Package 'infochimps'

December 17, 2010

Type Package

Title An R wrapper for the infochimps.com API services

Version 0.2.1

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 *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)

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

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)
```

Arguments

ip.address	Properly formatted IP address as character string

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

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

## The function is currently defined as
function(ip.address) {
    census.url<-paste(.InfochimpsEnv$data$ip,"combined.json?ip=",ip.address,"&apikey=",.I
    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, user.id.a = NA, user.id.b = NA)
```

Arguments

screen.name.a	The name of a Twitter user
screen.name.b	The name of a Twitter user
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

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

## The function is currently defined as
function(screen.name.a,screen.name.b,user.id.a=NA,user.id.b=NA) {
    if(is.na(user.id.a) & is.na(user.id.a)) {
        conversation.url<-paste(.InfochimpsEnv$data$base,"conversation.json?user_a_sn=",s
    }
    else {
        if(is.na(user.id.a)==FALSE & is.na(user.id.a)==FALSE) {
            conversation.url<-paste(.InfochimpsEnv$data$base,"conversation.json?user_a_ic
        }
        else {
            if(is.na(user.id.na)) {
                conversation.url<-paste(.InfochimpsEnv$data$base,"conversation.json?user_
            }
            else {
                conversation.url<-paste(.InfochimpsEnv$data$base,"conversation.json?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(conversatio
        }
        conversations.df<-cbind(user.id.a, user.id.b, conversations.matrix[,1], conversat
        conversations.df<-as.data.frame(conversations.df,stringsAsFactors=FALSE)
        conversation.names<-c("user.id.a","user.id.b","conversation.id","conversation.typ
        names(conversations.df)<-conversation.names
        for(c in 1:length(conversation.names)) {conversations.df[,c]<-unlist(conversation
        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)
```

Arguments

`ip.address`	Properly formatted IP address as character string

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

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

## The function is currently defined as
function(ip.address) {
    demographics.url<-paste(.InfochimpsEnv$data$de,"demographics.json?ip=",ip.address,"&a
    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)
```

Arguments

`ip.address`	Properly formatted IP address as character string

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

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

## The function is currently defined as
function(ip.address) {
    domain.url<-paste(.InfochimpsEnv$data$de,"domain.json?ip=",ip.address,"&apikey=",.Inf
    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, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`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

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

## The function is currently defined as
function(screen.name,user.id=NA) {
    if(is.na(user.id)) {
        influence.url<-paste(.InfochimpsEnv$data$base,"influence.json?screen_name=",scree
    }
    else{
        influence.url<-paste(.InfochimpsEnv$data$base,"influence.json?user_id=",user.id,"
    }
    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

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

```
infochimps("your.api.key")

## The function is currently defined as
function(api.key) {
    if(is.character(api.key)){
        .InfochimpsEnv$data$api.key<-api.key
    }
    else{
        warning("API key must be a string")
    }
}
```

---

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)
```

Arguments

ip.address	Properly formatted IP address as character string

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

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

## The function is currently defined as
function(ip.address) {
    geo.url<-paste(.InfochimpsEnv$data$de,"geo.json?ip=",ip.address,"&apikey=",.Infochimp
    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, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`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

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

## The function is currently defined as
function(screen.name,user.id=NA) {
    if(is.na(user.id)) {
        strong.url<-paste(.InfochimpsEnv$data$base,"strong_links.json?screen_name=",scree
    }
    else{
```

```
        strong.url<-paste(.InfochimpsEnv$data$base,"strong_links.json?user_id=",user.id,"
    }
    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$user_id,strong.edges)
        strong.df<-as.data.frame(strong.edges, stringsAsFactors=FALSE)
        strong.names<-c("user.id","strong.edge","link.weight")
        names(strong.df)<-strong.names
        for(c in 1:length(strong.names)) {strong.df[,c]<-unlist(strong.df[,c])}
        return(strong.df)
    }
    else{
        warning(strong.data$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, user.id = NA)
```

Arguments

`screen.name`	The name of a Twitter user
`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

```
infochimps("your.api.key")
trstrank("drewconway")

## The function is currently defined as
function(screen.name,user.id=NA) {
    if(is.na(user.id)) {
        trstrank.url<-paste(.InfochimpsEnv$data$base,"trstrank.json?screen_name=",screen.
    }
    else{
        trstrank.url<-paste(.InfochimpsEnv$data$base,"trstrank.json?user_id=",user.id,"&a
    }
    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, user.id = NA)
```

Arguments

screen.name	The name of a Twitter user
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

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

## The function is currently defined as
function(screen.name,user.id=NA) {
    if(is.na(user.id)) {
        wordbag.url<-paste(.InfochimpsEnv$data$base,"wordbag.json?screen_name=",screen.na
    }
    else{
        wordbag.url<-paste(.InfochimpsEnv$data$base,"wordbag.json?user_id=",user.id,"&api
    }
    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$user_id[[1]],words), stringsAsFactors=
        words.names<-c("user.id","rel.freq","tok","user.freq.ppb")
        names(words.df)<-words.names
        for(c in 1:length(word.names)) {words.df[,c]<-unlist(words.df[,c])}
        words.list<-list(user.id=wordbag.data$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)
```

Arguments

tok	The word you are searching (character)

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

```
infochimps("your.api.key")
word.stats("infochimps")

## The function is currently defined as
function(tok) {
    tok<-tolower(gsub("[[:punct:]]","",tok))
    word.url<-paste(.InfochimpsEnv$data$base,"word_stats.json?tok=",tok,"&apikey=",.Infoc
    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