

How To Be a Real Data Monkey Hacking the Infochimps API with R

Drew Conway

December 16, 2010

Introduction

What is infochimps?

- ▶ Data clearinghouse
- ▶ API

The infochimps R package

- ▶ Basic usage framework
- ▶ Looking at the guts

Examples

- ▶ Geo-location of blog hits
- ▶ Programming language mentions on Twitter



DISCLAIMER: I do not work for Infochimps

- ▶ I just think they're awesome
- ▶ Much more info at <http://infochimps.com/about/>

DISCLAIMER: I do not work for Infochimps

- ▶ I just think they're awesome
- ▶ Much more info at <http://infochimps.com/about/>

Mission

We make lists, spreadsheets and datasets easy to find and monkey around with.

DISCLAIMER: I do not work for Infochimps

- ▶ I just think they're awesome
- ▶ Much more info at <http://infochimps.com/about/>

Mission

We make lists, spreadsheets and datasets easy to find and monkey around with.

Data clearinghouse

- ▶ Buy and sell data sets
- ▶ Handle all overhead
- ▶ Many free data sets, very useful for researchers



Explore > Datasets Collections Tags sell solutions search

Retrosheet: Game Logs (box scores) for Major League Baseball Games

A record of major league games played from 1871-2008

The game logs contain a record of major league games played from 1871-2008. At a minimum, it provides a listing of the date and score of each game. Where our research is more complete, we include information such as team statistics, winning and losing pitchers, linescores, attendance, starting pitchers, umpires and more. There are 161 fields in each record, described in more detail in the Guide to Retrosheet Game Logs.

Visit source

This data is not yet in the Infochimps repository – please continue your journey offline to get it from its primary source.

Infochimps API

The screenshot shows the Infochimps API website. The header includes navigation links: Home, Sign Up, Data Suppliers, Documentation, and Help and Feedback. The main heading is "Infochimps API BETA" with a bar chart background. Below this, there's a section titled "Up-to-date" with the text "No need to keep scraping. No importing, no exporting - just works. Sign up for the beta >" and a "See Features and Pricing" button. A vertical sidebar on the right says "THERE'S AN API FOR THAT". The main content area features three columns of services: 1. "trstrank" showing a user's rank of 9.7 and a description of the algorithm. 2. "Digital Element IP to Geo Data" describing geolocation data services. 3. "Influencer Metrics" describing analytics for influencers. Below these are three more sections: "Conversations" with an icon of two people talking, "Wordbag" with a word cloud icon, and "And more!" with a list of other services like "Twitter Strong Links" and "Twitter Word Usage".

Infochimps Data API for Twitter

api.infochimps.com

Home Sign Up Data Suppliers Documentation Help and Feedback

Infochimps API BETA

Up-to-date

No need to keep scraping. No importing, no exporting - just works.
Sign up for the beta >

See Features and Pricing

1 2 3 4 5 6

trstrank

@aplusk's rank is:
9.7

Using an algorithm similar to Google PageRank, Infochimps rates the influence of a user using a single number. Check out [test.me](#) for a demo. [More info >](#)

Digital Element IP to Geo Data

digital element
Knowing is Elemental™

The industry's leading provider of IP geolocation data is now available through our API. Target users by demographic, geolocation, ISP and more. [More info >](#)

Influencer Metrics

Improve your product with the most powerful Twitter analytics available. Check out [test.me](#) for a demo. [More info >](#)

Conversations

Find who is talking to who on Twitter and discover where conversation is taking place, not just who is

Wordbag

Want to target Twitter users by keyword or find what a particular Twitter user talks about? Wordbag

And more!

Check out our documentation for all the rest:

- Twitter Strong Links
- Twitter Word Usage
- Twholes
- MaxMind GeoLite IP Census

THERE'S AN API FOR THAT

Infochimps API

The screenshot shows the Infochimps API website. The header includes navigation links: Home, Sign Up, Data Suppliers, Documentation, and Help and Feedback. The main heading is "Infochimps API BETA" with a bar chart background. Below this is a section titled "Up-to-date" with the text "No need to keep scraping. No importing, no exporting - just works. Sign up for the beta >" and a "See Features and Pricing" button. A vertical sidebar on the right says "THERE'S AN API FOR THAT". The main content area features several service cards: "trstrank" showing a user's rank of 9.7, "Digital Element IP to Geo Data" described as the industry's leading provider, "Influencer Metrics" for improving products with analytics, "Conversations" for finding who is talking to whom, "Wordbag" for targeting users by keyword, and "And more!" with a link to documentation. A vertical sidebar on the right says "THERE'S AN API FOR THAT".

Infochimps Data API for Twitter
api.infochimps.com

Home Sign Up Data Suppliers Documentation Help and Feedback

Infochimps API BETA

Up-to-date
No need to keep scraping. No importing, no exporting - just works.
Sign up for the beta >

See Features and Pricing

1 2 3 4 5 6

trstrank

@aplusk's rank is:
9.7

Using an algorithm similar to Google PageRank, Infochimps rates the influence of a user using a single number. Check out [test.me](#) for a demo. [More info >](#)

Digital Element IP to Geo Data

digital element
Knowing is Elemental™

The industry's leading provider of IP geolocation data is now available through our API. Target users by demographic, geolocation, ISP and more. [More info >](#)

Influencer Metrics

Improve your product with the most powerful Twitter analytics available. Check out [test.me](#) for a demo. [More info >](#)

Conversations

Find who is talking to who on Twitter and discover where conversation is taking place, not just who is

Wordbag

Want to target Twitter users by keyword or find what a particular Twitter user talks about? Wordbag

And more!

Check out our documentation for all the rest:

- Twitter Strong Links
- Twitter Word Usage
- Twholes
- MaxMind GeoLite IP Census

THERE'S AN API FOR THAT

For more info see vid by Flip Kromer: <http://vimeo.com/16819171>

infochimps R package

Idea: create functions for every API call to integrate querying in R

- ▶ My first package accepted to CRAN!
- ▶ <http://cran.r-project.org/web/packages/infochimps/>
- ▶ Update as new ones API calls roll-out

infochimps R package

Idea: create functions for every API call to integrate querying in R

- ▶ My first package accepted to CRAN!
- ▶ <http://cran.r-project.org/web/packages/infochimps/>
- ▶ Update as new ones API calls roll-out

Inspired by other R API wrappers

- ▶ `twitterR` by Jeff Gentry
- ▶ `iBrokers` by Jeffrey Ryan
- ▶ `nytR` by Shane Conway (archived)

infochimps R package

Idea: create functions for every API call to integrate querying in R

- ▶ My first package accepted to CRAN!
- ▶ <http://cran.r-project.org/web/packages/infochimps/>
- ▶ Update as new ones API calls roll-out

Inspired by other R API wrappers

- ▶ twitterR by Jeff Gentry
- ▶ iBrokers by Jeffrey Ryan
- ▶ nytr by Shane Conway (archived)

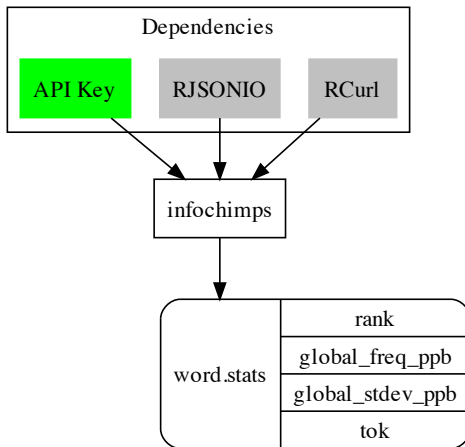
Twitter related

- ▶ conversations
- ▶ influence
- ▶ strong.links
- ▶ trstrank
- ▶ word.bag
- ▶ word.stats

Geo-location related

- ▶ census
- ▶ demographics
- ▶ domain
- ▶ ip.geo

Basic usage framework



My first infochimps call

Generate infochimps session

```
> library(infochimps)
> my.api <- "some.long.alpha.numeric"
> ic <- infochimps(my.api)
```

Get statistics for word "data"

```
> data.stats <- word.stats("data", ic)
> print(data.stats)
```

```
$global_stdev_ppb
[1] 2376464
```

```
$range
[1] 0.01266617
```

```
$tok
[1] "data"
```

```
$global_freq_ppb
[1] 151562.4
```

My first infochimps call

Generate infochimps session

```
> library(infochimps)
> my.api <- "some.long.alpha.numeric"
> ic <- infochimps(my.api)
```

Get statistics for word "data"

```
> data.stats <- word.stats("data", ic)
> print(data.stats)
```

```
$global_stdev_ppb
[1] 2376464
```

```
$range
[1] 0.01266617
```

```
$tok
[1] "data"
```

```
$global_freq_ppb
[1] 151562.4
```

The word.stats function

```
word.stats <-  
function(tok, session) {  
  word.url<-paste(session$base,"word_stats.json?tok=",  
    tok,"&apikey=",session$api.key,sep="")  
  word.get<-getURL(word.url)  
  word.data<-fromJSON(word.get)  
  # Simple error checking  
  if(is.null(word.data$error)) {  
    return(word.data)  
  }  
  else {  
    warning(word.data$message[[1]])  
    return(NA)  
  }  
}
```

The word.stats function

```
word.stats <-  
function(tok, session) {  
  word.url<-paste(session$base,"word_stats.json?tok=",  
    tok,"&apikey=",session$api.key,sep="")  
  word.get<-getURL(word.url)  
  word.data<-fromJSON(word.get)  
  # Simple error checking  
  if(is.null(word.data$error)) {  
    return(word.data)  
  }  
  else {  
    warning(word.data$message[[1]])  
    return(NA)  
  }  
}
```

All function follow this basic framework

Visualizing the location of blog visitors

The `ip.geo` function provides detailed geo-location data for a given IP address

- ▶ City, metro, country and continent codes (with confidence)
- ▶ Zip codes
- ▶ Latitude/Longitude
- ▶ Much more...

Visualizing the location of blog visitors

The `ip.geo` function provides detailed geo-location data for a given IP address

- ▶ City, metro, country and continent codes (with confidence)
- ▶ Zip codes
- ▶ Latitude/Longitude
- ▶ Much more...

Using web log data from <http://drewconway.com/zia>, visualize one days worth of blog hits

1. Parse log file by IP address and date/time
2. Use `ip.geo` to find lat/long for each hit
3. Plot on map using `ggplot2`

Step 1: Parse log file

```
# Load libraries
library(infochimps)
library(ggplot2)
library(maps)

# Need to load and clean the data
log.data<-read.delim("data/drewconway_com-Dec-2010.txt",
  sep=" ", header=FALSE, as.is=TRUE)
log.data<-data.frame(list("IP"=log.data$V1, "Date.Time"=log.data$V3,
  "Log"=log.data$V4), stringsAsFactors=FALSE)
log.data$IP<-gsub(" ", "", log.data$IP)

# First, get the dates in useable format
log.data$Date.Time<-gsub("[\\[_]" , "", log.data$Date.Time)
log.data$Date.Time<-strptime(log.data$Date.Time, format="%d/%b/%Y:%H:%M:%S ")

# Filter out only those logs accessing the right blog post
log.data<-log.data[ grep("(\\?p\\|=|index\\.php)", log.data$Log) ,]
```

Step 2: Get lat/long data

```
# Create infochimps session
api.key<-"my.long.alpha.numeric"
ic<-infochimps(api.key)

# Get latitude and longitude data for all of the IPs
ips<-unique(log.data$IP)

get.latlong<-function(ip) {
  geo.data<-ip.geo(ip,ic)
  return(c(ip, geo.data$lat,geo.data$longitude))
}

# Create data frame to merge into log data
geo.data<-lapply(ips, get.latlong)
geo.df<-as.data.frame(do.call("rbind", geo.data),stringsAsFactors=FALSE)
names(geo.df)<-c("IP","Latitude","Longitude")
log.geo<-merge(log.data,geo.df,by="IP")
log.geo$Latitude<-as.numeric(log.geo$Latitude)
log.geo$Longitude<-as.numeric(log.geo$Longitude)

# Create counts, and sort chronologically
log.count<-ddply(log.geo,.(IP, Date.Time, Latitude, Longitude),
  summarise, Count=length(Log))
log.count<-log.count[with(log.count, order(Date.Time)),]
```

Step 3: Visualize on global map

```
# Ready to visualize
world.map<-data.frame(map(plot=FALSE)[c("x","y")])

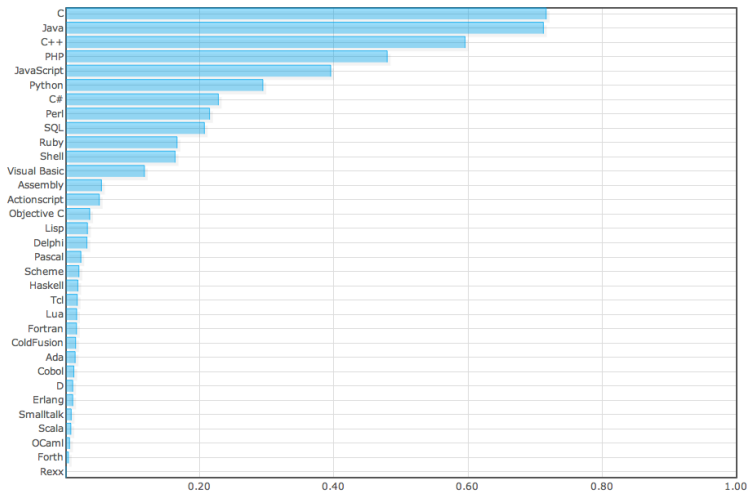
# Create frame for every second in data
plot.num<-1
for(d in strftime(log.count$Date.Time)) {
  log.sub<-log.count[which(strftime(log.count$Date.Time)==d),]
  geo.plot<-ggplot(world.map, aes(x=x,y=y))+geom_path(aes(colour='grey'))
  geo.plot<-geo.plot+geom_point(data=log.sub, aes(x=Longitude, y=Latitude,
    color='red', alpha=0.75, size=Count))+
    annotate('text',x=-125,y=-5,label=strftime(d, format='%H:%M:%S'))+
    theme_bw()+scale_colour_manual(values=c('grey'='grey','red'='red'),
      legend=FALSE)+
    scale_alpha(legend=FALSE)+scale_size(legend=FALSE)+
    coord_map(projection='lagrange',ylim=c(-40,70),xlim=c(-145,155))+
    opts(panel.grid.major=theme_blank(),axis.ticks=theme_blank(),
      axis.text.x=theme_blank(),axis.text.y=theme_blank())+
    xlab('')+ylab('')
  ggsave(plot=geo.plot, filename=paste('images/maps/',plot.num,'.png',sep=''),
    width=6,height=4)
  plot.num<-plot.num+1
}

# Run this at the command-line to join the files as a movie
# ffmpeg -f image2 -r 5 -i images/maps/%d.png -b 600k blogpost.mp4
```

The movie

Blog hits map

How do people tweet about different languages?



Source: <http://langpop.com/>, Last updated Sat Nov 27 08:45:50

Use the `word.stats` function

The `word.stats` function returns token frequency data

- ▶ Global frequency (parts per-billion)
- ▶ Standard deviation of frequency (parts per-billion)
- ▶ Range (normalized number of unique users who have used it)

Use the `word.stats` function

The `word.stats` function returns token frequency data

- ▶ Global frequency (parts per-billion)
- ▶ Standard deviation of frequency (parts per-billion)
- ▶ Range (normalized number of unique users who have used it)

Use the global frequency to create chart for Twitter mentions

1. Create vector of computer languages
2. Use `word.stats` to collect frequency data
3. Plot bar chart with `ggplot2`

Use the `word.stats` function

The `word.stats` function returns token frequency data

- ▶ Global frequency (parts per-billion)
- ▶ Standard deviation of frequency (parts per-billion)
- ▶ Range (normalized number of unique users who have used it)

Use the global frequency to create chart for Twitter mentions

1. Create vector of computer languages
2. Use `word.stats` to collect frequency data
3. Plot bar chart with `ggplot2`

Disadvantage: cannot get stats for languages like C, C, C++

Getting the data

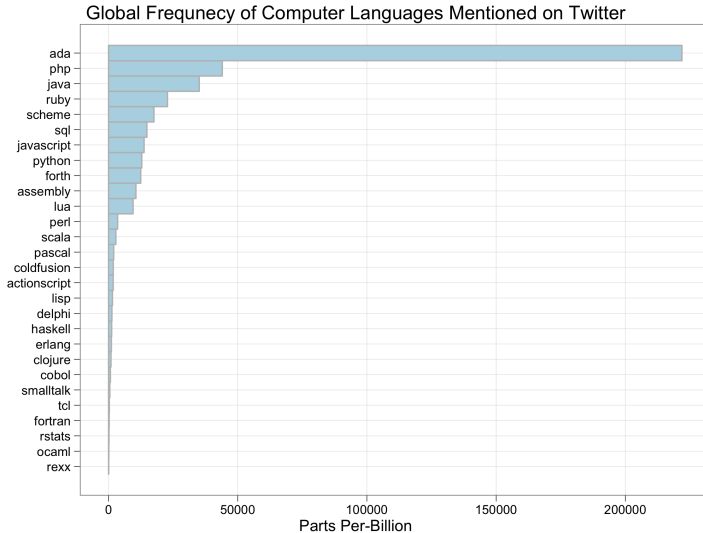
```
# rogramming languages
prog.langs<-c("java","php","javascript","python","sql","perl","ruby",
  "actionscript","assembly","lisp","delphi","pascal","scheme","haskell",
  "tcl","lua","fortran","coldfusion","ada","cobol","erlang","smalltalk",
  "scala","ocaml","forth","rexx","rstats")

# Get word stats for all languages
lang.stats<-lapply(prog.langs,function(t) unlist(word.stats(t,ic)))
lang.df<-as.data.frame(do.call("rbind",lang.stats), stringsAsFactors=FALSE)
lang.df$global_stdev_ppb<-as.numeric(lang.df$global_stdev_ppb)
lang.df$range<-as.numeric(lang.df$range)
lang.df$global_freq_ppb<-as.numeric(lang.df$global_freq_ppb)

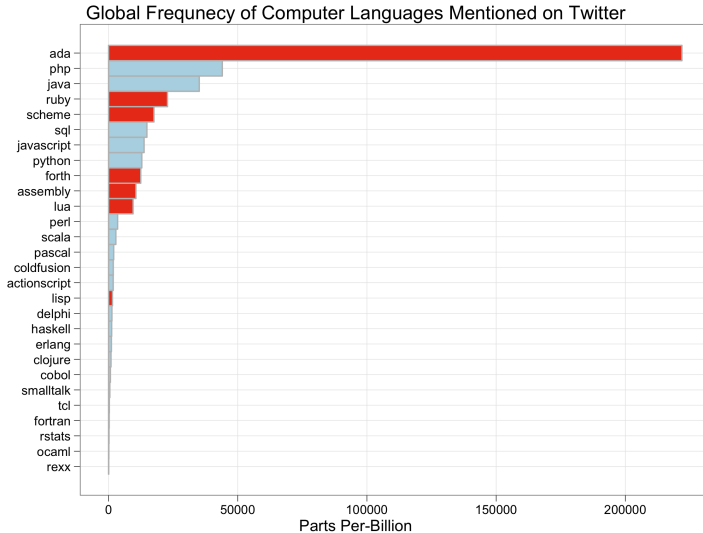
# Dummy for common words
common<-rep(0,nrow(lang.df))
common[match(c("ruby","assembly","lisp","scheme","ada","forth"),lang.df$tok)]<-1
lang.df$common<-as.factor(common)

# Sort by frequency
lang.df<-lang.df[with(lang.df, order(global_freq_ppb)),]
```

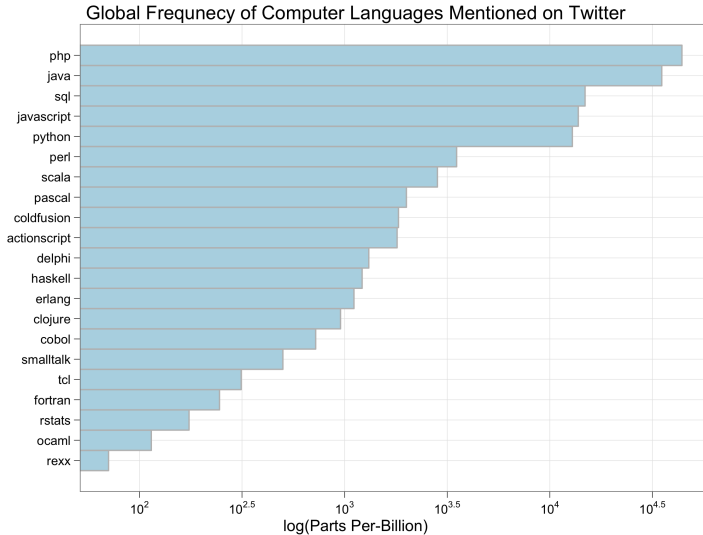
Visualizing computer languages mentioned on Twitter



Ambiguous terms



Terms cleans and logs taken



Thank You!

E-mail: drew.conway@nyu.edu
Web: <http://drewconway.com/zia>
Twitter: [@drewconway](https://twitter.com/drewconway)