

Package: TweetScraperR (via r-universe)

October 22, 2026

Type Package

Title Collect Posts and User Data from 'X' (Formerly 'Twitter')

Version 0.8.1.9000

Description Collects posts, user profiles, lists, followers and related metadata from the social network 'X' (formerly 'Twitter') and returns them as tidy data frames. Data are retrieved through a local 'Node.js' engine that drives a 'Chromium' browser with 'Playwright', reusing the sessions of the user's own accounts. Work can be distributed across several accounts of the user while respecting the per-endpoint rate limits reported by 'X'; the design of this account pool is ported from the 'Python' library 'twscrape' [\(<https://github.com/vladkens/twscrape>](https://github.com/vladkens/twscrape)). Also includes helpers to plot word, emoji and time frequencies, and optional sentiment and image analysis through the 'OpenAI' API. Documentation is written in Spanish.

URL <https://github.com/agusnieto77/TweetScraperR>,
<https://agusnieto77.github.io/TweetScraperR/>

BugReports <https://github.com/agusnieto77/TweetScraperR/issues>

Depends R (>= 4.1.0)

Imports rvest (>= 1.0.4), dplyr (>= 1.0.0), tidyr, lubridate, tibble, quanteda, wordcloud2, ggplot2, stringr, purrr (>= 1.0.0), httr, xml2, utils, stats, jsonlite, htmltools, tools, cli, processx, DBI, RSQLite, askpass

Suggests testthat (>= 3.0.0), withr, knitr, rmarkdown, ggimage

VignetteBuilder knitr

Config/testthat/edition 3

License MIT + file LICENSE

Copyright See file inst/COPYRIGHTS

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements Node.js (>= 18) with npm. The 'Playwright' engine and the 'Chromium' browser are not installed with the package: the user downloads them explicitly by calling `installPlaywrightEngine()`.

Config/roxygen2/version 8.1.0

Config/pak/sysreqs chromium cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev

Repository <https://agusnieto77.r-universe.dev>

Date/Publication 2026-10-21 22:56:51 UTC

RemoteUrl <https://github.com/agusnieto77/TweetScraperR>

RemoteRef HEAD

RemoteSha 28059c2e1e141a03b4e0dbf84aabe34be848d451

Contents

accountsX	3
addAccountsX	3
checkAccountsX	4
checkPlaywrightEngine	5
extractTweetsData	5
getListMembersAPI	7
getListTweetsAPI	8
getTweetsDataAPI	8
getTweetsHashtags	9
getTweetsImagesAnalysis	10
getTweetsLiveAPI	12
getTweetsRepliesAPI	13
getTweetsRetweetsAPI	14
getTweetsSearchAPI	15
getTweetsSentiments	16
getTweetsTimelinesAPI	17
getUserFollowersAPI	18
getUserFollowingAPI	19
getUserMediaAPI	20
getUsersAboutAPI	20
getUsersDataAPI	21
getUsersSearchAPI	22
getUserTweetsAPI	23
HTMLImgReport	24
importSessionX	25
installAgentSkill	26
installPlaywrightEngine	27
loginX	28
plotEmojis	29
plotEmojisPNG	30
plotTime	31
plotWords	31

accountsX

3

reactivateAccountX

32

removeAccountX

33

resetLocksX

34

setProxyX

34

setupAccountsX

35

Index

36

accountsX	Estado del pool de cuentas de X
-----------	---------------------------------

Description

Lista las cuentas importadas con `importSessionX()`, si estan activas, en pausa (y hasta cuando) por endpoint, y cuantas respuestas aportaron.

Usage

`accountsX()`

Value

Un tibble con una fila por cuenta y columnas: cuenta, usuario_x (@usuario real, lo completa `checkAccountsX()`), activa, pausas (endpoints en pausa y hora en que se liberan), requests (total de respuestas obtenidas), requests_por_endpoint, ultimo_uso, proxy, error.

Examples

```
## Not run:
accountsX()

## End(Not run)
```

addAccountsX	Agregar varias cuentas al pool de una vez
--------------	---

Description

Agregar varias cuentas al pool de una vez

Usage

`addAccountsX(accounts)`

Arguments

`accounts` Un data frame, o la ruta a un CSV, con las columnas `account`, `auth_token` y `ct0`, y opcionalmente `proxy`. El separador del CSV (,, ; o tabulador) se detecta solo, así que sirven los CSV guardados por Excel en configuración regional en español.

Value

Invisiblemente, los nombres de las cuentas agregadas.

Examples

```
## Not run:
# cuentas.csv: account,auth_token,ct0,proxy
addAccountsX("cuentas.csv")
accountsX()

## End(Not run)
```

checkAccountsX

Validar las sesiones del pool de cuentas

Description

Abre X con la sesión de cada cuenta y comprueba que siga autenticada, registrando el @usuario real que hay detrás (columna `usuario_x` de `accountsX()`). Las cuentas con sesión vencida o bloqueada se desactivan; las válidas se reactivan. No consume cupo de los endpoints de recolección. Si dos cuentas del pool resultan ser la misma cuenta de X (p. ej. la "default" de versiones previas cargada de nuevo con otro nombre), se reescribe: queda la agregada más recientemente, con las pausas y estadísticas de ambas.

Usage

```
checkAccountsX(accounts = NULL)
```

Arguments

`accounts` Nombres de las cuentas a validar. Por defecto, todas (incluidas las desactivadas).

Value

Un tibble con una fila por cuenta y columnas `cuenta`, `valida`, `usuario_x` y `motivo` (por que no es válida).

Examples

```
## Not run:
importSessionX(cookies = Sys.getenv("X2_COOKIES"), account = "cuenta2")
checkAccountsX()

## End(Not run)
```

checkPlaywrightEngine	<i>Comprobar que el motor Node/Playwright esta instalado y operativo</i>
-----------------------	--

Description

Ejecuta el comando doctor del motor y devuelve la informacion de versiones.

Usage

```
checkPlaywrightEngine()
```

Value

Invisiblemente, una lista con ok, version de Playwright, version de Node y ruta de Chromium.
Lanza un error si el motor no responde.

Examples

```
## Not run:
checkPlaywrightEngine()

## End(Not run)
```

extractTweetsData	<i>Extracts Relevant Information from Locally Stored Tweets</i>
-------------------	---

Description

Esta función procesa un conjunto de tweets almacenados localmente y extrae información relevante de cada uno. Puede manejar tanto un dataframe como una lista que contenga el HTML de los tweets, sus URLs correspondientes y las fechas de captura.

Usage

```
extractTweetsData(data)
```

Arguments

data	Un dataframe o una lista que contiene tres elementos: 'art_html' (el contenido HTML de los tweets), 'url' (las URLs de los tweets) y 'fecha_captura' (la fecha y hora de captura de cada tweet).
------	--

Details

La función utiliza expresiones XPath y selectores CSS para extraer información específica de cada tweet. Procesa cada tweet individualmente y maneja posibles errores, permitiendo continuar con el procesamiento incluso si algunos tweets fallan. La función ahora puede manejar tanto URLs de Twitter como de X.com. Se han añadido nuevas extracciones, como enlaces a videos de YouTube y se ha mejorado la extracción de emoticones.

Value

Un tibble con las siguientes columnas:

- fecha: La fecha y hora del tweet.
- username: El nombre de usuario del autor del tweet.
- texto: El texto principal del tweet.
- tweet_citado: El texto del tweet citado, si existe.
- user_citado: El nombre de usuario del autor del tweet citado, si existe.
- emoticones: Una lista de emoticones utilizados en el tweet.
- links_img_user: El enlace a la imagen de perfil del usuario.
- links_img_post: Una lista de enlaces a las imágenes incluidas en el tweet.
- links_youtube: Una lista de enlaces a videos de YouTube mencionados en el tweet.
- respuestas: El número de respuestas al tweet.
- reposteos: El número de reposteos del tweet.
- megustas: El número de "me gusta" del tweet.
- metricas: Información adicional sobre las métricas del tweet.
- urls: Una lista de URLs mencionadas en el tweet.
- hilo: Indica si el tweet es parte de un hilo (basado en el número de respuestas).
- url: La URL original del tweet.
- fecha_captura: La fecha y hora en que se capturó la información del tweet (heredada de los datos de entrada).

Examples

```
## Not run:
# Usando un dataframe
tweets_data <- data.frame(
  art_html = c("<html>...</html>", "<html>...</html>"),
  url = c("https://twitter.com/user1/status/123", "https://x.com/user2/status/456"),
  fecha_captura = c("2023-01-01 12:00:00", "2023-01-02 13:00:00")
)
resultados <- extractTweetsData(tweets_data)

# Usando una lista
tweets_list <- list(
  art_html = c("<html>...</html>", "<html>...</html>"),
  url = c("https://twitter.com/user1/status/123", "https://x.com/user2/status/456"),
```

```
    fecha_captura = c("2023-01-01 12:00:00", "2023-01-02 13:00:00")
  )
  resultados <- extractTweetsData(tweets_list)

## End(Not run)
```

getListMembersAPI	<i>Get the Members of an X List via the X API (experimental)</i>
-------------------	--

Description

Recupera las cuentas que integran una Lista de X consultando la **API GraphQL interna de X** (ListMembers). Requiere una sesion importada con [importSessionX\(\)](#).

Usage

```
getListMembersAPI(list_id, n_users = 100, dir = getwd(), save = FALSE)
```

Arguments

list_id	Id numerico de la lista o su URL (<a href="https://x.com/i/lists/<id>">https://x.com/i/lists/<id>).
n_users	Numero maximo de usuarios a recuperar. Por defecto 100.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con una fila por usuario (mismas columnas que [getUsersDataAPI\(\)](#)).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getListMembersAPI("53645372")

## End(Not run)
```

getListTweetsAPI	<i>Get the Tweets of an X List via the X API (experimental)</i>
------------------	---

Description

Recupera los tweets mas recientes de una Lista de X consultando la **API GraphQL interna de X** (ListLatestTweetsTimeline). Requiere una sesion importada con `importSessionX()`.

Usage

```
getListTweetsAPI(list_id, n_tweets = 40, dir = getwd(), save = FALSE)
```

Arguments

<code>list_id</code>	Id numerico de la lista o su URL (<code>https://x.com/i/lists/<id></code>).
<code>n_tweets</code>	Numero maximo de tweets a recuperar. Por defecto 40.
<code>dir</code>	Directorio de destino del RDS. Por defecto el de trabajo.
<code>save</code>	Logico. Si TRUE guarda el resultado en un RDS dentro de <code>dir</code> . Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con un tweet por fila (mismas columnas que `getUserTweetsAPI()`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getListTweetsAPI("https://x.com/i/lists/53645372", n_tweets = 100)

## End(Not run)
```

getTweetsDataAPI	<i>Get Tweet Data from URLs via the X API (experimental)</i>
------------------	--

Description

Recupera los datos de tweets a partir de sus URLs consultando la **API GraphQL interna de X** (TweetDetail). Devuelve datos estructurados del JSON. Es el reemplazo basado en API de `getTweetsData()`. Requiere una sesion importada con `importSessionX()`.

Usage

```
getTweetsDataAPI(urls_tweets, dir = getwd(), save = FALSE)
```

Arguments

urls_tweets	Vector de URLs de tweets (formato <code>https://x.com/u/status/123</code>).
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con una fila por tweet (mismas columnas que `getUserTweetsAPI`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getTweetsDataAPI("https://x.com/sanchezcastejon/status/2111426246517072370")

## End(Not run)
```

getTweetsHashtags	<i>Get Hashtags from Tweets</i>
-------------------	---------------------------------

Description

Esta función toma un dataframe de tweets y extrae todos los hashtags del campo 'texto' o 'tweet', añadiéndolos como una nueva columna al dataframe.

Usage

```
getTweetsHashtags(df)
```

Arguments

df	Un dataframe que contiene una columna 'texto' o 'tweet' con el contenido de los tweets.
----	---

Value

Un dataframe con una nueva columna 'hashtags' que contiene, para cada tweet, un vector de caracteres con sus hashtags. Los tweets sin hashtags devuelven `character(0)`. Nota: en versiones anteriores los hashtags llegaban anidados en una lista adicional (`x[[i]][[1]]`) y los tweets sin hashtags devolvían NA.

Examples

```
df1 <- data.frame(
  texto = c("Este es un #tweet con #hashtags", "Este no tiene hashtags", "Otro #ejemplo")
)
getTweetsHashtags(df1)

df2 <- data.frame(
  tweet = c("Este es un #tweet con #hashtags", "Este no tiene hashtags", "Otro #ejemplo")
)
getTweetsHashtags(df2)
```

`getTweetsImagesAnalysis`*Analyze images of tweets*

Description

Esta función analiza imágenes asociadas con tweets. Procesa un vector de URLs de imágenes o rutas de archivos locales, envía estas imágenes a la API de OpenAI para su análisis y devuelve un informe detallado sobre el contenido de cada imagen.

Usage

```
getTweetsImagesAnalysis(
  img_sources,
  modelo = "gpt-4o-mini",
  api_key = Sys.getenv("OPENAI_API_KEY"),
  dir = getwd(),
  save = FALSE
)
```

Arguments

<code>img_sources</code>	Un vector de caracteres que contiene las URLs de las imágenes o rutas de archivos locales a analizar.
<code>modelo</code>	Una cadena de caracteres que contiene el modelo de OpenAI. Por defecto es "gpt-4o-mini".
<code>api_key</code>	Una cadena de caracteres que contiene la clave de API de OpenAI. Por defecto es <code>Sys.getenv("OPENAI_API_KEY")</code> .
<code>dir</code>	Una cadena de caracteres que especifica el directorio donde se guardan los resultados cuando <code>save = TRUE</code> . Por defecto es <code>getwd()</code> .
<code>save</code>	Logico. Si <code>TRUE</code> guarda los resultados (totales y parciales) en archivos RDS dentro de <code>dir</code> . Por defecto <code>FALSE</code> : no se escribe ningún archivo.

Details

La función realiza los siguientes pasos:

1. Verifica la presencia de una clave de API válida.
2. Procesa el vector de URLs de imágenes o rutas de archivos locales, eliminando valores NA.
3. Convierte los archivos locales a formato base64 si es necesario.
4. Define un prompt de sistema detallado para la API de OpenAI, instruyéndola sobre cómo analizar las imágenes.
5. Para cada imagen:
 - Envía una solicitud a la API de OpenAI con la URL de la imagen o los datos base64.
 - Procesa la respuesta de la API para extraer los resultados del análisis.
6. Combina todos los resultados en un solo tibble.
7. Si `save = TRUE`, guarda los resultados como archivos RDS en el directorio especificado, con resultados parciales cada 7 imágenes si el total es mayor a 7.

El análisis de la imagen incluye:

- Clasificación del tipo de imagen (por ejemplo, foto, meme, captura de pantalla)
- Detección de texto en la imagen (limitado a 10 palabras si es extenso)
- Identificación de contenido discriminatorio, violento, pornográfico o inapropiado
- Una descripción detallada del contenido de la imagen
- Generación de palabras clave descriptivas

Value

Un tibble con las siguientes columnas:

<code>clasificacion</code>	La clasificación de la imagen
<code>contiene_texto</code>	Un booleano que indica si la imagen contiene texto
<code>texto_contenido</code>	El contenido de texto de la imagen (si lo hay, limitado a 10 palabras)
<code>contenido_discriminatorio</code>	Un booleano que indica la presencia de contenido discriminatorio
<code>contenido_violento</code>	Un booleano que indica la presencia de contenido violento
<code>contenido_pornografico</code>	Un booleano que indica la presencia de contenido pornográfico
<code>contenido_inapropiado</code>	Un booleano que indica la presencia de contenido inapropiado
<code>descripcion</code>	Una descripción detallada de la imagen
<code>palabras_clave</code>	Una cadena de texto con 2 a 4 palabras clave separadas por punto y coma
<code>img</code>	La URL o nombre del archivo de la imagen analizada

Note

Esta función requiere una conexión a internet activa y una clave de API de OpenAI válida para funcionar correctamente. La función está diseñada para manejar errores y continuar procesando otras imágenes si una falla. El tiempo de espera para las solicitudes HTTP está configurado en 300 segundos (5 minutos). Tenga en cuenta que las imágenes analizadas (sus URLs o su contenido codificado en base64) se transmiten a la API de OpenAI (un servicio de terceros).

Examples

```
## Not run:
# Ejemplo de uso con un vector de URLs de imágenes
urls <- c("https://ejemplo.com/imagen1.jpg", "https://ejemplo.com/imagen2.jpg")
resultados <- getTweetsImagesAnalysis(urls)

# Ejemplo de uso con rutas de archivos locales
archivos_locales <- c("./imagen1.jpg", "./imagen2.png")
resultados_locales <- getTweetsImagesAnalysis(archivos_locales)

## End(Not run)
```

getTweetsLiveAPI

Capture Tweets from a Search as They Are Published (experimental)

Description

Sigue una búsqueda "hacia adelante": repite `getTweetsSearchAPI()` (pestanía "Latest") cada `interval` segundos durante `duration` segundos y acumula solo los tweets nuevos. Desde la segunda ronda la consulta lleva el operador `since_id`: con el ID mas reciente ya recolectado, así que X devuelve solo lo publicado despues y no se vuelve a gastar cupo en lo ya traído.

No es un streaming real: la granularidad es `interval`. Si en una ronda aparecen `n_tweets` o mas tweets nuevos, los que exceden el tope quedan sin recolectar (se avisa con un mensaje); en ese caso conviene bajar `interval` o subir `n_tweets`. Un error en una ronda se informa y la captura sigue con la proxima. Interrumpir con Esc / Ctrl+C termina la captura y devuelve lo acumulado hasta ese momento.

Usage

```
getTweetsLiveAPI(
  search = "#RStats",
  interval = 300,
  duration = 3600,
  n_tweets = 100,
  max_tweets = Inf,
  dir = getwd(),
  save = FALSE
)
```

Arguments

search	Consulta de búsqueda (soporta operadores de X, p.ej. "lang:es", "-filter:retweets"). Por defecto "#RStats".
interval	Segundos entre rondas. Mínimo 60. Por defecto 300.
duration	Segundos totales de captura. Por defecto 3600 (una hora).
n_tweets	Tope de tweets por ronda. La primera ronda trae hasta n_tweets de los más recientes como punto de partida. Por defecto 100.
max_tweets	La captura termina al alcanzar este total acumulado. Por defecto Inf (solo limita duration).
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Lógico. Si TRUE guarda lo acumulado en un RDS dentro de dir después de cada ronda (si la sesión de R se corta, lo recolectado no se pierde). Por defecto FALSE: no se escribe ningún archivo.

Value

Un tibble con un tweet por fila, del más nuevo al más viejo (mismas columnas que `getTweetsSearchAPI()`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
# Una hora de captura, una ronda cada 5 minutos, guardando en cada ronda
vivo <- getTweetsLiveAPI("#RStats", interval = 300, duration = 3600, save = TRUE)

## End(Not run)
```

getTweetsRepliesAPI *Get Tweet Replies / Thread via the X API (experimental)*

Description

Recupera el tweet y sus respuestas (hilo de conversación) consultando la **API GraphQL interna de X**. Devuelve datos estructurados del JSON. Requiere una sesión importada con `importSessionX()`.

Usage

```
getTweetsRepliesAPI(url, n_tweets = 40, dir = getwd(), save = FALSE)
```

Arguments

url	URL del tweet del cual obtener las respuestas.
n_tweets	Número máximo de tweets (tweet + respuestas) a recuperar. Por defecto 40.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Lógico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningún archivo.

Value

Un tibble con un tweet por fila (mismas columnas que `getUserTweetsAPI()`, incluyendo media, hashtags, menciones, etc.).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getTweetsRepliesAPI("https://x.com/sanchezcastejon/status/2111426246517072370", n_tweets = 100)

## End(Not run)
```

`getTweetsRetweetsAPI` *Get the Users Who Retweeted a Tweet via the X API (experimental)*

Description

Recupera los usuarios que repostearon un tweet consultando la **API GraphQL interna de X** (Retweeters). Requiere una sesión importada con `importSessionX()`.

Usage

```
getTweetsRetweetsAPI(url, n_users = 100, dir = getwd(), save = FALSE)
```

Arguments

<code>url</code>	URL del tweet.
<code>n_users</code>	Número máximo de usuarios a recuperar. Por defecto 100.
<code>dir</code>	Directorio de destino del RDS. Por defecto el de trabajo.
<code>save</code>	Lógico. Si TRUE guarda el resultado en un RDS dentro de <code>dir</code> . Por defecto FALSE: no se escribe ningún archivo.

Value

Un tibble con una fila por usuario (mismas columnas que `getUserDataAPI()`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getTweetsRetweetsAPI("https://x.com/sanchezcastejon/status/2111426246517072370", n_users = 100)

## End(Not run)
```

getTweetsSearchAPI *Search Tweets via the X API (experimental)*

Description

Busca tweets que coinciden con una consulta consultando la **API GraphQL interna de X** (en lugar de scrapear HTML). Devuelve datos estructurados del JSON: texto completo, fecha exacta y metricas (respuestas, retweets, citas, me gusta, vistas). Requiere una sesion importada con [importSessionX\(\)](#).

Usage

```
getTweetsSearchAPI(
  search = "#RStats",
  n_tweets = 40,
  product = c("Latest", "Top", "Media"),
  since = NULL,
  until = NULL,
  dir = getwd(),
  save = FALSE
)
```

Arguments

search	Consulta de busqueda (soporta operadores de X, p.ej. "from:NASA", "#RStats", "lang:es"). Por defecto "#RStats".
n_tweets	Numero maximo de tweets a recuperar. Por defecto 40.
product	Pestania de resultados: "Latest" (recientes, por defecto), "Top" (destacados) o "Media".
since	Desde cuando (opcional). Un dia "YYYY-MM-DD" (incluye ese dia) o un instante "YYYY-MM-DD HH:MM[:SS]" en UTC (tambien con _ o T como separador y su-fijo _UTC). Acepta objetos Date y POSIXct. Con hora se traduce al operador since_time: de X.
until	Hasta cuando (opcional), mismos formatos que since. Un dia solo incluye ese dia; con hora se traduce a until_time:.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con un tweet por fila (mismas columnas que [getUserTweetsAPI\(\)](#), incluyendo media, hashtags, menciones, etc.).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
tw <- getTweetsSearchAPI("#RStats", n_tweets = 100)
getTweetsSearchAPI("from:NASA artemis", product = "Top")

## End(Not run)
```

getTweetsSentiments	<i>Analyze sentiments of tweets</i>
---------------------	-------------------------------------

Description

Esta función toma un vector de tweets y utiliza la API de OpenAI para realizar un análisis de sentimiento detallado de cada tweet. El análisis incluye el tono, el sentimiento general, la presencia de expresiones de odio, si el tweet está direccionado a alguien específico y si contiene un llamado a la acción.

Usage

```
getTweetsSentiments(
  tweets,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  model = "gpt-4o-mini",
  dir = getwd(),
  save = FALSE
)
```

Arguments

tweets	Un vector de caracteres que contiene los tweets a analizar.
api_key	Una cadena de caracteres con la clave de API de OpenAI. Por defecto, se intenta obtener de la variable de entorno OPENAI_API_KEY.
model	Una cadena de caracteres que especifica el modelo de OpenAI a utilizar. Por defecto es "gpt-4o-mini".
dir	Una cadena de caracteres que especifica el directorio donde se guardan los archivos RDS con los resultados cuando save = TRUE. Por defecto es el directorio de trabajo actual.
save	Logico. Si TRUE guarda los resultados (totales y parciales) en archivos RDS dentro de dir. Por defecto FALSE: no se escribe ningún archivo.

Details

La función realiza las siguientes operaciones:

1. Verifica que se haya proporcionado una clave de API válida.

2. Define un prompt detallado para el análisis de los tweets.
3. Define una función interna `analyzeTweet` que procesa cada tweet individualmente.
4. Procesa los tweets en lotes de 250 para manejar grandes volúmenes de datos.
5. Guarda resultados parciales cada 50 tweets procesados.
6. Devuelve los resultados como un tibble.

La función utiliza la API de OpenAI para realizar el análisis de sentimiento, por lo que requiere una conexión a Internet y una clave de API válida.

Value

Un tibble con los resultados del análisis para cada tweet. Cada fila contiene el tweet original y los resultados del análisis (tono, sentimiento, presencia de expresiones de odio, si está direccionado, si contiene un llamado a la acción y una explicación detallada).

Note

Esta función transmite el contenido de los tweets analizados a la API de OpenAI (un servicio de terceros). Tenga en cuenta esta transferencia de datos al trabajar con contenido de terceros o datos personales.

Examples

```
## Not run:
tweets <- c(
  "¡Qué día tan maravilloso! \U0001f60a",
  "Odio este producto, nunca lo compren. \U0001f620"
)
resultados <- getTweetsSentiments(tweets)
print(resultados)

## End(Not run)
```

`getTweetsTimelinesAPI` *Get the Combined Timeline of Several Users via the X API (experimental)*

Description

Recupera y combina los timelines de varias cuentas consultando la **API GraphQL interna de X**, en una sola sesión de navegador (batch). Util para monitorear un conjunto curado de cuentas (lo que harías con una Lista de X, pero sin necesitar un `list_id`). Devuelve un único tibble ordenado por fecha (más reciente primero) y deduplicado. Requiere una sesión importada con `importSessionX()`.

Usage

```
getTweetsTimelinesAPI(usernames, n_tweets = 40, dir = getwd(), save = FALSE)
```

Arguments

usernames	Vector de nombres de usuario (acepta "NASA", "@NASA" o la URL completa "https://x.com/NASA").
n_tweets	Numero maximo de tweets a recuperar por cuenta . Por defecto 40.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con un tweet por fila (mismas columnas que `getUserTweetsAPI()`), combinado y ordenado por fecha.

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getTweetsTimelinesAPI(c("elravignani", "NucleoIdaes", "BNMMArentina"), n_tweets = 100)

## End(Not run)
```

getUserFollowersAPI	<i>Get a User's Followers via the X API (experimental)</i>
---------------------	--

Description

Recupera los seguidores de un usuario consultando la **API GraphQL interna de X** (Followers). Requiere una sesion importada con `importSessionX()`.

Usage

```
getUserFollowersAPI(username, n_users = 100, dir = getwd(), save = FALSE)
```

Arguments

username	Nombre de usuario (sin @).
n_users	Numero maximo de usuarios a recuperar. Por defecto 100.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con una fila por usuario (mismas columnas que getUsersDataAPI).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUserFollowersAPI("rstatstweet", n_users = 200)

## End(Not run)
```

getUserFollowingAPI	<i>Get the Accounts a User Follows via the X API (experimental)</i>
---------------------	---

Description

Recupera las cuentas que sigue un usuario consultando la **API GraphQL interna de X** (Following). Requiere una sesión importada con `importSessionX()`.

Usage

```
getUserFollowingAPI(username, n_users = 100, dir = getwd(), save = FALSE)
```

Arguments

username	Nombre de usuario (sin @).
n_users	Número máximo de usuarios a recuperar. Por defecto 100.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Lógico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningún archivo.

Value

Un tibble con una fila por usuario (mismas columnas que getUsersDataAPI).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUserFollowingAPI("rstatstweet", n_users = 200)

## End(Not run)
```

getUserMediaAPI	<i>Get a User's Media Tweets via the X API (experimental)</i>
-----------------	---

Description

Recupera los tweets con media (fotos/videos) de unx usuariix consultando la **API GraphQL interna de X** (UserMedia). Devuelve datos estructurados del JSON. Requiere una sesion importada con `importSessionX()`.

Usage

```
getUserMediaAPI(username = "NASA", n_tweets = 40, dir = getwd(), save = FALSE)
```

Arguments

username	Nombre de usuariix (sin @). Por defecto "NASA".
n_tweets	Numero maximo de tweets a recuperar. Por defecto 40.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con un tweet por fila (mismas columnas que `getUserTweetsAPI`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUserMediaAPI("NASA", n_tweets = 100)

## End(Not run)
```

getUsersAboutAPI	<i>Get the "About this account" Data of Users via the X API (experimental)</i>
------------------	--

Description

Recupera la ficha "Acerca de esta cuenta" de unx o varixs usuariixs consultando la **API GraphQL interna de X** (AboutAccountQuery): el pais desde el que opera la cuenta, cuantas veces cambio de nombre de usuariix, desde cuando esta verificada, etc. Util para estudiar redes de cuentas coordinadas o de origen extranjero. Requiere una sesion importada con `importSessionX()`. X no publica todos los campos para todas las cuentas; los ausentes quedan en NA.

Usage

```
getUsersAboutAPI(usernames, dir = getwd(), save = FALSE)
```

Arguments

usernames	Vector de nombres de usuario (acepta "NASA", "@NASA" o la URL completa).
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con una fila por usuario y columnas user, nombre, user_id, pais_cuenta, ubicacion_precisa, cambios_usuario, ultimo_cambio_usuario, afiliada_a, origen_cuenta, identidad_verificada, verificada_desde, fecha_creacion y url.

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUsersAboutAPI(c("NASA", "rstatstweet"))

## End(Not run)
```

getUsersDataAPI

*Get a User's Profile Data via the X API (experimental)***Description**

Recupera los datos de perfil de uno o varios usuarios consultando la **API GraphQL interna de X** (UserByScreenName). Requiere una sesion importada con `importSessionX()`.

Usage

```
getUsersDataAPI(usernames, dir = getwd(), save = FALSE)
```

Arguments

usernames	Vector de nombres de usuario (sin @).
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con una fila por usuario y columnas user, nombre, user_id, descripcion, seguidores, siguiendo, tweets, favoritos, verificado, ubicacion, sitio_web, imagen_perfil, banner, fecha_creacion, url, tweets_media, protegida y tipo_verificacion.

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUsersDataAPI(c("NASA", "rstatstweet"))

## End(Not run)
```

getUsersSearchAPI	<i>Search Users via the X API (experimental)</i>
-------------------	--

Description

Busca cuentas (pestanía "Personas" del buscador de X) consultando la **API GraphQL interna de X** (SearchTimeline). Requiere una sesión importada con `importSessionX()`.

Usage

```
getUsersSearchAPI(search, n_users = 40, dir = getwd(), save = FALSE)
```

Arguments

search	Consulta de búsqueda (nombre, tema, etc.).
n_users	Numero máximo de usuarios a recuperar. Por defecto 40.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Lógico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningún archivo.

Value

Un tibble con una fila por usuario (mismas columnas que `getUsersDataAPI`).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
getUsersSearchAPI("rstats", n_users = 50)

## End(Not run)
```

getUserTweetsAPI	<i>Get Tweets from a User Timeline via the X API (experimental)</i>
------------------	---

Description

Recupera los tweets del timeline de un usuario consultando la **API GraphQL interna de X** (en lugar de scrapear HTML). Devuelve datos estructurados directamente del JSON: texto completo (sin truncar), fecha exacta, y metricas (respuestas, retweets, citas, me gusta, vistas). Requiere una sesion importada con `importSessionX()`.

Usage

```
getUserTweetsAPI(username = "NASA", n_tweets = 40, dir = getwd(), save = FALSE)
```

Arguments

username	Nombre de usuario (sin @). Por defecto "NASA".
n_tweets	Numero maximo de tweets a recuperar. Por defecto 40.
dir	Directorio de destino del RDS. Por defecto el de trabajo.
save	Logico. Si TRUE guarda el resultado en un RDS dentro de dir. Por defecto FALSE: no se escribe ningun archivo.

Value

Un tibble con un tweet por fila y columnas: fecha, user, texto (completo, tambien en tweets largos), idioma, respuestas, retweets, citas, me gustas, views, emoticones (lista), hashtags (lista), menciones (lista), urls_externas (lista), media (lista de URLs), media_tipo (lista: photo/video/animated_gif), es_retweet, es_cita, tweet_citado_id, conversation_id, url, tweet_id, user_id (autor), respuesta_a_id, respuesta_a_user, retweet_id (tweet original de un retweet), guardados, fuente (app de publicacion), sensible, editado, lugar y pais (geoetiqueta, si la hay).

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
tw <- getUserTweetsAPI("NASA", n_tweets = 100)

## End(Not run)
```

HTMLImgReport

*Generate HTML visualization of analyzed images***Description**

#'

Esta función crea una visualización HTML interactiva de imágenes analizadas, mostrando cada imagen junto con su clasificación, descripción, palabras clave y banderas de contenido en un diseño responsivo con un carrusel y miniaturas.

Usage

HTMLImgReport(results, output_file)

Arguments

results	Un data frame o tibble que contiene los resultados del análisis de imágenes de la función <code>getTweetsImagesAnalysis</code>. Debe incluir las siguientes columnas: • <code>clasificacion</code>: La clasificación de la imagen (character) • <code>descripcion</code>: Una descripción detallada de la imagen (character) • <code>palabras_clave</code>: Palabras clave que describen la imagen (character) • <code>contiene_texto</code>: Indica si la imagen contiene texto (logical) • <code>texto_contenido</code>: El texto contenido en la imagen, si lo hay (character) • <code>contenido_discriminatorio</code>: Indica si hay contenido discriminatorio (logical) • <code>contenido_violento</code>: Indica si hay contenido violento (logical) • <code>contenido_pornografico</code>: Indica si hay contenido pornográfico (logical) • <code>contenido_inapropiado</code>: Indica si hay contenido inapropiado (logical) • <code>img</code>: La ruta o URL de la imagen (character)
output_file	Una cadena de caracteres con el nombre (o la ruta) del archivo HTML de salida. Es obligatorio: la función no escribe archivos en una ubicación por defecto.

Details

La función crea una página HTML responsiva utilizando Bootstrap para el diseño. Cada imagen se presenta en una tarjeta que incluye la imagen, su clasificación, descripción, palabras clave y banderas de contenido potencialmente problemático. El diseño se ajusta automáticamente a diferentes tamaños de pantalla.

La visualización incluye:

- Un título centrado que menciona la función `getTweetsImagesAnalysis`
- Un carrusel de tarjetas de imágenes
- Una vista de miniaturas de todas las imágenes
- Banderas de colores para contenido problemático
- Estilos CSS personalizados para mejorar la presentación

Value

Invisiblemente, la ruta normalizada del archivo HTML generado. Como efecto secundario escribe el archivo en output_file y muestra un mensaje de confirmacion.

Examples

```
## Not run:
# Asumiendo que tienes un data frame llamado 'resultados_analisis'
HTMLImgReport(resultados_analisis,
               output_file = file.path(tempdir(), "visualizacion_imagenes.html"))

## End(Not run)
```

importSessionX

Importar una sesion de X/Twitter desde las cookies de tu navegador

Description

La via recomendada para autenticarse es iniciar sesion a mano en tu navegador habitual y pasarle a esta funcion las cookies auth_token y ct0 de tu sesion (las encuentras en DevTools -> Application/Almacenamiento -> Cookies -> x.com). La sesion queda guardada en el pool de cuentas (en tools::R_user_dir("TweetScraperR", "data")) y todas las funciones de recoleccion la reusan sin volver a loguearse.

Usage

```
importSessionX(
  auth_token,
  ct0,
  account = "default",
  proxy = NULL,
  cookies = NULL
)
```

Arguments

auth_token	Valor de la cookie auth_token de x.com.
ct0	Valor de la cookie ct0 de x.com.
account	Nombre local de la cuenta en el pool. Por defecto "default". Reimportar con el mismo nombre renueva la sesion y reactiva la cuenta, conservando su proxy y estadisticas.
proxy	Proxy opcional de esta cuenta ("http://host:puerto", "usuario:clave@host:puerto" o "socks5://host:puerto"). Tiene prioridad sobre el proxy global options(TweetScraperR.proxy = ...).
cookies	Alternativa a auth_token/ct0: el header de cookies ("auth_token=...; ct0=...") o el JSON exportado por una extension de cookies (p. ej. Cookie-Editor).

Details

Importando varias cuentas propias (con distintos account) se arma un **pool** que distribuye la carga de trabajo entre ellas: cada cuenta respeta los límites de uso que X informa por endpoint (p. ej. 50 búsquedas cada 15 minutos) y, cuando una llega a su límite, queda en pausa hasta que X lo renueve mientras la recolección continúa con otra cuenta disponible. Ver [accountsX\(\)](#).

Value

Invisiblemente, el nombre de la cuenta.

Examples

```
## Not run:
importSessionX(auth_token = "...", ct0 = "...")
importSessionX(cookies = "auth_token=...; ct0=...", account = "cuenta2",
               proxy = "http://usuario:clave@host:8080")

## End(Not run)
```

installAgentSkill

Install the TweetScraperR Skill for AI Agents

Description

Instala una *skill* (formato **Agent Skills**) que le enseña a un agente de IA a usar TweetScraperR de forma conversacional: puesta a punto del motor y la sesión, qué función corresponde a cada pedido, columnas devueltas, recetas de análisis y solución de errores frecuentes.

Pensada en particular para **Posit Assistant** (el agente nativo de RStudio y Positron), que la carga automáticamente cuando el pedido trata de X/Twitter y también la expone como el comando `/tweetscraperr`. Sirve para cualquier cliente compatible con Agent Skills.

Usage

```
installAgentSkill(
  agent = "posit",
  scope = c("user", "project"),
  path = NULL,
  overwrite = FALSE
)
```

Arguments

agent	Agente(s) destino. Uno o más de: "posit" (por defecto): Posit Assistant, <code>.posit/assistant/skills/</code>. "agents": ubicación genérica del estándar Agent Skills, <code>.agents/skills/</code> (también la lee Posit Assistant, Codex y otros). "claude": Claude Code, <code>.claude/skills/</code>.
-------	--

scope	"user" (por defecto) instala en tu carpeta personal, disponible en todos los proyectos. "project" instala en el directorio de trabajo actual (raíz del proyecto), solo para ese proyecto.
path	Carpeta de skills personalizada. Si se indica, ignora agent y scope y crea path/tweetscraper/.
overwrite	Lógico. Si TRUE reemplaza una instalación previa de la skill (útil tras actualizar el paquete). Por defecto FALSE.

Details

En Windows la carpeta personal es %USERPROFILE% (no la carpeta Documentos que R usa como ~), que es donde los agentes buscan sus skills. Las versiones antiguas de Posit Assistant usaban ~/.positai/skills/: si tenés una de esas versiones, usá path = "~/.positai/skills" o actualizá el asistente.

La skill queda disponible en la siguiente conversación del agente.

Value

Invisiblemente, las rutas de las carpetas de la skill instaladas.

Examples

```
## Not run:
# Posit Assistant, para todos los proyectos
installAgentSkill()

# Posit Assistant y Claude Code, solo en el proyecto actual
installAgentSkill(c("posit", "claude"), scope = "project")

# Actualizar la skill tras reinstalar el paquete
installAgentSkill(overwrite = TRUE)

## End(Not run)
```

```
installPlaywrightEngine
```

Instalar el motor Node/Playwright (npm install + navegador)

Description

Copia el motor (inst/node) a la carpeta de cache del usuario (tools::R_user_dir("TweetScraperR", "cache")) y ejecuta allí npm install, que instala 'Playwright', 'playwright-extra' y su plugin 'stealth' (ajustes de compatibilidad para que el navegador automatizado se presente como un 'Chromium' de escritorio estandar) y descarga el navegador 'Chromium'. Nada de esto ocurre al instalar el paquete: hay que llamar a esta funcion explicitamente, una vez por maquina, antes de usar las funciones de recoleccion. Requiere 'Node.js' (>= 18) con 'npm' y conexion a internet.

Usage

```
installPlaywrightEngine(npm = "npm")
```

Arguments

npm Ruta al binario de npm (por defecto "npm" del PATH).

Value

Invisiblemente, el codigo de salida de npm (0 si todo salio bien).

Examples

```
## Not run:
installPlaywrightEngine()

## End(Not run)
```

loginX	<i>Iniciar sesion en X/Twitter con Playwright y guardar la sesion</i>
--------	---

Description

Abre un navegador 'Chromium' controlado por 'Playwright', realiza el login en X/Twitter con el flujo del modal actual y guarda la sesion autenticada (cookies + storage) como una cuenta del pool. Las demas funciones del paquete reusan esa sesion sin volver a loguearse, lo que evita inicios de sesion repetidos. La via recomendada es `importSessionX()`, que reutiliza la sesion de tu navegador habitual.

Usage

```
loginX(
  xuser = Sys.getenv("TWITTER_USER", Sys.getenv("USER")),
  xpass = Sys.getenv("TWITTER_PASS", Sys.getenv("PASS")),
  email = Sys.getenv("TWITTER_EMAIL", ""),
  headless = TRUE,
  account = "default",
  proxy = NULL
)
```

Arguments

xuser Usuario de X/Twitter. Por defecto la variable de entorno TWITTER_USER (con fallback a USER).

xpass Contraseña de X/Twitter. Por defecto la variable de entorno TWITTER_PASS (con fallback a PASS).

email	Correo para el paso de verificación de identidad, si X lo pide. Por defecto la variable de entorno TWITTER_EMAIL.
headless	Logico. Si TRUE (por defecto) corre el navegador sin interfaz. Pone FALSE para ver/depurar el login.
account	Nombre local de la cuenta en el pool. Por defecto "default".
proxy	Proxy opcional para rutear el tráfico (p. ej. en redes que exigen salir por un proxy). Acepta un string "http://host:puerto" o "usuario:clave@host:puerto", o una lista list(server=, username=, password=). Si es un string queda guardado como proxy de la cuenta.

Value

Invisiblemente, el nombre de la cuenta. Lanza un error si el login falla o si X limita el intento.

Examples

```
## Not run:
loginX(xuser = "mi_usuario", xpass = Sys.getenv("TWITTER_PASS"),
      headless = FALSE)

## End(Not run)
```

plotEmojis

Create Bar Chart of Emoticons in Tweets

Description

Esta función toma un dataframe de tweets y crea un gráfico de barras mostrando la frecuencia de los emoticones utilizados en los tweets.

Usage

```
plotEmojis(df, top_n = 10, fill = "skyblue", color = "grey30")
```

Arguments

df	Un dataframe que contiene una columna 'emoticones' con listas de emoticones.
top_n	Número de emoticones más frecuentes a mostrar (por defecto 10).
fill	Color de las barras en el gráfico (por defecto "skyblue").
color	Color de los bordes de las barras en el gráfico (por defecto "grey30").

Value

Un objeto ggplot con el gráfico de barras.

Examples

```
## Not run:
df <- data.frame(emoticones = I(list(
  c("\U0001F60A", "\U0001F602", "\U0001F602"),
  c("\U0001F60A"),
  c("\U0001F602", "\U0001F60D"),
  character(0)
)))
plotEmojis(df)

## End(Not run)
```

plotEmojisPNG

Create Bar Chart of EmoticonsPNG in Tweets

Description

Esta función toma un dataframe de tweets y crea un gráfico de barras mostrando la frecuencia de los emoticones en colores utilizados en los tweets.

Usage

```
plotEmojisPNG(df, top_n = 10, fill = "skyblue", color = "grey30")
```

Arguments

df	Un dataframe que contiene una columna 'emoticones' con listas de emoticones.
top_n	Número de emoticones más frecuentes a mostrar (por defecto 10).
fill	Color de las barras en el gráfico (por defecto "skyblue").
color	Color de los bordes de las barras en el gráfico (por defecto "grey30").

Value

Un objeto ggplot con el gráfico de barras.

Examples

```
## Not run:
df <- data.frame(emoticones = I(list(
  c("\U0001F60A", "\U0001F602", "\U0001F602"),
  c("\U0001F60A"),
  c("\U0001F602", "\U0001F60D"),
  character(0)
)))
plotEmojisPNG(df)

## End(Not run)
```

`plotTime`*Create Line Graph of Tweets by Time*

Description

Esta función toma un dataframe de tweets y crea un gráfico de líneas mostrando la frecuencia de tweets a lo largo del tiempo, con opciones para agrupar por hora, día, semana, mes o año.

Usage

```
plotTime(df, group_by = "hour", color = "blue")
```

Arguments

<code>df</code>	Un dataframe que contiene una columna 'fecha' con las fechas de los tweets.
<code>group_by</code>	Una cadena que indica cómo agrupar las fechas. Opciones válidas son "hour", "day", "week", "month", "year".
<code>color</code>	Color de la línea en el gráfico (por defecto "blue").

Value

Un objeto ggplot con el gráfico de líneas.

Examples

```
df <- data.frame(fecha = c(
  seq(as.POSIXct("2023-01-02 00:00:00"), by = "hour", length.out = 3),
  seq(as.POSIXct("2023-01-02 01:00:00"), by = "hour", length.out = 2),
  seq(as.POSIXct("2023-01-02 03:00:00"), by = "hour", length.out = 3),
  seq(as.POSIXct("2023-01-02 01:00:00"), by = "hour", length.out = 2),
  seq(as.POSIXct("2023-01-02 03:00:00"), by = "hour", length.out = 3),
  seq(as.POSIXct("2023-01-02 01:00:00"), by = "hour", length.out = 2),
  seq(as.POSIXct("2023-01-02 03:00:00"), by = "hour", length.out = 3)
))
plotTime(df, group_by = "hour")
```

`plotWords`*Create Word Cloud from Tweets*

Description

Esta función toma un dataframe de tweets y crea una nube de palabras basada en el contenido de la columna 'texto' o 'tweet'.

Usage

```
plotWords(
  df,
  min_freq = 3,
  max_words = 100,
  random_order = FALSE,
  colors = "random-dark",
  size = 0.3,
  lang = "es",
  sw = NULL
)
```

Arguments

df	Un dataframe que contiene una columna 'texto' o 'tweet' con el contenido de los tweets.
min_freq	Frecuencia mínima de palabras para incluir en la nube (por defecto 3).
max_words	Número máximo de palabras a incluir en la nube (por defecto 100).
random_order	Booleano, si las palabras deben ordenarse aleatoriamente (por defecto FALSE).
colors	Vector de colores para las palabras (por defecto 'random-dark').
size	Tamaño de la fuente (por defecto 0.3).
lang	idioma para las stopwords 'es', 'en', 'de', 'pt', etc. (por defecto 'es').
sw	vector de palabras extras para sumar a la lista de stopwords (por defecto NULL).

Value

Un objeto de tipo wordcloud2.

Examples

```
df <- data.frame(texto = c("Este es un tweet de ejemplo", "Otro tweet para la nube de palabras"))
plotWords(df, min_freq = 1)
```

reactivateAccountX	<i>Reactivar cuentas desactivadas del pool</i>
--------------------	--

Description

Una cuenta se desactiva cuando X rechaza su sesion o la limita. Si fue un falso positivo (o ya se resolvió desde el navegador), se reactiva con esta funcion. Si la sesion vencio, conviene reimportarla con `importSessionX()`.

Usage

```
reactivateAccountX(accounts)
```

Arguments

accounts Nombres de las cuentas a reactivar.

Value

Invisiblemente, la cantidad de cuentas reactivadas.

Examples

```
## Not run:  
reactivateAccountX("cuenta2")  
  
## End(Not run)
```

removeAccountX	<i>Eliminar cuentas del pool</i>
----------------	----------------------------------

Description

Borra las cuentas y sus archivos de sesion.

Usage

```
removeAccountX(accounts)
```

Arguments

accounts Nombres de las cuentas a eliminar.

Value

Invisiblemente, la cantidad de cuentas eliminadas.

Examples

```
## Not run:  
removeAccountX("cuenta2")  
  
## End(Not run)
```

resetLocksX	<i>Quitar las pausas de rate-limit de todas las cuentas</i>
-------------	---

Description

Util si una recoleccion se corto a la fuerza (Ctrl+C, error) y dejo cuentas tomadas o pausas desactualizadas.

Usage

```
resetLocksX()
```

Value

Invisiblemente, la cantidad de cuentas afectadas.

Examples

```
## Not run:
resetLocksX()

## End(Not run)
```

setProxyX	<i>Asignar, cambiar o quitar el proxy de cuentas del pool</i>
-----------	---

Description

Cambia el proxy de cuentas ya cargadas sin tocar su sesion (no hace falta volver a pegar las cookies).
Conviene que cada cuenta salga siempre por la misma IP: usa proxies "sticky" (IP fija), no rotativos.

Usage

```
setProxyX(accounts, proxy, validate = TRUE)
```

Arguments

accounts	Nombres de las cuentas.
proxy	Direccion del proxy: "http://usuario:clave@host:puerto", "usuario:clave@host:puerto" o "socks5://host:puerto". Chromium no admite SOCKS5 con usuario y clave: para proxies autenticados usa la direccion HTTP del proveedor. NULL (o "") quita el proxy y la cuenta pasa a usar el global (<code>options(TweetScraperR.proxy = ...)</code>) o tu conexion directa.
validate	Logico. Si TRUE (por defecto) valida cada cuenta a traves del proxy nuevo con checkAccountsX() , lo que confirma que el proxy anda.

Value

Invisiblemente, el resultado de `checkAccountsX()` si `validate` es `TRUE`, o los nombres de las cuentas si no.

Examples

```
## Not run:
setProxyX("cuenta2", "http://usuario:clave@gate.proveedor.com:10000")
setProxyX("cuenta2", NULL) # quitarlo

## End(Not run)
```

 setupAccountsX

Cargar y validar cuentas de X desde la consola

Description

Asistente interactivo para armar el pool de cuentas sin escribir código ni dejar las cookies en el historial de R: pide el nombre de cada cuenta, su `auth_token` y su `ct0` (la entrada queda oculta) y un proxy opcional, la importa con `importSessionX()` y la valida en el momento con `checkAccountsX()`. Se termina dejando el nombre vacío.

Usage

```
setupAccountsX(validate = TRUE)
```

Arguments

`validate` Logico. Si `TRUE` (por defecto) valida cada cuenta apenas se carga.

Details

En el campo `auth_token` también se puede pegar el header completo de cookies ("`auth_token=...; ct0=...`") o el JSON de una extensión como Cookie-Editor; en ese caso no se pide `ct0`.

Para obtener las cookies de varias cuentas, abra cada una en un perfil de navegador distinto (o en una ventana de incógnito aparte) y cierre la ventana **sin cerrar sesión**: cerrar sesión invalida el `auth_token`.

Value

Invisiblemente, el estado del pool (`accountsX()`).

Examples

```
## Not run:
setupAccountsX()

## End(Not run)
```

Index

accountsX, 3
accountsX(), 4, 26, 35
addAccountsX, 3

checkAccountsX, 4
checkAccountsX(), 3, 34, 35
checkPlaywrightEngine, 5

extractTweetsData, 5

getListMembersAPI, 7
getListTweetsAPI, 8
getTweetsDataAPI, 8
getTweetsHashtags, 9
getTweetsImagesAnalysis, 10
getTweetsLiveAPI, 12
getTweetsRepliesAPI, 13
getTweetsRetweetsAPI, 14
getTweetsSearchAPI, 15
getTweetsSearchAPI(), 12, 13
getTweetsSentiments, 16
getTweetsTimelinesAPI, 17
getUserFollowersAPI, 18
getUserFollowingAPI, 19
getUserMediaAPI, 20
getUsersAboutAPI, 20
getUsersDataAPI, 21
getUsersDataAPI(), 7
getUsersSearchAPI, 22
getUserTweetsAPI, 23
getUserTweetsAPI(), 8, 14, 15, 18

HTMLImgReport, 24

importSessionX, 25
importSessionX(), 3, 7, 8, 13–15, 17–23, 28, 32, 35
installAgentSkill, 26
installPlaywrightEngine, 27

loginX, 28

plotEmojis, 29
plotEmojisPNG, 30
plotTime, 31
plotWords, 31

reactivateAccountX, 32
removeAccountX, 33
resetLocksX, 34

setProxyX, 34
setupAccountsX, 35