()`는 데이터 프레임에 합계 행 및 열을 추가한다. `clean_names()`는 공백이 있는 열 이름과 열 이름 중의 다른 비 R 친화적인 문자를 찾아 R과 호환되도록 바꾼다.

table() 대안들. 데이터 프레임에 있는 변수의 빈도(도수)를 계산할 필요가 있는가? 필자는 이럴 때 횡수와 백분율이 들어있는 크로스탭(Crosstab)을 생성하고 데이터 프레임을 되돌려 주는 janitor의 `crosstab()` 함수를 사용한다. janitor의 `tabyl()`은 개수와 백분율이 들어 있는 데

클라우드에서 R스튜디오 쓴다?

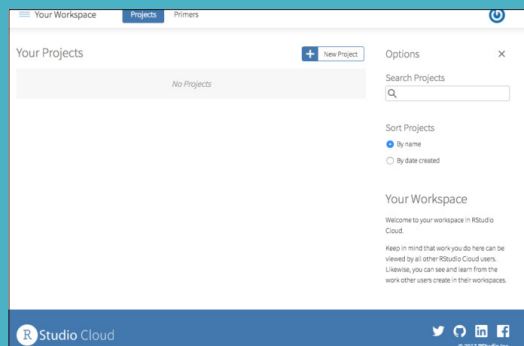
Sharon Machli | Computerworld

R스튜디오 클라우드와 관련해 흥미로운 소식이 있다. 'R스튜디오 자체로부터의 RStudio.cloud'다. 웹사이트(<https://rstudio.cloud>)에 접속해 보면 아직은 알파 버전 단계다. 홈페이지 권고대로 기존 shinyapps.io 계정을 사용해서 로그인해 봤다. shinyapps.io를 사용하지 않는다면 새 계정을 만들거나 구글 또는 깃허브 계정으로 인증할 수 있다.

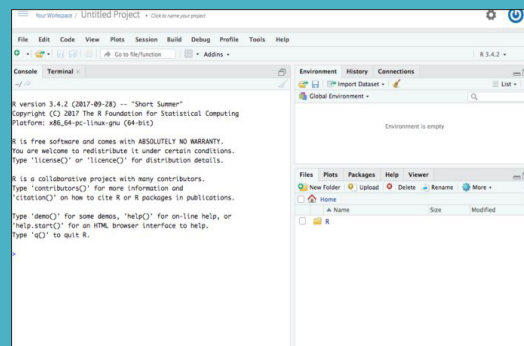
〈화면 1〉과 같은 시작 화면에서 사용자가 프로젝트를 생성할 수 있다. 그러면 〈화면 2〉처럼 친숙한 R스튜디오 인터페이스가 나타난다. 필자는 CRAN과 깃허

브에서 패키지를 설치하는 작업까지 할 수 있었다. 깃(Git)이 내장되지 않은 것을 제외하면 데스크톱 버전과 매우 비슷했다. 좋아하는 패키지 몇 가지를 설치한 다음 테스트해 봤다. ggplot2 그래프는 잘 동작했고, 심지어 확대 버튼을 클릭하고 그래프를 보는 커다란 새창을 띄울 수도 있었다.

단, 현재는 Rstudio.cloud가 공개 플랫폼이다. 즉 생성한 모든 프로젝트를 다른 모든 R스튜디오 클라우드 사용자가 볼 수 있다. 아직은 보완이 필요하지만 R스튜디오 클라우드의 활용 가능성은 충분히 보인다.



화면 1 | Rstudio.cloud의 프로젝트 생성 페이지



화면 2 | Rstudio.cloud의 친숙한 R스튜디오 인터페이스


이터 프레임을 반환하므로 R의 기본 `table()`보다 유용하다. 크로스탭용으로 `xtab()` 함수를 추천한 사람도 많았다. 사용하는 방법은 `xtabs(~df$col1 + df$col2)`이며, `col1`을 행으로 그리고 `col2`를 열로 갖는 도수분포 표를 반환한다.

인수 분해하지 않기. 또 다른 유용한 기능이다. `varhandle` 패키지(<http://bit.ly/2GPKoxc>)의 `unfactor()`로, 이 함수는 여러 인자로 구성된 R 데이터 프레임 열의 “진정한” 클래스(Class)를 감지해 수치나 문자 변수 중 하나로 변환한다.

텍스트 검색. 특정 문자열로 시작하거나 끝나는 텍스트를 검색하는 데 그동안 정규 표현식(Regular Expressions)을 사용했다면 여기 더 쉬운 방법이 있다. 바로 `startsWith()`와 `endsWith()`다. 쟁쟁한 데이터 과학자 중에도 이 함수를 잘 모르는 경우가 허다하다.

패키지를 로드하고 설치 안 됐으면 자동으로 설치하기. 재현 가능 연구(Reproducible Research)의 경우, R 스크립트는 외부 패키지를 간단하게 로드할 수 없다. 먼저 해당 패키지가 사용자 기기에 로드됐는지 확인한 후 그렇지 않으면 설치를 진행한다. 예를 들어 기본 R에서는 다양한 패키지의 로드 여부를 확인하는 `require()`를 사용한 다음 로드되지 않은 경우 패키지를 설치한다. 이때 `pacman` 패키지(<http://bit.ly/2qp3Y8H>)를 이용하면 이 작업을 더 편리하게 할 수 있다. `p_load("package1", "package2", "package3")` 구문을 통해 패키지를 로드하고 사용할 수 없는 경우 해당 패키지를 CRAN에서 설치한다. 깃허브에 있는 패키지에 대한 `p_load_gh()` 버전도 있다.

프로젝트의 홈 디렉터리 찾기. `here` 패키지의 `here()` 함수는 현재 R 프로젝트 작업 디렉터리를 알아낸다. 코드에서 다른 디렉터리를 액세스할 필요가 있거나, 해당 코드가 상이한 디렉터리 구조를 가진 다른 시스템에서도 작동해야 할 때 유용하다.

하나의 명령으로 최소/최대 값 얻기. 벡터에서 최소/최대 값을 찾아야 한다면 기본 R의 `range()` 함수가 유용하다. 가장 작은 값과 가장 큰 값을 가지고 있는 2가지 값 벡터를 반환한다. 도움말 파일에 따르면 `range()`는 숫자 값과 문자 값에 대해 동작한다고 돼 있지만 필자는 데이터 객체에도 사용할 수 있었다.  **ITWORLD**

R에서 지도를 만드는 쉬운 방법 10단계

Sharon Machlis | Computerworld

지도에 표시하고 싶은 지리 위치 정보 데이터가 있는가? 이때 가장 먼저 떠오르는 툴은 아마도 R이 아닐 것이다. 그러나 R 프로그래밍 언어는 새로운 패키지와 표준을 통해 놀라울 만큼 탄탄한 지형 공간 분석 플랫폼으로 성장했다. 예를 들면 R을 이용하면 여러 색깔로 구분된 단계구분도(Choropleth Map)를 쉽게 만들 수 있다. 여기서는 간단한 2인 경선과 3명 이상의 더 복잡한 경선 결선 데이터를 처리하는 방법을 살펴보자.

이 튜토리얼에서는 매핑(Mapping) 패키지 2개를 사용한다. 빠른 스태틱 맵(Static Map)을 위한 tmap과 tmaptools 그리고 인터랙티브 맵을 위한 리플렛(Leaflet: OpenStreetMap 기반으로 인터랙티브 지도를 만들 수 있는 대표적인 자바스크립트 라이브러리)이다. 이들 패키지는 다음과 같이 설치하고 로드할 수 있다. 이미 사용 중인 R 패키지가 있다면 해당 줄을 생략하면 된다.

```
install.packages("tmap")
install.packages("tmaptools")
install.packages("sf")
install.packages("leaflet")
library("tmap")
library("tmaptools")
library("sf")
library("leaflet")
```

1단계: 선거 결과 데이터 확보

여기서는 2016년 뉴햄프셔 민주당 예비선거 결과 데이터를 이용한다. 이 데이터는 엑셀 스프레드시트 형태로 뉴햄프셔주 정부 웹사이트에서 내려받을(<http://bit.ly/2qoAHLb>) 수 있다. 이 프로젝트의 가장 큰 과제는 선거 데이터를 지도 작성에 적합한 포맷으로 만드는 것이다. 실제로 지도를 만드는 것보다 더 어려운 작업이기도 하다. 작업을 간단히 하기 위해 각 도시와 선거구까지 심층 분석을 하는 대신 카운티(County)별 결과에만 집중한다.

문제는 결과 데이터를 보면 카운티, 선거구 또는 주 단위와 관계없이 열 제목이 후보 이름이고 모든 선거구 이름이 포함된 한 개의 열이 있어야 한다는 점이다. 그런데 많은 선거 개표 결과는 자체 행에 각 선거구가 들어있고 후보 결과는 행별로 정리됐다. 뉴햄프셔 공식 결과가 이런 경우다.

여기서는 이 문제를 해결하기 위해 R로 데이터를 가져오기 전에 데이터 순서를 바꾸거나 각 후보의 이름 뒤에 있는 ", d"를 제거하는 등 스프레드시트를 조금 정리했다. 이렇게 해서 첫 번째 열에는 카운티 이름이, 그리고 모든 추가 열에는 후보 이름이 들어가도록 했다. 각 행에는 카운티 선거 결과가 들어 있다. 필자는 데이터 정렬에 지장을 줄 수 있는 맨 밑에 있는 "합계" 항목도 삭제했다.

이 예제를 따라오기 위해 이런 데이터 정리 작업을 직접 할 수도 있다. 아니면, 데이터 파일과 R 코드를 포함해 여기서 사용하는 다른 파일까지 한 번에 내려받으려면 'R을 사용한 지도 작성' 파일 다운로드 페이지(<http://bit.ly/2Ht7vL2>)로 이동하면 된다(무료로 인사이더 등록을 해야 한다). 파일을 내려받아 압축을 풀면 정리된 데이터가 들어있는 NHD2016.xlsx 파일이 있을 것이다.

R 지도 작성 스크립트의 재사용성을 높이기 위해, 스크립트 맨 위에 데이터 파일 이름을 둘 것을 추천한다. 이렇게 하면, 어디에서 파일 이름이 등장하는지 찾기 위해 코드 전체를 찾아다닐 필요 없이 다른 데이터 파일로 쉽게 바꿔 넣을 수 있다. 다음과 같이 R 스크립트 맨 위쪽에 넣을 수 있다.

```
nhdadatafile <- "data/NHD2016.xlsx"
```

단, 필자의 데이터 파일은 R 스크립트와 같은 작업 디렉터리에 있지 않다. 필자는 데이터를 데이터 하위 디렉터리에 저장했다. 따라서 운영체제에서 슬래시를 사용해 시스템에 적합한 파일 경로를 포함했는지 확인해야 한다. 이제 엑셀 파일을 R로 가져오는 몇 가지 패키지가 있다. 여기서는 사용성이 뛰어난 rio를 사용한다. 다음과 같이 설치하면 된다.

```
install.packages("rio")
```

선거 결과 데이터가 시스템에 없다면 다음을 실행해 선거 결과 스프레드시트의 데이터를 'nhdata'라는 변수에 저장하면 된다.

```
nhdata <- rio::import(datafile)
```

이번 선거에는 후보자가 28명이었다. 그렇지만 이번 튜토리얼에서는 데이터 랭글링(Data Wrangling: 원자료(Raw Data)를 또 다른 형태로 수작업으로 전환하거나 매핑하는 과정)보다 지도 작성에 집중하기 위해 군소 후보는 빼고 후보자 2명만 있다고 가정한다. 여기서 힐러리 클린턴과 버니 샌더스다. 다음과 같이 카운티, 클린턴, 그리고 샌더스 열만 선택하면 된다.


```
nhdata <- nhdata[,c("County", "Clinton", "Sanders")]
```

2단계: 어떤 데이터를 지도로 그릴지 결정

이제 지도상에서 정확히 어떤 방식으로 컬러 코드(Color-code)할 것인지 생각해야 한다. 지도의 카운티 색상에 맞는 데이터 열 하나를 골라야 하는데, 우리가 확보한 데이터는 개표 집계 결과뿐이다. 그러나 우리가 원하는 것은 아마도 승리한 후보의 전체 승리 표차, 승리한 후보의 승리 백분율 p 차이, 또는 드물게는 투표수로 표현되는 승리한 후보의 득표 차이 중 하나를 계산하는 것이다. 목표가 전체 주에서 승리하는 것이라면, 인구가 밀집된 카운티에서 5%p 차이로 이겼다는 것이 인구가 더 적은 지역에서 10%p 차이로 승리한 것보다 더 가치 있을 수 있기 때문이다.

이번 데이터를 보면 샌더스가 모든 카운티에서 이긴 것으로 나타났다. 하지만 그렇지 않았더라도, 우리는 샌더스의 '승리 표차' 지도를 그리고 샌더스가 뒤진 카운티에 대해 마이너스 값을 사용할 수 있다. 이번에도 역시 2명의 유력 후보자에 대한 투표 결과만으로 두 후보자의 승리 표차(또는 패배 표차)와 투표 백분율에 대한 열을 추가하자. 전체 코드는 다음과 같다.

```
# Add columns for percents and margins
nhdata$SandersMarginVotes <- nhdata$Sanders - nhdata$Clinton
nhdata$SandersPct <- (nhdata$Sanders - nhdata$Clinton) /
(nhdata$Sanders + nhdata$Clinton) # Will use formatting
later to multiply by a hundred
nhdata$ClintonPct <- (nhdata$Clinton - nhdata$Sanders) /
(nhdata$Sanders + nhdata$Clinton)
nhdata$SandersMarginPctgPoints <- nhdata$SandersPct -
nhdata$ClintonPct
```

3단계: 위치 정보 데이터 확보

원하는 도시, 주 또는 국가에 대한 선거 결과 중 어떤 것을 매핑하든 선거 결과 외에 매핑하려는 지역에 대한 위치 정보 데이터가 필요하다. 이런 지리 데이터는 다양한 포맷이 있지만 여기서는 ESRI가 개발한 셰이프파일(shapefiles)을 사용한다. 가장 널리 사용되는 포맷 중 하나다. 원하는 도시 또는 소도시의 구 수준까지 선거 결과를 지도로 작성하려면 해당 지역 또는 주 GIS 사무실에서 파일을 구해야 한다. 여러 도시, 카운티 그리고 주처럼 더 큰 지역별로 매핑하려면, 미국 통계국에서 셰이프파일을 쉽게 찾을 수 있다.

카운티 단위 뉴햄프셔 매핑 프로젝트를 위해 필자는 통계국의 지도제작 경계 셰이프파일 페이지에서 파일을 내려받았다. 이 파일은 매우 정밀한 분계선이 필요하지 않은 매핑 프로젝트 용으로 설계된 더 작고, 간소화된 파일이다. 엔지니어링 프로젝트나 선거구 조정용 파일은 훨씬 더 큰 편이다. 필자는 해당 웹사이트(<http://bit.ly/2HusWeP>)에서 공식 카운티 파일을 선택한 후 필자의 데이터 서브 디렉터리에 압축을 풀었다. R을 사용하면, 단 하나 또는 더 많은 주에 대한 부분 집합을 쉽게 생성할 수 있다. 이렇게 해서 필자는 다른 주에 대해서도 카운티별로 재

사용할 수 있는 파일을 확보했다. 새로 압축을 풀어난 서브 디렉터리에는 많은 파일이 있다. 필요한 파일은 .shp 확장자를 가진 것이다. 여기서는 이 파일의 이름을 다음과 같이 usshapefile 이라는 변수에 저장했다.

```
usshapefile <- "data/cb_2014_us_county_5m/cb_2014_us_county_5m.shp"
```

몇몇 R 패키지에는 셰이프파일을 R로 가져오는 함수가 있다. 여기서는 tmaptools의 read_shape()를 사용한다. 매우 직관적인 것이 특징이다.

```
usgeo <- read_shape(file=usshapefile, as.sf = TRUE)
```

as.sf = TRUE는 usgeo를 심플 피처 객체(Simple Features Object)가 정의한다는 의미한다. 심플 피처 표준은 최근에 sf 패키지를 사용해 R에 구현됐다. R에서의 GIS 작업을 더 간편하게 만들어 준다. 이제, 지리정보 객체가 지리에 대한 특수한 복잡한 열이 있는 "통상적인" R 데이터 프레임과 비슷해 보인다. as.sf가 FALSE로 설정되면, usgeo는 전체적으로 더 복잡한 구조를 갖게 된다. usgeo 객체가 제대로 지도를 보여주고 있는지 확인하려면 tmap의 쿼크 테마틱 맵(Thematic Map: 주제도) 명령어인 **qtm(usgeo)**를 실행하면 된다. 로드하는 데 시간이 걸리고 작게 보이며 조금 지루하기도 하지만, 지역별로 구분된 미국 지도가 보인다면, 제대로 진행하고 있고 있다는 의미다.

usgeo 데이터 구조를 보기 위해 **str(usgeo)**를 실행하면, sfc_MULTIPOLYGON 정보가 있는 최종 기하학 정보를 제외하면 일반적인 데이터 프레임처럼 보일 것이다. 뉴햄프셔에 국한된 지리공간정보(Geodata)를 추출하는 것은 R에서 다른 유형의 데이터의 부분 집합을 취하는 것과 비슷하다. 여기서는 뉴햄프셔의 FIPS(연방 정보 처리 표준) 주 번호만 있으면 되고, 이 코드는 33이다. 다음은 FIPS 번호 33을 사용해 뉴햄프셔의 데이터를 추출하는 기본 R 명령어이다.

```
nhgeo <- usgeo[usgeo@data$STATEFP=="33",]
```

혹은 dplyr 버전 0.6 이상이 설치되어 있다면, 일반적인 데이터 프레임과 마찬가지로 sf 객체에 대해 dplyr::filter()를 사용할 수도 있다.

```
nhgeo <- filter(usgeo, STATEFP=="33")
```

nhgeo가 제대로 보이는지를 확인하려면 **qth(nhgeo)**로 다시 쿼크 테마틱 맵 함수를 실행하면 된다. 그러면 <화면 1>과 같은 화면이 나타날 것이다. 여전히 단순하지만



화면 1 | 카운티별로 구획된 뉴햄프셔주

카운티별로 구획된 뉴햄프셔주처럼 보이므로, 제대로 된 파일 세트를 확보했다는 것은 알 수 있다.

4단계: 공간 데이터와 선거 결과 데이터 병합

이제 공간 데이터와 선거 결과 데이터를 병합해야 한다. 다른 데이터베이스 조인(Join)이나 병합(Merge)과 마찬가지로 2가지 조건이 있다. 각 데이터 세트가 공유하는 열이어야 하고 같은 개체(Entity)를 똑같은 방식으로 참조하는 레코드여야 한다. 따라서 어떤 카운티를 한 파일에서는 "힐즈버러"로 열거하고 다른 파일에서는 FIPS 번호 "011"로 나열했다면, 일종의 변환 테이블 없이는 이 둘을 연결할 수 없다. 여기서는 원하는 카운티 이름에 대한 nhgeo\$NAME 벡터(Vector)가 카운티 이름에 대한 nhdata\$County 벡터와 같은 이름인지 확인하기 위해 다음과 같이 R 명령을 실행하자.

```
str(nhgeo$NAME)
```

```
# Factor w/ 1921 levels "Abbeville","Acadia",...: 1470 684
416 1653 138 282 1131 1657 334 791
```

```
str(nhdata$County)
```

```
# chr [1:11] "Belknap" "Carroll" "Cheshire" "Coos" "Grafton"
```

반환된 결과를 보면 지형 공간 파일은 카운티를 R 인자로 나열하고 있지만, 데이터에서는 평범한 문자 텍스트다. 이럴 때는 `nhgeo$NAME <- as.character(nhgeo$NAME)` 명령으로 인자를 문자열로 바꾸면 된다. 그다음 데이터 세트 2개를 군 이름별로 정렬하고 비교해보자.

```
nhgeo <- nhgeo[order(nhgeo$NAME),]
```

```
nhdata <- nhdata[order(nhdata$County),]
```

이제 두 카운티 열이 같아졌을 것이다. 다음과 같이 확인하면 된다.

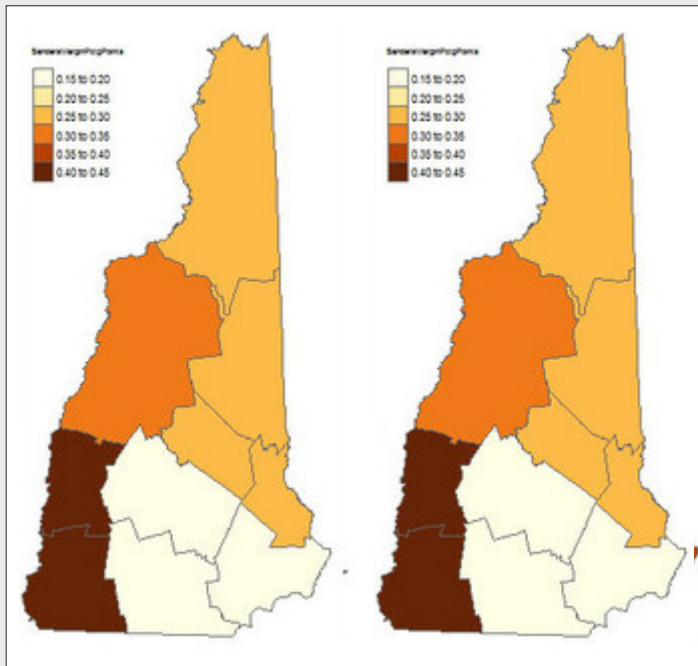
```
identical(nhgeo$NAME,nhdata$County)
```

```
[1] TRUE
```

이제는 두 개 파일을 조인할 수 있다. sp 패키지의 병합 함수는 이런 유형의 작업에 일반적으로 사용하지만, 구문이 직관적이고 2개의 조인 열 이름이 달라도 되므로 tmaptools의 `append_data()`가 꽤 유용하다.

```
nhmap <- append_data(nhgeo, nhdata, key.shp = "NAME", key.data="County")
```

이제 `str(nhmap)` 명령으로 새로운 데이터 구조를 볼 수 있다.



화면 2 | 샌더스 득표 결과

tg_shape() 함수를 사용하면 된다. ggplot2와 유사한 구문을 사용해 채우기, 테두리, 기타 다양한 속성을 설정할 수 있다.

```
tm_shape(nhmap) +
tm_fill("SandersMarginVotes", title="Sanders Margin, Total
Votes", palette = "PRGn") +
tm_borders(alpha=.5) +
tm_text("NAME", size=0.8)
```

코드 블록의 첫 번째 줄은 지도에 그릴 위치 정보 데이터 파일을 설정하고, tm_fill()은 색상 값 매핑을 위해 사용할 데이터 열을 지정한다. "PRGn" 팔레트 변수는 자주색과 녹색에 대한 컬러 브루어(ColorBrewer: 색상 설정) 팔레트이다. 컬러브루어에 익숙하지 않다면, colorbrewer2.org에서 사용가능한 다양한 팔레트를 확인할 수 있다. 컬러브루어 선택항목이 마음에 들지 않는다면 R에 내장된 팔레트(<http://bit.ly/2GWm8p9>)를 사용하거나 이름이 붙어있는 컬러브루어 옵션을 사용하는 대신 수작업으로 원하는 16진수(Hexa) 색상 값을 입력해 설정해도 된다. tm_style_calssic 같은 몇 가지 내장 tmap 테마도 있다.

```
tm_shape(nhmap) +
tm_fill("SandersMarginVotes", title="Sanders Margin, Total
Votes", palette = "PRGn") +
tm_borders(alpha=.5) +
tm_text("NAME", size=0.8) +
tm_style_classic()
```

5단계: 스태틱 맵 생성

데이터를 찾아 올바른 포맷으로 만들어 지형 공간 데이터와 병합하는 어려운 부분이 끝났다. 이제는 카운티별 투표수를 기준으로 샌더스의 득표 차에 대한 간단한 스태틱 맵을 생성해야 한다. qtm(nhmap, "SandersMarginVotes") 명령을 입력하면 된다. 백분율 기준 득표 차는 qtm(nhmap, "SandersMarginPctg-Points") 명령으로 매핑할 수 있다:

이제 <화면 2>처럼 샌더스에게 최고의 승리를 안겨 준 지역 대비 최다 투표자 이점에 가장 가치 있던 지역 간에 약간의 차이가 있음을 알 수 있다. 지도의 색상과 경계 등을 바꾸려면, 다음과 같이

tmap을 사용해 생성한 스테틱 맵을 save_tmap() 함수로 저장할 수 있다.

```
nhstaticmap <- tm_shape(nhmap) +
  tm_fill("SandersMarginVotes", title="Sanders Margin, Total
  Votes", palette = "PRGn") +
  tm_borders(alpha=.5) +
  tm_text("NAME", size=0.8)
save_tmap(nhstaticmap, filename="nhdemprimary.jpg")
```

파일명 확장자는 jpg, svg, pdf, png 등을 선택할 수 있다. 확장자를 지정하면 tmap은 현 지도작성 창 크기를 기본으로, 적합한 파일을 생성한다. 너비, 높이, 해상도(dpi) 등에 대한 인수도 있다. 더 많은 정보는 ?("save_tmap")을 실행하면 볼 수 있다.

6단계: 인터랙티브 맵용 팔레트와 알림창 생성

여기서 우리가 만드는 지도는 오픈소스 자바스크립트 리플렛 매핑 라이브러리에 대한 R 프론트 엔드를 제공하는 R스튜디오의 리플렛 패키지(<http://bit.ly/2v47D1m>)를 이용한다. 덕분에 사용자가 여러 맵 간을 전환할 뿐 아니라 기초 데이터를 보기 위해 클릭도 할 수 있다. 이번 리플렛 맵에는 우리가 이미 가지고 있는 데이터 외에 추가로 두 가지를 더 추가해 보자. 컬러 팔레트와 알림창 내용이다. 먼저 팔레트에 대해 우리가 매핑하는 데이터 범위와 우리가 원하는 컬러 팔레트 종류를 지정한다. 특정 컬러와 색채 계열(Color Scale) 등 총 4가지 유형이 내장돼 있다.

- colorNumeric은 비교적 낮은 품질 색상(Low Color)에서부터 원본 이미지처럼 최적의 색상(High Color)까지 연속적인 범위의 색상에 대한 것으로, 중간에 많은 단계적 차이(Gradation: 그라데이션)가 있는 매우 옅은 파란색에서 짙은 파란색으로 갈 수 있다.
- colorBin은 일련의 수치 데이터를 정확한 브레이크(Break)나 특정 개수의 bin으로 정의된 일련의 이산 bin(Discrete Bin)으로 매핑 한다. "low", "medium", "high" 같은 bin이 대표적이다.
- colorQuantile는 수치 데이터를 각 그룹(분위 수: Quantile)이 같은 개수의 레코드를 가지는 그룹으로 매핑한다. 하위 20%, 차상위 20% 등 소득 수준에 대한 그래프에도 종종 사용한다.
- colorFactor는 유로 존의 일부이거나 그렇지 않은 유럽 국가처럼 비 수치 범주에 사용한다.

이제 다음 구문으로 리플렛 팔레트를 만들어보자.

```
mypalette <- colorFunction(palette = "colors I want", domain =
mydataframe$dataColumnToMap)
```

colorFunction은 colorNumeric()이나 colorFactor 같은 4가지 색채 계열 유형 중 한 가지이며 "colors I want"는 색상에 대한 벡터이다. 여기서는 약간 변화를 주기 위해 샌더스 지도의 정반대인 힐러리 클린턴이 가장 많이 득표한 지도에 표시해 보자. 클린턴의 투표율을 매핑하기 위

해 다음 팔레트를 사용한다.

```
clintonPalette <- colorNumeric(palette = "Blues", domain=nhmap$ClintonPct)
```

"Blues"는 컬러브루어의 파란색 범위이고 domain은 색채 계열의 데이터 범위다. colorNumeric은 후보 이름처럼 특정 범주가 아니라 수치 변수를 사용하는 연속적인 색상 범위를 지정한다는 것을 의미한다. 여기서는 알림창도 추가해보자. 클릭하거나 탭해서 기초 데이터를 볼 수 있어야 인터랙티브 지도라고 말할 수 있을 것이다. 참고로 여기서는 알림창 텍스트를 표시하기 위해 0.7865 같은 십진수를 78.7% 같은 백분율로 표시한다. 간단한 공식을 직접 작성해도 되지만 더 간단하게 하기 위해 scales 패키지의 percent() 함수를 사용한다. 아직 설치하지 않았다면 다음과 같이 설치하고 로드하면 된다.

```
install.packages("scales")
library("scales")
```

알림창의 내용은 HTML과 R 변수의 간단한 결합이다.

```
nhpopup <- paste0("County: ", nhmap$County,
  "Sanders ", percent(nhmap$SandersPct), " - Clinton ",
  percent(nhmap$ClintonPct))
```

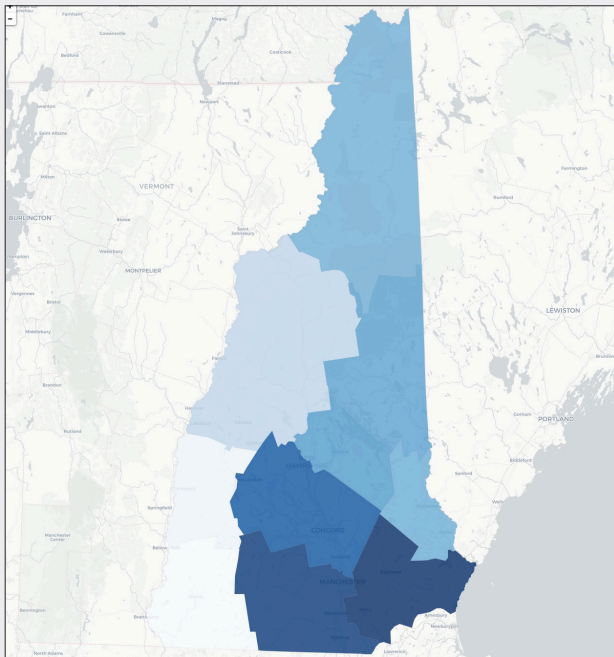
paste0 함수는 텍스트와 변수값을 하나의 문자열로 조인하는 연결(Concatenate) 함수다. 카운티 이름 열을 NAME보다는 County로 하는 것이 좋으므로, dplyr의 rename() 함수를 사용해 `nhmap <- rename(nhmap, County = NAME)`처럼 바꿀 수 있다.

7단계: 인터랙티브 맵 생성

다음은 지도를 작성하는 코드다.

```
leaflet(nhmap) %>%
  addProviderTiles("CartoDB.Positron") %>%
  addPolygons(stroke=FALSE,
    smoothFactor = 0.2,
    fillOpacity = .8,
    popup=nhpopup,
    color= ~clintonPalette(nhmap$ClintonPct)
  )
```

코드를 자세히 살펴보자. `leaflet(nhmap)`은 리플렛 맵 객체를 생성하고 데이터 소스로 nhmap을 설정한다. `addProviderTiles("CartoDB.Position")`은 배경 맵 타일을 카토디비(CartoDB)의 멋진 포지트론(Positron) 디자인으로 설정한다. 다른 것을 선택하려면 무료 배



화면 3 | R과 R스튜디오 리플렛 패키지에서 생성한 기초적인 인터랙티브 맵

경 타일과 이것이 어떻게 생겼는지 깃허브(<http://bit.ly/2Hu22DU>)에서 목록을 볼 수 있으니 참고하면 된다.

나머지는 `addPolygons()` 함수가 처리한다. 카운티 모양을 지도에 그려 넣고 그에 따라 색을 입히는 작업이다. `stroke=FALSE`는 카운티를 둘러싼 경계선이 없다는 의미이고, `fillOpacity`는 색상의 불투명성을 설정하며, `popup`은 알림창의 내용을 설정한다. 또한, `color`는 팔레트와 해당 색상에 어떤 데이터가 매핑되는지를 설정한다. 팔레트 이름 앞에 물결표시(~)가 필요한 이유는 뚜렷하지 않은데 일단 함수 포맷은 그렇다. 실행한 결과는 <화면 3>과 같다. 링크(<http://bit.ly/2GVK14c>)를 클릭하면 실제 동작하는 지도로 이동하는데, 거기서 지역을 클릭하면 기초 데이터를 볼 수 있다.

지도와 함께 “일관되지 않은 데이터(Datum)”에 대한 오류 메시지가 표시될 수도 있다. 투사도(Projection) 때문이다. 투사도는 복잡한 개념이지만 기본적으로 데이터 투사도가 하부 맵 타일의 투사도와 일치하는 것이 좋다. 이렇게 해야 3차원 구면의 점을 2차원으로 나타내는 과정에서 일관성을 보장하기 때문이다. 이 문제를 해결하려면 다음 명령으로 오류 메시지에 권고된 투사도를 추가하면 된다.

```
nhmap_projected <- sf::st_transform(nhmap, "+proj=longlat +datum=WGS84")
```

그리고 나서 `leaflet(nhmap_projected) %>%`를 지도 작성 코드의 첫 줄에 추가해 실행하면 된다. 리플렛 패키지는 범례 추가와 지도 도면 층을 표시하거나 감추는 기능을 포함해 우리가 아직 사용하지 않은 여러 가지 기능을 지원한다. 두 가지 기능 모두 2016년 공화당 예비 선거처럼 최소 3개의 선택사항을 가지고 있는 범주를 매핑할 때 매우 유용하다.

8단계: 다중 계층 지도(Multi-layer Map)용 팔레트 추가

이제 최상위 후보 3명의 사우스캐롤라이나 공화당 선거 결과를 살펴보자. 일단 데이터에 대해 별도의 데이터 랭글링은 하지 않는다. 카운티별 교육 수준에 대한 데이터는 인구 조사국에서, 선거 결과는 사우스캐롤라이나주 선거 관리위원회에서 내려받았다. 앞서 필자가 미리 정리해 둔 프로젝트 파일을 내려받았다면 후보자 투표율을 추가하고 모든 데이터를 사우스캐롤라이나 셰이프파일에 조인하기 위해 R 코드뿐 아니라 기초 데이터도 확인할 수 있다.

복수 후보 경선에는 데이터가 많으므로 “누가 이겼는가”를 떠나서 어떤 색을 사용할지 선택하는 것이 중요하다. 여기서는 각 카운티의 승자를 보여주는 맵 계층과 트럼프, 루비오, 그리고 크루즈 등 최상위 3명의 후보자에 대한 맵 계층 그리고 마지막으로 최소 학사 학위 이상을 소지

한 성인 비율을 보여주는 맵 계층을 만들어 본다. 교육 수준을 별도 계층으로 선택한 이유는 트럼프의 지지율과 상관관계가 있다는 보도가 나왔기 때문이다. 이를 매핑하면 실제로 어떤 패턴이 있는지 확인할 수 있을 것이다.

여기서는 컬러 팔레트를 만들 때 후보자 3명 모두에 대해 같은 수치 척도를 사용하기로 했다. 각 후보자의 최소값과 최대값에 대해 색 농도(Color Intensity)를 조정하면, 10~18%를 얻은 후보자는 45~52%를 얻은 후보자와 같은 색 농도를 가지게 된다. 이는 지고 있는 후보자의 역량에 대한 잘못된 인상을 줄 수 있으므로, 여기서는 먼저 트럼프, 루비오, 크루즈의 종합 카운티 선거 결과에 대한 최소값과 최대값을 계산했다.

```
minpct <- min(c(smap$Donald.J.TrumpPct, smap$Marco.
RubioPct , smap$Ted.CruzPct))
maxpct <- max(c(smap$Donald.J.TrumpPct, smap$Marco.
RubioPct , smap$Ted.CruzPct))
```

이제 서로 색은 서로 다르지만 같은 색 농도 범위를 사용해 각 후보자에 대한 팔레트를 생성할 수 있게 됐다.

```
trumpPalette <- colorNumeric(palette = "Purples",
domain=c(minpct, maxpct))
rubioPalette <- colorNumeric(palette = "Reds", domain =
c(minpct, maxpct))
cruzPalette <- colorNumeric(palette = "Oranges", domain =
c(minpct, maxpct))
```

다음과 같이 승자와 교육 계층에 대한 팔레트도 추가하자.

```
winnerPalette <- colorFactor(palette=c("#984ea3",
"#e41a1c"), domain = smap$winner)
edPalette <- colorNumeric(palette = "Blues",
domain=smap$PctCollegeDegree)
```

마지막으로, 카운티 이름과 승자 후보, 각 후보의 득표율, 그리고 학사 학위를 소지한 인구 비율을 보여주는 기본적인 알림창을 만든다.

```
scpopup <- paste0("County: ", smap@data$County,
"Winner: ", smap@data$winner,
"Trump: ", percent(smap$Donald.J.TrumpPct),
"Rubio: ", percent(smap$Marco.RubioPct),
"Cruz: ", percent(smap$Ted.CruzPct),
"Pct w college ed: ", smap$PctCollegeDegree, "% vs state-
```

```
wide avg of 25%")
```

끝으로, 매핑에 앞서 뉴햄프셔 지도에서 했던 것과 같은 투사도를 추가해야 한다. 다음 코드를 이용해 투사도를 scmap 객체에 추가하면 된다.

```
scmap <- sf::st_transform(scmap, "+proj=longlat +datum=WGS84")
```

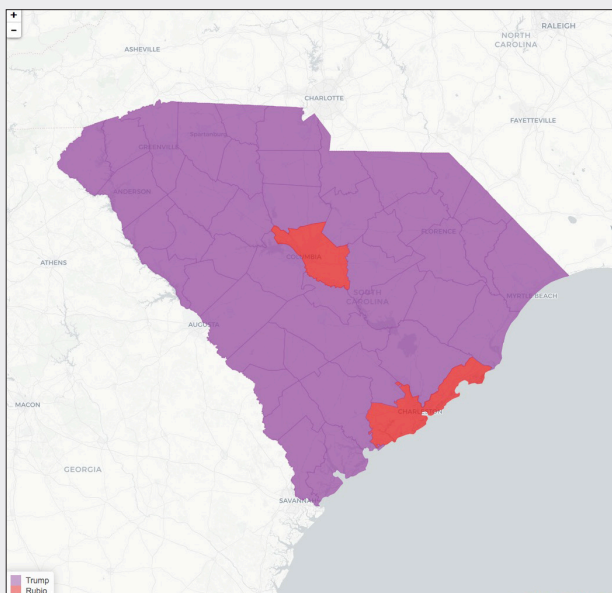
다음 코드는 승자에 대한 기본적인 지도를 카운티별로 보여준다. 사우스캐롤라이나에 있는 카운티에서는 트럼프와 루비오가 승리했으므로, 이 두 후보의 색깔과 이름만 보여주는 범례를 둘 수 있다.

```
eaflet(scmap) %>%
  addProviderTiles("CartoDB.Positron") %>%
  addPolygons(stroke=TRUE,
    weight=1,
    smoothFactor = 0.2,
    fillOpacity = .75,
    popup=scpopup,
    color= ~winnerPalette(scmap$winner),
    group="Winners"
  ) %>%
  addLegend(position="bottomleft", colors=c("#984ea3",
    "#e41a1c"), labels=c("Trump", "Rubio"))
```

최종 결과는 <화면 4>와 같다. 링크(<http://bit.ly/2v7xmWn>)를 클릭하면 실제 작동하는 인터랙티브 맵으로 연결된다. 이 지도는 2명 이상의 후보에 대한 데이터를 가지고 있다. 최상위 후보 3명에 대한 결과와 학사 학위를 가지고 있는 인구 비율을 보려면 아무 카운티나 클릭하면 된다.

9단계: 맵 계층과 컨트롤 추가

계층 컨트롤(Control) 기능을 가지고 있는 다중 계층 지도는 이전의 맵과 똑같이 시작하고 한 가지만 추가된다. 바로 그룹 이름이다. 이 지도는 각 계층이 하나의 그룹이지만 다중 계층을 함께 표시하거나 감출 수 있다. 다음 단계는 코드를 완성하는 계층 컨트롤과 함께 각 후보에 대해 추가 폴리곤 계층(Polygon Layer)과 대학 교육에 대한 최종 계층을 추가하는 것이다. 이번에는 맵을 변수에 저장한 다음 표시해보자. 구체적인 코드는 다음과 같다.



화면 4 | 2명 이상의 후보에 대한 데이터가 포함된 인터랙티브 맵

```

scGOPmap <- leaflet(scmap) %>%
addProviderTiles("CartoDB.Positron") %>%
addPolygons(stroke=TRUE,
weight=1,
smoothFactor = 0.2,
fillOpacity = .75,
popup=scpopup,
color= ~winnerPalette(scmap$winner),
group="Winners"
) %>%

addLegend(position="bottomleft", colors=c("#984ea3",
"#e41a1c"), labels=c("Trump", "Rubio")) %>%

addPolygons(stroke=TRUE,
weight=1,
smoothFactor = 0.2,
fillOpacity = .75,
popup=scpopup,
color= ~trumpPalette(scmap$Donald.J.TrumpPct),
group="Trump"
) %>%

addPolygons(stroke=TRUE,
weight=1,
smoothFactor = 0.2,
fillOpacity = .75,
popup=scpopup,
color= ~rubioPalette(scmap$Marco.RubioPct),
group="Rubio"
) %>%

addPolygons(stroke=TRUE,
weight=1,
smoothFactor = 0.2,
fillOpacity = .75,
popup=scpopup,
color= ~cruzPalette(scmap$Ted.CruzPct),
group="Cruz"
) %>%

addPolygons(stroke=TRUE,
weight=1,
smoothFactor = 0.2,

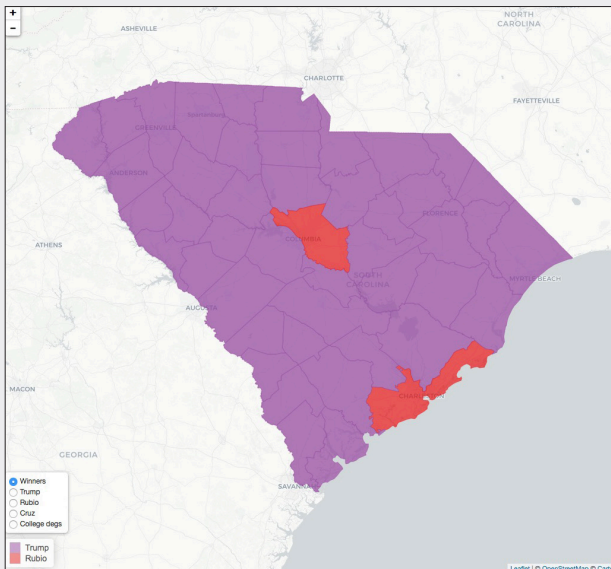
```

```

fillOpacity = .75,
popup=scpopup,
color= ~edPalette(scmap$PctCollegeDegree),
group="College degs"
) %>%

addLayersControl(
  baseGroups=c("Winners", "Trump", "Rubio", "Cruz", "College
  degs"),
  position = "bottomleft",
  options = layersControlOptions(collapsed = FALSE)
)

```



화면 5 | 다중 계층이 있는 인터랙티브 맵

이제 scGOPmap 명령으로 맵을 표시해보자. < 화면 5>와 같은 다중 계층이 있는 인터랙티브 맵이 표시될 것이다. 링크(<http://bit.ly/2HdaTMJ>)를 클릭하면 실제 작동하는 맵으로 이동한다. 표시할 계층을 변경하려면 왼쪽 아래에 있는 라디오 버튼을 클릭하면 된다. `addLayersControl`은 2가지 유형의 그룹을 가진다. 앞서 사용한 것처럼 한 번에 하나의 계층만 볼 수 있는 `baseGroups`, 그리고 다중 계층을 한 번에 볼 수 있고 각 계층을 독립적으로 숨길 수 있는 `overlayGroups`이다.

10단계: 인터랙티브 맵 저장

R마크다운(RMarkdown, <https://rmarkdown.rstudio.com>) 또는 샤이니(Shiny, <http://shiny.rstudio.com>)에 익숙하다면, 리플렛 지도를 R마크다운 문서나 샤이니 웹 애플리케이션에 내장하는 방법을 알 것이다. 지도를 웹사이트 등의 HTML 페이지로 사용하고 싶다면, HTML 위젯(htmlwidget) 패키지의 `saveWidget()` 함수를 사용해 리플렛 지도를 저장하면 된다.

```

# install.packages("htmlwidgets")
library("htmlwidgets")
saveWidget(widget=scGOPmap, file="scGOPprimary.html")

```

또한, `selfcontained=FALSE` 인수를 사용하고 종속 파일이 있는 서브 디렉토리를 선택해 jQuery와 리플렛 자바스크립트 코드 같은 외부 자원을 사용해 지도를 저장할 수도 있다.

```

# install.packages("htmlwidgets")
save(widget=scGOPmap2, file="scGOPprimary_withdependencies.

```

```
html", selfcontained=FALSE, libdir = "js")
```


추가: R 맵에 주소 검색 넣기

leaflet.extras(<http://bit.ly/2qrpcDu>) 덕분에 코드 한 줄로 지도에 주소 검색을 추가할 수 있다. 주 수준의 지도라면 소도시 또는 도로 이름 정도면 충분할 것이다. **%>% addSearchOSM()** 명령을 맵 코드 끝에 추가하면 된다. 이렇게 하면 맵에 돋보기가 추가된다. 지도를 클릭해 확대하고 싶은 장소를 입력할 수 있다. 단, 구글 지도에서 검색하는 것만큼 멋지지는 않다.

또 한 가지 제약은 OSM(Open Street Map: 오픈 스트리트 맵) 검색 데이터 소스가 492 Old Connecticut Path, Framingham, MA와 같은 주소를 인식하지 못한다는 것이다. 집이나 직장 전체 주소가 아니라 도로 이름, 도시 이름, 또는 "Old Connecticut Path, Framingham"이나 "Framingham High School"처럼 지명을 입력해야 한다. 그 후에 가용 OSM 데이터 중 사용할 수 있는 선택사항을 고르기 위해 드롭다운 목록이 표시될 때까지 기다려야 한다. 마지막으로, 긴 거리 또는 대도시의 경우 지도가 보고자 하는 곳까지 확대되지 않을 수 있다.

주소 수준의 검색이 필요한 경우를 위해 leaflet.extras는 구글 지도와 Bing 검색을 지원한다. 이런 서비스를 사용하려면 API 키가 필요하다. 구글 검색의 경우 구글 지도 지오매핑 API 키(<http://bit.ly/2JBNCx>)가 필요하다. 필자는 구글 개발자 콘솔에 구글 지도 자바스크립트 API도 활성화해 놓았다. leaflet.extras의 addSearchGoogle() 함수는 다음과 같은 시스템 환경 변수에서 지오코딩 API 키 GOOGLE_MAP_GEOCODING_KEY를 찾는다. 이를 설정하려면 다음과 같이 하면 된다.

```
Sys.setenv(GOOGLE_MAP_GEOCODING_KEY = "YourKeyHere")
```

그다음 맵 코드에 **%>% addSearchOSM()**를 추가하는 대신, **%>% addSearchGoogle()** 명령을 추가하면 된다. 더 자세한 정보는 패키지 개발자인 바스카 V. 카람벨카의 리플렛 검색 샘플 코드 페이지(<http://bit.ly/2GU4EtP>)를 참고하면 된다. 여기까지 잘 따라왔다면 이제 R을 사용해 등치 지역도(choropleth map)을 만들 수 있을 것이다. 경도/위도 기준점이 있는 지도를 생성하려면, 데이터 시각화와 분석을 위한 유용한 새 R 패키지(<http://bit.ly/2GUMdFh>)를 활용하는 것을 추천한다.  **ITWORLD**

‘파이썬 vs. R’ 데이터 과학자의 마음을 훔쳐라

Peter Wayner | InfoWorld

수 페타바이트에 이르는 데이터를 능숙하게 뒤지는 초고수가 있다. 하드 디스크가 돌아가는 소리 속에는 어떤 신호가 있는 것처럼 들린다. 마치 숫자로 이뤄진 세계에 지적 생명이 존재하는 것 같은 느낌도 든다. 이들 초고수는 방대한 데이터 속에서 새로운 수익이 될 인사이트를 찾아 헤맨다. 만약 이 작업이 나에게 떨어졌다면 어떻게 해야 할까? 며칠 내에 거대한 디지털 잡동사니를 뒤지고 뒤져 유용한 뭔가를 찾아 임원에게 보고해야 한다. 이때 개발자가 선택할 수 있는 것은 2가지다. 바로 R과 파이썬(Python)이다.

이미 시장에는 수많은 데이터 크런칭 솔루션이 있다. ‘비즈니스 인텔리전스’ 혹은 ‘데이터 시각화’ 같은 그럴듯한 이름이 붙어있다. 어떤 솔루션이 원하는 기능을 지원한다면 망설이지 않고 그냥 선택하면 된다. 그러나 솔루션이 해주지 않는 작업을 해야 한다면 결국 직접 코드를 작성하는 수밖에 없다. 또한, 데이터가 깨끗하게 준비돼 있다면 포괄적인 서비스 툴을 사용하면 된다. 그러나 이런 툴은 모든 부분이 완벽하지 않을 경우 문제를 일으키거나 삼킨 데이터를 제대로 소화하지 못하고 토해내는 문제가 있다.

파이썬과 R의 차이를 아는 것은 그래서 중요하다. 대부분 사고방식 측면에서 구별된다. 하나는 유닉스 스크립터가 개발해 통계학자, 빅데이터 전문가와 소셜 과학자가 사용하는 포괄적인 서비스 언어다. 다른 하나는 통계학자, 빅데이터 전문가, 소셜 과학자가 직접 설계하고 만든 데이터 분석용 툴이다. 사용하는 계층은 거의 똑같지만 접근 방식은 전혀 다르다. 하나는 유용한 라이브러리가 많은 범용 툴이고 다른 하나는 빅데이터 분석 전용으로 개발됐다. 이 중 어떤 것을 선택해야 할까? 그 해답을 찾기 위해 두 언어를 자세히 비교해 보자.

파이썬은 전처리가 쉽다

데이터 분석의 50%는 데이터를 정리하는 일이다. 일부는 그 비중이 99%라고 말하기도 한다. 정확한 수치야 어떻든 필요할 때 여러 가지 작업을 할 수 있는 포괄적인 서비스 언어로 데이터를 정리하는 것이 좋다. 파이썬은 포괄적인 서비스 명령형 언어이므로, 처음 사용하는 개발자도 그 구조와 접근 방법이 친숙하게 느껴진다. 손쉽게 새 함수와 새 계층을 추가해 데이터를 나누고 정리할 수 있다. 로컬 저장, 웹 서비스 접근 또는 컴퓨터 프로그램이 일상적으로 수행하는 다른 요소가 필요하다면 별 어려움 없이 해당 요소를 포함할 수 있다. 언어이기 때문에 가

능한 특징이다.

R은 어떤 데이터도 전처리할 수 있다

이처럼 파이썬을 사용하면 전처리(preprocessing)가 쉽지만 R 역시 데이터를 정리하는 용도로 사용할 수 있다. 이때 어느 언어를 사용해도 무방하다. 사실 많은 경우 데이터 순화 루틴과 분석 루틴을 혼합하는 것은 구조적으로 좋지 않다. 분리하는 편이 낫다. 어차피 둘을 분리한다면 그냥 선호하는 언어를 사용하는 것이 좋다. 파이썬일 수도 있고 자바, C, 어셈블리 코드도 괜찮다. 데이터베이스 내에서 또는 다른 스토리지 계층 내에서 데이터를 전처리할 수도 있다. R은 이런 언어를 전혀 가리지 않는다.

파이썬엔 수많은 라이브러리가 있다

파이썬은 인기가 많다. 리포지토리의 통계 수치에서도 확인할 수 있다. 파이썬 패키지 인덱스(PyPi, <https://pypi.python.org/pypi>)의 패키지 수는 10만 개가 넘는다. 지금 이 순간에도 계속 늘어나고 있다. 사실은 이 숫자조차 빙산의 일각에 불과하다. 깃허브부터 소셜 과학 웹사이트까지 곳곳에 코드가 있다. PyPi를 제외해도 좋은 파이썬 코드가 많이 있고 거의 모두가 오픈소스이므로 이를 통해 한결 수월하게 작업을 처리할 수 있다.

R에는 수많은 통계 분석용 라이브러리가 있다

R 역시 패키지가 있다. 종합 R 아카이브 네트워크(CRAN, <https://cran.r-project.org>) 패키지 수가 10만 개가 넘고 파이썬과 마찬가지로 계속 늘어나고 있다. 이러한 패키지의 목적은 하나다. 데이터의 통계 분석이다. 파일 시스템 검사나 서버 유지보수를 위한 것이 아니다. 그런 작업은 R의 소관이 아니다. 모든 오픈소스 리포지토리가 그렇듯 제대로 작동하지 않는 경우가 가끔 있지만 코드 대부분은 통계학자가 작성하고 검토했다.

파이썬은 계속 발전중이다

프랑스에서는 '르 위켄(le weekend)'이라고 말해도 모두가 '주말'이라고 잘 알아듣는다. 살아있는 언어란 그런 것이다. 파이썬은 프랑스어와 마찬가지로 계속 발전하고 더 좋아지고 있다. 버전 2.3에서 3.0으로 건너뛰면서 일부 예전 코드가 작동하지 않게 됐지만, 많은 파이썬 애호가들이 그 정도의 혼란은 감수할 가치가 있다고 평가한다. 비록 과거의 코드를 단절시킨다 해도 살아있는 코드는 계속 발전하는 것이다.

살아있는 언어는 곧 사람들이 사용하고 개선한다. 이는 더 많은 오픈소스 코드와 더 많은 솔루션으로 연결된다는 의미다. 표준의 변경과 코드의 단절은 인기 있고 발전하는, 살아있는 언어를 사용할 때 감수해야 할 비용이다.

R은 순수함을 유지한다

R이 변화하지 않는다고 말하는 것은 옳지 않은 지적이다. 사실 R은 통계 소프트웨어 'S'를 바탕으로 만



들어졌다. 구문 유효 범위(lexical scoping)를 적용해 대규모 코드 베이스를 더 개선한 것이 바로 R이다. 그래서 대부분의 경우 R 인터프리터에서 S를 실행할 수 있다. 파이썬처럼 코드 베이스가 2.3이나 3.0이나를 따져야 할 정도의 중대한 패러다임 변화는 없다. 단지 시간이 지날수록 더 익숙해지고 문제 발생 가능성이 줄어든 뿐이다. R 역시 '살아있는 언어'이므로 미래를 장담할 수는 없지만 변화의 폭이 크거나 급진적이지는 않을 것으로 보인다.

파이썬은 다른 언어가 할 수 있는 작업을 다 한다

파이썬은 개발자가 원하는 모든 작업을 할 수 있도록 개발자가 설계한 범용 언어다. 그러나 이 말이 곧 '튜링 완전(Turing-complete)'하다는 의미는 아니다. '게임 오브 라이프(Game of Life, <http://bit.ly/2Hg2hEP>)'가 튜링 완전하지만 이걸로 피보나치 수열을 계산하는 사람은 없다는 것과 비슷한 의미다. 보통 특정 작업을 수행해야 한다면 이를 위해 선택할 수 있는 도구가 다양하기 마련이다.

대신 파이썬은 이 작업을 쉽게 하도록 개발된 것이 특징이다. 다량의 코드로 채워진 실제 프로젝트를 위해 고안됐다. 프로젝트를 시작하는 시점에서 소소한 세부 사항을 정리하기 위해 코드 몇 줄을 쓸 때는 얼마나 유용한지 드러나지 않을 수 있다. 그러나 나중에 이 몇 줄이 몇천 줄이 되고 전체가 매우 복잡한 '스파게티 코드(spaghetti code)'가 되면 파이썬의 이러한 특성이 빛을 발한다. 파이썬은 규모가 큰 프로젝트에 적합하다. 언젠가는 초보 개발자에게도 파이썬의 이 기능이 필요할 날이 올 것이다.

R은 통계를 잘 처리한다

R은 통계 분석을 위해 만들어졌다. 지금 해야 할 일이 통계 분석이라면 다른 건 검토할 필요도 없다. 그 모든 작업이 그렇듯 딱 맞는 툴을 선택하는 것이 중요하다. 망치 대응으로 렌치를 쓸 수는 있지만 망치가 필요한 일에는 망치를 쓰는 것이 최선이다.

파이썬엔 명령줄이 있다

마우스로 가리키고 클릭하면서 자란 사람은 명령줄에 적응하기가 쉽지 않다. 그러나 익숙해지면 좋은 키보드와 조화를 이룬 명령줄의 힘과 표현력을 깨닫게 된다. 명령줄이라는 언어의 조합은 정말 놀랍다. 십여 개 메뉴 페이지를 거치며 마우스로 클릭해서 할 일을 문자열 하나로 할 수 있다. 파이썬은 후자의 세계에 속한다. 명령줄을 위해 태어났고 명령줄에서 힘을 발휘한다. 모양새는 터무니없이 뒤쳐져 보이지만 매우 효율적이고 매우 강력하다.

R에도 명령줄이 있고, 추가로 R스튜디오도 있다

R 역시 일종의 명령줄을 중심으로 만들어진 언어다. 물론 명령줄 안에서 이것저것 기능을 추가하긴 했다. 그러나 많은 사람이 모든 요소를 집어넣어 잘 포장한 두 가지 환경 즉 R스튜디오(RStudio, <https://www.rstudio.com>)와 R 커맨더(R commander, <http://www.rcommander.com>) 내에서 작업한다. 명령줄 외에 데이터 편집기와 디버깅 지원, 그래픽을 위한 창도 별도로 있다. 최근에는 파이썬 진영에서도 이클립스, 비주얼 스튜디오 등 기존 IDE를 통해 R의 이런 장점을 따라잡기 위해 노력하고 있다.

파이썬에는 웹이 있다

파이썬 개발을 위한 웹사이트가 있다는 것은 어쩌면 자연스러운 현상이다. 파이썬 자체가 유닉스 웹 서버와 함께 진화한 스크립팅 언어이기 때문이다. 이런 웹사이트에는 로데오(Rodeo, <http://rodeo.yhat.com>)와 주피터(Jupyter, <http://jupyter.org>)가 대표적이고, 앞으로 더 많이 생길 것이다. 인터프리터로 포트 80을 쉽게 링크할 수 있으므로 파이썬은 웹에서 잘 작동한다. 물론 스칼라, 줄리아, 꼭 원한다면 R과 같은 다른 언어에서도 주피터를 사용할 수 있다. 그러나 어느 언어가 가장 유리한지는 'Jupyter'라는 철자만 봐도 알 수 있을 것이다.

R은 레이텍과 잘 통한다

R을 사용해 데이터를 분석할 때 많은 사람이 레이텍(LaTeX)을 이용해 데이터에서 발견한 인사이트를 문서로 작성한다. 그리고 이처럼 데이터 분석과 문서 작성을 결합한 결과물이 바로 '스위브(Sweave)' 시스템이다. 이를 이용하면 데이터를 분석하고 그래프를 만드는 R 명령을 결과 보고서와 통합할 수 있다. 모두 한곳에서 처리되므로 데이터 손상이나 캐시 문제 발생 위험이 최소화된다. 버튼 하나만 누르면 소프트웨어가 데이터를 다시 분석하고 그 결과를 최종 문서에 넣어준다.

둘 다 사용한다면 어떨까

지금까지 파이썬과 R을 비교해 봤다. 둘 중 어느 것을 사용해야 할까? 고민할 필요 없다. 2가지 장점을 모두 사용하면 된다. 실제로 많은 데이터 과학자가 그렇게 하고 있다. 데이터 수집의 첫 단계는 파이썬으로 처리할 수 있다. 그다음 R에 내장된, 충분한 테스트와 최적화를 거친 통계 분석 루틴에 이 데이터를 넣으면 된다. R을 일종의 파이썬 라이브러리로 사용하거나 파이썬을 R을 위한 전처리 라이브러리로 사용하는 식이다. 결론적으로 특정 계층에 가장 잘 맞는 언어를 선택해 케이크처럼 쌓아 올리면 된다. 물론 R이 케이크이고 파이썬이 그 위에 올리는 설탕가루인지, 아니면 그 반대인지는 각자 선택의 몫이다. 