



# Reading mySQL

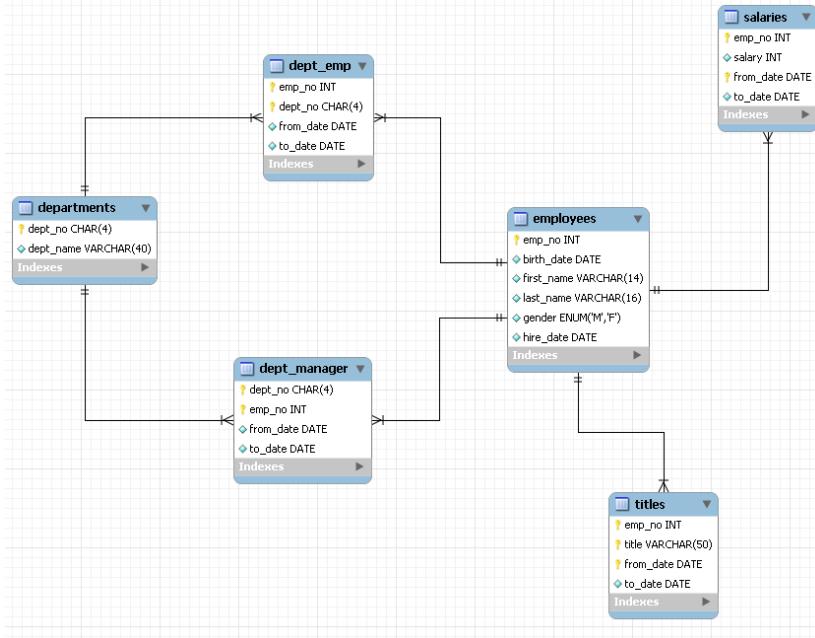
Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# mySQL

- Free and widely used open source database software
- Widely used in internet based applications
- Data are structured in
  - Databases
  - Tables within databases
  - Fields within tables
- Each row is called a record

<http://en.wikipedia.org/wiki/MySQL> <http://www.mysql.com/>

# Example structure



<http://dev.mysql.com/doc/employee/en/sakila-structure.html>

# Step 1 - Install MySQL

The screenshot shows a web browser window displaying the MySQL 5.7 Reference Manual. The title bar reads "MySQL :: MySQL 5.7 Reference". The URL in the address bar is "dev.mysql.com/doc/refman/5.7/en/installing.html". The main content area is titled "Chapter 2. Installing and Upgrading MySQL". A "Table of Contents" sidebar on the left lists various MySQL manual versions: MySQL 5.7 Manual, MySQL 5.6 Manual, MySQL 5.5 Manual, MySQL 5.1 Manual, MySQL 5.0 Manual, and MySQL 3.23/4.0/4.1 Manual. Below this is a search bar labeled "Search manual:" with a "Go" button. The main content area lists 13 sections under "Table of Contents": 2.1. General Installation Guidance, 2.2. Installing MySQL on Unix/Linux Using Generic Binaries, 2.3. Installing MySQL on Microsoft Windows, 2.4. Installing MySQL on Mac OS X, 2.5. Installing MySQL on Linux, 2.6. Installing MySQL on Solaris and OpenSolaris, 2.7. Installing MySQL on HP-UX, 2.8. Installing MySQL on FreeBSD, 2.9. Installing MySQL from Source, 2.10. Postinstallation Setup and Testing, 2.11. Upgrading or Downgrading MySQL, 2.12. Environment Variables, and 2.13. Perl Installation Notes. To the right of the main content is a "Section Navigation" sidebar with a "Preface and Legal Notices" section and a numbered list from 1 to 10, each with a corresponding icon. The icons include a star, a document, a gear, a person, a gear, a gear, a gear, a gear, a gear, and a gear.

<http://dev.mysql.com/doc/refman/5.7/en/installing.html>

# Step 2 - Install RMySQL

- On a Mac: `install.packages("RMySQL")`
- On Windows:
  - Official instructions - <http://biostat.mc.vanderbilt.edu/wiki/Main/RMySQL> (may be useful for Mac/UNIX users as well)
  - Potentially useful guide - <http://www.ahschulz.de/2013/07/23/installing-rmysql-under-windows/>

# Example - UCSC database

The screenshot shows the UCSC Genome Bioinformatics website. The header includes the title "UCSC Genome Bioinformatics" and a navigation bar with links to Genomes, Blat, Tables, Gene Sorter, PCR, VisiGene, Session, FAQ, and Help. A sidebar on the left lists various tools: Genome Browser, ENCODE, Neandertal, Blat, Table Browser, Gene Sorter, In Silico PCR, Genome Graphs, Galaxy, VisiGene, Utilities, Downloads, Release Log, Custom Tracks, and Cancer Browser. The main content area features a section titled "About the UCSC Genome Bioinformatics Site" with text about the genome browser's purpose and tools like Gene Sorter, Blat, and VisiGene. Below this is a "News" section with a Twitter icon, a link to "News Archives", and a paragraph about receiving announcements via email. It also highlights a new "100 Species Conservation Track now available on hg19" from November 27, 2013, and mentions the hard work of the UCSC Genome Browser staff.

View site information

genome.ucsc.edu

## UCSC Genome Bioinformatics

Genomes · Blat · Tables · Gene Sorter · PCR · VisiGene · Session · FAQ · Help

**About the UCSC Genome Bioinformatics Site**

Welcome to the UCSC Genome Browser website. This site contains the reference sequence and working draft assemblies for a large collection of genomes. It also provides portals to the [ENCODE](#) and [Neandertal](#) projects.

We encourage you to explore these sequences with our tools. The [Genome Browser](#) zooms and scrolls over chromosomes, showing the work of annotators worldwide. The [Gene Sorter](#) shows expression, homology and other information on groups of genes that can be related in many ways. [Blat](#) quickly maps your sequence to the genome. The [Table Browser](#) provides convenient access to the underlying database. [VisiGene](#) lets you browse through a large collection of *in situ* mouse and frog images to examine expression patterns. [Genome Graphs](#) allows you to upload and display genome-wide data sets.

The UCSC Genome Browser is developed and maintained by the Genome Bioinformatics Group, a cross-departmental team within the Center for Biomolecular Science and Engineering ([CBSE](#)) at the University of California Santa Cruz ([UCSC](#)). If you have feedback or questions concerning the tools or data on this website, feel free to contact us on our [public mailing list](#).

**News** News Archives ▶

To receive announcements of new genome assembly releases, new software features, updates and training seminars by email, subscribe to the [genome-announce](#) mailing list.

**27 November 2013 – 100 Species Conservation Track now available on hg19**

After 15.4 years of CPU run-time in 9,905,594 individual 'jobs' and 99 cluster runs for lastz pair-wise alignment...we are excited to announce the release of a 100 species alignment on the hg19/GRCh37 human Genome Browser.

This new Conservation track shows multiple alignments of 100 species and measurements of evolutionary conservation using two methods (phastCons and phyloP) from the PHAST package. This adds 40 more species to the existing 60-way on the mm10 mouse browser. For more information about the 100 species Conservation track, see its [description page](#).

We'd also like to acknowledge the hard work of the UCSC Genome Browser staff who pulled together the information for this track: Hiram Clawson and Pauline Fujita.

<http://genome.ucsc.edu/>

# UCSC MySQL

The screenshot shows a web browser window with the title bar "Downloading Data using MySQL". The address bar contains the URL "genome.ucsc.edu/goldenPath/help/mysql.html". The main content area is titled "UCSC Genome Bioinformatics" and features a navigation menu with links to Home, Genomes, Blat, Tables, Gene Sorter, PCR, Session, FAQ, and Help. Below the menu, a section titled "Downloading Data using MySQL" is displayed. It contains text about a MySQL database for public access at "genome-mysql.cse.ucsc.edu". It explains that the server allows MySQL access to the same set of data available on the public Genome Browser site, with synchronization occurring weekly. It also notes that the MySQL server can be intermittently out of sync with the main website. A "Connecting" section provides instructions for connecting to the MySQL server using the command "mysql --user=genome --host=genome-mysql.cse.ucsc.edu -A", where the "-A" flag is optional for speed. It states that once connected, a wide range of MySQL commands can be used to query the database. A "Conditions of Use" section lists several rules, including avoiding excessive queries, not using the MySQL server for bot access or program-driven use, and prohibiting attachments by local mirror sites. A final section, "Using the MySQL Server with our Utilities", mentions that the MySQL database can be used with various utilities found in the "Genome Browser source" tree, some of which require a password.

<http://genome.ucsc.edu/goldenPath/help/mysql.html>

# Connecting and listing databases

```
ucscDb <- dbConnect(MySQL(), user="genome",
                     host="genome-mysql.cse.ucsc.edu")
result <- dbGetQuery(ucscDb, "show databases;"); dbDisconnect(ucscDb);
```

```
[1] TRUE
```

```
result
```

```
      Database
1 information_schema
2         ailMell1
3        allMis1
4       anoCar1
5       anoCar2
6       anoGam1
7       apiMell1
8       apiMel2
```

# Connecting to hg19 and listing tables

```
hg19 <- dbConnect(MySQL(), user="genome", db="hg19",
                  host="genome-mysql.cse.ucsc.edu")
allTables <- dbListTables(hg19)
length(allTables)
```

```
[1] 10949
```

```
allTables[1:5]
```

```
[1] "HInv"           "HInvGeneMrna"  "acembly"        "acemblyClass"  "acemblyPep"
```

# Get dimensions of a specific table

```
dbListFields(hg19, "affyU133Plus2")
```

```
[1] "bin"          "matches"      "misMatches"   "repMatches"   "nCount"       "qNumInsert"  
[7] "qBaseInsert"  "tNumInsert"    "tBaseInsert"  "strand"       "qName"        "qSize"  
[13] "qStart"       "qEnd"         "tName"        "tSize"        "tStart"       "tEnd"  
[19] "blockCount"   "blockSizes"    "qStarts"      "tStarts"
```

```
dbGetQuery(hg19, "select count(*) from affyU133Plus2")
```

```
count(*)  
1      58463
```

# Read from the table

```
affyData <- dbReadTable(hg19, "affyU133Plus2")
head(affyData)
```

	bin	matches	misMatches	repMatches	nCount	qNumInsert	qBaseInsert	tNumInsert	tBaseInsert	strand
1	585	530	4	0	23	3	41	3	898	-
2	585	3355	17	0	109	9	67	9	11621	-
3	585	4156	14	0	83	16	18	2	93	-
4	585	4667	9	0	68	21	42	3	5743	-
5	585	5180	14	0	167	10	38	1	29	-
6	585	468	5	0	14	0	0	0	0	-
	qName	qSize	qStart	qEnd	tName	tSize	tStart	tEnd	blockCount	blockSizes
1	225995_x_at	637	5	603	chr1	249250621	14361	15816	5	93, 144, 229, 70, 21,
2	225035_x_at	3635	0	3548	chr1	249250621	14381	29483	17	73, 375, 71, 165, 303, 360, 198, 661, 201, 1, 260, 250, 74, 73, 98, 155, 163,
3	226340_x_at	4318	3	4274	chr1	249250621	14399	18745	18	
4	1557034_s_at	4834	48	4834	chr1	249250621	14406	24893	23	
5	231811_at	5399	0	5399	chr1	249250621	19688	25078	11	
6	236841_at	487	0	487	chr1	249250621	27542	28029	1	

# Select a specific subset

```
query <- dbSendQuery(hg19, "select * from affyU133Plus2 where misMatches between 1 and 3")
affyMis <- fetch(query); quantile(affyMis$misMatches)
```

```
0%   25%   50%   75% 100%
 1     1     2     2     3
```

```
affyMisSmall <- fetch(query,n=10); dbClearResult(query);
```

```
[1] TRUE
```

```
dim(affyMisSmall)
```

```
[1] 10 22
```

# Don't forget to close the connection!

```
dbDisconnect(hg19)
```

```
[1] TRUE
```

# Further resources

- RMySQL vignette <http://cran.r-project.org/web/packages/RMySQL/RMySQL.pdf>
- List of commands <http://www.pantz.org/software/mysql/mysqlcommands.html>
  - **Do not, do not, delete, add or join things from ensembl. Only select.**
  - In general be careful with mysql commands
- A nice blog post summarizing some other commands <http://www.r-bloggers.com/mysql-and-r/>



# Reading HDF5

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# HDF5

- Used for storing large data sets
- Supports storing a range of data types
- Hierarchical data format
- *groups* containing zero or more data sets and metadata
  - Have a *group header* with group name and list of attributes
  - Have a *group symbol table* with a list of objects in group
- *datasets* multidimensional array of data elements with metadata
  - Have a *header* with name, datatype, dataspace, and storage layout
  - Have a *data array* with the data

<http://www.hdfgroup.org/>

# R HDF5 package

```
source("http://bioconductor.org/biocLite.R")
biocLite("rhd5")
```

```
library(rhdf5)
created = h5createFile("example.h5")
created
```

```
[1] TRUE
```

- This will install packages from Bioconductor <http://bioconductor.org/>, primarily used for genomics but also has good "big data" packages
- Can be used to interface with hdf5 data sets.
- This lecture is modeled very closely on the rhdf5 tutorial that can be found here <http://www.bioconductor.org/packages/release/bioc/vignettes/rhdf5/inst/doc/rhdf5.pdf>

# Create groups

```
created = h5createGroup("example.h5", "foo")
created = h5createGroup("example.h5", "baa")
created = h5createGroup("example.h5", "foo/foobaa")
h5ls("example.h5")
```

group	name	otype	dclass	dim
0	/baa		H5I_GROUP	
1	/foo		H5I_GROUP	
2	/foo/foobaa		H5I_GROUP	

# Write to groups

```
A = matrix(1:10,nr=5,nc=2)
h5write(A, "example.h5","foo/A")
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, "example.h5","foo/foobaa/B")
h5ls("example.h5")
```

	group	name	otype	dclass	dim
0	/	baa	H5I_GROUP		
1	/	foo	H5I_GROUP		
2	/foo	A	H5I_DATASET	INTEGER	5 x 2
3	/foo	foobaa	H5I_GROUP		
4	/foo/foobaa	B	H5I_DATASET	FLOAT	5 x 2 x 2

# Write a data set

```
df = data.frame(1L:5L,seq(0,1,length.out=5),  
  c("ab","cde","fghi","a","s"), stringsAsFactors=FALSE)  
h5write(df, "example.h5","df")  
h5ls("example.h5")
```

	group	name	otype	dclass	dim
0	/	baa	H5I_GROUP		
1	/	df	H5I_DATASET	COMPOUND	5
2	/	foo	H5I_GROUP		
3	/foo	A	H5I_DATASET	INTEGER	5 x 2
4	/foo/foobaa		H5I_GROUP		
5	/foo/foobaa	B	H5I_DATASET	FLOAT	5 x 2 x 2

# Reading data

```
readA = h5read("example.h5", "foo/A")
readB = h5read("example.h5", "foo/foobaa/B")
readdir= h5read("example.h5", "df")
readA
```

```
[,1] [,2]
[1,] 1 6
[2,] 2 7
[3,] 3 8
[4,] 4 9
[5,] 5 10
```

# Writing and reading chunks

```
h5write(c(12,13,14),"example.h5","foo/A",index=list(1:3,1))  
h5read("example.h5","foo/A")
```

```
[,1] [,2]  
[1,] 12   6  
[2,] 13   7  
[3,] 14   8  
[4,]  4   9  
[5,]  5  10
```

# Notes and further resources

- hdf5 can be used to optimize reading/writing from disc in R
- The rhdf5 tutorial:
  - <http://www.bioconductor.org/packages/release/bioc/vignettes/rhdf5/inst/doc/rhdf5.pdf>
- The HDF group has information on HDF5 in general <http://www.hdfgroup.org/HDF5/>



# Reading data from the web

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# Webscraping

**Webscraping:** Programatically extracting data from the HTML code of websites.

- It can be a great way to get data [How Netflix reverse engineered Hollywood](#)
- Many websites have information you may want to programmaticaly read
- In some cases this is against the terms of service for the website
- Attempting to read too many pages too quickly can get your IP address blocked

[http://en.wikipedia.org/wiki/Web\\_scraping](http://en.wikipedia.org/wiki/Web_scraping)

# Example: Google scholar

Jeff Leek – Google Scholar C X

scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en&oi=ao

Web Images More...

jtleek@gmail.com

**Jeff Leek** [Edit](#)  
Assistant Professor of Biostatistics, Johns Hopkins Bloomberg School of Public Health [Edit](#)  
Statistics - Computing - Genomics - Personalized Medicine - Scientific Communication [Edit](#)  
Verified email at jhsp.h.edu [Edit](#)  
My profile is public [Edit](#) [Link](#) [Homepage](#) [Edit](#)

**Citation indices**

	All	Since 2008
Citations	1285	1146
h-index	10	10
i10-index	11	11

**Citations to my articles**

Select: All, None Actions [:](#)

Title / Author Cited by Year

<a href="#">Significance analysis of time course microarray experiments</a> JD Storey, W Xiao, JT Leek, RG Tompkins, RW Davis Proceedings of the National Academy of Sciences of the United States of ...	338	2005
<a href="#">Capturing heterogeneity in gene expression studies by surrogate variable analysis</a> JT Leek, JD Storey PLoS Genetics 3 (9), e161	171	2007
<a href="#">EDGE: extraction and analysis of differential gene expression</a> JT Leek, E Monsen, AR Dabney, JD Storey Bioinformatics 22 (4), 507-508	140	2006
<a href="#">Tackling the widespread and critical impact of batch effects in high-throughput data</a> JT Leek, RB Sharp, HC Bravo, D Simcha, B Langmead, WE Johnson, D German, K ... Nature Reviews Genetics 11 (10), 733-739	133	2010
<a href="#">The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments</a> JD Storey, JY Dai, JT Leek UW Biostatistics Working Paper Series, 260	107	2005
<a href="#">Systems-level dynamic analyses of fate change in murine embryonic stem</a>		

**Follow this author**  
5 Followers

Follow new articles  
Follow new citations

**Add co-authors**

John D. Storey	<a href="#">Add</a> <input checked="" type="checkbox"/>
Rafael A Irizarry	<a href="#">Add</a> <input checked="" type="checkbox"/>
Ben Langmead	<a href="#">Add</a> <input checked="" type="checkbox"/>
Hector Corrada Br...	<a href="#">Add</a> <input checked="" type="checkbox"/>
wenzhong xiao	<a href="#">Add</a> <input checked="" type="checkbox"/>
W. Evan Johnson	<a href="#">Add</a> <input checked="" type="checkbox"/>
Alexander Lachm...	<a href="#">Add</a> <input checked="" type="checkbox"/>
Olga Troyanskaya	<a href="#">Add</a> <input checked="" type="checkbox"/>
Avi Ma'ayan	<a href="#">Add</a> <input checked="" type="checkbox"/>
Eduardo M Airoldi	<a href="#">Add</a> <input checked="" type="checkbox"/>

[View all co-authors](#)

**Co-authors**  
No co-authors

Name   
Email   
 Inviting co-author

<http://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en>

# Getting data off webpages - readLines()

```
con = url("http://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en")
htmlCode = readLines(con)
close(con)
htmlCode
```

```
[1] "<!DOCTYPE html><html><head><title>Jeff Leek - Google Scholar Citations</title><meta name=\"robots\"
```

# Parsing with XML

```
library(XML)
url <- "http://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en"
html <- htmlTreeParse(url, useInternalNodes=T)

xpathSApply(html, "//title", xmlValue)
```

```
[1] "Jeff Leek - Google Scholar Citations"
```

```
xpathSApply(html, "//td[@id='col-citedby']", xmlValue)
```

```
[1] "Cited by"  "397"      "259"      "237"      "172"      "138"      "125"      "122"
[9] "109"       "101"      "34"       "26"       "26"       "24"       "19"       "13"
[17] "12"        "10"       "10"       "7"        "6"
```

# GET from the httr package

```
library(httr); html2 = GET(url)
content2 = content(html2,as="text")
parsedHtml = htmlParse(content2,asText=TRUE)
xpathSApply(parsedHtml, "//title", xmlValue)
```

```
[1] "Jeff Leek - Google Scholar Citations"
```

# Accessing websites with passwords

```
pg1 = GET("http://httpbin.org/basic-auth/user/passwd")  
pg1
```

```
Response [http://httpbin.org/basic-auth/user/passwd]  
Status: 401  
Content-type:
```

<http://cran.r-project.org/web/packages/httr/httr.pdf>

# Accessing websites with passwords

```
pg2 = GET("http://httpbin.org/basic-auth/user/passwd",
    authenticate("user", "passwd"))
pg2
```

```
Response [http://httpbin.org/basic-auth/user/passwd]
Status: 200
Content-type: application/json
{
  "authenticated": true,
  "user": "user"
}
```

```
names(pg2)
```

```
[1] "url"           "handle"        "status_code"   "headers"       "cookies"      "content"
[7] "times"          "config"
```

# Using handles

```
google = handle("http://google.com")
pg1 = GET(handle=google, path="/")
pg2 = GET(handle=google, path="search")
```

<http://cran.r-project.org/web/packages/httr/httr.pdf>

# Notes and further resources

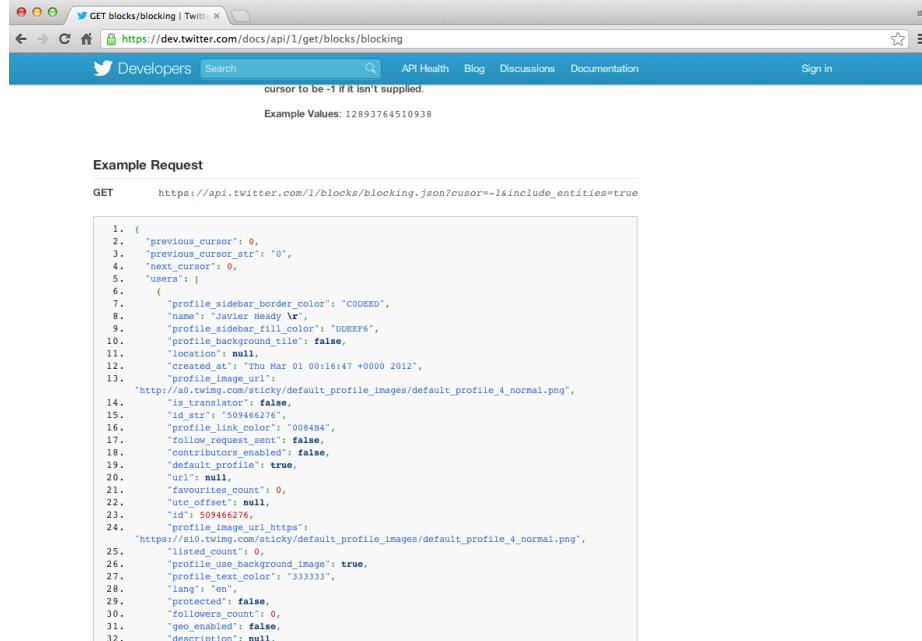
- R Bloggers has a number of examples of web scraping <http://www.r-bloggers.com/?s=Web+Scraping>
- The httr help file has useful examples <http://cran.r-project.org/web/packages/httr/httr.pdf>
- See later lectures on APIs



# Reading data from APIs

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# Application programming interfaces

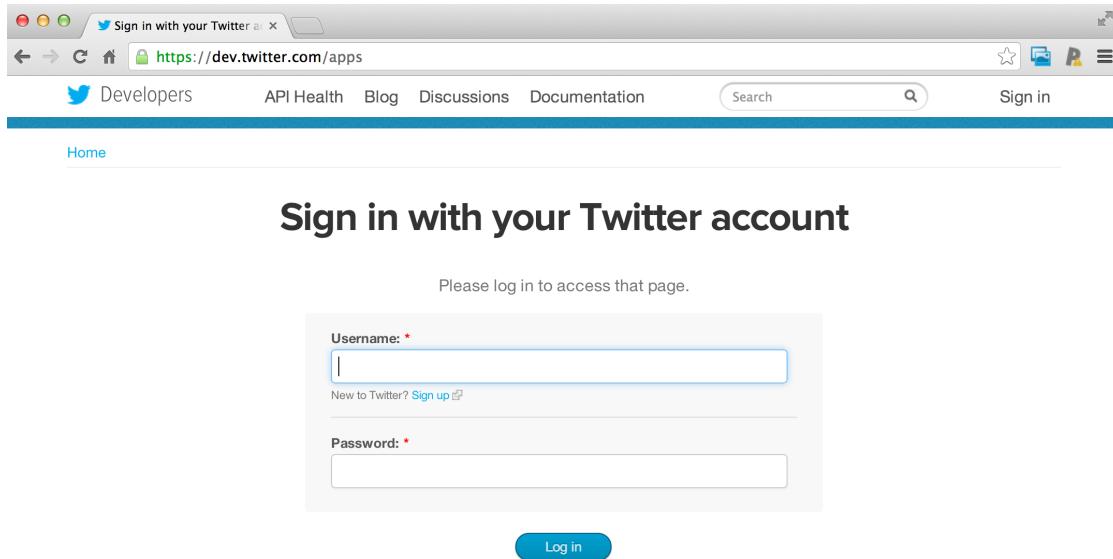


The screenshot shows a web browser window with the URL [https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)) in the address bar. The page title is "GET blocks/blocking | Twitter". The main content area displays the "Example Request" for the API endpoint. It includes a "GET" method and the URL [https://api.twitter.com/1/blocks/blocking.json?cursor=-1&include\\_entities=true](https://api.twitter.com/1/blocks/blocking.json?cursor=-1&include_entities=true). Below the URL is a large code block representing the JSON response from the API. The response contains a cursor value of -1 and a list of users, with the first user's profile details including name, profile picture, and creation date.

```
1. {
2.   "previous_cursor": 0,
3.   "previous_cursor_str": "0",
4.   "next_cursor": 0,
5.   "users": [
6.     {
7.       "profile_sidebar_border_color": "CODEDE",
8.       "name": "Tavian Ready",
9.       "profile_sidebar_fill_color": "DDDEF6",
10.      "profile_background_tile": false,
11.      "location": null,
12.      "created_at": "Thu Mar 01 00:16:47 +0000 2012",
13.      "profile_image_url": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
14.      "is_translator": false,
15.      "id_str": "509466276",
16.      "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
17.      "follow_request_sent": false,
18.      "contributors_enabled": false,
19.      "default_profile": true,
20.      "url": null,
21.      "favourites_count": 0,
22.      "utc_offset": null,
23.      "id": 509466276,
24.      "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
25.      "listed_count": 0,
26.      "profile_use_background_image": true,
27.      "profile_text_color": "333333",
28.      "lang": "en",
29.      "protected": false,
30.      "followers_count": 0,
31.      "geo_enabled": false,
32.      "description": null
33.    }
34.  ]
35.}
```

[https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)

# Creating an application



The screenshot shows a web browser window with the following details:

- Title Bar:** "Sign in with your Twitter account"
- Address Bar:** <https://dev.twitter.com/apps>
- Header:** Developers, API Health, Blog, Discussions, Documentation, Search, Sign in
- Content:** A "Sign in with your Twitter account" form. It includes fields for "Username:" and "Password:", both marked with a red asterisk indicating required fields. Below the password field is a "Log in" button.
- Text on Form:** "Please log in to access that page."
- Links on Form:** "New to Twitter? [Sign up](#)"

<https://dev.twitter.com/apps>)

# Creating an application

A screenshot of a web browser displaying the Twitter Developers 'My applications' page. The URL in the address bar is <https://dev.twitter.com/apps>. The page header includes the Twitter logo, 'Developers', 'API Health', 'Blog', 'Discussions', 'Documentation', a search bar, and a dropdown menu. Below the header, a 'Home' link is visible. The main content area is titled 'My applications' and features a card for an application named 'Simply Statistics'. The card includes the Twitter logo, the application name, and a subtitle 'Simply Statistics Blog'. A blue button labeled 'Create a new application' is located on the right side of the card.

Home

## My applications

[Create a new application](#)



[Simply Statistics](#)

Simply Statistics Blog

# Creating an application

The screenshot shows a web browser window for the Twitter Developers site at <https://dev.twitter.com/apps/3686814/show>. The page displays the configuration for an application named "Simply Statistics Blog" with the URL <http://simplystatistics.org/>.

**Organization**  
Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

**OAuth settings**  
Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read and write <a href="#">About the application permission model</a>
Consumer key	[REDACTED]
Consumer secret	[REDACTED]
Request token URL	[REDACTED]
Authorize URL	[REDACTED]

# Accessing Twitter from R

```
myapp = oauth_app("twitter",
                  key="yourConsumerKeyHere", secret="yourConsumerSecretHere")
sig = sign_oauth1.0(myapp,
                     token = "yourTokenHere",
                     token_secret = "yourTokenSecretHere")
homeTL = GET("https://api.twitter.com/1.1/statuses/home_timeline.json", sig)
```

# Converting the json object

```
json1 = content(homeTL)
json2 = jsonlite::fromJSON(toJSON(json1))
json2[1,1:4]
```

	created_at	id	id_str
1	Mon Jan 13 05:18:04 +0000 2014	4.22598398940684288	

1 Now that P. Norvig's regex golf IPython notebook hit Slashdot, let's see if our traffic spike tops th

# How did I know what url to use?

The screenshot shows a web browser displaying the Twitter API documentation. The URL in the address bar is [https://dev.twitter.com/docs/api/1.1/get/statuses/home\\_timeline](https://dev.twitter.com/docs/api/1.1/get/statuses/home_timeline). The page title is "GET statuses/home\_timeline". The top navigation bar includes links for Developers, API Health, Blog, Discussions, Documentation, and a search bar. Below the navigation, a breadcrumb trail shows Home → Documentation → REST API. On the right side, there is a "Tweet" button.

**GET statuses/home\_timeline**

[View](#) [What links here](#)

Updated on Wed, 2012-09-05 10:06 API version 1.1

Returns a collection of the most recent Tweets and retweets posted by the authenticating user and the users they follow. The home timeline is central to how most users interact with the Twitter service.

Up to 800 Tweets are obtainable on the home timeline. It is more volatile for users that follow many users or follow users who tweet frequently.

See [Working with Timelines](#) for instructions on traversing timelines efficiently.

**Resource URL**  
[https://api.twitter.com/1.1/statuses/home\\_timeline.json](https://api.twitter.com/1.1/statuses/home_timeline.json)

**Parameters**

<b>count</b> optional	Specifies the number of records to retrieve. Must be less than or equal to 200. Defaults to 20.  Example Values: 5
<b>since_id</b> optional	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.

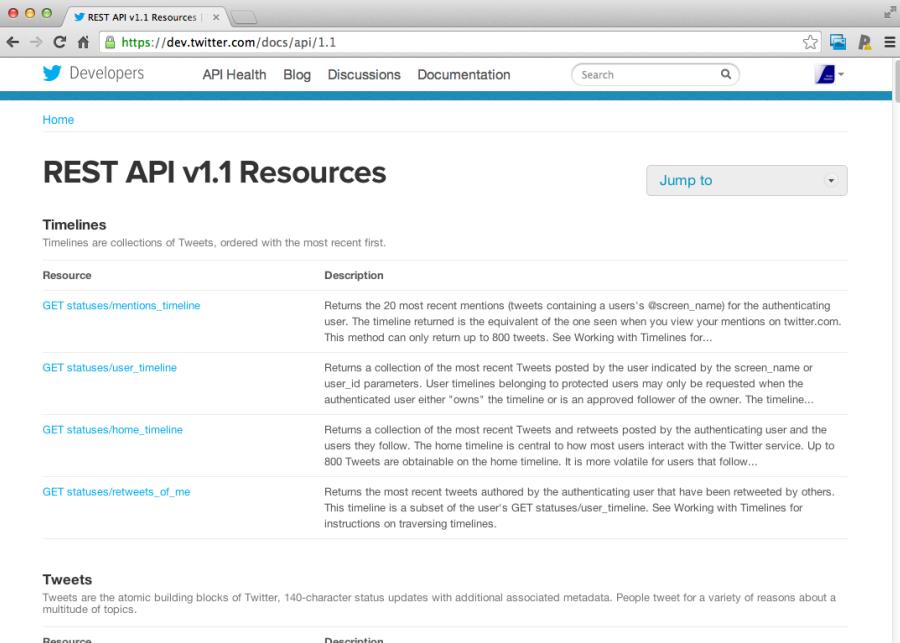
**Resource Information**

Rate Limited?	Yes
Requests per rate limit window	15/user
Authentication	Requires user context
Response Formats	json
HTTP Methods	GET
Resource family	statuses
Response Object	Tweets
API Version	v1.1

**OAuth tool**

<https://dev.twitter.com/docs/api/1.1/get/search/tweets>

# In general look at the documentation



The screenshot shows a web browser window with the title "REST API v1.1 Resources" and the URL "https://dev.twitter.com/docs/api/1.1". The page content includes a navigation bar with links for Developers, API Health, Blog, Discussions, Documentation, and a search bar. Below the navigation is a "Home" link. The main content area is titled "REST API v1.1 Resources" and features a "Jump to" dropdown menu. The first section, "Timelines", describes timelines as collections of Tweets, ordered with the most recent first. It lists five resources:

Resource	Description
<a href="#">GET statuses/mentions_timeline</a>	Returns the 20 most recent mentions (tweets containing a user's @screen_name) for the authenticating user. The timeline returned is the equivalent of the one seen when you view your mentions on twitter.com. This method can only return up to 800 tweets. See Working with Timelines for...
<a href="#">GET statuses/user_timeline</a>	Returns a collection of the most recent Tweets posted by the user indicated by the screen_name or user_id parameters. User timelines belonging to protected users may only be requested when the authenticated user either "owns" the timeline or is an approved follower of the owner. The timeline...
<a href="#">GET statuses/home_timeline</a>	Returns a collection of the most recent Tweets and retweets posted by the authenticating user and the users they follow. The home timeline is central to how most users interact with the Twitter service. Up to 800 Tweets are obtainable on the home timeline. It is more volatile for users that follow...
<a href="#">GET statuses/retweets_of_me</a>	Returns the most recent tweets authored by the authenticating user that have been retweeted by others. This timeline is a subset of the user's GET statuses/user_timeline. See Working with Timelines for instructions on traversing timelines.

The second section, "Tweets", defines tweets as the atomic building blocks of Twitter, 140-character status updates with additional associated metadata. People tweet for a variety of reasons about a multitude of topics.

<https://dev.twitter.com/docs/api/1.1/overview>

# In general look at the documentation

- httr allows GET, POST, PUT, DELETE requests if you are authorized
- You can authenticate with a user name or a password
- Most modern APIs use something like oauth
- httr works well with Facebook, Google, Twitter, Githb, etc.



# Reading from other sources

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# There is a package for that

- Roger has a nice video on how there are R packages for most things that you will want to access.
- Here I'm going to briefly review a few useful packages
- In general the best way to find out if the R package exists is to Google "data storage mechanism R package"
  - For example: "MySQL R package"

# Interacting more directly with files

- file - open a connection to a text file
- url - open a connection to a url
- gzfile - open a connection to a .gz file
- bzfile - open a connection to a .bz2 file
- *?connections* for more information
- Remember to close connections

# foreign package

- Loads data from Minitab, S, SAS, SPSS, Stata,Systat
- Basic functions *read.foo*
  - `read.arff` (Weka)
  - `read.dta` (Stata)
  - `read.mtp` (Minitab)
  - `read.octave` (Octave)
  - `read.spss` (SPSS)
  - `read.xport` (SAS)
- See the help page for more details <http://cran.r-project.org/web/packages/foreign/foreign.pdf>

# Examples of other database packages

- RPostresSQL provides a DBI-compliant database connection from R. Tutorial-<https://code.google.com/p/rpostgresql/>, help file-<http://cran.r-project.org/web/packages/RPostgreSQL/RPostgreSQL.pdf>
- RODBC provides interfaces to multiple databases including PostgreSQL, MySQL, Microsoft Access and SQLite. Tutorial - <http://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf>, help file - <http://cran.r-project.org/web/packages/RODBC/RODBC.pdf>
- RMongo <http://cran.r-project.org/web/packages/RMongo/RMongo.pdf> (example of Rmongo <http://www.r-bloggers.com/r-and-mongodb/>) and [rmongodb](#) provide interfaces to MongoDB.

# Reading images

- jpeg - <http://cran.r-project.org/web/packages/jpeg/index.html>
- readbitmap - <http://cran.r-project.org/web/packages/readbitmap/index.html>
- png - <http://cran.r-project.org/web/packages/png/index.html>
- EBImage (Bioconductor) - <http://www.bioconductor.org/packages/2.13/bioc/html/EBImage.html>

# Reading GIS data

- rgdal - <http://cran.r-project.org/web/packages/rgdal/index.html>
- rgeos - <http://cran.r-project.org/web/packages/rgeos/index.html>
- raster - <http://cran.r-project.org/web/packages/raster/index.html>

# Reading music data

- tuneR - <http://cran.r-project.org/web/packages/tuneR/>
- seewave - <http://rug.mnhn.fr/seewave/>