Package: connectViz (via r-universe)

May 24, 2026

Title Visualize Your 'RStudio Connect' Server Usage Data

Version 0.0.0.9146

Description A collection of helper functions and 'htmlwidgets' to help admins or user better understand how 'RStudio Connect' is used in their organization. The package provides plug and play visualizations that can be customized depending on needs.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), shinydashboard

Config/testthat/edition 3

URL https://github.com/RinteRface/connectViz

BugReports https://github.com/RinteRface/connectViz/issues

Imports magrittr, lubridate, dplyr (>= 1.0.0), data.table, purrr, tibble, apexcharter (>= 0.3.0.9100), scales, toastui, connectapi (>= 0.1.0.9031), shiny, echarts4r, tidyr, rlang (>= 1.0.0), visNetwork, htmlwidgets

Remotes rstudio/connectapi

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://cynkra.r-universe.dev

Date/Publication 2024-01-22 18:39:15 UTC

RemoteUrl https://github.com/RinteRface/connectviz

RemoteRef HEAD

RemoteSha 7fd35b9f17b13a814d8f735df681e4d8c61dd90e

Contents

`create_app_daily_session_chart`
Generic calendar chart generator

Description

Generic calendar chart generator

Usage

```
create_app_daily_session_chart(
  calendar_data,
  title = "Cumulated daily usage",
  subtitle = "time units: minutes",
  start_date = NULL,
  end_date = NULL,
  callback = NULL
)
```

Arguments

calendar_data	Calendar chart data.
title	Chart title.
subtitle	Chart subtitle.
start_date	Default to minimum calendar_data date. Could also be an input value with Shiny.
end_date	Default to maximum calendar_data date. Could also be an input value with Shiny.
callback	JS function to pass to onRender. This is useful to access the widget API on the R side at render time and add events on the fly.

Value

An echarts4r line chart

---

create_app_daily_usage_chart

Daily app usage chart

Description

Leverages echarts4r. See https://echarts4r.john-coene.com/articles/chart_types.html#calendar-1. .

Usage

```
create_app_daily_usage_chart(app_usage, ...)
```

Arguments

app_usage	Returned by get_app_daily_usage.
...	Pass down any param to create_calendar_chart.

Value

A calendar chart displaying daily app usage.

create_app_ranking *Process app data for ranking table*

Description

See create_app_ranking_table.

Usage

```
create_app_ranking(content, users, apps, start_date = NULL, end_date = NULL)
```

Arguments

content	Get from get_content. Can be reactive.
users	Get from get_users. Can be reactive.
apps	Get from get_usage_shiny. Can be reactive.
start_date	Default to minimum calendar_data date. Could also be an input value with Shiny.
end_date	Default to maximum calendar_data date. Could also be an input value with Shiny.

Value

A list containing: `[[1]]` merged data between app usage and users data. `[[2]]`: data to be digested by create_app_ranking_table.

create_app_ranking_table *Create app ranking table*

Description

Leverages the toastui htmlwidget. See https://dreamrs.github.io/toastui/articles/extras/grid.html.

Usage

```
create_app_ranking_table(ranking, pagination = 10, height = NULL)
```

Arguments

ranking	Ranked data from create_app_ranking.
pagination	Control number of lines to display per page. If NULL, all data are displayed.
height	Table height. CSS units.

Value

An htmlwidget table containing app ranking, owner sorted by usage.

`create_apps_consumer_ranking`

Create Shiny apps consumer ranking

Description

Sort consumers by number of views.

Usage

```
create_apps_consumer_ranking(apps, users)
```

Arguments

`apps`	Get from get_usage_shiny. Can be reactive or not.
`users`	Get from get_users. Can be reactive or not.

Value

A 3 columns tibble with apps consumer sorted by number of view. The role columns allows further analysis.

`create_apps_consumer_ranking_chart`

Create apps consumer ranking bar chart

Description

Create apps consumer ranking bar chart

Usage

```
create_apps_consumer_ranking_chart(ranking, threshold)
```

Arguments

`ranking`	Data obtained from create_apps_consumer_ranking.
`threshold`	Minimum number of app threshold. You'll need a numericInput wrapped by reactive.

Value

A bar chart with consumer sorted by descending number of total views.

`create_calendar_chart` *Generic calendar chart generator*

Description

Generic calendar chart generator

Usage

```
create_calendar_chart(
  calendar_data,
  title,
  subtitle = NULL,
  start_date = NULL,
  end_date = NULL,
  callback = NULL
)
```

Arguments

`calendar_data`	Calendar chart data.
`title`	Chart title.
`subtitle`	Chart subtitle.
`start_date`	Default to minimum calendar_data date. Could also be an input value with Shiny.
`end_date`	Default to maximum calendar_data date. Could also be an input value with Shiny.
`callback`	JS function to pass to onRender. This is useful to access the widget API on the R side at render time and add events on the fly.

Value

An echarts4r calendar chart

`create_cumulated_duration_per_user`
Create cumulated app duration/user

Description

Bar chart

Usage

```
create_cumulated_duration_per_user(
  apps_usage,
  start_date = NULL,
  end_date = NULL,
  selected_app
)
```

Arguments

apps_usage	First element returned by create_app_ranking. Can be reactive.
start_date	Default to minimum calendar_data date. Could also be an input value with Shiny.
end_date	Default to maximum calendar_data date. Could also be an input value with Shiny.
selected_app	Selected app name (string). You'll need a selectInput for instance wrapped by reactive.

Value

An echarts4r barchart.

---

create_cumulated_hits_per_user

Create cumulated app hits/user

Description

Bar chart

Usage

```
create_cumulated_hits_per_user(
  apps_usage,
  start_date = NULL,
  end_date = NULL,
  selected_app
)
```

Arguments

apps_usage	First element returned by create_app_ranking. Can be reactive.
start_date	Default to minimum calendar_data date. Could also be an input value with Shiny.
end_date	Default to maximum calendar_data date. Could also be an input value with Shiny.

selected_app	Selected app name (string). You'll need a selectInput for instance wrapped by reactive.

Value

An echarts4r barchart.

`create_dev_project_overview` *Create developer project network overview*

Description

Leverages visNetwork.

Usage

```
create_dev_project_overview(client, apps_usage, selected_dev)
```

Arguments

client	RSC client. See create_rsc_client.
apps_usage	First element returned by create_app_ranking. Can be reactive.
selected_dev	Developer to select. You'll need a selectInput wrapped by reactive.

Value

A visNetwork htmlwidget with developer projects.

`create_dev_ranking` *Create a developer ranking*

Description

Rank developers by number of published apps

Usage

```
create_dev_ranking(users, content)
```

Arguments

users	Get from get_users.
content	Get from get_content.

Value

A tibble with developer ranked by decreasing number of apps.

create_dev_ranking_chart *Create developers ranking bar chart*

Description

Devs are ranked by number of developed apps. See create_dev_ranking.

Usage

```
create_dev_ranking_chart(ranking, threshold)
```

Arguments

ranking	Obtained after calling create_dev_ranking. Can be reactive.
threshold	Minimum number of app threshold. You'll need a numericInput wrapped by reactive.

Value

An echarts4r bar chart.

create_pie_chart *Create standard pie chart*

Description

Pie chart with percentage data

Usage

```
create_pie_chart(data, x_axis)
```

Arguments

data	Get from sort_users_by_role.
x_axis	x variable.

Value

A pie chart.

create_rsc_client *Create a connection to RStudio Connect server*

Description

Leverages connectapi toolkit. Expect to have `Sys.getenv("CONNECT_SERVER")` and `Sys.getenv("CONNECT_API_KEY")` properly setup.

Usage

```
create_rsc_client()
```

create_user_daily_consumption_chart *Daily app consumption for selected user*

Description

Daily app consumption for selected user

Usage

```
create_user_daily_consumption_chart(usage)
```

Arguments

usage	Get from get_user_daily_consumption.

Value

An echarts4r calendar chart

generate_table *Generate htmlWidgets table*

Description

Leverage toastui API.

Usage

```
generate_table(logs, sparkline = FALSE, pagination = 10, height = NULL)
```

Arguments

logs	Obtained with get_rsc_apps_usage.
sparkline	Whether to draw a sparkline.
pagination	Control number of lines to display per page. If NULL, all data are displayed.
height	Table height. CSS units.

Value

An htmlwidget table.

---

get_app_daily_usage *Daily app usage*

Description

Used by create_app_daily_usage_chart .

Usage

```
get_app_daily_usage(apps_usage, selected_app)
```

Arguments

apps_usage	Second element returned by create_app_ranking. Can also be reactive.
selected_app	Selected app name (string). You'll need a selectInput for instance. wrapped by reactive.

Value

Calendar data for daily app usage.

---

get_max_rsc_apps_usage *Get most used app*

Description

Get most used app

Usage

```
get_max_rsc_apps_usage(logs)
```

Arguments

logs	Obtained after calling get_rsc_apps_usage.

Value

A numeric value.

---

get_rsc_app_dates *Extract app usage dates*

Description

Extract app usage dates

Usage

```
get_rsc_app_dates(app, logs)
```

Arguments

app	app_name to filter.
logs	Given by RSC database.

Value

A tibble containing dates of usage for the given app as well as the correponding session duration.

---

get_rsc_apps_usage *Get RSC apps usage*

Description

Called on apps_usage data

Usage

```
get_rsc_apps_usage(logs)
```

Arguments

logs	Given by RSC database.

Value

A 3 columns tibble with app name, usage and date of usage (nested tibble).

get_rsc_developer_apps_count

Get app count for each developer

Description

Get app count for each developer

Usage

```
get_rsc_developer_apps_count(developer, logs)
```

Arguments

developer	Developer unique id.
logs	All data logs.

Value

A number with all apps

get_rsc_developer_apps_list

Get app list for each developer + usage

Description

Get app list for each developer + usage

Usage

```
get_rsc_developer_apps_list(developer, logs)
```

Arguments

developer	Unique user id. Typically app_developer_id key.
logs	RSC logs.

Value

Tibble containing app name + count

`get_user_daily_consumption` *Get daily shiny app usage for a given user*

Description

Get daily shiny app usage for a given user

Usage

```
get_user_daily_consumption(content, users, apps, selected_user)
```

Arguments

`content`	Get from get_content. Can be reactive.
`users`	Get from get_users. Can be reactive.
`apps`	Get from get_usage_shiny. Can be reactive.
`selected_user`	User to select. You'll need a selectInput wrapped by reactive.

Value

A list. `[[1]]` contains the row events filtered for the given user. `[[2]]` is 2 columns tibble containing daily app consumption for given user (grouped by dates).

`merge_rsc_data` *Merge RStudio Connect data together*

Description

See create_app_ranking.

Usage

```
merge_rsc_data(content, users, apps)
```

Arguments

`content`	Get from get_content. Can be reactive.
`users`	Get from get_users. Can be reactive.
`apps`	Get from get_usage_shiny. Can be reactive.

Value

A list of 3 tibbles. `[[1]]`: light RSC content data (less columns); `[[2]]`: light RSC users data (less columns); `[[3]]`: merged apps_usage data.

process_rsc_content *Process RSC content*

Description

Select relevant information for RSC content data. See merge_rsc_data.

Usage

```
process_rsc_content(content)
```

Arguments

content	Get from get_content.

Value

A 3 columns tibble with content guid, app_name and owner guid. Owner guid is the same returned in process_rsc_user.

process_rsc_user *Process RSC user*

Description

Select relevant information for RSC users data. See merge_rsc_data.

Usage

```
process_rsc_user(users)
```

Arguments

users	Get from get_users.

Value

A 2 columns tibble with user guid and username.

`sort_content_by_access`

Sort RStudio Connect content by access type

Description

Sort RStudio Connect content by access type

Usage

```
sort_content_by_access(content, start_date = NULL, end_date = NULL)
```

Arguments

`content`	Get from get_content.
`start_date`	Date filter.
`end_date`	Date filter.

Value

A tibble with content grouped by access type.

`sort_content_by_appmode`

Sort RStudio Connect content by app mode

Description

Sort RStudio Connect content by app mode

Usage

```
sort_content_by_appmode(content, start_date = NULL, end_date = NULL)
```

Arguments

`content`	Get from get_content.
`start_date`	Date filter.
`end_date`	Date filter.

Value

A tibble with content grouped by app mode.

sort_content_by_pyversion

Sort RStudio Connect content by python version

Description

Sort RStudio Connect content by python version

Usage

```
sort_content_by_pyversion(content)
```

Arguments

content	Get from get_content.

Value

A tibble with content grouped by python version.

sort_content_by_rversion

Sort RStudio Connect content by R version

Description

Sort RStudio Connect content by R version

Usage

```
sort_content_by_rversion(content, start_date = NULL, end_date = NULL)
```

Arguments

content	Get from get_content.
start_date	Date filter.
end_date	Date filter.

Value

A tibble with content grouped by R version.

`sort_users_by_role` *Sort RStudio Connect users by role*

Description

Users are grouped by user_role to check the server role repartition. Using start_date and end_date allows to filter data given a specific time range.

Usage

```
sort_users_by_role(users, start_date = NULL, end_date = NULL)
```

Arguments

`users`	Get from get_users.
`start_date`	Date filter.
`end_date`	Date filter.

Value

A tibble with user grouped by role.

Index