Package ‘causalimages’

December 25, 2025

Title Causal Inference with Earth Observation, Bio-medical,
and Social Science Images

Version 0.1

Description Provides a system for performing causal inference with earth observation,
bio-medical, and social science images and image sequences (videos). The package
uses a 'JAX' backend for GPU/TPU acceleration. Key functionalities include building
conda-based backends (e.g., via 'BuildBackend'), implementing image-based confounder
and heterogeneity analyses (e.g., 'AnalyzeImageConfounding', 'AnalyzeImageHeterogeneity'),
and writing/reading large image corpora as '.tfrecord' files for use in training
(via 'WriteTfRecord' and 'GetElementFromTfRecordAtIndices'). This allows researchers
to scale causal inference to modern large-scale imagery data, bridging R with
hardware-accelerated Python libraries. The package is partly based on Jerzak
and Daoud (2023) <doi:10.48550/arXiv.2310.00233>.

URL `https://github.com/cjerzak/causalimages-software`

BugReports `https://github.com/cjerzak/causalimages-software/issues`

Depends R (>= 3.3.3)

License GPL-3

Encoding UTF-8

LazyData false

Imports reticulate,
geosphere,
raster,
rrapply,
glmnet,
sf,
data.table,
pROC

Suggests abind,
animation,
gtools,
knitr,
PRROC,
rmarkdown,
tensorflow,
testthat (>= 3.0.0),
viridis

VignetteBuilder knitr

RoxygenNote 7.3.3

RemoteType local

RemotePkgRef local::~/Documents/causalimages-software/causalimages

RemoteUrl /Users/cjerzak/Documents/causalimages-software/causalimages

Contents

`AnalyzeImageConfounding` *Perform causal estimation under image confounding*

Description

Perform causal estimation under image confounding

Usage

```
AnalyzeImageConfounding(
  obsW,
  obsY,
  X = NULL,
  file = NULL,
  imageKeysOfUnits = NULL,
  fileTransport = NULL,
  imageKeysOfUnitsTransport = NULL,
  nBoot = 10L,
  inputAvePoolingSize = 1L,
  useTrainingPertubations = T,
  useScalePertubations = F,
  kFolds = 2L,
  augmented = FALSE,
  orthogonalize = F,
  transportabilityMat = NULL,
  latTransport = NULL,
  longTransport = NULL,
  lat = NULL,
  long = NULL,
  conda_env = "CausalImagesEnv",
  conda_env_required = T,
  Sys.setenv_text = NULL,
  figuresTag = NULL,
  figuresPath = "./",
  plotBands = 1L,
  plotResults = T,
  XCrossModal = T,
  XForceModal = F,
  optimizeImageRep = T,
  nonLinearScaler = NULL,
  nWidth_ImageRep = 64L,
  nDepth_ImageRep = 1L,
  kernelSize = 5L,
  nWidth_Dense = 64L,
  nDepth_Dense = 1L,
  imageModelClass = "VisionTransformer",
  pretrainedModel = NULL,
  strides = 2L,
  nDepth_TemporalRep = 3L,
  patchEmbedDim = 16L,
  dropoutRate = 0.1,
  droppathRate = 0.1,
  batchSize = 16L,
  nSGD = 400L,
  earlyStopThreshold = NULL,
  testFrac = 0.05,
  TfRecords_BufferScaler = 4L,
  learningRateMax = 0.001,
  TFRecordControl = NULL,
  dataType = "image",
  temporalAggregation = "transformer",
  image_dtype = "float16",
  atError = "stop",
  seed = NULL
)
```

Arguments

obsW	A numeric vector where `0`'s correspond to control units and `1`'s to treated units.
obsY	A numeric vector containing observed outcomes.
X	An optional numeric matrix containing tabular information used if `orthogonalize = T`. `X` is normalized internally and salience maps with respect to `X` are trans-

	formed back to the original scale.
`file`	Path to a tfrecord file generated by `WriteTfRecord`.
`imageKeysOfUnits`	A vector of length `length(obsY)` specifying the unique image ID associated with each unit. Samples of `imageKeysOfUnits` are fed into the package to call images into memory.
`fileTransport`	Path to a tfrecord file for transportability analysis (out-of-sample prediction).
`imageKeysOfUnitsTransport`	A vector of image keys for transportability analysis units.
`nBoot`	Number of bootstrap iterations for uncertainty estimation.
`inputAvePoolingSize`	Integer specifying average pooling size for downshifting image resolution. Default = `1L` (no downshift).
`useTrainingPertubations`	Boolean specifying whether to randomly perturb the image axes during training to reduce overfitting.
`useScalePertubations`	Boolean specifying whether to use scale perturbations during training. Default = `FALSE`.
`kFolds`	Integer specifying the number of cross-validation folds. Default = `2L`.
`augmented`	Boolean specifying whether to use augmented inverse probability weighting. Default = `FALSE`.
`orthogonalize`	Boolean specifying whether to orthogonalize outcomes with respect to tabular covariates `X`. Default = `FALSE`.
`transportabilityMat`	Optional matrix with a column named `imageKeysOfUnits` specifying keys to be used by the package for generating treatment effect predictions for out-of-sample points.
`latTransport`	Optional vector of latitudes for transportability analysis units.
`longTransport`	Optional vector of longitudes for transportability analysis units.
`long, lat`	Optional vectors specifying longitude and latitude coordinates for units. Used only for describing highest and lowest probability neighborhood units if specified.
`conda_env`	A conda environment where computational environment lives, usually created via `causalimages::BuildBackend()`. Default = `"CausalImagesEnv"`.
`conda_env_required`	A Boolean stating whether use of the specified conda environment is required.
`Sys.setenv_text`	Optional string for setting environment variables before Python initialization.
`figuresTag`	A string specifying an identifier that is appended to all figure names.
`figuresPath`	A string specifying file path for saved figures made in the analysis.
`plotBands`	An integer or vector specifying which band position (from the image representation) should be plotted in the visual results. If a vector, `plotBands` should have 3 (and only 3) dimensions (corresponding to the 3 dimensions to be used in RGB plotting).
`plotResults`	(default = `T`) Should analysis results be plotted?

`XCrossModal`	Boolean specifying whether to use cross-modal learning with tabular data. Default = `TRUE`.
`XForceModal`	Boolean specifying whether to force modal learning. Default = `FALSE`.
`optimizeImageRep`	Boolean specifying whether to optimize over the image model representation (or only over downstream parameters).
`nonLinearScaler`	Optional string specifying non-linear scaling function for outputs.
`nWidth_ImageRep`	Integer specifying width of image model representation.
`nDepth_ImageRep`	Integer specifying depth of image model representation.
`kernelSize`	Dimensions used in spatial convolutions.
`nWidth_Dense`	Integer specifying width of image model representation.
`nDepth_Dense`	Integer specifying depth of dense model representation.
`imageModelClass`	String specifying the image model architecture. Options include `"VisionTransformer"` (default) or `"CNN"`.
`pretrainedModel`	Optional string specifying a pretrained model to use. Options include `"vit-base"`, `"clip-rsicd"`, or HuggingFace model names with `"transformers-"` prefix.
`strides`	(default = `2L`) Integer specifying the strides used in the convolutional layers.
`nDepth_TemporalRep`	Integer specifying depth of temporal representation model. Default = `3L`.
`patchEmbedDim`	Integer specifying patch embedding dimension for Vision Transformer. Default = `16L`.
`dropoutRate`	Dropout rate used in training to prevent overfitting (`dropoutRate = 0` corresponds to no dropout).
`droppathRate`	Droppath rate used in training to prevent overfitting (`droppathRate = 0` corresponds to no droppath).
`batchSize`	Batch size used in SGD optimization. Default = `50L`.
`nSGD`	Number of stochastic gradient descent (SGD) iterations. Default = `400L`
`earlyStopThreshold`	Optional numeric threshold for early stopping based on validation loss.
`testFrac`	Default = `0.1`. Fraction of observations held out as a test set to evaluate out-of-sample loss values.
`TfRecords_BufferScaler`	The buffer size used in `tfrecords` mode is `batchSize*TfRecords_BufferScaler`. Lower `TfRecords_BufferScaler` towards 1 if out-of-memory problems.
`learningRateMax`	Maximum learning rate for the optimizer. Default = `0.001`.
`TFRecordControl`	Optional list for advanced TFRecord configuration.
`dataType`	(default = `"image"`) String specifying whether to assume `"image"` or `"video"` data types.

`temporalAggregation`	String specifying how to aggregate embeddings across time periods for video/image sequence data. Options are `"transformer"` (default) which uses a temporal transformer with attention pooling, or `"concatenate"` which simply concatenates the frame-level embeddings.
`image_dtype`	String specifying image data type. Options are `"float16"` (default), `"bfloat16"`, or `"float32"`.
`atError`	String specifying behavior on error. Options are `"stop"` (default) or `"debug"`.
`seed`	Optional integer for reproducibility.

Value

Returns a list consisting of

- `ATE_est` ATE estimate.
- `ATE_se` Standard error estimate for the ATE.
- `plotResults` If set to `TRUE`, causal salience plots are saved to disk, characterizing the image confounding structure. See references for details.

References

- Connor T. Jerzak, Fredrik Johansson, Adel Daoud. Integrating Earth Observation Data into Causal Inference: Challenges and Opportunities. *ArXiv Preprint*, 2023.

Examples

```
# For a tutorial, see
# github.com/cjerzak/causalimages-software/
```

`AnalyzeImageHeterogeneity` *Decompose treatment effect heterogeneity by image or image sequence*

Description

Implements the image heterogeneity decomposition analysis of Jerzak, Johansson, and Daoud (2023). Users input in treatment and outcome data, along with a function specifying how to load in images using keys referenced to each unit (since loading in all image data will usually not be possible due to memory limitations). This function by default performs estimation, constructs salience maps, and can optionally perform estimation for new areas outside the original study sites in a transportability analysis.

Usage

```
AnalyzeImageHeterogeneity(
  obsW,
  obsY,
  X = NULL,
  orthogonalize = F,
  imageKeysOfUnits = 1:length(obsY),
  kClust_est = 2,
  file = NULL,
  transportabilityMat = NULL,
  lat = NULL,
  long = NULL,
  conda_env = "CausalImagesEnv",
  conda_env_required = T,
  figuresTag = "",
  figuresPath = "./",
  plotBands = 1L,
  heterogeneityModelType = "variational_minimal",
  plotResults = F,
  optimizeImageRep = T,
  nWidth_ImageRep = 64L,
  nDepth_ImageRep = 1L,
  nWidth_Dense = 64L,
  nDepth_Dense = 1L,
  nDepth_TemporalRep = 1L,
  useTrainingPertubations = T,
  strides = 2L,
  nonLinearScaler = NULL,
  pretrainedModel = NULL,
  testFrac = 0.1,
  kernelSize = 5L,
  learningRateMax = 0.001,
  TFRecordControl = NULL,
  patchEmbedDim = 16L,
  nSGD = 500L,
  batchSize = 16L,
  seed = NULL,
  Sys.setenv_text = NULL,
  imageModelClass = "VisionTransformer",
  nMonte_predictive = 10L,
  nMonte_salience = 10L,
  nMonte_variational = 2L,
  TfRecords_BufferScaler = 4L,
  temperature = 1,
  inputAvePoolingSize = 1L,
  dataType = "image",
  temporalAggregation = "transformer"
)
```

Arguments

`obsW`	A numeric vector where `0`'s correspond to control units and `1`'s to treated units.
`obsY`	A numeric vector containing observed outcomes.
`X`	Optional numeric matrix containing tabular information used if `orthogonalize = T`.
`orthogonalize`	A Boolean specifying whether to perform the image decomposition after orthogonalizing with respect to tabular covariates specified in `X`.
`imageKeysOfUnits`	A vector of length `length(obsY)` specifying the unique image ID associated

	with each unit. Samples of `imageKeysOfUnits` are fed into the package to call images into memory.
`kClust_est`	Integer specifying the number of clusters used in estimation. Default is `2L`.
`file`	Path to a tfrecord file generated by `WriteTfRecord`.
`transportabilityMat`	An optional matrix with a column named `key` specifying keys to be used for generating treatment effect predictions for out-of-sample points in earth observation data settings.
`long`, `lat`	Optional vectors specifying longitude and latitude coordinates for units. Used only for describing highest and lowest probability neighborhood units if specified.
`conda_env`	A conda environment where computational environment lives, usually created via `causalimages::BuildBackend()`. Default = `"CausalImagesEnv"`.
`conda_env_required`	A Boolean stating whether use of the specified conda environment is required.
`figuresTag`	A string specifying an identifier that is appended to all figure names.
`figuresPath`	A string specifying file path for saved figures made in the analysis.
`plotBands`	An integer or vector specifying which band position (from the acquired image representation) should be plotted in the visual results. If a vector, `plotBands` should have 3 (and only 3) dimensions (corresponding to the 3 dimensions to be used in RGB plotting).
`heterogeneityModelType`	String specifying the heterogeneity model type. Options include `"variational_minimal"` (default).
`plotResults`	Should analysis results be plotted?
`optimizeImageRep`	Boolean specifying whether to optimize over the image model representation (or only over downstream parameters).
`nWidth_ImageRep`	Integer specifying width of image model representation.
`nDepth_ImageRep`	Integer specifying depth of image model representation.
`nWidth_Dense`	Integer specifying width of image model representation.
`nDepth_Dense`	Integer specifying depth of dense model representation.
`nDepth_TemporalRep`	Integer specifying depth of temporal representation model for video data. Default = `1L`.
`useTrainingPertubations`	Boolean specifying whether to use image perturbations during training. Default = `TRUE`.
`strides`	Integer specifying the strides used in the convolutional layers.
`nonLinearScaler`	Optional string specifying non-linear scaling function for outputs.
`pretrainedModel`	Optional string specifying a pretrained model to use. Options include `"vit-base"`, `"clip-rsicd"`, or HuggingFace model names with `"transformers-"` prefix.
`testFrac`	Fraction of observations held out as a test set. Default = `0.1`.

`kernelSize`	Dimensions used in spatial convolutions.
`learningRateMax`	Maximum learning rate for the optimizer. Default = `0.001`.
`TFRecordControl`	Optional list for advanced TFRecord configuration.
`patchEmbedDim`	Integer specifying patch embedding dimension for Vision Transformer. Default = `16L`.
`nSGD`	Number of stochastic gradient descent (SGD) iterations.
`batchSize`	Batch size used in SGD optimization.
`seed`	Optional integer for reproducibility.
`Sys.setenv_text`	Optional string for setting environment variables before Python initialization.
`imageModelClass`	String specifying the image model architecture. Options include `"VisionTransformer"` (default) or `"CNN"`.
`nMonte_predictive`	An integer specifying how many Monte Carlo iterations to use in the calculation of posterior means (e.g., mean cluster probabilities).
`nMonte_salience`	An integer specifying how many Monte Carlo iterations to use in the calculation of the salience maps (e.g., image gradients of expected cluster probabilities).
`nMonte_variational`	An integer specifying how many Monte Carlo iterations to use in the calculation of the expected likelihood in each training step.
`TfRecords_BufferScaler`	The buffer size used in `tfrecords` mode is `batchSize*TfRecords_BufferScaler`. Lower `TfRecords_BufferScaler` towards 1 if out-of-memory problems.
`temperature`	Temperature parameter for the relaxed categorical distribution in variational inference. Default = `1`.
`inputAvePoolingSize`	Integer specifying average pooling size for downshifting image resolution. Default = `1L` (no downshift).
`dataType`	String specifying whether to assume `"image"` or `"video"` data types.
`temporalAggregation`	String specifying how to aggregate embeddings across time periods for video/image sequence data. Options are `"transformer"` (default) which uses a temporal transformer with attention pooling, or `"concatenate"` which simply concatenates the frame-level embeddings.

Value

Returns a list consisting of

- `clusterTaus_mean`. Estimated mean treatment effects for each cluster.
- `clusterProbs_mean`. Estimated mean image effect cluster probabilities.
- `clusterTaus_sigma`. Estimated cluster standard deviations.
- `clusterProbs_lowerConf`. Estimated lower confidence for effect cluster probabilities.
- `impliedATE`. Implied ATE.

- `individualTau_est`. Estimated individual-level image-based treatment effects.
- `transportabilityMat`. Transportability matrix with estimated cluster information.
- `plottedCoordinates`. List containing coordinates plotted in salience maps.
- `whichNA_dropped`. A vector containing observations dropped due to missingness.

References

- Connor T. Jerzak, Fredrik Johansson, Adel Daoud. Image-based Treatment Effect Heterogeneity. Forthcoming in *Proceedings of the Second Conference on Causal Learning and Reasoning (CLeaR), Proceedings of Machine Learning Research (PMLR)*, 2023.

Examples

```
# For a tutorial, see
# github.com/cjerzak/causalimages-software/
```

`BuildBackend` *Build the environment for CausalImages models. Builds a conda environment in which jax, tensorflow, tensorflow-probability optax, equinox, and jmp are installed.*

Description

Build the environment for CausalImages models. Builds a conda environment in which jax, tensorflow, tensorflow-probability optax, equinox, and jmp are installed.

Usage

```
BuildBackend(conda_env = "CausalImagesEnv", conda = "auto")
```

Arguments

conda_env	(default = `"CausalImagesEnv"`) Name of the conda environment in which to place the backends.
conda	(default = `auto`) The path to a conda executable. Using `"auto"` allows reticulate to attempt to automatically find an appropriate conda binary.

Value

Builds the computational environment for `causalimages`. This function requires an Internet connection. You may find out a list of conda Python paths via: `system("which python")`

Examples

```
# For a tutorial, see
# github.com/cjerzak/causalimages-software/
```

`CausalImagesTutorialData` *CausalImages Tutorial Data*

Description

A tutorial dataset containing satellite imagery and associated variables for demonstrating causal inference with earth observation data. The data is loaded via ‘data(CausalImagesTutorialData)‘ and provides the following objects.

Format

The dataset contains the following objects:

FullImageArray A 4-dimensional array of satellite images with dimensions (n_images, height, width, channels). Each image is a 35x35 pixel RGB image.

KeysOfImages A character vector of unique identifiers for each image in `FullImageArray`. Used to link images to observations.

KeysOfObservations A character vector of length n_observations specifying which image key corresponds to each observation. Multiple observations may share the same image key.

LongLat A data frame with columns `geo_long` and `geo_lat` containing the longitude and latitude coordinates for each observation.

obsW A binary numeric vector indicating treatment assignment (0 = control, 1 = treated) for each observation.

obsY A numeric vector of observed outcomes for each observation.

X A numeric matrix of tabular covariates for each observation. The first column is typically an intercept and may be dropped in analysis.

Details

This dataset is designed for tutorial purposes to demonstrate the key functionalities of the causal-images package, including:

- Writing image data to TFRecord format via `WriteTfRecord`
- Extracting image representations via `GetImageRepresentations`
- Performing image-based confounding analysis via `AnalyzeImageConfounding`
- Analyzing treatment effect heterogeneity via `AnalyzeImageHeterogeneity`

Source

Simulated data for package tutorials.

Examples

```
# Load the tutorial data
data(CausalImagesTutorialData)

# View dimensions of the image array
dim(FullImageArray)
```

```
# Check the number of observations
length(obsY)

# View treatment distribution
table(obsW)
```

`CheckDataQuality` *Check data quality for causalimages analyses*

Description

Validates input data structures before WriteTfRecord or analysis functions. Catches common errors early with informative messages explaining issues and fixes.

Usage

```
CheckDataQuality(
  obsW = NULL,
  obsY = NULL,
  imageKeysOfUnits = NULL,
  uniqueImageKeys = NULL,
  acquireImageFxn = NULL,
  X = NULL,
  file = NULL,
  dataType = "image",
  batchSize = NULL,
  learningRateMax = NULL,
  nSGD = NULL,
  testFrac = NULL,
  checkContext = "pre_analysis",
  stopOnError = TRUE,
  quiet = FALSE
)
```

Arguments

`obsW`	A numeric vector where 0's correspond to control and 1's to treated units (optional).
`obsY`	A numeric vector containing observed outcomes (optional).
`imageKeysOfUnits`	A vector mapping observations to image keys (optional).
`uniqueImageKeys`	A vector of unique image identifiers for tfrecord writing (optional).
`acquireImageFxn`	A function returning image arrays given keys (optional).
`X`	An optional numeric matrix of tabular covariates (optional).
`file`	Path to a tfrecord file (optional).
`dataType`	String: "image" or "video". Default = "image".

`batchSize`	Integer for batch size validation. Default = NULL.
`learningRateMax`	Numeric for learning rate validation. Default = NULL.
`nSGD`	Integer for SGD iterations validation. Default = NULL.
`testFrac`	Numeric for test fraction validation. Default = NULL.
`checkContext`	String: "pre_write" for WriteTfRecord checks, "pre_analysis" for analysis checks. Default = "pre_analysis".
`stopOnError`	Logical. If TRUE, stops on first error. If FALSE, collects all errors. Default = TRUE.
`quiet`	Logical. If TRUE, suppresses informative messages. Default = FALSE.

Value

Invisibly returns a list with:

- `valid` Logical indicating if all checks passed.
- `errors` Character vector of error messages (empty if valid).
- `warnings` Character vector of warning messages.
- `summary` Named list with validation results by category.

Examples

```
## Not run:
# Check data before writing TfRecord
result <- CheckDataQuality(
  uniqueImageKeys = unique(imageKeys),
  acquireImageFxn = myImageLoader,
  file = "data.tfrecord",
  checkContext = "pre_write"
)

# Check data before analysis
result <- CheckDataQuality(
  obsW = treatment,
  obsY = outcome,
  imageKeysOfUnits = imageKeys,
  file = "data.tfrecord",
  checkContext = "pre_analysis"
)

## End(Not run)
```

`GetAndSaveGeolocatedImages`
Getting and saving geo-located images from a pool of .tif's

Description

A function that finds the image slice associated with the `long` and `lat` values, saves images by band (if `save_as = "csv"`) in save_folder.

Usage

```
GetAndSaveGeolocatedImages(
  long,
  lat,
  keys,
  tif_pool,
  image_pixel_width = 256L,
  save_folder = ".",
  save_as = "csv",
  lyrs = NULL
)
```

Arguments

`long`	Vector of numeric longitudes.
`lat`	Vector of numeric latitudes.
`keys`	The image keys associated with the long/lat coordinates.
`tif_pool`	A character vector specifying the fully qualified path to a corpus of .tif files.
`image_pixel_width`	An even integer specifying the pixel width (and height) of the saved images.
`save_folder`	(default = `"."`) What folder should be used to save the output? Example: `"~/Downloads"`
`save_as`	(default = `".csv"`) What format should the output be saved as? Only one option currently (`.csv`)
`lyrs`	(default = NULL) Integer (vector) specifying the layers to be extracted. Default is for all layers to be extracted.

Value

Finds the image slice associated with the `long` and `lat` values, saves images by band (if `save_as = "csv"`) in save_folder. The save format is: `sprintf("%s/Key%s_BAND%s.csv", save_folder, keys[i], band_)`

Examples

```
# Example use (not run):
#MASTER_IMAGE_POOL_FULL_DIR <- c("./LargeTifs/tif1.tif","./LargeTifs/tif2.tif")
#GetAndSaveGeolocatedImages(
                       #long = GeoKeyMat$geo_long,
                       #lat = GeoKeyMat$geo_lat,
                       #image_pixel_width = 500L,
                       #keys = row.names(GeoKeyMat),
                       #tif_pool = MASTER_IMAGE_POOL_FULL_DIR,
                       #save_folder = "./Data/Uganda2000_processed",
                       #save_as = "csv",
                       #lyrs = NULL)
```

GetElementFromTfRecordAtIndices

Reads unique key indices from a `.tfrecord` file.

Description

Reads unique key indices from a `.tfrecord` file saved via a call to `causalimages::WriteTfRecord`.

Usage

```
GetElementFromTfRecordAtIndices(
  uniqueKeyIndices,
  filename,
  nObs,
  readVideo = F,
  conda_env = NULL,
  conda_env_required = F,
  image_dtype = "float16",
  iterator = NULL,
  return_iterator = F
)
```

Arguments

`uniqueKeyIndices`	(integer vector) Unique image indices to be retrieved from a `.tfrecord`
`filename`	(character string) A character string stating the path to a `.tfrecord`
`nObs`	(integer) Total number of observations in the tfrecord file
`readVideo`	(default = `FALSE`) A Boolean indicating whether to read video/sequence data
`conda_env`	(Default = `NULL`) A `conda` environment where tensorflow v2 lives. Used only if a version of tensorflow is not already active.
`conda_env_required`	(default = `FALSE`) A Boolean stating whether use of the specified conda environment is required.
`image_dtype`	(default = `"float16"`) A string specifying the image data type
`iterator`	(default = `NULL`) An optional iterator object for sequential reading
`return_iterator`	(default = `FALSE`) A Boolean indicating whether to return the iterator for subsequent calls

Value

Returns content from a `.tfrecord` associated with `uniqueKeyIndices`

Examples

```
# Example usage (not run):
#GetElementFromTfRecordAtIndices(
  #uniqueKeyIndices = 1:10,
  #filename = "./NigeriaConfoundApp.tfrecord",
  #nObs = 100)
```

`GetImageRepresentations` *Generates image and video representations useful in earth observation tasks for causal inference.*

Description

Generates image and video representations useful in earth observation tasks for causal inference.

Usage

```
GetImageRepresentations(
  X = NULL,
  imageKeysOfUnits = NULL,
  file = NULL,
  conda_env = "CausalImagesEnv",
  conda_env_required = T,
  returnContents = T,
  getRepresentations = T,
  imageModelClass = "VisionTransformer",
  NORM_MEAN = NULL,
  NORM_SD = NULL,
  Sys.setenv_text = NULL,
  InitImageProcess = NULL,
  pretrainedModel = NULL,
  lat = NULL,
  long = NULL,
  image_dtype = NULL,
  image_dtype_tf = NULL,
  XCrossModal = T,
  XForceModal = F,
  nWidth_ImageRep = 64L,
  nDepth_ImageRep = 1L,
  nDepth_TemporalRep = 1L,
  batchSize = 16L,
  nonLinearScaler = NULL,
  optimizeImageRep = T,
  strides = 1L,
  kernelSize = 3L,
  patchEmbedDim = 16L,
  TfRecords_BufferScaler = 10L,
  dropoutRate = 0,
  droppathRate = 0,
  dataType = "image",
  temporalAggregation = "concatenate",
  bn_momentum = 0.99,
  inputAvePoolingSize = 1L,
  CleanupEnv = FALSE,
  initializingFxns = FALSE,
  seed = NULL
)
```

Arguments

`X`	Optional numeric matrix of tabular covariates for cross-modal learning.
`imageKeysOfUnits`	A vector of length `length(imageKeysOfUnits)` specifying the unique image ID associated with each unit. Samples of `imageKeysOfUnits` are fed into the package to call images into memory.
`file`	Path to a tfrecord file generated by `causalimages::WriteTfRecord`.
`conda_env`	A conda environment where computational environment lives, usually created via `causalimages::BuildBackend()`. Default = `"CausalImagesEnv"`
`conda_env_required`	A Boolean stating whether use of the specified conda environment is required.
`returnContents`	Boolean specifying whether to return internal model contents. Default = `TRUE`.
`getRepresentations`	Boolean specifying whether to compute and return representations. Default = `TRUE`.
`imageModelClass`	String specifying the image model architecture. Options include `"VisionTransformer"` (default) or `"CNN"`.
`NORM_MEAN`	Numeric vector of length 1-3 specifying dataset mean(s) for normalization. Used with pretrained models.
`NORM_SD`	Numeric vector of length 1-3 specifying dataset standard deviation(s) for normalization. Used with pretrained models.
`Sys.setenv_text`	Optional string for setting environment variables before Python initialization.
`InitImageProcess`	(default = `NULL`) Initial image processing function. Usually left `NULL`.
`pretrainedModel`	Optional string specifying a pretrained vision model to use for feature extraction. Built-in options include: • `"vit-base"` - Google's Vision Transformer (ViT-Base, 768-dim embeddings) • `"clip-rsicd"` - CLIP fine-tuned on remote sensing data (512-dim embeddings) • `"clip-rsicd-v0"` - Legacy CLIP-RSICD implementation For generic HuggingFace models, use the `"transformers-"` prefix followed by the model name: • `"transformers-facebook/dinov2-base"` - DINOv2 self-supervised ViT • `"transformers-microsoft/resnet-50"` - ResNet-50

- `"transformers-facebook/convnext-base-224"` - ConvNeXt
- `"transformers-<any-huggingface-model>"` - Any vision model from HuggingFace

The generic handler uses `AutoModel.from_pretrained()` and automatically detects hidden dimensions, input sizes, and handles various output formats. Default is `NULL` (uses custom model architecture).

`lat, long`	Optional vectors specifying latitude and longitude coordinates for spatial context.
`image_dtype`	JAX dtype for image data. Usually set internally.
`image_dtype_tf`	TensorFlow dtype for image data. Usually set internally.
`XCrossModal`	Boolean specifying whether to use cross-modal learning with tabular data. Default = `TRUE`.
`XForceModal`	Boolean specifying whether to force modal learning. Default = `FALSE`.
`nWidth_ImageRep`	Number of embedding features output.
`nDepth_ImageRep`	Integer specifying depth of image representation model. Default = `1L`.
`nDepth_TemporalRep`	Integer specifying depth of temporal representation model for video data. Default = `1L`.
`batchSize`	Integer specifying batch size in obtaining representations.
`nonLinearScaler`	Optional string specifying non-linear scaling function for outputs.
`optimizeImageRep`	Boolean specifying whether to optimize image representation parameters. Default = `TRUE`.
`strides`	Integer specifying the strides used in the convolutional layers.
`kernelSize`	Dimensions used in the convolution kernels.
`patchEmbedDim`	Integer specifying patch embedding dimension for Vision Transformer. Default = `16L`.
`TfRecords_BufferScaler`	The buffer size used in `tfrecords` mode is `batchSize*TfRecords_BufferScaler`. Lower `TfRecords_BufferScaler` towards 1 if out-of-memory problems.
`dropoutRate`	Dropout rate used in training. Default = `0`.
`droppathRate`	Droppath rate used in training. Default = `0`.
`dataType`	String specifying whether to assume `"image"` or `"video"` data types. Default is `"image"`.
`temporalAggregation`	String specifying how to aggregate embeddings across time periods for video/image sequence data. Options are `"transformer"` which uses a temporal transformer with attention pooling, `"concatenate"` which simply concatenates the frame-level embeddings, `"difference"` which computes temporal differences to capture change dynamics (outputs mean embeddings concatenated with mean of first-order differences), or `"variance"` which concatenates temporal mean with temporal variance to capture both typical state and volatility/instability.
`bn_momentum`	Batch normalization momentum. Default = `0.99`.

`inputAvePoolingSize`	Integer specifying average pooling size for downshifting image resolution. Default = `1L`.
`CleanupEnv`	Boolean specifying whether to clean up environment after processing. Default = `FALSE`.
`initializingFxns`	Boolean specifying whether to only initialize functions without computing representations. Default = `FALSE`.
`seed`	Optional integer for reproducibility.

Value

A list containing two items:

- `Representations` (matrix) A matrix containing image/video representations, with rows corresponding to observations.
- `ImageRepArm_OneObs,ImageRepArm_batch_R, ImageRepArm_batch` (functions) Image modeling functions.
- `ImageModel_And_State_And_MPPolicy_List` List containing image model parameters fed into functions.

References

- Rolf, Esther, et al. "A generalizable and accessible approach to machine learning with global satellite imagery." *Nature Communications* 12.1 (2021): 4392.

Examples

```
# For a tutorial, see
# github.com/cjerzak/causalimages-software/
```

`GetMoments` *Get moments for normalization (internal function)*

Description

An internal function for obtaining moments for channel normalization.

Usage

```
GetMoments(iterator, dataType, image_dtype, momentCalIters = 34L)
```

Arguments

`iterator`	An iterator
`dataType`	A string denoting data type
`image_dtype`	A string specifying the image data type (e.g., "float16", "float32")
`momentCalIters`	Number of minibatches with which to estimate moments

Value

Returns mean/sd arrays for normalization.

Examples

```
# (Not run)
# GetMoments(iterator, dataType, image_dtype, momentCalIters = 34L)
```

---

image2 — *Visualizing matrices as heatmaps with correct north-south-east-west orientation*

Description

A function for generating a heatmap representation of a matrix with correct spatial orientation.

Usage

```
image2(
  x,
  xaxt = NULL,
  yaxt = NULL,
  xlab = "",
  ylab = "",
  main = NULL,
  cex.main = NULL,
  col.lab = "black",
  col.main = "black",
  cex.lab = 1.5,
  box = F
)
```

Arguments

`x`	The numeric matrix to be visualized.
`xaxt`	The x-axis tick labels.
`yaxt`	The y-axis tick labels.
`xlab`	The x-axis labels.
`ylab`	The y-axis labels.
`main`	The main figure label.
`cex.main`	The main figure label sizing factor.
`col.lab`	Axis label color.
`col.main`	Main label color.
`cex.lab`	Cex for the labels.
`box`	Draw a box around the image?

Value

Returns a heatmap representation of the matrix, `x`, with correct north/south/east/west orientation.

Examples

```
#set seed
set.seed(1)

#Generate data
x <- matrix(rnorm(50*50), ncol = 50)
diag(x) <- 3

# create plot
image2(x, main = "Example Text", cex.main = 2)
```

LongLat2CRS *Get the spatial point of long/lat coordinates*

Description

Convert longitude and latitude coordinates to a different coordinate reference system (CRS).

Usage

```
LongLat2CRS(long, lat, CRS_ref)
```

Arguments

long	Vector of numeric longitudes.
lat	Vector of numeric latitudes.
CRS_ref	A CRS into which the long-lat point should be projected.

Value

Numeric vector of length two giving the coordinates of the supplied location in the CRS defined by `CRS_ref`.

Examples

```
# (Not run)
#spatialPt <- LongLat2CRS(long = 49.932,
#                  lat = 35.432,
#            CRS_ref = sf::st_crs("+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84"))
```

`message2` *message2 message() with timestamps*

Description

A function that displays a message with date and time.

Usage

```
message2(text, quiet = FALSE)
```

Arguments

`text`	Character string to be displayed as message, with date and time.
`quiet`	Logical. If TRUE, suppresses the message output. Default is FALSE.

Value

Displays message with date and time to stderr.

Examples

```
message2("Hello world")
message2("Process completed", quiet = FALSE)
```

`PredictiveRun` *Perform predictive modeling using images or videos*

Description

Perform predictive modeling using images or videos

Usage

```
PredictiveRun(
  obsY,
  imageKeysOfUnits = NULL,
  file = NULL,
  fileTransport = NULL,
  imageKeysOfUnitsTransport = NULL,
  nBoot = 10L,
  inputAvePoolingSize = 1L,
  useTrainingPertubations = T,
  useScalePertubations = F,
  X = NULL,
  conda_env = "CausalImagesEnv",
  conda_env_required = T,
  Sys.setenv_text = NULL,
  figuresTag = NULL,
  figuresPath = "./",
  plotBands = 1L,
  plotResults = T,
  XCrossModal = T,
  XForceModal = F,
  optimizeImageRep = T,
  nWidth_ImageRep = 64L,
  nDepth_ImageRep = 1L,
  kernelSize = 5L,
  nWidth_Dense = 64L,
  nDepth_Dense = 1L,
  imageModelClass = "VisionTransformer",
  pretrainedModel = NULL,
  strides = 2L,
  nonLinearScaler = NULL,
  nDepth_TemporalRep = 3L,
  patchEmbedDim = 16L,
  dropoutRate = 0.1,
  droppathRate = 0.1,
  batchSize = 16L,
  nSGD = 400L,
  testFrac = 0.05,
  TfRecords_BufferScaler = 4L,
  learningRateMax = 0.001,
  TFRecordControl = NULL,
  dataType = "image",
  temporalAggregation = "transformer",
  image_dtype = "float16",
  atError = "stop",
  seed = NULL,
  modelPath = "./trained_model.eqx",
  metricsPath = "./evaluation_metrics.rds"
)
```

Arguments

`obsY`	A numeric vector containing observed outcomes to predict.
`imageKeysOfUnits`	A vector of length `length(obsY)` specifying the unique image ID associated with each unit. Samples of `imageKeysOfUnits` are fed into the package to call images into memory.
`file`	Path to a tfrecord file generated by `WriteTfRecord`.
`fileTransport`	Path to a tfrecord file for transportability analysis (out-of-sample prediction).
`imageKeysOfUnitsTransport`	A vector of image keys for transportability analysis units.
`nBoot`	Number of bootstrap iterations for uncertainty estimation.
`inputAvePoolingSize`	Integer specifying average pooling size for downshifting image resolution. Default = `1L` (no downshift).
`useTrainingPertubations`	Boolean specifying whether to randomly perturb the image axes during training to reduce overfitting.

`useScalePertubations`	Boolean specifying whether to use scale perturbations during training. Default = `FALSE`.
`X`	An optional numeric matrix containing tabular information. `X` is normalized internally.
`conda_env`	A conda environment where computational environment lives, usually created via `causalimages::BuildBackend()`. Default = `"CausalImagesEnv"`.
`conda_env_required`	A Boolean stating whether use of the specified conda environment is required.
`Sys.setenv_text`	Optional string for setting environment variables before Python initialization.
`figuresTag`	A string specifying an identifier that is appended to all figure names.
`figuresPath`	A string specifying file path for saved figures made in the analysis.
`plotBands`	An integer or vector specifying which band position (from the image representation) should be plotted in the visual results. If a vector, `plotBands` should have 3 (and only 3) dimensions (corresponding to the 3 dimensions to be used in RGB plotting).
`plotResults`	(default = `T`) Should analysis results be plotted?
`XCrossModal`	Boolean specifying whether to use cross-modal learning with tabular data. Default = `TRUE`.
`XForceModal`	Boolean specifying whether to force modal learning. Default = `FALSE`.
`optimizeImageRep`	Boolean specifying whether to optimize over the image model representation (or only over downstream parameters).
`nWidth_ImageRep`	Integer specifying width of image model representation.
`nDepth_ImageRep`	Integer specifying depth of image model representation.
`kernelSize`	Dimensions used in spatial convolutions.
`nWidth_Dense`	Integer specifying width of image model representation.
`nDepth_Dense`	Integer specifying depth of dense model representation.
`imageModelClass`	String specifying the image model architecture. Options include `"VisionTransformer"` (default) or `"CNN"`.
`pretrainedModel`	Optional string specifying a pretrained model to use. Options include `"vit-base"`, `"clip-rsicd"`, or HuggingFace model names with `"transformers-"` prefix.
`strides`	(default = `2L`) Integer specifying the strides used in the convolutional layers.
`nonLinearScaler`	Optional string specifying non-linear scaling function for outputs.
`nDepth_TemporalRep`	Integer specifying depth of temporal representation model. Default = `3L`.
`patchEmbedDim`	Integer specifying patch embedding dimension for Vision Transformer. Default = `16L`.
`dropoutRate`	Dropout rate used in training to prevent overfitting (`dropoutRate = 0` corresponds to no dropout).

`droppathRate`	Droppath rate used in training to prevent overfitting (`droppathRate = 0` corresponds to no droppath).
`batchSize`	Batch size used in SGD optimization. Default = `50L`.
`nSGD`	Number of stochastic gradient descent (SGD) iterations. Default = `400L`
`testFrac`	Default = `0.1`. Fraction of observations held out as a test set to evaluate out-of-sample loss values.
`TfRecords_BufferScaler`	The buffer size used in `tfrecords` mode is `batchSize*TfRecords_BufferScaler`. Lower `TfRecords_BufferScaler` towards 1 if out-of-memory problems.
`learningRateMax`	Maximum learning rate for the optimizer. Default = `0.001`.
`TFRecordControl`	Optional list for advanced TFRecord configuration.
`dataType`	(default = `"image"`) String specifying whether to assume `"image"` or `"video"` data types.
`temporalAggregation`	String specifying how to aggregate embeddings across time periods for video/image sequence data. Options are `"transformer"` (default) which uses a temporal transformer with attention pooling, or `"concatenate"` which simply concatenates the frame-level embeddings.
`image_dtype`	String specifying image data type. Options are `"float16"` (default), `"bfloat16"`, or `"float32"`.
`atError`	String specifying behavior on error. Options are `"stop"` (default) or `"debug"`.
`seed`	Optional integer for reproducibility.
`modelPath`	Path to save the trained model. Default = `"./trained_model.eqx"`.
`metricsPath`	Path to save the evaluation metrics as a RDS file. Default = `"./evaluation_metrics.rds"`.

Value

Returns a list consisting of

- `predictedY` Predicted values for all units.
- `ModelEvaluationMetrics` Rigorous evaluation metrics (e.g., MSE, R2 for continuous; AUC, accuracy for binary).

References

- Connor T. Jerzak, Fredrik Johansson, Adel Daoud. Integrating Earth Observation Data into Causal Inference: Challenges and Opportunities. *ArXiv Preprint*, 2023.

Examples

```
# For a tutorial, see
# github.com/cjerzak/causalimages-software/
```

print2 *print2 print() with timestamps*

Description

A function that prints a string with date and time.

Usage

```
print2(text, quiet = F)
```

Arguments

`text`	Character string to be printed, with date and time.
`quiet`	Logical. If TRUE, suppresses the print output. Default is FALSE.

Value

Prints with date and time.

Examples

```
print2("Hello world")
```

TFRecordManagement *Defines an internal TFRecord management routine (internal function)*

Description

Defines management defined in TFRecordManagement(). Internal function. This function takes no parameters and is called within parent functions to set up TFRecord data pipelines.

Usage

```
TFRecordManagement()
```

Value

Internal function defining a tfrecord management sequence.

`TrainDefine` *Defines an internal training routine (internal function)*

Description

Defines trainers defined in TrainDefine(). Internal function. This function takes no parameters and is called within parent functions to set up the optimizer and training step.

Usage

```
TrainDefine()
```

Value

Internal function defining a training sequence.

`TrainDo` *Runs a training routine (internal function)*

Description

Runs trainers defined in TrainDefine(). Internal function. This function takes no parameters and is called within parent functions to execute the training loop.

Usage

```
TrainDo()
```

Value

Internal function performing model training.

`WriteTfRecord` *Write an image corpus as a .tfrecord file*

Description

Writes an image corpus to a `.tfrecord` file for rapid reading of images into memory for fast ML training. Specifically, this function serializes an image or video corpus into a `.tfrecord` file, enabling efficient data loading for machine learning tasks, particularly for image-based causal inference training. It requires that users define an `acquireImageFxn` function that accepts keys and returns the corresponding image or video as an array of dimensions `(length(keys), nSpatialDim1, nSpatialDim2, nC` for images or `(length(keys), nTimeSteps, nSpatialDim1, nSpatialDim2, nChannels)` for video sequences.

Usage

```
WriteTfRecord(
  file,
  uniqueImageKeys,
  acquireImageFxn,
  writeVideo = F,
  image_dtype = "float16",
  conda_env = "CausalImagesEnv",
  conda_env_required = T,
  Sys.setenv_text = NULL
)
```

Arguments

`file`	A character string naming a file for writing.
`uniqueImageKeys`	A vector specifying the unique image keys of the corpus. A key grabs an image/video array via `acquireImageFxn(key)`.
`acquireImageFxn`	A function whose input is an observation keys and whose output is an array with dimensions `(length(keys), nSpatialDim1, nSpatialDim2, nChannels)` for images and `(length(keys), nTimeSteps, nSpatialDim1, nSpatialDim2, nChannels)` for image sequence data.
`writeVideo`	(default = `FALSE`) Should we assume we're writing image sequence data of form batch by time by height by width by channels?
`image_dtype`	String specifying image data type for storage. Default is `"float16"`.
`conda_env`	(default = `"CausalImagesEnv"`) A conda environment where computational environment lives, usually created via `causalimages::BuildBackend()`
`conda_env_required`	(default = `T`) A Boolean stating whether use of the specified conda environment is required.
`Sys.setenv_text`	Optional string for setting environment variables before Python initialization.

Value

Writes a unique key-referenced `.tfrecord` from an image/video corpus for use in image-based causal inference training.

Examples

```
# Example usage (not run):
#WriteTfRecord(
#  file = "./NigeriaConfoundApp.tfrecord",
#  uniqueImageKeys = 1:n,
#  acquireImageFxn = acquireImageFxn)
```

Index