Package: fmridataset (via r-universe)

May 11, 2026

Type Package

Title Unified Container for fMRI Datasets

Version 0.8.9

Description Provides a unified S3 class 'fmri_dataset' for representing functional magnetic resonance imaging (fMRI) data from various sources including raw NIfTI files, BIDS projects, pre-loaded NeuroVec objects, and in-memory matrices. Features lazy loading, flexible data access patterns, and integration with neuroimaging analysis workflows.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.3.0)

Imports assertthat, cachem, deflist, delarr, fmrihrf, fs, lifecycle, memoise, Matrix, methods, neuroim2, purrr, tibble, utils

Suggests arrow, bench, bidser, blosc, crayon, DelayedArray, DelayedMatrixStats, devtools, dplyr, fmrilatent, fmristore, foreach, hdf5r, iterators, jsonlite, knitr, microbenchmark, mockery, pryr, rmarkdown, testthat (>= 3.0.0), withr, yaml, zarr

VignetteBuilder knitr

Remotes bbuchsbaum/delarr

URL https://github.com/bbuchsbaum/fmridataset, https://bbuchsbaum.github.io/fmridataset/

BugReports https://github.com/bbuchsbaum/fmridataset/issues

Config/pak/sysreqs cmake make libhdf5-dev libicu-dev libuv1-dev

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

Date/Publication 2026-05-05 20:26:36 UTC

RemoteUrl https://github.com/bbuchsbaum/fmridataset

RemoteRef HEAD

RemoteSha ad1ae84cc129c12cc131940f52f32b59de32fe28

Contents

`all_selector` *All Voxels Series Selector*

Description

Select all voxels in the dataset mask.

Usage

```
all_selector()
```

Value

An object of class `all_selector`

Examples

```
# Select all voxels
sel <- all_selector()
```

`as.matrix.fmri_series` *Convert fmri_series to Matrix*

Description

This method realizes the underlying lazy matrix and returns an ordinary matrix with timepoints in rows and voxels in columns.

Usage

```
## S3 method for class 'fmri_series'
as.matrix(x, ...)
```

Arguments

x	An fmri_series object
...	Additional arguments (ignored)

Value

A matrix with timepoints as rows and voxels as columns

See Also

fmri_series for the class definition, as_tibble.fmri_series for tibble conversion

Examples

```
# Create small example
mat <- matrix(rnorm(20), nrow = 4, ncol = 5)

backend <- matrix_backend(mat, mask = rep(TRUE, ncol(mat)))
dataset <- fmri_dataset(backend, TR = 1, run_length = nrow(mat))
fs <- fmri_series(dataset)

# Convert to matrix
mat_result <- as.matrix(fs)
dim(mat_result) # 4 x 5
```

---

as.matrix_dataset *Convert to Matrix Dataset*

Description

Generic function to convert various fMRI dataset types to matrix_dataset objects. Provides a unified interface for getting matrix-based representations.

Usage

```
as.matrix_dataset(x, ...)
```

Arguments

x	An fMRI dataset object
...	Additional arguments passed to methods

Details

This function converts different dataset representations to the standard matrix_dataset format, which stores data as a matrix with timepoints in rows and voxels in columns. This is useful for algorithms that require matrix operations or when a consistent data format is needed.

Value

A matrix_dataset object with the same data as the input

See Also

`matrix_dataset` for creating matrix datasets, `get_data_matrix` for extracting data as matrix

Examples

```
# Convert various dataset types to matrix_dataset
# (example requires actual dataset object)
# mat_ds <- as.matrix_dataset(some_dataset)
```

---

as_delarr *Convert backend to a delarr lazy matrix*

Description

Provides a lightweight S3 interface that defers materialization of backend data. The returned object is compatible with `delarr::collect()` as well as base `as.matrix()` for realization.

Usage

```
as_delarr(backend, ...)

## S3 method for class 'matrix_backend'
as_delarr(backend, ...)

## S3 method for class 'nifti_backend'
as_delarr(backend, ...)

## S3 method for class 'study_backend'
as_delarr(backend, ...)

## Default S3 method:
as_delarr(backend, ...)
```

Arguments

backend	A storage backend object
...	Passed to methods

Value

A `delarr` lazy matrix

`as_delayed_array` *Convert Backend to DelayedArray*

Description

Provides a DelayedArray interface for storage backends. The returned object lazily retrieves data via the backend when subsets of the array are accessed.

Usage

```
as_delayed_array(backend, sparse_ok = FALSE, ...)

## S3 method for class 'nifti_backend'
as_delayed_array(backend, sparse_ok = FALSE, ...)

## S3 method for class 'matrix_backend'
as_delayed_array(backend, sparse_ok = FALSE, ...)

## S3 method for class 'study_backend'
as_delayed_array(backend, sparse_ok = FALSE, ...)

## Default S3 method:
as_delayed_array(backend, sparse_ok = FALSE, ...)
```

Arguments

`backend`	A storage backend object
`sparse_ok`	Logical, allow sparse representation when possible
`...`	Additional arguments passed to methods

Value

A DelayedArray object

Examples

```
## Not run:
b <- matrix_backend(matrix(rnorm(20), nrow = 5))
da <- as_delayed_array(b)
dim(da)

## End(Not run)
```

as_delayed_array-dataset *Convert Dataset Objects to DelayedArray*

Description

Provides DelayedArray interface for dataset objects. These methods convert fmri_dataset and matrix_dataset objects to DelayedArrays for memory-efficient operations.

Usage

```
## S4 method for signature 'matrix_dataset'
as_delayed_array(backend, sparse_ok = FALSE)

## S4 method for signature 'fmri_file_dataset'
as_delayed_array(backend, sparse_ok = FALSE)

## S4 method for signature 'fmri_mem_dataset'
as_delayed_array(backend, sparse_ok = FALSE)
```

Arguments

backend	A storage backend object
sparse_ok	Logical, allow sparse representation when possible

as_fmri_group *Coerce a data frame into an fmri_group*

Description

Coerce a data frame into an fmri_group

Usage

```
as_fmri_group(
  subjects,
  id,
  dataset_col = "dataset",
  space = NULL,
  mask_strategy = c("subject_specific", "intersect", "union")
)
```

Arguments

subjects	A data.frame (or tibble) with one row per subject where one column contains per-subject fmri_dataset objects stored as a list column.
id	Character scalar giving the name of the subject identifier column.
dataset_col	Character scalar naming the list column that stores the per-subject dataset handles.
space	Optional character string describing the nominal common space for all subjects (e.g., "MNI152NLin2009cAsym").
mask_strategy	One of "subject_specific", "intersect", or "union" describing how masks should be handled when combining subjects. This is a declarative flag only; no resampling is performed by the constructor.

Value

An fmri_group object.

---

as_tibble.fmri_series *Convert fmri_series to Tibble*

Description

The returned tibble contains one row per voxel/timepoint combination with metadata columns from temporal_info and voxel_info and a signal column with the data values.

Usage

```
## S3 method for class 'fmri_series'
as_tibble(x, ...)
```

Arguments

x	An fmri_series object
...	Additional arguments (ignored)

Value

A tibble with columns from temporal_info, voxel_info, and a signal column containing the fMRI signal values

See Also

fmri_series for the class definition, as.matrix.fmri_series for matrix conversion

Examples

```
# Create small example
mat <- matrix(rnorm(12), nrow = 3, ncol = 4)

backend <- matrix_backend(mat, mask = rep(TRUE, ncol(mat)))
dataset <- fmri_dataset(backend, TR = 1, run_length = nrow(mat))
fs <- fmri_series(dataset)

# Convert to tibble
tbl_result <- tibble::as_tibble(fs)
# Result has 12 rows (3 timepoints x 4 voxels)
# with columns: time, voxel, signal
```

---

as_tibble.fmri_study_dataset

Convert fmri_study_dataset to a tibble or lazy matrix

Description

Primary data access method for study-level datasets. By default this returns a lazy matrix (typically a `delarr` object) with row-wise metadata attached. When `materialise = TRUE`, the data matrix is materialised and returned as a tibble with metadata columns prepended.

Usage

```
## S3 method for class 'fmri_study_dataset'
as_tibble(x, materialise = FALSE, ...)
```

Arguments

`x`	An `fmri_study_dataset` object
`materialise`	Logical; return a materialised tibble? Default `FALSE`.
`...`	Additional arguments (unused)

Value

Either a lazy matrix with metadata attributes or a tibble when `materialise = TRUE`.

bids_h5_dataset *Open a BIDS HDF5 Study Archive*

Description

Opens a BIDS HDF5 archive created by `compress_bids_study()` and returns a `bids_h5_study_dataset` that is a subclass of `fmri_study_dataset`. All standard **fmridataset** methods (`get_data_matrix`, `data_chunks`, `as_delarr`, etc.) work on the returned object.

Usage

```
bids_h5_dataset(file, preload = FALSE)
```

Arguments

`file`	Character string. Path to the `.h5` BIDS archive.
`preload`	Logical. Reserved for future use (ignored in Phase 1).

Value

A `bids_h5_study_dataset` object (subclass of `fmri_study_dataset`).

See Also

`compress_bids_study`, `subset_bids_h5`, `participants`, `tasks`, `sessions`

blockids *Get Block IDs from Sampling Frame*

Description

Generates a vector of block/run identifiers for each timepoint.

Usage

```
blockids(x, ...)

## S3 method for class 'fmri_dataset'
blockids(x, ...)

## S3 method for class 'sampling_frame'
blockids(x, ...)
```

Arguments

x	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
...	Additional arguments passed to methods

Details

This function creates a vector where each element indicates which run/block the corresponding timepoint belongs to. This is useful for run-wise analyses or modeling run effects.

Value

Integer vector of length equal to total timepoints, with values indicating run membership (1 for first run, 2 for second, etc.)

See Also

`get_run_lengths` for run lengths, `samples` for timepoint indices

Examples

```
# Create a sampling frame with 2 runs of different lengths
sf <- fmrihrf::sampling_frame(blocklens = c(3, 4), TR = 2)
blockids(sf) # Returns: c(1, 1, 1, 2, 2, 2, 2)
```

---

blocklens *Get Block Lengths from Objects*

Description

Generic function to extract block/run lengths from various objects. Extends the sampling_frame generic to work with dataset objects.

Usage

```
blocklens(x, ...)

## S3 method for class 'fmri_dataset'
blocklens(x, ...)

## S3 method for class 'sampling_frame'
blocklens(x, ...)
```

Arguments

x	An object with block structure (e.g., sampling_frame, fmri_dataset)
...	Additional arguments passed to methods

Details

In fMRI experiments, data is often collected in multiple runs or blocks. This function extracts the length (number of timepoints) of each run. The sum of block lengths equals the total number of timepoints.

Value

Integer vector where each element represents the number of timepoints in the corresponding run/block

See Also

`n_runs` for number of runs, `n_timepoints` for total timepoints, `get_run_lengths` for alternative function name

Examples

```
# Create a dataset with 3 runs
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120, 110), TR = 2)
blocklens(sf) # c(100, 120, 110)

# Create dataset with multiple runs
mat <- matrix(rnorm(330 * 50), nrow = 330, ncol = 50)
ds <- matrix_dataset(mat, TR = 2, run_length = c(100, 120, 110))
blocklens(ds) # c(100, 120, 110)
```

---

collect_chunks *Collect all chunks from a chunk iterator*

Description

This function collects all chunks from a chunk iterator into a list.

Usage

```
collect_chunks(chunk_iter)
```

Arguments

chunk_iter	A chunk iterator object created by chunk_iter()

Value

A list containing all chunks from the iterator

compress_bids_study *Compress a BIDS Study into a Single HDF5 Archive*

Description

Converts a BIDS directory (or `bidser::bids_project`) into a single compressed HDF5 file containing compressed fMRI data, events, confounds, and study metadata. The output file can be opened with `bids_h5_dataset`.

Usage

```
compress_bids_study(
  x,
  file,
  mode = c("parcellated", "latent"),
  clusters = NULL,
  summary_fun = mean,
  encoding = NULL,
  n_components = NULL,
  template = NULL,
  mask = NULL,
  space = "MNI152NLin2009cAsym",
  tasks = NULL,
  subjects = NULL,
  sessions = NULL,
  confounds = NULL,
  compression = 4L,
  verbose = TRUE
)
```

Arguments

`x`	A `bidser::bids_project` object **or** a character path to a BIDS directory (automatically opened with `bidser::bids_project()`).
`file`	Character. Path for the output `.h5` file. Parent directory must exist. Existing files are overwritten.
`mode`	Character. Compression strategy: `"parcellated"` (default) or `"latent"`.
`clusters`	A `neuroim2::ClusteredNeuroVol` defining the parcellation atlas in study space. Required for `mode = "parcellated"`; ignored for `"latent"`.
`summary_fun`	Function applied to voxel time-series within each parcel to produce a scalar summary (default: `mean`). Only used for `mode = "parcellated"`.
`encoding`	A `fmrilatent` encoding specification object (e.g. `fmrilatent::spec_time_dct(k = 15)`). Required for `mode = "latent"` unless `n_components` is provided.
`n_components`	Integer. Shorthand for latent PCA with K components. If `encoding` is `NULL` and `n_components` is provided, `fmrilatent::spec_space_pca(k = n_components)` is used. Only used for `mode = "latent"`.

template	Optional fmrilatent template object (e.g. from fmrilatent::parcel_basis_template() or fmrilatent::build_hierarchical_template()). When provided, the template's spatial loadings are stored once in /latent_meta/template/ and per-scan data is reduced to [T, K] projection coefficients (no per-scan loadings). This significantly reduces file size for multi-subject studies. Only used for mode = "latent".
mask	A neuroim2::LogicalNeuroVol brain mask. For mode = "parcellated", derived from clusters when NULL. For mode = "latent", mask is required (cannot be derived without clusters).
space	Character. Template space name stored as metadata (default: "MNI152NLin2009cAsym").
tasks	Character vector. Task filter; NULL means all tasks.
subjects	Character vector. Subject filter; NULL means all subjects.
sessions	Character vector. Session filter; NULL means all sessions (including session-less datasets).
confounds	A confound specification passed to bidser::read_confounds(), e.g. a character vector of column names, a bidser::confound_set(), or NULL to skip confound writing.
compression	Integer 0–9. HDF5 gzip compression level (default 4).
verbose	Logical. If TRUE (default) print progress messages.

Details

The writer streams scans one at a time — only one NIfTI image is held in memory at a time. For each scan it:

1. Reads the NIfTI via neuroim2::read_vec().
2. For **parcellated** mode: computes parcel averages via fmristore::summarize_by_clusters() and writes [T, K] to /scans/<name>/data/summary_data.
3. For **latent** mode: encodes via fmrilatent::encode() and writes basis [T, K], loadings [V, K], and (optionally) offset [V] to /scans/<name>/data/.
4. Writes events, confounds, censor, and metadata sub-groups.
5. Releases the NIfTI from memory.

After all scans are written the /scan_index/ lookup table is populated and the function returns a bids_h5_dataset reader object.

Value

A bids_h5_dataset object (reader for the newly created file). If the reader is not yet available the file path is returned invisibly.

HDF5 schema

See bids_plan.md in the package source for the full v1.0 schema. The root compression_mode attribute reflects the chosen mode.

Examples

```
## Not run:
library(bidser)
library(neuroim2)
library(fmristore)

bids_dir  <- system.file("extdata", "ds001", package = "bidser")
atlas     <- fmristore::get_schaefer_atlas(100)   # example atlas

# Parcellated mode
study <- compress_bids_study(
  x          = bids_dir,
  file       = tempfile(fileext = ".h5"),
  clusters   = atlas,
  tasks      = "nback",
  verbose    = TRUE
)

# Latent mode (PCA with 50 components)
study_lat <- compress_bids_study(
  x            = bids_dir,
  file         = tempfile(fileext = ".h5"),
  mode         = "latent",
  n_components = 50L,
  mask         = brain_mask,
  tasks        = "nback",
  verbose      = TRUE
)

## End(Not run)
```

`create_backend` *Create Backend Instance*

Description

Creates a backend instance using the registered factory function. This is the main interface for creating backends by name.

Usage

```
create_backend(name, ..., validate = TRUE)
```

Arguments

`name`	Character string, name of registered backend type
`...`	Arguments passed to the backend factory function
`validate`	Logical, whether to validate the created backend (default: TRUE)

Value

A storage backend object

Examples

```
## Not run:
# Create a NIfTI backend (assuming it's registered)
backend <- create_backend("nifti",
  source = "data.nii",
  mask_source = "mask.nii"
)

# Create with validation disabled (faster, but riskier)
backend <- create_backend("nifti",
  source = "data.nii",
  mask_source = "mask.nii",
  validate = FALSE
)

## End(Not run)
```

---

data_chunk *Create a Data Chunk Object*

Description

Creates a data chunk object that represents a subset of data from an fMRI dataset. A data chunk contains the data matrix along with indices indicating which voxels and time points (rows) are included in the chunk.

Usage

```
data_chunk(mat, voxel_ind, row_ind, chunk_num)
```

Arguments

mat	A matrix containing the chunk data (rows = time points, columns = voxels)
voxel_ind	Integer vector of voxel indices included in this chunk
row_ind	Integer vector of row (time point) indices included in this chunk
chunk_num	Integer indicating the chunk number

Value

A data_chunk object containing:

data The data matrix for this chunk

voxel_ind Indices of voxels in this chunk

row_ind Indices of rows (time points) in this chunk

chunk_num The chunk number

Examples

```
# Create a simple data chunk
mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
chunk <- data_chunk(mat, voxel_ind = 1:10, row_ind = 1:10, chunk_num = 1)
print(chunk)
```

---

data_chunks *Create Data Chunks for Processing*

Description

Generic function to create data chunks for parallel processing from various fMRI dataset types. Supports different chunking strategies.

Usage

```
data_chunks(x, nchunks = 1, runwise = FALSE, ...)
```

Arguments

x	An fMRI dataset object
nchunks	Number of chunks to create (default: 1)
runwise	If TRUE, create run-wise chunks (default: FALSE)
...	Additional arguments passed to methods

Details

Large fMRI datasets can be processed more efficiently by dividing them into chunks. This function creates an iterator that yields data chunks for parallel or sequential processing. Two chunking strategies are supported:

- Equal-sized chunks: Divides voxels into approximately equal groups
- Run-wise chunks: Each chunk contains all voxels from one or more complete runs

Value

A chunk iterator object that yields data chunks when iterated

See Also

iter for iteration concepts

Examples

```
# Create a dataset
mat <- matrix(rnorm(100 * 1000), nrow = 100, ncol = 1000)
ds <- matrix_dataset(mat, TR = 2, run_length = 100)

# Create 4 chunks for parallel processing
chunks <- data_chunks(ds, nchunks = 4)

# Process chunks (example)
# results <- foreach(chunk = chunks) %dopar% {
#   process_chunk(chunk)
# }
```

`data_chunks.fmri_file_dataset` *Create Data Chunks for fmri_file_dataset Objects*

Description

This function creates data chunks for fmri_file_dataset objects. It allows for the retrieval of run-wise or sequence-wise data chunks, as well as arbitrary chunks.

Usage

```
## S3 method for class 'fmri_file_dataset'
data_chunks(x, nchunks = 1, runwise = FALSE, ...)
```

Arguments

`x`	An object of class 'fmri_file_dataset'.
`nchunks`	The number of data chunks to create. Default is 1.
`runwise`	If TRUE, the data chunks are created run-wise. Default is FALSE.
`...`	Additional arguments.

Value

A list of data chunks, with each chunk containing the data, voxel indices, row indices, and chunk number.

`data_chunks.fmri_mem_dataset` *Create Data Chunks for fmri_mem_dataset Objects*

Description

This function creates data chunks for fmri_mem_dataset objects. It allows for the retrieval of run-wise or sequence-wise data chunks, as well as arbitrary chunks.

Usage

```
## S3 method for class 'fmri_mem_dataset'
data_chunks(x, nchunks = 1, runwise = FALSE, ...)
```

Arguments

x	An object of class 'fmri_mem_dataset'.
nchunks	The number of data chunks to create. Default is 1.
runwise	If TRUE, the data chunks are created run-wise. Default is FALSE.
...	Additional arguments.

Value

A list of data chunks, with each chunk containing the data, voxel indices, row indices, and chunk number.

Examples

```
## Not run:
# Create a simple fmri_mem_dataset for demonstration
d <- c(10, 10, 10, 10)
nvec <- neuroim2::NeuroVec(array(rnorm(prod(d)), d), space = neuroim2::NeuroSpace(d))
mask <- neuroim2::LogicalNeuroVol(array(TRUE, d[1:3]), neuroim2::NeuroSpace(d[1:3]))
dset <- fmri_mem_dataset(list(nvec), mask, TR = 2)

# Create an iterator with 5 chunks
iter <- data_chunks(dset, nchunks = 5)
`%do%` <- foreach::`%do%`
y <- foreach::foreach(chunk = iter) %do% {
  colMeans(chunk$data)
}
length(y) == 5

# Create an iterator with 100 chunks
iter <- data_chunks(dset, nchunks = 100)
y <- foreach::foreach(chunk = iter) %do% {
  colMeans(chunk$data)
}
```

```
length(y) == 100

# Create a "runwise" iterator
iter <- data_chunks(dset, runwise = TRUE)
y <- foreach::foreach(chunk = iter) %do% {
  colMeans(chunk$data)
}
length(y) == 1

## End(Not run)
```

`data_chunks.fmri_study_dataset` *Create Data Chunks for fmri_study_dataset Objects*

Description

This function creates data chunks for multi-subject study datasets.

Usage

```
## S3 method for class 'fmri_study_dataset'
data_chunks(x, nchunks = 1, runwise = FALSE, ...)
```

Arguments

`x`	An object of class 'fmri_study_dataset'
`nchunks`	The number of chunks to split the data into. Default is 1.
`runwise`	If TRUE, creates run-wise chunks instead of arbitrary chunks
`...`	Additional arguments passed to methods

Value

A list of data chunks, each containing data, indices and chunk number

---

`data_chunks.matrix_dataset` *Create Data Chunks for matrix_dataset Objects*

Description

This function creates data chunks for matrix_dataset objects. It allows for the retrieval of run-wise or sequence-wise data chunks, as well as arbitrary chunks.

Usage

```
## S3 method for class 'matrix_dataset'
data_chunks(x, nchunks = 1, runwise = FALSE, ...)
```

Arguments

x	An object of class 'matrix_dataset'
nchunks	The number of chunks to split the data into. Default is 1.
runwise	If TRUE, creates run-wise chunks instead of arbitrary chunks
...	Additional arguments passed to methods

Value

A list of data chunks, each containing data, indices and chunk number

---

dim.fmri_series *Dimensions of fmri_series*

Description

Dimensions of fmri_series

Usage

```
## S3 method for class 'fmri_series'
dim(x)
```

Arguments

x	An fmri_series object

Value

Integer vector of length 2 (timepoints, voxels)

encoding_info *Get Encoding Metadata from a Latent-Mode BIDS H5 Dataset*

Description

Returns encoding metadata (family, parameters, number of components) stored in the `/latent_meta/` group of a latent-mode BIDS HDF5 archive. Returns `NULL` for parcellated-mode archives.

Usage

```
encoding_info(x, ...)

## S3 method for class 'bids_h5_study_dataset'
encoding_info(x, ...)
```

Arguments

`x`	A `bids_h5_study_dataset` object.
`...`	Additional arguments passed to methods.

Value

A named list with elements `encoding_family`, `encoding_params`, and `n_components`, or `NULL`.

exec_strategy *Create an Execution Strategy for Data Processing*

Description

This function creates an execution strategy that can be used to process fMRI datasets in different ways: voxelwise, runwise, or chunkwise.

Usage

```
exec_strategy(
  strategy = c("voxelwise", "runwise", "chunkwise"),
  nchunks = NULL
)
```

Arguments

`strategy`	Character string specifying the processing strategy. Options are "voxelwise", "runwise", or "chunkwise".
`nchunks`	Number of chunks to use for "chunkwise" strategy. Ignored for other strategies.

Value

A function that takes a dataset and returns a chunk iterator configured according to the specified strategy.

`filter_subjects` *Filter subjects in an fmri_group*

Description

Expressions are evaluated in the context of `subjects(gd)` and may refer to its columns directly. Multiple expressions are combined with logical AND.

Usage

```
filter_subjects(gd, ...)
```

Arguments

`gd`	An `fmri_group`.
`...`	Logical expressions used to filter rows.

Value

An updated `fmri_group` containing only the rows that satisfy all predicates.

`fmri_cache_info` *Get cache information and statistics*

Description

Returns information about the current state of the fmridataset cache, including size, number of objects, hit/miss rates, and memory usage.

Usage

```
fmri_cache_info()
```

Value

Named list with cache statistics

Examples

```
## Not run:
# Get cache information
cache_info <- fmri_cache_info()
print(cache_info)

## End(Not run)
```

`fmri_cache_resize` *Resize the fmridataset cache*

Description

Changes the maximum size of the cache. This will immediately evict objects if the new size is smaller than the current cache contents.

Usage

```
fmri_cache_resize(size_mb)
```

Arguments

`size_mb`	Numeric cache size in megabytes

Value

NULL (invisibly)

Examples

```
## Not run:
# Resize cache to 1GB
fmri_cache_resize(1024)

# Check new size
fmri_cache_info()

## End(Not run)
```

fmri_clear_cache *Clear fmridataset cache*

Description

Clears the internal cache used by fmridataset for memoized file operations. This can be useful to free memory or force re-reading of files.

Usage

```
fmri_clear_cache()
```

Value

NULL (invisibly)

Examples

```
## Not run:
# Clear the cache to free memory
fmri_clear_cache()

## End(Not run)
```

fmri_dataset *Create an fMRI Dataset Object from a Set of Scans*

Description

This function creates an fMRI dataset object from a set of scans, design information, and other data. The new implementation uses a pluggable backend architecture.

Usage

```
fmri_dataset(
  scans,
  mask = NULL,
  TR,
  run_length,
  event_table = data.frame(),
  base_path = ".",
  censor = NULL,
  preload = FALSE,
  mode = c("normal", "bigvec", "mmap", "filebacked"),
  backend = NULL,
  dummy_mode = FALSE
)
```

Arguments

scans	A vector of one or more file names of the images comprising the dataset, or a pre-created storage backend object.
mask	Name of the binary mask file indicating the voxels to include in the analysis. Ignored if scans is a backend object.
TR	The repetition time in seconds of the scan-to-scan interval.
run_length	A vector of one or more integers indicating the number of scans in each run.
event_table	A data.frame containing the event onsets and experimental variables. Default is an empty data.frame.
base_path	Base directory for relative file names. Absolute paths are used as-is.
censor	A binary vector indicating which scans to remove. Default is NULL.
preload	Read image scans eagerly rather than on first access. Default is FALSE.
mode	The type of storage mode ('normal', 'bigvec', 'mmap', filebacked'). Default is 'normal'. Ignored if scans is a backend object.
backend	Deprecated. Use scans parameter to pass a backend object.
dummy_mode	Logical, if TRUE allows non-existent files (for testing purposes only). Default is FALSE.

Value

An fMRI dataset object of class c("fmri_file_dataset", "volumetric_dataset", "fmri_dataset", "list").

Examples

```
## Not run:
# Create an fMRI dataset with 3 scans and a mask
dset <- fmri_dataset(c("scan1.nii", "scan2.nii", "scan3.nii"),
  mask = "mask.nii", TR = 2, run_length = rep(300, 3),
  event_table = data.frame(
    onsets = c(3, 20, 99, 3, 20, 99, 3, 20, 99),
    run = c(1, 1, 1, 2, 2, 2, 3, 3, 3)
  )
)

# Create an fMRI dataset with 1 scan and a mask
dset <- fmri_dataset("scan1.nii",
  mask = "mask.nii", TR = 2,
  run_length = 300,
  event_table = data.frame(onsets = c(3, 20, 99), run = rep(1, 3))
)

# Create an fMRI dataset with a backend
backend <- nifti_backend(c("scan1.nii", "scan2.nii"), mask_source = "mask.nii")
dset <- fmri_dataset(backend, TR = 2, run_length = c(150, 150))

# Create a dummy dataset for testing (files don't need to exist)
dset_dummy <- fmri_dataset(
```

```
  scans = c("dummy1.nii", "dummy2.nii"),
  mask = "dummy_mask.nii",
  TR = 2,
  run_length = c(100, 100),
  dummy_mode = TRUE # Enable dummy mode for testing
)

## End(Not run)
```

`fmri_dataset_legacy` *Legacy fMRI Dataset Constructor*

Description

Backward compatibility wrapper for fmri_dataset. This function provides the same interface as the original fmri_dataset function.

Usage

```
fmri_dataset_legacy(scans, mask, TR, run_length, preload = FALSE, ...)
```

Arguments

`scans`	Either a character vector of file paths to scans or a list of NeuroVec objects
`mask`	Either a character file path to a mask or a NeuroVol mask object
`TR`	The repetition time
`run_length`	Numeric vector of run lengths
`preload`	Whether to preload data into memory
`...`	Additional arguments passed to fmri_dataset

Value

An fmri_dataset object

Examples

```
## Not run:
# Create dataset from files
dset <- fmri_dataset_legacy(
  scans = c("scan1.nii", "scan2.nii"),
  mask = "mask.nii",
  TR = 2,
  run_length = c(100, 100)
)

## End(Not run)
```

fmri_group *Create an fmri_group (one row per subject)*

Description

Create an fmri_group (one row per subject)

Usage

```
fmri_group(
  subjects,
  id,
  dataset_col = "dataset",
  space = NULL,
  mask_strategy = c("subject_specific", "intersect", "union")
)
```

Arguments

subjects	A data.frame (or tibble) with one row per subject where one column contains per-subject fmri_dataset objects stored as a list column.
id	Character scalar giving the name of the subject identifier column.
dataset_col	Character scalar naming the list column that stores the per-subject dataset handles.
space	Optional character string describing the nominal common space for all subjects (e.g., "MNI152NLin2009cAsym").
mask_strategy	One of "subject_specific", "intersect", or "union" describing how masks should be handled when combining subjects. This is a declarative flag only; no resampling is performed by the constructor.

Value

An object of class fmri_group that wraps the input table.

fmri_h5_dataset *Create an fMRI Dataset Object from H5 Files*

Description

This function creates an fMRI dataset object specifically from H5 files using the fmristore package. Each scan is stored as an H5 file that loads to an H5NeuroVec object.

Usage

```
fmri_h5_dataset(
  h5_files,
  mask_source,
  TR,
  run_length,
  event_table = data.frame(),
  base_path = ".",
  censor = NULL,
  preload = FALSE,
  mask_dataset = "data/elements",
  data_dataset = "data"
)
```

Arguments

h5_files	A vector of one or more file paths to H5 files containing the fMRI data.
mask_source	File path to H5 mask file, regular mask file, or in-memory NeuroVol object.
TR	The repetition time in seconds of the scan-to-scan interval.
run_length	A vector of one or more integers indicating the number of scans in each run.
event_table	A data.frame containing the event onsets and experimental variables. Default is an empty data.frame.
base_path	Base directory for relative file names. Absolute paths are used as-is.
censor	A binary vector indicating which scans to remove. Default is NULL.
preload	Read H5NeuroVec objects eagerly rather than on first access. Default is FALSE.
mask_dataset	Character string specifying the dataset path within H5 file for mask (default: "data/elements").
data_dataset	Character string specifying the dataset path within H5 files for data (default: "data").

Value

An fMRI dataset object of class c("fmri_file_dataset", "volumetric_dataset", "fmri_dataset", "list").

Examples

```
## Not run:
# Create an fMRI dataset with H5NeuroVec files (standard fmristore format)
dset <- fmri_h5_dataset(
  h5_files = c("scan1.h5", "scan2.h5", "scan3.h5"),
  mask_source = "mask.h5",
  TR = 2,
  run_length = c(150, 150, 150)
)

# Create an fMRI dataset with H5 files and NIfTI mask
dset <- fmri_h5_dataset(
```

```
  h5_files = "single_scan.h5",
  mask_source = "mask.nii",
  TR = 2,
  run_length = 300
)

# Custom dataset paths (if using non-standard H5 structure)
dset <- fmri_h5_dataset(
  h5_files = "custom_scan.h5",
  mask_source = "custom_mask.h5",
  TR = 2,
  run_length = 200,
  data_dataset = "my_data_path",
  mask_dataset = "my_mask_path"
)

## End(Not run)
```

`fmri_latent_dataset` *Create an fMRI Dataset Object from LatentNeuroVec Files or Objects*

Description

[Deprecated]

This function is deprecated. Please use `latent_dataset()` instead, which provides a proper interface for latent space data.

Usage

```
fmri_latent_dataset(
  latent_files,
  mask_source = NULL,
  TR,
  run_length,
  event_table = data.frame(),
  base_path = ".",
  censor = NULL,
  preload = FALSE
)
```

Arguments

`latent_files`	Source files or objects
`mask_source`	Ignored
`TR`	The repetition time in seconds
`run_length`	Vector of run lengths
`event_table`	Event table

base_path	Base path for files
censor	Censor vector
preload	Whether to preload data

Value

A latent_dataset object

See Also

latent_dataset

Examples

```
## Not run:
# Use latent_dataset() instead:
dset <- latent_dataset(
  source = c("run1.lv.h5", "run2.lv.h5", "run3.lv.h5"),
  TR = 2,
  run_length = c(150, 150, 150)
)

## End(Not run)
```

---

fmri_mem_dataset *Create an fMRI Memory Dataset Object*

Description

This function creates an fMRI memory dataset object, which is a list containing information about the scans, mask, TR, number of runs, event table, base path, sampling frame, and censor.

Usage

```
fmri_mem_dataset(
  scans,
  mask,
  TR,
  run_length = sapply(scans, function(x) dim(x)[4]),
  event_table = data.frame(),
  base_path = ".",
  censor = NULL
)
```

Arguments

scans	A list of objects of class NeuroVec from the neuroim2 package.
mask	A binary mask of class NeuroVol from the neuroim2 package indicating the set of voxels to include in analyses.
TR	Repetition time (TR) of the fMRI acquisition.
run_length	A numeric vector specifying the length of each run in the dataset. Default is the length of the scans.
event_table	An optional data frame containing event information. Default is an empty data frame.
base_path	Base directory for relative file names. Absolute paths are used as-is.
censor	An optional numeric vector specifying which time points to censor. Default is NULL.

Value

An fMRI memory dataset object of class c("fmri_mem_dataset", "volumetric_dataset", "fmri_dataset", "list").

Examples

```
# Create a NeuroVec object
d <- c(10, 10, 10, 10)
nvec <- neuroim2::NeuroVec(array(rnorm(prod(d)), d), space = neuroim2::NeuroSpace(d))

# Create a NeuroVol mask
mask <- neuroim2::NeuroVol(array(rnorm(10 * 10 * 10), d[1:3]), space = neuroim2::NeuroSpace(d[1:3]))
mask[mask < .5] <- 0

# Create an fmri_mem_dataset
dset <- fmri_mem_dataset(list(nvec), mask, TR = 2)
```

---

fmri_series *fmri_series: fMRI Time Series Container*

Description

An S3 class representing lazily accessed fMRI time series data. The underlying data is stored in a lazy matrix (typically a delarr object) with rows corresponding to timepoints and columns corresponding to voxels.

Core interface for retrieving voxel time series from fMRI datasets.

Usage

```
fmri_series(
  dataset,
  selector = NULL,
  timepoints = NULL,
  output = c("fmri_series", "DelayedMatrix"),
  event_window = NULL,
  ...
)
```

Arguments

dataset	An fmri_dataset object.
selector	Spatial selector or NULL for all voxels.
timepoints	Optional temporal subset or NULL for all.
output	Return type - "FmriSeries" (default) or "DelayedMatrix".
event_window	Reserved for future use.
...	Additional arguments passed to methods.

Details

An fmri_series object contains:

- data: A lazy matrix with timepoints as rows and voxels as columns
- voxel_info: A data.frame containing spatial metadata for each voxel
- temporal_info: A data.frame containing metadata for each timepoint
- selection_info: A list describing how the data were selected
- dataset_info: A list describing the source dataset and backend

Value

An object of class fmri_series

An fmri_series (with a delarr lazy matrix payload) or a DelayedMatrix when output = "DelayedMatrix".

See Also

as.matrix.fmri_series for converting to standard matrix, as_tibble.fmri_series for converting to tibble format

Examples

```
# Create example fmri_series object
mat <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
backend <- matrix_backend(mat, mask = rep(TRUE, ncol(mat)))
dataset <- fmri_dataset(backend, TR = 1, run_length = rep(25, 4))
fs <- fmri_series(dataset)
fs
```

fmri_series_resolvers *Helpers for fmri_series spatial and temporal selection*

Description

These functions convert user-facing selectors into numeric indices used by the fmri_series implementation. They are not exported to users directly.

fmri_study_dataset *Create an fmri_study_dataset*

Description

High level constructor that combines multiple `fmri_dataset` objects into a single study-level dataset using `study_backend`.

Usage

```
fmri_study_dataset(datasets, subject_ids = NULL)
```

Arguments

`datasets`	A list of `fmri_dataset` objects
`subject_ids`	Optional vector of subject identifiers

Value

An object of class `fmri_study_dataset`

`fmri_zarr_dataset` *Create an fMRI Dataset from Zarr Arrays*

Description

Creates an fMRI dataset object from Zarr array files. Zarr is a cloud-native array format that supports chunked, compressed storage and is ideal for large neuroimaging datasets.

Usage

```
fmri_zarr_dataset(
  zarr_source,
  TR,
  run_length,
  event_table = data.frame(),
  censor = NULL,
  preload = FALSE
)
```

Arguments

`zarr_source`	Path to Zarr store (directory or URL)
`TR`	The repetition time in seconds
`run_length`	Vector of integers indicating the number of scans in each run
`event_table`	Optional data.frame containing event onsets and experimental variables
`censor`	Optional binary vector indicating which scans to remove
`preload`	Whether to load all data into memory (default: FALSE)

Details

The Zarr backend expects data organized as a 4D array with dimensions (x, y, z, time). The data is accessed lazily by default, loading only the requested chunks into memory.

Zarr stores should contain a single 4D array. For mask data, provide it separately through the fmri_dataset interface if needed.

Value

An fMRI dataset object of class c("fmri_file_dataset", "volumetric_dataset", "fmri_dataset", "list")

Experimental

This function uses the CRAN zarr package which is relatively new (v0.1.1, Dec 2025). It supports Zarr v3 format only - Zarr v2 stores cannot be read. Please report any issues to help improve the package.

See Also

zarr_backend, fmri_dataset

Examples

```
## Not run:
# Local Zarr store
dataset <- fmri_zarr_dataset(
  "path/to/data.zarr",
  TR = 2,
  run_length = c(150, 150, 150)
)

# Remote store
dataset <- fmri_zarr_dataset(
  "https://example.com/subject01.zarr",
  TR = 1.5,
  run_length = 300
)

# Preload small dataset into memory
dataset <- fmri_zarr_dataset(
  "small_data.zarr",
  TR = 2,
  run_length = 100,
  preload = TRUE
)

## End(Not run)
```

---

generics *Generic Functions for fMRI Dataset Operations*

Description

This file contains all generic function declarations for the refactored fmridataset package. These establish the interface contracts that are implemented by dataset-specific methods in other files.

---

get_backend_registry *Get Registered Backend Information*

Description

Retrieves information about a registered backend or lists all registered backends.

Usage

```
get_backend_registry(name = NULL)
```

Arguments

name	Character string, name of backend to query. If NULL, returns all registrations.

Value

For a specific backend: a list with registration details. For all backends: a named list where each element contains registration details.

Examples

```
# List all registered backends
get_backend_registry()

# Get specific backend info
get_backend_registry("nifti")
```

---

get_component_info *Get Component Information*

Description

Get metadata about the latent components in the dataset.

Usage

```
get_component_info(x, ...)
```

Arguments

x	A latent_dataset object
...	Additional arguments

Value

A list containing component metadata

See Also

Other latent_data: get_latent_scores(), get_spatial_loadings(), latent_dataset(), reconstruct_voxels()

get_confounds *Get Confound Regressors from a BIDS H5 Dataset*

Description

Generic function to retrieve confound regressor matrices stored in a BIDS HDF5 dataset.

Usage

```
get_confounds(x, ...)

## S3 method for class 'bids_h5_study_dataset'
get_confounds(x, scan_name = NULL, subject = NULL, task = NULL, ...)
```

Arguments

x	A BIDS H5 study dataset object
...	Additional arguments passed to methods
scan_name	Character. Scan name key (exact match), or NULL.
subject	Character. Subject ID filter, or NULL.
task	Character. Task filter, or NULL.

Value

A tibble (single scan) or named list of tibbles (multiple scans)

get_data *Get Data from fMRI Dataset Objects*

Description

Generic function to extract data from various fMRI dataset types. Returns the underlying data in its native format (NeuroVec, matrix, etc.).

Usage

```
get_data(x, ...)
```

Arguments

x	An fMRI dataset object (e.g., fmri_dataset, matrix_dataset)
...	Additional arguments passed to methods

Details

This function extracts the raw data from dataset objects, preserving the original data type. For NeuroVec-based datasets, returns a NeuroVec object. For matrix-based datasets, returns a matrix.

Value

Dataset-specific data object:

- For `fmri_dataset`: Returns the underlying NeuroVec or matrix
- For `matrix_dataset`: Returns the data matrix

See Also

`get_data_matrix` for extracting data as a matrix, `get_mask` for extracting the mask

Examples

```
# Create a matrix dataset
mat <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
ds <- matrix_dataset(mat, TR = 2, run_length = 100)

# Extract the data
data <- get_data(ds)
identical(data, mat) # TRUE
```

---

get_data_matrix *Get Data Matrix from fMRI Dataset Objects*

Description

Generic function to extract data as a matrix from various fMRI dataset types. Always returns a matrix with timepoints as rows and voxels as columns.

Usage

```
get_data_matrix(x, ...)
```

Arguments

`x`	An fMRI dataset object (e.g., fmri_dataset, matrix_dataset)
`...`	Additional arguments passed to methods

Details

This function provides a unified interface for accessing fMRI data as a matrix, regardless of the underlying storage format. The returned matrix always has timepoints in rows and voxels in columns, matching the conventional fMRI data organization.

Value

A numeric matrix with dimensions:

- Rows: Number of timepoints
- Columns: Number of voxels (within mask)

See Also

`get_data` for extracting data in native format, `as.matrix_dataset` for converting to matrix dataset

Examples

```
# Create a matrix dataset
mat <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
ds <- matrix_dataset(mat, TR = 2, run_length = 100)

# Extract as matrix
data_mat <- get_data_matrix(ds)
dim(data_mat) # 100 x 50
```

---

get_latent_scores *Get Latent Scores from Dataset*

Description

Extract the latent scores (temporal components) from a latent dataset. This is the primary data access method for latent datasets.

Usage

```
get_latent_scores(x, rows = NULL, cols = NULL, ...)
```

Arguments

`x`	A latent_dataset object
`rows`	Optional row indices (timepoints) to extract
`cols`	Optional column indices (components) to extract
`...`	Additional arguments

Value

Matrix of latent scores (time × components)

See Also

Other latent_data: `get_component_info()`, `get_spatial_loadings()`, `latent_dataset()`, `reconstruct_voxels()`

get_loadings *Get Spatial Loadings from a Latent-Mode BIDS H5 Dataset*

Description

Retrieves the spatial loadings matrix (or matrices) from a latent-mode BIDS HDF5 archive. Only available when `compression_mode = "latent"`.

Usage

```
get_loadings(x, scan_name = NULL, ...)

## S3 method for class 'bids_h5_study_dataset'
get_loadings(x, scan_name = NULL, ...)
```

Arguments

`x`	A `bids_h5_study_dataset` object opened in latent mode.
`scan_name`	Character. Name of a specific scan, or `NULL` to return loadings for all scans as a named list.
`...`	Additional arguments passed to methods.

Value

A numeric matrix `[V, K]` for a single scan, or a named list of such matrices when `scan_name = NULL`.

get_mask *Get Mask from fMRI Dataset Objects*

Description

Generic function to extract masks from various fMRI dataset types. Returns the mask in its appropriate format for the dataset type.

Usage

```
get_mask(x, ...)
```

Arguments

`x`	An fMRI dataset object (e.g., fmri_dataset, matrix_dataset)
`...`	Additional arguments passed to methods

Details

The mask defines which voxels are included in the analysis. Different dataset types may store masks in different formats (logical vectors, NeuroVol objects, etc.). This function provides a unified interface for mask extraction.

Value

Mask object appropriate for the dataset type:

- For `matrix_dataset`: Logical vector
- For `fmri_dataset`: NeuroVol or logical vector

See Also

`get_data` for extracting data, `get_data_matrix` for extracting data as matrix

Examples

```
# Create a matrix dataset (matrix_dataset creates default mask internally)
mat <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
ds <- matrix_dataset(mat, TR = 2, run_length = 100)

# Extract the mask (matrix_dataset creates all TRUE mask by default)
extracted_mask <- get_mask(ds)
sum(extracted_mask) # 50 (all TRUE values)
```

`get_run_duration` *Get Run Duration from Sampling Frame*

Description

Calculates the duration of each run in seconds.

Usage

```
get_run_duration(x, ...)

## S3 method for class 'fmri_dataset'
get_run_duration(x, ...)

## S3 method for class 'sampling_frame'
get_run_duration(x, ...)
```

Arguments

`x`	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
`...`	Additional arguments passed to methods

Value

Numeric vector where each element represents the duration of the corresponding run in seconds

See Also

`get_total_duration` for total duration, `get_run_lengths` for run lengths in timepoints

Examples

```
# Create a sampling frame with different run lengths
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120), TR = 2)
get_run_duration(sf) # Returns: c(200, 240) seconds
```

---

get_run_lengths *Get Run Lengths from Sampling Frame*

Description

Extracts the lengths of individual runs/blocks from objects containing temporal structure information.

Usage

```
get_run_lengths(x, ...)

## S3 method for class 'fmri_dataset'
get_run_lengths(x, ...)

## S3 method for class 'sampling_frame'
get_run_lengths(x, ...)
```

Arguments

x	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
...	Additional arguments passed to methods

Details

This function is synonymous with `blocklens` but uses terminology more common in fMRI analysis. Each run represents a continuous acquisition period, and the run length is the number of timepoints (volumes) in that run.

Value

Integer vector where each element represents the number of timepoints in the corresponding run

See Also

blocklens for equivalent function, n_runs for number of runs, n_timepoints for total timepoints

Examples

```
# Create a sampling frame with 3 runs
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120, 110), TR = 2)
get_run_lengths(sf) # Returns: c(100, 120, 110)
```

---

get_spatial_loadings *Get Spatial Loadings from Dataset*

Description

Extract the spatial loadings (spatial components) from a latent dataset.

Usage

```
get_spatial_loadings(x, components = NULL, ...)
```

Arguments

x	A latent_dataset object
components	Optional component indices to extract
...	Additional arguments

Value

Matrix or sparse matrix of spatial loadings (voxels × components)

See Also

Other latent_data: get_component_info(), get_latent_scores(), latent_dataset(), reconstruct_voxels()

`get_total_duration` *Get Total Duration from Sampling Frame*

Description

Calculates the total duration of the fMRI acquisition in seconds across all runs.

Usage

```
get_total_duration(x, ...)

## S3 method for class 'fmri_dataset'
get_total_duration(x, ...)

## S3 method for class 'sampling_frame'
get_total_duration(x, ...)
```

Arguments

`x`	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
`...`	Additional arguments passed to methods

Value

Numeric value representing total duration in seconds

See Also

`get_run_duration` for individual run durations, `get_TR` for repetition time

Examples

```
# Create a sampling frame: 220 timepoints with TR = 2 seconds
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120), TR = 2)
get_total_duration(sf) # Returns: 440 seconds
```

get_TR *Get TR (Repetition Time) from Sampling Frame*

Description

Extracts the repetition time (TR) in seconds from objects containing temporal information about fMRI acquisitions.

Usage

```
get_TR(x, ...)

## S3 method for class 'fmri_dataset'
get_TR(x, ...)

## S3 method for class 'sampling_frame'
get_TR(x, ...)
```

Arguments

x	An object containing temporal information (e.g., sampling_frame, fmri_dataset)
...	Additional arguments passed to methods

Details

The TR (repetition time) is the time between successive acquisitions of the same slice in an fMRI scan, typically measured in seconds. This parameter is crucial for temporal analyses and hemodynamic modeling.

Value

Numeric value representing TR in seconds

See Also

`fmrihrf::sampling_frame()` for creating temporal structures, `get_total_duration` for total scan duration

Examples

```
# Create a sampling frame with TR = 2 seconds
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120), TR = 2)
get_TR(sf) # Returns: 2
```

group_map *Map a function over subjects in an fmri_group*

Description

Map a function over subjects in an fmri_group

Usage

```
group_map(
  gd,
  .f,
  ...,
  out = c("list", "bind_rows"),
  order_by = NULL,
  on_error = c("stop", "warn", "skip")
)
```

Arguments

gd	An fmri_group.
.f	A function with signature function(row, ...) where row is a one-row data.frame corresponding to a single subject.
...	Additional arguments passed through to .f.
out	Either "list" (default) or "bind_rows" describing how to collect outputs.
order_by	Optional column name used to define iteration order.
on_error	One of "stop", "warn", or "skip" describing how to handle errors raised by .f.

Value

Either a list (for out = "list") or a bound table (for out = "bind_rows").

group_reduce *Reduce over subjects in a single pass*

Description

Reduce over subjects in a single pass

Usage

```
group_reduce(
  gd,
  .map,
  .reduce,
  .init,
  order_by = NULL,
  on_error = c("stop", "warn", "skip"),
  ...
)
```

Arguments

gd	An fmri_group.
.map	Function applied to each subject row. Should return an object that can be combined by .reduce.
.reduce	Binary function combining the accumulator and the mapped value.
.init	Initial accumulator value.
order_by	Optional ordering column.
on_error	Error handling policy: "stop", "warn", or "skip".
...	Additional arguments passed to .map.

Value

The reduced value after visiting all subjects.

---

index_selector *Index-based Series Selector*

Description

Select voxels by their direct indices in the masked data.

Usage

```
index_selector(indices)
```

Arguments

indices	Integer vector of voxel indices

Value

An object of class index_selector

Examples

```
# Select first 10 voxels
sel <- index_selector(1:10)

# Select specific voxels
sel <- index_selector(c(1, 5, 10, 20))
```

---

is.fmri_series *Check if object is an fmri_series*

Description

Check if object is an fmri_series

Usage

```
is.fmri_series(x)
```

Arguments

x	An object to test

Value

Logical TRUE if x is an fmri_series object

---

is.sampling_frame *Test if Object is a Sampling Frame*

Description

This function tests whether an object is of class 'sampling_frame'.

Usage

```
is.sampling_frame(x)
```

Arguments

x	An object to test

Value

TRUE if x is a sampling_frame object, FALSE otherwise

is_backend_registered *Check if Backend is Registered*

Description

Tests whether a backend type is registered in the system.

Usage

```
is_backend_registered(name)
```

Arguments

name	Character string, name of backend to check

Value

Logical, TRUE if backend is registered

Examples

```
is_backend_registered("nifti") # TRUE (built-in)
is_backend_registered("custom") # FALSE (unless registered)
```

iter_subjects *Iterate subjects one-by-one (streaming)*

Description

Iterate subjects one-by-one (streaming)

Usage

```
iter_subjects(gd, order_by = NULL)
```

Arguments

gd	An fmri_group.
order_by	Optional character scalar giving the column used to order iteration. If supplied, subjects are iterated in ascending order of this column (with NA values placed last).

Value

A list with a single element next that yields a one-row data.frame for each subject when called repeatedly. The dataset column is automatically flattened to the underlying fmri_dataset object.

`latent_dataset` *Latent Dataset Interface*

Description

A specialized dataset interface for working with latent space representations of fMRI data. Unlike traditional fMRI datasets that work with voxel-space data, latent datasets operate on compressed representations using basis functions.

This interface is designed for data that has been decomposed into temporal components (basis functions) and spatial loadings, such as from PCA, ICA, or dictionary learning methods.

Creates a dataset object for working with latent space representations of fMRI data. This is the primary constructor for latent datasets.

Usage

```
latent_dataset(
  source,
  TR,
  run_length,
  event_table = data.frame(),
  base_path = ".",
  censor = NULL,
  preload = FALSE
)
```

Arguments

`source`	Character vector of file paths to LatentNeuroVec HDF5 files (.lv.h5), or a list of LatentNeuroVec objects from the fmristore or fmrilatent packages. File paths require the fmristore package; in-memory objects do not.
`TR`	The repetition time in seconds.
`run_length`	Vector of integers indicating the number of scans in each run.
`event_table`	Optional data.frame containing event onsets and experimental variables.
`base_path`	Base directory for relative file paths.
`censor`	Optional binary vector indicating which scans to remove.
`preload`	Logical indicating whether to preload all data into memory.

Details

Key Differences from Standard Datasets::

- **Data Access**: Returns latent scores (time × components) instead of voxel data
- **Mask**: Represents active components, not spatial voxels
- **Dimensions**: Component space rather than voxel space
- **Reconstruction**: Can optionally reconstruct to voxel space on demand

Data Structure::

Latent representations store data as:

- `basis`: Temporal components (n_timepoints × k_components)
- `loadings`: Spatial components (n_voxels × k_components)
- `offset`: Optional per-voxel offset terms
- Reconstruction: `data = basis %*% t(loadings) + offset`

Value

A `latent_dataset` object with class `c("latent_dataset", "fmri_dataset")`.

See Also

Other latent_data: `get_component_info()`, `get_latent_scores()`, `get_spatial_loadings()`, `reconstruct_voxels()`

Other latent_data: `get_component_info()`, `get_latent_scores()`, `get_spatial_loadings()`, `reconstruct_voxels()`

Examples

```
## Not run:
# From LatentNeuroVec files
dataset <- latent_dataset(
  source = c("run1.lv.h5", "run2.lv.h5"),
  TR = 2,
  run_length = c(100, 100)
)

# From pre-loaded objects
lvec1 <- fmristore::read_vec("run1.lv.h5")
lvec2 <- fmristore::read_vec("run2.lv.h5")
dataset <- latent_dataset(
  source = list(lvec1, lvec2),
  TR = 2,
  run_length = c(100, 100)
)

# Access latent scores
scores <- get_latent_scores(dataset)

# Get component metadata
comp_info <- get_component_info(dataset)

## End(Not run)
```

left_join_subjects *Left join additional subject metadata*

Description

Left join additional subject metadata

Usage

```
left_join_subjects(gd, y, by = NULL, ...)
```

Arguments

gd	An fmri_group.
y	A data frame containing additional subject-level columns.
by	Character vector of join keys (defaults to the group id column).
...	Additional arguments passed to dplyr::left_join() when available.

Value

An fmri_group with metadata from y attached.

list_backend_names *List Registered Backend Names*

Description

Returns a character vector of all registered backend names.

Usage

```
list_backend_names()
```

Value

Character vector of backend names

Examples

```
list_backend_names()
```

mask_selector *Mask-based Series Selector*

Description

Select voxels that are TRUE in a binary mask.

Usage

```
mask_selector(mask)
```

Arguments

mask	A logical vector matching the dataset's mask length, or a 3D logical array

Value

An object of class `mask_selector`

Examples

```
## Not run:
# Using a logical vector
mask_vec <- backend_get_mask(dataset$backend)
sel <- mask_selector(mask_vec > 0.5)

## End(Not run)
```

matrix_dataset *Matrix Dataset Constructor*

Description

This function creates a matrix dataset object, which is a list containing information about the data matrix, TR, number of runs, event table, sampling frame, and mask.

Usage

```
matrix_dataset(datamat, TR, run_length, event_table = data.frame())
```

Arguments

datamat	A matrix where each column is a voxel time-series.
TR	Repetition time (TR) of the fMRI acquisition.
run_length	A numeric vector specifying the length of each run in the dataset.
event_table	An optional data frame containing event information. Default is an empty data frame.

Value

A matrix dataset object of class c("matrix_dataset", "fmri_dataset", "list").

Examples

```
# A matrix with 100 rows and 100 columns (voxels)
X <- matrix(rnorm(100 * 100), 100, 100)
dset <- matrix_dataset(X, TR = 2, run_length = 100)
```

---

mutate_subjects *Mutate subject-level attributes*

Description

Adds or modifies columns on the underlying subjects table. Expressions are evaluated sequentially so newly created columns are available to later expressions.

Usage

```
mutate_subjects(gd, ...)
```

Arguments

gd	An fmri_group.
...	Logical expressions used to filter rows.

Value

An updated fmri_group with modified subject attributes.

---

n_runs *Get Number of Runs from Sampling Frame*

Description

Extracts the total number of runs/blocks from objects containing temporal structure information.

Usage

```
n_runs(x, ...)

## S3 method for class 'fmri_dataset'
n_runs(x, ...)

## S3 method for class 'fmri_study_dataset'
n_runs(x, ...)

## S3 method for class 'sampling_frame'
n_runs(x, ...)
```

Arguments

x	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
...	Additional arguments passed to methods

Value

Integer representing the total number of runs

See Also

get_run_lengths for individual run lengths, n_timepoints for total timepoints

Examples

```
# Create a sampling frame with 3 runs
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120, 110), TR = 2)
n_runs(sf) # Returns: 3
```

---

n_subjects *Number of subjects in a group*

Description

Number of subjects in a group

Usage

```
n_subjects(gd)
```

Arguments

gd	An fmri_group.

Value

Integer number of subjects.

`n_timepoints` *Get Number of Timepoints from Sampling Frame*

Description

Extracts the total number of timepoints (volumes) across all runs from objects containing temporal structure information.

Usage

```
n_timepoints(x, ...)

## S3 method for class 'fmri_dataset'
n_timepoints(x, ...)

## S3 method for class 'sampling_frame'
n_timepoints(x, ...)
```

Arguments

`x`	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
`...`	Additional arguments passed to methods

Value

Integer representing the total number of timepoints

See Also

`n_runs` for number of runs, `get_run_lengths` for individual run lengths

Examples

```
# Create a sampling frame with 3 runs
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120, 110), TR = 2)
n_timepoints(sf) # Returns: 330 (sum of run lengths)
```

ncol.fmri_series *Number of columns in fmri_series*

Description

Number of columns in fmri_series

Usage

```
## S3 method for class 'fmri_series'
ncol(x)
```

Arguments

x	An fmri_series object

Value

Number of voxels

nrow.fmri_series *Number of rows in fmri_series*

Description

Number of rows in fmri_series

Usage

```
## S3 method for class 'fmri_series'
nrow(x)
```

Arguments

x	An fmri_series object

Value

Number of timepoints

parcellation_info *Get Parcellation Information from a BIDS H5 Dataset*

Description

Generic function to retrieve parcellation metadata from a BIDS HDF5 dataset stored in parcellated mode.

Usage

```
parcellation_info(x, ...)

## S3 method for class 'bids_h5_study_dataset'
parcellation_info(x, ...)
```

Arguments

x	A BIDS H5 study dataset object
...	Additional arguments passed to methods

Value

A list with elements: cluster_ids, cluster_map, labels, n_parcels

participants *Get Participant IDs from a Dataset*

Description

Generic function to extract participant/subject identifiers from study-level fMRI dataset objects. Defined here in fmridataset so methods work without requiring bidser to be installed.

Usage

```
participants(x, ...)

## S3 method for class 'bids_h5_study_dataset'
participants(x, ...)
```

Arguments

x	A study dataset object (e.g., bids_h5_study_dataset)
...	Additional arguments passed to methods

Value

Character vector of participant identifiers

print *Print Methods for fmridataset Objects*

Description

Display formatted summaries of fmridataset objects including datasets, chunk iterators, and data chunks.

This function prints a summary of a chunk iterator.

This function prints a summary of a data chunk.

Usage

```
## S3 method for class 'fmri_dataset'
print(x, full = FALSE, ...)

## S3 method for class 'fmri_dataset'
summary(object, ...)

## S3 method for class 'chunkiter'
print(x, ...)

## S3 method for class 'data_chunk'
print(x, ...)

## S3 method for class 'matrix_dataset'
print(x, ...)
```

Arguments

x	A data_chunk object.
full	Logical; if TRUE, print additional details for datasets (default: FALSE)
...	Additional arguments (ignored).
object	An object to summarize (for summary methods)

Value

The object invisibly

Examples

```
# Print dataset summary
# dataset <- fmri_dataset(...)
# print(dataset)
# print(dataset, full = TRUE)  # More details
```

print.backend_registry *Print Backend Registry*

Description

Prints a formatted summary of registered backends.

Usage

```
## S3 method for class 'backend_registry'
print(x, ...)
```

Arguments

x	Result from get_backend_registry()
...	Additional arguments (ignored)

Value

Invisibly returns the input

print.fmri_series *Print Method for fmri_series Objects*

Description

Display a concise summary of an fmri_series object, including dimensions, selector type, backend, and data orientation.

Usage

```
## S3 method for class 'fmri_series'
print(x, ...)
```

Arguments

x	An fmri_series object
...	Additional arguments (unused)

Value

Returns x invisibly. Called for its side effect of printing to the console.

Examples

```
# This method is called automatically when printing
# Create example object (see fmri_series documentation)
# fs <- new_fmri_series(...)
# fs  # Automatically calls print method
```

`print.series_selector` *Print Methods for Series Selectors*

Description

Display formatted summaries of series selector objects.

Usage

```
## S3 method for class 'series_selector'
print(x, ...)
```

Arguments

`x`	A series selector object
`...`	Additional arguments (currently unused)

Value

The object invisibly

Examples

```
# Print different selector types
sel1 <- index_selector(1:10)
print(sel1)

sel2 <- voxel_selector(c(10, 20, 15))
print(sel2)
```

read_fmri_config *read a basic fMRI configuration file*

Description

Reads a fMRI configuration file in YAML or JSON format. This replaces the previous implementation that used source() for security reasons.

Usage

```
read_fmri_config(file_name, base_path = NULL)
```

Arguments

file_name	name of configuration file (YAML or JSON format)
base_path	the file path to be prepended to relative file names

Value

a fmri_config instance

reconstruct_voxels *Reconstruct Voxel-Space Data from a Latent-Mode BIDS H5 Dataset*

Description

Reconstructs full voxel-space time series for a scan by computing basis %*% t(loadings) + offset. Only available when compression_mode = "latent".

Reconstruct the full voxel-space data from the latent representation. This is computationally expensive and should be used sparingly.

Usage

```
reconstruct_voxels(x, scan_name = NULL, rows = NULL, voxels = NULL, ...)

## S3 method for class 'bids_h5_study_dataset'
reconstruct_voxels(x, scan_name, rows = NULL, voxels = NULL, ...)

## S3 method for class 'latent_dataset'
reconstruct_voxels(x, scan_name = NULL, rows = NULL, voxels = NULL, ...)
```

Arguments

x	A bids_h5_study_dataset object opened in latent mode.
scan_name	Character. Name of the scan to reconstruct.
rows	Integer vector of timepoint indices to return, or NULL for all.
voxels	Integer vector of voxel indices to return, or NULL for all.
...	Additional arguments passed to methods.

Value

A numeric matrix [T, V] (or subset thereof).

See Also

Other latent_data: get_component_info(), get_latent_scores(), get_spatial_loadings(), latent_dataset()

---

register_backend *Register a Storage Backend*

Description

Registers a new storage backend type in the global registry. External packages can use this function to add custom backend support.

Usage

```
register_backend(
  name,
  factory,
  description = NULL,
  validate_function = NULL,
  overwrite = FALSE
)
```

Arguments

name	Character string, unique name for the backend type
factory	Function that creates backend instances, must accept ... arguments
description	Optional character string describing the backend
validate_function	Optional function to validate backend instances beyond the standard contract
overwrite	Logical, whether to overwrite existing registration (default: FALSE)

Details

The factory function should:

- Accept all necessary parameters to create a backend instance
- Return an object that inherits from "storage_backend"
- Implement all required storage backend methods

The validate_function should:

- Accept a backend object as first argument
- Return TRUE if valid, or throw informative error if invalid
- Perform any backend-specific validation beyond the standard contract

Value

Invisibly returns TRUE on successful registration

Examples

```
## Not run:
# Register a custom backend
my_backend_factory <- function(source, ...) {
  # Create and return backend instance
  backend <- list(source = source, ...)
  class(backend) <- c("my_backend", "storage_backend")
  backend
}

register_backend(
  name = "my_backend",
  factory = my_backend_factory,
  description = "Custom backend for my data format"
)

## End(Not run)
```

---

resolve_indices *Resolve Indices from Series Selector*

Description

Converts a series selector specification into actual voxel indices within the dataset mask.

Usage

```
resolve_indices(selector, dataset, ...)
```

Arguments

selector	A series selector object (e.g., index_selector, voxel_selector)
dataset	An fMRI dataset object providing spatial context
...	Additional arguments passed to methods

Details

Series selectors provide various ways to specify spatial subsets of fMRI data. This generic function resolves these specifications into actual indices that can be used to extract data. Different selector types support different selection methods:

- index_selector: Direct indices into masked data
- voxel_selector: 3D coordinates
- roi_selector: Region of interest masks
- sphere_selector: Spherical regions

Value

Integer vector of indices into the masked data

See Also

series_selector for selector types, fmri_series for using selectors to extract data

Examples

```
# Example with index selector
sel <- index_selector(1:10)
# indices <- resolve_indices(sel, dataset)

# Example with voxel coordinates
sel <- voxel_selector(cbind(x = 10, y = 20, z = 15))
# indices <- resolve_indices(sel, dataset)
```

---

roi_selector *ROI-based Series Selector*

Description

Select voxels within a region of interest (ROI) volume or mask.

Usage

```
roi_selector(roi)
```

Arguments

roi	A 3D array, ROIVol, LogicalNeuroVol, or similar mask object

Value

An object of class roi_selector

Examples

```
## Not run:
# Using a binary mask
mask <- array(FALSE, dim = c(64, 64, 30))
mask[30:40, 30:40, 15:20] <- TRUE
sel <- roi_selector(mask)

## End(Not run)
```

---

sample_subjects *Sample subjects from an fmri_group*

Description

Sample subjects from an fmri_group

Usage

```
sample_subjects(gd, n, replace = FALSE, strata = NULL)
```

Arguments

gd	An fmri_group.
n	Number of subjects to sample. When strata is supplied and n has length 1, the same number is drawn from each stratum. Provide a named vector to request different counts per stratum.
replace	Logical indicating whether to sample with replacement.
strata	Optional column name used to stratify the sampling.

Value

A sampled fmri_group.

`samples` *Get Sample Indices from Sampling Frame*

Description

Generates a vector of timepoint indices, typically used for time series analysis or indexing operations.

Usage

```
samples(x, ...)

## S3 method for class 'fmri_dataset'
samples(x, ...)

## S3 method for class 'sampling_frame'
samples(x, ...)
```

Arguments

`x`	An object containing temporal structure (e.g., sampling_frame, fmri_dataset)
`...`	Additional arguments passed to methods

Value

Integer vector from 1 to the total number of timepoints

See Also

`n_timepoints` for total number of samples, `blockids` for run membership

Examples

```
# Create a sampling frame
sf <- fmrihrf::sampling_frame(blocklens = c(100, 120), TR = 2)
s <- samples(sf)
length(s) # 220
range(s) # c(1, 220)
```

scan_manifest *Get Scan Manifest from a BIDS H5 Dataset*

Description

Generic function to extract the per-scan metadata table from a BIDS HDF5 study dataset.

Usage

```
scan_manifest(x, ...)

## S3 method for class 'bids_h5_study_dataset'
scan_manifest(x, ...)
```

Arguments

x	A BIDS H5 study dataset object
...	Additional arguments passed to methods

Value

A tibble with columns: scan_name, subject, task, session, run, n_time, has_events, has_confounds

series *Deprecated alias for* fmri_series

Description

series() forwards to fmri_series() for backward compatibility. A deprecation warning is emitted once per session.

Usage

```
series(
  dataset,
  selector = NULL,
  timepoints = NULL,
  output = c("fmri_series", "DelayedMatrix"),
  event_window = NULL,
  ...
)
```

Arguments

dataset	An fmri_dataset object.
selector	Spatial selector or NULL for all voxels.
timepoints	Optional temporal subset or NULL for all.
output	Return type - "FmriSeries" (default) or "DelayedMatrix".
event_window	Reserved for future use.
...	Additional arguments passed to methods.

Value

See fmri_series()

---

series_selector *Series Selector Classes for fMRI Data*

Description

A family of S3 classes for specifying spatial selections in fMRI datasets. These selectors provide a type-safe, explicit interface for selecting voxels in fmri_series() and related functions.

---

sessions *Get Session Names from a Dataset*

Description

Generic function to extract session names from study-level fMRI dataset objects. Defined here in fmridataset so methods work without requiring bidser to be installed.

Usage

```
sessions(x, ...)

## S3 method for class 'bids_h5_study_dataset'
sessions(x, ...)
```

Arguments

x	A study dataset object (e.g., bids_h5_study_dataset)
...	Additional arguments passed to methods

Value

Character vector of session names, or NULL if no sessions

sphere_selector *Spherical ROI Series Selector*

Description

Select voxels within a spherical region.

Usage

```
sphere_selector(center, radius)
```

Arguments

center	Numeric vector of length 3 (x, y, z) specifying sphere center
radius	Numeric radius in voxel units

Value

An object of class sphere_selector

Examples

```
# Select 10-voxel radius sphere around voxel (30, 30, 20)
sel <- sphere_selector(center = c(30, 30, 20), radius = 10)
```

stream_subjects *Stream subjects with optional ordering*

Description

Stream subjects with optional ordering

Usage

```
stream_subjects(gd, prefetch = 1L, order_by = NULL)
```

Arguments

gd	An fmri_group.
prefetch	Number of subjects to prefetch. Currently only 1L is supported; higher values are accepted for future compatibility but do not change behaviour.
order_by	Optional column name used to order subjects.

Value

An iterator identical to iter_subjects().

study_backend *Study Backend*

Description

Composite backend that lazily combines multiple subject-level backends.

Usage

```
study_backend(
  backends,
  subject_ids = NULL,
  strict = getOption("fmridataset.mask_check", "identical")
)
```

Arguments

backends	list of storage_backend objects
subject_ids	vector of subject identifiers matching backends
strict	mask validation mode. "identical" or "intersect"

Value

A study_backend object

study_to_group *Convert a BIDS H5 Study Dataset to an fmri_group*

Description

Converts a bids_h5_study_dataset (or any fmri_study_dataset) to an fmri_group object with one row per subject. Use this when you need per-subject group operations via group_map().

Usage

```
study_to_group(x, ...)

## S3 method for class 'bids_h5_study_dataset'
study_to_group(x, ...)

## S3 method for class 'fmri_study_dataset'
study_to_group(x, ...)
```

Arguments

x	A bids_h5_study_dataset or fmri_study_dataset.
...	Currently unused.

Value

An fmri_group with columns subject_id and dataset.

---

subject_ids *Get Subject IDs from Multi-Subject Dataset*

Description

Generic function to extract subject identifiers from multi-subject fMRI dataset objects.

Usage

```
subject_ids(x, ...)

## S3 method for class 'fmri_study_dataset'
subject_ids(x, ...)
```

Arguments

x	A multi-subject dataset object (e.g., fmri_study_dataset)
...	Additional arguments passed to methods

Details

Multi-subject datasets contain data from multiple participants. This function extracts the subject identifiers associated with each dataset. The order of subject IDs corresponds to the order of datasets.

Value

Character vector of subject identifiers

subjects *Access the subjects tibble stored inside an fmri_group*

Description

Access the subjects tibble stored inside an fmri_group

Usage

```
subjects(x)

subjects(x) <- value
```

Arguments

x	An fmri_group.
value	A replacement table containing the dataset column used by the group.

Value

The underlying data.frame with one row per subject.

An updated fmri_group.

subset_bids_h5 *Subset a BIDS H5 Study Dataset*

Description

Filters a bids_h5_study_dataset by task, subject, session, and/or run using standard (non-NSE) evaluation. Returns a new bids_h5_study_dataset built from the matching scans, sharing the same underlying HDF5 file handle.

Usage

```
subset_bids_h5(x, task = NULL, subject = NULL, session = NULL, run = NULL)
```

Arguments

x	A bids_h5_study_dataset object.
task	Character vector of task names to keep, or NULL for all.
subject	Character vector of subject IDs to keep, or NULL for all.
session	Character vector of session names to keep, or NULL for all.
run	Character vector of BIDS run labels to keep, or NULL for all.

Value

A new bids_h5_study_dataset containing only the matching scans.

tasks *Get Task Names from a Dataset*

Description

Generic function to extract task names from study-level fMRI dataset objects. Defined here in fmridataset so methods work without requiring bidser to be installed.

Usage

```
tasks(x, ...)

## S3 method for class 'bids_h5_study_dataset'
tasks(x, ...)
```

Arguments

x	A study dataset object (e.g., bids_h5_study_dataset)
...	Additional arguments passed to methods

Value

Character vector of task names present in the dataset

unregister_backend *Unregister a Backend*

Description

Removes a backend from the registry. Use with caution as this may break code that depends on the backend.

Usage

```
unregister_backend(name)
```

Arguments

name	Character string, name of backend to remove

Value

Invisibly returns TRUE if backend was removed, FALSE if it wasn't registered

Examples

```
## Not run:
# Remove a custom backend
unregister_backend("my_custom_backend")

## End(Not run)
```

`validate_fmri_group` *Validate an fmri_group object*

Description

Validate an fmri_group object

Usage

```
validate_fmri_group(x)
```

Arguments

`x`	Object to validate.

Value

The validated `fmri_group` (invisibly).

`voxel_selector` *Voxel Coordinate Series Selector*

Description

Select voxels by their 3D coordinates in the image space.

Usage

```
voxel_selector(coords)
```

Arguments

`coords`	Matrix with 3 columns (x, y, z) or vector of length 3

Value

An object of class `voxel_selector`

Examples

```
# Select single voxel
sel <- voxel_selector(c(10, 20, 15))

# Select multiple voxels
coords <- cbind(x = c(10, 20), y = c(20, 30), z = c(15, 15))
sel <- voxel_selector(coords)
```

---

with_rowData *Attach rowData metadata to a lazy matrix*

Description

Helper for reattaching metadata after DelayedMatrixStats operations.

Usage

```
with_rowData(x, rowData)
```

Arguments

x	A lazy matrix or matrix-like object
rowData	A data.frame of row-wise metadata

Value

x with rowData attribute set

---

write_fmri_config *Write fMRI configuration file*

Description

Writes a fMRI configuration to a YAML file for easy editing and sharing.

Usage

```
write_fmri_config(config, file_name)
```

Arguments

config	A fmri_config object or list with configuration parameters
file_name	Output file name (should end in .yaml or .yml)

Index