Package: MARMOT (via r-universe)

November 24, 2026

Type Package

Title MARMOT

Version 1.3.2

Author Peter Leary [aut, cre]

Maintainer Peter Leary <peter.leary@uzh.ch>

Description MARMOT provides an extensible framework for high-dimensional single-cell cytometry analysis, including quality control, clustering, dimensionality reduction, differential testing, and visualisation.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.5)

Imports anndataR, BiocGenerics, BiocManager, CATALYST, chameleon, circlize, clustree, colorspace, colourpicker, ComplexHeatmap, cowplot, data.table, diffcyt, dplyr, DT, fireworks, flowCore, FlowSOM, fresh, future, future.apply, ggnewscale, ggprism, ggplot2, ggpubr, ggbeeswarm, ggrepel, ggridges, glue, grDevices, grid, gridExtra, gtools, htmltools, igraph, jsonlite, kableExtra, later, limma, MASS, Matrix, matrixStats, methods, openxlsx, pacman, pals, patchwork, plotly, purrr, quarto, RANN, RColorBrewer, RcppHNSW, readr, readxl, reticulate, rhdf5, rlang, reshape2, rstatix, S4Vectors, scales, scattermore, scico, shiny, shinyBS, shinyalert, shinycssloaders, shinydashboard, shinyjs, shinyWidgets, SingleCellExperiment, sortable, stats, tibble, tidyr, SummarizedExperiment, viridis, waiter, writexl, zip, qs2

Suggests ragg, pak, remotes, flowAI, knitr, ks, Nebulosa, PeacoQC, scGate, SCpubr, Seurat, Biobase, testthat (>= 3.0.0), UCell, withr

Config/testthat/edition 3

Remotes hypebright/fireworks, scverse/anndataR

Config/pak/sysreqs
cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git libglpk-dev make libharfbuzz-dev libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl python3 libx11-dev zlib1g-dev

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

Date/Publication 2026-07-07 09:07:48 UTC

RemoteUrl https://github.com/peterleary/marmot

RemoteRef HEAD

RemoteSha 3d65070f1d5a57a7b7c404cb8b864903c231c5ec

Contents

addMetadataToFCSFolder . . . 3
check_setup . . . 3
export_claude_skill . . . 4
h5ad_io . . . 5
install_claude_skill . . . 5
install_marmot_extras . . . 6
install_marmot_python . . . 7
launch_gui . . . 7
load_h5ad_for_shiny . . . 8
load_h5ad_to_env . . . 8
marmot . . . 9
marmot-python . . . 9
marmot-python-subprocess . . . 10
marmot_python_status . . . 10
MfastPG . . . 10
Mpacmap . . . 12
Mparc . . . 13
Mphenograph . . . 14
plotExprHeatmap . . . 15
plotFreqHeatmap . . . 17
reconstruct_sce_from_h5ad . . . 18
save_da_ds_h5ad . . . 19
save_h5ad_data . . . 19
setup_python . . . 20
shinyMarmot . . . 20

Index **22**

addMetadataToFCSFolder *addMetadataToFCSFolder*

Description

This function copies the metadata file from the package to a user-defined location. Ideally the folder where the gated FCS files live.

Usage

```
addMetadataToFCSFolder(FCS_folder = ".", name = NULL, overwrite = FALSE)
```

Arguments

FCS_folder	Path to the folder containing FCS files (default ".")
name	Name suffix for the metadata file (default NULL)
overwrite	Logical; overwrite existing file? (default FALSE)

Value

A message indicating the location of the metadata file.

Author(s)

Peter Leary

Examples

```
## Not run:
addMetadataToFCSFolder("Files/MARMOT_Metadata.xlsx")

## End(Not run)
```

---

check_setup *Check MARMOT installation status*

Description

Prints a formatted status report showing which packages and the Python environment are installed and available. Core packages (CRAN/Bioconductor) show a red X if missing; GitHub and optional packages show a yellow warning.

Usage

```
check_setup()
```

Value

Invisibly returns a data.frame with columns `package`, `type`, `status`, and `version`.

`export_claude_skill` *export_claude_skill*

Description

Bundle the MARMOT Claude Skill into a zip file suitable for uploading to the Claude desktop app or claude.ai web app (**Customize** → **Skills** → **"+" button** → **Upload a skill**). Code execution and file creation must be enabled for skill uploads to be available. For Claude Code users, use `install_claude_skill()` instead, which copies the skill directly into the skills folder.

Usage

```
export_claude_skill(path = "~/Desktop/marmot-skill.zip")
```

Arguments

`path`	Path to write the zip file to. Defaults to `~/Desktop/marmot-skill.zip`. If the file already exists it will be overwritten.

Value

Invisibly, the path to the written zip file.

Author(s)

Peter Leary

Examples

```
## Not run:
export_claude_skill()
export_claude_skill(path = "~/Downloads/marmot-skill.zip")

## End(Not run)
```

h5ad_io *h5ad I/O Functions for MARMOT*

Description

Save and load pipeline results as h5ad (AnnData HDF5) files. Uses anndataR for pure-R h5ad read/write — no Python required.

Author(s)

Peter Leary

install_claude_skill *install_claude_skill*

Description

Install the bundled MARMOT Claude Skill into the user's local Claude Code skills directory (`~/.claude/skills/marmot/`). Once installed, the skill auto-loads in Claude Code conversations whenever the user mentions MARMOT, asks about analysing flow data with it, etc. For the Claude desktop app or claude.ai web app, use `export_claude_skill()` to produce an uploadable zip instead.

Usage

```
install_claude_skill(destination = "~/.claude/skills/", overwrite = FALSE)
```

Arguments

`destination`	Path to the user's Claude skills folder. Defaults to `~/.claude/skills/`. The skill is installed as `<destination>/marmot/`.
`overwrite`	Logical; overwrite an existing `marmot/` skill folder at `destination`? Default `FALSE` (refuses to clobber).

Value

Invisibly, the path the skill was installed to.

Author(s)

Peter Leary

Examples

```
## Not run:
install_claude_skill()
install_claude_skill(overwrite = TRUE)

## End(Not run)
```

`install_marmot_extras` *Install optional MARMOT extras*

Description

Installs additional packages and sets up the Python environment on top of the base MARMOT install.

Usage

```
install_marmot_extras(include_suggests = TRUE, include_python = TRUE)
```

Arguments

`include_suggests`	If `TRUE` (the default), also install optional CRAN packages such as Seurat.
`include_python`	If `TRUE` (the default), also create the `p4r` conda environment for PARC and PaCMAP. Requires conda or mamba (install miniforge from https://github.com/conda-forge/miniforge). Fails gracefully if conda is not found.

Details

MARMOT has a two-tier install:

Tier 1 (base) `BiocManager::install("peterleary/MARMOT")` — installs FlowSOM, CATALYST, diffcyt, UMAP, TSNE, and everything R-based. Works immediately.

Tier 2 (extras) `install_marmot_extras()` — installs optional R packages (PeacoQC, flowAI, Seurat, etc.) and, when `include_python = TRUE`, creates a `p4r` conda environment for PARC clustering and PaCMAP dimensionality reduction. Requires conda or mamba (e.g. miniforge).

Value

Invisibly returns a character vector of package names that failed or were skipped.

install_marmot_python *Install the p4r conda environment for PARC/PaCMAP*

Description

Creates (or recreates) the `p4r` conda environment from the bundled `environment.yml`. Requires conda or mamba (e.g. miniforge).

Usage

```
install_marmot_python(force = FALSE)
```

Arguments

`force`	If `TRUE`, remove and recreate the environment even if it already exists and appears healthy. Default `FALSE`.

Value

Invisibly returns `TRUE` on success, `FALSE` on failure.

launch_gui *launch_gui*

Description

Launch the MARMOT desktop GUI application. On first use, downloads the pre-built binary for your platform from GitHub Releases (~5 MB, one-time). Subsequent launches use the cached binary.

Usage

```
launch_gui(version = "latest", force_download = FALSE)
```

Arguments

`version`	Character. Release tag to download (default: "latest").
`force_download`	Logical. If TRUE, re-download even if cached binary exists.

Value

Invisible NULL. The GUI application is launched as a background process.

Author(s)

Peter Leary

Examples

```
## Not run:
launch_gui()

## End(Not run)
```

`load_h5ad_for_shiny` *Load h5ad Data for Shiny App*

Description

Returns a list suitable for use as `inputDataReactive$Results` in the Shiny app.

Usage

```
load_h5ad_for_shiny(h5ad_path)
```

Arguments

`h5ad_path`	Path to the marmot_results.h5ad file

Value

A named list matching the Shiny app's expected Results structure

---

`load_h5ad_to_env` *Load h5ad Data into Environment*

Description

Reads a MARMOT h5ad file and reconstructs all pipeline R objects in the specified environment.

Usage

```
load_h5ad_to_env(h5ad_path, envir = .GlobalEnv)
```

Arguments

`h5ad_path`	Path to the marmot_results.h5ad file
`envir`	Environment to load objects into (default: .GlobalEnv)

`marmot` *marmot*

Description

The main MARMOT pipeline. Use to generate a customised MARMOT pipeline script based on the edited metadata file, and render if required.

Usage

```
marmot(metadata = NULL, name = "Title", render = FALSE)
```

Arguments

`metadata`	Path to the MARMOT metadata Excel file (.xlsx)
`name`	Name for this analysis run (used in output filenames)
`render`	Logical; if TRUE, render the pipeline report (default FALSE)

Value

A results folder in the directory of the metadata containing an HTML report, and a folder with all resulting PDFs, Excel files, and R data files.

Author(s)

Peter Leary

Examples

```
## Not run:
marmot(metadata = "~/Desktop/Flow_Data/MARMOT_metadata.xlsx", name = "Study Name", render = FALSE)

## End(Not run)
```

`marmot-python` *MARMOT Python environment helpers*

Description

Functions for discovering, using, and installing the `p4r` conda environment that provides PARC clustering and PaCMAP dimensionality reduction. No external Python manager (basilisk, pyenv) is required – only conda or mamba (e.g. miniforge).

marmot-python-subprocess

Run PaCMAP / PARC in an isolated Python subprocess

Description

On macOS arm64, running Python's `pacmap` / `parc` in the same process as R (via `reticulate`) segfaults inside ANNOY/hnswlib because R's `libomp.dylib` (linked by `data.table` and other CRAN binaries) and conda's `libomp.dylib` (linked by the `annoy` wheel) cannot coexist as two separate OpenMP runtimes in one process.

These helpers bypass reticulate entirely: the matrix is written to a temp file as raw `float64`, a subprocess is spawned against the `p4r` conda env's Python binary, and the result is read back from another temp file. Python never shares a process with R, so no libomp collision.

marmot_python_status *Check Python availability for PARC/PaCMAP (read-only)*

Description

Probes for a `p4r` conda environment and verifies that `parc` and `pacmap` can be imported. Uses `system2()` so it never binds reticulate to a Python session – safe to call from status checks.

Usage

```
marmot_python_status()
```

Value

A list with elements `available` (logical), `python_path` (character or `NA`), and `conda_path` (character or `NA`).

MfastPG *Fast PhenoGraph Clustering (MARMOT Edition)*

Description

Approximate k-NN variant of PhenoGraph clustering, based on and owing full credit to **FastPG** by Bodenheimer et al. (2020) and the original **PhenoGraph** algorithm by Levine et al. (2015). MfastPG reimplements the FastPG approach — HNSW approximate k-nearest-neighbor search followed by Jaccard-weighted graph construction and Louvain community detection — in a form that always installs from CRAN and respects `set.seed()` for full reproducibility.

Usage

```
MfastPG(data, k = 30, seed = 2, ef = 100, verbose = FALSE)
```

Arguments

`data`	matrix or data.frame; input data (rows = cells, columns = features)
`k`	integer; number of nearest neighbours (default: 30)
`seed`	integer; random seed for HNSW index construction and Louvain community detection (default: 2)
`ef`	integer; HNSW search list size — higher values give better recall at slight speed cost (default: 100, giving >99.9% recall)
`verbose`	logical; print progress messages (default: FALSE)

Details

Compared to `Mphenograph` which uses exact k-NN (RANN), MfastPG uses approximate k-NN (RcppHNSW) — typically 10-17x faster at 50-100k cells. With the default `ef = 100`, recall is >99.9% (nearly identical to exact k-NN).

The algorithm follows the same three-step pipeline as FastPG:

1. **Approximate k-NN** via HNSW (`RcppHNSW::hnsw_knn()`), the same algorithm used by FastPG, with single-threaded index construction for reproducibility
2. **Jaccard-weighted edge list** using the same formula as PhenoGraph/Rphenograph/FastPG: `w = u / (4k - 2u)`
3. **Louvain community detection** via `igraph::cluster_louvain()`

The key differences from the original FastPG are: (1) reproducibility via seed control (the original FastPG does not respect `set.seed()`), and (2) serial Louvain (igraph) instead of parallel Grappolo. Since k-NN dominates total runtime at scale (>92%), the Louvain difference is negligible in practice.

Value

A list of two elements (matching Rphenograph/Mphenograph format):

`[[1]]`	igraph graph object with Jaccard-weighted edges
`[[2]]`	igraph communities object from Louvain clustering

References

Bodenheimer, T. et al. (2020). FastPG: Fast PhenoGraph-like clustering of very large single-cell datasets. *bioRxiv*. https://github.com/sararselitsky/FastPG

Levine, J.H. et al. (2015). Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1), 184–197. doi:10.1016/j.cell.2015.05.047

See Also

`Mphenograph`, `membership`

`Mpacmap` *PaCMAP dimensionality reduction (pure R implementation)*

Description

PaCMAP dimensionality reduction (pure R implementation)

Usage

```
Mpacmap(
  data,
  n_components = 2L,
  n_neighbors = 10L,
  MN_ratio = 0.5,
  FP_ratio = 2,
  num_iters = c(100L, 100L, 250L),
  lr = 1,
  apply_pca = TRUE,
  init = "pca",
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`data`	Numeric matrix (cells x features)
`n_components`	Output dimensions (default 2)
`n_neighbors`	Number of nearest-neighbor pairs per point (default 10)
`MN_ratio`	Mid-near pair ratio (default 0.5)
`FP_ratio`	Further pair ratio (default 2.0)
`num_iters`	Iteration counts per phase: c(phase1, phase2, phase3)
`lr`	Adam learning rate (default 1.0)
`apply_pca`	Apply PCA to 100D if ncol > 100 (default TRUE)
`init`	Initialization: "pca" or "random"
`verbose`	Print progress
`seed`	Random seed (NULL for non-deterministic)

Value

n x n_components numeric matrix of embedding coordinates

Mparc *PARC clustering (pure R implementation)*

Description

PARC clustering (pure R implementation)

Usage

```
Mparc(
  data,
  k = 30L,
  dist_std_local = 3,
  jac_std_global = "median",
  keep_all_local_dist = "auto",
  too_big_factor = 0.4,
  small_pop = 10L,
  jac_weighted_edges = TRUE,
  n_iter_leiden = 5L,
  seed = 42L,
  num_threads = 1L,
  distance = "l2",
  time_smallpop = 15,
  partition_type = "ModularityVP",
  resolution_parameter = 1,
  hnsw_param_ef_construction = 150L,
  verbose = FALSE
)
```

Arguments

data	Numeric matrix (cells x features)
k	Number of nearest neighbors (default 30)
dist_std_local	Local pruning threshold in SDs above mean (default 3)
jac_std_global	"median" or numeric SDs below mean for Jaccard threshold
keep_all_local_dist	"auto", TRUE, or FALSE — skip local pruning
too_big_factor	Fraction of n above which clusters get sub-clustered
small_pop	Minimum cluster size; smaller clusters get merged
jac_weighted_edges	Use Jaccard weights in Leiden (default TRUE)
n_iter_leiden	Number of Leiden iterations (default 5)
seed	Random seed
num_threads	Threads for HNSW (default 1)

distance	Distance metric for HNSW ("l2", "cosine", "ip")
time_smallpop	Max seconds for small-pop merging loop
partition_type	"ModularityVP" or "RBVP"
resolution_parameter	Leiden resolution (triggers RBVP if != 1)
hnsw_param_ef_construction	HNSW ef_construction (default 150)
verbose	Print progress messages

Value

Integer vector of 1-indexed cluster labels (length = nrow(data))

Mphenograph *Pure R PhenoGraph Clustering (MARMOT Edition)*

Description

Pure R reimplementation of the PhenoGraph algorithm for high-dimensional single-cell data analysis. Builds a k-nearest-neighbor graph with Jaccard-weighted edges and identifies communities using the Louvain method.

Usage

```
Mphenograph(data, k = 30, seed = 2, verbose = FALSE)
```

Arguments

data	matrix or data.frame; input data (rows = cells, columns = features)
k	integer; number of nearest neighbours (default: 30)
seed	integer; random seed for Louvain community detection (default: 2)
verbose	logical; print progress messages (default: FALSE)

Details

This function is based on and owes full credit to the **PhenoGraph** algorithm by Levine et al. (2015) and its R implementation **Rphenograph** by Hao Chen (original) and the i-cyto fork. The Jaccard weighting, graph construction, and community detection logic are faithfully reimplemented from the Rphenograph C++ source to produce byte-identical results. Mphenograph exists solely to provide an always-installable pure R alternative that requires no C++ compilation.

It uses RANN for exact k-NN search and computes Jaccard coefficients in pure R using either sparse matrix operations (for n <= 200,000) or a pre-sorted merge-scan loop (for larger datasets).

The Jaccard weight formula matches the C++ Rphenograph implementation exactly: `w = u / (4k - 2u)` where u is the intersection count of neighbor sets. This is the standard Jaccard coefficient `u / (2k - u)` divided by 2 for symmetrization, since both directional edges are retained in the graph.

Isolated nodes (those sharing no neighbors with any of their k-NN) receive a self-loop with weight 0.5, matching the C++ behavior.

Value

A list of two elements (matching Rphenograph return format):

`[[1]]`	igraph graph object with Jaccard-weighted edges
`[[2]]`	igraph communities object from Louvain clustering

References

Levine, J.H. et al. (2015). Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1), 184–197. doi:10.1016/j.cell.2015.05.047

Chen, H. (2015). Rphenograph: R implementation of the PhenoGraph algorithm. https://github.com/JinmiaoChenLab/Rphenograph

i-cyto fork of Rphenograph. https://github.com/i-cyto/Rphenograph

See Also

membership, modularity, MfastPG

---

plotExprHeatmap *Expression Heatmap*

Description

Create a heatmap of expression values with customizable clustering and annotations. This function masks the CATALYST::plotExprHeatmap function but removes the factor relevelling

Usage

```
plotExprHeatmap(
  x,
  features = NULL,
  by = c("sample_id", "cluster_id", "both"),
  k = "meta20",
  m = NULL,
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  scale = c("first", "last", "never"),
  q = 0.01,
  row_anno = TRUE,
  col_anno = TRUE,
  row_clust = TRUE,
  col_clust = TRUE,
  row_dend = TRUE,
  col_dend = TRUE,
  bars = FALSE,
  perc = FALSE,
```

```
  bin_anno = FALSE,
  hm_pal = rev(RColorBrewer::brewer.pal(11, "RdYlBu")),
  k_pal = .marmot_cluster_cols,
  m_pal = k_pal,
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  linkage = c("average", "ward.D", "single", "complete", "mcquitty", "median",
    "centroid", "ward.D2")
)
```

Arguments

x	A SingleCellExperiment object
features	Character vector of features to include. If NULL, uses all features.
by	Character specifying grouping: "sample_id", "cluster_id", or "both"
k	Character specifying clustering to use (default: "meta20")
m	Character specifying metaclustering (optional)
assay	Character specifying which assay to use (default: "exprs")
fun	Character specifying aggregation function: "median", "mean", or "sum"
scale	Character specifying when to scale: "first", "last", or "never"
q	Numeric quantile for scaling (default: 0.01)
row_anno	Logical or character vector for row annotations (default: TRUE)
col_anno	Logical or character vector for column annotations (default: TRUE)
row_clust	Logical for row clustering (default: TRUE)
col_clust	Logical for column clustering (default: TRUE)
row_dend	Logical for row dendrogram display (default: TRUE)
col_dend	Logical for column dendrogram display (default: TRUE)
bars	Logical for frequency bars (default: FALSE)
perc	Logical for percentage display (default: FALSE)
bin_anno	Logical for binary annotation display (default: FALSE)
hm_pal	Colour palette for heatmap (default: rev(RColorBrewer::brewer.pal(11, "RdYlBu")))
k_pal	Colour palette for cluster annotations (default: .marmot_cluster_cols)
m_pal	Colour palette for metacluster annotations (default: k_pal)
distance	Character specifying distance metric for clustering
linkage	Character specifying linkage method for clustering

Value

A ComplexHeatmap object

Note

This function masks CATALYST::plotExprHeatmap. Load this package after CATALYST or use `conflicted` package to manage function conflicts.

Author(s)

Helena Crowell (original CATALYST implementation)

`plotFreqHeatmap` *Frequency Heatmap*

Description

Create a heatmap of cluster frequencies across samples. This function masks the CATALYST::plotFreqHeatmap function but removes the factor relevelling

Usage

```
plotFreqHeatmap(
  x,
  k = "meta20",
  m = NULL,
  normalize = TRUE,
  row_anno = TRUE,
  col_anno = TRUE,
  row_clust = TRUE,
  col_clust = TRUE,
  row_dend = TRUE,
  col_dend = TRUE,
  bars = TRUE,
  perc = FALSE,
  hm_pal = rev(RColorBrewer::brewer.pal(11, "RdBu")),
  k_pal = .marmot_cluster_cols,
  m_pal = k_pal
)
```

Arguments

`x`	A SingleCellExperiment object
`k`	Character specifying clustering to use (default: "meta20")
`m`	Character specifying metaclustering (optional)
`normalize`	Logical whether to z-normalize frequencies using arcsine-square-root transformation (default: TRUE)
`row_anno`	Logical or character vector for row annotations (default: TRUE)
`col_anno`	Logical or character vector for column annotations (default: TRUE)
`row_clust`	Logical for row clustering (default: TRUE)
`col_clust`	Logical for column clustering (default: TRUE)
`row_dend`	Logical for row dendrogram display (default: TRUE)
`col_dend`	Logical for column dendrogram display (default: TRUE)

bars	Logical for frequency bars (default: TRUE)
perc	Logical for percentage display (default: FALSE)
hm_pal	Colour palette for heatmap (default: rev(RColorBrewer::brewer.pal(11, "RdBu")))
k_pal	Colour palette for cluster annotations (default: .marmot_cluster_cols)
m_pal	Colour palette for metacluster annotations (default: k_pal)

Value

A ComplexHeatmap object

Note

This function masks CATALYST::plotFreqHeatmap. Load this package after CATALYST or use `conflicted` package to manage function conflicts.

Author(s)

Helena Crowell (original CATALYST implementation)

`reconstruct_sce_from_h5ad`

Reconstruct a SingleCellExperiment from h5ad

Description

Reads the h5ad file and converts to SCE, restoring factor columns and injecting cluster_codes + experiment_info into metadata.

Usage

```
reconstruct_sce_from_h5ad(h5ad_path, ad = NULL)
```

Arguments

h5ad_path	Path to the marmot_results.h5ad file

Value

A SingleCellExperiment object, or NULL if file is missing

save_da_ds_h5ad *Save DA/DS Results to h5ad*

Description

Reads the existing h5ad, updates DA/DS results in uns, and rewrites. Called both during fresh pipeline runs and during RDataFolder reload (DA/DS always re-runs).

Usage

```
save_da_ds_h5ad(
  h5ad_path,
  daList = NULL,
  dsList = NULL,
  selectedClustersList = NULL,
  daPValToUse = NULL
)
```

Arguments

h5ad_path	Path to the marmot_results.h5ad file
daList	List of DA result data frames (one per contrast)
dsList	List of DS result data frames (one per contrast)
selectedClustersList	List of selected DA clusters
daPValToUse	Character: "p_adj" or "p_val"

save_h5ad_data *Save All Pipeline Data to h5ad*

Description

Packs all pipeline R objects into a single h5ad file using the AnnData format. SCE data maps to standard AnnData slots (X, obs, var, obsm, layers). MARMOT-specific metadata is stored in the uns slot.

Usage

```
save_h5ad_data(output_dir, envir = .GlobalEnv)
```

Arguments

output_dir	Path to the R_files output directory
envir	Environment containing the pipeline variables (default: .GlobalEnv)

setup_python *Set up the Python environment for PARC and PaCMAP*

Description

Creates (or repairs) the `p4r` conda environment from the bundled `environment.yml`, giving MARMOT access to PARC (clustering) and PaCMAP (dimensionality reduction).

Usage

```
setup_python(conda = NULL, force = FALSE)
```

Arguments

`conda`	Path to conda/mamba binary. If `NULL` (default), auto-detected via `reticulate::conda_binary()`.
`force`	If `TRUE`, recreate the environment even if it already exists and appears healthy. Default `FALSE`.

Value

Invisibly returns the path to the Python binary in the environment.

shinyMarmot *shinyMarmot*

Description

Start the shinyMarmot application locally using results generated from the main MARMOT pipeline.

Usage

```
shinyMarmot(marmot_output = NA, demo = NA)
```

Arguments

`marmot_output`	Path to the pipeline output directory (R_files folder)
`demo`	Logical; if TRUE, launch with bundled demo data (default NA)

Value

An interactive shiny app session in console.

Author(s)

Peter Leary

Examples

```
## Not run:
shinyMarmot(marmot_output = "~/Desktop/Flow_Data/Results_2025-03-10_11_01_01/R_files")

## End(Not run)
```

Index

addMetadataToFCSFolder, 3

check_setup, 3

export_claude_skill, 4

h5ad_io, 5

install_claude_skill, 5
install_marmot_extras, 6
install_marmot_python, 7

launch_gui, 7
load_h5ad_for_shiny, 8
load_h5ad_to_env, 8

marmot, 9
marmot-python, 9
marmot-python-subprocess, 10
marmot_python_status, 10
membership, *11*, *15*
MfastPG, 10, *15*
modularity, *15*
Mpacmap, 12
Mparc, 13
Mphenograph, *11*, 14

plotExprHeatmap, 15
plotFreqHeatmap, 17

reconstruct_sce_from_h5ad, 18

save_da_ds_h5ad, 19
save_h5ad_data, 19
setup_python, 20
shinyMarmot, 20