Package 'infochimps'

November 22, 2010

Type Package

Title An R wrapper for the infochimps.com API services

Version 0.1.2

Date 2010-11-22

Author Drew Conway

Maintainer Drew Conway <drew.conway@nyu.edu>

Depends RCurl, RJSONIO

Description This package provides functions to access all of the APIs
currently available infochimps.com. For more information see http://api.infochimps.com/.

License BSD

LazyLoad yes

R topics documented:

infochimps-package 2
census 2
conversations 3
demographics 5
domain 6
influence 7
infochimps 8
ip.geo 9
strong.links 10
trstrank 11
word.bag 12
word.stats 14

Index **16**

infochimps-package *An R wrapper for the infochimps.com API services*

Description

This package provides functions to access all of the APIs currently available infochimps.com.

Details

Package:	infochimps
Type:	Package
Version:	0.1.2
Date:	2010-11-22
License:	BSD
LazyLoad:	yes

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/.

Examples

```
library(infochimps)

my.infochimps<-infochimps("your.api.key")
drew<-influence("drewconway",my.infochimps)
```

census *Gather U.S. Census data for a given IP address*

Description

A function to return combined census data for a given IP address using the inforchimps.com APIs

Usage

```
census(ip.address, session)
```

Arguments

ip.address	Properly formatted IP address as character string
session	Object containing session API key information, created with infochimps() function

Value

list : see reference for listing of all data returned (extensive)

Author(s)

Drew Conway, <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/web/an/ip_census/combined

Examples

```
my.infochimps<-infochimps("your.api.key")
nyu<-census("128.122.79.165",my.infochimps)

## The function is currently defined as
function(ip.address,session) {
    census.url<-paste(session$ip.url,"combined.json?ip=",ip.address,"&apikey=",session$ap
    census.get<-getURL(census.url)
    census.data<-fromJSON(census.get)
    if(is.null(census.data$error)) {
        return(census.data)
    }
    else {
        warning(census.data$message[[1]])
        return(NA)
    }
  }
```

---

conversations *Create data frame of recent conversations between two Twitter users*

Description

A function to return the interactions between two Twitter users with infochimps.com API

Usage

```
conversations(screen.name.a, screen.name.b, session, user.id.a = NA, user.id.b =
```

Arguments

screen.name.a	The name of a Twitter user
screen.name.b	The name of a Twitter user
session	Object containing session API key information, created with infochimps() function
user.id.a	a Twitter user ID
user.id.b	a Twitter user ID

Value

Data frame with the following columns:

`user.id.a`	First Twitter user (numeric)
`user.id.b`	Second Twitter user (numeric)
`conversation.id`	Internal Twitter ID for tweet (numeric)
`conversation.type`	Factor describing conversation type (factor). See ref.
`reply.to.id`	If RE type, internal Twitter ID for reply-to tweet (numeric)

If user.name not found, or no data, return NA

Author(s)

Drew Conway, <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/conversation

Examples

```
my.infochimps<-infochimps("my.api.key")
jd.tweets<-conversations("drewconway","CMastication",my.infochimps)
head(jd.tweets)

## The function is currently defined as
function(screen.name.a,screen.name.b,session,user.id.a=NA,user.id.b=NA) {
    if(is.na(user.id.a) & is.na(user.id.a)) {
        conversation.url<-paste(session$base,"conversation.json?user_a_sn=",screen.name.a
    }
    else {
        if(is.na(user.id.a)==FALSE & is.na(user.id.a)==FALSE) {
            conversation.url<-paste(session$base,"conversation.json?user_a_id=",user.id.a
        }
        else {
            if(is.na(user.id.na)) {
                conversation.url<-paste(session$base,"conversation.json?user_a_sn=",scree
            }
            else {
                conversation.url<-paste(session$base,"conversation.json?user_a_id=",user.
            }
        }
    }
    conversation.get<-getURL(conversation.url)
    # Fix JSON for proper handling for conversation IDs
    conversation.get<-gsub("([0-9]+)","\"\1\"\2",conversation.get,perl=TRUE)
    conversation.data<-fromJSON(conversation.get)
    # Simple error checking
    if(is.null(conversation.data$error)) {
        user.id.a<-conversation.data$user_a_id[[1]]
        user.id.b<-conversation.data$user_b_id[[1]]
        conversations.matrix<-suppressWarnings(do.call("rbind", conversation.data$convers
        if(dim(conversations.matrix)[2]<3) {
```

```
            # JSON request returns no reply-to data
            reply.to<-NA
        }
        else {
            reply.to<-sapply(1:nrow(conversations.matrix), function(x) ifelse(conversatic
        }
        conversations.df<-cbind(user.id.a, user.id.b, conversations.matrix[,1], conversat
        conversations.df<-as.data.frame(conversations.df)
        names(conversations.df)<-c("user.id.a","user.id.b","conversation.id","conversatic
        return(conversations.df)
    }
    else {
        warning(conversation.data$message[[1]])
        return(NA)
    }
}
```

`demographics` *Gather demographic data for a given IP address from the U.S. Census*

Description

A function to return infochimps.com census data for a given IP address from the Digital Elements IP data and U.S. censu data with infochimps.com APIs.

Usage

```
demographics(ip.address, session)
```

Arguments

`ip.address`	Properly formatted IP address as character string
`session`	Object containing session API key information, created with infochimps() function

Value

list : see reference for listing of all data returned (extensive)

Author(s)

Drew Conway, <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/web/an/de/demographics

Examples

```
my.infochimps<-infochimps("your.api.key")
nyu<-demographics("128.122.79.165",my.infochimps)

## The function is currently defined as
function(ip.address,session) {
    demographics.url<-paste(session$de.url,"demographics.json?ip=",ip.address,"&apikey=",
    demographics.get<-getURL(demographics.url)
    demographics.data<-fromJSON(demographics.get)
    if(is.null(demographics.data$error)) {
        return(demographics.data)
    }
    else {
        warning(demographics.data$message[[1]])
        return(NA)
    }
  }
```

`domain` *Return domain information for a given domian*

Description

A function to return Digitial Elements IP domain data from the infochimps.com API

Usage

```
domain(ip.address, session)
```

Arguments

`ip.address`	Properly formatted IP address as character string
`session`	Object containing session API key information, created with infochimps() function

Value

A list containing the following elements:

`domain`	Domain name (character)
`company`	Registered company name (character)
`isp`	Internet service provider (character)
`proxy_type`	Proxy type (character)
`naics_code`	NAICS Code (numeric)

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/web/an/de/domain

Examples

```
my.infochimps<-infochimps("your.api.key")
nyu<-domain("128.122.79.165",my.infochimps)

## The function is currently defined as
function(ip.address,session) {
    domain.url<-paste(session$de.url,"domain.json?ip=",ip.address,"&apikey=",session$api.
    domain.get<-getURL(domain.url)
    domain.data<-fromJSON(domain.get)
    if(is.null(domain.data$error)) {
        return(domain.data)
    }
    else {
        warning(domain.data$message[[1]])
        return(NA)
    }
  }
```

`influence` *Find the level of influence for a given Twitter user*

Description

A function to return infochimps.com influence scores for a Twitter user

Usage

```
influence(screen.name, session, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`session`	Object containing session API key information, created with infochimps() function
`user.id`	a Twitter user ID

Value

list : see reference for listing of all data returned (extensive)

Author(s)

Drew Conway, <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/influence

Examples

```
my.infochimps<-infochimps("your.api.key")
drew<-influence("drewconway",my.infochimps)

## The function is currently defined as
function(screen.name,session,user.id=NA) {
    if(is.na(user.id)) {
        influence.url<-paste(session$base.url,"influence.json?screen_name=",screen.name,"
    }
    else{
        influence.url<-paste(session$base.url,"influence.json?user_id=",user.id,"&apikey=
    }
    influence.get<-getURL(influence.url)
    influence.data<-fromJSON(influence.get)
    # Simple error checking
    if(is.null(influence.data$error)){
        return(influence.data)
    }
    else {
        warning(influence.data$message[[1]])
        return(NA)
    }
  }
```

---

infochimps *Create an infochimps.com API session.*

Description

List object to hold a user's API key, as well as all API URLS. Needed in all functions

Usage

```
infochimps(api.key)
```

Arguments

api.key	A valid infochimps.com API key

Value

api.key	A valid infochimps.com API key
base	Base URL for most infochimps.com API calls
de	Base URL for Digital Elements API calls
ip	Base URL for IP address related API calls

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

To get an API key from infochimps.com see, http://api.infochimps.com/about/features-and-pricing

Examples

```
my.infochimps<-infochimps("your.api.key")

## The function is currently defined as
function(api.key)
    structure(list(api.key=api.key,
                   base="http://api.infochimps.com/soc/net/tw/",
                   de="http://api.infochimps.com/web/an/de/",
                   ip="http://api.infochimps.com/web/an/ip_census/"),
    class="infochimps")
```

---

ip.geo *IP address geo-location*

Description

A function to return Digitial Elements IP Intelligence geo-loaction data from the infochimps.com API

Usage

```
ip.geo(ip.address, session)
```

Arguments

ip.address	Properly formatted IP address as character string
session	Object containing session API key information, created with infochimps() function

Value

list : see reference for listing of all data returned (extensive)

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/web/an/de/geo

Examples

```
my.infochimps<-infochimps("your.api.key")
nyu<-ip.geo("128.122.79.165",my.infochimps)

## The function is currently defined as
function(ip.address,session) {
    geo.url<-paste(session$de.url,"geo.json?ip=",ip.address,"&apikey=",session$api.key,se
    geo.get<-getURL(geo.url)
    geo.data<-fromJSON(geo.get)
    if(is.null(geo.data$error)) {
        return(geo.data)
```

```
    }
    else {
        warning(geo.data$message[[1]])
        return(NA)
    }
}
```

`strong.links` *Find all of the Strong Links of a given Twitter user*

Description

A function to return infochimps.com Strong Links data

Usage

```
strong.links(screen.name, session, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`session`	Object containing session API key information, created with infochimps() function
`user.id`	a Twitter user ID

Value

A data frame with the following columns:

`user.id`	Twitter user ID (numeric)
`strong.link`	Twitter user ID with Strong Link (numeric)
`link.weight`	Strength of Strong Link (numeric)

If user.name not found, return NA

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/strong_links

Examples

```
my.infochimps<-infochimps("your.api.key")
drew.links<-strong.links("drewconway",my.infochimps)
head(drew.links)

## The function is currently defined as
function(screen.name,session,user.id=NA) {
    if(is.na(user.id)) {
        strong.url<-paste(session$base.url,"strong_links.json?screen_name=",screen.name,"
    }
    else{
        strong.url<-paste(session$base.url,"strong_links.json?user_id=",user.id,"&apikey=
    }
    strong.get<-getURL(strong.url)
    strong.data<-fromJSON(strong.get)
    # Simple error checking
    if(is.null(strong.data$error)){
        strong.edges<-do.call("rbind",strong.data$strong_links)
        strong.edges<-cbind(strong.data$user_id,strong.edges)
        strong.df<-as.data.frame(strong.edges)
        names(strong.df)<-c("user.id","strong.edge","link.weight")
        return(strong.df)
    }
    else{
        warning(strong.data$message[[1]])
        return(NA)
    }
  }
```

`trstrank` *Get the trstrank score for a given Twitter user*

Description

A function to return infochimps.com trstrank score for a Twitter user

Usage

```
trstrank(screen.name, session, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`session`	Object containing session API key information, created with infochimps() function
`user.id`	a Twitter user ID

Value

A list with the following elements:

`user_id`	A Twitter user ID (numeric)
`screen_name`	Screen name of a Twitter user (character)

trstrank	trstrank score (numeric)
tq	trstrank quotient (numeric)

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/trstrank

Examples

```
my.infochimps<-infochimps("your.api.key")
trstrank("drewconway",my.infochimps)

## The function is currently defined as
function(screen.name,session,user.id=NA) {
    if(is.na(user.id)) {
        trstrank.url<-paste(session$base.url,"trstrank.json?screen_name=",screen.name,"&a
    }
    else{
        trstrank.url<-paste(session$base.url,"trstrank.json?user_id=",user.id,"&apikey=",
    }
    trstrank.get<-getURL(trstrank.url)
    trstrank.data<-fromJSON(trstrank.get)
    # Simple error checking
    if(is.null(trstrank.data$error)) {
        return(trstrank.data)
    }
    else {
        warning(trstrank.data$message[[1]])
        return(NA)
    }
  }
```

---

word.bag *Find the words most associated with a given Twitter user*

Description

A function to return infochimps.com Word Bag for a Twitter user

Usage

```
word.bag(screen.name, session, user.id = NA)
```

Arguments

screen.name	The name of a Twitter user
session	Object containing session API key information, created with infochimps() function
user.id	a Twitter user ID

Value

A list with the following elements:

`user_id`	Twitter used ID (numeric)
`vocab`	Number of distinct tokens ever emitted (numeric)
`total.usages`	Total number of tokens emitted (numeric)
`tok.df`	Data frame with columns: user.id (numeric), rel.freq (numeric), tok user (character), freq.ppb (numeric)

If user.name not found, return NA

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/wordbag

Examples

```
my.infochimps<-infochimps("your.api.key")
hilary<-word.bag("hmason",my.infochimps)

## The function is currently defined as
function(screen.name,session,user.id=NA) {
    if(is.na(user.id)) {
        wordbag.url<-paste(session$base.url,"wordbag.json?screen_name=",screen.name,"&api
    }
    else{
        wordbag.url<-paste(session$base.url,"wordbag.json?user_id=",user.id,"&apikey=",se
    }
    wordbag.get<-getURL(wordbag.url)
    wordbag.data<-fromJSON(wordbag.get)
    if(is.null(wordbag.data$error)) {
        # Get wordbag data
        words<-do.call("rbind", wordbag.data$toks)
        words.df<-as.data.frame(cbind(wordbag.data$user_id[[1]],words))
        names(words.df)<-c("user.id","rel.freq","tok","user.freq.ppb")
        words.list<-list(user.id=wordbag.data$user_id[[1]],vocab=wordbag.data$vocab[[1]],
        return(words.list)
    }
    else {
        warning(wordbag.data$message[[1]])
        return(NA)
    }
  }
```

`word.stats` *Get basic statistics associated with a given word on Twitter*

Description

A function to return infochimps.com Word Stats data

Usage

```
word.stats(tok, session)
```

Arguments

`tok`	The word you are searching (character)
`session`	Object containing session API key information, created with infochimps() function

Value

A list with the following elements:

`global_stdev_ppb`
: Standard deviation (numeric)

`range`
: Range (numeric)

`tok`
: The word (character)

`global_freq_ppb`
: Global frequency in parts-per-billion (numeric)

If tok not found, return NA

Author(s)

Drew Conway <drew.conway@nyu.edu>

References

http://api.infochimps.com/describe/soc/net/tw/word_stats

Examples

```
my.infochimps<-infochimps("your.api.key")
word.stats("infochimps",my.infochimps)

## The function is currently defined as
function(tok,session) {
    word.url<-paste(session$base.url,"word_stats.json?tok=",tok,"&apikey=",session$api.ke
    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)
    }
}
```

Index

∗Topic **datagen**
census, 2
conversations, 3
demographics, 5
domain, 6
influence, 7
ip.geo, 9
strong.links, 10
trstrank, 11
word.bag, 12
word.stats, 14
∗Topic **list**
infochimps, 8
∗Topic **package**
infochimps-package, 2

census, 2
conversations, 3

demographics, 5
domain, 6

influence, 7
infochimps, 8
infochimps-package, 2
ip.geo, 9

strong.links, 10

trstrank, 11

word.bag, 12
word.stats, 14