Avatar R Tutorial

This R interface communicates with the avatarization engine. For more information about the concepts and avatarization, checkout our main docs.

System prerequisites

Make sure your system has `libcurl` properly installed.

On a shell on your Ubuntu/Debian system:

```
apt-get update && apt-get install libcurl4-openssl-dev libssl-dev
```

Installation

Some packages must be installed before `avatar`.

R packages dependencies

```
# requires libcurl to work (see above)
install.packages(c("tibble", "magrittr", "readr", "httr", "dplyr"))
```

Make sure you're using a recent version of `readr`:

```
# Make sure you're using a recent version, e.g. 2.1.0
packageVersion("readr")
```

Install the avatar package

Install the package by pointing to the `.tar.gz` source (replace path below).

```
install.packages("~/Downloads/avatar_0.x.x.tar.gz", repos = NULL, type = "source")
```

Loading the library

Once installed, load the library:

```
library(avatar)
```

Getting help

You can list avatar's vignettes:

```
browseVignettes(package = "avatar")
```

You can also get documentation about the main functions:

```
help(set_server)
```

Preliminary steps

First you need to load necessary packages and set the server.

```
library(avatar)

# Packages below are not necessary to run avatar but must be installed to run this tutorial
library(knitr)
library(dplyr)
library(ade4)
library(ggplot2)
library(plotly)
library(gridExtra)
library(corrplot)
library(readr)
```

Then configure the API endpoint:

```
# Authorized self signed tls certificate
httr::set_config(httr::config(ssl_verifypeer = 0L))

# Set the server and check connection
# Note: replace localhost with the server's URL
set_server("http://localhost:8000")
```

Quickstart

This is all you need to run and evaluate and avatarization:

```
# Replace with your server address
set_server("http://localhost:8000")

# Authenticate
authenticate("username", "strong_password")

# Upload the dataset
dataset_id <- upload_dataset(iris)
# Start the avatarization job
job <- start_job(dataset_id, list(k = 20))
# Retrieve the avatars
avatars <- get_avatars(job$id)

# Retrieve the privacy metrics
result <- get_job_result(job$id)
print(result$privacy_metrics$hidden_rate)
```

Avatarization Step by Step

Import Dataset

```
# Import data
my_data <- iris

# Display dataframe
kable(head(my_data))
```

```r
# You can control that R understood the data types for each column during importation:
sapply(my_data, class)
```

If R understood the wrong data type, you need to recast it before avatarization or it will change the results. Here for instance, the variable `Species` could have been coded as `(1, 2, 3)`. In this case R would have casted `Species` as numeric while it is a factor.

Set parameters

Here's the list of parameters you can use for avatarization. The description for each parameter is available in our main docs.

- `k` (required)
- `column_weights`: default=1 for each variable
- `ncp`: default=5.
- `imputation`: imputation parameters.
  - `k` (default=5): number of neighbors for the knn imputation.
  - `method` (default=`knn`): method used for the imputation.
  - `training_fraction` (default=1): the fraction of the dataset used to train the knn imputer.
- `seed`: default=NULL.

```r
# Upload your dataset
dataset_id <- upload_dataset(iris)

# Set the parameters you want to use for avatarization
parameters <- list(k = 20, column_weights = list(Sepal.Width = 3), ncp = 3, seed = 42, imputation = lis

# You don't need to specify every parameter. Most of them have default values (see above).
parameters <- list(k = 120)
```

Run avatarization

```r
# This launches the avatarization and returns immediately
job <- start_job(dataset_id, parameters)

# You can retrieve the avatarized dataset.
# This call will block until the job is done or a timeout is expired.
# You can call this function as often as you want.
avatars <- get_avatars(job$id)

# Once the avatarization is finished, you will be able to manipulate the avatarized dataset:
# Note that the order of the lines have been shuffled, which means that the link
# between original and avatar individuals cannot be made.
kable(head(avatars))

# You can also retrieve the result metadata, including privacy metrics (see below)
# For the full response, checkout help(get_job_result)
result <- avatar::get_job_result(job$id)
```

You can follow the same process to retrieve the avatars with a higher `k`:

```r
parameters_higher_k <- list(k = 120)
job2 <- start_job(dataset_id, parameters_higher_k)
large_k_avatars <- get_avatars(job2$id)
kable(head(large_k_avatars))
```

Evaluate privacy and utility

You can retrieve the privacy and utility metrics from the result object (see our main docs for details about each metric):

```
result$privacy_metrics$hidden_rate
result$privacy_metrics$local_cloaking
```

Retrieve unshuffled avatars

Caution - Beware before using the code below

The unshuffled dataset is not protected as the linkage between the unshuffled avatars dataset and the original data is obvious.

This dataset contains sensitive data. You will need to shuffle it in order to make it safe.

```
# Retrieve your job result
# For the full response, checkout help(get_job_result)
result <- avatar::get_job_result(job$id)
# Download the dataset
columns <- result$sensitive_unshuffled_avatars_datasets$columns
download_url <- result$sensitive_unshuffled_avatars_datasets$download_url
sensitive_unshuffled_avatars <- get_dataset(download_url, columns)
```

Reset password

This following section only applies to servers that have configured email authentication.

To reset your password, you first need to call `forgotten_password`.

```
avatar::forgotten_password("yourmail@mail.com")
```

You will receive an email with a token.

Copy that token, and call the next command, `reset_password`:

```
avatar::reset_password("yourmail@mail.com", "new_password", "new_password", "token-received-by-mail")
```