

Package: ojoutils (via r-universe)

October 10, 2026

Title A Collection of Nifty Functions and Objects for OJO Analysts

Version 0.3.0

Description We find ourselves repeating the same simple tasks or running a series of R commands over and over again. No more!

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports arrow, cli, dbplyr, dplyr, fs, gargle, gert, gh, glue, googleCloudStorageR, janitor, lubridate, nanoarrow, processx, quarto, readr, renv, rlang, stringr, targets, tidyr, usethis, withr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tibble, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://openjusticeok.github.io/ojoutils/>

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65.0, xz

Depends R (>= 4.2)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake git libglpk-dev make libgit2-dev libicu-dev libuv1-dev libxml2-dev libzstd-dev libssl-dev libx11-dev xz-utils libclang-dev

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

Date/Publication 2026-05-06 23:37:04 UTC

RemoteUrl <https://github.com/openjusticeok/ojoutils>

RemoteRef HEAD

RemoteSha 4195f568074eb263666f021e70f1898de0c8c054

Contents

count_interval	2
describe_change	3
dir_empty	5
gcs_auth_bucket	5
gcs_list_objects	6
gcs_read_csv	7
gcs_write_csv	8
limit	10
ojo_counties	10
ojo_create_project	11
ojo_parse_county	12
ojo_use_template	12
tar_gcs_csv	13
Index	16

count_interval	<i>Count intervals over time periods</i>
----------------	--

Description

Counts the number of active intervals for each time period (day, hour, etc.) given start and end dates. Useful for occupancy or population counts over time.

Usage

```
count_interval(  
  data,  
  start,  
  end,  
  period = "day",  
  date_name = "date",  
  count_name = "n",  
  .by = character(),  
  .fill = list(start = NULL, end = NULL),  
  .inclusive = c(TRUE, TRUE)  
)
```

Arguments

data	A data frame or Arrow table containing the interval data.
start	Character string. Name of the column containing interval start dates.
end	Character string. Name of the column containing interval end dates.
period	Character string. Time period for counting (e.g., "day", "hour", "week", "month", "quarter", "year"). Defaults to "day".

date_name	Character string. Name for the output date column. Defaults to "date".
count_name	Character string. Name for the output count column. Defaults to "n".
.by	Character vector. Column names to group by. Defaults to empty (no grouping).
.fill	Named list with "start" and "end" elements. Values to use when filling NA start/end dates. If NULL (default), uses min/max of data.
.inclusive	Logical vector of length 2. Whether start and end boundaries are inclusive. Defaults to c(TRUE, TRUE).

Value

A tibble with columns for date, count, and any grouping variables.

Examples

```
## Not run:
# Basic usage
df <- data.frame(
  start = as.Date(c("2024-01-01", "2024-01-05")),
  end = as.Date(c("2024-01-03", "2024-01-06"))
)
count_interval(df, start = "start", end = "end", period = "day")

# With grouping
df <- data.frame(
  start = as.Date(c("2024-01-01", "2024-01-02")),
  end = as.Date(c("2024-01-03", "2024-01-04")),
  ward = c("A", "B")
)
count_interval(df, start = "start", end = "end", period = "day", .by = "ward")

## End(Not run)
```

describe_change	<i>Describe Change</i>
-----------------	------------------------

Description

Generate descriptive statements about changes between two values.

Usage

```
describe_change(
  before,
  after,
  input_unit,
  output_unit,
  template = NULL,
```

```

direction_phrases = c(increase = "increased by", decrease = "decreased by", none =
  "remained unchanged"),
include_values = FALSE
)

```

Arguments

before	The initial numeric value to compare. For some unit combinations before must be non-zero to avoid divide-by-zero errors.
after	The numeric value to compare to the initial value.
input_unit	One of "number", "percent", or "ratio". Determines how the before and after values are treated.
output_unit	One of "number", "percent", "times", or "points". "points" may only be used when input_unit is "percent" or "ratio".
template	A {glue} template for the string returned when there is a change. Defaults are provided based on output_unit. Possible template variables are: direction, change, and unit.
direction_phrases	A named vector with three items: "increase", "decrease", and "none", used to customize either the default or custom template.
include_values	A logical value indicating whether to include before and after in the change description string.

Value

A string describing the change between two values, optionally including the values themselves.

Examples

```

# Basic usage with defaults
describe_change(
  before = 1,
  after = 0.5,
  input_unit = "ratio",
  output_unit = "percent",
)
#> decreased by 50 percent

# Using different phrasing for changes
describe_change(
  before = 33,
  after = 66,
  input_unit = "percent",
  output_unit = "percent",
  direction_phrases = c(
    increase = "rose by",
    decrease = "fell by",
    none = "stagnated"
  )
)

```

```

)
#> rose by 100 percent

# Customizing the template
describe_change(
  before = 10,
  after = 12,
  input_unit = "number",
  output_unit = "number",
  template = "{direction} {change} people"
)
#> increased by 2 people

```

dir_empty

Directory Empty?

Description

Takes a path and determines whether it is empty.

Usage

```
dir_empty(path)
```

Arguments

path A relative or absolute path to the directory to test.

Value

TRUE if the directory is empty, otherwise FALSE. If the directory doesn't exist, or the string supplied to path isn't recognized as a valid path, then the function returns an error.

gcs_auth_bucket

Authenticate with Google Cloud Storage

Description

Authenticates with Google Cloud Storage (GCS) using the gargle package for OAuth2 token fetching, then sets the specified bucket as the global bucket for subsequent GCS operations.

Usage

```
gcs_auth_bucket(bucket)
```

Arguments

bucket	Character string. The name of the GCS bucket to set as the global bucket for subsequent operations.
--------	---

Details

This function uses `gargle::token_fetch()` to obtain an OAuth2 token with cloud-platform scope, then authenticates with `googleCloudStorageR`. It sets the global bucket so that subsequent GCS operations don't need to specify the bucket parameter repeatedly.

The function returns the bucket name invisibly, making it suitable for use in pipelines where you want to authenticate and continue processing.

Value

The bucket name (invisibly).

See Also

[googleCloudStorageR::gcs_auth\(\)](#) for more authentication options

Examples

```
## Not run:
# Authenticate and set a specific bucket as global
gcs_auth_bucket("my-project-data")

# Can be used in a pipeline
"my-project-data" |> gcs_auth_bucket()

## End(Not run)
```

<code>gcs_list_objects</code>	<i>List objects in a Google Cloud Storage bucket</i>
-------------------------------	--

Description

Lists all object names in a GCS bucket, optionally filtered by a prefix. Returns a character vector of object names.

Usage

```
gcs_list_objects(bucket, prefix = NULL)
```

Arguments

bucket	Character string. The name of the GCS bucket to list objects from.
prefix	Character string (optional). A prefix to filter objects. Only objects whose names begin with this prefix will be returned.

Details

This function wraps `googleCloudStorageR::gcs_list_objects()` and extracts just the object names as a character vector using `dplyr::pull()`.

The `prefix` parameter can be used to filter results to objects within a specific "folder" or matching a specific pattern. Note that GCS uses a flat namespace, so prefixes simulate directory structures.

Value

A character vector of object names in the bucket.

See Also

[googleCloudStorageR::gcs_list_objects\(\)](#) for full object details

Examples

```
## Not run:
# List all objects in a bucket
all_objects <- gcs_list_objects("my-project-data")

# List objects in a specific "folder"
csv_files <- gcs_list_objects("my-project-data", prefix = "raw/")

# List objects with a specific prefix
sales_files <- gcs_list_objects("my-project-data", prefix = "sales_2024")

## End(Not run)
```

`gcs_read_csv`

Read a CSV file from Google Cloud Storage

Description

Reads a CSV file from Google Cloud Storage into a data frame using the arrow package for efficient reading. Optionally cleans column names using `janitor::clean_names()`.

Usage

```
gcs_read_csv(bucket, object, clean_names = TRUE)
```

Arguments

<code>bucket</code>	Character string. The name of the GCS bucket containing the file.
<code>object</code>	Character string. The path to the CSV file within the bucket.
<code>clean_names</code>	Logical. If TRUE (default), column names are cleaned using janitor::clean_names() . If FALSE, original column names are preserved.

Details

This function uses arrow's CSV reader which is optimized for performance and can handle large files efficiently. The GCS path is constructed using glue for safe string interpolation.

By default, column names are cleaned using `janitor::clean_names()` to convert them to snake_case and remove special characters. Set `clean_names = FALSE` to preserve original column names.

Value

A data frame (tibble) containing the CSV data.

See Also

`arrow::read_csv_arrow()` for reading options, `janitor::clean_names()` for name cleaning details

Examples

```
## Not run:
# Read a CSV and clean column names
data <- gcs_read_csv("my-project-data", "raw/customers.csv")

# Read a CSV preserving original column names
data <- gcs_read_csv("my-project-data", "raw/customers.csv", clean_names = FALSE)

# Use with gcs_auth_bucket for authenticated reading
gcs_auth_bucket("my-project-data")
data <- gcs_read_csv("my-project-data", "processed/sales_2024.csv")

## End(Not run)
```

`gcs_write_csv`*Write a CSV file to Google Cloud Storage*

Description

Writes a data frame to a CSV file in Google Cloud Storage using the arrow package. Optionally returns metadata about the uploaded object.

Usage

```
gcs_write_csv(data, bucket, object, meta = FALSE)
```

Arguments

data	A data frame to write to GCS.
bucket	Character string. The name of the GCS bucket to write to.
object	Character string. The destination path for the CSV file within the bucket (e.g., "folder/subfolder/filename.csv").
meta	Logical. If TRUE, returns a list with object metadata. If FALSE (default), returns the GCS path invisibly.

Details

This function uses arrow's CSV writer for efficient writing of data frames to GCS. The file is written directly to the specified GCS path without requiring temporary local storage.

When meta = TRUE, the function retrieves and returns metadata about the uploaded object including:

- The GCS path
- MD5 hash for integrity verification
- Generation number for versioning
- File size
- Last updated timestamp

Value

If meta = TRUE, a list containing:

- path: The full GCS path to the object
- md5_hash: The MD5 hash of the uploaded object
- generation: The object's generation number
- size: The object size in bytes
- updated: The last update timestamp

If meta = FALSE, the GCS path (invisibly).

See Also

[arrow::write_csv_arrow\(\)](#) for write options, [googleCloudStorageR::gcs_get_object\(\)](#) for metadata retrieval

Examples

```
## Not run:
# Write a data frame to GCS
gcs_write_csv(mtcars, "my-project-data", "output/mtcars.csv")

# Write and get metadata back
meta <- gcs_write_csv(mtcars, "my-project-data", "output/mtcars.csv", meta = TRUE)
print(meta$md5_hash)
```

```
# Use with authentication
gcs_auth_bucket("my-project-data")
gcs_write_csv(my_data, "my-project-data", "processed/results.csv")

## End(Not run)
```

limit	<i>Limit</i>
-------	--------------

Description

A thin wrapper around [head](#) to sooth the pain of context switching between R and SQL

Usage

```
limit(x, ...)
```

Arguments

- x The object to limit
- ... Additional arguments to [head](#)

Details

A thin wrapper around [head](#) that allows `limit` to be used in place of `head`. This is useful because sometimes it is hard to context switch between R and SQL. It can be used on lazy data frames since this package imports `dbplyr`.

Value

The limited object

ojo_counties	<i>Oklahoma Counties</i>
--------------	--------------------------

Description

Oklahoma Counties

Usage

```
ojo_counties
```

ojo_create_project	<i>Create Project</i>
--------------------	-----------------------

Description

Creates a new R project with a standard directory structure

Usage

```
ojo_create_project(  
  name = NULL,  
  description = NULL,  
  dir = ".",  
  private = TRUE,  
  packages = NULL  
)
```

Arguments

name	Character string. Name of the project/repo. If NULL, will attempt interactive prompt.
description	Character string. Description of the project for GitHub.
dir	Character string. Directory where project should be created. Defaults to current working directory.
private	Logical. Whether the GitHub repository should be private. Defaults to TRUE.
packages	Character vector. Additional packages to install in the project's renv environment. Currently not implemented.

Value

Invisible path to the created project directory.

Getting Started

To learn more about creating projects, see the vignette: `vignette("project-creation", package = "ojotools")`

Examples

```
## Not run:  
ojo_create_project("my-analysis", "Analysis of court data")  
  
## End(Not run)
```

ojo_parse_county	<i>Parse Oklahoma Counties</i>
------------------	--------------------------------

Description

Parse Oklahoma Counties

Usage

```
ojo_parse_county(
  county,
  ...,
  case = "lower",
  squish = NULL,
  suffix = NULL,
  counties = ojo_counties,
  .silent = FALSE
)
```

Arguments

county	A string or character vector that represents an Oklahoma county
...	Placeholder for future arguments
case	The case to format the county string to. One of "lower", "upper", or "title".
squish	A boolean indicator of whether to remove whitespace
suffix	Specification of a suffix to remove from each item in county. For example, "County, OK".
counties	A vector of valid county names to match on.
.silent	A currently unused argument.

ojo_use_template	<i>Function to use the OKPolicy quarto website template</i>
------------------	---

Description

Wrapper for quarto use template command. Creates a new project directory and installs a Quarto template. The project name is derived from the last component of the path and must be in kebab-case format.

Usage

```
ojo_use_template(
  path = NULL,
  template = c("website", "report"),
  .interactive = rlang::is_interactive()
)
```

Arguments

path	Character. The path to the project directory. Can be absolute or relative to the current working directory. The last component of the path will be used as the project name and must be in kebab-case (lowercase letters, numbers, and hyphens only). If NULL (default) and in interactive mode, prompts for project name and location.
template	Character. The type of project template to use. Must be one of: • "website" (default) - Multi-page website template • "report" - Single-page report template
.interactive	Logical. Whether to prompt for user confirmation in interactive mode. Defaults to <code>rlang::is_interactive()</code> .

Value

Invisibly returns the result of the quarto command execution.

Examples

```
## Not run:
# Interactive mode (prompts for project name and location)
ojo_use_template()

# Create a new website project (default)
ojo_use_template("my-new-website")

# Create a report project
ojo_use_template("my-new-report", template = "report")

# Create a project in a specific directory
ojo_use_template("~/Documents/Reports/my-new-website")

# Create a project with interactive turned off
ojo_use_template("my-new-website", .interactive = FALSE)

## End(Not run)
```

tar_gcs_csv

*Create a targets pipeline target for writing CSVs to GCS***Description**

Creates a targets pipeline target that writes a data frame to Google Cloud Storage as a CSV file. This is a convenience wrapper around `targets::tar_target()` specifically designed for GCS CSV outputs.

Usage

```
tar_gcs_csv(name, data, bucket, object, ...)
```

Arguments

name	Symbol. The name of the target (unquoted).
data	Expression. The data frame to write to GCS. Can reference upstream targets or other R objects.
bucket	Character string. The name of the GCS bucket to write to.
object	Character string. The destination path for the CSV file within the bucket.
...	Additional arguments passed to <code>targets::tar_target()</code> .

Details

This function creates a targets target that:

1. Authenticates with GCS using `gcs_auth_bucket()`
2. Writes the data to GCS using `gcs_write_csv()` with metadata enabled
3. Returns the metadata as the target value

The target is created with `format = "qs"` for efficient serialization of the metadata list.

The data parameter is captured as an expression and evaluated at build time, allowing you to reference upstream targets or other R objects.

Value

A targets target object suitable for use in a `_targets.R` file.

See Also

`targets::tar_target()` for general target options, `gcs_write_csv()` for the underlying write operation

Examples

```
## Not run:
# In your _targets.R file:
library(targets)
library(ojoutils)

tar_plan(
  # Process some data
  tar_target(raw_data, read_csv("input.csv")),
  tar_target(clean_data, clean_my_data(raw_data)),

  # Write to GCS as a target
  tar_gcs_csv(
    gcs_output,
    clean_data,
    bucket = "my-project-data",
    object = "processed/clean_data.csv"
  )
)
```

```
# With additional tar_target options
tar_gcs_csv(
  gcs_output,
  processed_data,
  bucket = "my-project-data",
  object = "output/results.csv",
  priority = 1
)

## End(Not run)
```

Index

`arrow::read_csv_arrow()`, [8](#)
`arrow::write_csv_arrow()`, [9](#)

`count_interval`, [2](#)

`describe_change`, [3](#)
`dir_empty`, [5](#)

`gcs_auth_bucket`, [5](#)
`gcs_list_objects`, [6](#)
`gcs_read_csv`, [7](#)
`gcs_write_csv`, [8](#)
`gcs_write_csv()`, [14](#)
`googleCloudStorageR::gcs_auth()`, [6](#)
`googleCloudStorageR::gcs_get_object()`,
[9](#)
`googleCloudStorageR::gcs_list_objects()`,
[7](#)

`head`, [10](#)

`janitor::clean_names()`, [7](#), [8](#)

`limit`, [10](#)

`ojo_counties`, [10](#)
`ojo_create_project`, [11](#)
`ojo_parse_county`, [12](#)
`ojo_use_template`, [12](#)

`tar_gcs_csv`, [13](#)
`targets::tar_target()`, [14](#)