# Package: ArchR (via r-universe)

February 18, 2025

**Type** Package

**Date** 2025-01-19

**Title** Analyzing single-cell regulatory chromatin in R.

**Version** 1.0.3

**Description** This package is designed to streamline scATAC analyses in R.

**Roxygen** list(markdown = TRUE)

**License** MIT

**LinkingTo** Rcpp, RcppArmadillo

**LazyData** TRUE

**URL** https://www.ArchRProject.com

**RoxygenNote** 7.3.2

**Encoding** UTF-8

**Imports** BiocGenerics, Biostrings, chromVAR, chromVARmotifs, ComplexHeatmap, data.table, devtools, GenomicRanges, ggplot2, ggrepel, gridExtra, gtable, gtools, harmony, magrittr, Matrix, matrixStats, motifmatchr, nabor, plyr, presto, Rcpp (>= 0.12.16), RcppArmadillo, rhdf5, Rsamtools, S4Vectors (>= 0.9.25), Seurat, SeuratObject, sparseMatrixStats, stringr, SummarizedExperiment, uwot

**Suggests** Cairo, DESeq2, edgeR, GenomicFeatures, ggridges, ggseqlogo, hexbin, leiden, limma, monocle3, pdftools, pheatmap, rhandsontable, scran, shiny, shinythemes, slingshot, testthat

**Remotes** GreenleafLab/chromVARmotifs, cole-trapnell-lab/monocle3, immunogenomics/presto

**Collate** 'AllClasses.R' 'AnnotationGenome.R' 'AnnotationPeaks.R' 'ArchRBrowser.R' 'ArchRHeatmap.R' 'ArrowRead.R' 'ArrowUtils.R' 'ArrowWrite.R' 'BulkProjection.R' 'Clustering.R' 'ColorPalettes.R' 'CreateArrow.R' 'DoubletsScores.R' 'Embedding.R' 'FilterCells.R' 'Footprinting.R' 'GRangesUtils.R' 'GgplotUtils.R' 'GlobalDefaults.R' 'GroupCoverages.R'

'HelperUtils.R' 'GroupExport.R' 'Harmony.R' 'HiddenUtils.R' 'Imputation.R' 'InputData.R' 'IntegrativeAnalysis.R' 'IterativeLSI.R' 'LoggerUtils.R' 'MarkerFeatures.R' 'MatrixDeviations.R' 'MatrixFeatures.R' 'MatrixGeneExpression.R' 'MatrixGeneScores.R' 'MatrixTiles.R' 'ModuleScore.R' 'MultiModal.R' 'ProjectMethods.R' 'QualityControl.R' 'RNAIntegration.R' 'RcppExports.R' 'ReproduciblePeakSet.R' 'SparseUtils.R' 'Trajectory.R' 'ValidationUtils.R' 'VisualizeData.R'

**Config/pak/sysreqs** libfontconfig1-dev libfreetype6-dev libfribidi-dev git libglpk-dev make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev perl python3 libx11-dev zlib1g-dev

**Repository** https://tttpob.r-universe.dev

**RemoteUrl** https://github.com/GreenleafLab/ArchR

**RemoteRef** HEAD

**RemoteSha** 6feec354ad6c8052ddbc4626a2ca2d858ed465bf

## Contents

`.DollarNames.ArchRProject`

*Accessing cellColData directly from dollar.sign accessor*

## Description

This function will allow direct access to cellColData with a `$` accessor.

## Usage

```
.DollarNames.ArchRProject(x, pattern = "")
```

`.suppressSpecificWarnings`

*This function allows you to suppress specific warnings and was originally created by Antoine Fabri ("Moody_Mudskipper"): see https://stackoverflow.com/a/55182432/697473 Sometimes R throws warning messages that we don't want to see. The base* `suppressWarnings()` *function permits one to suppress warnings, but it is tricky to selectively suppress only certain warnings on the basis of a regular expression or another condition. This function allows one to do that.*

## Description

This function allows you to suppress specific warnings and was originally created by Antoine Fabri ("Moody_Mudskipper"): see https://stackoverflow.com/a/55182432/697473 Sometimes R throws warning messages that we don't want to see. The base `suppressWarnings()` function permits one to suppress warnings, but it is tricky to selectively suppress only certain warnings on the basis of a regular expression or another condition. This function allows one to do that.

## Usage

```
.suppressSpecificWarnings(.expr, .f, ...)
```

## Arguments

| | |
|---|---|
| `.expr` | Expression to be evaluated. |
| `.f` | String or function. If a string (possibly representing a regular expression), any warning message generated when `.expr` is evaluated will be suppressed if `grepl{}` finds that the string matches the warning message. If a function, the warning message will be passed to the function, and the function must return `TRUE` or `FALSE`. See the examples for details. |

## addArchRAnnotations *Add ArchR annotations to an ArchRProject*

### Description

This function adds information about which peaks in the ArchR database contain input regions to a given ArchRProject. For each peak, a binary value is stored indicating whether each region is observed within the peak region.

### Usage

```
addArchRAnnotations(
  ArchRProj = NULL,
  db = "ArchR",
  collection = "EncodeTFBS",
  name = collection,
  force = FALSE,
  logFile = createLogFile("addArchRAnnotations")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `db` | A string indicating the database or a path to a database to use for peak annotation. Options include ArchR, LOLA, and a valid path to a file of class `ArchRAnno`. |
| `collection` | A string indicating which collection within the database to collect for annotation. For ArchR, options are "ATAC", "EncodeTFBS", "CistromeTFBS", or "Codex". For LOLA, options include "EncodeTFBS" "CistromeTFBS", "CistromeEpigenome", "Codex", or "SheffieldDnase". If supplying a custom `ArchRAnno` file please select a valid collection from within that database. |
| `name` | The name of the `peakAnnotation` object to be stored in the `ArchRProject`. |
| `force` | A boolean value indicating whether to force the `peakAnnotation` object indicated by `name` to be overwritten if it already exists in the given `ArchRProject`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Motif Annotations
proj <- addArchRAnnotations(proj, name = "test")
```

## `addArchRChrPrefix` *Add a globally-applied requirement for chromosome prefix*

### Description

This function will set the default requirement of chromosomes to have a "chr" prefix.

### Usage

```
addArchRChrPrefix(chrPrefix = TRUE)
```

### Arguments

| | |
|---|---|
| `chrPrefix` | A boolean describing the requirement of chromosomes to have a "chr" prefix. |

### Examples

```
# Add ArchR Chr Prefix
addArchRChrPrefix()
```

## `addArchRDebugging` *Set ArchR Debugging*

### Description

This function will set ArchR Debugging which will save an RDS if an error is encountered.

### Usage

```
addArchRDebugging(debug = FALSE)
```

### Arguments

| | |
|---|---|
| `debug` | A boolean describing whether to use logging with ArchR. |

### Examples

```
# Add ArchR Debugging
addArchRDebugging()
```

## `addArchRGenome` *Add a globally defined genome to all ArchR functions.*

### Description

This function will set the genome across all ArchR functions.

### Usage

```
addArchRGenome(genome = NULL, install = TRUE)
```

### Arguments

| | |
|---|---|
| `genome` | A string indicating the default genome to be used for all ArchR functions. Currently supported values include "hg19","hg38","mm9", and "mm10". This value is stored as a global environment variable, not part of the `ArchRProject`. This can be overwritten on a per-function basis using the given function's `geneAnnotation`and `genomeAnnotation` parameter. For something other than one of the currently supported, see `createGeneAnnnotation()` and `createGenomeAnnnotation()`. |
| `install` | A boolean value indicating whether the `BSgenome` object associated with the provided `genome` should be automatically installed if it is not currently installed. This is useful for helping reduce user download requirements. |

### Examples

```
# Add ArchR Genome to use globally
addArchRGenome("hg19test2")
```

## `addArchRH5Level` *Add a globally-applied compression level for h5 files*

### Description

This function will set the default compression level to be used for h5 file execution across all ArchR functions.

### Usage

```
addArchRH5Level(level = 0)
```

### Arguments

| | |
|---|---|
| `level` | The default compression level to be used for h5 file execution across all ArchR functions. |

### Examples

```
# Add ArchR H5 Compression level
addArchRH5Level()
```

---

## addArchRLocking *Add a globally-applied H5 file locking setup*

---

### Description

This function will set the default H5 file locking parameters

### Usage

```
addArchRLocking(locking = FALSE)
```

### Arguments

| | |
|---|---|
| `locking` | The default value for H5 File Locking |

### Examples

```
# Disable/Add ArchR H5 Locking Globally
addArchRLocking(locking=FALSE)
```

---

## addArchRLogging *Set ArchR Logging*

---

### Description

This function will set ArchR logging

### Usage

```
addArchRLogging(useLogs = TRUE)
```

### Arguments

| | |
|---|---|
| `useLogs` | A boolean describing whether to use logging with ArchR. |

### Examples

```
# Add ArchR Logging
addArchRLogging()
```

`addArchRThreads` *Add a globally-applied number of threads to use for parallel computing.*

## Description

This function will set the number of threads to be used for parallel computing across all ArchR functions.

## Usage

```
addArchRThreads(threads = floor(parallel::detectCores()/2), force = FALSE)
```

## Arguments

| | |
|---|---|
| `threads` | The default number of threads to be used for parallel execution across all ArchR functions. This value is stored as a global environment variable, not part of the `ArchRProject`. This can be overwritten on a per-function basis using the given function's `threads` parameter. |
| `force` | If you request more than the total number of CPUs minus 2, ArchR will set `threads` to `(nCPU - 2)`. To bypass this, setting `force = TRUE` will use the number provided to `threads`. |

## Examples

```
# Add ArchR Threads
addArchRThreads()
```

`addArchRVerbose` *Set ArchR Verbosity for Log Messaging*

## Description

This function will set ArchR logging verbosity.

## Usage

```
addArchRVerbose(verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `verbose` | A boolean describing whether to printMessages in addition to logging with ArchR. |

## Examples

```
# Add ArchR Verbose
addArchRVerbose()
```

## `addBgdPeaks` *Add Background Peaks to an ArchRProject*

### Description

This function will compute background peaks controlling for total accessibility and GC-content and add this information to an ArchRProject.

### Usage

```
addBgdPeaks(
  ArchRProj = NULL,
  nIterations = 50,
  w = 0.1,
  binSize = 50,
  seed = 1,
  method = "chromVAR",
  outFile = file.path(getOutputDirectory(ArchRProj), "Background-Peaks.rds"),
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `nIterations` | The number of background peaks to sample. See `chromVAR::getBackgroundPeaks()`. |
| `w` | The parameter controlling similarity of background peaks. See `chromVAR::getBackgroundPeaks()`. |
| `binSize` | The precision with which the similarity is computed. See `chromVAR::getBackgroundPeaks()`. |
| `seed` | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `method` | A string indicating whether to use chromVAR or ArchR for background peak identification. |
| `outFile` | The path to save the `backgroundPeaks` object as a `.RDS` file for the given `ArchRProject`. The default action is to save this file in the `outputDirectory` of the `ArchRProject`. |
| `force` | A boolean value indicating whether to force the file indicated by `outFile` to be overwritten if it already exists. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Background Peaks
proj <- addBgdPeaks(proj, force = TRUE)
```

---

## `addCellColData` *Add information to cellColData in an ArchRProject*

---

### Description

This function adds new data to cellColData in a given ArchRProject.

### Usage

```
addCellColData(
  ArchRProj = NULL,
  data = NULL,
  name = NULL,
  cells = NULL,
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `data` | The data to add to `cellColData`. |
| `name` | The column header name to be used for this new data in `cellColData`. If a column with this name already exists, you may set `force` equal to `TRUE` to overwrite the data in this column. |
| `cells` | The names of the cells corresponding to `data`. Typically new data is added to all cells but you may use this argument to only add data to a subset of cells. Cells where `data` is not added are set to `NA`. |
| `force` | A boolean value indicating whether or not to overwrite data in a given column when the value passed to `name` already exists as a column name in `cellColData`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Cell Column Data
addCellColData(proj, data = proj$TSSEnrichment, name = "TSS2", cells = getCellNames(proj))
```

## addClusters — *Add cluster information to an ArchRProject*

### Description

This function will identify clusters from a reduced dimensions object in an ArchRProject or from a supplied reduced dimensions matrix.

### Usage

```
addClusters(
  input = NULL,
  reducedDims = "IterativeLSI",
  name = "Clusters",
  sampleCells = NULL,
  seed = 1,
  method = "Seurat",
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75,
  knnAssign = 10,
  nOutlier = 5,
  maxClusters = 25,
  testBias = TRUE,
  filterBias = FALSE,
  biasClusters = 0.01,
  biasCol = "nFrags",
  biasVals = NULL,
  biasQuantiles = c(0.05, 0.95),
  biasEnrich = 10,
  biasProportion = 0.5,
  biasPval = 0.05,
  nPerm = 500,
  prefix = "C",
  FCmethod = NULL,
  ArchRProj = NULL,
  verbose = TRUE,
  tstart = NULL,
  force = FALSE,
  logFile = createLogFile("addClusters"),
  ...
)
```

### Arguments

| | |
|---|---|
| `input` | Either (i) an `ArchRProject` object containing the dimensionality reduction matrix passed by `reducedDims` or (ii) a dimensionality reduction matrix. This object will be used for cluster identification. |

| | |
|---|---|
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. Not required if input is a matrix. |
| `name` | The column name of the cluster label column to be added to `cellColData` if `input` is an `ArchRProject` object. |
| `sampleCells` | An integer specifying the number of cells to subsample and perform clustering on. The remaining cells that were not subsampled will be assigned to the cluster of the nearest subsampled cell. This enables a decrease in run time but can sacrifice granularity of clusters. |
| `seed` | A number to be used as the seed for random number generation required in cluster determination. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `method` | A string indicating the clustering method to be used. Supported methods are "Seurat" and "Scran". |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `knnAssign` | The number of nearest neighbors to be used during clustering for assignment of outliers (clusters with less than nOutlier cells). |
| `nOutlier` | The minimum number of cells required for a group of cells to be called as a cluster. If a group of cells does not reach this threshold, then the cells will be considered outliers and assigned to nearby clusters. |
| `maxClusters` | The maximum number of clusters to be called. If the number exceeds this the clusters are merged unbiasedly using hclust and cutree. This is useful for contraining the cluster calls to be reasonable if they are converging on large numbers. Useful in iterativeLSI as well for initial iteration. Default is set to 25. |
| `testBias` | A boolean value that indicates whether or not to test clusters for bias. |
| `filterBias` | A boolean value indicates whether or not to filter clusters that are identified as biased. |
| `biasClusters` | A numeric value between 0 and 1 indicating that clusters that are smaller than the specified proportion of total cells are to be checked for bias. This should be set close to 0. We recommend a default of 0.01 which specifies clusters below 1 percent of the total cells. |
| `biasCol` | The name of a column in `cellColData` that contains the numeric values used for testing bias enrichment. |
| `biasVals` | A set of numeric values used for testing bias enrichment if `input` is not an `ArchRProject`. |

| | |
|---|---|
| `biasQuantiles` | A vector of two numeric values, each between 0 and 1, that describes the lower and upper quantiles of the bias values to use for computing bias enrichment statistics. |
| `biasEnrich` | A numeric value that specifies the minimum enrichment of biased cells over the median of the permuted background sets. |
| `biasProportion` | A numeric value between 0 and 1 that specifies the minimum proportion of biased cells in a cluster required to determine that the cluster is biased during testing for bias-enriched clusters. |
| `biasPval` | A numeric value between 0 and 1 that specifies the p-value to use when testing for bias-enriched clusters. |
| `nPerm` | An integer specifying the number of permutations to perform for testing bias-enriched clusters. |
| `prefix` | A character string to be added before each cluster identity. For example, if "Cluster" then cluster results will be "Cluster1", "Cluster2" etc. |
| `FCmethod` | A character string that will be passed as the method parameter of `Seurat::FindClusters()`. |
| `ArchRProj` | An `ArchRProject` object containing the dimensionality reduction matrix passed by `reducedDims`. This argument can also be supplied as `input`. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `tstart` | A timestamp that is typically passed internally from another function (for ex. "IterativeLSI") to measure how long the clustering analysis has been running relative to the start time when this process was initiated in another function. This argument is rarely manually specified. |
| `force` | A boolean value that indicates whether or not to overwrite data in a given column when the value passed to `name` already exists as a column name in `cellColData`. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `...` | Additional arguments to be provided to `Seurat::FindClusters()` or `scran::buildSNNGraph()` (for example, knn = 50, jaccard = TRUE). Note that to pass values to the `method` parameter of `Seurat::FindClusters()` you should use the `FCmethod` parameter in `addClusters()` because `addClusters()` already has a `method` parameter. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Peak Annotations
proj <- addClusters(proj, force = TRUE)
```

## `addCoAccessibility` *Add Peak Co-Accessibility to an ArchRProject*

### Description

This function will add co-accessibility scores to peaks in a given ArchRProject

### Usage

```
addCoAccessibility(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  dimsToUse = 1:30,
  scaleDims = NULL,
  corCutOff = 0.75,
  cellsToUse = NULL,
  excludeChr = NULL,
  k = 100,
  knnIteration = 500,
  overlapCutoff = 0.8,
  maxDist = 1e+05,
  scaleTo = 10^4,
  log2Norm = TRUE,
  seed = 1,
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("addCoAccessibility")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |

| | |
|---|---|
| `cellsToUse` | A character vector of cellNames to compute coAccessibility on if desired to run on a subset of the total cells. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from this analysis. |
| `k` | The number of k-nearest neighbors to use for creating single-cell groups for correlation analyses. |
| `knnIteration` | The number of k-nearest neighbor groupings to test for passing the supplied `overlapCutoff`. |
| `overlapCutoff` | The maximum allowable overlap between the current group and all previous groups to permit the current group be added to the group list during k-nearest neighbor calculations. |
| `maxDist` | The maximum allowable distance in basepairs between two peaks to consider for co-accessibility. |
| `scaleTo` | The total insertion counts from the designated group of single cells is summed across all relevant peak regions from the `peakSet` of the `ArchRProject` and normalized to the total depth provided by `scaleTo`. |
| `log2Norm` | A boolean value indicating whether to log2 transform the single-cell groups prior to computing co-accessibility correlations. |
| `seed` | A number to be used as the seed for random number generation required in knn determination. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Co-Accessibility
proj <- addCoAccessibility(proj, k = 20)
```

---

## `addCombinedDims` *Combine two or more modalities dimensionality reductions.*

---

## Description

This function will combine two or more modalities dimensionality reductions into a single reduction.

### Usage

```
addCombinedDims(
  ArchRProj = NULL,
  name = "CombinedDims",
  reducedDims = NULL,
  dimWeights = NULL,
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | The name for the combinedDims to be stored as. |
| `reducedDims` | The name of the `reducedDims` objects (i.e. "IterativeLSI") to use from the designated `ArchRProject`. |
| `dimWeights` | A vector of weights to be used to weight each dimensionality reduction when combining. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` objects to use. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |

## `addDemuxletResults` *Add Demuxlet Results to ArchR Project*

### Description

This function will read in the .best file output from demuxlet and add the doublet classifications into the cellColData for the ArchR Project

### Usage

```
addDemuxletResults(ArchRProj = NULL, bestFiles = NULL, sampleNames = NULL)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| bestFiles | The file path to the .best files created by Demuxlet. There should be one .best file for each sample in the ArchRProject. |
| sampleNames | The sample names corresponding to the .best files. These must match the sample names present in the ArchRProject. |

---

addDeviationsMatrix *Add a matrix of deviations for a given peakAnnotation to Arrow Files in ArchRProject*

---

## Description

This function will compute peakAnnotation deviations for each ArrowFiles independently while controlling for global biases (low-memory requirement).

## Usage

```
addDeviationsMatrix(
  ArchRProj = NULL,
  peakAnnotation = NULL,
  matches = NULL,
  bgdPeaks = getBgdPeaks(ArchRProj, method = "chromVAR"),
  matrixName = NULL,
  out = c("z", "deviations"),
  binarize = FALSE,
  version = 2,
  threads = getArchRThreads(),
  verbose = TRUE,
  parallelParam = NULL,
  force = FALSE,
  logFile = createLogFile("addDeviationsMatrix")
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| peakAnnotation | The name of the peakAnnotation stored in the ArchRProject. |
| matches | A custom peakAnnotation matches object used as input for the hypergeometric test. See motifmatchr::matchmotifs() for additional information. |
| bgdPeaks | A SummarizedExperiment that contains for each peak a set of background peaks matched by biases such as total accessibility and GC nucleotide content. This can be computed using addBgdPeaks and accessed by getBgdPeaks. |
| matrixName | The name to be used for storage of the deviations matrix in the provided ArchRProject. |

| | |
|---|---|
| `out` | A string or character vector that indicates whether to save the ouptut matrices as deviations ("deviations") z-scores ("z"), or both (c("deviations","z")). |
| `binarize` | A boolean value indicating whether the input matrix should be binarized before calculating deviations. This is often desired when working with insertion counts. |
| `version` | An integer for which deviations algorithm to use (1 = R native or 2 = C++ native). |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `force` | A boolean value indicating whether to force the matrix indicated by `matrixName` to be overwritten if it already exists in the ArrowFiles associated with the given `ArchRProject`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Background Peaks
proj <- addBgdPeaks(proj, force = TRUE)

# Add Motif Deviations
proj <- addDeviationsMatrix(
  ArchRProj = proj,
  peakAnnotation = "Motif",
  force = TRUE
)
```

---

## `addDoubletScores` *Add Doublet Scores to a collection of ArrowFiles or an ArchRProject*

---

## Description

For each sample in the ArrowFiles or ArchRProject provided, this function will independently assign inferred doublet information to each cell. This allows for removing strong heterotypic doublet-based clusters downstream. A doublet results from a droplet that contained two cells, causing the ATAC-seq data to be a mixture of the signal from each cell.

## Usage

```
addDoubletScores(
  input = NULL,
  useMatrix = "TileMatrix",
  k = 10,
  nTrials = 5,
  dimsToUse = 1:30,
  LSIMethod = 1,
  scaleDims = FALSE,
  corCutOff = 0.75,
  knnMethod = "UMAP",
  UMAPParams = list(n_neighbors = 40, min_dist = 0.4, metric = "euclidean", verbose =
    FALSE),
  LSIParams = list(outlierQuantiles = NULL, filterBias = FALSE),
  outDir = getOutputDirectory(input),
  threads = getArchRThreads(),
  force = FALSE,
  parallelParam = NULL,
  verbose = TRUE,
  logFile = createLogFile("addDoubletScores")
)
```

## Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or a character vector containing the paths to the ArrowFiles to be used. |
| `useMatrix` | The name of the matrix to be used for performing doublet identification analyses. Options include "TileMatrix" and "PeakMatrix". |
| `k` | The number of cells neighboring a simulated doublet to be considered as putative doublets. |
| `nTrials` | The number of times to simulate nCell (number of cells in the sample) doublets to use for doublet simulation when calculating doublet scores. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `LSIMethod` | A number or string indicating the order of operations in the TF-IDF normalization. Possible values are: 1 or "tf-logidf", 2 or "log(tf-idf)", and 3 or "logtf-logidf". |
| `scaleDims` | A boolean that indicates whether to z-score the reduced dimensions for each cell during the LSI method performed for doublet determination. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |

| | |
|---|---|
| `knnMethod` | The name of the dimensionality reduction method to be used for k-nearest neighbors calculation. Possible values are "UMAP" or "LSI". |
| `UMAPParams` | The list of parameters to pass to the UMAP function if "UMAP" is designated to `knnMethod`. See the function `umap` in the uwot package. |
| `LSIParams` | The list of parameters to pass to the `IterativeLSI()` function. See `IterativeLSI()`. |
| `outDir` | The relative path to the output directory for relevant plots/results from doublet identification. |
| `threads` | The number of threads to be used for parallel computing. |
| `force` | If the UMAP projection is not accurate (when R < 0.8 for the reprojection of the training data - this occurs when you have a very homogenous population of cells), setting `force=FALSE` will return -1 for all doubletScores and doubletEnrichments. If you would like to override this (not recommended!), you can bypass this warning by setting `force=TRUE`. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `verbose` | A boolean value that determines whether standard output is printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Doublet Scores for Small Project
proj <- addDoubletScores(proj, dimsToUse = 1:5, LSIParams = list(dimsToUse = 1:5, varFeatures=1000, iterations = 2)
```

---

`addFeatureCounts` *This function will add total counts of scATAC cells in provided features into ArchRProject.*

---

## Description

This function will add total counts of scATAC cells in provided features into ArchRProject.

## Usage

```
addFeatureCounts(
  ArchRProj = NULL,
  features = NULL,
  name = NULL,
  addRatio = TRUE,
  threads = getArchRThreads(),
  logFile = createLogFile("addFeatureCounts")
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| features | A GRanges object of features to count scATAC-seq data in. |
| name | A character defining the name of the features. "nameCounts" and "nameRatio" will be added to the ArchRProject. |
| addRatio | A boolean indicating whether to add the "nameRatio" to the ArchRProject. |
| threads | The number of threads to use for parallel execution. |
| logFile | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Feature Counts
proj <- addFeatureCounts(proj, features = getPeakSet(proj), name = 'ReadsInPeaks')
```

## addFeatureMatrix *Add a feature matrix to an ArchRProject or a set of ArrowFiles*

### Description

This function for each sample will independently compute counts for each feature per cell in the provided ArchRProject or set of ArrowFiles.

### Usage

```
addFeatureMatrix(
  input = NULL,
  features = NULL,
  matrixName = "FeatureMatrix",
  ceiling = 10^9,
  binarize = FALSE,
  verbose = TRUE,
  threads = getArchRThreads(),
  parallelParam = NULL,
  force = TRUE,
  logFile = createLogFile("addFeatureMatrix")
)
```

## Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or character vector of paths to ArrowFiles. |
| `features` | A `GRanges` object containing the regions (aka features) to use for counting insertions for each cell. |
| `matrixName` | The name to be used for storage of the feature matrix in the provided `ArchRProject` or ArrowFiles. |
| `ceiling` | The maximum counts per feature allowed. This is used to prevent large biases in feature counts. |
| `binarize` | A boolean value indicating whether the feature matrix should be binarized prior to storage. This can be useful for downstream analyses when working with insertion counts. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `force` | A boolean value indicating whether to force the matrix indicated by `matrixName` to be overwritten if it already exists in the `input`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Custom Matrix Which Is Just Peak Set
proj <- addFeatureMatrix(proj, features = getPeakSet(proj))
```

---

`addGeneExpressionMatrix`

*Add Gene Expression Matrix to ArrowFiles or an ArchRProject*

---

## Description

This function, for each sample, will add gene expression values from a paired scATAC-seq + scRNA-seq multi modal assay to the ArrowFiles or ArchRProject.

## Usage

```
addGeneExpressionMatrix(
  input = NULL,
  seRNA = NULL,
  chromSizes = getChromSizes(input),
  excludeChr = c("chrM", "chrY"),
  scaleTo = 10000,
  verbose = TRUE,
  threads = getArchRThreads(),
  parallelParam = NULL,
  strictMatch = FALSE,
  force = TRUE,
  logFile = createLogFile("addGeneExpressionMatrix")
)
```

### Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or character vector of ArrowFiles. |
| `seRNA` | A a scRNA-seq `SummarizedExperiment` (cell x gene) to be integrated with the scATAC-seq data. Cell names from this object much match those of the cell names in the ArrowFiles/ArchRProject. We will add support shortly for Seurat Objects (see `Seurat::as.SingleCellExperiment`). The provided values `MUST` be in counts (integer), not log transformed. |
| `chromSizes` | A GRanges object of the chromosome lengths. See `getChromSizes` for more info. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from this analysis. |
| `scaleTo` | Each column in the calculated gene score matrix will be normalized to a column sum designated by `scaleTo`. |
| `verbose` | A boolean describing whether to print to console messages of progress. |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `strictMatch` | A boolean value indicating whether every cell in `input` must be represented in `seRNA`. If set to `FALSE`, and this `GeneExpressionMatrix` is used for certain downstream analyses such as `addIterativeLSI()`, then errors may occur because not all cells will have relevant information. |
| `force` | A boolean value indicating whether to force the matrix indicated by `matrixName` to be overwritten if it already exist in the given `input`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## `addGeneIntegrationMatrix`

*Add a GeneIntegrationMatrix to ArrowFiles or an ArchRProject*

### Description

This function, will integrate multiple subsets of scATAC cells with a scRNA experiment, compute matched scRNA profiles and then store this in each samples ArrowFile.

## Usage

```
addGeneIntegrationMatrix(
  ArchRProj = NULL,
  useMatrix = "GeneScoreMatrix",
  matrixName = "GeneIntegrationMatrix",
  reducedDims = "IterativeLSI",
  seRNA = NULL,
  groupATAC = NULL,
  groupRNA = NULL,
  groupList = NULL,
  sampleCellsATAC = 10000,
  sampleCellsRNA = 10000,
  embeddingATAC = NULL,
  embeddingRNA = NULL,
  dimsToUse = 1:30,
  scaleDims = NULL,
  corCutOff = 0.75,
  plotUMAP = TRUE,
 UMAPParams = list(n_neighbors = 40, min_dist = 0.4, metric = "cosine", verbose = FALSE),
  nGenes = 2000,
  useImputation = TRUE,
  reduction = "cca",
  addToArrow = TRUE,
  scaleTo = 10000,
  genesUse = NULL,
  nameCell = "predictedCell",
  nameGroup = "predictedGroup",
  nameScore = "predictedScore",
  transferParams = list(),
  threads = getArchRThreads(),
  verbose = TRUE,
  force = FALSE,
  logFile = createLogFile("addGeneIntegrationMatrix"),
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of a matrix in the `ArchRProject` containing gene scores to be used for RNA integration. |
| `matrixName` | The name to use for the output matrix containing scRNA-seq integration to be stored in the `ArchRProject`. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. This `reducedDims` will be used in weighting the transfer of data to scRNA to scATAC. See `Seurat::TransferData` for more info. |

| | |
|---|---|
| `seRNA` | A `SeuratObject` or a scRNA-seq `SummarizedExperiment` (cell x gene) to be integrated with the scATAC-seq data. |
| `groupATAC` | A column name in `cellColData` of the `ArchRProj` that will be used to determine the subgroupings specified in `groupList`. This is used to constrain the integration to occur across biologically relevant groups. |
| `groupRNA` | A column name in either `colData` (if `SummarizedExperiment`) or `metadata` (if `SeuratObject`) of `seRNA` that will be used to determine the subgroupings specified in `groupList`. This is used to constrain the integration to occur across biologically relevant groups. Additionally this groupRNA is used for the `nameGroup` output of this function. |
| `groupList` | A list of cell groupings for both ATAC-seq and RNA-seq cells to be used for RNA-ATAC integration. This is used to constrain the integration to occur across biologically relevant groups. The format of this should be a list of groups with subgroups of ATAC and RNA specifying cells to integrate from both platforms. For example `groupList` <- list(groupA = list(ATAC = cellsATAC_A, RNA = cellsRNA_A), groupB = list(ATAC = cellsATAC_B, RNA = cellsRNA_B)) |
| `sampleCellsATAC` | An integer describing the number of scATAC-seq cells to be used for integration. This number will be evenly sampled across the total number of cells in the ArchRProject. |
| `sampleCellsRNA` | An integer describing the number of scRNA-seq cells to be used for integration. |
| `embeddingATAC` | A `data.frame` of cell embeddings such as a UMAP for scATAC-seq cells to be used for density sampling. The `data.frame` object should have a row for each single cell described in `row.names` and 2 columns, one for each dimension of the embedding. |
| `embeddingRNA` | A `data.frame` of cell embeddings such as a UMAP for scRNA-seq cells to be used for density sampling. The `data.frame` object should have a row for each single cell described in `row.names` and 2 columns, one for each dimension of the embedding. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `plotUMAP` | A boolean determining whether to plot a UMAP for each integration block. |
| `UMAPParams` | The list of parameters to pass to the UMAP function if "plotUMAP = TRUE". See the function `umap` in the uwot package. |

| | |
|---|---|
| `nGenes` | The number of variable genes determined by `Seurat::FindVariableGenes()` to use for integration. |
| `useImputation` | A boolean value indicating whether to use imputation for creating the Gene Score Matrix prior to integration. |
| `reduction` | The Seurat reduction method to use for integrating modalities. See `Seurat::FindTransferAnchors()` for possible reduction methods. |
| `addToArrow` | A boolean value indicating whether to add the log2-normalized transcript counts from the integrated matched RNA to the Arrow files. |
| `scaleTo` | Each column in the integrated RNA matrix will be normalized to a column sum designated by `scaleTo` prior to adding to Arrow files. |
| `genesUse` | If desired a character vector of gene names to use for integration instead of determined ones from Seurat::variableGenes. |
| `nameCell` | A column name to add to `cellColData` for the predicted scRNA-seq cell in the specified `ArchRProject`. This is useful for identifying which cell was closest to the scATAC-seq cell. |
| `nameGroup` | A column name to add to `cellColData` for the predicted scRNA-seq group in the specified `ArchRProject`. See `groupRNA` for more details. |
| `nameScore` | A column name to add to `cellColData` for the predicted scRNA-seq score in the specified `ArchRProject`. These scores represent the assignment accuracy of the group in the RNA cells. Lower scores represent ambiguous predictions and higher scores represent precise predictions. |
| `transferParams` | Additional params to be passed to `Seurat::TransferData`. |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `force` | A boolean value indicating whether to force the matrix indicated by `matrixName` to be overwritten if it already exists in the given `input`. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `...` | Additional params to be added to `Seurat::FindTransferAnchors` |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Get RNA Matrix
sePBMC <- readRDS(
  file.path(system.file("testdata", package = "ArchR"), "seRNA_PBMC.rds")
)

#Gene Integration Matrix
proj <- addGeneIntegrationMatrix(
    ArchRProj = proj,
    useMatrix = "GeneScoreMatrix",
    matrixName = "GeneIntegrationMatrix",
    reducedDims = "IterativeLSI",
    seRNA = sePBMC,
    addToArrow = FALSE,
    groupRNA = "CellType",
    nameCell = "predictedCell_Un2",
    nameGroup = "predictedGroup_Un2",
    nameScore = "predictedScore_Un2",
    dimsToUse = 1:10,
    nGenes = 250,
    force = TRUE
)
```

## addGeneScoreMatrix *Add GeneScoreMatrix to ArrowFiles or an ArchRProject*

### Description

This function, for each sample, will independently compute counts for each tile per cell and then infer gene activity scores.

### Usage

```
addGeneScoreMatrix(
  input = NULL,
  genes = getGenes(input),
  geneModel = "exp(-abs(x)/5000) + exp(-1)",
  matrixName = "GeneScoreMatrix",
  extendUpstream = c(1000, 1e+05),
  extendDownstream = c(1000, 1e+05),
  geneUpstream = 5000,
  geneDownstream = 0,
  useGeneBoundaries = TRUE,
  useTSS = FALSE,
  extendTSS = TRUE,
  tileSize = 500,
  ceiling = 4,
  geneScaleFactor = 5,
  scaleTo = 10000,
  excludeChr = c("chrY", "chrM"),
  blacklist = getBlacklist(input),
  saveGeneRegions = NULL,
  threads = getArchRThreads(),
  parallelParam = NULL,
  subThreading = TRUE,
  force = FALSE,
  logFile = createLogFile("addGeneScoreMatrix")
)
```

## Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or character vector of ArrowFiles. |
| `genes` | A stranded `GRanges` object containing the ranges associated with all gene start and end coordinates. |
| `geneModel` | A string giving a "gene model function" used for weighting peaks for gene score calculation. This string should be a function of `x`, where `x` is the stranded distance from the transcription start site of the gene. |
| `matrixName` | The name to be used for storage of the gene activity score matrix in the provided `ArchRProject` or ArrowFiles. |
| `extendUpstream` | The minimum and maximum number of basepairs upstream of the transcription start site to consider for gene activity score calculation. |
| `extendDownstream` | The minimum and maximum number of basepairs downstream of the transcription start site or transcription termination site (based on 'useTSS' and 'extendTSS') to consider for gene activity score calculation. |
| `geneUpstream` | An integer describing the number of bp upstream the gene to extend the gene body. This effectively makes the gene body larger as there are proximal peaks that should be weighted equally to the gene body. This parameter is used if 'useTSS=FALSE'. |
| `geneDownstream` | An integer describing the number of bp downstream the gene to extend the gene body.This effectively makes the gene body larger as there are proximal peaks that should be weighted equally to the gene body. This parameter is used if 'useTSS=FALSE'. |
| `useGeneBoundaries` | A boolean value indicating whether gene boundaries should be employed during gene activity score calculation. Gene boundaries refers to the process of preventing tiles from contributing to the gene score of a given gene if there is a second gene's transcription start site between the tile and the gene of interest. |
| `useTSS` | A boolean describing whether to build gene model based on gene TSS or the gene body. |
| `extendTSS` | A boolean describing whether to extend the gene TSS. By default useTSS uses the 1bp TSS while this parameter enables the extension of this region with 'geneUpstream' and 'geneDownstream' respectively. |
| `tileSize` | The size of the tiles used for binning counts prior to gene activity score calculation. |
| `ceiling` | The maximum counts per tile allowed. This is used to prevent large biases in tile counts. |
| `geneScaleFactor` | A numeric scaling factor to weight genes based on the inverse of there length i.e. (Scale Factor)/(Gene Length). This is scaled from 1 to the scale factor. Small genes will be the scale factor while extremely large genes will be closer to 1. This scaling helps with the relative gene score value. |
| `scaleTo` | Each column in the calculated gene score matrix will be normalized to a column sum designated by `scaleTo`. |

| | |
|---|---|
| excludeChr | A character vector containing the seqnames of the chromosomes that should be excluded from this analysis. |
| blacklist | A GRanges object containing genomic regions to blacklist that may be extremeley over-represented and thus biasing the geneScores for genes nearby that locus. saveGeneRegions The full path to a .rds file to save the GRanges object containing the regions used for calculating gene scores for each gene. |
| threads | The number of threads to be used for parallel computing. |
| parallelParam | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| subThreading | A boolean determining whether possible use threads within each multi-threaded subprocess if greater than the number of input samples. |
| force | A boolean value indicating whether to force the matrix indicated by matrixName to be overwritten if it already exist in the given input. |
| logFile | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Gene Score Matrix With New Model
proj <- addGeneScoreMatrix(proj, matrixName = "GeneScoreMatrix2", geneModel = "exp(-abs(x)/10000) + exp(-1)")
```

---

addGroupCoverages *Add Group Coverages to an ArchRProject object*

---

## Description

This function will merge cells within each designated cell group for the generation of pseudo-bulk replicates and then merge these replicates into a single insertion coverage file.

## Usage

```
addGroupCoverages(
  ArchRProj = NULL,
  groupBy = "Clusters",
  useLabels = TRUE,
  sampleLabels = "Sample",
  minCells = 40,
  maxCells = 500,
  maxFragments = 25 * 10^6,
  minReplicates = 2,
  maxReplicates = 5,
  sampleRatio = 0.8,
  excludeChr = NULL,
  kmerLength = 6,
  threads = getArchRThreads(),
  returnGroups = FALSE,
  parallelParam = NULL,
  force = FALSE,
  verbose = TRUE,
  logFile = createLogFile("addGroupCoverages")
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| groupBy | The name of the column in cellColData to use for grouping multiple cells together prior to generation of the insertion coverage file. |
| useLabels | A boolean value indicating whether to use sample labels to create sample-aware subgroupings during as pseudo-bulk replicate generation. |
| sampleLabels | The name of a column in cellColData to use to identify samples. In most cases, this parameter should be left as Sample and you should only use this parameter if you do not want to use the default sample labels stored in cellColData$Sample. However, if your individual Arrow files do not map to individual samples, then you should set this parameter to accurately identify your samples. This is the case in (for example) multiplexing applications where cells from different biological samples are mixed into the same reaction and demultiplexed based on a lipid barcode or genotype. |
| minCells | The minimum number of cells required in a given cell group to permit insertion coverage file generation. |
| maxCells | The maximum number of cells to use during insertion coverage file generation. |
| maxFragments | The maximum number of fragments per cell group to use in insertion coverage file generation. This prevents the generation of excessively large files which would negatively impact memory requirements. |
| minReplicates | The minimum number of pseudo-bulk replicates to be generated. |
| maxReplicates | The maximum number of pseudo-bulk replicates to be generated. |
| sampleRatio | The fraction of the total cells that can be sampled to generate any given pseudo-bulk replicate. |
| excludeChr | A character vector containing the seqnames of the chromosomes that should be excluded from this analysis. |
| kmerLength | The length of the k-mer used for estimating Tn5 bias. |
| threads | The number of threads to be used for parallel computing. |
| returnGroups | A boolean value that indicates whether to return sample-guided cell-groupings without creating coverages. This is used mainly in addReproduciblePeakSet() when MACS2 is not being used to call peaks but rather peaks are called from a TileMatrix (peakMethod = "Tiles"). |
| parallelParam | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| force | A boolean value that indicates whether or not to skip validation and overwrite the relevant data in the ArchRProject object if insertion coverage / pseudo-bulk replicate information already exists. |

| | |
|---|---|
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Group Coverages
proj <- addGroupCoverages(proj, force = TRUE)
```

---

## addHarmony *Add Harmony Batch Corrected Reduced Dims to an ArchRProject*

---

### Description

This function will add the Harmony batch-corrected reduced dimensions to an ArchRProject.

### Usage

```
addHarmony(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75,
  name = "Harmony",
  groupBy = "Sample",
  verbose = TRUE,
  force = FALSE,
  ...
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object containing the dimensionality reduction matrix passed by `reducedDims`. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `scaleDims` | A boolean that indicates whether to z-score the reduced dimensions for each cell. This is useful forminimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to |

| | |
|---|---|
| | `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `name` | The name to store harmony output as a `reducedDims` in the `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for vars in harmony batch correction. The value of `groupBy` is passed to the `vars_use` parameter in `harmony::HarmonyMatrix()`. When run through ArchR, this parameter defines which variables to correct for during batch correction. See `harmony::HarmonyMatrix()` for more information. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `force` | A boolean value that indicates whether or not to overwrite data in a given column when the value passed to `name` already exists as a column name in `cellColData`. |
| `...` | Additional arguments to be provided to harmony::HarmonyMatrix |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Confounder
proj <- addCellColData(proj, data = proj$TSSEnrichment > 10, name = "TSSQC", cells = getCellNames(proj))

# Run Harmony
proj <- addHarmony(proj, groupBy = "TSSQC")
```

## `addImputeWeights` *Add Imputation Weights to an ArchRProject*

### Description

This function computes imputations weights that describe each cell as a linear combination of many cells based on a MAGIC diffusion matrix.

### Usage

```
addImputeWeights(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75,
  td = 3,
  ka = 4,
  sampleCells = 5000,
  nRep = 2,
  k = 15,
  epsilon = 1,
  useHdf5 = TRUE,
  randomSuffix = FALSE,
  threads = getArchRThreads(),
  seed = 1,
  verbose = TRUE,
  logFile = createLogFile("addImputeWeights")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use. |
| `scaleDims` | A boolean that indicates whether to z-score the reduced dimensions for each cell. This is useful forminimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded. |
| `td` | The diffusion time parameter determines the number of smoothing iterations to be performed (see MAGIC from van Dijk et al Cell 2018). |
| `ka` | The k-nearest neighbors autotune parameter to equalize the effective number of neighbors for each cell, thereby diminishing the effect of differences in density. (see MAGIC from van Dijk et al Cell 2018). |
| `sampleCells` | The number of cells to sub-sample to compute an imputation block. An imputation block is a cell x cell matrix that describes the linear combination for imputation for numerical values within these cells. ArchR creates many blocks to keep this cell x cell matrix sparse for memory concerns. |
| `nRep` | An integer representing the number of imputation replicates to create when downsampling extremely low. |
| `k` | The number of nearest neighbors for smoothing to use for MAGIC (see MAGIC from van Dijk et al Cell 2018). |
| `epsilon` | The value for the standard deviation of the kernel for MAGIC (see MAGIC from van Dijk et al Cell 2018). |

| | |
|---|---|
| `useHdf5` | A boolean value that indicates whether HDF5 format should be used to store the impute weights. |
| `randomSuffix` | A boolean value that indicates whether a random suffix should be appended to the saved imputation weights hdf5 files. |
| `threads` | The number of threads to be used for parallel computing. |
| `seed` | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Impute Weights
proj <- addImputeWeights(proj)
```

---

## addIterativeLSI *Add an Iterative LSI-based dimensionality reduction to an ArchRProject*

---

### Description

This function will compute an iterative LSI dimensionality reduction on an ArchRProject.

### Usage

```
addIterativeLSI(
  ArchRProj = NULL,
  useMatrix = "TileMatrix",
  name = "IterativeLSI",
  iterations = 2,
 clusterParams = list(resolution = c(2), sampleCells = 10000, maxClusters = 6, n.start =
    10),
  firstSelection = "top",
  depthCol = "nFrags",
  varFeatures = 25000,
  dimsToUse = 1:30,
  LSIMethod = 2,
  scaleDims = TRUE,
  corCutOff = 0.75,
  binarize = TRUE,
  outlierQuantiles = c(0.02, 0.98),
  filterBias = TRUE,
  sampleCellsPre = 10000,
  projectCellsPre = FALSE,
  sampleCellsFinal = NULL,
  selectionMethod = "var",
  scaleTo = 10000,
  totalFeatures = 5e+05,
  filterQuantile = 0.995,
  excludeChr = c(),
  keep0lsi = FALSE,
  saveIterations = TRUE,
  UMAPParams = list(n_neighbors = 40, min_dist = 0.4, metric = "cosine", verbose = FALSE,
    fast_sgd = TRUE),
  nPlot = 10000,
  outDir = getOutputDirectory(ArchRProj),
  threads = getArchRThreads(),
  seed = 1,
  verbose = TRUE,
  force = FALSE,
  logFile = createLogFile("addIterativeLSI")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the data matrix to retrieve from the ArrowFiles associated with the `ArchRProject`. Valid options are "TileMatrix" or "PeakMatrix". |
| `name` | The name to use for storage of the IterativeLSI dimensionality reduction in the `ArchRProject` as a `reducedDims` object. |
| `iterations` | The number of LSI iterations to perform. |
| `clusterParams` | A list of additional parameters to be passed to `addClusters()` for clustering within each iteration. These params can be constant across each iteration, or specified for each iteration individually. Thus each param must be of length == 1 or the total number of `iterations` - 1. If you want to use `scran` for clustering, you would pass this as `method="scran"`. |
| `firstSelection` | First iteration selection method for features to use for LSI. Either "Top" for the top accessible/average or "Var" for the top variable features. "Top" should be used for all scATAC-seq data (binary) while "Var" should be used for all scRNA/other-seq data types (non-binary). |
| `depthCol` | A column in the `ArchRProject` that represents the coverage (scATAC = unique fragments, scRNA = unique molecular identifiers) per cell. These values are used to minimize the related biases in the reduction related. For scATAC we recommend "nFrags" and for scRNA we recommend "Gex_nUMI". |
| `varFeatures` | The number of N variable features to use for LSI. The top N features will be used based on the `selectionMethod`. |

| | |
|---|---|
| `dimsToUse` | A vector containing the dimensions to use in LSI. The total dimensions used in LSI will be `max(dimsToUse)`. If you set this too high, it could impact downstream functionalities including increasing the time required to run `addClusters()`. |
| `LSIMethod` | A number or string indicating the order of operations in the TF-IDF normalization. Possible values are: 1 or "tf-logidf", 2 or "log(tf-idf)", and 3 or "logtf-logidf". |
| `scaleDims` | A boolean that indicates whether to z-score the reduced dimensions for each cell. This is useful forminimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `binarize` | A boolean value indicating whether the matrix should be binarized before running LSI. This is often desired when working with insertion counts. |
| `outlierQuantiles` | Two numerical values (between 0 and 1) that describe the lower and upper quantiles of bias (number of acessible regions per cell, determined by `nFrags` or `colSums`) to filter cells prior to LSI. For example a value of c(0.02, 0.98) results in the cells in the bottom 2 percent and upper 98 percent to be filtered prior to LSI. These cells are then projected back in the LSI subspace. This prevents spurious 'islands' that are identified based on being extremely biased. These quantiles are also used for sub-sampled LSI when determining which cells are used. |
| `filterBias` | A boolean indicating whether to drop bias clusters when computing clusters during iterativeLSI. |
| `sampleCellsPre` | An integer specifying the number of cells to sample in iterations prior to the last in order to perform a sub-sampled LSI and sub-sampled clustering. This greatly reduced memory usage and increases speed for early iterations. |
| `projectCellsPre` | A boolean indicating whether to reproject all cells into the sub-sampled LSI (see `sampleCellsPre`). Setting this to `FALSE` allows for using the sub-sampled LSI for clustering and variance identification. If `TRUE` the cells are all projected into the sub-sampled LSI and used for cluster and variance identification. |
| `sampleCellsFinal` | An integer specifying the number of cells to sample in order to perform a sub-sampled LSI in final iteration. |
| `selectionMethod` | The selection method to be used for identifying the top variable features. Valid options are "var" for log-variability or "vmr" for variance-to-mean ratio. |
| `scaleTo` | Each column in the matrix designated by `useMatrix` will be normalized to a column sum designated by `scaleTo` prior to variance calculation and TF-IDF normalization. |
| `totalFeatures` | The number of features to consider for use in LSI after ranking the features by the total number of insertions. These features are the only ones used throught the |

| | |
|---|---|
| | variance identification and LSI. These are an equivalent when using a `TileMatrix` to a defined peakSet. |
| `filterQuantile` | A number 0,1 that indicates the quantile above which features should be removed based on insertion counts prior to the first iteration of the iterative LSI paradigm. For example, if `filterQuantile = 0.99`, any features above the 99th percentile in insertion counts will be ignored for the first LSI iteration. |
| `excludeChr` | A string of chromosomes to exclude for iterativeLSI procedure. |
| `keep0lsi` | A boolean whether to keep cells with no reads in features used for LSI. |
| `saveIterations` | A boolean value indicating whether the results of each LSI iterations should be saved as compressed `.rds` files in the designated `outDir`. |
| `UMAPParams` | The list of parameters to pass to the UMAP function if "UMAP" if `saveIterations=TRUE`. See the function `uwot::umap()`. |
| `nPlot` | If `saveIterations=TRUE`, how many cells to sample make a UMAP and plot for each iteration. |
| `outDir` | The output directory for saving LSI iterations if desired. Default is in the `outputDirectory` of the `ArchRProject`. |
| `threads` | The number of threads to be used for parallel computing. |
| `seed` | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `force` | A boolean value that indicates whether or not to overwrite relevant data in the `ArchRProject` object. |
| `logFile` | The path to a file to be used for logging ArchR output. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Iterative LSI
proj <- addIterativeLSI(proj, dimsToUse = 1:5, varFeatures=1000, iterations = 2, force=TRUE)
```

## addModuleScore *Add Module Scores to an ArchRProject*

### Description

This function calculates a module score from a set of features across all cells. This allows for grouping of multiple features together into a single quantitative measurement. Currently, this function only works for modules derived from the `GeneScoreMatrix`. Each module is added as a new column in `cellColData`

## Usage

```
addModuleScore(
  ArchRProj = NULL,
  useMatrix = NULL,
  name = "Module",
  features = NULL,
  nBin = 25,
  nBgd = 100,
  seed = 1,
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("addModuleScore")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the matrix to be used for calculation of the module score. See `getAvailableMatrices()` to view available options. |
| `name` | The name to be given to the designated module. If `features` is a list, this name will be prepended to the feature set names given in the list as shown below. |
| `features` | A list of feature names to be grouped into modules. For example, `list(BScore = c("MS4A1", "CD79A", "CD74"), TScore = c("CD3D", "CD8A", "GZMB", "CCR7", "LEF1"))`. Each named element in this list will be stored as a separate module. The examples given in these parameters would yield two modules called `Module.Bscore` and `Module.Tscore`. If the elements of this list are not named, they will be numbered in order, i.e. `Module1`, `Module2`. |
| `nBin` | The number of bins to use to divide all features for identification of signal-matched features for background calculation |
| `nBgd` | The number of background features to use for signal normalization. |
| `seed` | A number to be used as the seed for random number generation required when sampling cells for the background set. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Module Score
proj <- addModuleScore(proj, useMatrix = "GeneScoreMatrix", nBin = 25, nBgd = 25, features = list(TScore = c('CD3D'
#Check
split(proj@cellColData$Module.TScore, proj@cellColData$CellType) %>% lapply(mean) %>% unlist
#        B         M         T
# -4.352769 -8.438259  9.942678

#Get T cell Features
features <- getGenes()
T <- features[features$symbol %in% c("CD3D", "CD3E")]
B <- features[features$symbol %in% c("MS4A1")]

# Add Module Score
proj <- addModuleScore(proj, useMatrix = "TileMatrix", nBin = 25, nBgd = 25, features = list(TScore = T, BScore = B))

#Check
split(proj@cellColData$Module.TScore, proj@cellColData$CellType) %>% lapply(mean) %>% unlist
#          B           M           T
# -0.03866667 -0.05303030  0.10306122

split(proj@cellColData$Module.BScore, proj@cellColData$CellType) %>% lapply(mean) %>% unlist
#          B           M           T
# 0.10000000 -0.03939394 -0.05387755
```

---

`addMonocleTrajectory` *Add a Monocle Trajectory to an ArchR Project*

---

## Description

This function will add a trajectory from a monocle CDS created from `getMonocleTrajectories` to an ArchRProject.

## Usage

```
addMonocleTrajectory(
  ArchRProj = NULL,
  name = "Trajectory",
  useGroups = NULL,
  groupBy = "Clusters",
  monocleCDS = NULL,
  force = FALSE
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | A string indicating the name of the fitted trajectory to be added in `cellColData`. |
| `useGroups` | The cell groups to be used for creating trajectory analysis. |

| | |
|---|---|
| groupBy | A string indicating the column name from cellColData that contains the cell group definitions used in useGroups to constrain trajectory analysis. |
| monocleCDS | A monocle CDS object created from getMonocleTrajectories. |
| force | A boolean value indicating whether to force the trajactory indicated by name to be overwritten if it already exists in the given ArchRProject. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Create Monocole Trajectory
cds <- getMonocleTrajectories(
  ArchRProj = proj,
  useGroups = c("C1", "C2", "C3"),
  principalGroup = "C1",
  groupBy = "Clusters",
  embedding = "UMAP"
)

# Add Monocole Trajectory
proj <- addMonocleTrajectory(
  ArchRProj = proj,
  name = "Trajectory_Monocole",
  useGroups = c("C1", "C2", "C3"),
  monocleCDS = cds
)
```

## addMotifAnnotations *Add motif annotations to an ArchRProject*

## Description

This function adds information about which peaks contain motifs to a given ArchRProject. For each peak, a binary value is stored indicating whether each motif is observed within the peak region.

## Usage

```
addMotifAnnotations(
  ArchRProj = NULL,
  motifSet = "cisbp",
  annoName = "Motif",
  species = NULL,
  collection = "CORE",
  motifPWMs = NULL,
  cutOff = 5e-05,
  width = 7,
  version = 2,
  force = FALSE,
  logFile = createLogFile("addMotifAnnotations"),
  ...
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| motifSet | The name of a curated motif set to be used for annotation. Options include: (i) "JASPAR2016", "JASPAR2018", "JASPAR2020" which gives the 2016, 2018, or 2020 version of JASPAR motifs, (ii) one of "cisbp", "encode", or "homer" which gives the corresponding motif sets from the chromVAR package, or (iii) "vierstra" which gives the clustered archetype motifs created by Jeff Vierstra (https://github.com/jvierstra/motif-clustering). |
| annoName | The name of the peakAnnotation object to be stored in the provided ArchRProject |
| species | The latin name of the species relevant to the supplied ArchRProject. This is used for identifying which motif to be used from CisBP/JASPAR. For JASPAR, species is passed to TFBS::getMatrixSet and some species names are not recognized. In these cases it is possible to use the NCBI taxonomy ID. By default, this function will attempt to guess the species based on the value from getGenome(). |
| collection | If one of the JASPAR motif sets is used via motifSet, this parameter allows you to indicate the JASPAR collection to be used. See getMatrixSet() from TFBSTools for all options to supply for collection. If motifSet is "vierstra", then this must either be "archetype" (for the v2.1 clustered models) or "individual" (for the original v1 individual motif models). NOTE: vierstra archetype motifs are currently in beta and have not been finalized by Jeff Vierstra. |
| motifPWMs | A custom set of motif PWMs as a PWMatrixList to be used instead of motifSet for adding motif annotations. If motifPWMs is used, motifSet will be ignored. |
| cutOff | The p-value cutoff to be used for motif search. The p-value is determined vs a background set of sequences (see MOODS for more details on this determination). |
| width | The width in basepairs to consider for motif matches. See the motimatchr package for more information. |
| version | An integer specifying version 1 or version 2 of chromVARmotifs see github for more info GreenleafLab/chromVARmotifs. |
| force | A boolean value indicating whether to force the peakAnnotation object indicated by annoName to be overwritten if it already exists in the given ArchRProject. |
| logFile | The path to a file to be used for logging ArchR output. |
| ... | Additional parameters to be passed to TFBSTools::getMatrixSet for getting a JASPAR PWM object. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()
```

```
# Add Motif Annotations
proj <- addMotifAnnotations(proj, motifSet = "cisbptest", annoName = "test")
```

## addPeak2GeneLinks *Add Peak2GeneLinks to an ArchRProject*

### Description

This function will add peak-to-gene links to a given ArchRProject

### Usage

```
addPeak2GeneLinks(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  useMatrix = "GeneIntegrationMatrix",
  dimsToUse = 1:30,
  scaleDims = NULL,
  corCutOff = 0.75,
  cellsToUse = NULL,
  excludeChr = NULL,
  k = 100,
  knnIteration = 500,
  overlapCutoff = 0.8,
  maxDist = 250000,
  scaleTo = 10^4,
  log2Norm = TRUE,
  predictionCutoff = 0.4,
  addEmpiricalPval = FALSE,
  addPermutedPval = FALSE,
  nperm = 100,
  seed = 1,
  threads = max(floor(getArchRThreads()/2), 1),
  verbose = TRUE,
  logFile = createLogFile("addPeak2GeneLinks")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. |
| `useMatrix` | The name of the matrix containing gene expression information to be used for determining peak-to-gene links. See `getAvailableMatrices(ArchRProj)` |

| | |
|---|---|
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in clustering. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `cellsToUse` | A character vector of cellNames to compute peak-to-gene links on if desired to run on a subset of the total cells. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from this analysis. |
| `k` | The number of k-nearest neighbors to use for creating single-cell groups for correlation analyses. |
| `knnIteration` | The number of k-nearest neighbor groupings to test for passing the supplied `overlapCutoff`. |
| `overlapCutoff` | The maximum allowable overlap between the current group and all previous groups to permit the current group be added to the group list during k-nearest neighbor calculations. |
| `maxDist` | The maximum allowable distance in basepairs between two peaks to consider for co-accessibility. |
| `scaleTo` | The total insertion counts from the designated group of single cells is summed across all relevant peak regions from the `peakSet` of the `ArchRProject` and normalized to the total depth provided by `scaleTo`. |
| `log2Norm` | A boolean value indicating whether to log2 transform the single-cell groups prior to computing co-accessibility correlations. |
| `predictionCutoff` | A numeric describing the cutoff for RNA integration to use when picking cells for groupings. |
| `addEmpiricalPval` | Add empirical p-values based on randomly correlating peaks and genes not on the same seqname. |
| `addPermutedPval` | Add permuted p-values based on shuffle sample correlating peaks and genes. This approach was adapted from Regner et al 2021 "A multi-omic single-cell landscape of human gynecologic malignancies". |
| `nperm` | An integer representing the number of permutations to run for Regner et al 2021 approach. |
| `seed` | A number to be used as the seed for random number generation required in knn determination. It is recommended to keep track of the seed used so that you can reproduce results downstream. |

| | |
|---|---|
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add P2G Links
proj <- addPeak2GeneLinks(proj, k = 20)

# Get P2G Links
p2g <- getPeak2GeneLinks(proj)
```

---

## `addPeakAnnotations` *Add peak annotations to an ArchRProject*

---

### Description

This function adds information about which peaks contain input regions to a given ArchRProject. For each peak, a binary value is stored indicating whether each region is observed within the peak region.

### Usage

```
addPeakAnnotations(
  ArchRProj = NULL,
  regions = NULL,
  name = "Region",
  force = FALSE,
  logFile = createLogFile("addPeakAnnotations")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `regions` | A named `list` of `GRanges` that are to be overlapped with the `peakSet` in the `ArchRProject`. |
| `name` | The name of `peakAnnotation` object to be stored as in `ArchRProject`. |
| `force` | A boolean value indicating whether to force the `peakAnnotation` object indicated by `name` to be overwritten if it already exists in the given `ArchRProject`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Motif Positions Can Be Any Interval GRanges List
positions <- getPositions(proj)

# Add Peak Annotations
proj <- addPeakAnnotations(proj, regions = positions)
```

---

## `addPeakMatrix` *Add a Peak Matrix to the ArrowFiles of an ArchRProject*

---

### Description

This function, for each sample, will independently compute counts for each peak per cell in the provided ArchRProject using the "PeakMatrix".

### Usage

```
addPeakMatrix(
  ArchRProj = NULL,
  ceiling = 4,
  binarize = FALSE,
  verbose = TRUE,
  threads = getArchRThreads(),
  parallelParam = NULL,
  force = TRUE,
  logFile = createLogFile("addPeakMatrix")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `ceiling` | The maximum counts per feature allowed. This is used to prevent large biases in peak counts. |
| `binarize` | A boolean value indicating whether the peak matrix should be binarized prior to storage. This can be useful for downstream analyses when working with insertion counts. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `force` | A boolean value indicating whether to force the "PeakMatrix" to be overwritten if it already exist in the given `ArchRProject`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Peak Matrix
proj <- addPeakMatrix(proj)
```

---

`addPeakSet` *Add a peak set to an ArchRProject*

---

## Description

This function adds a peak set as a GRanges object to a given ArchRProject.

## Usage

```
addPeakSet(
  ArchRProj = NULL,
  peakSet = NULL,
  genomeAnnotation = getGenomeAnnotation(ArchRProj),
  force = FALSE
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `peakSet` | A `GRanges` object containing the set of regions that define all peaks in the desired peak set. |
| `genomeAnnotation` | The genomeAnnotation (see `createGenomeAnnotation()`) to be used for generating peak metadata such as nucleotide information (GC content) or chromosome sizes. |
| `force` | If a `peakSet` object has already been added to the given `ArchRProject`, the value of `force` determines whether or not to overwrite this `peakSet`. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add PeakSet
addPeakSet(proj, peakSet = getPeakSet(proj), force = TRUE)
```

## addProjectSummary *Add projectSummary to an ArchRProject*

### Description

This function adds info to the projectSummary of an ArchRProject

### Usage

```
addProjectSummary(ArchRProj = NULL, name = NULL, summary = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | The name of the summary information to add to the `ArchRProject` object. |
| `summary` | A vector to add as summary information to the `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Project Summary
addProjectSummary(proj, name = "test", summary = "test successful")
```

## addReproduciblePeakSet *Add a Reproducible Peak Set to an ArchRProject*

### Description

This function will get insertions from coverage files, call peaks, and merge peaks to get a "Union Reproducible Peak Set".

### Usage

```
addReproduciblePeakSet(
  ArchRProj = NULL,
  groupBy = "Clusters",
  peakMethod = "Macs2",
  reproducibility = "2",
  peaksPerCell = 500,
  maxPeaks = 150000,
  minCells = 25,
  excludeChr = c("chrM", "chrY"),
  pathToMacs2 = if (tolower(peakMethod) == "macs2") findMacs2() else NULL,
  genomeSize = NULL,
  shift = -75,
  extsize = 150,
  method = if (tolower(peakMethod) == "macs2") "q" else "p",
  cutOff = 0.1,
  additionalParams = "--nomodel --nolambda",
  extendSummits = 250,
  promoterRegion = c(2000, 100),
  genomeAnnotation = getGenomeAnnotation(ArchRProj),
  geneAnnotation = getGeneAnnotation(ArchRProj),
  plot = TRUE,
  threads = getArchRThreads(),
  parallelParam = NULL,
  force = FALSE,
  verbose = TRUE,
  logFile = createLogFile("addReproduciblePeakSet"),
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for peak calling. |
| `peakMethod` | The name of peak calling method to be used. Options include "Macs2" for using macs2 callpeak or "Tiles" for using a TileMatrix. |
| `reproducibility` | A string that indicates how peak reproducibility should be handled. This string is dynamic and can be a function of `n` where `n` is the number of samples being assessed. For example, `reproducibility = "2"` means at least 2 samples must have a peak call at this locus and `reproducibility = "(n+1)/2"` means that the majority of samples must have a peak call at this locus. |
| `peaksPerCell` | The upper limit of the number of peaks that can be identified per cell-grouping in `groupBy`. This is useful for controlling how many peaks can be called from cell groups with low cell numbers. |
| `maxPeaks` | A numeric threshold for the maximum peaks to retain per group from `groupBy` in the union reproducible peak set. |
| `minCells` | The minimum allowable number of unique cells that was used to create the coverage files on which peaks are called. This is important to allow for exclusion of pseudo-bulk replicates derived from very low cell numbers. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from peak calling. |
| `pathToMacs2` | The full path to the MACS2 executable. |
| `genomeSize` | The genome size to be used for MACS2 peak calling (see MACS2 documentation). This is required if genome is not hg19, hg38, mm9, or mm10. |

| | |
|---|---|
| `shift` | The number of basepairs to shift each Tn5 insertion. When combined with `extsize` this allows you to create proper fragments, centered at the Tn5 insertion site, for use with MACS2 (see MACS2 documentation). |
| `extsize` | The number of basepairs to extend the MACS2 fragment after `shift` has been applied. When combined with `extsize` this allows you to create proper fragments, centered at the Tn5 insertion site, for use with MACS2 (see MACS2 documentation). |
| `method` | The method to use for significance testing in MACS2. Options are "p" for p-value and "q" for q-value. When combined with `cutOff` this gives the method and significance threshold for peak calling (see MACS2 documentation). |
| `cutOff` | The numeric significance cutOff for the testing method indicated by `method` (see MACS2 documentation). |
| `additionalParams` | A string of additional parameters to pass to MACS2 (see MACS2 documentation). |
| `extendSummits` | The number of basepairs to extend peak summits (in both directions) to obtain final fixed-width peaks. For example, `extendSummits = 250` will create 501-bp fixed-width peaks from the 1-bp summits. |
| `promoterRegion` | A vector of two integers specifying the distance in basepairs upstream and downstream of a TSS to be included as a promoter region. Peaks called within one of these regions will be annotated as a "promoter" peak. For example, `promoterRegion = c(2000, 100)` will annotate any peak within the region 2000 bp upstream and 100 bp downstream of a TSS as a "promoter" peak. |
| `genomeAnnotation` | The genomeAnnotation (see `createGenomeAnnotation()`) to be used for generating peak metadata such as nucleotide information (GC content) or chromosome sizes. |
| `geneAnnotation` | The geneAnnotation (see `createGeneAnnotation()`) to be used for labeling peaks as "promoter", "exonic", etc. |
| `plot` | A boolean describing whether to plot peak annotation results. |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `force` | A boolean value indicating whether to force the reproducible peak set to be overwritten if it already exist in the given `ArchRProject` peakSet. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `...` | Additional parameters to be pass to `addGroupCoverages()` to get sample-guided pseudobulk cell-groupings. Only used for TileMatrix-based peak calling (not for MACS2). See `addGroupCoverages()` for more info. |

## Examples

```
# Get Test ArchR Project
```

```
proj <- getTestProject()

# Add Peak Matrix Tiles
proj <- addReproduciblePeakSet(proj, peakMethod = "tiles")

# Add Peak Matrix Macs2 (Preferred)
proj <- addReproduciblePeakSet(proj, peakMethod = "macs2")
```

---

## `addSampleColData` *Add information to sampleColData in an ArchRProject*

---

### Description

This function adds new data to sampleColData in an ArchRProject.

### Usage

```
addSampleColData(
  ArchRProj = NULL,
  data = NULL,
  name = NULL,
  samples = NULL,
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `data` | A vector containing the data to be added to `sampleColData`. |
| `name` | The column header name to be used for this new data in `sampleColData`. If a column with this name already exists, you may set `force` equal to `TRUE` to overwrite the data in this column. |
| `samples` | The names of the samples corresponding to `data`. Typically new data is added to all samples but you may use this argument to only add data to a subset of samples. Samples where `data` is not added are set to `NA`. |
| `force` | A boolean value that indicates whether or not to overwrite data in a given column when the value passed to `name` already exists as a column name in `sampleColData`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Sample Column Data
addSampleColData(proj, data = 1, name = "Test", samples = "PBSmall")
```

## addSlingShotTrajectories *Add a Slingshot Trajectories to an ArchR Project*

### Description

This function will fit a supervised trajectory in a lower dimensional space that can then be used for downstream analyses.

### Usage

```
addSlingShotTrajectories(
  ArchRProj = NULL,
  name = "SlingShot",
  useGroups = NULL,
  principalGroup = NULL,
  groupBy = NULL,
  embedding = NULL,
  reducedDims = NULL,
  force = FALSE,
  seed = 1
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | A string indicating the name of the fitted trajectory to be added in `cellColData`. |
| `useGroups` | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData`. This limits the groups used to identify trajectories. |
| `principalGroup` | The principal group which represents the group that will be the starting point for all trajectories. |
| `groupBy` | A string indicating the column name from `cellColData` that contains the cell group definitions used in `useGroups` to constrain trajectory analysis. |
| `embedding` | A string indicating the name of the `embedding` object from the `ArchRProject` that should be used for trajectory analysis. |
| `reducedDims` | A string indicating the name of the `reducedDims` object from the `ArchRProject` that should be used for trajectory analysis. `embedding` must equal NULL to use. |
| `force` | A boolean value indicating whether to force the trajactory indicated by `name` to be overwritten if it already exists in the given `ArchRProject`. |
| `seed` | A number to be used as the seed for random number generation for trajectory creation. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add SlingShot Trajectory
proj <- addSlingShotTrajectories(
  ArchRProj = proj,
  name = "Trajectory_SlingShot",
  useGroups = c("C1", "C2", "C3"),
  principalGroup = "C1",
  groupBy = "Clusters",
  embedding = "UMAP"
)
```

---

`addTileMatrix` *Add TileMatrix to ArrowFiles or an ArchRProject*

---

## Description

This function, for each sample, will independently compute counts for each tile

## Usage

```
addTileMatrix(
  input = NULL,
  chromSizes = if (inherits(input, "ArchRProject")) getChromSizes(input) else NULL,
  blacklist = if (inherits(input, "ArchRProject")) getBlacklist(input) else NULL,
  tileSize = 500,
  binarize = TRUE,
  excludeChr = c("chrM", "chrY"),
  threads = getArchRThreads(),
  parallelParam = NULL,
  force = FALSE,
  logFile = createLogFile("addTileMatrix")
)
```

## Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or character vector of ArrowFiles. |
| `chromSizes` | A named numeric vector containing the chromsome names and lengths. The default behavior is to retrieve this from the `ArchRProject` using `getChromSizes()`. |
| `blacklist` | A `GRanges` object containing genomic regions to blacklist counting in these tiles. The default behavior is to retrieve this from the `ArchRProject` using `getBlacklist()`. |
| `tileSize` | The size of the tiles used for binning counts in the "TileMatrix". |

| | |
|---|---|
| `binarize` | A boolean value indicating whether the "TileMatrix" should be binarized prior to storage. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from the "TileMatrix". |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `force` | A boolean value indicating whether to force the "TileMatrix' to be overwritten if it already exist in the given `input`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Tile Matrix
proj <- addTileMatrix(proj, force = TRUE, tileSize = 25000)
```

---

## `addTrajectory` *Add a Supervised Trajectory to an ArchR Project*

---

### Description

This function will fit a supervised trajectory in a lower dimensional space that can then be used for downstream analyses.

### Usage

```
addTrajectory(
  ArchRProj = NULL,
  name = "Trajectory",
  trajectory = NULL,
  groupBy = "Clusters",
  reducedDims = "IterativeLSI",
  embedding = NULL,
  preFilterQuantile = 0.9,
  postFilterQuantile = 0.9,
  useAll = FALSE,
  dof = 250,
  spar = 1,
  saveDF = NULL,
  force = FALSE,
  seed = 1,
  logFile = createLogFile("addTrajectory")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | A string indicating the name of the fitted trajectory to be added in `cellColData`. |
| `trajectory` | The order of cell groups to be used for constraining the initial supervised fitting procedure. For example, to get a trajectory from Cluster1 to Cluster2 to Cluster3, input should be c("Cluster1", "Cluster2", "Cluster3"). Cells will then be used from these 3 groups to constrain an initial fit in the group order. |
| `groupBy` | A string indicating the column name from `cellColData` that contains the cell group definitions used in `trajectory` to constrain the initial supervised fitting procedure. |
| `reducedDims` | A string indicating the name of the `reducedDims` object from the `ArchRProject` that should be used for distance computation. |
| `embedding` | A string indicating the name of the `embedding` object from the `ArchRProject` that should be used for distance computation. |
| `preFilterQuantile` | Prior to the initial supervised trajectory fitting, cells whose euclidean distance from the cell-grouping center is above the provided quantile will be excluded. |
| `postFilterQuantile` | After initial supervised trajectory fitting, cells whose euclidean distance from the cell-grouping center is above the provided quantile will be excluded. |
| `useAll` | A boolean describing whether to use cells outside of trajectory groups for post-fitting procedure. |
| `dof` | The number of degrees of freedom to be used in the spline fit. See `stats::smooth.spline()` for more information. |
| `spar` | The sparsity to be used in the spline fit. See `stats::smooth.spline()` for more information. |
| `saveDF` | A full or relative path to use for creating a `.RDS` R object file containing a DataFrame with additional information about the trajectory. This includes the distance of each cell to the splines of the trajectory and the corresponding indicies returned by `nabor::knn`. |
| `force` | A boolean value indicating whether to force the trajactory indicated by `name` to be overwritten if it already exists in the given `ArchRProject`. |
| `seed` | A number to be used as the seed for random number generation for trajectory creation. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add Trajectory
proj <- addTrajectory(proj, trajectory = c("C1", "C2", "C3"), embedding = "UMAP", force = TRUE)
```

| addTSNE | *Add a TSNE embedding of a reduced dimensions object to an ArchRProject* |
|---|---|

## Description

This function will compute a TSNE embedding and add it to an ArchRProject.

## Usage

```
addTSNE(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  method = "RTSNE",
  name = "TSNE",
  perplexity = 50,
  maxIterations = 1000,
  learningRate = 200,
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75,
  saveModel = FALSE,
  verbose = TRUE,
  seed = 1,
  force = FALSE,
  threads = max(floor(getArchRThreads()/2), 1),
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to use from the designated `ArchRProject`. |
| `method` | The method for computing a TSNE embedding to add to the `ArchRProject` object. Possible options are "RTSNE", which uses `Rtsne::Rtsne()`, and "FFRTSNE", which uses `Seurat::RunTSNE()`. |
| `name` | The name for the TSNE embedding to store in the given `ArchRProject` object. |
| `perplexity` | An integer describing the number of nearest neighbors to compute an `Rtsne`. This argument is passed to `perplexity` in `Rtsne::Rtsne()`. |
| `maxIterations` | An integer describing the maximum number of iterations when computing a TSNE. This argument is passed to `max_iter` in `Rtsne::Rtsne()`. |
| `learningRate` | An integer controlling how much the weights are adjusted at each iteration. This argument is passed to `eta` in `Rtsne::Rtsne()`. |

| | |
|---|---|
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in computing the embedding. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `verbose` | A boolean value that indicates whether printing TSNE output. |
| `seed` | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `force` | A boolean value that indicates whether to overwrite the relevant data in the `ArchRProject` object if the embedding indicated by `name` already exists. |
| `threads` | The number of threads to be used for parallel computing. |
| `...` | Additional parameters for computing the TSNE embedding to pass to `Rtsne::Rtsne()` (when `method = "RTSNE"`) or to `Seurat::RunTSNE()` (when method = "FFRTSNE"). |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add UMAP for Small Project
proj <- addTSNE(proj, force = TRUE)
```

---

`addUMAP` *Add a UMAP embedding of a reduced dimensions object to an ArchRProject*

---

## Description

This function will compute a UMAP embedding and add it to an ArchRProject.

## Usage

```
addUMAP(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  name = "UMAP",
  nNeighbors = 40,
  minDist = 0.4,
  metric = "cosine",
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75,
  sampleCells = NULL,
  outlierQuantile = 0.9,
  saveModel = TRUE,
  verbose = TRUE,
  seed = 1,
  force = FALSE,
  threads = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to use from the designated `ArchRProject`. |
| `name` | The name for the UMAP embedding to store in the given `ArchRProject` object. |
| `nNeighbors` | An integer describing the number of nearest neighbors to compute a UMAP. This argument is passed to `n_neighbors` in `uwot::umap()`. |
| `minDist` | A number that determines how tightly the UMAP is allowed to pack points together. This argument is passed to `min_dist` in `uwot::umap()`. For more info on this see https://jlmelville.github.io/uwot/abparams.html. |
| `metric` | A number that determines how distance is computed in the `reducedDims` to compute a UMAP. This argument is passed to `metric` in `uwot::umap()`. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in computing the embedding. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `sampleCells` | An integer specifying the number of cells to subsample and perform UMAP Embedding on. The remaining cells that were not subsampled will be re-projected using uwot::umap_transform to the UMAP Embedding. This enables a decrease in run time and memory but can lower the overal quality of the UMAP Embedding. Only recommended for extremely large number of cells. |

| | |
|---|---|
| outlierQuantile | A numeric (0 to 1) describing the distance quantile in the subsampled cels (see sampleCells) to use to filter poor quality re-projections. This is necessary because there are lots of outliers if undersampled significantly. |
| saveModel | A boolean value indicating whether or not to save the UMAP model in an RDS file for downstream usage such as projection of data into the UMAP embedding. |
| verbose | A boolean value that indicates whether printing UMAP output. |
| seed | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| force | A boolean value that indicates whether to overwrite the relevant data in the ArchRProject object if the embedding indicated by name already exists. |
| threads | The number of threads to be used for parallel computing. Default set to 1 because if set to high can cause C stack usage errors. |
| ... | Additional parameters to pass to uwot::umap() |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add UMAP for Small Project
proj <- addUMAP(proj, force = TRUE)
```

---

## ArchRBrowser *Launch ArchR Genome Browser*

---

### Description

This function will open an interactive shiny session in style of a browser track. It allows for normalization of the signal which enables direct comparison across samples. Note that the genes displayed in this browser are derived from your geneAnnotation (i.e. the BSgenome object you used) so they may not match other online genome browsers that use different gene annotations.

### Usage

```
ArchRBrowser(
  ArchRProj = NULL,
  features = getPeakSet(ArchRProj),
  loops = getCoAccessibility(ArchRProj),
  sampleLabels = "Sample",
  minCells = 25,
  baseSize = 10,
  borderWidth = 0.5,
  tickWidth = 0.5,
  facetbaseSize = 12,
  geneAnnotation = getGeneAnnotation(ArchRProj),
  browserTheme = "cosmo",
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("ArchRBrowser")
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| features | A GRanges object containing the "features" to be plotted via the "featureTrack". This should be thought of as a bed track. i.e. the set of peaks obtained using getPeakSet(ArchRProj)). |
| loops | A GRanges object containing the "loops" to be plotted via the "loopTrack". This GRanges object start represents the center position of one loop anchor and the end represents the center position of another loop anchor. A "loopTrack" draws an arc between two genomic regions that show some type of interaction. This type of track can be used to display chromosome conformation capture data or co-accessibility links obtained using getCoAccessibility(). |
| sampleLabels | The name of a column in cellColData to use to identify samples. In most cases, this parameter should be left as NULL and you should only use this parameter if you do not want to use the default sample labels stored in cellColData$Sample. However, if your individual Arrow files do not map to individual samples, then you should set this parameter to accurately identify your samples. This is the case in (for example) multiplexing applications where cells from different biological samples are mixed into the same reaction and demultiplexed based on a lipid barcode or genotype. |
| minCells | The minimum number of cells contained within a cell group to allow for this cell group to be plotted. This argument can be used to exclude pseudo-bulk replicates generated from low numbers of cells. |
| baseSize | The numeric font size to be used in the plot. This applies to all plot labels. |
| borderWidth | The numeric line width to be used for plot borders. |
| tickWidth | The numeric line width to be used for axis tick marks. |
| facetbaseSize | The numeric font size to be used in the facets (gray boxes used to provide track labels) of the plot. |
| geneAnnotation | The geneAnnotation object to be used for plotting the "geneTrack" object. See createGeneAnnotation() for more info. |
| browserTheme | A shinytheme from shinythemes for viewing the ArchR Browser. If not installed this will be NULL. To install try devtools::install_github("rstudio/shinythemes"). |
| threads | The number of threads to use for parallel execution. |
| verbose | A boolean value that determines whether standard output should be printed. |
| logFile | The path to a file to be used for logging ArchR output. |

## Examples

```
proj <- getTestProject()

#Launch Browser with `ArchRBrowser(proj)`
```

---

`ArchRHeatmap` *Plot Nice Lookng Heatmap Using Complex Heatmap*

---

## Description

Plot Nice Lookng Heatmap Using Complex Heatmap

## Usage

```
ArchRHeatmap(
  mat = NULL,
  scale = FALSE,
  limits = c(min(mat), max(mat)),
  colData = NULL,
  color = paletteContinuous(set = "solarExtra", n = 100),
  clusterCols = TRUE,
  clusterRows = FALSE,
  labelCols = FALSE,
  labelRows = FALSE,
  colorMap = NULL,
  useRaster = TRUE,
  rasterQuality = 5,
  split = NULL,
  fontSizeRows = 10,
  fontSizeCols = 10,
  fontSizeLabels = 8,
  colAnnoPerRow = 4,
  showRowDendrogram = FALSE,
  showColDendrogram = FALSE,
  customRowLabel = NULL,
  customRowLabelIDs = NULL,
  customColLabel = NULL,
  customColLabelIDs = NULL,
  customLabelWidth = 0.75,
  rasterDevice = "png",
  padding = 45,
  borderColor = NA,
  draw = TRUE,
  name = "Heatmap"
)
```

**Arguments**

| | |
|---|---|
| `mat` | A matrix to plot heatmap from. |
| `scale` | A boolean whether to convert to row Z-scores |
| `limits` | A vector of two values describing min and mix limits to plot |
| `colData` | A DataFrame matching the columns of the input matrix to overlay on the columns of the heatmap |
| `color` | A palette to use for heatmap continuous color scheme. See `paletteContinuous`. |
| `clusterCols` | A boolean describing whether to cluster columns for heatmap. |
| `clusterRows` | A boolean describing whether to cluster rows for heatmap. |
| `colorMap` | A list of color mappings matching the column names in colData. |
| `useRaster` | A boolean whether to use rastering when plotting heatmap. |
| `rasterQuality` | Raster resolution of raster. Default is set to 5 higher numbers increase resolution. |
| `split` | A vector of groupings that will split the rows of heatmap (must be equal to nrow(mat)). |
| `fontSizeRows` | A numeric value representing the font size for rownames. |
| `fontSizeCols` | A numeric value representing the font size for colnames. |
| `fontSizeLabels` | A numeric value representing the font size for labels. |
| `colAnnoPerRow` | An integer value describing the number of column annotations per row in the legend. |
| `showRowDendrogram` | A boolean whether to show row dendrogram in heatmap. |
| `showColDendrogram` | A boolean whether to show column dendrogram in heatmap. |
| `customRowLabel` | A vector of indices to custom label from rows. |
| `customRowLabelIDs` | A vector of custom labels to overlay. Should match length of `customRowLabel`. |
| `customColLabel` | A vector of indices to custom label from columns. |
| `customColLabelIDs` | A vector of custom labels to overlay. Should match length of `customColLabel`. |
| `customLabelWidth` | A numeric describing the width of the column labels. |
| `rasterDevice` | Which device to use for rastering see `ComplexHeatmap`. |
| `padding` | A numeric (in cm) to pad the heatmap ie adding white space so that the final plot doesnt cutoff. |
| `borderColor` | A character representing the border color for each cell in the heatmap. |
| `draw` | A boolean whether to draw the heatmap immediately. If FALSE this will return a ComplexHeatmap object. |
| `name` | A character that will appear above color bar legend in heatmap. |

## ArchRPalettes — *List of color palettes that can be used in plots*

### Description

A collection of some original and some borrowed color palettes to provide appealing color aesthetics for plots in ArchR

### Usage

```
ArchRPalettes
```

### Format

An object of class `list` of length 30.

## ArchRProject — *Create ArchRProject from ArrowFiles*

### Description

This function will create an ArchRProject from the provided ArrowFiles.

### Usage

```
ArchRProject(
  ArrowFiles = NULL,
  outputDirectory = "ArchROutput",
  copyArrows = TRUE,
  geneAnnotation = getGeneAnnotation(),
  genomeAnnotation = getGenomeAnnotation(),
  showLogo = TRUE,
  threads = getArchRThreads()
)
```

### Arguments

| | |
|---|---|
| `ArrowFiles` | A character vector containing the relative paths to the ArrowFiles to be used. |
| `outputDirectory` | A name for the relative path of the outputDirectory for ArchR results. Relative to the current working directory. |
| `copyArrows` | A boolean value indicating whether ArrowFiles should be copied into `outputDirectory`. |
| `geneAnnotation` | The `geneAnnotation` object (see `createGeneAnnotation()`) to be used for downstream analyses such as calculating TSS Enrichment Scores, Gene Scores, etc. |

| | |
|---|---|
| genomeAnnotation | The genomeAnnotation object (see createGenomeAnnotation()) to be used for downstream analyses requiring genome information such as nucleotide information or chromosome sizes. |
| showLogo | A boolean value indicating whether to show the ascii ArchR logo after successful creation of an ArchRProject. |
| threads | The number of threads to use for parallel execution. |

## Examples

```
# Get Test Arrow
arrow <- getTestArrow()

# Create ArchR Project for Analysis
proj <- ArchRProject(arrow)
```

---

## confusionMatrix *Create a Confusion Matrix based on two value vectors*

---

### Description

This function creates a confusion matrix based on two value vectors.

### Usage

```
confusionMatrix(i = NULL, j = NULL)
```

### Arguments

| | |
|---|---|
| i | A character/numeric value vector to see concordance with j. |
| j | A character/numeric value vector to see concordance with i. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Overlap of Clusters and CellType
confusionMatrix(proj$Clusters, proj$CellType)

# Overlap of Cell Type and RNA Predict
confusionMatrix(proj$CellType, proj$predictedGroup_Un)
```

## correlateMatrices *Correlate Matrices within an ArchRProject*

### Description

This function will correlate 2 matrices within an ArchRProject by name matching.

### Usage

```
correlateMatrices(
  ArchRProj = NULL,
  useMatrix1 = NULL,
  useMatrix2 = NULL,
  useSeqnames1 = NULL,
  useSeqnames2 = NULL,
  removeFromName1 = c("underscore", "dash"),
  removeFromName2 = c("underscore", "dash"),
  log2Norm1 = TRUE,
  log2Norm2 = TRUE,
  reducedDims = "IterativeLSI",
  dimsToUse = 1:30,
  scaleDims = NULL,
  corCutOff = 0.75,
  k = 100,
  knnIteration = 500,
  overlapCutoff = 0.8,
  seed = 1,
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("correlateMatrices")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix1` | A character describing the first matrix to use. See `getAvailableMatrices` for valid options. |
| `useMatrix2` | A character describing the second matrix to use. See `getAvailableMatrices` for valid options. |
| `useSeqnames1` | A character vector describing which seqnames to use in matrix 1. |
| `useSeqnames2` | A character vector describing which seqnames to use in matrix 2. |
| `removeFromName1` | A character vector describing how to filter names in matrix 1. Options include "underscore", "dash", "numeric" and "dot". The string portion prior to these will be kept. |

| | |
|---|---|
| `removeFromName2` | A character vector describing how to filter names in matrix 2. Options include "underscore", "dash", "numeric" and "dot". The string portion prior to these will be kept. |
| `log2Norm1` | A boolean describing whether to log2 normalize matrix 1. |
| `log2Norm2` | A boolean describing whether to log2 normalize matrix 2. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to use from the designated `ArchRProject`. |
| `dimsToUse` | A vector containing the dimensions from the `reducedDims` object to use in computing the embedding. |
| `scaleDims` | A boolean value that indicates whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If set to `NULL` this will scale the dimensions based on the value of `scaleDims` when the `reducedDims` were originally created during dimensionality reduction. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded from analysis. |
| `k` | The number of k-nearest neighbors to use for creating single-cell groups for correlation analyses. |
| `knnIteration` | The number of k-nearest neighbor groupings to test for passing the supplied `overlapCutoff`. |
| `overlapCutoff` | The maximum allowable overlap between the current group and all previous groups to permit the current group be added to the group list during k-nearest neighbor calculations. |
| `seed` | A number to be used as the seed for random number generation required in knn determination. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Correlate Matrices
dfCor <- correlateMatrices(
  ArchRProj = proj,
  useMatrix1 = "GeneScoreMatrix",
  useMatrix2 = "GeneIntegrationMatrix",
  dimsToUse = 1:5,
  k = 20
)
```

## correlateTrajectories *Correlate Trajectories*

### Description

This function will correlate 2 trajectory matrices from getTrajectory.

### Usage

```
correlateTrajectories(
  seTrajectory1 = NULL,
  seTrajectory2 = NULL,
  corCutOff = 0.5,
  varCutOff1 = 0.8,
  varCutOff2 = 0.8,
  removeFromName1 = c("underscore", "dash"),
  removeFromName2 = c("underscore", "dash"),
  useRanges = FALSE,
  fix1 = "center",
  fix2 = "start",
  maxDist = 250000,
  log2Norm1 = TRUE,
  log2Norm2 = TRUE,
  force = FALSE,
  logFile = createLogFile("correlateTrajectories")
)
```

### Arguments

| | |
|---|---|
| `seTrajectory1` | A `SummarizedExperiment` object that results from calling `getTrajectory()`. |
| `seTrajectory2` | A `SummarizedExperiment` object that results from calling `getTrajectory()`. |
| `corCutOff` | A numeric describing the cutoff for determining correlated features. |
| `varCutOff1` | The "Variance Quantile Cutoff" to be used for identifying the top variable features across `seTrajectory1`. Only features with a variance above the provided quantile will be retained. |
| `varCutOff2` | The "Variance Quantile Cutoff" to be used for identifying the top variable features across `seTrajectory2`. Only features with a variance above the provided quantile will be retained. |
| `removeFromName1` | A character vector describing how to filter names in matrix 1. Options include "underscore", "dash", "numeric" and "dot". The string portion prior to these will be kept. |

| | |
|---|---|
| `removeFromName2` | A character vector describing how to filter names in matrix 2. Options include "underscore", "dash", "numeric" and "dot". The string portion prior to these will be kept. |
| `useRanges` | A boolean describing whether to use range overlap matching for correlation analysis. |
| `fix1` | A character describing where to resize the coordinates of `seTrajectory1`. Options include "start", "center", "end". |
| `fix2` | A character describing where to resize the coordinates of `seTrajectory2`. Options include "start", "center", "end". |
| `maxDist` | A integer specifying the maximum distance between the coordinates of `seTrajectory1` and `seTrajectory2` for computing correlations. |
| `log2Norm1` | A boolean describing whether to log2 normalize `seTrajectory1`. |
| `log2Norm2` | A boolean describing whether to log2 normalize `seTrajectory2`. |
| `force` | A boolean value that determines whether analysis should continue if resizing coordinates in `seTrajectory1` or `seTrajectory2` does not align with the strandedness. Only when `useRanges = TRUE`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add Trajectory
proj <- addTrajectory(proj, trajectory = c("C1", "C2", "C3"), embedding = "UMAP", force = TRUE)

#Get Trajectories
seTraj1 <- getTrajectory(proj, useMatrix = "GeneScoreMatrix")
seTraj2 <- getTrajectory(proj, useMatrix = "GeneIntegrationMatrix")

#Correlate
corTraj <- correlateTrajectories(seTraj1, seTraj2, corCutOff = 0.35, varCutOff1 = 0.6, varCutOff2 = 0.6)
```

---

## `createArrowFiles` *Create Arrow Files*

---

## Description

This function will create ArrowFiles from input files. These ArrowFiles are the main constituent for downstream analysis in ArchR.

## Usage

```
createArrowFiles(
  inputFiles = NULL,
  sampleNames = names(inputFiles),
  outputNames = sampleNames,
  validBarcodes = NULL,
  geneAnnotation = getGeneAnnotation(),
  genomeAnnotation = getGenomeAnnotation(),
  minTSS = 4,
  minFrags = 1000,
  maxFrags = 1e+05,
  minFragSize = 10,
  maxFragSize = 2000,
  QCDir = "QualityControl",
  nucLength = 147,
  promoterRegion = c(2000, 100),
  TSSParams = list(),
  excludeChr = c("chrM", "chrY"),
  nChunk = 5,
  bcTag = "qname",
  gsubExpression = NULL,
  bamFlag = NULL,
  offsetPlus = 4,
  offsetMinus = -5,
  addTileMat = TRUE,
  TileMatParams = list(),
  addGeneScoreMat = TRUE,
  GeneScoreMatParams = list(),
  force = FALSE,
  threads = getArchRThreads(),
  parallelParam = NULL,
  subThreading = TRUE,
  verbose = TRUE,
  cleanTmp = TRUE,
  logFile = createLogFile("createArrows"),
  filterFrags = NULL,
  filterTSS = NULL
)
```

## Arguments

| | |
|---|---|
| `inputFiles` | A character vector containing the paths to the input files to use to generate the ArrowFiles. These files can be in one of the following formats: (i) scATAC tabix files, (ii) fragment files, or (iii) bam files. |
| `sampleNames` | A character vector containing the names to assign to the samples that correspond to the `inputFiles`. Each input file should receive a unique sample name. This list should be in the same order as `inputFiles`. |

| | |
|---|---|
| `outputNames` | The prefix to use for output files. Each input file should receive a unique output file name. This list should be in the same order as "inputFiles". For example, if the predix is "PBMC" the output file will be named "PBMC.arrow" |
| `validBarcodes` | A list of valid barcode strings to be used for filtering cells read from each input file (see `getValidBarcodes()` for 10x fragment files). |
| `geneAnnotation` | The geneAnnotation (see `createGeneAnnotation()`) to associate with the ArrowFiles. This is used downstream to calculate TSS Enrichment Scores etc. |
| `genomeAnnotation` | The genomeAnnotation (see `createGenomeAnnotation()`) to associate with the ArrowFiles. This is used downstream to collect chromosome sizes and nucleotide information etc. |
| `minTSS` | The minimum numeric transcription start site (TSS) enrichment score required for a cell to pass filtering for use in downstream analyses. Cells with a TSS enrichment score greater than or equal to `minTSS` will be retained. TSS enrichment score is a measurement of signal-to-background in ATAC-seq. |
| `minFrags` | The minimum number of mapped ATAC-seq fragments required per cell to pass filtering for use in downstream analyses. Cells containing greater than or equal to `minFrags` total fragments will be retained. |
| `maxFrags` | The maximum number of mapped ATAC-seq fragments required per cell to pass filtering for use in downstream analyses. Cells containing greater than or equal to `maxFrags` total fragments will be retained. |
| `minFragSize` | The minimum fragment size to be included into Arrow File. Fragments lower than this number are discarded. Must be less than maxFragSize. |
| `maxFragSize` | The maximum fragment size to be included into Arrow File. Fragments above than this number are discarded. Must be greater than maxFragSize. |
| `QCDir` | The relative path to the output directory for QC-level information and plots for each sample/ArrowFile. |
| `nucLength` | The length in basepairs that wraps around a nucleosome. This number is used for identifying fragments as sub-nucleosome-spanning, mono-nucleosome-spanning, or multi-nucleosome-spanning. |
| `promoterRegion` | A integer vector describing the number of basepairs upstream and downstream c(upstream, downstream) of the TSS to include as the promoter region for downstream calculation of things like the fraction of reads in promoters (FIP). |
| `TSSParams` | A list of parameters for computing TSS Enrichment scores. This includes the `window` which is the size in basepairs of the window centered at each TSS (default 101), the `flank` which is the size in basepairs of the flanking window (default 2000), and the `norm` which describes the size in basepairs of the flank window to be used for normalization of the TSS enrichment score (default 100). For example, given `window = 101, flank = 2000, norm = 100`, the accessibility within the 101-bp surrounding the TSS will be normalized to the accessibility in the 100-bp bins from -2000 bp to -1901 bp and 1901:2000. |
| `excludeChr` | A character vector containing the names of chromosomes to be excluded from downstream analyses. In most human/mouse analyses, this includes the mitochondrial DNA (chrM) and the male sex chromosome (chrY). This does, however, not exclude the corresponding fragments from being stored in the ArrowFile. |

| | |
|---|---|
| `nChunk` | The number of chunks to divide each chromosome into to allow for low-memory parallelized reading of the `inputFiles`. Higher numbers reduce memory usage but increase compute time. |
| `bcTag` | The name of the field in the input bam file containing the barcode tag information. See `ScanBam` in Rsamtools. |
| `gsubExpression` | A regular expression used to clean up the barcode tag string read in from a bam file. For example, if the barcode is appended to the readname or qname field like for the mouse atlas data from Cusanovic* and Hill* et al. (2018), the gsubExpression would be ":.*". This would retrieve the string after the colon as the barcode. |
| `bamFlag` | A vector of bam flags to be used for reading in fragments from input bam files. Should be in the format of a `scanBamFlag` passed to `ScanBam` in Rsamtools. |
| `offsetPlus` | The numeric offset to apply to the start (left-most Tn5 insertion) of a fragment to account for the precise Tn5 binding site. This parameter only applies to bam file input and it is assumed that fragment files have already been offset which is the standard from 10x output. See Buenrostro et al. Nature Methods 2013. |
| `offsetMinus` | The numeric offset to apply to the end (right-most Tn5 insertion) of a fragment to account for the precise Tn5 binding site. This parameter only applies to bam file input and it is assumed that fragment files have already been offset which is the standard from 10x output. See Buenrostro et al. Nature Methods 2013. |
| `addTileMat` | A boolean value indicating whether to add a "Tile Matrix" to each ArrowFile. A Tile Matrix is a counts matrix that, instead of using peaks, uses a fixed-width sliding window of bins across the whole genome. This matrix can be used in many downstream ArchR operations. |
| `TileMatParams` | A list of parameters to pass to the `addTileMatrix()` function. See `addTileMatrix()` for options. |
| `addGeneScoreMat` | A boolean value indicating whether to add a Gene-Score Matrix to each ArrowFile. A Gene-Score Matrix uses ATAC-seq signal proximal to the TSS to estimate gene activity. |
| `GeneScoreMatParams` | A list of parameters to pass to the `addGeneScoreMatrix()` function. See `addGeneScoreMatrix()` for options. |
| `force` | A boolean value indicating whether to force ArrowFiles to be overwritten if they already exist. |
| `threads` | The number of threads to be used for parallel computing. |
| `parallelParam` | A list of parameters to be passed for biocparallel/batchtools parallel computing. |
| `subThreading` | A boolean determining whether possible use threads within each multi-threaded subprocess if greater than the number of input samples. |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `cleamTmp` | A boolean value that determines whether to clean temp folder of all intermediate ".arrow" files. |

## Examples

```
# Get Test Fragments
fragments <- getTestFragments()

# Create Arrow Files
arrowFiles <- createArrowFiles(
  inputFiles = fragments,
  sampleNames = "PBSmall",
  minFrags = 100,
  nChunk = 1,
  TileMatParams=list(tileSize=10000),
  force = TRUE
)
```

---

`createGeneAnnotation` *Create a gene annotation object for ArchR*

---

## Description

This function will create a gene annotation object that can be used for creating ArrowFiles or an ArchRProject, etc.

## Usage

```
createGeneAnnotation(
  genome = NULL,
  TxDb = NULL,
  OrgDb = NULL,
  genes = NULL,
  exons = NULL,
  TSS = NULL,
  annoStyle = NULL,
  singleStrand = TRUE
)
```

## Arguments

| | |
|---|---|
| `genome` | A string that specifies the genome (ie "hg38", "hg19", "mm10", "mm9"). If genome is not supplied, `TxDb` and `OrgDb` are required. If genome is supplied, `TxDb` and `OrgDb` will be ignored. |
| `TxDb` | A `TxDb` object (transcript database) from Bioconductor which contains information for gene/transcript coordinates. For example, from `txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene` |
| `OrgDb` | An `OrgDb` object (organism database) from Bioconductor which contains information for gene/transcript symbols from ids. For example, from `orgdb <- org.Hs.eg.db`. |

| | |
|---|---|
| `genes` | A `GRanges` object containing gene coordinates (start to end). Must have a symbols column matching the symbols column of `exons`. |
| `exons` | A `GRanges` object containing gene exon coordinates. Must have a symbols column matching the symbols column of `genes`. |
| `TSS` | A `GRanges` object containing standed transcription start site coordinates for computing TSS enrichment scores downstream. |
| `annoStyle` | annotation style to map between gene names and various gene identifiers e.g. "ENTREZID", "ENSEMBL". |
| `singleStrand` | A boolean for GenomicFeatures::genes(`single.strand.genes.only`) parameter |

### Examples

```
if (!require("TxDb.Hsapiens.UCSC.hg19.knownGene", quietly = TRUE)) BiocManager::install("TxDb.Hsapiens.UCSC.hg19
if (!require("org.Hs.eg.db", quietly = TRUE)) BiocManager::install("org.Hs.eg.db", update = FALSE)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)

# Get Txdb
TxDb <- TxDb.Hsapiens.UCSC.hg19.knownGene

# Get OrgDb
OrgDb <- org.Hs.eg.db

# Create Genome Annotation
geneAnno <- createGeneAnnotation(TxDb=TxDb, OrgDb=OrgDb)

# Also can create from a string if BSgenome exists
geneAnno <- createGeneAnnotation("hg19")
```

## createGenomeAnnotation

*Create a genome annotation object for ArchR*

### Description

This function will create a genome annotation object that can be used for creating ArrowFiles or an ArchRProject, etc.

### Usage

```
createGenomeAnnotation(
  genome = NULL,
  chromSizes = NULL,
  blacklist = NULL,
  filter = TRUE,
  filterChr = c("chrM")
)
```

### Arguments

| | |
|---|---|
| `genome` | Either (i) a string that is a valid `BSgenome` or (ii) a `BSgenome` object (ie "hg38" or "BSgenome.Hsapiens.UCSC.hg38"). |
| `chromSizes` | A `GRanges` object containing chromosome start and end coordinates. |
| `blacklist` | A `GRanges` object containing regions that should be excluded from analyses due to unwanted biases. |
| `filter` | A boolean value indicating whether non-standard chromosome scaffolds should be excluded. These "non-standard" chromosomes are defined by `filterChrGR()` and by manual annotation using the `filterChr` parameter. |
| `filterChr` | A character vector indicating the seqlevels that should be removed if manual removal is desired for certain seqlevels. If no manual removal is desired, `filterChr` should be set to `NULL`. If `filter` is set to `TRUE` but `filterChr` is set to `NULL`, non-standard chromosomes will still be removed as defined in `filterChrGR()`. |

### Examples

```
if (!require("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)) BiocManager::install("BSgenome.Hsapiens.UCSC.hg19",
library(BSgenome.Hsapiens.UCSC.hg19)

# Get Genome
genome <- BSgenome.Hsapiens.UCSC.hg19

# Create Genome Annotation
genomeAnno <- createGenomeAnnotation(genome)

# Also can create from a string if BSgenome exists
genomeAnno <- createGenomeAnnotation("hg19")
```

---

## `createLogFile` *Create a Log File for ArchR*

---

### Description

This function will create a log file for ArchR functions. If ArchRLogging is not TRUE this function will return NULL.

### Usage

```
createLogFile(name = NULL, logDir = "ArchRLogs", useLogs = getArchRLogging())
```

### Arguments

| | |
|---|---|
| `name` | A character string to add a more descriptive name in log file. |
| `logDir` | The path to a directory where log files should be written. |

## Examples

```
# Create Log File
createLogFile(name = "test")
```

---

`customEnrichment` *Hypergeometric Enrichment in input peak ranges.*

---

## Description

This function will perform hypergeometric enrichment of a given peak matches object and ranges.

## Usage

```
customEnrichment(ranges = NULL, matches = NULL, bgdPeaks = NULL)
```

## Arguments

| | |
|---|---|
| `ranges` | A `GenomicRanges` object of peaks/regions to overlap with peaks. |
| `matches` | A custom `peakAnnotation` matches object used as input for the hypergeometric test. See `motifmatchr::matchmotifs()` for additional information. |
| `bgdPeaks` | A `SummarizedExperiment` of background peaks from `getBgdPeaks` can be NULL for using all peaks. |

## Examples

```
#Project
proj <- getTestProject()

#Get Peaks
peaks <- getPeakSet(proj)

#Custom C1 Mono
peaks1 <- peaks[names(peaks)=="C1"]

#All Peaks
customEnrichment(
  ranges = peaks1,
  matches = getMatches(proj)
)
#         feature CompareFrequency nCompare CompareProportion BackgroundFrequency
# CEBPB_1 CEBPB_1               70      635        0.11023622                 122
# CEBPA_2 CEBPA_2               81      635        0.12755906                 156
# IRF4_4   IRF4_4              103      635        0.16220472                 276
# EOMES_6 EOMES_6               36      635        0.05669291                 120
# ETS1_3   ETS1_3               38      635        0.05984252                 149
# PAX5_5   PAX5_5               23      635        0.03622047                 124
#         nBackground BackgroundProporition Enrichment mlog10p mlog10Padj
# CEBPB_1        2142           0.05695612  1.9354589 10.3279   9.373657
# CEBPA_2        2142           0.07282913  1.7514839  8.9394   7.985157
# IRF4_4         2142           0.12885154  1.2588497  2.6938   1.739557
# EOMES_6        2142           0.05602241  1.0119685  0.3001   0.000000
# ETS1_3         2142           0.06956116  0.8602864  0.0487   0.000000
# PAX5_5         2142           0.05788982  0.6256795  0.0006   0.000000

#Background Peaks
customEnrichment(
 ranges = peaks1,
  matches = getMatches(proj),
 bgdPeaks = getBgdPeaks(proj, force=TRUE)
)
#         feature CompareFrequency nCompare CompareProportion BackgroundFrequency
# CEBPB_1 CEBPB_1               70      635        0.11023622                2459
# CEBPA_2 CEBPA_2               81      635        0.12755906                3154
# IRF4_4   IRF4_4              103      635        0.16220472                4836
# ETS1_3   ETS1_3               38      635        0.05984252                1781
# EOMES_6 EOMES_6               36      635        0.05669291                1975
# PAX5_5   PAX5_5               23      635        0.03622047                1495
#         nBackground BackgroundProporition Enrichment mlog10p mlog10Padj
# CEBPB_1       32385           0.07593021  1.4518097  2.9544   2.000157
# CEBPA_2       32385           0.09739077  1.3097654  2.1341   1.179857
# IRF4_4        32385           0.14932839  1.0862283  0.7142   0.000000
# ETS1_3        32385           0.05499460  1.0881527  0.4975   0.000000
# EOMES_6       32385           0.06098502  0.9296203  0.1551   0.000000
# PAX5_5        32385           0.04616335  0.7846154  0.0422   0.000000
#
```

## exportPeakMatrixForSTREAM

*Get a PeakMatrix stored in an ArchRProject and write out for STREAM*

### Description

This function gets a PeakMatrix from an `ArchRProject` and writes it to a set of files for STREAM (https://github.com/pinellolab/STREAM)

### Usage

```
exportPeakMatrixForSTREAM(
  ArchRProj = NULL,
  useSeqnames = NULL,
  verbose = TRUE,
  binarize = FALSE,
  threads = getArchRThreads(),
  logFile = createLogFile("exportMatrixForSTREAM")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object to get data matrix from. |
| `useSeqnames` | A character vector of chromosome names to be used to subset the data matrix being obtained. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `binarize` | A boolean value indicating whether the matrix should be binarized before return. This is often desired when working with insertion counts. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Export Matrix For Stream
exportPeakMatrixForSTREAM(proj)
```

---

`extendGR` *Extend regions from a Genomic Ranges object*

---

## Description

This function extends each region in a Genomic Ranges object by a designated upstream and downstream extension in a strand-aware fashion

## Usage

```
extendGR(gr = NULL, upstream = NULL, downstream = NULL)
```

## Arguments

| | |
|---|---|
| `gr` | A `GRanges` object. |
| `upstream` | The number of basepairs upstream (5') to extend each region in `gr` in a strand-aware fashion. |
| `downstream` | The number of basepairs downstream (3') to extend each region in `gr` in a strand-aware fashion. |

### Examples

```
# Dummy GR
gr <- GRanges(
  seqnames = "chr1",
  ranges = IRanges(
    start = c(1, 4, 11),
    end = c(10, 12, 20)
  ),
  score = c(1, 2, 3)
)

# Non Overlapping
extendGR(gr, 1, 2)
```

---

## filterChrGR *Filters unwanted seqlevels from a Genomic Ranges object or similar object*

---

### Description

This function allows for removal of manually designated or more broadly undesirable seqlevels from a Genomic Ranges object or similar object

### Usage

```
filterChrGR(
  gr = NULL,
  remove = NULL,
  underscore = TRUE,
  standard = TRUE,
  pruningMode = "coarse"
)
```

### Arguments

| | |
|---|---|
| `gr` | A `GRanges` object or another object containing `seqlevels`. |
| `remove` | A character vector indicating the seqlevels that should be removed if manual removal is desired for certain seqlevels. If no manual removal is desired, `remove` should be set to `NULL`. |
| `underscore` | A boolean value indicating whether to remove all seqlevels whose names contain an underscore (for example "chr11_KI270721v1_random"). |
| `standard` | A boolean value indicating whether only standard chromosomes should be kept. Standard chromosomes are defined by `GenomeInfoDb::keepStandardChromosomes()`. |

| | |
|---|---|
| pruningMode | The name of the pruning method to use (fromGenomeInfoDb::seqinfo()) when seqlevels must be removed from a GRanges object. When some of the seqlevels to drop from the given GRanges object are in use (i.e. have ranges on them), the ranges on these sequences need to be removed before the seqlevels can be dropped. Four pruning modes are currently defined: "error", "coarse", "fine", and "tidy". |

## Examples

```
# Add ArchR Genome
addArchRGenome("hg19test2")

# Filter Chr
filterChrGR(getChromSizes(), remove = "chr5")
```

---

## filterDoublets *Filter Doublets From an ArchRProject*

---

## Description

This function will filter doublets from an ArchRProject after addDoubletScores() has been run.

## Usage

```
filterDoublets(
  ArchRProj = NULL,
  cutEnrich = 1,
  cutScore = -Inf,
  filterRatio = 1
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| cutEnrich | The minimum numeric cutoff for DoubletEnrichment. This number is equivalent to the number of simulated doublets identified as a nearest neighbor to the cell divided by the expected number given a random uniform distribution. |
| cutScore | The minimum numeric cutoff for DoubletScore which represents the -log10(binomial adjusted p-va for the DoubletEnrichment. |
| filterRatio | The maximum ratio of predicted doublets to filter based on the number of pass-filter cells. For example, if there are 5000 cells, the maximum would be filterRatio * 5000^2 / (100000) (which simplifies to filterRatio * 5000 * 0.05). This filterRatio allows you to apply a consistent filter across multiple different samples that may have different percentages of doublets because they were run with different cell loading concentrations. The higher the filterRatio, the greater the number of cells potentially removed as doublets. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Doublet Scores for Small Project
proj <- addDoubletScores(proj, dimsToUse = 1:5, LSIParams = list(dimsToUse = 1:5, varFeatures=1000, iterations = 2)

# Filter Doublets (Since Low Cells filterRatio has to be high before removing 1 cell!)
proj <- filterDoublets(proj, filterRatio=10)
```

## findMacs2 *Find the installed location of the MACS2 executable*

### Description

This function attempts to find the path to the MACS2 executable by serting the path and python's pip.

### Usage

```
findMacs2()
```

### Examples

```
# Get Macs2
findMacs2()
```

## getArchRChrPrefix *Get a globally-applied requirement for chromosome prefix*

### Description

This function will get the default requirement of chromosomes to have a "chr" prefix.

### Usage

```
getArchRChrPrefix()
```

### Examples

```
# Get ArchR Chr Prefix
getArchRChrPrefix()
```

## `getArchRDebugging` *Get ArchR Debugging*

### Description

This function will get ArchR Debugging which will save an RDS if an error is encountered.

### Usage

```
getArchRDebugging()
```

### Examples

```
# Get ArchR Debugging
getArchRDebugging()
```

## `getArchRGenome` *Get the globally defined genome, the geneAnnotation, or genomeAnnotation objects associated with the globally defined genome.*

### Description

This function will retrieve the genome that is currently in use by ArchR. Alternatively, this function can return either the `geneAnnotation` or the `genomeAnnotation` associated with the globally defined genome if desired.

### Usage

```
getArchRGenome(geneAnnotation = FALSE, genomeAnnotation = FALSE)
```

### Arguments

| | |
|---|---|
| `geneAnnotation` | A boolean value indicating whether the `geneAnnotation` associated with the ArchRGenome should be returned instead of the globally defined genome. The `geneAnnotation` is used downstream to calculate things like TSS Enrichment Scores. This function is not meant to be run with both `geneAnnotation` and `genomeAnnotation` set to `TRUE` (it is an either/or return value). |
| `genomeAnnotation` | A boolean value indicating whether the `genomeAnnotation` associated with the ArchRGenome should be returned instead of the globally defined genome. The `genomeAnnotation` is used downstream to determine things like chromosome sizes and nucleotide content. This function is not meant to be run with both `geneAnnotation` and `genomeAnnotation` set to `TRUE` (it is an either/or return value). |

### Examples

```
# Get ArchR Genome to use globally
getArchRGenome()
```

---

## getArchRH5Level *Get globally-applied compression level for h5 files*

---

### Description

This function will get the default compression level to be used for h5 file execution across all ArchR functions.

### Usage

```
getArchRH5Level()
```

### Examples

```
# Get ArchR H5 Compression level
getArchRH5Level()
```

---

## getArchRLogging *Get ArchR Logging*

---

### Description

This function will get ArchR logging

### Usage

```
getArchRLogging()
```

### Examples

```
# Get ArchR Logging
getArchRLogging()
```

## getArchRThreads — *Get globally-applied number of threads to use for parallel computing.*

### Description

This function will get the number of threads to be used for parallel execution across all ArchR functions.

### Usage

```
getArchRThreads()
```

### Examples

```
# Get ArchR Threads
getArchRThreads()
```

## getArchRVerbose — *Set ArchR Verbosity for Log Messaging*

### Description

This function will get ArchR logging verbosity.

### Usage

```
getArchRVerbose()
```

### Examples

```
# Get ArchR Verbose
addArchRVerbose()
```

## getArrowFiles *Get ArrowFiles from an ArchRProject*

### Description

This function gets the names of all ArrowFiles associated with a given ArchRProject.

### Usage

```
getArrowFiles(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Arrow Files
getArrowFiles(proj)
```

## getAvailableMatrices *Get a list available matrices in the ArrowFiles storted in an ArchRProject*

### Description

This function gets the available matrices from the ArrowFiles in a given ArchRProject object.

### Usage

```
getAvailableMatrices(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Available Matrices in ArchR Project
getAvailableMatrices(proj)
```

## getBgdPeaks — *Get Background Peaks from an ArchRProject*

### Description

This function will get/compute background peaks controlling for total accessibility and GC-content from an ArchRProject.

### Usage

```
getBgdPeaks(
  ArchRProj = NULL,
  nIterations = 50,
  w = 0.1,
  binSize = 50,
  seed = 1,
  method = "chromVAR",
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `nIterations` | The number of background peaks to sample. See `chromVAR::getBackgroundPeaks()`. |
| `w` | The parameter controlling similarity measure of background peaks. See `chromVAR::getBackgroundPeak` |
| `binSize` | The precision with which the similarity is computed. See `chromVAR::getBackgroundPeaks()`. |
| `seed` | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| `method` | A string indicating whether to use chromVAR or ArchR for background peak identification. |
| `force` | A boolean value indicating whether to force the file indicated by `outFile` to be overwritten if it already exists. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Background Peaks
bgdPeaks <- getBgdPeaks(proj, force = TRUE)
```

## getBlacklist *Get the blacklist from an ArchRProject*

### Description

This function gets the blacklist (the regions to be excluded from analysis) as a GRanges object from the genomeAnnotation of a given ArchRProject.

### Usage

```
getBlacklist(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Blacklist in ArchRProj
getBlacklist(proj)

# Get Blacklist loaded globally
getBlacklist()
```

## getCellColData *Get cellColData from an ArchRProject*

### Description

This function gets the cellColData from a given ArchRProject.

### Usage

```
getCellColData(ArchRProj = NULL, select = NULL, drop = FALSE)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `select` | A character vector of column names to select from `cellColData` if you would like to subset the returned data. |
| `drop` | A boolean value that indicates whether to drop the `dataframe` structure and convert to a vector if selecting only one column. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Cell Column Data
getCellColData(proj)
```

## getCellNames *Get cellNames from an ArchRProject*

### Description

This function gets the cellNames from a given ArchRProject object.

### Usage

```
getCellNames(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Cell Names
getCellNames(proj)
```

## getChromLengths *Get chromLengths from ArchRProject*

### Description

This function gets the chromosome lengths as a vector from the genomeAnnotation of a given ArchRProject.

### Usage

```
getChromLengths(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get chromLengths in ArchRProj
getChromLengths(proj)

# Get chromLengths loaded globally
getChromLengths()
```

---

## getChromSizes *Get chromSizes from ArchRProject*

---

### Description

This function gets the chromosome lengths as a GRanges object from the genomeAnnotation of a given ArchRProject.

### Usage

```
getChromSizes(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get chromSizes in ArchRProj
getChromSizes(proj)

# Get chromSizes loaded globally
getChromSizes()
```

## `getCoAccessibility` *Get the peak co-accessibility from an ArchRProject*

### Description

This function obtains co-accessibility data from an ArchRProject.

### Usage

```
getCoAccessibility(
  ArchRProj = NULL,
  corCutOff = 0.5,
  resolution = 1,
  returnLoops = TRUE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `corCutOff` | A numeric describing the minimum numeric peak-to-peak correlation to return. |
| `resolution` | A numeric describing the bp resolution to use when returning loops. This helps with overplotting of correlated regions. This only takes affect if `returnLoops = TRUE`. |
| `returnLoops` | A boolean indicating to return the co-accessibility signal as a `GRanges` "loops" object designed for use with the `ArchRBrowser()` or as an `ArchRBrowserTrack()`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Co Accessibility
proj <- addCoAccessibility(proj, k = 20)

# Get Co Accessibility
CoA <- getCoAccessibility(proj)
```

## `getEmbedding` *Get embedding information stored in an ArchRProject*

### Description

This function gets an embedding (i.e. UMAP) from a given ArchRProject.

## Usage

```
getEmbedding(ArchRProj = NULL, embedding = "UMAP", returnDF = TRUE)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `embedding` | The name of the `embeddings` object (i.e. UMAP, TSNE see `embeddingOut` of the `addEmbeddings()` function) to retrieve from the designated `ArchRProject`. |
| `returnDF` | A boolean value indicating whether to return the embedding object as a `data.frame`. Otherwise, it will return the full embedding object. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get UMAP
getEmbedding(proj, embedding = "UMAP")
```

---

## `getExons` *Get the exons from an ArchRProject*

---

## Description

This function gets the exons coordinates as a GRanges object from the geneAnnotation of a given ArchRProject.

## Usage

```
getExons(ArchRProj = NULL, symbols = NULL)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `symbols` | A character vector containing the gene symbols for the genes where exons should be extracted. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Exons in ArchRProj
getExons(proj)

# Get Exons globally
getExons()
```

## getFeatures — *Get the features that could be selected from a given data matrix within an ArchRProject*

### Description

This function will identify available features from a given data matrix (i.e. "GeneScoreMatrix", or "TileMatrix") and return them for downstream plotting utilities.

### Usage

```
getFeatures(
  ArchRProj = NULL,
  useMatrix = "GeneScoreMatrix",
  select = NULL,
  ignoreCase = TRUE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the data matrix as stored in the ArrowFiles of the `ArchRProject`. Options include "TileMatrix", "GeneScoreMatrix", etc. |
| `select` | A string specifying a specific feature name (or rowname) to be found with `grep` or granges to overlap. |
| `ignoreCase` | A boolean value indicating whether to ignore the case (upper-case / lower-case) when searching via grep for the string passed to `select`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Features
getFeatures(proj, useMatrix = "GeneScoreMatrix", select = 'CD3')
```

## getFootprints — *Calculate footprints from an ArchRProject*

### Description

This function will get footprints for all samples in a given ArchRProject and return a summarized experiment object that can be used for downstream analyses

## Usage

```
getFootprints(
  ArchRProj = NULL,
  positions = NULL,
  plotName = "Plot-Footprints",
  groupBy = "Clusters",
  useGroups = NULL,
  flank = 250,
  minCells = 25,
  nTop = NULL,
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("getFootprints")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `positions` | A `list` or `GRangesList` of `GRanges` containing the positions to incorporate into the footprint. Each position should be stranded. |
| `plotName` | The prefix to add to the file name for the output PDF file containing the footprint plots. |
| `groupBy` | The name of the column in `cellColData` used in the `addGroupCoverages()` function for grouping multiple cells together. |
| `useGroups` | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData`. This limits the groups used to perform footprinting. |
| `flank` | The number of basepairs from the position center (+/-) to consider as the flank. |
| `minCells` | The minimum number of cells required in a given cell group to permit footprint generation. |
| `nTop` | The number of genomic regions to consider. Only the top `nTop` genomic regions based on the "score" column in the `GRanges` object will be considered for the footprint. |
| `threads` | The number of threads to be used for parallel computing. |
| `verbose` | A boolean value that determines whether standard output includes verbose sections. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Motif Positions
positions <- getPositions(proj)
```

```
# Get Footprints
seFoot <- getFootprints(ArchRProj = proj, positions = positions, groupBy = "Clusters", minCells = 10)
```

## getFragmentsFromArrow *Get the fragments from an ArrowFile*

### Description

This function retrieves the fragments from a given ArrowFile as a GRanges object.

### Usage

```
getFragmentsFromArrow(
  ArrowFile = NULL,
  chr = NULL,
  cellNames = NULL,
  verbose = TRUE,
  logFile = createLogFile("getFragmentsFromArrow")
)
```

### Arguments

| | |
|---|---|
| `ArrowFile` | The path to the ArrowFile from which fragments should be obtained. |
| `chr` | A name of a chromosome to be used to subset the fragments `GRanges` object to a specific chromsome if desired. |
| `cellNames` | A character vector indicating the cell names of a subset of cells from which fragments whould be extracted. This allows for extraction of fragments from only a subset of selected cells. By default, this function will extract all cells from the provided ArrowFile using `getCellNames()`. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to `FALSE` for a cleaner output. |
| `logFile` | The path to a file to be used for logging ArchR output. |

### Examples

```
#Get Test Arrow
arrow <- getTestArrow()

# Get Fragments
frags <- getFragmentsFromArrow(arrow)
```

## getFragmentsFromProject

*Get the fragments from an ArchRProject*

### Description

This function retrieves the fragments from a given ArchRProject as a GRangesList object.

### Usage

```
getFragmentsFromProject(
  ArchRProj = NULL,
  subsetBy = NULL,
  cellNames = NULL,
  verbose = FALSE,
  logFile = createLogFile("getFragmentsFromProject")
)
```

### Arguments

| | |
|---|---|
| `subsetBy` | A Genomic Ranges object to subset fragments by. |
| `cellNames` | A character vector indicating the cell names of a subset of cells from which fragments whould be extracted. This allows for extraction of fragments from only a subset of selected cells. By default, this function will extract all cells from the provided ArrowFile using `getCellNames()`. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to `FALSE` for a cleaner output. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `ArchRProject` | An `ArchRProject` object to get fragments from. |

### Examples

```
#Get Test Project
proj <- getTestProject()

# Get Fragments
frags <- getFragmentsFromProject(proj)
```

## getGeneAnnotation *Get geneAnnotation from an ArchRProject*

### Description

This function gets the geneAnnotation from a given ArchRProject

### Usage

```
getGeneAnnotation(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Gene Annotation in ArchRProj
getGeneAnnotation(proj)

# Get Gene Annotation loaded globally
getGeneAnnotation()
```

## getGenes *Get the genes from an ArchRProject*

### Description

This function gets the genes start and end coordinates as a GRanges object from the geneAnnotation of a given ArchRProject.

### Usage

```
getGenes(ArchRProj = NULL, symbols = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `symbols` | A character vector containing the gene symbols to subset from the `geneAnnotation`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Genes in ArchRProj
getGenes(proj)

# Get Genes globally
getGenes()
```

---

## getGenome *Get the genome used by an ArchRProject*

---

### Description

This function gets the name of the genome from the genomeAnnotation used by a given ArchRProject.

### Usage

```
getGenome(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Genome in ArchRProj
getGenome(proj)

# Get Genome loaded globally
getGenome()
```

## `getGenomeAnnotation` *Get the genomeAnnotation from an ArchRProject*

### Description

This function gets the genomeAnnotation from a given ArchRProject.

### Usage

```
getGenomeAnnotation(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Genome Annotation in ArchRProj
getGenomeAnnotation(proj)

# Get Genome Annotation loaded globally
getGenomeAnnotation()
```

## `getGroupBW` *Export Group BigWigs*

### Description

This function will group, summarize and export a bigwig for each group in an ArchRProject.

### Usage

```
getGroupBW(
  ArchRProj = NULL,
  groupBy = "Sample",
  normMethod = "ReadsInTSS",
  tileSize = 100,
  maxCells = 1000,
  ceiling = 4,
  verbose = TRUE,
  threads = getArchRThreads(),
  logFile = createLogFile("getGroupBW")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | A string that indicates how cells should be grouped. This string corresponds to one of the standard or user-supplied `cellColData` metadata columns (for example, "Clusters"). Cells with the same value annotated in this metadata column will be grouped together and the average signal will be plotted. |
| `normMethod` | The name of the column in `cellColData` by which normalization should be performed. The recommended and default value is "ReadsInTSS" which simultaneously normalizes tracks based on sequencing depth and sample data quality. Accepted values are "None", "ReadsInTSS", "nCells", "ReadsInPromoter", or "nFrags". |
| `tileSize` | The numeric width of the tile/bin in basepairs for plotting ATAC-seq signal tracks. All insertions in a single bin will be summed. |
| `maxCells` | Maximum number of cells used for each bigwig. |
| `ceiling` | Maximum contribution of accessibility per cell in each tile. |
| `verbose` | A boolean specifying to print messages during computation. |
| `threads` | An integer specifying the number of threads for parallel. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Group BW
bw <- getGroupBW(proj, groupBy = "Clusters")
```

---

## getGroupFragments *Export Group Fragment Files*

---

## Description

This function will group export fragment files for each group in an ArchRProject.

## Usage

```
getGroupFragments(
  ArchRProj = NULL,
  groupBy = "Clusters",
  threads = getArchRThreads(),
  logFile = createLogFile("getGroupFragments")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | A string that indicates how cells should be grouped. This string corresponds to one of the standard or user-supplied `cellColData` metadata columns (for example, "Clusters"). Cells with the same value annotated in this metadata column will be grouped together and their fragments exported to `outputDirectory`/GroupFragments. |
| `threads` | An integer specifying the number of threads for parallel. |
| `logFile` | The path to a file to be used for logging ArchR output. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Group BW
frags <- getGroupFragments(proj, groupBy = "Clusters")
```

## getGroupSE *Export Group Summarized Experiment*

### Description

This function will group, summarize and export a summarized experiment for a assay in a ArchRProject.

### Usage

```
getGroupSE(
  ArchRProj = NULL,
  useMatrix = NULL,
  groupBy = "Sample",
  divideN = TRUE,
  scaleTo = NULL,
  threads = getArchRThreads(),
  verbose = TRUE,
  logFile = createLogFile("getGroupSE")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the matrix in the ArrowFiles. See getAvailableMatrices to see options |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for summarizing. |

| | |
|---|---|
| `divideN` | A boolean describing whether to divide by the number of cells. |
| `scaleTo` | Depth normalize to this value if not NULL. |
| `threads` | An integer specifying the number of threads for parallel. |
| `verbose` | A boolean specifying to print messages during computation. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Group SE
se <- getGroupSE(proj, useMatrix = "PeakMatrix", groupBy = "Clusters")
```

---

## getGroupSummary *Get summary for Groups in ArchRProject*

---

### Description

This function summarizes a numeric cellColData entry across groupings in a ArchRProject.

### Usage

```
getGroupSummary(
  ArchRProj = NULL,
  groupBy = "Sample",
  select = "TSSEnrichment",
  summary = "median",
  removeNA = TRUE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping multiple cells together for summarizing information. |
| `select` | A character vector containing the column names to select from `cellColData`. |
| `summary` | A character vector describing which method for summarizing across group. Options include "median", "mean", or "sum". |
| `removeNA` | Remove NA's from summary method. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Group Summary
getGroupSummary(proj, groupBy = "Clusters", select = "TSSEnrichment", summary = "mean")
```

---

## `getImputeWeights` *Get Imputation Weights from ArchRProject*

---

### Description

This function gets imputation weights from an ArchRProject to impute numeric values.

### Usage

```
getImputeWeights(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Impute Weights
proj <- addImputeWeights(proj)

# Get Impute Weights
getImputeWeights(proj)
```

---

## `getInputFiles` *Get Input Files from paths to create arrows*

---

### Description

This function will look for fragment files and bam files in the input paths and return the full path and sample names. Only files that match ".fragments.tsv.gz" and ".bam" will be captured. This function requires there to be a prefix before ".fragments.tsv.gz" or ".bam" because this prefix will be used as the sample name. For example, "Sample1.fragments.tsv.gz" would be imported with the name "Sample1".

### Usage

```
getInputFiles(paths = NULL)
```

### Arguments

| | |
|---|---|
| `paths` | A character vector of paths to search for usable input files. |

---

## `getMarkerFeatures` *Identify Marker Features for each cell grouping*

---

### Description

This function will identify features that are definitional of each provided cell grouping where possible

### Usage

```
getMarkerFeatures(
  ArchRProj = NULL,
  groupBy = "Clusters",
  useGroups = NULL,
  bgdGroups = NULL,
  useMatrix = "GeneScoreMatrix",
  bias = c("TSSEnrichment", "log10(nFrags)"),
  normBy = NULL,
  testMethod = "wilcoxon",
  maxCells = 500,
  scaleTo = 10^4,
  threads = getArchRThreads(),
  k = 100,
  bufferRatio = 0.8,
  binarize = FALSE,
  useSeqnames = NULL,
  closest = FALSE,
  verbose = TRUE,
  logFile = createLogFile("getMarkerFeatures")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for marker feature identification. |
| `useGroups` | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData`. This limits the groups used to perform marker feature identification. |

| | |
|---|---|
| bgdGroups | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData` to be used for background calculations in marker feature identification. |
| useMatrix | The name of the matrix to be used for performing differential analyses. Options include "GeneScoreMatrix", "PeakMatrix", etc. |
| bias | A character vector indicating the potential bias variables (i.e. c("TSSEnrichment", "log10(nFrags)")) to account for in selecting a matched null group for marker feature identification. These should be column names from `cellColData`. |
| normBy | The name of a numeric column in `cellColData` that should be normalized across cells (i.e. "ReadsInTSS") prior to performing marker feature identification. |
| testMethod | The name of the pairwise test method to use in comparing cell groupings to the null cell grouping during marker feature identification. Valid options include "wilcoxon", "ttest", and "binomial". |
| maxCells | The maximum number of cells to consider from a single-cell group when performing marker feature identification. |
| scaleTo | Each column in the matrix designated by `useMatrix` will be normalized to a column sum designated by `scaleTo`. |
| threads | The number of threads to be used for parallel computing. |
| k | The number of nearby cells to use for selecting a biased-matched background while accounting for `bgdGroups` proportions. |
| bufferRatio | When generating optimal biased-matched background groups of cells to determine significance, it can be difficult to find sufficient numbers of well-matched cells to create a background group made up of an equal number of cells. The `bufferRatio` indicates the fraction of the total cells that must be obtained when creating the biased-matched group. For example to create a biased-matched background for a group of 100 cells, when `bufferRatio` is set to 0.8 the biased-matched background group will be composed of the 80 best-matched cells. This option provides flexibility in the generation of biased-matched background groups given the stringency of also maintaining the group proportions from `bgdGroups`. |
| binarize | A boolean value indicating whether to binarize the matrix prior to differential testing. This is useful when `useMatrix` is an insertion counts-based matrix. |
| useSeqnames | A character vector that indicates which `seqnames` should be plotted in the heatmap. Features from `seqnames` that are not listed will be ignored. In the context of a `Sparse.Assays.Matrix`, such as a matrix containing chromVAR deviations, the `seqnames` do not correspond to chromosomes, rather they correspond to the sub-portions of the matrix, for example raw deviations ("deviations") or deviation z-scores ("z") for a chromVAR deviations matrix. |
| closest | A boolean value that indicated whether to use closest cells from foreground and background instead of random sampling of the foreground cells. |
| verbose | A boolean value that determines whether standard output is printed. |
| logFile | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)
```

---

## getMarkers — *Get Marker Features from a marker summarized experiment*

---

### Description

This function will identify Markers and return a List of Features or a GRangesList for each group of significant marker features.

### Usage

```
getMarkers(
  seMarker = NULL,
  cutOff = "FDR <= 0.1 & Log2FC >= 0.5",
  n = NULL,
  returnGR = FALSE
)
```

### Arguments

| | |
|---|---|
| `seMarker` | A `SummarizedExperiment` object returned by `getMarkerFeatures()`. |
| `cutOff` | A valid-syntax logical statement that defines which marker features from `seMarker`. `cutoff` can contain any of the `assayNames` from `seMarker`. |
| `n` | An integer that indicates the maximum number of features to return per group. |
| `returnGR` | A boolean indicating whether to return as a `GRanges` object. Only valid when `seMarker` is computed for a PeakMatrix. |

### Examples

```
#Get Test Project
proj <- getTestProject()

#Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)

#Get Markers
getMarkers(seMarker)
```

---

## `getMatches` *Get peak annotation matches from an ArchRProject*

---

### Description

This function gets peak annotation matches from a given ArchRProject. The peaks in the returned object are in the same order as the peaks returned by `getPeakSet()`.

### Usage

```
getMatches(ArchRProj = NULL, name = NULL, annoName = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | The name of the `peakAnnotation` object (i.e. Motifs) to retrieve from the designated `ArchRProject`. |
| `annoName` | The name of a specific annotation to subset within the `peakAnnotation`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Annotation Matches
matches <- getMatches(proj)
```

---

## `getMatrixFromArrow` *Get a data matrix stored in an ArrowFile*

---

### Description

This function gets a given data matrix from an individual ArrowFile.

## Usage

```
getMatrixFromArrow(
  ArrowFile = NULL,
  useMatrix = "GeneScoreMatrix",
  useSeqnames = NULL,
  excludeChr = NULL,
  cellNames = NULL,
  ArchRProj = NULL,
  verbose = TRUE,
  binarize = FALSE,
  logFile = createLogFile("getMatrixFromArrow")
)
```

## Arguments

| | |
|---|---|
| `ArrowFile` | The path to an ArrowFile from which the selected data matrix should be obtained. |
| `useMatrix` | The name of the data matrix to retrieve from the given ArrowFile. Options include "TileMatrix", "GeneScoreMatrix", etc. |
| `useSeqnames` | A character vector of chromosome names to be used to subset the data matrix being obtained. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from this analysis. |
| `cellNames` | A character vector indicating the cell names of a subset of cells from which fragments whould be extracted. This allows for extraction of fragments from only a subset of selected cells. By default, this function will extract all cells from the provided ArrowFile using `getCellNames()`. |
| `ArchRProj` | An `ArchRProject` object to be used for getting additional information for cells in `cellColData`. In some cases, data exists within the `ArchRProject` object that does not exist within the ArrowFiles. To access this data, you can provide the `ArchRProject` object here. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `binarize` | A boolean value indicating whether the matrix should be binarized before return. This is often desired when working with insertion counts. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
#Get Test Arrow
arrow <- getTestArrow()

# Get Fragments
se <- getMatrixFromArrow(arrow)
```

## `getMatrixFromProject` *Get a data matrix stored in an ArchRProject*

### Description

This function gets a given data matrix from an `ArchRProject` and returns it as a `SummarizedExperiment`. This function will return the matrix you ask it for, without altering that matrix unless you tell it to. For example, if you added your `PeakMatrix` using `addPeakMatrix()` with `binarize = TRUE`, then `getMatrixFromProject()` will return a binarized `PeakMatrix`. Alternatively, you could set `binarize = TRUE` in the parameters passed to `getMatrixFromProject()` and the `PeakMatrix` will be binarized as you pull it out. No other normalization is applied to the matrix by this function.

### Usage

```
getMatrixFromProject(
  ArchRProj = NULL,
  useMatrix = "GeneScoreMatrix",
  useSeqnames = NULL,
  excludeChr = NULL,
  verbose = TRUE,
  binarize = FALSE,
  threads = getArchRThreads(),
  logFile = createLogFile("getMatrixFromProject")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object to get data matrix from. |
| `useMatrix` | The name of the data matrix to retrieve from the given ArrowFile. Options include "TileMatrix", "GeneScoreMatrix", etc. |
| `useSeqnames` | A character vector of chromosome names to be used to subset the data matrix being obtained. |
| `excludeChr` | A character vector containing the `seqnames` of the chromosomes that should be excluded from this analysis. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `binarize` | A boolean value indicating whether the matrix should be binarized before return. This is often desired when working with insertion counts. Note that if the matrix has already been binarized previously, this should be set to `TRUE`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

### Examples

```
#Get Test Project
proj <- getTestProject()
```

```
# Get Fragments
se <- getMatrixFromProject(proj)
```

---

`getMonocleTrajectories` *Get a Monocle CDS of Trajectories that can be added to an ArchRProject*

---

## Description

This function will use monocle3 to find trajectories and then returns a monocle CDS object that can be used as input for `addMonocleTrajectory`.

## Usage

```
getMonocleTrajectories(
  ArchRProj = NULL,
  name = "Trajectory",
  useGroups = NULL,
  principalGroup = NULL,
  groupBy = NULL,
  embedding = NULL,
  clusterParams = list(),
  graphParams = list(),
  seed = 1
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | A string indicating the name of the fitted trajectory. |
| `useGroups` | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData`. This limits the groups used to identify trajectories. |
| `principalGroup` | The principal group which represents the group that will be the starting point for all trajectories. |
| `groupBy` | A string indicating the column name from `cellColData` that contains the cell group definitions used in `useGroups` to constrain trajectory analysis. |
| `embedding` | A string indicating the name of the `embedding` object from the `ArchRProject` that should be used for trajectory analysis. |
| `clusterParams` | A list of parameters to be added when clustering cells for monocle3 with `monocle3::cluster_cells`. |
| `graphParams` | A list of parameters to be added when learning graphs for monocle3 with `monocle3::learn_graph`. |
| `seed` | A number to be used as the seed for random number generation for trajectory creation. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Create Monocole Trajectory
cds <- getMonocleTrajectories(
  ArchRProj = proj,
  useGroups = c("C1", "C2", "C3"),
  principalGroup = "C1",
  groupBy = "Clusters",
  embedding = "UMAP"
)
```

---

## `getOutputDirectory` *Get outputDirectory from an ArchRProject*

---

### Description

This function gets the outputDirectory from a given ArchRProject. If null this returns "QualityControl" directory.

### Usage

```
getOutputDirectory(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Output Directory
getOutputDirectory(proj)
```

## getPBGroupSE *Export PseudoBulk Group Summarized Experiment*

### Description

This function will determine cell groups for pseudobulk, summarize and export a summarized experiment for a assay in a ArchRProject.

### Usage

```
getPBGroupSE(
  ArchRProj = NULL,
  useMatrix = "GeneScoreMatrix",
  groupBy = "Clusters",
  divideN = TRUE,
  scaleTo = 10000,
  useLabels = TRUE,
  sampleLabels = "Sample",
  minCells = 40,
  maxCells = 500,
  minReplicates = 2,
  maxReplicates = 5,
  sampleRatio = 0.8,
  verbose = TRUE,
  threads = getArchRThreads(),
  logFile = createLogFile("getPBGroupSE")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the matrix in the ArrowFiles. See getAvailableMatrices to see options |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for summarizing. |
| `divideN` | A boolean describing whether to divide by the number of cells. |
| `scaleTo` | Depth normalize to this value if not NULL. |
| `useLabels` | A boolean value indicating whether to use sample labels to create sample-aware subgroupings during as pseudo-bulk replicate generation. |
| `sampleLabels` | The name of a column in `cellColData` to use to identify samples. In most cases, this parameter should be left as `NULL` and you should only use this parameter if you do not want to use the default sample labels stored in `cellColData$Sample`. However, if your individual Arrow files do not map to individual samples, then you should set this parameter to accurately identify your samples. This is the |

| | |
|---|---|
| | case in (for example) multiplexing applications where cells from different biological samples are mixed into the same reaction and demultiplexed based on a lipid barcode or genotype. |
| `minCells` | The minimum number of cells required in a given cell group to permit insertion coverage file generation. |
| `maxCells` | The maximum number of cells to use during insertion coverage file generation. |
| `minReplicates` | The minimum number of pseudo-bulk replicates to be generated. |
| `maxReplicates` | The maximum number of pseudo-bulk replicates to be generated. |
| `sampleRatio` | The fraction of the total cells that can be sampled to generate any given pseudo-bulk replicate. |
| `verbose` | A boolean specifying to print messages during computation. |
| `threads` | An integer specifying the number of threads for parallel. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Group SE
se <- getPBGroupSE(proj, useMatrix = "PeakMatrix", groupBy = "Clusters")
```

---

## `getPeak2GeneLinks` *Get the peak-to-gene links from an ArchRProject*

---

## Description

This function obtains peak-to-gene links from an ArchRProject.

## Usage

```
getPeak2GeneLinks(
  ArchRProj = NULL,
  corCutOff = 0.45,
  FDRCutOff = 1e-04,
  varCutOffATAC = 0.25,
  varCutOffRNA = 0.25,
  resolution = 1,
  returnLoops = TRUE
)
```

### Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| corCutOff | A numeric describing the minimum numeric peak-to-gene correlation to return. If this value is positive, then only positive correlations greater than or equal to corCutOff will be returned. If this value is negative, then only negative correlations less than or equal to corCutOff will be returned. If corCutOff = 0, only positive correlations will be returned. |
| FDRCutOff | A numeric describing the maximum numeric peak-to-gene false discovery rate to return. |
| varCutOffATAC | A numeric describing the minimum variance quantile of the ATAC peak accessibility when selecting links. |
| varCutOffRNA | A numeric describing the minimum variance quantile of the RNA gene expression when selecting links. |
| resolution | A numeric describing the bp resolution to return loops as. This helps with overplotting of correlated regions. |
| returnLoops | A boolean indicating to return the peak-to-gene links as a GRanges "loops" object designed for use with the ArchRBrowser() or as an ArchRBrowserTrack(). |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add P2G Links
proj <- addPeak2GeneLinks(proj, k = 20)

# Get P2G Links
p2g <- getPeak2GeneLinks(proj)
```

## getPeakAnnotation *Get peakAnnotation from an ArchRProject*

### Description

This function gets a peakAnnotation from a given ArchRProject.

### Usage

```
getPeakAnnotation(ArchRProj = NULL, name = NULL)
```

### Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| name | The name of the peakAnnotation object (i.e. Motifs) to retrieve from the designated ArchRProject. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Peak Annotations
peakAnno <- getPeakAnnotation(proj)
```

---

## getPeakSet — *Get the peak set from an ArchRProject*

---

### Description

This function gets the peak set as a GRanges object from an ArchRProject.

### Usage

```
getPeakSet(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get PeakSet
getPeakSet(proj)
```

---

## getPositions — *Get peak annotation positions from an ArchRProject*

---

### Description

This function gets the peak annotation positions (i.e. Motifs) from a given ArchRProject.

### Usage

```
getPositions(ArchRProj = NULL, name = NULL, annoName = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | The name of the `peakAnnotation` object (i.e. Motifs) to retrieve from the designated `ArchRProject`. |
| `annoName` | The name of a specific annotation to subset within the `peakAnnotation`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Annotation Positions
positions <- getPositions(proj)
```

---

## `getProjectSummary` *Get projectSummary from an ArchRProject*

---

### Description

This function prints the projectSummary from an ArchRProject

### Usage

```
getProjectSummary(ArchRProj = NULL, returnSummary = FALSE)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `returnSummary` | A boolean value indicating whether to return a summary of the `ArchRProject` or to just print the summary. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Project Summary
getProjectSummary(proj)
```

## getReducedDims — *Get dimensionality reduction information stored in an ArchRProject*

### Description

This function gets a dimensionality reduction object (i.e. UMAP, tSNE, etc) from a given ArchRProject.

### Usage

```
getReducedDims(
  ArchRProj = NULL,
  reducedDims = "IterativeLSI",
  returnMatrix = TRUE,
  dimsToUse = NULL,
  scaleDims = NULL,
  corCutOff = 0.75
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `reducedDims` | The name of the `reducedDims` object (i.e. "IterativeLSI") to retrieve from the designated `ArchRProject`. |
| `returnMatrix` | If set to "mat" or "matrix", the function will return the `reducedDims` object as a matrix with entries for each individual cell. Otherwise, it will return the full `reducedDims` object. |
| `dimsToUse` | A vector containing the dimensions (i.e. 1:30) to return from the `reducedDims` object. |
| `scaleDims` | A boolean describing whether to z-score the reduced dimensions for each cell. This is useful for minimizing the contribution of strong biases (dominating early PCs) and lowly abundant populations. However, this may lead to stronger sample-specific biases since it is over-weighting latent PCs. If `NULL` this will scale the dimensions depending on if this were set true when the `reducedDims` were created by the dimensionality reduction method. This idea was introduced by Timothy Stuart. |
| `corCutOff` | A numeric cutoff for the correlation of each dimension to the sequencing depth. If the dimension has a correlation to sequencing depth that is greater than the `corCutOff`, it will be excluded. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Iterative LSI
```

```
getReducedDims(proj, reducedDims = "IterativeLSI")
```

## getSampleColData *Get sampleColData from an ArchRProject*

### Description

This function gets the sampleColData from a given ArchRProject.

### Usage

```
getSampleColData(ArchRProj = NULL, select = NULL, drop = FALSE)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `select` | A character vector containing the column names to select from `sampleColData`. |
| `drop` | A boolean value that indicates whether to drop the `dataframe` structure and convert to a vector if selecting only one column. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Sample Column Data
getSampleColData(proj)
```

## getSampleNames *Get the sample names from an ArchRProject*

### Description

This function gets the names of all samples from a given ArchRProject.

### Usage

```
getSampleNames(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Sample Names
getSampleNames(proj)
```

---

## getSeqnames *Get the seqnames that could be selected from a given data matrix within an ArchRProject*

---

### Description

This function will identify available seqnames from a given data matrix (i.e. "GeneScoreMatrix", or "TileMatrix") and return them for downstream plotting utilities.

### Usage

```
getSeqnames(ArchRProj = NULL, useMatrix = "GeneScoreMatrix")
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `useMatrix` | The name of the data matrix or "Fragments" as stored in the ArrowFiles of the `ArchRProject`. Options include "TileMatrix", "GeneScoreMatrix", etc. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Seqnames in Matrix
getSeqnames(proj, useMatrix = "GeneScoreMatrix")
```

---

## getTestArrow *Get PBMC Small Test Arrow file*

---

### Description

V2 : This function will return a test arrow file in your cwd.

### Usage

```
getTestArrow(version = 2)
```

## Arguments

| | |
|---|---|
| `version` | version of test arrow to return |

## Examples

```
# Get Test Arrow
arrow <- getTestArrow()
```

---

`getTestFragments` *Get PBMC Small Test Fragments*

---

## Description

V1 : This function will download fragments for a small PBMC test dataset. V2 : This function will return test fragments for a small PBMC test dataset in your cwd.

## Usage

```
getTestFragments(version = 2)
```

## Arguments

| | |
|---|---|
| `version` | version of test fragments to return |

## Examples

```
# Get Test Fragments
fragments <- getTestFragments()
```

---

`getTestProject` *Get PBMC Small Test Project*

---

## Description

V1 : This function will download an ArchRProject for a small PBMC test dataset. V2 : This function will return an ArchRProject for a small PBMC test dataset in your cwd.

## Usage

```
getTestProject(version = 2)
```

## Arguments

| | |
|---|---|
| `version` | version of test fragments to return |

## Examples

```
# Get Test Project
proj <- getTestProject()
```

---

## getTrajectory *Get Supervised Trajectory from an ArchR Project*

---

### Description

This function will get a supervised trajectory from an `ArchRProject` (see `addTrajectory`), get data from a desired matrix, and smooth each value across the input trajectory.

### Usage

```
getTrajectory(
  ArchRProj = NULL,
  name = "Trajectory",
  useMatrix = "GeneScoreMatrix",
  groupEvery = 1,
  log2Norm = TRUE,
  scaleTo = 10000,
  smoothWindow = 11,
  trajectoryLabel = NULL,
  threads = getArchRThreads()
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `name` | A string indicating the name of the fitted trajectory in `cellColData` to retrieve from the given `ArchRProject`. |
| `useMatrix` | The name of the data matrix from the `ArrowFiles` to get numerical values for each cell from. Recommended matrices are "GeneScoreMatrix", "PeakMatrix", or "MotifMatrix". |
| `groupEvery` | The number of sequential percentiles to group together when generating a trajectory. This is similar to smoothing via a non-overlapping sliding window across pseudo-time. If `groupEvery = 2`, the values for percentiles 1 and 2, 3 and 4, 5 and 6, etc. will be grouped together. |
| `log2Norm` | A boolean value that indicates whether the summarized trajectory matrix should be log2 transformed. If you are using a "MotifMatrix" set to FALSE. |
| `scaleTo` | Once the sequential trajectory matrix is created, each column in that matrix will be normalized to a column sum indicated by `scaleTo`. Setting this to `NULL` will prevent any normalization and should be done in certain circumstances (for ex. if you are using a "MotifMatrix"). |

| | |
|---|---|
| smoothWindow | An integer value indicating the smoothing window in size (relaive to `groupEvery`) for the sequential trajectory matrix to better reveal temporal dynamics. |
| trajectoryLabel | The name of a column in `cellColData` to be used for labeling the quantile bins of the trajectory. This is used in `plotTrajectoryHeatmap()` when you want to create a color label for the columns across the top of the heatmap. Typically, this should be the same column used in `groupBy` in the `addTrajectory()` function. |
| threads | The number of threads to be used for parallel computing. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add Trajectory
proj <- addTrajectory(proj, trajectory = c("C1", "C2", "C3"), embedding = "UMAP", force = TRUE)

#Get Trajectory
seTraj <- getTrajectory(proj)
```

## getTSS *Get the transcription start sites of all genes in an ArchRProject*

### Description

This function gets the transcription start sites (TSSs) as a GRanges object of all genes from the geneAnnotation of a given ArchRProject.

### Usage

```
getTSS(ArchRProj = NULL)
```

### Arguments

| | |
|---|---|
| ArchRProj | An `ArchRProject` object. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get TSS in ArchRProj
getTSS(proj)

# Get TSS loaded globally
getTSS()
```

## getTutorialData *Get Relevant Data For ArchR Tutorials*

### Description

This function will download data for a given tutorial and return the input files required for ArchR.

### Usage

```
getTutorialData(tutorial = "hematopoiesis", threads = getArchRThreads())
```

### Arguments

| | |
|---|---|
| `tutorial` | The name of the available tutorial for which to retreive the tutorial data. The main option is "Hematopoiesis". "Hematopoiesis" is a small scATAC-seq dataset that spans the hematopoieitic hierarchy from stem cells to differentiated cells. This dataset is made up of cells from peripheral blood, bone marrow, and CD34+ sorted bone marrow. The second option is "Test" which is downloading a small test PBMC fragments file mainly used to test the url capabilities of this function. |
| `threads` | The number of threads to be used for parallel computing. |

### Examples

```
# Get Tutorial Fragments using `test` since its smaller
fragments <- getTutorialData(tutorial = "test")
```

## getValidBarcodes *Get Valid Barcodes from 10x Cell Ranger output to pre-filter barcodes*

### Description

This function will read in processed 10x cell ranger files and identify barcodes that are associated with a cell that passed QC.

### Usage

```
getValidBarcodes(csvFiles = NULL, sampleNames = NULL)
```

### Arguments

| | |
|---|---|
| `csvFiles` | A character vector of names from 10x CSV files to be read in for identification of valid cell barcodes. |
| `sampleNames` | A character vector containing the sample names to be associated with each individual entry in `csvFiles`. |

## getVarDeviations *Get Variable Deviations across cells in ArchRProject.*

### Description

This function will rank the variability of the deviations computed by ArchR and label the top variable annotations.

### Usage

```
getVarDeviations(ArchRProj = NULL, name = "MotifMatrix", plot = TRUE, n = 25)
```

### Arguments

| | |
|---|---|
| ArchRProj | An `ArchRProject` object. |
| name | The name of the `DeviationsMatrix` object stored in the `ArchRProject`. See `addDeviationsMatrix()`. |
| plot | A boolean value indicating whether the ranked variability should be plotted for each peakAnnotation in `DeviationsMatrix`. |
| n | The number of annotations to label with `ggrepel`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Variable Motif Deviations
varDev <- getVarDeviations(proj)
```

## ggAlignPlots *Align ggplot plots vertically or horizontally*

### Description

This function aligns ggplots vertically or horizontally

### Usage

```
ggAlignPlots(..., plotList = NULL, sizes = NULL, type = "v", draw = TRUE)
```

## Arguments

| | |
|---|---|
| `...` | All additional arguments will be interpreted as `ggplot2` plot objects and used if and only if `plotList` is `NULL` |
| `plotList` | A list of `ggplot2` plot objects to be aligned. |
| `sizes` | A numeric vector or list of values indicating the relative size for each of the objects in `plotList` or supplied in `...`. If the plot is supplied in `...` the order is the same as the input in this function. If set to NULL all plots will be evenly distributed. |
| `type` | A string indicating wheter vertical ("v") or horizontal ("h") alignment should be used for the multi-plot layout. |
| `draw` | A boolean value indicating whether to draw the plot(s) (`TRUE`) or return a graphical object (`FALSE`). |

## Examples

```
# Create Random Data
m <- data.frame(x=matrix(rnorm(10, 2),ncol=1))
m$color <- sample(c("A", "B"), 10, replace = TRUE)

# Plot
p <- ggGroup(x = m$color, y = m$x)

# To PDF
pdf("test.pdf", width = 4, height = 7)
ggAlignPlots(p, p)
dev.off()
```

---

## ggGroup *A ggplot-based ridge/violin plot wrapper function*

---

## Description

This function is a wrapper around ggplot geom_density_ridges or geom_violin to allow for plotting group distribution plots in ArchR.

## Usage

```
ggGroup(
  x = NULL,
  y = NULL,
  xlabel = NULL,
  ylabel = NULL,
  groupOrder = NULL,
  groupSort = FALSE,
  size = 1,
  baseSize = 10,
  ridgeScale = 1,
  ratioYX = NULL,
  alpha = 1,
  title = "",
  pal = paletteDiscrete(values = x, set = "stallion"),
  addBoxPlot = TRUE,
  plotAs = "ridges",
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A character vector containing the categorical x-axis values for each y-axis value. |
| `y` | A numeric vector containing the y-axis values for each point. |
| `xlabel` | The label to plot for the x-axis. |
| `ylabel` | The label to plot for the y-axis. |
| `groupOrder` | A character vector indicating a custom order for plotting x-axis categorical values. Should contain all possible values of `x` in the desired order. |
| `groupSort` | A boolean indicating whether to sort groups based on the average value of the group. |
| `size` | The line width for boxplot/summary lines. |
| `baseSize` | The base font size (in points) to use in the plot. |
| `ridgeScale` | A numeric indicating the relative size for each ridge in the ridgeplot. |
| `ratioYX` | The aspect ratio of the x and y axes on the plot. |
| `alpha` | A number indicating the transparency to use for each point. See `ggplot2` for more details. |
| `title` | The title of the plot. |
| `pal` | A named custom palette (see `paletteDiscrete()` and `ArchRPalettes`) for discrete coloring. |
| `addBoxPlot` | A boolean indicating whether to add a boxplot to the plot if `plotAs="violin"`. |
| `plotAs` | A string indicating how the groups should be plotted. Acceptable values are "ridges" (for a `ggrides`-style plot) or "violin" (for a violin plot). |
| `...` | Additional parameters to pass to `ggplot2` for plotting. |

## Examples

```
# Create Random Data
m <- data.frame(x=matrix(rnorm(10, 2),ncol=1))
m$color <- sample(c("A", "B"), 10, replace = TRUE)

# Plot
p <- ggGroup(x = m$color, y = m$x)

# To PDF
```

```
pdf("test.pdf", width = 4, height = 4)
p
dev.off()
```

---

ggHex *A ggplot-based Hexplot wrapper function summary of points in a standardized manner*

---

## Description

This function will plot x,y coordinate values summarized in hexagons in a standardized manner

## Usage

```
ggHex(
  x = NULL,
  y = NULL,
  color = NULL,
  pal = paletteContinuous(set = "solarExtra"),
  bins = 200,
  xlim = NULL,
  ylim = NULL,
  extend = 0.05,
  xlabel = "x",
  ylabel = "y",
  title = "",
  colorTitle = "values",
  baseSize = 6,
  ratioYX = 1,
  FUN = "median",
  hexCut = c(0.02, 0.98),
  addPoints = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A numeric vector containing the x-axis values for each point. |
| `y` | A numeric vector containing the y-axis values for each point. |
| `color` | A numeric/categorical vector containing coloring information for each point. |
| `pal` | A custom continuous palette from `ArchRPalettes` for coloration of hexes. |
| `bins` | The number of bins to be used for plotting the hexplot. `bins` indicates the total number of hexagons that will fit within the surface area of the plot. |
| `xlim` | A numeric vector of two values indicating the lower and upper bounds of the x-axis on the plot. |

| | |
|---|---|
| `ylim` | A numeric vector of two values indicating the lower and upper bounds of the y-axis on the plot. |
| `extend` | A numeric value indicating the fraction to extend the x-axis and y-axis beyond the maximum and minimum values if `xlim` and `ylim` are not provided. For example, 0.05 will extend the x-axis and y-axis by 5 percent on each end. |
| `xlabel` | The label to plot for the x-axis. |
| `ylabel` | The label to plot for the y-axis. |
| `title` | The title of the plot. |
| `colorTitle` | The label to use for the legend corresponding to `color`. |
| `baseSize` | The base font size (in points) to use in the plot. |
| `ratioYX` | The aspect ratio of the x and y axes on the plot. |
| `FUN` | The function to use for summarizing data into hexagons. Typically "mean" or something similar. |
| `hexCut` | If this is not null, a quantile cut is performed to threshold the top and bottom of the distribution of values. This prevents skewed color scales caused by strong outliers. The format of this should be c(a,b) where `a` is the upper threshold and `b` is the lower threshold. For example, hexCut = c(0.025,0.975) will take the top and bottom 2.5 percent of values and set them to the value of the 97.5th and 2.5th percentile values respectively. |
| `addPoints` | A boolean value indicating whether individual points should be shown on the hexplot. |
| `...` | Additional params for plotting |

## Examples

```
# Create Random Data
m <- data.frame(matrix(rnorm(300, 2),ncol=3))

# Plot
p <- ggHex(x = m[,1], y = m[,2], color = m[,3])

# To PDF
pdf("test.pdf", width = 4, height = 4)
p
dev.off()
```

---

## `ggOneToOne` *A ggplot-based one-to-one dot plot wrapper function*

---

## Description

This function is a wrapper around ggplot geom_point to allow for plotting one-to-one sample comparisons in ArchR.

## Usage

```
ggOneToOne(
  x = NULL,
  y = NULL,
  size = 2,
  alpha = 1,
  xlabel = "x",
  ylabel = "y",
  title = "Correlation",
  min = 0.05,
  max = 0.9999,
  nPlot = 100 * 10^3,
  nKernel = 100,
  densityMax = 0.95,
  extend = 0.05,
  baseSize = 6,
  rastr = TRUE,
  pal = paletteContinuous(set = "blueYellow"),
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A numeric vector containing the x-axis values for each point. |
| `y` | A numeric vector containing the y-axis values for each point. |
| `size` | The numeric size of the points to plot. |
| `alpha` | A number indicating the transparency to use for each point. See `ggplot2` for more details. |
| `xlabel` | The label to plot for the x-axis. |
| `ylabel` | The label to plot for the y-axis. |
| `title` | The title of the plot. |
| `min` | The lower limit of the x and y axes as a numeric quantile between 0 and 1. |
| `max` | The upper limit of the x and y axes as a numeric quantile between 0 and 1. |
| `nPlot` | The number of points to plot. When this value is less than the total points, the `sample` function is used to extract random data points to be plotted. |
| `nKernel` | The number of grid points in each direction to use when computing the kernel with `MASS::kde2d()`. |
| `densityMax` | The quantile that should be represented by the maximum color on the continuous scale designated by `pal`. Values above `densityMax` will be thresholded to the maximum color on the color scale. |
| `extend` | A numeric value indicating the fraction to extend the x-axis and y-axis beyond the maximum value on either axis. For example, 0.05 will extend the x-axis and y-axis by 5 percent on each end beyond `quantile(c(x,y), max)` and `quantile(c(x,y), min)`. |

| | |
|---|---|
| `baseSize` | The base font size (in points) to use in the plot. |
| `rastr` | A boolean value that indicates whether the plot should be rasterized. This does not rasterize lines and labels, just the internal portions of the plot. |
| `pal` | A custom palette from `ArchRPalettes` used to display the density of points on the plot. |
| `...` | Additional params to be supplied to ggPoint |

## Examples

```
# Create Random Data
m <- data.frame(matrix(rnorm(20, 2),ncol=2))

# Plot
p <- ggOneToOne(x = m[,1], y = m[,2])

# To PDF
pdf("test.pdf", width = 4, height = 4)
p
dev.off()
```

---

`ggPoint` *A ggplot-based dot plot wrapper function*

---

## Description

This function is a wrapper around ggplot geom_point to allow for a more intuitive plotting of ArchR data.

## Usage

```
ggPoint(
  x = NULL,
  y = NULL,
  color = NULL,
  discrete = TRUE,
  discreteSet = "stallion",
  continuousSet = "solarExtra",
  labelMeans = TRUE,
  pal = NULL,
  defaultColor = "lightGrey",
  highlightPoints = NULL,
  colorDensity = FALSE,
  size = 1,
  xlim = NULL,
  ylim = NULL,
  extend = 0.05,
  xlabel = "x",
  ylabel = "y",
  title = "",
  randomize = FALSE,
  seed = 1,
  colorTitle = NULL,
  colorOrder = NULL,
  colorLimits = NULL,
  alpha = 1,
  baseSize = 10,
  legendSize = 3,
  ratioYX = 1,
  labelAsFactors = TRUE,
  fgColor = "black",
  bgColor = "white",
  bgWidth = 1,
  labelSize = 3,
  addFit = NULL,
  rastr = FALSE,
  dpi = 300,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A numeric vector containing the x-axis values for each point. |
| `y` | A numeric vector containing the y-axis values for each point. |
| `color` | A numeric/categorical vector used to determine the coloration for each point. |
| `discrete` | A boolean value indicating whether the supplied data is discrete (`TRUE`) or continuous (`FALSE`). |
| `discreteSet` | The name of a custom palette from `ArchRPalettes` to use for categorical/discrete color. This argument is only used if `discrete` is set to `TRUE`. |
| `continuousSet` | The name of a custom palette from `ArchRPalettes` to use for numeric color. This argument is only used if `discrete` is set to `FALSE`. |
| `labelMeans` | A boolean value indicating whether the mean of each categorical/discrete color should be labeled. |
| `pal` | A custom palette used to override discreteSet/continuousSet for coloring vector. |
| `defaultColor` | The default color for points that do not have another color applied (i.e. `NA` values). |
| `highlightPoints` | A integer vector describing which points to hightlight. The remainder of points will be colored light gray. |
| `colorDensity` | A boolean value indicating whether the density of points on the plot should be indicated by color. If `TRUE`, continuousSet is used as the color palette. |
| `size` | The numeric size of the points to be plotted. |

| | |
|---|---|
| `xlim` | A numeric vector of two values indicating the lower and upper bounds of the x-axis on the plot. |
| `ylim` | A numeric vector of two values indicating the lower and upper bounds of the y-axis on the plot. |
| `extend` | A numeric value indicating the fraction to extend the x-axis and y-axis beyond the maximum and minimum values if `xlim` and `ylim` are not provided. For example, 0.05 will extend the x-axis and y-axis by 5 percent on each end. |
| `xlabel` | The label to plot for the x-axis. |
| `ylabel` | The label to plot for the y-axis. |
| `title` | The title of the plot. |
| `randomize` | A boolean value indicating whether to randomize the order of the points when plotting. |
| `seed` | A numeric seed number for use in randomization. |
| `colorTitle` | A title to be added to the legend if `color` is supplied. |
| `colorOrder` | A vector that allows you to control the order of palette colors associated with the values in `color`. For example if you have `color` as `c("a","b","c")` and want to have the first color selected from the palette be used for "c", the second color for "b", and the third color for "a", you would supply the `colorOrder` as `c("c", "b", "a")`. |
| `colorLimits` | A numeric vector of two values indicating the lower and upper bounds of colors if numeric. Values beyond these limits are thresholded. |
| `alpha` | A number indicating the transparency to use for each point. See `ggplot2` for more details. |
| `baseSize` | The base font size (in points) to use in the plot. |
| `legendSize` | The size in inches to use for plotting the color legend. |
| `ratioYX` | The aspect ratio of the x and y axes on the plot. |
| `labelAsFactors` | A boolean indicating whether to label the `color` input as a numeric factor (`TRUE`) or with a character string (`FALSE`). |
| `fgColor` | The foreground color of the plot. |
| `bgColor` | The background color of the plot. |
| `bgWidth` | The background relative width size of the halos in the labeling. |
| `labelSize` | The numeric font size of labels. |
| `addFit` | A string indicating the method to use for adding a fit/regression line to the plot (see `ggplot2::geom_smooth()` methods). If set to `NULL`, no fit/regression line is added. |
| `rastr` | A boolean value that indicates whether the plot should be rasterized using `ggrastr`. This does not rasterize lines and labels, just the internal portions of the plot. |
| `dpi` | The resolution in dots per inch to use for the plot. |

## Examples

```
# Create Random Data
m <- data.frame(matrix(rnorm(20, 2),ncol=2))
m$color <- sample(c("A", "B"), 10, replace = TRUE)

# Plot
p <- ggPoint(x = m[,1], y = m[,2], color = m[,3])

# To PDF
pdf("test.pdf", width = 4, height = 4)
p
dev.off()
```

---

`import10xFeatureMatrix`
*Import Feature Matrix from 10x Feature HDF5 file.*

---

## Description

This function will import the feature matrix from a 10x feature hdf5 file.

## Usage

```
import10xFeatureMatrix(
  input = NULL,
  names = NULL,
  strictMatch = TRUE,
  verbose = TRUE,
  featureType = "Gene Expression",
  features = NULL,
  genesToExclude = NULL,
  force = FALSE
)
```

## Arguments

| | |
|---|---|
| `input` | A character of paths to 10x feature hdf5 file(s). These will traditionally have a suffix similar to "filtered_feature_bc_matrix.h5". |
| `names` | A character of sample names associated with each input file. |
| `strictMatch` | Only relevant when multiple input files are used. A boolean that indictes whether rows (genes) that do not match perfectly in the matrices should be removed (`strictMatch = TRUE`) or coerced (`strictMatch = FALSE`). Cell Ranger seems to occassionally use different ensembl ids for the same gene across different samples. If you are comfortable tolerating such mismatches, you can coerce all matrices to fit together, in which case the gene metadata present in the first listed sample will be applied to all matrices for that particular gene entry. Sometimes, |

| | |
|---|---|
| | the h5 files from Cell Ranger have mismatches in the actual gene names. See the `force` paramter for handling mismatched gene names. |
| `verbose` | Only relevant when multiple input files are used. A boolean that indicates whether messaging about mismatches should be verbose (`TRUE`) or minimal (`FALSE`) |
| `featureType` | The name of the feature to extract from the 10x feature file. See https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/advanced/h5_matrices for more information. |
| `features` | A genomic ranges object containing a "name" column to help fill missing 10x intervals for RSE. With the default CellRanger output, mitochondrial genes are not given genomic ranges. If you wish to recover these transcripts, for example for QC-based filtering, you must provide ranges for these genes. The same applies for any genes in the provided `h5` files that are missing ranges. As an example, in hg38 you could use Bioconductors `features = genes(EnsDb.Hsapiens.v86::EnsDb.Hsapiens.v86)`. Note that you must load the corresponding Bioconductor package to have access to the `genes` function. |
| `genesToExclude` | A character vector of gene names to be excluded from the returned `SummarizedExperiment` object. This can be useful when dealing with very large datasets because R has an upper limit for how large a given object can be. If you have >1.5 million cells, you cannot create a matrix in R with all ~30,000 genes so you can use this parameter to remove genes upfront. |

## `imputeMatrix` *Impute a matrix with impute weights*

### Description

This function gets imputation weights from an ArchRProject to impute a numerical matrix

### Usage

```
imputeMatrix(
  mat = NULL,
  imputeWeights = NULL,
  threads = getArchRThreads(),
  verbose = FALSE,
  logFile = createLogFile("imputeMatrix")
)
```

### Arguments

| | |
|---|---|
| `mat` | A matrix or sparseMatrix of class dgCMatrix to be imputed. |
| `imputeWeights` | An R object containing impute weights as returned by `getImputeWeights(ArchRProj)`. See `addImputeWeights()` for more details. |
| `threads` | The number of threads to be used for parallel computing. |

| | |
|---|---|
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add Impute Weights
proj <- addImputeWeights(proj)

# Get Impute Weights
iW <- getImputeWeights(proj)

# Get Matrix
se <- getMatrixFromProject(proj, useMatrix = "GeneScoreMatrix")

# Impute
mat <- imputeMatrix(assay(se), iW)
```

## `installExtraPackages` *Install extra packages used in ArchR that are not installed by default*

### Description

This function will install extra packages used in ArchR that are not installed by default.

### Usage

```
installExtraPackages(force = FALSE)
```

### Arguments

| | |
|---|---|
| `force` | If you want to force a reinstall of these pacakges. |

### Examples

```
# Install
installExtraPackages()
```

## `loadArchRProject` *Load Previous ArchRProject into R*

### Description

This function will load a previously saved ArchRProject and re-normalize paths for usage.

### Usage

```
loadArchRProject(path = "./", force = FALSE, showLogo = TRUE)
```

### Arguments

| | |
|---|---|
| `path` | A character path to an `ArchRProject` directory that was previously saved using `saveArchRProject()`. |
| `force` | A boolean value indicating whether missing optional `ArchRProject` components (i.e. peak annotations / background peaks) should be ignored when re-normalizing file paths. If set to `FALSE` loading of the `ArchRProject` will fail unless all components can be found. |
| `showLogo` | A boolean value indicating whether to show the ascii ArchR logo after successful creation of an `ArchRProject`. |

### Examples

```
# Get Small PBMC Project Location
zipProj <- file.path(system.file("testdata", package="ArchR"), "PBSmall.zip")

# Copy to current directory
file.copy(zipProj, basename(zipProj), overwrite = TRUE)

# Unzip
unzip(basename(zipProj), overwrite = TRUE)

# Remove
file.remove(basename(zipProj))

# Load
loadArchRProject("PBSmall")
```

## mapLabels *Re-map a character vector of labels from an old set of labels to a new set of labels*

### Description

This function takes a character vector of labels and uses a set of old and new labels to re-map from the old label set to the new label set.

### Usage

```
mapLabels(labels = NULL, newLabels = NULL, oldLabels = names(newLabels))
```

### Arguments

| | |
|---|---|
| `labels` | A character vector containing lables to map. |
| `newLabels` | A character vector (same length as oldLabels) to map labels to from oldLabels. |
| `oldLabels` | A character vector (same length as newLabels) to map labels from to newLabels |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Peak Annotations
proj$ClusterLabels <- mapLabels(proj$Clusters, c("T", "B", "M"), c("C1", "C2", "C3"))
```

## nCells *Get the number of cells from an ArchRProject/ArrowFile*

### Description

This function gets number of cells from an ArchRProject or ArrowFile

### Usage

```
nCells(input = NULL)
```

### Arguments

| | |
|---|---|
| `input` | An `ArchRProject` object or the path to an ArrowFile. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Number of Cells
nCells(proj)
```

---

`nonOverlappingGR` *Retreive a non-overlapping set of regions from a Genomic Ranges object*

---

## Description

This function returns a GRanges object containing a non-overlapping set regions derived from a supplied Genomic Ranges object.

## Usage

```
nonOverlappingGR(gr = NULL, by = "score", decreasing = TRUE, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| gr | A GRanges object. |
| by | The name of a column in mcols(gr) that should be used to determine how overlapping regions should be resolved. The resolution of overlapping regions also depends on decreasing. For example, if a column named "score" is used for by, decreasing = TRUE means that the highest "score" in the overlap will be retained and decreasing = FALSE means that the lowest "score" in the overlap will be retained. |
| decreasing | A boolean value indicating whether the values in the column indicated via by should be ordered in decreasing order. If TRUE, the higher value in by will be retained. |
| verbose | A boolean value indicating whether the output should include extra reporting. |

## Examples

```
# Dummy GR
gr <- GRanges(
  seqnames = "chr1",
  ranges = IRanges(
    start = c(1, 4, 11),
    end = c(10, 12, 20)
  ),
  score = c(1, 2, 3)
)
```

```
# Non Overlapping
nonOverlappingGR(gr)
```

---

## `paletteContinuous` *Continuous Color Palette*

---

### Description

Continuous Color Palette

### Usage

```
paletteContinuous(set = "solarExtra", n = 256, reverse = FALSE)
```

### Arguments

| | |
|---|---|
| `set` | The name of a color palette provided in the `ArchRPalettes` list object. |
| `n` | The number of unique colors to generate as part of this continuous color palette. |
| `reverse` | A boolean variable that indicates whether to return the palette colors in reverse order. |

### Examples

```
# Color Palette
pal <- paletteContinuous()
```

---

## `paletteDiscrete` *Optimized discrete color palette generation*

---

### Description

This function assesses the number of inputs and returns a discrete color palette that is tailored to provide the most possible color contrast from the designated color set.

### Usage

```
paletteDiscrete(values = NULL, set = "stallion", reverse = FALSE)
```

### Arguments

| | |
|---|---|
| `values` | A character vector containing the sample names that will be used. Each entry in this character vector will be given a unique color from the designated palette set. |
| `set` | The name of a color palette provided in the `ArchRPalettes` list object. |
| `reverse` | A boolean variable that indicates whether to return the palette colors in reverse order. |

## Examples

```
# Vector
v <- c("A", "B")

# Color Palette
pal <- paletteDiscrete(values = v)
```

---

## peakAnnoEnrichment *Peak Annotation Hypergeometric Enrichment in Marker Peaks.*

---

### Description

This function will perform hypergeometric enrichment of a given peak annotation within the defined marker peaks.

### Usage

```
peakAnnoEnrichment(
  seMarker = NULL,
  ArchRProj = NULL,
  peakAnnotation = NULL,
  matches = NULL,
  cutOff = "FDR <= 0.1 & Log2FC >= 0.5",
  background = "all",
  logFile = createLogFile("peakAnnoEnrichment")
)
```

### Arguments

| | |
|---|---|
| `seMarker` | A `SummarizedExperiment` object returned by `markerFeatures()`. |
| `ArchRProj` | An `ArchRProject` object. |
| `peakAnnotation` | A `peakAnnotation` object in the provided `ArchRProject` to be used for hypergeometric testing. |
| `matches` | A custom `peakAnnotation` matches object used as input for the hypergeometric test. See `motifmatchr::matchmotifs()` for additional information. |
| `cutOff` | A valid-syntax logical statement that defines which marker features from `seMarker` to use. `cutoff` can contain any of the `assayNames` from `seMarker`. |
| `background` | A string that indicates whether to use a background set of matched peaks to compare against ("bgdPeaks") or all peaks ("all"). |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)

# Get Peak Annotation Enrichment
annoEnrich <- peakAnnoEnrichment(
  seMarker = seMarker,
  ArchRProj = proj,
  cutOff = "FDR <= 0.1 & Log2FC >= 0"
)
```

---

## `plotBrowserTrack` *Plot an ArchR Region Track*

---

### Description

This function will plot the coverage at an input region in the style of a browser track. It allows for normalization of the signal which enables direct comparison across samples. Note that the genes displayed in these plots are derived from your `geneAnnotation` (i.e. the `BSgenome` object you used) so they may not match other online genome browsers that use different gene annotations.

### Usage

```
plotBrowserTrack(
  ArchRProj = NULL,
  region = NULL,
  groupBy = "Clusters",
  useGroups = NULL,
  sampleLabels = "Sample",
  plotSummary = c("bulkTrack", "featureTrack", "loopTrack", "geneTrack"),
  sizes = c(10, 1.5, 3, 4),
  features = getPeakSet(ArchRProj),
  loops = getCoAccessibility(ArchRProj),
  geneSymbol = NULL,
  useMatrix = NULL,
  log2Norm = TRUE,
  upstream = 50000,
  downstream = 50000,
  tileSize = 250,
  maxCells = 500,
  minCells = 25,
  normMethod = "ReadsInTSS",
  highlight = NULL,
  highlightFill = "firebrick3",
  threads = getArchRThreads(),
  ylim = NULL,
  pal = NULL,
  baseSize = 7,
  scTileSize = 0.5,
  scCellsMax = 100,
  borderWidth = 0.4,
  tickWidth = 0.4,
  facetbaseSize = 7,
  geneAnnotation = getGeneAnnotation(ArchRProj),
  title = "",
  verbose = TRUE,
  logFile = createLogFile("plotBrowserTrack")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `region` | A `GRanges` region that indicates the region to be plotted. If more than one region exists in the `GRanges` object, all will be plotted. If no region is supplied, then the `geneSymbol` argument can be used to center the plot window at the transcription start site of the supplied gene. |
| `groupBy` | A string that indicates how cells should be grouped. This string corresponds to one of the standard or user-supplied `cellColData` metadata columns (for example, "Clusters"). Cells with the same value annotated in this metadata column will be grouped together and the average signal will be plotted. |
| `useGroups` | A character vector that is used to select a subset of groups by name from the designated `groupBy` column in `cellColData`. This limits the groups to be plotted. |
| `sampleLabels` | The name of a column in `cellColData` to use to identify samples. In most cases, this parameter should be left as `NULL` and you should only use this parameter if you do not want to use the default sample labels stored in `cellColData$Sample`. However, if your individual Arrow files do not map to individual samples, then you should set this parameter to accurately identify your samples. This is the case in (for example) multiplexing applications where cells from different biological samples are mixed into the same reaction and demultiplexed based on a lipid barcode or genotype. |
| `plotSummary` | A character vector containing the features to be potted. Possible values include "bulkTrack" (the ATAC-seq signal), "scTrack" (scATAC-seq signal), "featureTrack" (i.e. the peak regions), "geneTrack" (line diagrams of genes with introns and exons shown. Blue-colored genes are on the minus strand and red-colored genes are on the plus strand), and "loopTrack" (links between a peak and a gene). |

| | |
|---|---|
| `sizes` | A numeric vector containing up to 3 values that indicate the sizes of the individual components passed to `plotSummary`. The order must be the same as `plotSummary`. |
| `features` | A `GRanges` (for a single feature track) or `GRangesList` (for multiple feature tracks) object containing the "features" to be plotted via the "featureTrack". This should be thought of as a bed track. i.e. the set of peaks obtained using `getPeakSet(ArchRProj))`. If you provide a `GRangesList`, then each element of that object must be named and this name will be used on the plot. For example - `GRangesList("peaks" = peak_gr, "other" = other_gr)`. |
| `loops` | A `GRanges` object containing the "loops" to be plotted via the "loopTrack". This `GRanges` object start represents the center position of one loop anchor and the end represents the center position of another loop anchor. A "loopTrack" draws an arc between two genomic regions that show some type of interaction. This type of track can be used to display chromosome conformation capture data or co-accessibility links obtained using `getCoAccessibility()`. |
| `geneSymbol` | If `region` is not supplied, plotting can be centered at the transcription start site corresponding to the gene symbol(s) passed here. |
| `useMatrix` | If supplied geneSymbol, one can plot the corresponding GeneScores/GeneExpression within this matrix. I.E. "GeneScoreMatrix" |
| `log2Norm` | If supplied geneSymbol, Log2 normalize the corresponding GeneScores/GeneExpression matrix before plotting. |
| `upstream` | The number of basepairs upstream of the transcription start site of `geneSymbol` to extend the plotting window. If `region` is supplied, this argument is ignored. |
| `downstream` | The number of basepairs downstream of the transcription start site of `geneSymbol` to extend the plotting window. If `region` is supplied, this argument is ignored. |
| `tileSize` | The numeric width of the tile/bin in basepairs for plotting ATAC-seq signal tracks. All insertions in a single bin will be summed. |
| `maxCells` | The maximum number of cells to use for obtaining data to plot as a bulk track. Using more cells can increase the resolution of your plots at the expense of increased processing time. |
| `minCells` | The minimum number of cells contained within a cell group to allow for this cell group to be plotted. This argument can be used to exclude pseudo-bulk replicates generated from low numbers of cells. |
| `normMethod` | The name of the column in `cellColData` by which normalization should be performed. The recommended and default value is "ReadsInTSS" which simultaneously normalizes tracks based on sequencing depth and sample data quality. |
| `highlight` | A `GRanges` object containing a region or regions on the plot to highlight. Multiple highlighted regions within the GRanges object are allowed Any highlight region that does not overlap the displayed region will be ignored. |
| `highlightFill` | The color to be used for the highlighted region designated by `highlight`. This can be a valid R color (i.e. "lightblue1") or a hex color (i.e. "#bfefff") |
| `threads` | The number of threads to use for parallel execution. |
| `ylim` | The numeric quantile y-axis limit to be used for for "bulkTrack" plotting. This should be expressed as `c(lower limit, upper limit)` such as `c(0,0.99)`. If not provided, the y-axis limit will be c(0, 0.999). |

| | |
|---|---|
| `pal` | A custom palette (see `paletteDiscrete` or `ArchRPalettes`) used to override coloring for groups. |
| `baseSize` | The numeric font size to be used in the plot. This applies to all plot labels. |
| `scTileSize` | The width of the tiles in scTracks. Larger numbers may make cells overlap more. Default is 0.5 for about 100 cells. |
| `scCellsMax` | The maximum number of cells for scTracks. |
| `borderWidth` | The numeric line width to be used for plot borders. |
| `tickWidth` | The numeric line width to be used for axis tick marks. |
| `facetbaseSize` | The numeric font size to be used in the facets (gray boxes used to provide track labels) of the plot. |
| `geneAnnotation` | The `geneAnnotation` object to be used for plotting the "geneTrack" object. See `createGeneAnnotation()` for more info. |
| `title` | The title to add at the top of the plot next to the plot's genomic coordinates. |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
#Get Test ArchR Project
proj <- getTestProject()

#Highlight
genes <- getGenes()
genes <- genes[which(genes$symbol %in% c("CD3D", "MS4A1"))]

#Plot Track
p <- plotBrowserTrack(proj, geneSymbol = c("CD3D", "MS4A1"), groupBy = "CellType", highlight = genes, highlightFil

#Plot PDF
plotPDF(p, name = "Track-CD3D-MS4A1", ArchRProj = proj)
```

---

`plotEmbedding` *Visualize an Embedding from ArchR Project*

---

## Description

This function will plot an embedding stored in an ArchRProject

## Usage

```
plotEmbedding(
  ArchRProj = NULL,
  embedding = "UMAP",
  colorBy = "cellColData",
  name = "Sample",
  log2Norm = NULL,
 imputeWeights = if (!grepl("coldata", tolower(colorBy[1]))) getImputeWeights(ArchRProj),
  pal = NULL,
  size = 0.1,
  sampleCells = NULL,
  highlightCells = NULL,
  rastr = TRUE,
  quantCut = c(0.01, 0.99),
  discreteSet = NULL,
  continuousSet = NULL,
  randomize = TRUE,
  keepAxis = FALSE,
  baseSize = 10,
  plotAs = NULL,
  threads = getArchRThreads(),
  logFile = createLogFile("plotEmbedding"),
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `embedding` | The name of the embedding stored in the `ArchRProject` to be plotted. See `computeEmbedding()` for more information. |
| `colorBy` | A string indicating whether points in the plot should be colored by a column in `cellColData` ("cellColData") or by a data matrix in the corresponding ArrowFiles (i.e. "GeneScoreMatrix", "MotifMatrix", "PeakMatrix"). |
| `name` | The name of the column in `cellColData` or the featureName/rowname of the data matrix to be used for plotting. For example if colorBy is "cellColData" then `name` refers to a column name in the `cellcoldata` (see `getCellcoldata()`). If `colorBy` is "GeneScoreMatrix" then `name` refers to a gene name which can be listed by `getFeatures(ArchRProj, useMatrix = "GeneScoreMatrix")`. |
| `log2Norm` | A boolean value indicating whether a log2 transformation should be performed on the values (if continuous) in plotting. |
| `imputeWeights` | The weights to be used for imputing numerical values for each cell as a linear combination of other cells values. See `addImputationWeights()` and `getImutationWeights()` for more information. |
| `pal` | A custom palette used to override discreteSet/continuousSet for coloring cells. Typically created using `paletteDiscrete()` or `paletteContinuous()`. To make a custom palette, you must construct this following strict specifications. If the coloring is for discrete data (i.e. "Clusters"), then this palette must be a named vector of colors where each color is named for the corresponding group (e.g. `"C1" = "#F97070"`). If the coloring for continuous data, then it just needs to be a vector of colors. If you are using `pal` in conjuction with `highlightCells`, your palette must be a named vector with two entries, one named for the value of the cells in the `name` column of `cellColData` and the |

| | |
|---|---|
| | other named "Non.Highlighted". For example, `pal=c("Mono" = "green", "Non.Highlighted" = "lightgrey")` would be used to change the color of cells with the value "Mono" in the `cellColData` column indicated by `name`. Because of this, the cells indicated by `highlightCells` must also match this value in the `name` column. |
| `size` | A number indicating the size of the points to plot if `plotAs` is set to "points". |
| `sampleCells` | A numeric describing number of cells to use for plot. If using impute weights, this will occur after imputation. |
| `highlightCells` | A character vector of cellNames describing which cells to hightlight if using `plotAs = "points"` (default if discrete). The remainder of cells will be colored light gray. |
| `rastr` | A boolean value that indicates whether the plot should be rasterized. This does not rasterize lines and labels, just the internal portions of the plot. |
| `quantCut` | If this is not `NULL`, a quantile cut is performed to threshold the top and bottom of the distribution of numerical values. This prevents skewed color scales caused by strong outliers. The format of this should be c(x,y) where x is the lower threshold and y is the upper threshold. For example, quantileCut = c(0.025,0.975) will take the 2.5th percentile and 97.5 percentile of values and set values below/above to the value of the 2.5th and 97.5th percentile values respectively. |
| `discreteSet` | The name of a discrete palette from `ArchRPalettes` for visualizing `colorBy` in the embedding if a discrete color set is desired. |
| `continuousSet` | The name of a continuous palette from `ArchRPalettes` for visualizing `colorBy` in the embedding if a continuous color set is desired. |
| `randomize` | A boolean value that indicates whether to randomize points prior to plotting to prevent cells from one cluster being uniformly present at the front of the plot. |
| `keepAxis` | A boolean value that indicates whether the x- and y-axis ticks and labels should be plotted. |
| `baseSize` | The base font size to use in the plot. |
| `plotAs` | A string that indicates whether points ("points") should be plotted or a hexplot ("hex") should be plotted. By default if `colorBy` is numeric, then `plotAs` is set to "hex". |
| `threads` | The number of threads to be used for parallel computing. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `...` | Additional parameters to pass to `ggPoint()` or `ggHex()`. |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Plot UMAP
p <- plotEmbedding(proj, name = "Clusters")

#PDF
plotPDF(p, name = "UMAP-Clusters", ArchRProj = proj)
```

## plotEnrichHeatmap — *Plot a Heatmap of Peak Annotation Hypergeometric Enrichment in Marker Peaks.*

### Description

This function will plot a heatmap of hypergeometric enrichment of a given peakAnnotation within the defined marker peaks.

### Usage

```
plotEnrichHeatmap(
  seEnrich = NULL,
  pal = paletteContinuous(set = "comet", n = 100),
  n = 10,
  cutOff = 20,
  pMax = Inf,
  clusterCols = TRUE,
  binaryClusterRows = TRUE,
  labelRows = TRUE,
  rastr = TRUE,
  transpose = FALSE,
  returnMatrix = FALSE,
  logFile = createLogFile("plotEnrichHeatmap")
)
```

### Arguments

| | |
|---|---|
| `seEnrich` | A `SummarizedExperiment` object containing peak enrichment information returned by `peakAnnoEnrichment()`. |
| `pal` | A custom continuous palette (see `paletteContinuous()`) used to override the default continuous palette for the heatmap. |
| `n` | The number of top enriched peakAnnotations per column from the `seMarker` to display in the heatmap. This number can be lowered to improve visibility of the heatmap. |
| `cutOff` | A numeric cutOff that indicates the minimum P-adj enrichment to be included in the heatmap. |
| `pMax` | A numeric representing the maximum P-adj for plotting in the heatmap. |
| `clusterCols` | A boolean indicating whether or not to cluster columns in the heatmap. |
| `binaryClusterRows` | A boolean indicating whether or not to cluster rows using binary classification in the heatmap. |
| `labelRows` | A boolean indicating whether or not to label all rows in the heatmap. |
| `rastr` | A boolean value that indicates whether the plot should be rasterized using `ComplexHeatmap`. This does not rasterize lines and labels, just the internal portions of the plot. |

| | |
|---|---|
| `transpose` | A boolean determining whether to transpose the heatmap in the plot. |
| `returnMatrix` | A boolean determining whether to return the matrix corresponding to the heatmap rather than generate a plot. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)

# Get Peak Annotation Enrichment
annoEnrich <- peakAnnoEnrichment(
  seMarker = seMarker,
  ArchRProj = proj,
  cutOff = "FDR <= 0.1 & Log2FC >= 0"
)

# Multiply by 50 since this is a super small test sample
assay(annoEnrich) <- assay(annoEnrich) * 50

#Plot
p <- plotEnrichHeatmap(annoEnrich)

#PDF
plotPDF(p, name = "PeakAnnoEnrich", ArchRProj = proj)
```

---

`plotFootprints` *Plot Footprints*

---

## Description

This function will get footprints for all samples in a given ArchRProject or a properly-formatted Summarized Experiment

## Usage

```
plotFootprints(
  seFoot = NULL,
  names = NULL,
  pal = NULL,
  flank = 250,
  flankNorm = 50,
  normMethod = "Subtract",
  smoothWindow = NULL,
  baseSize = 6,
  plot = TRUE,
  ArchRProj = NULL,
  plotName = paste0("Plot-Footprints-", normMethod),
  height = 6,
  width = 4,
  addDOC = TRUE,
  force = FALSE,
  logFile = createLogFile("plotFootprints")
)
```

## Arguments

| | |
|---|---|
| `seFoot` | A summarized experiment object containing information on footprints returned by the `getFootprints()` function. |
| `names` | A character vector containing the names of the transcription factors to be plotted. These should match colnames of `seFoot`. |
| `pal` | The name of a custom palette from `ArchRPalettes` to use for plotting the lines corresponding to the footprints. |
| `flank` | The number of basepairs from the position center (+/-) to consider as the flank. |
| `flankNorm` | The number of basepairs to consider at the edge of the flank region (+/-) to be used for footprint normalization. |
| `normMethod` | The name of the normalization method to use to normalize the footprint relative to the Tn5 insertion bias. Options include "none", "subtract", "divide". "Subtract" means subtracting the normalized Tn5 Bias. "Divide" means dividing the normalized Tn5 Bias. |
| `smoothWindow` | The size in basepairs of the sliding window to be used for smoothing of the footprint signal. |
| `baseSize` | A numeric specifying the baseSize of font in the plots. |
| `plot` | A boolean value indicating whether or not the footprints should be plotted (`TRUE`) or returned as grob objects (`FALSE`). |
| `ArchRProj` | An `ArchRProject` object to be used for plotting directory in `getOutputDirectory`. If no `ArchRProj` is supplied, then plots will be stored in a directory called "Plots" in the current working directory. |
| `plotName` | A string indicating the name/prefix of the file to be used for output plots. |
| `height` | The height in inches to be used for the output PDF file. |
| `width` | The width in inches to be used for the output PDF file. |
| `addDOC` | A boolean variable that determines whether to add the date of creation to end of the PDF file name. This is useful for preventing overwritting of old plots. |
| `force` | If many footprints are requested when plot = FALSE, please set force = TRUE. This prevents large amount of footprint plots stored as an object. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Motif Positions
positions <- getPositions(proj)

# Get Footprints
seFoot <- getFootprints(ArchRProj = proj, positions = positions, groupBy = "Clusters", minCells = 10)

# Plot Footprints
plotFootprints(seFoot, smoothWindow = 11)
```

---

`plotFragmentSizes` *Plot the fragment size distribution for each sample*

---

## Description

This function will plot a fragment size distribution for each sample. Only cells in the `ArchRProject` are used when making this plot.

## Usage

```
plotFragmentSizes(
  ArchRProj = NULL,
  groupBy = "Sample",
  chromSizes = getChromSizes(ArchRProj),
  maxSize = 750,
  pal = NULL,
  returnDF = FALSE,
  threads = getArchRThreads(),
  logFile = createLogFile("plotFragmentSizes")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for summarizing. |
| `chromSizes` | A GRanges object of the chromosome lengths. See `getChromSizes` for more info. |
| `maxSize` | The maximum fragment size (in basepairs) to be included when plotting the fragment size distribution. |
| `pal` | A color palette representing the groups from groupBy in fragment size plot. |

| | |
|---|---|
| `returnDF` | A boolean value that indicates whether to return a `data.frame` containing the plot information instead of plotting the fragment size distribution. |
| `threads` | An integer specifying the number of threads to use for calculation. By default this uses the number of threads set by `addArchRThreads()`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Plot Frag Sizes
p <- plotFragmentSizes(proj, groupBy = "Clusters")

# PDF
plotPDF(p, name = "Frag-Sizes", ArchRProj = proj)
```

---

`plotGroups` *Visualize Groups from ArchR Project*

---

## Description

This function will group, summarize and then plot data from an ArchRProject for visual comparison.

## Usage

```
plotGroups(
  ArchRProj = NULL,
  groupBy = "Sample",
  colorBy = "colData",
  name = "TSSEnrichment",
 imputeWeights = if (!grepl("coldata", tolower(colorBy[1]))) getImputeWeights(ArchRProj),
  maxCells = 1000,
  quantCut = c(0.002, 0.998),
  log2Norm = NULL,
  pal = NULL,
  discreteSet = "stallion",
  ylim = NULL,
  size = 0.5,
  baseSize = 6,
  ratioYX = NULL,
  ridgeScale = 2,
  plotAs = "ridges",
  threads = getArchRThreads(),
  ...
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for summarizing and plotting. |
| `colorBy` | A string indicating whether the numeric values to be used in the violin plot should be from a column in `cellColData` ("cellColData") or from a data matrix in the ArrowFiles (i.e. "GeneScoreMatrix", "MotifMatrix", "PeakMatrix"). |
| `name` | The name of the column in `cellColData` or the featureName/rowname of the data matrix to be used for plotting. For example if `colorBy` is "cellColData" then `name` refers to a column name in the cellcoldata (see `getCellcoldata()`). If `colorBy` is "GeneScoreMatrix" then `name` refers to a gene name which can be listed by `getFeatures(ArchRProj, useMatrix = "GeneScoreMatrix")`. |
| `imputeWeights` | The weights to be used for imputing numerical values for each cell as a linear combination of other cells values. See `addImputationWeights()` and `getImutationWeights()` for more information. |
| `maxCells` | The maximum cells to consider when making the plot. |
| `quantCut` | If this is not null, a quantile cut is performed to threshold the top and bottom of the distribution of values. This prevents skewed color scales caused by strong outliers. The format of this should be c(a,b) where `a` is the upper threshold and `b` is the lower threshold. For example, quantCut = c(0.025,0.975) will take the top and bottom 2.5 percent of values and set them to the value of the 97.5th and 2.5th percentile values respectively. |
| `log2Norm` | A boolean value indicating whether a log2 transformation should be performed on the values (if continuous) in plotting. |
| `pal` | A custom palette (see `paletteDiscrete` or `ArchRPalettes`) used to override discreteSet/continuousSet for coloring vector. |
| `discreteSet` | The name of a discrete palette from `ArchRPalettes` for visualizing `colorBy` if a discrete color set is desired. |
| `ylim` | A vector of two numeric values indicating the lower and upper bounds of the y-axis on the plot. |
| `size` | The numeric size of the points to be plotted. |
| `baseSize` | The base font size to use in the plot. |
| `ratioYX` | The aspect ratio of the x and y axes on the plot. |
| `ridgeScale` | The scale factor for the relative heights of each ridge when making a ridgeplot with `ggridges`. |
| `plotAs` | A string that indicates whether a rigdge plot ("ridges") should be plotted or a violin plot ("violin") should be plotted. |
| `threads` | The number of threads to be used for parallel computing. |
| `...` | Additional parameters to pass to `ggGroup()`. |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Plot Groups
p <- plotGroups(proj, groupBy = "Clusters", colorBy = "colData", name = "TSSEnrichment", plotAs = "violin", alpha =

#PDF
plotPDF(p, name = "Clusters-TSS", ArchRProj = proj)
```

---

## plotMarkerHeatmap *Plot a Heatmap of Identified Marker Features*

---

## Description

This function will plot a heatmap of the results from markerFeatures

## Usage

```
plotMarkerHeatmap(
  seMarker = NULL,
  cutOff = "FDR <= 0.01 & Log2FC >= 0.5",
  log2Norm = TRUE,
  scaleTo = 10^4,
  scaleRows = TRUE,
  plotLog2FC = FALSE,
  limits = c(-2, 2),
  grepExclude = NULL,
  pal = NULL,
  binaryClusterRows = TRUE,
  clusterCols = TRUE,
  subsetMarkers = NULL,
  labelMarkers = NULL,
  nLabel = 15,
  nPrint = 15,
  labelRows = FALSE,
  returnMatrix = FALSE,
  transpose = FALSE,
  invert = FALSE,
  logFile = createLogFile("plotMarkerHeatmap")
)
```

## Arguments

| | |
|---|---|
| `seMarker` | A `SummarizedExperiment` object returned by `getMarkerFeatures()`. |

| | |
|---|---|
| `cutOff` | A valid-syntax logical statement that defines which marker features from `seMarker` will be plotted in the heatmap. `cutoff` can contain any of the `assayNames` from `seMarker`. |
| `log2Norm` | A boolean value indicating whether a log2 transformation should be performed on the values in `seMarker` prior to plotting. Should be set to `TRUE` for counts-based assays (but not assays like "DeviationsMatrix"). This only applies if `plotLog2FC = FALSE`. |
| `scaleTo` | Each column in the assay Mean from `seMarker` will be normalized to a column sum designated by `scaleTo` prior to log2 normalization. If `log2Norm = FALSE` this option has no effect. |
| `scaleRows` | A boolean value that indicates whether the heatmap should display row-wise z-scores instead of raw values. |
| `plotLog2FC` | A boolean value that indicates whether the `log2(fold change)` values should be plotted on the heatmap or not. If `TRUE`, the `log2(fold change)` value (stored in the assay of `seMarker` labeled "Log2FC") is plotted. If `FALSE`, the mean value (stored in the assay of `seMarker` labeled "Mean") is plotted instead, with optional log normalization based on the `log2Norm` parameter. |
| `limits` | A numeric vector of two numbers that represent the lower and upper limits of the heatmap color scheme. |
| `grepExclude` | A character vector or string that indicates the `rownames` or a specific pattern that identifies rownames from `seMarker` to be excluded from the heatmap. |
| `pal` | A custom continuous palette from `ArchRPalettes` (see `paletteContinuous()`) used to override the default continuous palette for the heatmap. |
| `binaryClusterRows` | A boolean value that indicates whether a binary sorting algorithm should be used for fast clustering of heatmap rows. |
| `clusterCols` | A boolean value that indicates whether the columns of the marker heatmap should be clustered. |
| `subsetMarkers` | A vector of rownames from seMarker to use for subsetting of seMarker to only plot specific features on the heatmap. Note that these rownames are expected to be integers that come from `rownames(rowData(seMarker))`. If this parameter is used for subsetting, then the values provided to `cutOff` are effectively ignored. |
| `labelMarkers` | A character vector listing the `rownames` of `seMarker` that should be labeled on the side of the heatmap. |
| `nLabel` | An integer value that indicates how many of the top `n` features for each column in `seMarker` should be labeled on the side of the heatmap. To remove all feature labels, set `nLabel = 0`. |
| `nPrint` | If provided `seMarker` is from "GeneScoreMatrix" print the top `n` genes for each group based on how uniquely up-regulated the gene is. |
| `labelRows` | A boolean value that indicates whether all rows should be labeled on the side of the heatmap. |
| `returnMatrix` | A boolean value that indicates whether the final heatmap matrix should be returned in lieu of plotting the actual heatmap. |

| | |
|---|---|
| `transpose` | A boolean value that indicates whether the heatmap should be transposed prior to plotting or returning. |
| `invert` | A boolean value that indicates whether the heatmap will display the features with the lowest `log2(fold change)`. In this case, the heatmap will display features that are specifically lower in the given cell group compared to all other cell groups. Additionally, the color palette is inverted for visualization. This is useful when looking for down-regulated markers (`log2(fold change) < 0`) instead of up-regulated markers (`log2(fold change) > 0`). |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)

#Plot Markers
p <- plotMarkerHeatmap(seMarker)

#PDF
plotPDF(p, name = "Marker-Heatmap", ArchRProj = proj)
```

---

`plotMarkers` *Plot Differential Markers*

---

## Description

This function will plot one group/column of a differential markers as an MA or Volcano plot.

## Usage

```
plotMarkers(
  seMarker = NULL,
  name = NULL,
  cutOff = "FDR <= 0.01 & abs(Log2FC) >= 0.5",
  plotAs = "Volcano",
  scaleTo = 10^4,
  rastr = TRUE
)
```

## Arguments

| | |
|---|---|
| `seMarker` | A `SummarizedExperiment` object returned by `getMarkerFeatures()`. |
| `name` | The name of a column in `seMarker` (i.e. cell grouping in `groupBy` or `useGroups` for `getMarkerFeatures()`) to be plotted. To see available options try `colnames(seMarker)`. |
| `cutOff` | A valid-syntax logical statement that defines which marker features from `seMarker` will be plotted. `cutoff` can contain any of the `assayNames` from `seMarker`. |
| `plotAs` | A string indicating whether to plot a volcano plot ("Volcano") or an MA plot ("MA"). |
| `rastr` | A boolean value that indicates whether the plot should be rasterized using `ggrastr`. This does not rasterize lines and labels, just the internal portions of the plot. |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Get Markers
seMarker <- getMarkerFeatures(
  ArchRProj = proj,
  useMatrix = "PeakMatrix",
  testMethod = "binomial",
  binarize = TRUE
)

#Plot Markers
p <- plotMarkers(seMarker, name = "C1")

#PDF
plotPDF(p, name = "Marker-Plot", ArchRProj = proj)
```

---

`plotPDF` *Plot PDF in outputDirectory of an ArchRProject*

---

## Description

This function will save a plot or set of plots as a PDF file in the outputDirectory of a given ArchRProject.

## Usage

```
plotPDF(
  ...,
  name = "Plot",
  width = 6,
  height = 6,
  ArchRProj = NULL,
  addDOC = TRUE,
  useDingbats = FALSE,
  plotList = NULL
)
```

## Arguments

| | |
|---|---|
| `...` | vector of plots to be plotted (if input is a list use plotList instead) |
| `name` | The file name to be used for the output PDF file. |
| `width` | The width in inches to be used for the output PDF file. |
| `height` | The height in inches to be used for the output PDF. |
| `ArchRProj` | An `ArchRProject` object to be used for retrieving the desired `outputDirectory` which will be used to store the output plots in a subfolder called "plots". |
| `addDOC` | A boolean variable that determines whether to add the date of creation to the end of the PDF file name. This is useful for preventing overwritting of old plots. |
| `useDingbats` | A boolean variable that determines wheter to use dingbats characters for plotting points. |
| `plotList` | A `list` of plots to be printed to the output PDF file. Each element of `plotList` should be a printable plot formatted object (ggplot2, plot, heatmap, etc). |

## Examples

```
#Get Test Project
proj <- getTestProject()

#Plot UMAP
p <- plotEmbedding(proj, name = "Clusters")

#PDF
plotPDF(p, name = "UMAP-Clusters", ArchRProj = proj)
```

---

## `plotPeak2GeneHeatmap` *Plot Peak2Gene Heatmap from an ArchRProject*

---

## Description

This function plots side by side heatmaps of linked ATAC and Gene regions from `addPeak2GeneLinks`.

## Usage

```
plotPeak2GeneHeatmap(
  ArchRProj = NULL,
  corCutOff = 0.45,
  FDRCutOff = 1e-04,
  varCutOffATAC = 0.25,
  varCutOffRNA = 0.25,
  k = 25,
  nPlot = 25000,
  limitsATAC = c(-2, 2),
  limitsRNA = c(-2, 2),
  groupBy = "Clusters",
  palGroup = NULL,
  palATAC = paletteContinuous("solarExtra"),
  palRNA = paletteContinuous("blueYellow"),
  verbose = TRUE,
  returnMatrices = FALSE,
  seed = 1,
  logFile = createLogFile("plotPeak2GeneHeatmap")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `corCutOff` | A numeric describing the minimum numeric peak-to-gene correlation to return. |
| `FDRCutOff` | A numeric describing the maximum numeric peak-to-gene false discovery rate to return. |
| `varCutOffATAC` | A numeric describing the minimum variance quantile of the ATAC peak accessibility when selecting links. |
| `varCutOffRNA` | A numeric describing the minimum variance quantile of the RNA gene expression when selecting links. |
| `k` | An integer describing the number of k-means clusters to group peak-to-gene links prior to plotting heatmaps. |
| `nPlot` | An integer describing the maximum number of peak-to-gene links to plot in heatmap. |
| `limitsATAC` | An integer describing the maximum number of peak-to-gene links to plot in heatmap. |
| `limitsRNA` | An integer describing the maximum number of peak-to-gene links to plot in heatmap. |
| `groupBy` | The name of the column in `cellColData` to use for labeling KNN groupings. The maximum group appeared in the KNN groupings is used. |
| `palGroup` | A color palette describing the colors in `groupBy`. For example, if groupBy = "Clusters" try paletteDiscrete(ArchRProj$Clusters) for a color palette. |
| `palATAC` | A color palette describing the colors to be used for the ATAC heatmap. For example, paletteContinuous("solarExtra"). |
| `palRNA` | A color palette describing the colors to be used for the RNA heatmap. For example, paletteContinuous("blueYellow"). |
| `verbose` | A boolean value that determines whether standard output should be printed. |
| `returnMatrices` | A boolean value that determines whether the matrices should be returned with kmeans id versus plotting. |

| | |
|---|---|
| seed | A number to be used as the seed for random number generation. It is recommended to keep track of the seed used so that you can reproduce results downstream. |
| logFile | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Add P2G Links
proj <- addPeak2GeneLinks(proj, k = 20)

# Get P2G Links
p2g <- getPeak2GeneLinks(proj)

# Plot P2G
p <- plotPeak2GeneHeatmap(proj)
plotPDF(p, name = "P2G-Heatmap", ArchRProj = proj)
```

---

## plotTrajectory *Visualize a Trajectory from ArchR Project*

---

### Description

This function will plot a trajectory that was created onto an embedding.

### Usage

```
plotTrajectory(
  ArchRProj = NULL,
  embedding = "UMAP",
  trajectory = "Trajectory",
  colorBy = "colData",
  name = "Trajectory",
  log2Norm = NULL,
 imputeWeights = if (!grepl("coldata", tolower(colorBy[1]))) getImputeWeights(ArchRProj),
  pal = NULL,
  size = 0.2,
  rastr = TRUE,
  quantCut = c(0.01, 0.99),
  quantHex = 0.5,
  discreteSet = NULL,
  continuousSet = NULL,
  randomize = TRUE,
  keepAxis = FALSE,
  baseSize = 6,
  addArrow = TRUE,
  plotAs = NULL,
  smoothWindow = 5,
  logFile = createLogFile("plotTrajectory"),
  ...
)
```

## Arguments

| | |
|---|---|
| ArchRProj | An ArchRProject object. |
| embedding | The name of the embedding to use to visualize the given trajectory. See addEmbedding() for more information. |
| trajectory | The column name in cellColData that refers the trajectory to be plotted. See addTrajectory() for more information. |
| colorBy | A string indicating whether points in the plot should be colored by a column in cellColData ("cellColData") or by a data matrix in the associated ArrowFiles (i.e. "GeneScoreMatrix", "MotifMatrix", "PeakMatrix"). |
| name | The name of the column in cellColData or the featureName/rowname of the data matrix to be used for plotting. For example if colorBy is "cellColData" then name refers to a column name in the cellcoldata (see getCellcoldata()). If colorBy is "GeneScoreMatrix" then name refers to a gene name which can be listed by getFeatures(ArchRProj, useMatrix = "GeneScoreMatrix"). |
| log2Norm | A boolean value indicating whether a log2 transformation should be performed on the values from colorBy. |
| imputeWeights | The weights to be used for imputing numerical values for each cell as a linear combination of other cells' values. See addImputationWeights() and getImutationWeights() for more information. |
| pal | The name of a custom palette from ArchRPalettes to use for coloring cells. |
| size | A number indicating the size of the points to plot if plotAs is set to "points". |
| rastr | A boolean value that indicates whether the plot should be rasterized. This does not rasterize lines and labels, just the internal portions of the plot. |
| quantCut | If this is not NULL, a quantile cut is performed to threshold the top and bottom of the distribution of numerical values. This prevents skewed color scales caused by strong outliers. The format of this should be c(x,y) where x is the lower threshold and y is the upper threshold. For example, quantileCut = c(0.025,0.975) will take the 2.5th percentile and 97.5 percentile of values and set values below/above to the value of the 2.5th and 97.5th percentile values respectively. |
| quantHex | The numeric xth quantile of all dots within each individual hexagon will determine the numerical value for coloring to be displayed. This occurs when (i) plotAs is set to "hex" or (ii) plotAs is set to NULL and the values of colorBy are numeric. |
| discreteSet | The name of a discrete palette from ArchRPalettes for visualizing colorBy in the embedding if a discrete color set is desired. |
| continuousSet | The name of a continuous palette from ArchRPalettes for visualizing colorBy in the embedding if a continuous color set is desired. |

| | |
|---|---|
| `randomize` | A boolean value that indicates whether to randomize points prior to plotting to prevent cells from one cluster being present at the front of the plot. |
| `keepAxis` | A boolean value that indicates whether the x and y axis ticks and labels should be plotted. |
| `baseSize` | The base font size to use in the plot. |
| `addArrow` | A boolean value that indicates whether to add a smoothed arrow in the embedding based on the aligned trajectory. |
| `plotAs` | A string that indicates whether points ("points") should be plotted or a hexplot ("hex") should be plotted. By default if `colorBy` is numeric, then `plotAs` is set to "hex". |
| `smoothWindow` | An integer value indicating the smoothing window for creating inferred Arrow overlay on to embedding. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `...` | Additional parameters to pass to `ggPoint()` or `ggHex()`. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add Trajectory
proj <- addTrajectory(proj, trajectory = c("C1", "C2", "C3"), embedding = "UMAP", force = TRUE)

#Plot Trajectory
p <- plotTrajectory(proj, smoothWindow = 20)

#PDF
plotPDF(p, name = "Trajcetory", ArchRProj = proj)
```

---

`plotTrajectoryHeatmap` *Plot a Heatmap of Features across a Trajectory*

---

## Description

This function will plot a heatmap of the results from getTrajectory

## Usage

```
plotTrajectoryHeatmap(
  seTrajectory = NULL,
  varCutOff = 0.9,
  maxFeatures = 25000,
  scaleRows = TRUE,
  limits = c(-1.5, 1.5),
  grepExclude = NULL,
  pal = NULL,
  colorColumns = FALSE,
  columnPal = NULL,
  labelMarkers = NULL,
  labelTop = 50,
  labelRows = FALSE,
  rowOrder = NULL,
  useSeqnames = NULL,
  returnMatrix = FALSE,
  force = FALSE,
  logFile = createLogFile("plotTrajectoryHeatmap")
)
```

## Arguments

| | |
|---|---|
| `seTrajectory` | A `SummarizedExperiment` object that results from calling `getTrajectory()`. |
| `varCutOff` | The "Variance Quantile Cutoff" to be used for identifying the top variable features across the given trajectory. Only features with a variance above the provided quantile will be retained. |
| `maxFeatures` | The maximum number of features, ordered by variance, to consider from `useMatrix` when generating a trajectory. This prevents smoothing a large number number of features which can be very time consuming. |
| `scaleRows` | A boolean value that indicates whether row-wise z-scores should be computed on the matrix provided by `seTrajectory`. |
| `limits` | A numeric vector of two numbers that represent the lower and upper limits of the heatmap color scheme. |
| `grepExclude` | A character vector or string that indicates the `rownames` or a specific pattern that identifies rownames from `seTrajectory` to be excluded from the heatmap. |
| `pal` | A custom continuous palette (see `paletteContinuous()`) used to override the default continuous palette for the heatmap. |
| `colorColumns` | A boolean value that indicates whether a color bar should be added to label the columns. The color for each column will represent the most common label observed in the corresponding trajectory quantile as the heatmap is divided into quantile bins and thus does not display information for each cell. `colorColumns` can only be set to `TRUE` if the `trajectoryLabel` parameter was used in `getTrajectory()`. |
| `columnPal` | A discrete palette (see `paletteDiscrete()`) that maps the different labels of the trajectory to a unique color. This parameter is ignored unless `colorColumns = TRUE`. All labels shown in `unique(colData(seTrajectory)$label)` must be represented. |
| `labelMarkers` | A character vector listing the `rownames` of `seTrajectory` that should be labeled on the side of the heatmap. |
| `labelTop` | A number indicating how many of the top N features, based on variance, in `seTrajectory` should be labeled on the side of the heatmap. |
| `labelRows` | A boolean value that indicates whether all rows should be labeled on the side of the heatmap. |

| | |
|---|---|
| `rowOrder` | If wanting to set the order of rows to be plotted, the indices (integer or character correpsonding to rownmaes) can be provided here. |
| `useSeqnames` | A character vector that indicates which `seqnames` should be plotted in the heatmap. Features from `seqnames` that are not listed will be ignored. In the context of a `Sparse.Assays.Matrix`, such as a matrix containing chromVAR deviations, the `seqnames` do not correspond to chromosomes, rather they correspond to the sub-portions of the matrix, for example raw deviations ("deviations") or deviation z-scores ("z") for a chromVAR deviations matrix. |
| `returnMatrix` | A boolean value that indicates whether the final heatmap matrix should be returned in lieu of plotting the actual heatmap. |
| `force` | If useSeqnames is longer than 1 if matrixClass is "Sparse.Assays.Matrix" to continue. This is not recommended because these matrices can be in different units. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

#Add Trajectory
proj <- addTrajectory(proj, trajectory = c("C1", "C2", "C3"), embedding = "UMAP", force = TRUE)

#Get Trajectory
seTraj <- getTrajectory(proj)

#Plot Trajectory Heatmap
p <- plotTrajectoryHeatmap(seTraj)

#Plot PDF
plotPDF(p, name = "Trajectory-Heatmap", ArchRProj = proj)
```

---

## `plotTSSEnrichment` *Plot a TSS Enrichment Plot for Each Sample*

---

### Description

This function will plot a TSS enrichment plot for each sample. Cells in `ArchRProject` are the only ones used when making this plot.

### Usage

```
plotTSSEnrichment(
  ArchRProj = NULL,
  groupBy = "Sample",
  chromSizes = getChromSizes(ArchRProj),
  TSS = getTSS(ArchRProj),
  flank = 2000,
  norm = 100,
  smooth = 11,
  pal = NULL,
  returnDF = FALSE,
  threads = getArchRThreads(),
  logFile = createLogFile("plotTSSEnrichment")
)
```

## Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `groupBy` | The name of the column in `cellColData` to use for grouping cells together for summarizing. |
| `chromSizes` | A GRanges object of the chromosome lengths. See `getChromSizes` for more info. |
| `TSS` | A `GRanges` object containing the locations of stranded TSS regions. The default behavior is to try to retrieve this information from the `geneAnnotation` stored in the `ArchRProject`. |
| `flank` | An integer that specifies how far in bp (+/-) to extend the TSS for plotting. |
| `norm` | An integer that specifies the number of base pairs from the ends of the flanks to be used for normalization. For example if `flank=2000` and `norm=100`, the TSS insertions will be normalized by +/- 1900-2000 bp from the TSS. |
| `smooth` | An integer that indicates the smoothing window (in basepairs) to be applied to the TSS plot. |
| `pal` | A color palette representing the groups from groupBy in TSS plot. |
| `returnDF` | A boolean value that indicates whether to return a `data.frame` containing the plot information instead of plotting the TSS enrichment plot. |
| `threads` | An integer specifying the number of threads to use for calculation. By default this uses the number of threads set by `addArchRThreads()`. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Plot TSS
p <- plotTSSEnrichment(proj, groupBy = "Clusters")

# PDF
plotPDF(p, name = "TSS-Enrich", ArchRProj = proj)
```

## `projectBulkATAC` *Project Bulk ATAC-seq data into single cell subspace*

### Description

This function will Project Bulk ATAC-seq data into single cell subspace.

### Usage

```
projectBulkATAC(
  ArchRProj = NULL,
  seATAC = NULL,
  reducedDims = "IterativeLSI",
  embedding = "UMAP",
  n = 250,
  verbose = TRUE,
  threads = getArchRThreads(),
  force = FALSE,
  logFile = createLogFile("projectBulkATAC")
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object containing the dimensionality reduction matrix passed by `reducedDims`. |
| `seATAC` | A `SummarizedExperiment` object containing bulk ATAC-seq data. |
| `reducedDims` | A string specifying the name of the `reducedDims` object to be used. |
| `embedding` | A string specifying the name of the `embedding` object to be used. |
| `n` | An integer specifying the number of subsampled "pseudo single cells" per bulk sample. |
| `verbose` | A boolean value indicating whether to use verbose output during execution of this function. Can be set to FALSE for a cleaner output. |
| `threads` | The number of threads used for parallel execution |
| `force` | A boolean value indicating whether to force the projection of bulk ATAC data even if fewer than 25% of the features are present in the bulk ATAC data set. |
| `logFile` | The path to a file to be used for logging ArchR output. |

## `recoverArchRProject` *Recover ArchRProject if broken sampleColData/cellColData*

### Description

This function will recover an ArchRProject if it has broken sampleColData or cellColData due to different versions of bioconductor s4vectors.

### Usage

```
recoverArchRProject(ArchRProj)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |

### Examples

```
# Get Test Project
proj <- getTestProject()

# Try to Recover ArchR Project
proj <- recoverArchRProject(proj)
```

## `reformatFragmentFiles` *Reformat Fragment Files to be Tabix and Chr Sorted*

### Description

This function provides help in reformatting Fragment Files for reading in createArrowFiles. It will handle weird anomalies found that cause errors in reading tabix bgzip'd fragment files.

### Usage

```
reformatFragmentFiles(
  fragmentFiles = NULL,
  checkChrPrefix = getArchRChrPrefix()
)
```

### Arguments

| | |
|---|---|
| `fragmentFiles` | A character vector the paths to fragment files to be reformatted |
| `checkChrPrefix` | A boolean value that determines whether seqnames should be checked to contain "chr". IF set to `TRUE`, any seqnames that do not contain "chr" will be removed from the fragment files. |

### Examples

```
# Get Test Fragments
fragments <- getTestFragments()

# Get Peak Annotations
fragments2 <- reformatFragmentFiles(fragments)
```

## saveArchRProject *Save ArchRProject for Later Usage*

### Description

This function will organize arrows and project output into a directory and save the ArchRProject for later usage.

### Usage

```
saveArchRProject(
  ArchRProj = NULL,
  outputDirectory = getOutputDirectory(ArchRProj),
  overwrite = TRUE,
  load = TRUE,
  dropCells = FALSE,
  logFile = createLogFile("saveArchRProject"),
  threads = getArchRThreads()
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `outputDirectory` | A directory path to save all ArchR output and `ArchRProject` to. Default is outputDirectory of the `ArchRProject`. |
| `overwrite` | When writing to outputDirectory, overwrite existing files with new files. |
| `dropCells` | A boolean indicating whether to drop cells that are not in `ArchRProject` from corresponding Arrow Files. |
| `logFile` | The path to a file to be used for logging ArchR output. |
| `threads` | The number of threads to use for parallel execution. |

### Examples

```
# Get Small Test Project
proj <- getTestProject()

# Save
saveArchRProject(proj)
```

## setArchRLocking *Set a globally-applied H5 file locking setup*

### Description

This function will set the default H5 file locking parameters to the system

### Usage

```
setArchRLocking()
```

### Examples

```
# Set ArchR H5 Locking Globally
setArchRLocking()
```

## subsetArchRProject *Subset an ArchRProject for downstream analysis*

### Description

This function will subset and ArchRProject by cells and save the output to a new directory and re-load the subsetted ArchRProject.

### Usage

```
subsetArchRProject(
  ArchRProj = NULL,
  cells = getCellNames(ArchRProj),
  outputDirectory = "ArchRSubset",
  dropCells = TRUE,
  logFile = NULL,
  threads = getArchRThreads(),
  force = FALSE
)
```

### Arguments

| | |
|---|---|
| `ArchRProj` | An `ArchRProject` object. |
| `cells` | A vector of cells to subset `ArchRProject` by. Alternatively can provide a subset `ArchRProject`. |
| `outputDirectory` | A directory path to save all ArchR output and the subsetted `ArchRProject` to. |

| | |
|---|---|
| dropCells | A boolean indicating whether to drop cells that are not in `ArchRProject` from corresponding Arrow Files. |
| logFile | The path to a file to be used for logging ArchR output. |
| threads | The number of threads to use for parallel execution. |
| force | If output directory exists overwrite. |

## Examples

```
# Get Small Test Project
proj <- getTestProject()

#Subset
proj <- subsetArchRProject(proj, cells = getCellNames(proj)[1:50])
```

## subsetCells *Subset cells in an ArchRProject.*

### Description

This function returns an ArchRProject object that contains a specified subset of cells.

### Usage

```
subsetCells(ArchRProj = NULL, cellNames = NULL)
```

### Arguments

| | |
|---|---|
| ArchRProj | An `ArchRProject` object. |
| cellNames | A character vector of `cellNames` that will be subsetted of the current `ArchRProject`. |

### Examples

```
# Get Test ArchR Project
proj <- getTestProject()

# Get Peak Annotations
proj <- subsetCells(proj, getCellNames(proj)[1:50])
```

## theme_ArchR *ggplot2 default theme for ArchR*

### Description

This function returns a ggplot2 theme that is black borded with black font.

### Usage

```
theme_ArchR(
  color = "black",
  textFamily = "sans",
  baseSize = 10,
  baseLineSize = 0.5,
  baseRectSize = 0.5,
  plotMarginCm = 1,
  legendPosition = "bottom",
  legendTextSize = 5,
  axisTickCm = 0.1,
  xText90 = FALSE,
  yText90 = FALSE
)
```

### Arguments

| | |
|---|---|
| `color` | The color to be used for text, lines, ticks, etc for the plot. |
| `textFamily` | The font default family to be used for the plot. |
| `baseSize` | The base font size (in points) to use in the plot. |
| `baseLineSize` | The base line width (in points) to be used throughout the plot. |
| `baseRectSize` | The base line width (in points) to use for rectangular boxes throughout the plot. |
| `plotMarginCm` | The width in centimeters of the whitespace margin around the plot. |
| `legendPosition` | The location to put the legend. Valid options are "bottom", "top", "left", and "right. |
| `legendTextSize` | The base text size (in points) for the legend text. |
| `axisTickCm` | The length in centimeters to be used for the axis ticks. |
| `xText90` | A boolean value indicating whether the x-axis text should be rotated 90 degrees counterclockwise. |
| `yText90` | A boolean value indicating whether the y-axis text should be rotated 90 degrees counterclockwise. |

## Examples

```
# Create Random Data
m <- data.frame(x=matrix(rnorm(10, 2),ncol=1))
m$color <- sample(c("A", "B"), 10, replace = TRUE)

# Plot
p <- ggGroup(x = m$color, y = m$x) + theme_ArchR()

# To PDF
pdf("test.pdf", width = 4, height = 7)
p
dev.off()
```

---

## `validBSgenome` *Get/Validate BSgenome*

---

### Description

This function will attempt to get or validate an input as a BSgenome.

### Usage

```
validBSgenome(genome = NULL, masked = FALSE)
```

### Arguments

| | |
|---|---|
| `genome` | This option must be one of the following: (i) the name of a valid ArchR-supported genome ("hg38", "hg19", or "mm10"), (ii) the name of a `BSgenome` package (for ex. "BSgenome.Hsapiens.UCSC.hg19"), or (iii) a `BSgenome` object. |
| `masked` | A boolean describing whether or not to access the masked version of the selected genome. See `BSgenome::getBSgenome()`. |

---

## `[.ArchRProject` *Subset cells directly from ArchRProject*

---

### Description

This function will allow adding directly to cellColData with a `$` accessor.

### Usage

```
## S3 method for class 'ArchRProject'
x[i, j]
```

## `$.ArchRProject` *Accessing cellColData directly from dollar.sign accessor*

### Description

This function will allow direct access to cellColData with a `$` accessor.

### Usage

```
## S3 method for class 'ArchRProject'
x$i
```

## `$<-.ArchRProject` *Add directly to cellColData directly from dollar.sign accessor*

### Description

This function will allow adding directly to cellColData with a `$` accessor.

### Usage

```
## S3 replacement method for class 'ArchRProject'
x$i <- value
```

## `%bcin%` *Generic matching function for S4Vector objects*

### Description

This function provides a generic matching function for S4Vector objects primarily to avoid ambiguity.

### Usage

```
x %bcin% table
```

### Arguments

| | |
|---|---|
| `x` | An `S4Vector` object to search for in `table`. |
| `table` | The set of `S4Vector` objects to serve as the base for the match function. |

### Examples

```
#Test
Rle(c("A", "B", "C")) %bcin% Rle(c("A", "C"))
```

## %bcni% *Negated matching function for S4Vector objects*

### Description

This function provides the reciprocal of %bcin% for S4Vector objects primarily to avoid ambiguity.

### Usage

```
x %bcni% table
```

### Arguments

| | |
|---|---|
| x | An S4Vector object to search for in table. |
| table | The set of S4Vector objects to serve as the base for the match function. |

### Examples

```
#Test
Rle(c("A", "B", "C")) %bcni% Rle(c("A", "C"))
```

## %ni% *Negated Value Matching*

### Description

This function is the reciprocal of %in%. See the match funciton in base R.

### Usage

```
x %ni% table
```

### Arguments

| | |
|---|---|
| x | The value to search for in table. |
| table | The set of values to serve as the base for the match function. |

### Examples

```
#Test
c("A", "B", "C") %ni% c("A", "C")
```

# Index