

Package: sceptre (via r-universe)

September 30, 2026

Title Analysis of Single-Cell CRISPR Screen Data

Version 0.99.0

Description Analysis of single-cell CRISPR screen data. `sceptre` is an R package for statistically rigorous, massively scalable, and user-friendly single-cell CRISPR screen data analysis. The `sceptre` pipeline consists of several distinct steps: (1) import data from 10X CellRanger, Parse CRISPR Detect, or a set of R objects; (2) set the analysis parameters, which are the parameters that govern how the statistical analysis is to be conducted; (3) assign gRNAs to cells using one of three methods; (4) run quality control; (5) run the calibration check, which is an analysis that verifies that `sceptre` controls the rate of false positives on the dataset under analysis; (6) run the power check, which is an analysis that verifies that `sceptre` is capable of discovering true associations on the dataset under analysis; (7) run the discovery analysis to identify high-confidence perturbation-gene links among the set of perturbations and genes whose association status we do not know but seek to learn; (8) write outputs to a specified directory, including results, plots, and a textual summary of the analysis. `sceptre` leverages several novel statistical and computational algorithms to achieve high statistical accuracy and computational speed.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

LinkingTo Rcpp, BH

biocViews CRISPR, SingleCell, DifferentialExpression, GeneRegulation, DataImport, GeneTarget

Imports cowplot, crayon, data.table, dplyr, ggplot2, Matrix, methods, Rcpp (>= 1.0.9), scales, stats, purrr, parallelly (>= 1.23.0), withr

Depends R (>= 4.1)

Suggests ondisc, R.utils, sessioninfo, rmarkdown, knitr, testthat (>= 3.0.0), MASS

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://katsevich-lab.github.io/sceptre/>,
<https://timothy-barry.github.io/sceptre-book/>,
<https://github.com/Katsevich-Lab/sceptre>

BugReports <https://github.com/Katsevich-Lab/sceptre/issues>

LazyData false

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

Date/Publication 2026-08-31 16:31:54 UTC

RemoteUrl <https://github.com/BiocStaging/sceptre>

RemoteRef HEAD

RemoteSha 21f9ea098b69c444a884a49bc254de711968e3b5

Contents

sceptre-package	3
assign_grnas	6
construct_cis_pairs	8
construct_positive_control_pairs	9
construct_trans_pairs	10
gene_position_data_frame_grch38	12
get_grna_assignments	13
get_response_matrix	14
get_result	15
grna_target_data_frame_highmoi	16
highmoi_example_data	17
import_data	17
import_data_from_cellranger	19
import_data_from_parse	21
lowmoi_example_data	23
plot,sceptre_object-method	23
plot_assign_grnas	24
plot_covariates	26
plot_grna_count_distributions	27
plot_response_grna_target_pair	29
plot_run_calibration_check	30
plot_run_discovery_analysis	31
plot_run_power_check	33
plot_run_qc	34
print,sceptre_object-method	36

read_ondisc_backed_sceptre_object	37
run_calibration_check	38
run_discovery_analysis	40
run_power_check	41
run_qc	42
set_analysis_parameters	44
write_outputs_to_directory	46
Index	48

sceptre-package	sceptre
-----------------	---------

Description

sceptre is an R package for single-cell CRISPR screen data analysis, emphasizing statistical rigor, massive scalability, and ease of use.

Author(s)

Maintainer: Timothy Barry <tbarry@hsph.harvard.edu> ([ORCID](#))

Authors:

- Louis Deutsch
- Eugene Katsevich <ekatsevi@wharton.upenn.edu> ([ORCID](#))

Other contributors:

- Wharton Analytics [funder]
- National Science Foundation (Grants DMS-2113072 and DMS-2310654) [funder]

See Also

Useful links:

- <https://katsevich-lab.github.io/sceptre/>
- <https://timothy-barry.github.io/sceptre-book/>
- <https://github.com/Katsevich-Lab/sceptre>
- Report bugs at <https://github.com/Katsevich-Lab/sceptre/issues>

Examples

```
#####
# Low-MOI CRISPRko example
#####

# 1. create the sceptre object
data("lowmoi_example_data")
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  extra_covariates = lowmoi_example_data$extra_covariates,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
  moi = "low"
)
print(sceptre_object)

# 2. set the analysis parameters
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_trans_pairs(
  sceptre_object = sceptre_object,
  positive_control_pairs = positive_control_pairs,
  pairs_to_exclude = "pc_pairs"
)

sceptre_object <- set_analysis_parameters(
  sceptre_object = sceptre_object,
  discovery_pairs = discovery_pairs,
  positive_control_pairs = positive_control_pairs
)
print(sceptre_object)

# 3. assign grnas
plot_grna_count_distributions(sceptre_object)
sceptre_object <- sceptre_object |> assign_grnas()
plot(sceptre_object)
print(sceptre_object)

# 4. run qc
plot_covariates(sceptre_object, p_mito_threshold = 0.075)
sceptre_object <- sceptre_object |> run_qc(p_mito_threshold = 0.075)
plot(sceptre_object)
print(sceptre_object)

# 5. run the calibration check
sceptre_object <- run_calibration_check(sceptre_object)
plot(sceptre_object)
print(sceptre_object)

# 6. run power check
sceptre_object <- run_power_check(sceptre_object)
plot(sceptre_object)
print(sceptre_object)
```

```

# 7. run discovery analysis
sceptre_object <- run_discovery_analysis(sceptre_object)
plot(sceptre_object)
print(sceptre_object)

# 8. write results to a directory. tempdir() is used so this example is
# self-contained; for a real analysis choose a directory you can find again.
output_dir <- file.path(tempdir(), "sceptre_outputs_lowmoi")
write_outputs_to_directory(
  sceptre_object = sceptre_object,
  directory = output_dir
)
message(
  "sceptre outputs written to a temporary directory; ",
  'open with `browseURL("`', output_dir, "`")`'
)

#####
# High-MOI CRISPRi example
#####
# 1. create the sceptre object from cellranger output
directories <- paste0(
  system.file("extdata", package = "sceptre"),
  "/highmoi_example/gem_group_", c(1, 2)
)
data(grna_target_data_frame_highmoi)
sceptre_object <- import_data_from_cellranger(
  directories = directories,
  moi = "high",
  grna_target_data_frame = grna_target_data_frame_highmoi
)
print(sceptre_object)

# 2. set the analysis parameters
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)

sceptre_object <- set_analysis_parameters(
  sceptre_object = sceptre_object,
  discovery_pairs = discovery_pairs,
  positive_control_pairs = positive_control_pairs,
  side = "left"
)
print(sceptre_object)

# 3. assign grnas
plot_grna_count_distributions(sceptre_object)
sceptre_object <- sceptre_object |> assign_grnas()
plot(sceptre_object)

```

```

print(sceptre_object)

# 4. run qc
plot_covariates(sceptre_object, p_mito_threshold = 0.075)
sceptre_object <- sceptre_object |> run_qc(p_mito_threshold = 0.075)
plot(sceptre_object)
print(sceptre_object)

# 5. run the calibration check
sceptre_object <- run_calibration_check(sceptre_object)
plot(sceptre_object)
print(sceptre_object)

# 6. run the power check
sceptre_object <- run_power_check(sceptre_object)
plot(sceptre_object)
print(sceptre_object)

# 7. run discovery analysis
sceptre_object <- run_discovery_analysis(sceptre_object)
plot(sceptre_object)
print(sceptre_object)

# 8. write results to a directory. tempdir() is used so this example is
# self-contained; for a real analysis choose a directory you can find again.
output_dir <- file.path(tempdir(), "sceptre_outputs_highmoi")
write_outputs_to_directory(
  sceptre_object = sceptre_object,
  directory = output_dir
)
message(
  "sceptre outputs written to a temporary directory; ",
  "open by running `browseURL('', output_dir, '')` in the console."
)

```

assign_grnas

Assign gRNAs to cells

Description

assign_grnas() performs the gRNA-to-cell assignments. sceptre provides three gRNA-to-cell assignment strategies: the mixture method, the thresholding method, and the maximum method. The mixture method assigns gRNAs by thresholding fitted posterior membership probabilities. Next, the thresholding method assigns a gRNA to a cell if the UMI count of the gRNA in the cell is greater than or equal to some integer threshold. Finally, the maximum method assigns the gRNA that accounts for the greatest number of UMIs in a given cell to that cell. The maximum method is available only in low MOI. See [Chapter 3 of the manual](#) for more detailed information about assign_grnas().

Usage

```
assign_grnas(
  sceptre_object,
  method = "default",
  print_progress = TRUE,
  parallel = FALSE,
  n_processors = "auto",
  log_dir = tempdir(),
  ...
)
```

Arguments

sceptre_object	a sceptre_object
method	(optional) a string indicating the method to use to assign the gRNAs to cells, one of "mixture", "thresholding", or "maximum". The default is "maximum" in low MOI and "mixture" in high MOI.
print_progress	(optional; default TRUE) a logical indicating whether to print progress updates
parallel	(optional; default FALSE) a logical indicating whether to run the function in parallel. parallel = TRUE is recommended only on Mac; it is not supported on Windows and may behave unreliably on Linux clusters.
n_processors	(optional; default "auto") an integer specifying the number of processors to use if parallel is set to TRUE. The default, "auto", uses half the physical cores. The fraction may be tuned via the <code>parallelly.availableCores.fraction</code> R option.
log_dir	(optional; default tempdir()) a string indicating the directory in which to write the log files (ignored if parallel = FALSE)
...	optional method-specific additional arguments

Value

an updated `sceptre_object` in which the gRNA assignments have been carried out

Note

The "mixture" method retains the historical fitting algorithm used in `sceptre` version 0.9.0 and later. Owing to an implementation inconsistency, the perturbation parameter is estimated under a multiplicative parameterization but applied additively in the subsequent E-step. The resulting parameter estimates therefore do not maximize the likelihood of the mixture model used to compute the fitted probabilities. The historical implementation is retained because it has performed well in benchmarks. This limitation affects only "mixture"; "maximum" and "thresholding" are unaffected. We plan to replace the historical mixture implementation after a statistically coherent and empirically validated alternative is available. Any change to the public API or high-MOI default will follow that validation. Progress is tracked in [GitHub issue #220](#). See the manual for information about the method-specific additional arguments.

Examples

```
data("lowmoi_example_data")
# 1. import data, set default analysis parameters
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  extra_covariates = lowmoi_example_data$extra_covariates,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
  moi = "low"
) |> set_analysis_parameters()

# 2. assign gRNAs (three different methods)
sceptre_object <- sceptre_object |> assign_grnas(method = "thresholding")
sceptre_object <- sceptre_object |> assign_grnas(method = "maximum")
sceptre_object <- sceptre_object |> assign_grnas(method = "mixture")
```

construct_cis_pairs	<i>Construct cis pairs</i>
---------------------	----------------------------

Description

`construct_cis_pairs()` is a helper function to facilitate construction the *cis* pairs. `construct_cis_pairs()` returns the set of target-response pairs for which the target and response are located on the same chromosome and in close physical proximity to one another. `construct_cis_pairs()` is a useful pair constructor function for screens that aim to map noncoding regulatory elements (e.g., enhancers or noncoding GWAS variants) to target genes in *cis*. `construct_cis_pairs()` assumes that the columns `chr`, `start`, and `stop` are present in the `grna_target_data_frame`, giving the chromosome, start position, and end position, respectively, of the region that each gRNA targets. `construct_cis_pairs()` takes several arguments: `sceptre_object` (required), `distance_threshold` (optional), `positive_control_pairs` (optional), and `response_position_data_frame` (optional). By default, `construct_cis_pairs()` pairs each gRNA target to the set of responses on the same chromosome as that target and within `distance_threshold` bases of that target. (The default value of `distance_threshold` is 500,000 bases, or half a megabase.) The `positive_control_pairs` data frame optionally can be passed to `construct_cis_pairs()`, in which case the positive control targets (i.e., the entries within the `grna_target` column of `positive_control_pairs`) are excluded from the *cis* pairs. One may want to exclude these from the discovery analysis if these targets are intended for positive control purposes only. See [Section 2.2.2 of the manual](#) for more detailed information about this function.

Usage

```
construct_cis_pairs(
  sceptre_object,
  positive_control_pairs = data.frame(),
  distance_threshold = 500000L,
  response_position_data_frame = NULL
)
```

Arguments

`sceptre_object` a `sceptre_object`

`positive_control_pairs`
(optional) a data frame with columns `grna_target` and `response_id` containing the positive control pairs; if supplied, the positive control targets are excluded from the *cis* pairs.

`distance_threshold`
(optional) target-response pairs located within `distance_threshold` bases of one another and on the same chromosome are included in the *cis* discovery set.

`response_position_data_frame`
(optional) a data frame with columns `response_id`, `chr`, and `position` giving the genomic coordinate of each response; by default, the packaged GRCh38 gene position data frame is used.

Value

a data frame with columns `grna_target` and `response_id` containing the *cis* pairs

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)
```

```
construct_positive_control_pairs
```

Construct positive control pairs

Description

`construct_positive_control_pairs()` is a helper function to facilitate construction of the positive control pairs. Positive control pairs are target-response pairs for which we know (or have strong reason to believe) that there is a regulatory relationship between the target and the response. We

can use positive control pairs to verify that sceptre is sensitive (i.e., capable of detecting true associations) on the dataset under analysis. `construct_positive_control_pairs()` takes as an argument a `sceptre_object` and returns a data frame with columns `grna_target` and `response_id`, where gRNA targets and response IDs with matching names are paired. Typically, the positive control set consists of transcription start sites paired to the gene regulated by those transcription start sites. See [Section 2.2 in the manual](#) for more detailed information about this function.

Usage

```
construct_positive_control_pairs(sceptre_object)
```

Arguments

`sceptre_object` a `sceptre_object`

Value

a data frame with columns `grna_target` and `response_id` containing the positive control pairs

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
```

`construct_trans_pairs` *Construct trans pairs*

Description

`construct_trans_pairs()` is a helper function to facilitate construction the set of *trans* pairs. `construct_trans_pairs()` returns the entire set of possible target-response pairs. `construct_trans_pairs()` is a useful pair constructor function for analyses in which we seek to conduct a *trans* analysis, testing each target against each response. `construct_trans_pairs()` takes as arguments `sceptre_object` (required), `positive_control_pairs` (optional), and `pairs_to_exclude` (optional). By default `construct_trans_pairs()` returns a data frame with columns `grna_target` and `response_id`, where each gRNA target is mapped to each response ID.

Usage

```
construct_trans_pairs(
  sceptre_object,
  positive_control_pairs = data.frame(),
  pairs_to_exclude = "none"
)
```

Arguments

```
sceptre_object  a sceptre_object
positive_control_pairs
                  (optional) the set of positive control pairs
pairs_to_exclude
                  (optional; default "none") a string specifying pairs to exclude from the trans
                  pairs, one of "none", "pc_pairs", or "pairs_containing_pc_targets"
```

Details

The optional argument `pairs_to_exclude` enables the user to remove specific pairs from the *trans* set and takes values "none", "pc_pairs", or "pairs_containing_pc_targets". If `pairs_to_exclude` is set to "none" (the default), then no pairs are removed from the *trans* set. Next, if `pairs_to_exclude` is set to "pc_pairs" (and the `positive_control_pairs` data frame is passed), then the positive control target-response pairs are excluded from the *trans* set. Finally, if `pairs_to_exclude` is set to "pairs_containing_pc_targets" (and `positive_control_pairs` is passed), then *all* pairs containing a positive control gRNA target are excluded from the *trans* pairs. (In this sense setting `pairs_to_exclude` to "pairs_containing_pc_targets" is stronger than setting `pairs_to_exclude` to "pc_pairs".) Typically, in gene-targeting (resp., noncoding-regulatory-element-targeting) screens, we set `pairs_to_exclude` to "pc_pairs" (resp., "pairs_containing_pc_targets"). See [Section 2.2.2 of the manual](#) for more detailed information about this function.

Value

a data frame with columns `grna_target` and `response_id` containing the *trans* discovery set

Examples

```
# 1. low-moi, gene-targeting screen
data("lowmoi_example_data")
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  extra_covariates = lowmoi_example_data$extra_covariates,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
  moi = "low"
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_trans_pairs(
  sceptre_object = sceptre_object,
  positive_control_pairs = positive_control_pairs,
  pairs_to_exclude = "pc_pairs"
```

```

)

# 2. high-moi, enhancer-targeting screen
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_trans_pairs(
  sceptre_object = sceptre_object,
  positive_control_pairs = positive_control_pairs,
  pairs_to_exclude = "pairs_containing_pc_targets"
)

```

gene_position_data_frame_grch38

Gene position data frames

Description

gene_position_data_frame_grch38 and gene_position_data_frame_grch37 contain the coordinate and transcription start site position of each gene relative to reference genome GRCh38 and GRCh37, respectively. Both gene_position_data_frame_grch38 and gene_position_data_frame_grch37 were constructed from reference genomes available on the 10x Genomics website. The GRCh38 reference genome has been used by 10x Cell Ranger since 2020.

Usage

```
data(gene_position_data_frame_grch38)
```

```
data(gene_position_data_frame_grch37)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 36572 rows and 4 columns.

An object of class `data.table` (inherits from `data.frame`) with 57773 rows and 4 columns.

Examples

```

data(gene_position_data_frame_grch38)
data(gene_position_data_frame_grch37)
head(gene_position_data_frame_grch38)
head(gene_position_data_frame_grch37)

```

get_grna_assignments *Get gRNA assignments*

Description

get_grna_assignments() returns the gRNA-to-cell assignments contained within a sceptre_object. The output is a sparse logical matrix, with gRNAs in the rows and cells in the columns. A given entry of the matrix is set to TRUE if the given gRNA is assigned to the given cell (and FALSE otherwise).

Usage

```
get_grna_assignments(sceptre_object, apply_cellwise_qc = FALSE)
```

Arguments

sceptre_object a sceptre_object that has had assign_grnas() called on it
 apply_cellwise_qc a logical value (i.e., TRUE or FALSE) indicating whether to return the gRNA-to-cell assignment matrix after cellwise QC has been applied (default FALSE)

Details

When using the "maximum" assignment strategy, exactly one gRNA is assigned to a given cell. In other words, each column of the gRNA-to-cell assignment matrix contains exactly one TRUE entry.

Value

a sparse logical matrix containing the gRNA-to-cell assignments

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
discovery_pairs <- construct_cis_pairs(sceptre_object)
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    discovery_pairs = discovery_pairs,
    side = "left"
  ) |>
```

```

assign_grnas(method = "mixture") |>
run_qc()

grna_assignment_matrix <- get_grna_assignments(
  sceptre_object = sceptre_object
)
grna_assignment_matrix_with_qc <- get_grna_assignments(
  sceptre_object = sceptre_object,
  apply_cellwise_qc = TRUE
)

```

get_response_matrix *Data getter functions*

Description

The data getter functions (i.e., `get_response_matrix()`, `get_grna_matrix()`, `get_cell_covariates()`) return of a specified data field from a `sceptre_object`.

Usage

```

get_response_matrix(sceptre_object)

get_grna_matrix(sceptre_object)

get_cell_covariates(sceptre_object)

```

Arguments

`sceptre_object` a `sceptre_object`

Value

`get_response_matrix()` returns the response matrix contained within a `sceptre_object`.
`get_grna_matrix()` returns the gRNA matrix contained within a `sceptre_object`.
`get_cell_covariates()` returns the cell covariate data frame contained within a `sceptre_object`.

Examples

```

# 1. create a sceptre_object

data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",

```

```

    extra_covariates = highmoi_example_data$extra_covariates,
    response_names = highmoi_example_data$gene_names
  )

# 2. extract data fields from the sceptre_object
response_matrix <- get_response_matrix(sceptre_object)
grna_matrix <- get_grna_matrix(sceptre_object)
cell_covariates <- get_cell_covariates(sceptre_object)

```

get_result

Get result

Description

`get_result()` returns a data frame containing the result of a calibration check, power check, or discovery analysis. We pass as arguments `sceptre_object` and `analysis`, where the latter is a string indicating the function whose results we are querying. The output is a data frame, the rows of which correspond to target-response pairs, and the columns of which are as follows: `response_id`, `grna_target`, `n_nonzero_trt`, `n_nonzero_cntrl`, `pass_qc` (a TRUE/FALSE value indicating whether the pair passes pairwise QC), `p_value`, `fold_change`, `se_fold_change` (standard error for the fold change estimate), `log2_fold_change`, and `significant` (a TRUE/FALSE value indicating whether the pair is called as significant). The p-value contained within the `p_value` column is a raw (i.e., non-multiplicity-adjusted) p-value. See [Section 8 of the introductory chapter in the manual](#) for more information about this function.

Usage

```
get_result(sceptre_object, analysis)
```

Arguments

`sceptre_object` a `sceptre_object`

`analysis` a string indicating the name of the analysis whose results we are querying, one of "run_calibration_check", "run_power_check", or "run_discovery_analysis".

Value

a data frame containing the results of the analysis

Note

If `output_amount` is set to 2 or 3 in `run_calibration_check()`, `run_power_check()`, or `run_discovery_analysis()`, then the result data frame contains additional columns; see [Chapter 6 in the manual](#) for more information.

Examples

```

data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
pc_result <- sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations",
    positive_control_pairs = positive_control_pairs
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_power_check() |>
  get_result("run_power_check")

```

```

grna_target_data_frame_highmoi
      gRNA target data frame

```

Description

The gRNA target data frame corresponding to the example high MOI data. The data frame contains the columns `grna_id` (ID of an individual gRNA), `grna_target` (genomic target of the gRNA), `chr` (target chromosome), `start` (start coordinate of target location), and `end` (end coordinate of target location).

Usage

```
data(grna_target_data_frame_highmoi)
```

Format

An object of class `data.frame` with 60 rows and 5 columns.

highmoi_example_data	<i>Example high-MOI data</i>
----------------------	------------------------------

Description

The example high-MOI CRISPRi data are a small simulated dataset modeled on that of "A genome-wide framework for mapping gene regulation via cellular genetic screens" by Gasperini et al., 2019. These data contain perturbations both of gene transcription start sites (TSSs) and enhancers, as well as non- targeting perturbations.

Usage

```
data(highmoi_example_data)
```

Format

An object of class `list` of length 4.

Details

The example data are stored in a list containing the following components:

- `response_matrix`: the gene-by-cell expression matrix
- `grna_matrix`: the gRNA-by-cell expression matrix
- `extra_covariates`: a data frame containing a single column, `batch`, specifying the batch in which each cell was sequenced (`batch_1` or `batch_2`); the row names give the cell IDs and match the matrix column names
- `gene_names`: the human-readable name of each gene

import_data	<i>Import data</i>
-------------	--------------------

Description

`import_data()` imports data from a collection of R objects to create a `sceptre_object`. Users can create either a standard `sceptre` object or an `ondisc-backed` `sceptre` object; the latter is more appropriate for large-scale data. See [Chapter 1 of the manual](#) for more detailed information about this function.

Usage

```
import_data(
  response_matrix,
  grna_matrix,
  grna_target_data_frame,
  moi,
  extra_covariates = data.frame(),
  response_names = NA_character_,
  use_ondisc = FALSE,
  directory_to_write = NULL
)
```

Arguments

<code>response_matrix</code>	a matrix of response UMI counts, with responses in rows and cells in columns. The matrix should be of type "matrix", "dgCMatrix", "dgRMatrix", or "dgTMatrix". The row names of the matrix should give the response IDs. The column names, when supplied, should give the cell IDs.
<code>grna_matrix</code>	a matrix of gRNA UMI counts, with gRNAs in rows and cells in columns. The matrix should be of type "matrix", "dgCMatrix", "dgRMatrix", or "dgTMatrix". The row names of the matrix should give the gRNA IDs. The column names, when supplied, should give the cell IDs.
<code>grna_target_data_frame</code>	a data frame containing columns <code>grna_id</code> and <code>grna_target</code> mapping each individual gRNA to its target. Non-targeting gRNAs should be assigned a label of "non-targeting". Optionally, <code>grna_target_data_frame</code> can contain columns <code>chr</code> , <code>start</code> , and <code>end</code> , giving the chromosome, start coordinate, and end coordinate, respectively, of each gRNA. Additionally, <code>grna_target_data_frame</code> can contain the column <code>vector_id</code> specifying the vector to which a given gRNA belongs.
<code>moi</code>	a string indicating the MOI of the dataset, either "low" or "high".
<code>extra_covariates</code>	(optional) a data frame containing extra covariates (e.g., batch, biological replicate) beyond those that <code>sceptre</code> can compute. The rows should follow the same cell ordering as the columns of <code>response_matrix</code> and <code>grna_matrix</code> . Cell IDs may be supplied as row names. When cell IDs are supplied for at least two of these three objects, they must be identical.
<code>response_names</code>	(optional) a vector of human-readable response names; names with the prefix "MT-" are taken as mitochondrial genes and are used to compute the covariate <code>response_p_mito</code> .
<code>use_ondisc</code>	(default FALSE) a logical value (i.e., TRUE or FALSE) indicating whether to create an ondisc-backed <code>sceptre_object</code> (TRUE) or a standard <code>sceptre_object</code> (FALSE).
<code>directory_to_write</code>	(optional) a file path to a directory in which to write the backing <code>.odm</code> files for the response and gRNA expression matrices. Must be supplied if <code>use_ondisc</code>

is set to TRUE.

Value

an initialized `sceptre_object`

Examples

```
data("lowmoi_example_data")
# 1. initialize a standard sceptre_object from R objects
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
  extra_covariates = lowmoi_example_data$extra_covariates,
  moi = "low"
)

# 2. initialize an ondisc-backed sceptre_object from R objects
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
  extra_covariates = lowmoi_example_data$extra_covariates,
  moi = "low",
  use_ondisc = TRUE,
  directory_to_write = tempdir()
)
```

import_data_from_cellranger

Import data from Cell Ranger

Description

`import_data_from_cellranger()` imports data from the output of one or more calls to Cell Ranger count. Each directory supplied as an input to this function should be in feature-barcode format, containing the files `features.tsv.gz` and `matrix.mtx.gz` (and optionally `barcodes.tsv.gz`). For standard in-memory objects, when every supplied directory contains a barcode file, the cell barcodes are retained as the column names of the response and gRNA matrices and the row names of the cell covariate data frame. When multiple directories are supplied, the retained cell barcodes are prefixed with their batch identifiers (for example, `b1_` and `b2_`) to keep the cell IDs unique and stable across imports. Supplying barcode files for only some directories is an error. Users can create either a standard `sceptre` object or an ondisc-backed `sceptre` object; the latter is more appropriate for large-scale data. See [the introductory chapter](#) or [Chapter 1](#) of the manual for more information about this function.

Usage

```
import_data_from_cellranger(
  directories,
  moi,
  grna_target_data_frame,
  extra_covariates = data.frame(),
  use_ondisc = FALSE,
  directory_to_write = NULL
)
```

Arguments

<code>directories</code>	a character vector of file paths to directories containing the output of one or more calls to Cell Ranger count. Each directory should contain the files <code>matrix.mtx.gz</code> and <code>features.tsv.gz</code> (and optionally <code>barcodes.tsv.gz</code>).
<code>moi</code>	a string indicating the MOI of the dataset, either "low" or "high".
<code>grna_target_data_frame</code>	a data frame containing columns <code>grna_id</code> and <code>grna_target</code> mapping each individual gRNA to its target. Non-targeting gRNAs should be assigned a label of "non-targeting". Optionally, <code>grna_target_data_frame</code> can contain columns <code>chr</code> , <code>start</code> , and <code>end</code> , giving the chromosome, start coordinate, and end coordinate, respectively, of each gRNA. Additionally, <code>grna_target_data_frame</code> can contain the column vector <code>vector_id</code> specifying the vector to which a given gRNA belongs.
<code>extra_covariates</code>	(optional) a data frame containing extra covariates (e.g., batch, biological replicate) beyond those that <code>sceptre</code> can compute. If row names are supplied and cell barcodes are retained, the row names must match the retained (and, for multiple directories, batch-prefixed) cell IDs.
<code>use_ondisc</code>	(optional; default FALSE) a logical indicating whether to store the expression data in a disk-backed ondisc matrix (TRUE) or an in-memory sparse matrix (FALSE).
<code>directory_to_write</code>	(optional) a string indicating the directory in which to write the backing <code>.odm</code> files (must be specified if <code>use_ondisc</code> is set to TRUE).

Value

an initialized `sceptre_object`

Examples

```
data(grna_target_data_frame_highmoi)
directories <- paste0(
  system.file("extdata", package = "sceptre"),
  "/highmoi_example/gem_group-", c(1, 2)
)
```

```
# 1. create a standard sceptre_object from Cell Ranger output
sceptre_object <- import_data_from_cellranger(
  directories = directories,
  moi = "high",
  grna_target_data_frame = grna_target_data_frame_highmoi,
)

# 2. create an ondisc-backed sceptre_object from Cell Ranger output
sceptre_object <- import_data_from_cellranger(
  directories = directories,
  moi = "high",
  grna_target_data_frame = grna_target_data_frame_highmoi,
  use_ondisc = TRUE,
  directory_to_write = tempdir()
)
```

import_data_from_parse

Import data from Parse (experimental)

Description

import_data_from_parse() imports data from the output of the Parse count matrix generation program. See [Chapter 1 of the manual](#) for more information about this function.

Usage

```
import_data_from_parse(
  gene_mat_fp,
  grna_mat_fp,
  all_genes_fp,
  all_grnas_fp,
  moi,
  grna_target_data_frame,
  extra_covariates = data.frame()
)
```

Arguments

gene_mat_fp	file path to the gene count_matrix.mtx file.
grna_mat_fp	file path to the gRNA count_matrix.mtx file.
all_genes_fp	file path to the all_genes.csv file containing the gene IDs.
all_grnas_fp	file path to the all_guides.csv file containing the gRNA IDs. The gRNA IDs are assumed to be in the second column (i.e., the "gene_name" column) of this file.
moi	a string indicating the MOI of the dataset, either "low" or "high".

`grna_target_data_frame`

a data frame containing columns `grna_id` and `grna_target` mapping each individual gRNA to its target. Non-targeting gRNAs should be assigned a label of "non-targeting". Optionally, `grna_target_data_frame` can contain columns `chr`, `start`, and `end`, giving the chromosome, start coordinate, and end coordinate, respectively, of each gRNA. Additionally, `grna_target_data_frame` can contain the column `vector_id` specifying the vector to which a given gRNA belongs.

`extra_covariates`

(optional) a data frame containing extra covariates (e.g., batch, biological replicate) beyond those that `sceptre` can compute.

Details

`import_data_from_parse()` is experimental, and the API of this function is subject to change. We expect the API to solidify as we learn more about the Parse platform and the structure of the Parse count matrix generation program output.

The example data shipped with the package (used in the examples below) are a small subset of Parse Biosciences' "Technical Performance of CRISPR Detect in Cell Lines" demo dataset, used under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, with attribution to Parse Biosciences. See <https://www.parsebiosciences.com/datasets/technical-performance-of-crispr-detect-in-c>

Value

an initialized `sceptre_object`

Examples

```
directory <- paste0(
  system.file("extdata", package = "sceptre"),
  "/parse_example/"
)
gene_mat_fp <- paste0(directory, "gene_mat.mtx")
grna_mat_fp <- paste0(directory, "grna_mat.mtx")
all_genes_fp <- paste0(directory, "all_genes.csv")
all_grnas_fp <- paste0(directory, "all_grnas.csv")
grna_target_data_frame <- data.frame(
  grna_id = c("guide_A", "guide_B", "guide_C"),
  grna_target = c("target-A", "target-B", "non-targeting")
)
sceptre_object <- import_data_from_parse(
  gene_mat_fp = gene_mat_fp,
  grna_mat_fp = grna_mat_fp,
  all_genes_fp = all_genes_fp,
  all_grnas_fp = all_grnas_fp,
  moi = "low",
  grna_target_data_frame = grna_target_data_frame
)
```

lowmoi_example_data	<i>Example low-MOI data</i>
---------------------	-----------------------------

Description

The example low-MOI data are a small simulated dataset modeled on that in the paper "Characterizing the molecular regulation of inhibitory immune checkpoints with multimodal single-cell screens" by Papalexi et al., 2021. This dataset includes gene-targeting CRISPRko perturbations as well as non-targeting perturbations.

Usage

```
data(lowmoi_example_data)
```

Format

An object of class `list` of length 4.

Details

The example data are stored in a list containing four components:

- `response_matrix`: the gene-by-cell expression matrix
- `grna_matrix`: the gRNA-by-cell expression matrix
- `grna_target_data_frame`: a data frame containing the columns `grna_id` (ID of an individual gRNA) and `grna_target` (genomic target of the gRNA)
- `extra_covariates`: a data frame with a single column, `batch`, specifying the batch in which each cell was sequenced (`batch_1` or `batch_2`); the row names give the cell IDs and match the matrix column names

<code>plot,sceptre_object-method</code>
<i>Plot</i>

Description

`plot()` creates a plot depicting the current state of a `sceptre_object`.

Usage

```
## S4 method for signature 'sceptre_object'  
plot(x, y, ...)
```

Arguments

x	a sceptre_object
y	ignored argument
...	arguments passed to the plotting function dispatched by plot()

Details

plot() is "generic" in the sense that it dispatches a specific plotting function based on the pipeline function that was most recently called on the sceptre_object. For example, if run_assign_grnas() is the most recently called pipeline function, then plot() dispatches plot_run_assign_grnas(). Similarly, if run_power_check() is the most recently called pipeline function, then plot() dispatches plot_run_power_check(), and so on. Users can pass arguments to the function dispatched by plot() as named arguments to plot().

Value

a single cowplot object containing the combined panels (if return_indiv_plots is set to TRUE) or a list of the individual panels (if return_indiv_plots is set to FALSE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
# set analysis parameters, assign grnas
sceptre_object <- sceptre_object |>
  set_analysis_parameters() |>
  assign_grnas(method = "thresholding") |>
  plot()
```

plot_assign_grnas	<i>Plot assign gRNAs</i>
-------------------	--------------------------

Description

plot_assign_grnas() plots the outcome of the gRNA-to-cell assignment step. The top panel plots the gRNA-to-cell assignments of n_grnas_to_plot (default 3) randomly selected gRNAs. In each plot the points represent cells; the vertical axis indicates the UMI count of the gRNA in a given cell, and the horizontal axis indicates whether the cell has been classified as "perturbed" (i.e., it contains

the gRNA) or unperturbed (i.e., it does not contain the gRNA). Perturbed (resp., unperturbed) cells are shown in the left (resp., right) column. The bottom left panel is a barplot of the number of cells to which each gRNA has been mapped. Finally, the bottom right panel is a histogram of the number of gRNAs contained in each cell. The mean number of gRNAs per cell — i.e., the MOI — is displayed in purple text.

Usage

```
plot_assign_grnas(
  sceptre_object,
  n_grnas_to_plot = 3L,
  grnas_to_plot = NULL,
  point_size = 0.9,
  transparency = 0.8,
  return_indiv_plots = FALSE
)
```

Arguments

sceptre_object a `sceptre_object` that has had `assign_grnas()` called on it

n_grnas_to_plot (optional; default 3) the number of gRNAs to display in the plots of gRNA count versus cell assignment

grnas_to_plot (optional; default NULL) a character vector giving the names of specific gRNAs to plot; if NULL, then the gRNAs are chosen at random.

point_size (optional; default 0.9) the size of the individual points in the plot

transparency (optional; default 0.8) the transparency of the individual points in the plot

return_indiv_plots (optional; default FALSE) if FALSE, then a list of `ggplot` objects is returned; if TRUE then a single `cowplot` object is returned.

Value

a single `cowplot` object containing the combined panels (if `return_indiv_plots` is set to TRUE) or a list of the individual panels (if `return_indiv_plots` is set to FALSE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
) |>
  set_analysis_parameters() |>
```

```
assign_grnas(method = "thresholding") |>
plot_assign_grnas()
```

plot_covariates

Plot covariates

Description

`plot_covariates()` creates a histogram of the covariates `response_n_nonzero`, `response_n_umis`, and (if applicable) `response_p_mito`. Cellwise QC removes cells that lie in the extreme right tail of the `response_p_mito` distribution or that lie in the extreme left *or* right tail of the `response_n_nonzero` or `response_n_umis` distribution. To help guide the selection of QC thresholds, `plot_covariates()` plots candidate QC thresholds as vertical lines on the histograms. The optional arguments `response_n_nonzero_range`, `response_n_umis_range`, and `p_mito_threshold` control the location of these candidate QC thresholds. `response_n_nonzero_range` (resp., `response_n_umis_range`) is a length-two vector of quantiles (default: `c(0.01, 0.99)`) indicating the location at which to draw candidate QC thresholds on the `response_n_nonzero` (resp., `response_n_umis`) histogram. Next, `p_mito_threshold` is a single numeric value in the interval `[0,1]` specifying the location at which to draw a candidate QC threshold on the `response_p_mito` plot.

Usage

```
plot_covariates(
  sceptre_object,
  response_n_umis_range = c(0.01, 0.99),
  response_n_nonzero_range = c(0.01, 0.99),
  p_mito_threshold = 0.2,
  return_indiv_plots = FALSE
)
```

Arguments

`sceptre_object` a `sceptre_object`

`response_n_umis_range`

(optional; default `c(0.01, 0.99)`) a length-2 vector of quantiles indicating the location at which to draw vertical lines on the `response_n_umis` histogram

`response_n_nonzero_range`

(optional; default `c(0.01, 0.99)`) a length-2 vector of quantiles indicating the location at which to draw vertical lines on the `response_n_nonzero` histogram

`p_mito_threshold`

(optional; default `0.2`) a single numeric value in the interval `[0,1]` specifying the location at which to draw a vertical line on the `response_p_mito` histogram. Note that `p_mito_threshold` is an absolute number rather than a percentile.

`return_indiv_plots`

(optional; default `FALSE`) if `FALSE`, then a list of `ggplot` objects is returned; if `TRUE` then a single `cowplot` object is returned.

Value

a single cowplot object containing the combined panels (if return_indiv_plots is set to TRUE)
or a list of the individual panels (if return_indiv_plots is set to FALSE)

Note

If run_qc() has already been called on the sceptre_object, then the parameters response_n_umis_range, response_n_nonzero_range, and p_mito_threshold are set to the corresponding parameters within the sceptre_object.

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
) |> plot_covariates(p_mito_threshold = 0.07)
```

plot_grna_count_distributions

Plot gRNA count distributions

Description

plot_grna_count_distributions() plots the empirical UMI count distribution of one or more gRNAs. plot_grna_count_distributions() can be called on a sceptre_object at any point in the pipeline after import_data().

Usage

```
plot_grna_count_distributions(
  sceptre_object,
  n_grnas_to_plot = 4L,
  grnas_to_plot = NULL,
  threshold = NULL,
  return_indiv_plots = FALSE
)
```

Arguments

`sceptre_object` a `sceptre_object`

`n_grnas_to_plot` (optional; default 4) an integer specifying the number of randomly selected gRNAs to plot

`grnas_to_plot` (optional; default NULL) a character vector giving the names of one or more specific gRNAs to plot. If NULL, then `n_grnas_to_plot` random gRNAs are plotted.

`threshold` (optional; default NULL) an integer representing a gRNA count cut-off; if provided, the bins of length 1 will go up to and include this value, after which the exponentially growing bins begin. A vertical line is also drawn at this value. If NULL, then 10 is the largest gRNA count with its own bin. Non-integer values will be rounded.

`return_indiv_plots` (optional; default FALSE) if FALSE, a single combined cowplot object is returned; if TRUE, a list of the per-gRNA ggplot panels is returned instead.

Details

The x-axis is a piecewise linear-log scale, with bins of size 1 going from gRNA counts of 0 up to `max(10, threshold)`, and then the bin widths grow exponentially in size. The number under each bar indicates the first value that is counted for that bar, and that bar includes all integers from that label up until the integer immediately preceding the label of the next bar on the right. For example, if one bar has a label of "23" and the next bar on the right has a label of "26" then the bar with the label of "23" counts values of 23, 24, and 25 in the data.

Value

a single cowplot object containing the combined panels (if `return_indiv_plots` is FALSE, the default) or a list of the per-gRNA ggplot2 panels (if `return_indiv_plots` is TRUE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
) |> plot_grna_count_distributions()
```

```
plot_response_grna_target_pair
```

Plot response-gRNA-target pair

Description

`plot_response_grna_target_pair()` creates a violin plot of the expression level of a given response as a function of the "treatment status" (i.e., treatment or control) of a given gRNA target. The left (resp., right) violin plot shows the expression level of the response in treatment (resp., control) cells. The expression level is normalized by dividing by `n_response_umis`, adding a pseudo-count of 1 and then taking the log transform. If the given response-gRNA-target pair has been analyzed, the p-value for the test of association also is displayed.

Usage

```
plot_response_grna_target_pair(sceptre_object, response_id, grna_target)
```

Arguments

`sceptre_object` a `sceptre_object` that has had `run_qc()` called on it

`response_id` a string containing a response ID

`grna_target` a string containing a gRNA target (or, if `grna_integration_strategy` is set to "singleton", an individual gRNA ID)

Details

If `grna_integration_strategy` is set to "singleton", then `grna_target` should be set to a gRNA ID.

Value

a violin plot

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
discovery_pairs <- construct_cis_pairs(sceptre_object)
sceptre_object |>
```

```

set_analysis_parameters(
  side = "left",
  discovery_pairs = discovery_pairs,
  resampling_mechanism = "permutations",
) |>
assign_grnas(method = "thresholding") |>
run_qc() |>
run_discovery_analysis() |>
plot_response_grna_target_pair(
  response_id = "ENSG00000211641",
  grna_target = "candidate_enh_15"
)

```

plot_run_calibration_check

Plot run calibration check

Description

plot_run_calibration_check() creates a visualization of the outcome of the calibration check. The visualization consists of four panels, which we describe below.

- The upper left panel is a QQ plot of the p-values plotted on an untransformed scale. The p-values ideally should lie along the diagonal line, indicating uniformity of the p-values in the *bulk* of the distribution.
- The upper right panel is a QQ plot of the p-values plotted on a negative log-10 transformed scale. The p-values ideally should lie along the diagonal line (with the majority of the p-values falling within the gray confidence band), indicating uniformity of the p-values in the *tail* of the distribution.
- The lower left panel is a histogram of the estimated log-2 fold changes. The histogram ideally should be roughly symmetric and centered around zero.
- Finally, the bottom right panel is a text box displaying (i) the number of false discoveries that sceptre has made on the negative control data and (ii) the mean estimated log-fold change.

Usage

```

plot_run_calibration_check(
  sceptre_object,
  point_size = 0.55,
  transparency = 0.8,
  return_indiv_plots = FALSE
)

```

Arguments

sceptre_object a sceptre_object that has had run_calibration_check called on it

point_size (optional; default 0.55) the size of the individual points in the plot

transparency (optional; default 0.8) the transparency of the individual points in the plot

return_indiv_plots (optional; default FALSE) if FALSE then a list of ggplot is returned; if TRUE then a single cowplot object is returned.

Value

a single cowplot object containing the combined panels (if return_indiv_plots is set to TRUE) or a list of the individual panels (if return_indiv_plots is set to FALSE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations"
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_calibration_check(
    n_calibration_pairs = 500,
    calibration_group_size = 2
  ) |>
  plot_run_calibration_check()
```

plot_run_discovery_analysis

Plot run discovery analysis

Description

plot_run_discovery_analysis() creates a visualization of the outcome of the discovery analysis. The visualization consists of four plots:

- The upper left plot superimposes the discovery p-values (blue) on top of the negative control p-values (red) on an untransformed scale.

- The upper right plot is the same as the upper left plot, but the scale is negative log-10 transformed. The discovery p-values ideally should trend above the diagonal line, indicating the presence of signal in the discovery set. The horizontal dashed line indicates the multiple testing threshold; discovery pairs whose p-value falls above this line are called as significant.
- The bottom left panel is a volcano plot of the p-values and log fold changes of the discovery pairs. Each point corresponds to a pair; the estimated log-2 fold change of the pair is plotted on the horizontal axis, and the (negative log-10 transformed) p-value is plotted on the vertical axis. The horizontal dashed line again indicates the multiple testing threshold. Points above the dashed line (colored in purple) are called as discoveries, while points below (colored in blue) are called as insignificant.
- The bottom right panel is a text box displaying the number of discovery pairs called as significant.

Usage

```
plot_run_discovery_analysis(
  sceptre_object,
  x_limits = c(-1.5, 1.5),
  point_size = 0.55,
  transparency = 0.8,
  return_indiv_plots = FALSE
)
```

Arguments

sceptre_object a `sceptre_object` that has had `run_discovery_analysis` called on it

x_limits (optional; default `c(-1.5, 1.5)`) a numeric vector of length 2 giving the lower and upper limits of the x-axis (corresponding to log-2 fold change) for the "Discovery volcano plot" panel

point_size (optional; default 0.55) the size of the individual points in the plot

transparency (optional; default 0.8) the transparency of the individual points in the plot

return_indiv_plots (optional; default FALSE) if FALSE then a list of ggplot is returned; if TRUE then a single cowplot object is returned.

Value

a single cowplot object containing the combined panels (if `return_indiv_plots` is set to TRUE) or a list of the individual panels (if `return_indiv_plots` is set to FALSE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
```

```

grna_target_data_frame = grna_target_data_frame_highmoi,
moi = "high",
extra_covariates = highmoi_example_data$extra_covariates,
response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)
sceptre_object |>
  set_analysis_parameters(
    side = "left",
    discovery_pairs = discovery_pairs,
    resampling_mechanism = "permutations",
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_calibration_check() |>
  run_discovery_analysis() |>
  plot_run_discovery_analysis()

```

plot_run_power_check *Plot run power check*

Description

plot_run_power_check() creates a visualization of the outcome of the power check analysis. Each point in the plot corresponds to a target-response pair, with positive control pairs in the left column and negative control pairs in the right column. The vertical axis indicates the p-value of a given pair; smaller (i.e., more significant) p-values are positioned higher along this axis (p-values truncated at clip_to for visualization). The positive control p-values should be small, and in particular, smaller than the negative control p-values.

Usage

```

plot_run_power_check(
  sceptre_object,
  point_size = 1,
  transparency = 0.8,
  clip_to = 1e-20
)

```

Arguments

sceptre_object	a sceptre_object that has had run_power_check() called on it
point_size	(optional; default 1) the size of the individual points in the plot
transparency	(optional; default 0.8) the transparency of the individual points in the plot
clip_to	(optional; default 1e-20) p-values smaller than this value are set to clip_to for better visualization. If clip_to=0 is used then no clipping is done.

Value

a single ggplot2 plot

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
sceptre_object |>
  set_analysis_parameters(
    side = "left",
    positive_control_pairs = positive_control_pairs,
    resampling_mechanism = "permutations",
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_calibration_check(
    n_calibration_pairs = 500,
    calibration_group_size = 2
  ) |>
  run_power_check() |>
  plot_run_power_check()
```

plot_run_qc

Plot run QC

Description

plot_run_qc() creates a visualization of the outcome of the QC step. The top panel depicts the outcome of the cellwise QC. The various cellwise QC filters (e.g., "N nonzero responses," "N response UMIs," "Percent mito", etc.) are shown on the horizontal axis, and the percentage of cells removed due application of a given QC filter is shown on the vertical axis. Note that a cell can be flagged by multiple QC filters; for example, a cell might have an extremely high response_n_umi value *and* an extremely high response_n_nonzero value. Thus, the height of the "any filter" bar (which indicates the percentage of cells removed due to application of *any* filter) need not be equal to the sum of the heights of the other bars. The bottom panel depicts the outcome of the pairwise QC. Each point corresponds to a target-response pair; the vertical axis (resp., horizontal axis) indicates the n_nonzero_trt (resp., n_nonzero_cntrl) value of that pair. Pairs for which n_nonzero_trt or n_nonzero_cntrl fall below the threshold are removed (red), while the remaining pairs are retained (green).

Usage

```
plot_run_qc(
  sceptre_object,
  downsample_pairs = 10000L,
  point_size = 0.55,
  transparency = 0.8,
  return_indiv_plots = FALSE
)
```

Arguments

sceptre_object a sceptre_object that has had run_qc() called on it

downsample_pairs (optional; default 10000) the maximum number of points to plot in the lower panel of the figure (i.e., the "pairwise QC" plot)

point_size (optional; default 0.55) the size of the individual points in the plot

transparency (optional; default 0.8) the transparency of the individual points in the plot

return_indiv_plots (optional; default FALSE) if FALSE then a list of ggplot is returned; if TRUE then a single cowplot object is returned.

Value

a single cowplot object containing the combined panels (if return_indiv_plots is set to TRUE) or a list of the individual panels (if return_indiv_plots is set to FALSE)

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
discovery_pairs <- construct_cis_pairs(sceptre_object)
sceptre_object |>
  set_analysis_parameters(
    discovery_pairs = discovery_pairs,
    side = "left"
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  plot_run_qc()
```

`print,sceptre_object-method`*Print*

Description

`print()` prints information about the dataset and the status of the analysis to the console. The output contains several fields: Attributes of the data (summarizing key features of the data), Analysis status (indicating the analysis functions that have been called), Analysis parameters (summarizing the analysis parameters set in `set_analysis_parameters()`), gRNA-to-cell assignment information (summarizing the outcome of the gRNA-to-cell assignment step), and Summary of results (summarizing the key analysis results). A subset of these fields may be printed, depending on the status of the analysis.

Usage

```
## S4 method for signature 'sceptre_object'
print(x)
```

Arguments

`x` a `sceptre_object`

Value

the value `NULL`

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
print(sceptre_object)
```

read_ondisc_backed_sceptre_object

Write or read an ondisc-backed sceptre_object

Description

write_ondisc_backed_sceptre_object() and read_ondisc_backed_sceptre_object() enable the writing and reading of ondisc-backed sceptre_objects, respectively. First, write_ondisc_backed_sceptre_object() writes an ondisc-backed sceptre_object to disk, creating a file sceptre_object.rds in the specified directory. Next, read_ondisc_backed_sceptre_object() reads and initializes a sceptre_object from a sceptre_object.rds file, response.odm file, and grna.odm file stored on disk.

Usage

```
read_ondisc_backed_sceptre_object(
  sceptre_object_fp,
  response_odm_file_fp,
  grna_odm_file_fp
)
```

```
write_ondisc_backed_sceptre_object(sceptre_object, directory_to_write)
```

Arguments

```
sceptre_object_fp
    file path to a sceptre_object.rds file
response_odm_file_fp
    file path to the backing .odm file for the response modality
grna_odm_file_fp
    file path to the backing .odm file for the gRNA modality
sceptre_object  a sceptre_object
directory_to_write
    the directory in which to write the sceptre_object.rds file
```

Value

write_ondisc_backed_sceptre_object() returns NULL, and read_ondisc_backed_sceptre_object() returns an ondisc-backed sceptre_object

Examples

```
data(lowmoi_example_data)
# 1. create ondisc-backed sceptre_object
sceptre_object <- import_data(
  response_matrix = lowmoi_example_data$response_matrix,
  grna_matrix = lowmoi_example_data$grna_matrix,
  grna_target_data_frame = lowmoi_example_data$grna_target_data_frame,
```

```

    extra_covariates = lowmoi_example_data$extra_covariates,
    moi = "low",
    use_ondisc = TRUE,
    directory_to_write = tempdir()
  )

  # 2. write
  write_ondisc_backed_sceptre_object(
    sceptre_object = sceptre_object,
    directory_to_write = tempdir()
  )

  # 3. read
  rm(sceptre_object)
  sceptre_object <- read_ondisc_backed_sceptre_object(
    sceptre_object_fp = paste0(tempdir(), "/sceptre_object.rds"),
    response_odm_file_fp = paste0(tempdir(), "/response.odm"),
    grna_odm_file_fp = paste0(tempdir(), "/grna.odm")
  )

```

run_calibration_check *Run calibration check*

Description

run_calibration_check() runs the calibration check. The calibration check involves applying sceptre to analyze negative control target-response pairs — pairs for which we know there is no association between the target and response — to ensure control of the false discovery rate. The calibration check enables us to verify that the discovery set that sceptre ultimately produces is not contaminated by excess false positives. See [Chapter 5 of the manual](#) for more detailed information about this function.

Usage

```

run_calibration_check(
  sceptre_object,
  n_calibration_pairs = NULL,
  calibration_group_size = NULL,
  print_progress = TRUE,
  parallel = FALSE,
  n_processors = "auto",
  log_dir = tempdir(),
  output_amount = 1
)

```

Arguments

sceptre_object a sceptre_object

`n_calibration_pairs` (optional) the number of negative control pairs to construct and test for association

`calibration_group_size` (optional) the number of negative control gRNAs to randomly assemble to form each negative control target

`print_progress` (optional; default TRUE) a logical indicating whether to print progress updates

`parallel` (optional; default FALSE) a logical indicating whether to run the function in parallel. `parallel = TRUE` is recommended only on Mac; it is not supported on Windows and may behave unreliably on Linux clusters.

`n_processors` (optional; default "auto") an integer specifying the number of processors to use if `parallel` is set to TRUE. The default, "auto", uses half the physical cores. The fraction may be tuned via the `parallelly.availableCores.fraction` R option.

`log_dir` (optional; default `tempdir()`) a string indicating the directory in which to write the log files (ignored if `parallel = FALSE`)

`output_amount` (optional; default 1) an integer taking values 1, 2, or 3 specifying the amount of information to return. 1 returns the least amount of information and 3 the most.

Value

an updated `sceptre_object` in which the calibration check has been carried out

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)

# set analysis parameters, assign grnas, run qc, run calibration check
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations"
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_calibration_check(
    n_calibration_pairs = 500,
    calibration_group_size = 2
  )
```

run_discovery_analysis

Run discovery analysis

Description

run_discovery_analysis() runs the discovery analysis. The discovery analysis involves applying sceptre to analyze discovery pairs, or target-response pairs whose association status we do not know but seek to learn. Identifying associations among the discovery pairs is the primary objective of the single-cell CRISPR screen analysis. See [Chapter 6 of the manual](#) for more detailed information about this function.

Usage

```
run_discovery_analysis(
  sceptre_object,
  output_amount = 1,
  print_progress = TRUE,
  parallel = FALSE,
  n_processors = "auto",
  log_dir = tempdir()
)
```

Arguments

sceptre_object	a sceptre_object
output_amount	(optional; default 1) an integer taking values 1, 2, or 3 specifying the amount of information to return. 1 returns the least amount of information and 3 the most.
print_progress	(optional; default TRUE) a logical indicating whether to print progress updates
parallel	(optional; default FALSE) a logical indicating whether to run the function in parallel. parallel = TRUE is recommended only on Mac; it is not supported on Windows and may behave unreliably on Linux clusters.
n_processors	(optional; default "auto") an integer specifying the number of processors to use if parallel is set to TRUE. The default, "auto", uses half the physical cores. The fraction may be tuned via the parallelly.availableCores.fraction R option.
log_dir	(optional; default tempdir()) a string indicating the directory in which to write the log files (ignored if parallel = FALSE)

Value

an updated sceptre_object in which the discovery analysis has been carried out

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
# set analysis parameters, assign grnas, run qc
discovery_pairs <- construct_cis_pairs(sceptre_object)
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations",
    discovery_pairs = discovery_pairs
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_discovery_analysis()
```

run_power_check	<i>Run power check</i>
-----------------	------------------------

Description

`run_power_check()` runs the power check. The power check entails applying `sceptre` to analyze positive control pairs — pairs for which we know there is an association between the target and the response — to ensure that `sceptre` is capable of detecting true associations. See [Chapter 6 of the manual](#) for more detailed information about this function.

Usage

```
run_power_check(
  sceptre_object,
  output_amount = 1,
  print_progress = TRUE,
  parallel = FALSE,
  n_processors = "auto",
  log_dir = tempdir()
)
```

Arguments

`sceptre_object` a `sceptre_object`

output_amount (optional; default 1) an integer taking values 1, 2, or 3 specifying the amount of information to return. 1 returns the least amount of information and 3 the most.

print_progress (optional; default TRUE) a logical indicating whether to print progress updates

parallel (optional; default FALSE) a logical indicating whether to run the function in parallel. `parallel = TRUE` is recommended only on Mac; it is not supported on Windows and may behave unreliably on Linux clusters.

n_processors (optional; default "auto") an integer specifying the number of processors to use if `parallel` is set to `TRUE`. The default, "auto", uses half the physical cores. The fraction may be tuned via the `parallelly.availableCores.fraction` R option.

log_dir (optional; default `tempdir()`) a string indicating the directory in which to write the log files (ignored if `parallel = FALSE`)

Value

an updated `sceptre_object` in which the power check has been carried out

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)

# set analysis parameters, assign grnas, run qc
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations",
    positive_control_pairs = positive_control_pairs
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_power_check()
```

Description

run_qc() runs cellwise and pairwise QC on the data. Cellwise QC involves filtering cells on the covariates response_n_nonzero, response_n_umis, and response_p_mito. In low-MOI we additionally remove cells that contain zero or multiple gRNAs. Next, pairwise QC involves filtering out target-response pairs whose data are too sparse to be analyzed reliably. In this context we define the “number of nonzero treatment cells” (resp., the “number of nonzero control cells”) as the number of cells in the treatment group (resp., control group) that contain nonzero expression of the response. (We sometimes use the shorthand n_nonzero_trt and n_nonzero_cntrl to refer to the number of nonzero treatment cells and control cells, respectively.) Pairwise QC involves filtering target-response pairs on n_nonzero_trt and n_nonzero_cntrl. See [Chapter 4 of the manual](#) for more detailed information about this function.

Usage

```
run_qc(
  sceptre_object,
  n_nonzero_trt_thresh = 7L,
  n_nonzero_cntrl_thresh = 7L,
  response_n_umis_range = c(0.01, 0.99),
  response_n_nonzero_range = c(0.01, 0.99),
  p_mito_threshold = 0.2,
  additional_cells_to_remove = integer()
)
```

Arguments

sceptre_object a sceptre_object

n_nonzero_trt_thresh
(optional; default 7L) an integer specifying the number of nonzero *treatment* cells a pair must contain for it to be retained

n_nonzero_cntrl_thresh
(optional; default 7L) an integer specifying the number of nonzero *control* cells a pair must contain for it to be retained

response_n_umis_range
(optional; default c(0.01, 0.99)) a length-two vector of percentiles specifying the location at which to clip the left and right tails of the response_n_umis distribution

response_n_nonzero_range
(optional; default c(0.01, 0.99)) a length-two vector of percentiles specifying the location at which to clip the left and right tails of the response_n_nonzero distribution

p_mito_threshold
(optional; default 0.2) a numeric value specifying the location at which to clip the right tail of the response_p_mito distribution

additional_cells_to_remove
(optional) a vector of integer indices specifying additional cells to remove

Value

an updated sceptre_object in which cellwise and pairwise QC have been applied

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)

# set analysis parameters, assign grnas, run qc
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    discovery_pairs = discovery_pairs,
    positive_control_pairs = positive_control_pairs,
    side = "left"
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc()
```

set_analysis_parameters

Set analysis parameters

Description

set_analysis_parameters() sets the analysis parameters that control how the statistical analysis is to be conducted. See [Chapter 2 of the manual](#) for more detailed information about this function.

Usage

```
set_analysis_parameters(
  sceptre_object,
  discovery_pairs = data.frame(grna_target = character(0), response_id = character(0)),
  positive_control_pairs = data.frame(grna_target = character(0), response_id =
    character(0)),
  side = "both",
```

```

    grna_integration_strategy = "union",
    formula_object = "default",
    resampling_approximation = "skew_normal",
    control_group = "default",
    resampling_mechanism = "default",
    multiple_testing_method = "BH",
    multiple_testing_alpha = 0.1
)

```

Arguments

sceptre_object a `sceptre_object`

discovery_pairs
(optional) a data frame with columns `grna_target` and `response_id` specifying the discovery pairs to analyze

positive_control_pairs
(optional) a data frame with columns `grna_target` and `response_id` specifying the positive control pairs to analyze

side
(optional; default "both") the sidedness of the test, one of "left", "right", or "both"

grna_integration_strategy
(optional; default "union") a string specifying the gRNA integration strategy, either "singleton", "union", or "bonferroni"

formula_object (optional) a formula object specifying how to adjust for the covariates in the model

resampling_approximation
(optional; default "skew_normal") a string indicating the resampling approximation to make to the null distribution of test statistics, either "skew_normal" or "no_approximation"

control_group (optional) a string specifying the control group to use in the differential expression analysis, either "complement" or "nt_cells"

resampling_mechanism
(optional) a string specifying the resampling mechanism to use, either "permutations" or "crt"

multiple_testing_method
(optional; default "BH") a string specifying the multiple testing correction method to use; see `p.adjust.methods` for options

multiple_testing_alpha
(optional; default 0.1) a numeric specifying the nominal level of the multiple testing correction method

Value

an updated `sceptre_object` in which the analysis parameters have been set

Note

Every argument to this function is optional, but typically, users want to specify `discovery_pairs` at minimum.

Examples

```

data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)

# set analysis parameters
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)
sceptre_object <- sceptre_object |>
  set_analysis_parameters(
    discovery_pairs = discovery_pairs,
    positive_control_pairs = positive_control_pairs,
    side = "left"
  )

```

```
write_outputs_to_directory
```

Write outputs to directory

Description

`write_outputs_to_directory()` writes the outputs of a `sceptre` analysis to a directory on disk. `write_outputs_to_directory()` writes several files to the specified directory: a text-based summary of the analysis (`analysis_summary.txt`), the various plots (`*.png`), the calibration check, power check, discovery analysis results (`results_run_calibration_check.rds`, `results_run_power_check.rds`, and `results_run_discovery_analysis.rds`, respectively), and the binary gRNA-to-cell assignment matrix (`grna_assignment_matrix.rds`). See [Section 8 of the introductory chapter in the manual](#) for more information about this function.

Usage

```
write_outputs_to_directory(sceptre_object, directory)
```

Arguments

`sceptre_object` a `sceptre_object`

`directory` a string giving the file path to a directory on disk in which to write the results

Value

No return value; called for its side effect of writing files to directory.

Examples

```
data(highmoi_example_data)
data(grna_target_data_frame_highmoi)
# import data
sceptre_object <- import_data(
  response_matrix = highmoi_example_data$response_matrix,
  grna_matrix = highmoi_example_data$grna_matrix,
  grna_target_data_frame = grna_target_data_frame_highmoi,
  moi = "high",
  extra_covariates = highmoi_example_data$extra_covariates,
  response_names = highmoi_example_data$gene_names
)
positive_control_pairs <- construct_positive_control_pairs(sceptre_object)
discovery_pairs <- construct_cis_pairs(sceptre_object,
  positive_control_pairs = positive_control_pairs,
  distance_threshold = 5e6
)
sceptre_object |>
  set_analysis_parameters(
    side = "left",
    resampling_mechanism = "permutations",
    discovery_pairs = discovery_pairs,
    positive_control_pairs = positive_control_pairs
  ) |>
  assign_grnas(method = "thresholding") |>
  run_qc() |>
  run_calibration_check() |>
  run_power_check() |>
  run_discovery_analysis() |>
  write_outputs_to_directory(paste0(tempdir(), "/sceptre_outputs"))
# files written to "sceptre_outputs" in tempdir()
```

Index

- * **datasets**
 - gene_position_data_frame_grch38,
12
 - grna_target_data_frame_highmoi, 16
 - highmoi_example_data, 17
 - lowmoi_example_data, 23
- * **package**
 - sceptre-package, 3
- assign_grnas, 6
- construct_cis_pairs, 8
- construct_positive_control_pairs, 9
- construct_trans_pairs, 10
- gene_position_data_frame_grch37
(gene_position_data_frame_grch38),
12
- gene_position_data_frame_grch38, 12
- get_cell_covariates
(get_response_matrix), 14
- get_grna_assignments, 13
- get_grna_matrix (get_response_matrix),
14
- get_response_matrix, 14
- get_result, 15
- grna_target_data_frame_highmoi, 16
- highmoi_example_data, 17
- import_data, 17
- import_data_from_cellranger, 19
- import_data_from_parse, 21
- lowmoi_example_data, 23
- plot, sceptre_object-method, 23
- plot_assign_grnas, 24
- plot_covariates, 26
- plot_grna_count_distributions, 27
- plot_response_grna_target_pair, 29
- plot_run_calibration_check, 30
- plot_run_discovery_analysis, 31
- plot_run_power_check, 33
- plot_run_qc, 34
- print, sceptre_object-method, 36
- read_ondisc_backed_sceptre_object, 37
- run_calibration_check, 38
- run_discovery_analysis, 40
- run_power_check, 41
- run_qc, 42
- sceptre (sceptre-package), 3
- sceptre-package, 3
- set_analysis_parameters, 44
- write_ondisc_backed_sceptre_object
(read_ondisc_backed_sceptre_object),
37
- write_outputs_to_directory, 46