* 21 페이지 중간에서

**호환성 문제도 없을 것이다. 뒤에 다음을 추가**

다만 한글을 출력하는 예제 코드는 MS-Windows에서는 아무 설정없이 동작하지만 Mac OS X에서는 다음과 같이 한글 폰트를 지정한 후 실행해야 한다. (자세한 내용을 9장의 **9.2 한글 그래픽 환경설정하기**를 참고하기 바란다.)

> par(family = ‘AppleGothic’)

* 23 페이지 마지막에

**다음을 추가**

1.6.4 버전업에 대한 대처

R과 3rd party 패키지는 버전업이 잦은 편이다. 그만큼 새로운 기술과 추가적인 기능을 능동적으로 수용하고 버그 등의 문제점을 재빨리 해결하는 장점이 있다. 그러나 반대로 버전업은 이전에 개발해 놓은 프로그램과의 호환성에 충돌이 생기는 경우를 초래하기도 한다.

본서는 R을 이용한 데이터 시각화의 광범위한 영역을 다루기 때문에, 본문에 게시한 예제의 환경과 다른 버전의 패키지가 설치된 독자의 환경에서는 일부 영역에서 예기치 않은 결과가 초래할 수도 있을 것이다.

필자는 오랜 시간 동안 R을 사용하면서 버전의 상이함으로 인해서 과거에 작성했던 프로그램 코드가 정상적으로 수행되지 않는 경험을 간혹 하게 되었다. 이럴 경우에는 그 오류 메시지를 Google에서 검색하여 효과적으로 대체하는 편이다. 다만, NAVER 등과 같은 국내 검색 환경에서는 상대적으로 해답을 찾기가 쉽지 않았다. 책임감이 없는 이야기로 들릴 수 있겠지만, 버전의 상이함으로 문제가 발생할 경우에는 검색을 통해서 해결하기 바란다.

* 115 페이지 중간 소스에서

> title(“len ~ sup == ”VC”") 을

> title("len ~ supp == \"VC\"") 으로 수정

> title(“len ~ dose, subset = sup == "VC"") 을

> title("len ~ dose, subset = supp == \"VC\"") 으로 수정

> title(“len[supp == \”VC\"] ~ dose[supp == "VC"]") 을

> title("len[supp == \"VC\"] ~ dose[supp == \"VC\"]") 으로 수정

* 124 페이지 중간 소스에서

> boxplot(x, log = “y”, main = "log = "y"") 을

> boxplot(x, log = "y", main = "log = \"y\"") 으로 수정

> boxplot(x, log = “x”, main = "log = "x"") 을

> boxplot(x, log = "x", main = "log = \"x\"") 으로 수정

* 235 페이지 중간 소스에서

plot(c(-1, 1), c(-1, 1), type=’n’, axes=F, xlab=", ylab=") 을

plot(c(-1, 1), c(-1, 1), type='n', axes=F, xlab="", ylab="") 으로 수정

* 236 페이지 중간 소스에서

plot(c(-1, 1), c(-1, 1), type=’n’, axes=F, xlab=", ylab=") 을

plot(c(-1, 1), c(-1, 1), type='n', axes=F, xlab="", ylab="") 으로 수정

* 416 페이지 상단 맨 첫 줄

**> dim(movies) 윗 줄 앞에 다음을 추가함**

ggplot2의 저작자인 Hadley Wickham은 movies라는 데이터 프레임이 용량이 크고 효용성이 떨어진다고 판단하여 2.0.0 이상 버전에서 제외하였다. 그리고 새로운 패키지 ggplot2movies에 이 데이터 프레임을 옮겨 놓았다. 그러므로 ggplot2 2.0.0 이상의 버전을 설치한 독자는 다음의 스크립트처럼 ggplot2movies 패키지를 설치해야 한다.

> # ggplot2 2.0.0 이상에서는 movies 데이터프레임이 빠지고, ggplot2movies 패키지로 이동함

> major <- strsplit((installed.packages()["ggplot2", "Version"]), "\\.")[[1]][1]

> if (as.numeric(major) > 1) {

if (!"ggplot2movies" %in% installed.packages()[, 1])

install.packages("ggplot2movies")

library(ggplot2movies)

}

* 429 페이지 하단 소스에서

> g + stat\_binhex(bin=20) 을

> g + stat\_binhex(bins=20) 으로 수정

* 616 페이지 하단 소스에서

**소스를 다음으로 대체함**

> if (!require(maps)) {

install.packages("maps")

require(maps)

}

> # maps 패키지의 데이터가 있는 디렉터리 이름 가져오기 (Mac의 경우)

> pos <- grep("maps", searchpaths(), value=TRUE)

> Version <- installed.packages()["maps", "Version"]

>

> if (Version < "3.1.0") {

pos <- paste(pos, "data", sep="/")

# 데이터 파일 이름 중에서 postfix가 'MapEnv'인 맵 데이터베이스 이름 가져오기

map.data <- grep("MapEnv", list.files(pos), value=TRUE)

map.data <- sapply(map.data, function(x) strsplit(x, "MapEnv")[[1]][1])

names(map.data) <- NULL

map.data

} else {

pos <- paste(pos, "mapdata", sep="/")

map.data <- list.files(pos)

map.data <- sapply(map.data, function(x) strsplit(x, "\\.")[[1]][1])

map.data <- unique(map.data)

}

> pos

[1] "/Library/Frameworks/R.framework/Versions/3.1/Resources/library/maps/mapdata"

> map.data

[1] "county" "france" "italy" "nz" "state.carto"

[6] "state.vbm" "state" “usa” "world2" "world2"

**본문에서,**

**크기와 모양이 변경된 맵 데이터베이스다. 다음에 아래의 문장을 추가**

본 예제는 필자의 maps 패키지 2.3-6 버전에서 수행하였으며, 3.1.0 이상의 패키지를 설치한 독자는 다소 상이한 맵 데이터베이스를 가지고 있을 것이다. 그러나 maps() 함수 예제의 수행에는 차이는 없다.

* 623 페이지 ~ 624 페이지 에서

**소스를 다음으로 대체함**

> library(maps)

> # 데이터 조작을 위한 사용자 정의 함수

> # (문자열을 Mixed Case로 변환하는 함수)

> .simpleCap <- function(x) {

s <- strsplit(x, " ")[[1]]

paste(toupper(substring(s, 1, 1)), substring(s, 2), sep = "", collapse = " ")

}

> # map 패키지 지리정보에 USArrests 범죄정보의 색상을 매핑하는 함수

> getMatch <- function(mstates, arrests) {

idx <- grep(paste0("^", mstates), as.character(arrests$states))

ifelse(length(idx) == 0, "#FFFFFF00", as.character(arrests$col.level[idx]))

}

> # 본토에서 떨어진 Alaska, Hawaii 데이터 제외

> sub.usa <- subset(USArrests,!rownames(USArrests) %in% c("Alaska", "Hawaii"))

> # 주이름, 폭행범 수를 갖는 데이터 프레임 생성

> usa.data <- data.frame(states = rownames(sub.usa), Assault = sub.usa$Assault)

> # 범례 데이터 생성

> col.level <- cut(sub.usa[, 2], c(0, 100, 150, 200, 250, 300, 350))

> legends <- levels(col.level)

> # 주이름, 폭행범 수, 색상을 갖는 데이터 프레임 생성

> levels(col.level) <- sort(heat.colors(6), decreasing = TRUE)

> usa.data <- data.frame(usa.data, col.level = col.level)

> # 지리정보와 USArrests 매핑

> mstate <- map('state', plot=FALSE)$names

> mstate <- sapply(mstate, function(x) .simpleCap(strsplit(x, ":")[[1]][1]))

> mstate <- unlist(sapply(t(mstate), getMatch, usa.data))

> # Map 데이터 시각화

> map('state', fill = TRUE, col = mstate)

> title("USA Assault map")

> legend(-76, 35, legends, fill = sort(heat.colors(6), decreasing = TRUE), cex = 0.7)

**그림을 7.27을 첨부한 파일 그림\_7\_28.pdf로 변경**

**본문에서 다음을 대체함**

그 이유는 많은 사람들이 흰색보다는 빨간색이 더 큰 수치를 설명하는 것처럼 인식하고 있기 때문이다.

maps 패키지에 포함된 미국의 지도 데이터는 주(states)의 하부 구역 단위인 카운티(county) 정보까지 포함하고 있다. 를

그 이유는 많은 사람들이 흰색보다는 빨간색이 더 큰 수치를 설명하는 것처럼 인식하고 있기 때문이다.

미국의 주(stats)에 대한 맵 정보는 63개로 세분화되어 있다. 예를 들면 USArrests 데이터 프레임의 “Michigan”주는 “michigan:north”과 “michigan:south”으로 구분되어 있다. 그래서 48개 주의 데이터 프레임 정보와 63개의 맵 정보를 매핑하기 위해 예제에서 사용자 정의 함수로 getMatch( ) 함수와 .simpleCap( ) 함수를 만들어서 예제를 수행하였다.

maps 패키지에 포함된 미국의 지도 데이터는 주(states)의 하부 구역 단위인 카운티(county) 정보까지 포함하고 있다. 로 대체

* 636 페이지 소스에서

**소스를 다음처럼 대체함**

> level1\_dbf <- sqldf("select level1\_dbf.\*, data1.\* from level1\_dbf, data1

where level1\_dbf.code = data1.C행정구역별\_읍면동")

> level2\_dbf <- sqldf("select level2\_dbf.\*, data2.\* from level2\_dbf, data2

where level2\_dbf.code = data2.C행정구역별\_읍면동")

> level3\_dbf <- sqldf("select level3\_dbf.\*, data3.\* from level3\_dbf, data3

where level3\_dbf.code = data3.C행정구역별\_읍면동") 을

> # 한글 변수명을 백업한 후 영문으로 수정

> tabname <- c(names(level1\_dbf), names(data1))

> names(data1) <- names(data2) <- names(data3) <- paste("C", 1:23, sep="")

> level1\_dbf <- sqldf("select level1\_dbf.\*, data1.\* from level1\_dbf, data1

where level1\_dbf.code = data1.C1")

> level2\_dbf <- sqldf("select level2\_dbf.\*, data2.\* from level2\_dbf, data2

where level2\_dbf.code = data2.C1")

> level3\_dbf <- sqldf("select level3\_dbf.\*, data3.\* from level3\_dbf, data3

where level3\_dbf.code = data3.C1")

> # 영문 변수명을 한글 변수명으로 돌려놓기

> names(level1\_dbf) <- names(level2\_dbf) <- names(level3\_dbf) <- tabname

으로 대체함

**본문을 다음처럼 대체함**

C행정구역별\_읍면동 변수의 값이 같은 조건을 찾아서 두 데이터를 병합했다.

를

C행정구역별\_읍면동 변수의 값이 같은 조건을 찾아서 두 데이터를 병합했다. MS-Windows 환경에서는 변수 이름에 한글이 있을 경우 sqldf( ) 함수에서 에러가 발생한다. 그래서 변수명 C행정구역별\_읍면동을 C1으로 변경한 다음 sqldf( ) 함수를 호출하였다.

* 641 페이지 소스에서

**소스를 다음처럼 대체함**

> mountains <- c(“설악산”, “태백산”, "지리산", "소백산", "한라산", "내장산", "북한산")

> # 명산들의 위도와 경도의 조회

> xy <- geocode(mountains) 을

> mountains <- c("설악산", "태백산", "지리산노고단", "소백산", "한라산", "내장산", "북한산")

> # 명산들의 위도와 경도의 조회

> xy <- geocode(mountains)

> mountains[3] <- “지리산" 으로 대체함

* 654 페이지 소스에서

**소스를 다음처럼 대체함**

> library(repmis)

> traffic <- source\_DropboxData("traffic.csv", "wsjmfxx1cex5mbv", sep=",", header=TRUE) 을

> url <- "https://dl.dropboxusercontent.com/u/46305178/datas/traffic.csv"

> if (.Platform$OS.type == "unix") {

traffic <- read.csv(url, row.names=NULL)

} else if (.Platform$OS.type == "windows") {

Sys.setlocale(category = "LC\_ALL", locale = "C")

traffic <- read.csv(url, row.names = NULL, encoding = "UTF-8")

Sys.setlocale(category = "LC\_ALL", locale = "Korean")

} 로 대체함

**본문에서**

샘플 데이터는 https://www.dropbox.com/s/wsjmfxx1cex5mbv/traffic.csv라는 공유링크로 를

샘플 데이터는 https://dl.dropboxusercontent.com/u/46305178/datas/traffic.csv라는 공유링크로 로 대체

먼저 repmis 패키지의 source\_DropboxData() 함수로 드롭박스의 공개 데이터를 다운로드하여 traffic이라는 데이터 프레임을 생성한다. 를

먼저 read.csv() 함수로 드롭박스의 공개 데이터를 다운로드하여 traffic이라는 데이터 프레임을 생성한다. 이 방법은 예제처럼 Mac OS X 환경과 MS-Windows 환경에서의 사용 방법이 다르므로 주의해야 한다. 로 변경

* 659 페이지 소스에서

**소스를 다음처럼 대체함**

> p <- map + geom\_hex(aes(x=start.pos.x, y=start.pos.y), bins=12,

alpha=0.6, color="white", data=traffic) 를

> p <- map + coord\_cartesian() + geom\_hex(aes(x=start.pos.x, y=start.pos.y),

bins=12, alpha=0.6, color="white", data=traffic) 로 대체

* **660 페이지 7.3.1 twitter 검색하기 ~ 664 페이지 7.3.1.4 searchTwitter 사용하기 전까지의 내용을 다음으로 대치함**

7.3.1 트위터 검색하기

2015년 4월 본서의 초판 시에는 R에서 트위터의 트윗을 검색하는 twitteR 패키지의 버전이 1.1.7이었다. 그런데 1.1.8(2016년 7월 23일 현재 기준 1.1.9)이후 버전의 사용자 인증 기능이 변경되었고 2쇄가 인쇄되는 시점에 1.1.9 버전 기준으로 수정하였음을 밝혀둔다. 단, 버전업으로 인해서 작업 과정이 제법 많이 변경되었지만 그 차이점에 대한 비교는 다루지 않겠다.

트위터의 오픈 API를 이용하여 특정 주제어가 들어 있는 트윗을 검색하는 환경을 만들자. R에서 트위터의 오픈 API를 사용하기 위해서는 다음 과정을 수행해 야 한다.

1. 트위터 가입하기

2. Application 생성하기

3. Access Token 생성하기

4. twitteR 패키지 설치하기

5. twitteR 패키지로 인증 후 데이터 검색하기

7.3.1.1 트위터 가입과 Application 및 Access Token 생성하기

https://twitter.com/에서 ‘트위터 가입하기’를 수행한다. 이미 계정이 있다면 로그인을 수행한다. 그리고 그림7.45와 같은 https://apps.twitter.com/ 사이트에서 ‘Create New App’ 버튼을 눌러 신규 Application을 만든다. 이미 Application이 있는 사용자는 기존 Application을 사용해도 된다.

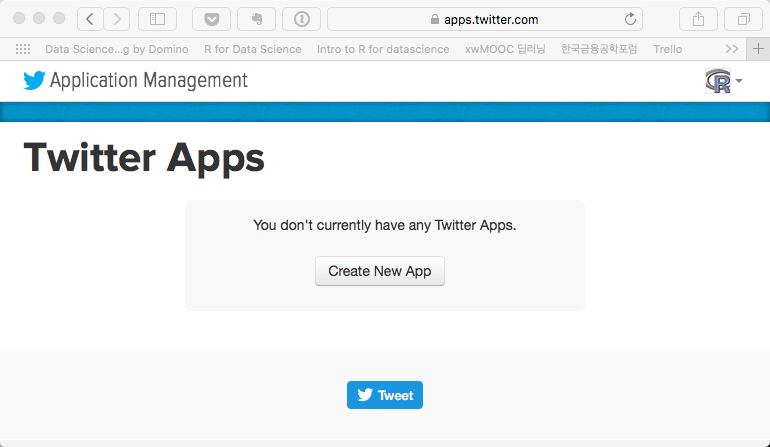


그림 7.45. apps.twitter.com 사이트 화면

Application 생성화면에서는 그림 7.46처럼 Name, Description, Website의 값을 채워 넣어야 한다. 필자는 ‘Scrap Twitt’, ‘Scraping twitter with R’를 기입하였는데 독자들도 각자 원하는 내용을 입력하면 된다. 마지막의 ‘Your Web site URL’은 독자의 웹 사이트 주소를 입력하라는 의미를 표현한 것이다. 웹 사이트 주소에는 독자들의 블로그나 카페 주소 등을 입력하면 된다. 필수 기입 사항을 입력한 후 하단의 동의 여부를 체크한 후 ‘Create your Twitter application’ 버튼을 눌러 Application을 생성한다. 이때, 트위터 프로필에 휴대전화를 추가하지 않았다면 Application이 생성되지 않는다. 이 경우에는 <https://twitter.com/> 사이트로 돌아가 ‘프로필 및 설정’ > ‘설정’ > ‘모바일’ 메뉴에서 휴대폰 번호를 추가한 후 수행해야 한다.

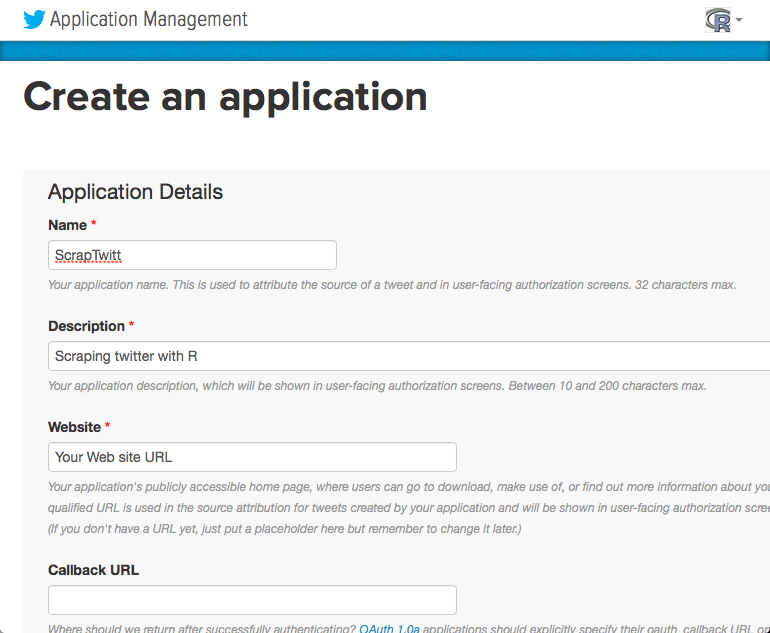


그림 7.46. Application 생성화면

Application이 정상적으로 생성되면 그림 7.47과 같은 화면을 만나게 된다. 필자가 기입한대로 Application 이름이 ‘Scrap Twitt’로 설정되었다. 이제 Application을 만들었으니 ‘Keys and Access Tokens’ 탭을 눌러 Access Tokens을 만들 차례다.

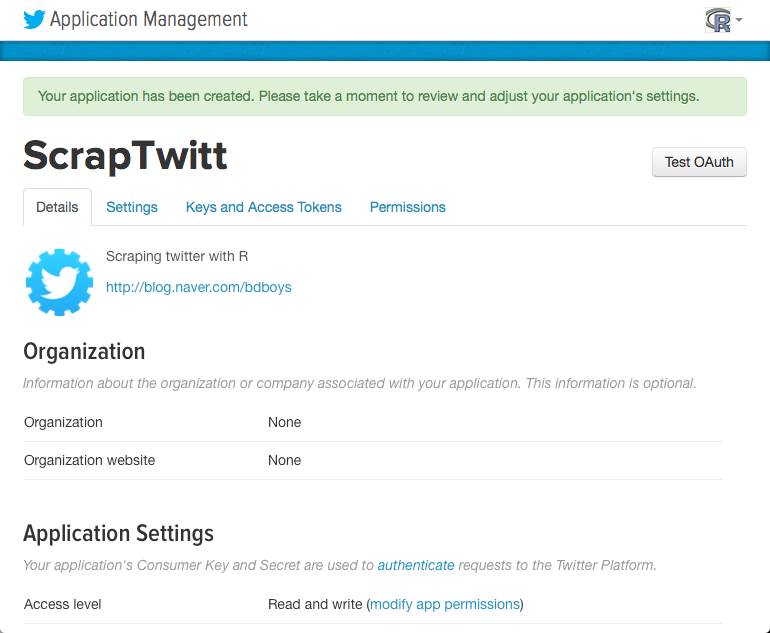


그림 4.47. Application 화면

‘Keys and Access Tokens’ 탭 화면은 그림 7.48과 같다. 이 화면에서 ‘Consumer Key(API Key)’와 ‘Consumer Secret(API Secret)’ 정보는 트윗을 검색할 때 사용하므로 메모해 놓자. 필자는 그림 7.48에서 이 정보에 이미지로 마스킹처리했다. 공유되어서는 안될 정보이기 때문이다. 그러므로 독자들도 가급적 비밀번호처럼 소중히 다루기 바란다.

이 화면의 하단에는 Access Token 정보 영역이 있는데 아직은 Access Token 정보가 없기 때문에 ‘Create my access token’ 버튼을 눌러 Access Token을 생성한다.

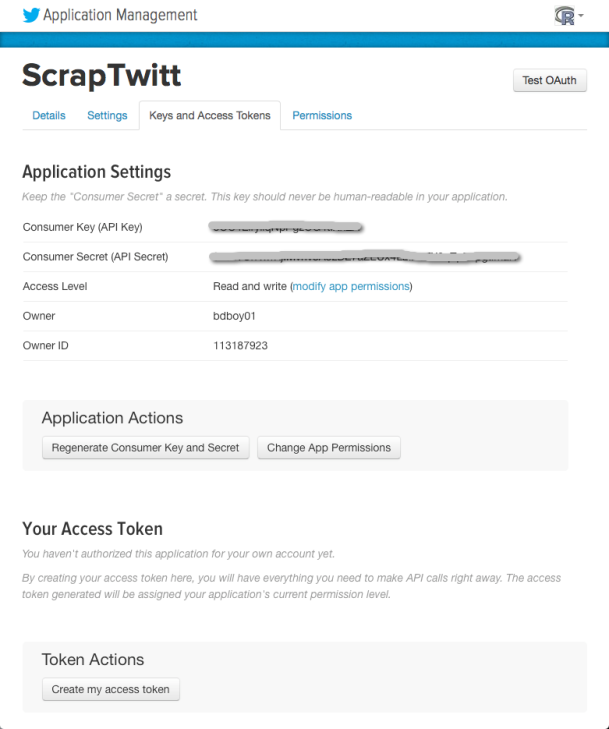


그림 4.48. Keys and Access Tokens 화면

그림 7.49는 Access Token과 Access Token Secret가 생성된 화면이다. 필자는 역시 이 정보도 마스킹처리했다. 그러므로 이 정보 역시 소중히 다루어야 한다. 이제 비로소 R에서 트윗을 검색하기 위한 네 가지의 정보를 모두 얻었다.

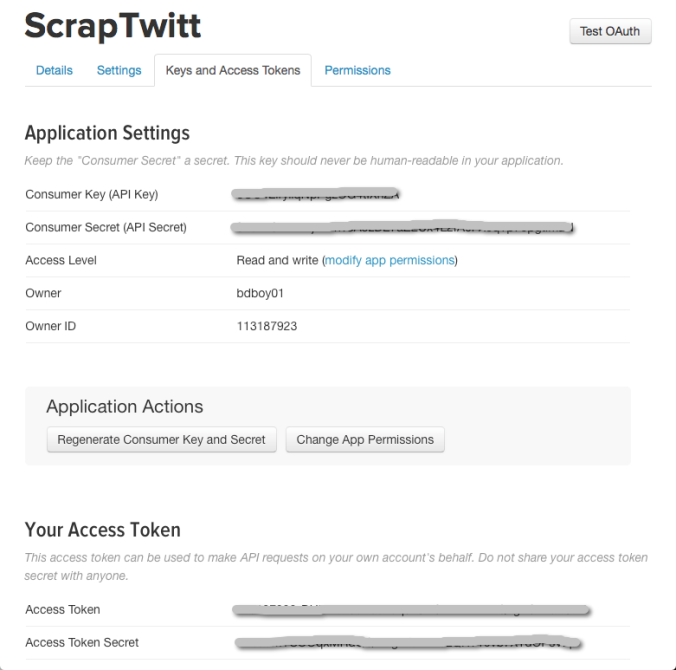


그림 4.49. Access Token이 생성된 화면

7.3.1.1 트위터 접근 권한 획득하기

twitteR 패키지를 설치한 후 로드한다. consumerKey와 consumerSecret는 Application의 Keys and Access Tokens 화면에서 적어 놓았던 Consumer Key와 Consumer Secret를 입력한다. accessToken와 accessSecret에는 Access Token과 Access Token Secret를 입력한다. 중요한 것은 개인에게만 부여된 정보이므로 타인에게 절대로 오픈해서는 안 된다. 그래서 필자는 이 부분을 ‘x’ 문자를 이용해서 예시로 표현했다. 그러므로 실제 여러분은 자신이 획득한 정보들을 입력해야 한다.

options( ) 함수로 httr\_oauth\_cache 인수값을 TRUE로 지정한 후 setup\_twitter\_oauth( ) 함수를 호출하면 R에서 트윗을 검색할 수 있는 권한을 획득하게 된다. 그런데 이 권한은 R 세션이 유지되어 있는 동안만 유효하므로 다시 R을 구동하고 작업할 경우에는 다시 이 작업을 수행해야 한다.

> # twitteR 패키지 설치하기

> if (!require(twitteR)) {

install.packages("twitteR")

require(twitteR)

}

>

> # Application의 Consumer Key 설정하기

> consumerKey <- "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"

>

> # Application의 Consumer Secret 설정하기

> consumerSecret <- "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

>

> # Application의 Access Token 설정하기

> accessToken <- 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

>

> # Application의 Access Secret 설정하기

> accessSecret <- 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

>

> # OAuth token의 캐싱을 위한 환경 설정

> options(httr\_oauth\_cache = TRUE)

>

> # R에서 트위터 접근 권한 확득하기

> setup\_twitter\_oauth(consumer\_key = consumerKey,

+ consumer\_secret = consumerSecret,

+ access\_token = accessToken,

+ access\_secret = accessSecret)

[1] "Using direct authentication"

>

7.3.1.3 twitteR 패키지로 트윗 데이터 검색하기

정상적인 권한을 획득하였다면 트윗 데이터를 검색하자. searchTwitter( ) 함수로 원하는 키워드를 포함하는 트윗을 검색하면 된다.

> library(twitteR)

>

> searchTwitter("Big Data", n=3)

[[1]]

[1] "phabeebmcse: RT @jimkaskade: Security Will Need Big Insight, Not Just Big Data - TechCrunch (blog) http://t.co/w6EgvUTIFP"

[[2]]

[1] "dwest999: Big Data and Analytics in Three Verbs :: MSP :: Midsize Insider http://t.co/bbkkcmjOMf"

[[3]]

[1] "itknowingness: RT @wytedocinisa: Check Out Our Blog Post: How Do You Build Intelligence from Social Media Big Data? t,co/ornLatVfYH #BigData"

>

> searchTwitter("Visualization", n=3, lang="en")

[[1]]

[1] "D1Visualization: Teradata, Datawatch Ally for Data Visualization - http://t.co/g8ZGYI1llR http://t.co/0ZqWEfTk6g"

[[2]]

[1] "seo\_\_\_\_\_\_\_\_: #PabloenEcuador Top 10 Free Data Visualization Tools For Content Marketers http://t.co/wGzVBwLeOv"

[[3]]

[1] "JaydenHawkins1: Kenya slide tours-an turn of events against encourage visualization: MeJPeEsKr"

* 665 페이지 소스에서

**소스를 다음처럼 대체함 (소스 추가)**

> vis.en <- searchTwitter("Visualization Analytics", lang="en")

> vis.ko <- searchTwitter("데이터 시각화", lang="ko")

> save(vis.en, vis.ko, file="vis\_twit.RData") 를

> vis.en <- searchTwitter("Visualization Analytics", lang="en")

> vis.ko <- searchTwitter("데이터 시각화", lang="ko")

> # 한글 검색 시 MS-Windows에서는 enc2utf8( ) 함수를 사용해야 정상적으로 검색됨

> # vis.ko <- searchTwitter(enc2utf8("데이터 시각화"), lang="ko")

> save(vis.en, vis.ko, file="vis\_twit.RData") 로 대체

**본문에서 다음을 대체 (문장 추가)**

먼저 필자는 다음과 같은 키워드로 며칠 동안 한글과 영문으로 두 가지의 주제로 트윗 자료를 검색하여 각각 vis.en와 vis.ko를 만들고, 이를 vis\_twit.RData라는 이름의 R 데이터 파일로 저장해 두었다. 을

먼저 필자는 다음과 같은 키워드로 며칠 동안 한글과 영문으로 두 가지의 주제로 트윗 자료를 검색하여 각각 vis.en와 vis.ko를 만들고, 이를 vis\_twit.RData라는 이름의 R 데이터 파일로 저장해 두었다. 특히 MS-Windows 환경에서 한글 키워드로 검색할 경우에는 예제처럼 enc2utf8( ) 함수를 사용해야 한다. 필자의 Mac OS X 환경에서는 이 작업이 필요하지 않다. 로 대체

* 738 페이지 소스에서

**소스를 다음처럼 대체함**

install\_github('arcdiagram', username='gastonstat')를

install\_github('gastonstat/arcdiagram')로 대체

* 796 페이지 소스에서

**소스를 다음처럼 대체함**

+ interval = “prediction”을

interval = “prediction”으로 대체

* 807 페이지 소스에서

**소스를 다음처럼 대체함**

+ xlab = "신장(cm)"을

xlab = "신장(cm)"으로 대체

* 819 페이지 소스에서

**소스를 다음처럼 대체함**

smain="3학년 5반 과일 선호도"를

main="3학년 5반 과일 선호도"로 대체

* 837 페이지 소스에서

**소스에서**

> # car 패키지의 로드, 없으면 설치 후 로드 를

> # car 패키지의 설치를 위한 사전 작업

> if (!require(pbkrtest) &

paste(R.version$major, sub("\\.", "", R.version$minor), sep = ".") < 3.23) {

install.packages("lme4")

cran <- "https://cran.r-project.org/"

if (.Platform$OS.type == "unix") {

packageurl <- paste0(cran, "src/contrib/Archive/pbkrtest/pbkrtest\_0.4-4.tar.gz")

install.packages(packageurl, repos=NULL, type="source")

} else if (.Platform$OS.type == "windows") {

packageurl <- paste0(cran, "bin/windows/contrib/3.1/pbkrtest\_0.4-4.zip")

install.packages(packageurl, repos=NULL, type="win.binary")

}

}

> # car 패키지의 로드, 없으면 설치 후 로드 로 대체

**본문에서**

**car 패키지의 dataEllipse() 함수는 앞에 다음을 추가**

car 패키지는 pbkrtest 패키지에 종속성이 있어 car 패키지 설치 시에 pbkrtest 패키지를 설치한다. 그런데 2016년 7월 현재 pbkrtest 패키지는 버전 0.4.6으로 R 버전 3.2.3 이상에서만 설치된다. 만약 R 버전이 3.2.3 미만이고 pbkrtest 패키지가 설치되지 않는다면, 예제처럼 0.4-4 버전의 pbkrtest 패키지를 설치한 후 car 패키지를 설치한다.