Package: GIMP (via r-universe)

October 8, 2026

Title Genomic Imprinting Methylation Patterns

Version 0.2.0

Description A package for analyzing Imprinting Control Regions (ICRs) DNA methylation. Supports both processed methylation data and raw IDAT files from Illumina arrays. Provides specialized tools for imprinting analysis including defect detection and interactive visualizations.

License GPL-3

Encoding UTF-8

URL https://github.com/SAADAT-Abu/GIMP

BugReports https://github.com/SAADAT-Abu/GIMP/issues

LazyData false

LazyDataCompression xz

RoxygenNote 7.3.3

Depends R (>= 4.4.0)

Imports shiny, shinydashboard, DT, plotly, minfi, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b4.hg19, IlluminaHumanMethylationEPICv2anno.20a1.hg38, IlluminaHumanMethylationEPICmanifest, IlluminaHumanMethylation450kmanifest, IlluminaHumanMethylationEPICv2manifest, tools, utils, stats, valr, reshape2, ggplot2, pheatmap, viridisLite, ggplotify, readr, BiocManager, R.utils, Biobase, GEOquery, limma, tibble, dplyr, tidyr, BiocParallel, readxl, doParallel, BiocGenerics

Suggests knitr, rmarkdown, testthat, AnnotationHub

VignetteBuilder knitr

biocViews DNAMethylation, Microarray, Preprocessing, QualityControl, DifferentialMethylation, Epigenetics, MethylationArray

NeedsCompilation no

Config/pak/sysreqs cmake make libbz2-dev libicu-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://saadat-abu.r-universe.dev

Date/Publication 2026-09-08 19:02:35 UTC

RemoteUrl https://github.com/SAADAT-Abu/GIMP

RemoteRef HEAD

RemoteSha b3c382574a5b27503bfa138a2033eb511ab9d903

Contents

bed450k . . . 2
bedEPICv1 . . . 3
bedEPICv2 . . . 4
calculate_detection_pvalues . . . 5
check_minfi_functions . . . 5
create_bedmeth . . . 6
create_sample_sheet_template . . . 7
diagnose_geo_dataset . . . 8
DMRs.hg19 . . . 9
DMRs.hg38 . . . 9
get_geo_phenotype_data . . . 10
GIMP_app . . . 11
ICRs_heatmap . . . 12
iDMPs . . . 13
make_cpgs . . . 15
make_ICRs . . . 16
plot_cpgs_coverage . . . 17
plot_line_ICR . . . 18
preview_idat_zip . . . 20
process_geo_dataset . . . 20
process_geo_with_mappings . . . 22
read_idat_zip . . . 23
validate_geo_dataset . . . 25

Index **26**

`bed450k` *BED 450K probes*

Description

This dataset contains the 450K array probes coordinates. Data is now hosted on AnnotationHub and can be accessed using the internal helper function get_bed_data("450k").

Usage

```
data(bed450k)
```

Format

BED file with chromosome, start, end, and probeID columns.

Note

For Bioconductor submissions, this data is available via AnnotationHub. Use functions like make_cpgs(), make_ICRs(), and plot_cpgs_coverage() which automatically retrieve the data from AnnotationHub when needed.

Examples

```
## Not run:
# Data is now accessed via AnnotationHub
# Use GIMP functions directly - they handle data loading:
# ICRcpg <- make_cpgs(Bmatrix = your_beta_matrix, bedmeth = "450k")

## End(Not run)
```

---

bedEPICv1 *BED EPICv1 probes*

Description

This dataset contains the EPICv1 array probes coordinates. Data is now hosted on AnnotationHub and can be accessed using the internal helper function get_bed_data("v1").

Usage

```
data(bedEPICv1)
```

Format

BED file with chromosome, start, end, and probeID columns.

Note

For Bioconductor submissions, this data is available via AnnotationHub. Use functions like make_cpgs(), make_ICRs(), and plot_cpgs_coverage() which automatically retrieve the data from AnnotationHub when needed.

Examples

```
## Not run:
# Data is now accessed via AnnotationHub
# Use GIMP functions directly - they handle data loading:
# ICRcpg <- make_cpgs(Bmatrix = your_beta_matrix, bedmeth = "v1")

## End(Not run)
```

---

bedEPICv2 *BED EPICv2 probes*

Description

This dataset contains the EPICv2 array probes coordinates. Data is now hosted on AnnotationHub and can be accessed using the internal helper function get_bed_data("v2").

Usage

```
data(bedEPICv2)
```

Format

BED file with chromosome, start, end, and probeID columns.

Note

For Bioconductor submissions, this data is available via AnnotationHub. Use functions like make_cpgs(), make_ICRs(), and plot_cpgs_coverage() which automatically retrieve the data from AnnotationHub when needed.

Examples

```
## Not run:
# Data is now accessed via AnnotationHub
# Use GIMP functions directly - they handle data loading:
# ICRcpg <- make_cpgs(Bmatrix = your_beta_matrix, bedmeth = "v2")

## End(Not run)
```

calculate_detection_pvalues *Alternative Detection P-value Calculation*

Description

Alternative method for calculating detection p-values when standard minfi functions are not available.

Usage

```
calculate_detection_pvalues(rgSet)
```

Arguments

rgSet	RGChannelSet object from minfi

Value

Matrix of detection p-values or NULL if calculation fails

Examples

```
# This function is designed to work with RGChannelSet objects from minfi
# Check if function is available
if (exists("calculate_detection_pvalues")) {
  print("Detection p-value calculation function is available")
}


# This function requires actual IDAT data
# rgSet <- read.metharray.exp("path/to/idat/files")
# det_p <- calculate_detection_pvalues(rgSet)
```

check_minfi_functions *Check Available minfi Functions*

Description

Diagnostic function to check which minfi functions are available in the current installation.

Usage

```
check_minfi_functions()
```

Value

Character vector of all available minfi functions

Examples

```
# Check if function exists
if (exists("check_minfi_functions")) {
  print("Diagnostic function is available")
}


# Check which minfi functions are available
available_funcs <- check_minfi_functions()
```

`create_bedmeth` *Create BED File Data from Methylation Array Annotations*

Description

This function generates a BED-format data frame from Illumina Human Methylation annotation files. The BED data includes chromosome, position, and probe ID information, and supports multiple annotation versions.

Usage

```
create_bedmeth(version = "v1")
```

Arguments

`version`	A character string specifying the annotation version to use. Options include ‘"450k"‘ for 450k array, ‘"v1"‘ for the EPIC version1 and ‘"v2"‘ for EPIC version2. Default is ‘"v1"‘.

Value

A data frame in BED format containing columns:

`chr`	Chromosome name.
`pos`	Position on the chromosome.
`probeID`	Unique identifier for each probe.
`end`	End position, which is the same as ‘pos‘ in this output.

Examples

```
# Example showing the structure of the function call
# Note: Requires minfi annotation packages to be installed
# bed_data <- create_bedmeth(version = "v1")
# head(bed_data)

# Check available versions
available_versions <- c("v1", "v2", "450k")
print(available_versions)

## Not run:
# Create BED-format data with the default version (EPIC v1)
bed_data <- create_bedmeth()
head(bed_data) # View the first few rows

# Use a different annotation version if available
bed_data_v2 <- create_bedmeth(version = "v2")

## End(Not run)
```

---

create_sample_sheet_template

Create Sample Sheet Template

Description

Creates a template sample sheet for IDAT files to help users format their data correctly.

Usage

```
create_sample_sheet_template(
  sample_names = NULL,
  sentrix_ids = NULL,
  sentrix_positions = NULL,
  groups = NULL
)
```

Arguments

sample_names	Vector of sample names
sentrix_ids	Vector of Sentrix IDs (slide IDs)
sentrix_positions	Vector of Sentrix positions
groups	Optional vector of sample groups

Value

Data frame with sample sheet structure

Examples

```
# Create a basic template
template <- create_sample_sheet_template(
    sample_names = c("Sample1", "Sample2", "Sample3"),
    sentrix_ids = c("200123456", "200123456", "200123457"),
    sentrix_positions = c("R01C01", "R02C01", "R01C01"),
    groups = c("Control", "Control", "Case")
)
print(template)

# Create template with default values (minimal input)
simple_template <- create_sample_sheet_template(
    sample_names = c("Ctrl_01", "Case_01")
)
head(simple_template)
```

---

diagnose_geo_dataset *Diagnose GEO dataset structure*

Description

Downloads and examines a GEO dataset to understand its file structure without processing the data. Useful for troubleshooting.

Usage

```
diagnose_geo_dataset(geo_id, temp_dir = NULL)
```

Arguments

geo_id	Character string of GEO accession (e.g., "GSE12345")
temp_dir	Temporary directory for download (default: temporary directory)

Value

List with diagnostic information

Examples

```
# Example showing diagnostic function usage
# Note: Requires internet connection and GEOquery package
# diag <- diagnose_geo_dataset("GSE289527")
# print(diag$summary)

# Check if function is available
if (exists("diagnose_geo_dataset")) {
  print("GEO diagnostic function is available")
}
```

```
# Diagnose a problematic dataset
diag <- diagnose_geo_dataset("GSE289527")
print(diag$summary)
```

DMRs.hg19 *Imprinted Regions*

Description

This dataset contains the Human Imprinted regions coordinates in hg19.

Usage

```
data(DMRs.hg19)
```

Format

A data frame with iDMRs coordinates.

Examples

```
## Not run:
data(DMRs.hg19)
head(DMRs.hg19)

## End(Not run)
```

DMRs.hg38 *Imprinted Regions*

Description

This dataset contains the Human Imprinted regions coordinates in hg38.

Usage

```
data(DMRs.hg38)
```

Format

A data frame with iDMRs coordinates.

Examples

```
## Not run:
data(DMRs.hg38)
head(DMRs.hg38)

## End(Not run)
```

`get_geo_phenotype_data` *Get phenotypic data preview for GEO dataset*

Description

Downloads and returns phenotypic data from a GEO dataset for user review before processing. This allows users to select appropriate grouping columns.

Usage

```
get_geo_phenotype_data(geo_id)
```

Arguments

`geo_id`	Character string of GEO accession (e.g., "GSE12345")

Value

List containing phenotypic data and metadata

Examples

```
# Example usage structure
# Note: Requires internet connection and GEOquery package
# pheno_preview <- get_geo_phenotype_data("GSE68777")
# head(pheno_preview$pheno_data)

# Check if function is available
if (exists("get_geo_phenotype_data")) {
  print("GEO phenotype data function is available")
}


# Get phenotypic data for review
pheno_preview <- get_geo_phenotype_data("GSE68777")
head(pheno_preview$pheno_data)
```

GIMP_app *Launch GIMP Shiny Application*

Description

Launches an interactive Shiny application for GIMP analysis. The app provides a graphical user interface for analyzing methylation patterns at Imprinted Control Regions (ICRs) without requiring R programming knowledge.

Usage

```
GIMP_app(max_upload_size_mb = 500)
```

Arguments

max_upload_size_mb	Maximum file upload size in MB (default: 500)

Details

The GIMP Shiny app includes the following features:

- Upload methylation beta matrices from CSV files or raw IDAT files
- Analyze CpG coverage at ICRs
- Generate methylation heatmaps (beta, delta, and defect plots)
- Identify differentially methylated positions (DMPs)
- Explore specific ICR regions with interactive visualizations
- Export results and plots

Value

Opens the Shiny application in the default web browser. The function returns invisibly once the app is closed.

Examples

```
# Example showing how to launch the app
# GIMP_app()

# Check if function is available
if (exists("GIMP_app")) {
  print("GIMP Shiny app launcher is available")
}


# Launch the GIMP Shiny app
GIMP_app()
```

```
# Launch with larger upload limit
GIMP_app(max_upload_size_mb = 1000)
```

`ICRs_heatmap` *Generate Heatmap of ICRs Methylation*

Description

This function generates a heatmap for visualizing methylation data of ICRs.

Usage

```
ICRs_heatmap(
  df_ICR,
  sampleInfo,
  control_label = "Control",
  case_label = "Case",
  bedmeth = "v1",
  order_by = "cord",
  annotation_col = NULL,
  plot_type = "beta",
  sd_threshold = 3
)
```

Arguments

`df_ICR`	A data frame or matrix containing methylation beta values for ICRs.
`sampleInfo`	A vector indicating the group labels (e.g., "Control" and "Case") for each sample in ‘df_ICR‘. Each element in ‘sampleInfo‘ should correspond to a sample in ‘df_ICR‘.
`control_label`	A character string specifying the label for the control group in ‘sampleInfo‘. Default is ‘"Control"‘.
`case_label`	A character string specifying the label for the case group in ‘sampleInfo‘. Default is ‘"Case"‘.
`bedmeth`	A character string specifying the BED data version for DMR coordinates. Options are ‘"v1"‘, ‘"v2"‘, or ‘"450k"‘. Default is ‘"v1"‘.
`order_by`	A character string specifying the ordering rows in the heatmap. Options are ‘"cord"‘ for coordinates or ‘"meth"‘ for methylation values. Default is ‘"cord"‘.
`annotation_col`	A named list of colors for each unique value in ‘sampleInfo‘. If ‘NULL‘, default colors are assigned using the "viridis" palette. Default is ‘NULL‘.
`plot_type`	A character string specifying the type of heatmap to generate. Options are ‘"beta"‘ for beta values, ‘"delta"‘ for values normalized against controls, and ‘"defect"‘ for defect matrix based on standard deviations. Default is ‘"beta"‘.

sd_threshold A numeric value specifying the standard deviation threshold for detecting defects in the defect matrix. Only used if ‘plot_type‘ is ‘"defect"‘. Default is ‘3‘.

Value

A heatmap plot visualizing methylation of ICRs.

Examples

```
# Create synthetic ICR matrix for demonstration
set.seed(42)
n_icrs <- 10
n_samples <- 8

# Generate random ICR names and methylation data
icr_names <- paste0("ICR_", 1:n_icrs)
df_ICR_example <- matrix(runif(n_icrs * n_samples, 0.2, 0.8),
                         nrow = n_icrs, ncol = n_samples)
rownames(df_ICR_example) <- icr_names
colnames(df_ICR_example) <- paste0("Sample_", 1:n_samples)
df_ICR_example <- as.data.frame(df_ICR_example)

# Create sample info
sampleInfo_example <- c(rep("Control", 4), rep("Case", 4))

# Note: This uses synthetic data. In practice, use make_ICRs() output
# heatmap <- ICRs_heatmap(df_ICR = df_ICR_example,
#                         sampleInfo = sampleInfo_example,
#                         plot_type = "beta")

## Not run:
# Example sampleInfo with "Case" and "Control" labels for each sample
sampleInfo <- c(rep("Case", 10), rep("Control", 10))
ICRs_heatmap(
    df_ICR = my_ICR_data, sampleInfo = sampleInfo,
    annotation_col = list(Sample = c("darkgreen", "darkred"))
)

## End(Not run)
```

---

iDMPs *Identify Differentially Methylated Positions in ICRs*

Description

This function identifies differentially methylated positions (DMPs) between control and case groups using linear modeling and empirical Bayes methods.

Usage

```
iDMPs(data, sampleInfo, pValueCutoff = 0.05)
```

Arguments

data	A data frame containing CpG methylation data with annotation columns
sampleInfo	A factor or character vector indicating sample groups
pValueCutoff	P-value threshold for significance (default: 0.05)

Value

A list containing:

fit	Linear model fit object
eBayesfit	Empirical Bayes fit object
topDMPs	Data frame of significant DMPs
allResults	Data frame of all results
groupLabels	Group labels used in analysis

Examples

```
# Create synthetic methylation data with annotation
set.seed(123)
n_cpgs <- 50
n_controls <- 3
n_cases <- 3

# Generate synthetic data
meth_data <- matrix(runif(n_cpgs * (n_controls + n_cases), 0.3, 0.7),
                    nrow = n_cpgs, ncol = n_controls + n_cases)
colnames(meth_data) <- c(paste0("Ctrl_", 1:n_controls),
                         paste0("Case_", 1:n_cases))

# Add annotation columns
ICRcpg_example <- data.frame(
  meth_data,
  cstart = seq(1000000, by = 100, length.out = n_cpgs),
  ICR = rep(c("ICR1", "ICR2"), length.out = n_cpgs),
  start = 1000000,
  end = 1005000
)

# Create sample info
sampleInfo_example <- factor(c(rep("Control", n_controls),
                               rep("Case", n_cases)))

# Note: This uses synthetic data for demonstration
# dmps <- iDMPs(data = ICRcpg_example,
#               sampleInfo = sampleInfo_example,
#               pValueCutoff = 0.05)

## Not run:
# Run DMP analysis
dmps <- iDMPs(data = ICRcpg, sampleInfo = sample_groups)
```

```
significant_dmps <- dmps$topDMPs

## End(Not run)
```

---

make_cpgs *Create ICR CpG Matrix*

Description

This function generates a CpG matrix for Imprinted Control Regions (ICR) using methylation data. The CpG matrix is constructed based on the provided BED data version.

Usage

```
make_cpgs(Bmatrix, bedmeth = "v1")
```

Arguments

Bmatrix	A data frame or matrix containing methylation beta values. Rows typically represent individual probes or CpGs, and columns represent samples.
bedmeth	A character string specifying the BED data version to use for CpG mapping. Options are ‘"v1"‘ (EPIC v1), ‘"v2"‘ (EPIC v2), or ‘"450k"‘ (450k array). Default is ‘"v1"‘.

Value

A data frame representing the ICR CpG matrix, with rows as CpG probes and columns as samples.

Examples

```
# Create a simple example beta matrix
set.seed(123)
beta_matrix <- matrix(runif(100 * 4, 0, 1), nrow = 100, ncol = 4)
rownames(beta_matrix) <- paste0("cg", sprintf("%08d", 1:100))
colnames(beta_matrix) <- paste0("Sample_", 1:4)

# Extract ICR CpGs (will be empty if no probes overlap with ICRs)
ICRcpg <- make_cpgs(Bmatrix = beta_matrix, bedmeth = "v1")


# Create sample beta matrix for demonstration
set.seed(123)
n_probes <- 1000
n_samples <- 6

# Generate random probe IDs that might overlap with ICRs
sample_probes <- paste0("cg", sprintf("%08d", sample(1:50000000, n_probes)))
beta_matrix <- matrix(runif(n_probes * n_samples, 0.3, 0.8),
    nrow = n_probes, ncol = n_samples
```

```
)
rownames(beta_matrix) <- sample_probes
colnames(beta_matrix) <- paste0("Sample_", 1:n_samples)

# Generate the ICR CpG matrix with default BED version (EPIC v1)
ICRcpg <- make_cpgs(Bmatrix = beta_matrix, bedmeth = "v1")

# Use a different BED version, such as EPIC v2
ICRcpg_v2 <- make_cpgs(Bmatrix = beta_matrix, bedmeth = "v2")


# Simple usage with your own data:
# ICRcpg <- make_cpgs(Bmatrix = your_beta_matrix, bedmeth = "v1")
```

`make_ICRs` *Create the ICR Matrix*

Description

This function generates an ICR (Imprinted Control Region) matrix from a given beta matrix, using specified BED data for CpG mapping. The ICR matrix provides data organized by CpG probes and samples. The coordinates of the Human Imprinted regions are taken from https://doi.org/10.1080/15592294.2016.1264561

Usage

```
make_ICRs(Bmatrix, bedmeth = "v1")
```

Arguments

`Bmatrix`	A data frame or matrix containing methylation beta values. Rows should represent CpG probes, and columns represent samples.
`bedmeth`	A character string indicating the BED data version to use for CpG mapping. Options are ‘"v1"‘ (EPIC v1), ‘"v2"‘ (EPIC v2), or ‘"450k"‘ (450k array). Default is ‘"v1"‘.

Value

A data frame representing the ICR matrix, structured by CpG probes and samples.

Examples

```
# Create a simple synthetic beta matrix
set.seed(123)
beta_mat <- matrix(runif(100 * 4, 0.3, 0.7), nrow = 100, ncol = 4)
rownames(beta_mat) <- paste0("cg", sprintf("%08d", 1:100))
colnames(beta_mat) <- paste0("Sample_", 1:4)

# Note: This example uses synthetic data for demonstration.
```

```
# In practice, use real methylation data from your arrays.
# ICRmatrix <- make_ICRs(Bmatrix = beta_mat, bedmeth = "v1")


# Create sample beta matrix for demonstration
set.seed(123)
n_probes <- 1000
n_samples <- 6

# Generate random probe IDs that might overlap with ICRs
sample_probes <- paste0("cg", sprintf("%08d", sample(1:50000000, n_probes)))
beta_matrix <- matrix(runif(n_probes * n_samples, 0.3, 0.8),
    nrow = n_probes, ncol = n_samples
)
rownames(beta_matrix) <- sample_probes
colnames(beta_matrix) <- paste0("Sample_", 1:n_samples)

# Generate the ICR matrix with default BED version (EPIC v1)
ICRmatrix <- make_ICRs(Bmatrix = beta_matrix, bedmeth = "v1")


# Simple usage with your own data:
# ICRmatrix <- make_ICRs(Bmatrix = your_beta_matrix, bedmeth = "v1")
```

`plot_cpgs_coverage` *Plot ICR CpG Matrix with Counts and Percentage Coverage*

Description

This function plots the CpG coverage for Imprinted Control Regions (ICRs) using the provided data frame of CpG counts. It compares CpG counts in the specified BED data version for visual analysis and includes an additional plot for percentage coverage.

Usage

```
plot_cpgs_coverage(df_ICR_cpg, bedmeth = "v1")
```

Arguments

`df_ICR_cpg`	A data frame containing CpG counts for ICR regions. Each row represents a different CpG probe, and columns contain sample-related information.
`bedmeth`	A character string specifying the BED data version to use for mapping CpG coverage. Options are ‘"v1"‘ (EPIC v1), ‘"v2"‘ (EPIC v2), or ‘"450k"‘ (450k array). Default is ‘"v1"‘.

Value

A list containing two plots (counts and percentage coverage) and the data frame with CpG counts and coverage information.

Examples

```
# Create synthetic CpG data with ICR annotation
set.seed(456)
n_cpgs <- 100

df_ICR_cpg_example <- data.frame(
  probeID = paste0("cg", sprintf("%08d", 1:n_cpgs)),
  cstart = seq(1000000, by = 50, length.out = n_cpgs),
  ICR = sample(c("ICR1", "ICR2", "ICR3"), n_cpgs, replace = TRUE),
  start = 1000000,
  end = 1005000
)

# Note: This uses synthetic data. In practice, use output from make_cpgs()
# coverage_plots <- plot_cpgs_coverage(df_ICR_cpg = df_ICR_cpg_example,
#                                      bedmeth = "v1")

## Not run:
plot_cpgs_coverage(df_ICR_cpg_counts, bedmeth = "v1")

## End(Not run)
```

`plot_line_ICR` *Plot Line Plot for ICR Methylation*

Description

This function generates a line plot to visualize methylation values across a specified ICR. Users can choose between a static ‘ggplot2‘ plot or an interactive ‘plotly‘ plot.

Usage

```
plot_line_ICR(significantDMPs, ICRcpg, ICR, sampleInfo, interactive = TRUE)
```

Arguments

`significantDMPs`	A data frame containing information about significant DMPs. Must include columns ‘ICR‘, ‘start‘, and ‘end‘.
`ICRcpg`	A data frame or matrix containing CpG methylation data. Includes CpG coordinates (‘cstart‘) and methylation values.
`ICR`	A character string specifying the name of the ICR region to be plotted.
`sampleInfo`	A character vector providing group labels (e.g., ‘"Control"‘ or ‘"Case"‘) for each sample in the methylation data.
`interactive`	A logical value indimessageing whether to return an interactive ‘plotly‘ plot (‘TRUE‘) or a static ‘ggplot2‘ plot (‘FALSE‘). Default is ‘TRUE‘.

Value

A plot representing the line plot of methylation values across the specified ICR region, highlighting significant DMPs. The plot is either a ‘ggplot2‘ object or a ‘plotly‘ object, depending on the value of ‘interactive‘.

Examples

```
# Create synthetic data for demonstration
set.seed(789)
n_cpgs <- 20
n_samples <- 6

# Create significant DMPs data
significantDMPs_example <- data.frame(
  cstart = seq(1000000, by = 100, length.out = n_cpgs),
  ICR = rep("KCNQ1OT1:TSS-DMR", n_cpgs),
  start = 1000000,
  end = 1002000,
  P.Value = runif(n_cpgs, 0.001, 0.049)
)

# Create ICRcpg data
meth_values <- matrix(runif(n_cpgs * n_samples, 0.3, 0.7),
                      nrow = n_cpgs, ncol = n_samples)
colnames(meth_values) <- paste0("Sample_", 1:n_samples)

ICRcpg_example <- data.frame(
  meth_values,
  cstart = seq(1000000, by = 100, length.out = n_cpgs),
  ICR = rep("KCNQ1OT1:TSS-DMR", n_cpgs),
  start = 1000000,
  end = 1002000
)

# Create sample info
sampleInfo_example <- c(rep("Control", 3), rep("Case", 3))

# Note: This uses synthetic data. In practice, use output from iDMPs()
# plot <- plot_line_ICR(significantDMPs = significantDMPs_example,
#                       ICRcpg = ICRcpg_example,
#                       ICR = "KCNQ1OT1:TSS-DMR",
#                       sampleInfo = sampleInfo_example,
#                       interactive = FALSE)

## Not run:
# Example data for significantDMPs
plot <- plot_line_ICR(significantDMPs, ICRcpg, ICR = "KCNQ1OT1:TSS-DMR", sampleInfo = sampleInfo, interactive = T)
print(plot)

## End(Not run)
```

preview_idat_zip *Preview IDAT ZIP Contents*

Description

Preview the contents of an IDAT ZIP file without processing it. Useful for checking file structure before full processing.

Usage

```
preview_idat_zip(zip_file)
```

Arguments

zip_file	Path to ZIP file

Value

List with ZIP contents summary

Examples

```
# Example showing how to use the function
# Requires an actual ZIP file with IDAT data
# preview_info <- preview_idat_zip("path/to/methylation_data.zip")
# print(preview_info)

# Check if function exists
if (exists("preview_idat_zip")) {
  print("Function preview_idat_zip is available")
}


# Preview IDAT ZIP file contents
# preview_info <- preview_idat_zip("methylation_data.zip")
# print(preview_info)
```

process_geo_dataset *Download and process GEO methylation dataset*

Description

Downloads IDAT files and phenotypic data from a GEO dataset and processes them for GIMP analysis.

Usage

```
process_geo_dataset(
  geo_id,
  download_dir = NULL,
  group_column = NULL,
  control_terms = c("control", "normal", "healthy", "ctrl"),
  case_terms = c("case", "disease", "tumor", "cancer", "patient"),
  max_samples = NULL,
  normalize_method = "quantile",
  n_cores = NULL
)
```

Arguments

geo_id	Character string of GEO accession (e.g., "GSE12345")
download_dir	Directory to download files (default: temporary directory)
group_column	Column name for sample groups (auto-detect if NULL)
control_terms	Terms that indicate control samples
case_terms	Terms that indicate case/treatment samples
max_samples	Maximum number of samples to process (NULL for all)
normalize_method	Normalization method for minfi
n_cores	Number of CPU cores for parallel processing

Value

List containing processed beta matrix and sample information

Examples

```
# Example showing function structure
# Note: Requires internet connection, GEOquery, and IDAT files in dataset
# geo_data <- process_geo_dataset("GSE68777")
# str(geo_data)

# Check if function is available
if (exists("process_geo_dataset")) {
  print("GEO processing function is available")
}


# Process a GEO dataset
geo_data <- process_geo_dataset(
    geo_id = "GSE68777",
    group_column = "characteristics_ch1.1",
    control_terms = c("normal", "control"),
    case_terms = c("tumor", "cancer")
)
```

```
# Use with GIMP functions
ICRcpg <- make_cpgs(Bmatrix = geo_data$beta_matrix, bedmeth = "v1")
```

`process_geo_with_mappings` *Process GEO dataset with user-specified group mappings*

Description

Downloads and processes a GEO dataset using user-defined group mappings instead of auto-detection. This is the main processing function after users have reviewed phenotypic data and defined their group mappings.

Usage

```
process_geo_with_mappings(
  geo_id,
  group_column,
  group_mappings,
  max_samples = NULL,
  normalize_method = "quantile",
  n_cores = NULL,
  download_dir = NULL
)
```

Arguments

`geo_id`	Character string of GEO accession
`group_column`	Column name containing group information
`group_mappings`	Named list mapping values to "Case", "Control", or "Exclude"
`max_samples`	Maximum number of samples to process
`normalize_method`	Normalization method for minfi
`n_cores`	Number of CPU cores for parallel processing
`download_dir`	Directory to download files (default: temporary directory)

Value

List containing processed beta matrix and sample information

Examples

```
# Example showing function structure
# Note: Requires internet connection and GEOquery package
# mappings <- list("tumor" = "Case", "normal" = "Control")
# result <- process_geo_with_mappings("GSE68777", "characteristics_ch1.1", mappings)

# Check if function is available
if (exists("process_geo_with_mappings")) {
  print("GEO processing with mappings function is available")
}


# Process with user-defined mappings
mappings <- list("tumor" = "Case", "normal" = "Control", "unknown" = "Exclude")
geo_data <- process_geo_with_mappings(
    geo_id = "GSE68777",
    group_column = "characteristics_ch1.1",
    group_mappings = mappings
)
```

`read_idat_zip` *Read IDAT Files from ZIP Archive This function extracts and processes IDAT files from a ZIP archive containing IDAT files and a sample sheet, returning a beta value matrix ready for GIMP analysis.*

Description

Read IDAT Files from ZIP Archive This function extracts and processes IDAT files from a ZIP archive containing IDAT files and a sample sheet, returning a beta value matrix ready for GIMP analysis.

Usage

```
read_idat_zip(
  zip_file,
  sample_sheet_name = "samplesheet.csv",
  array_type = c("EPIC", "450k", "EPICv2"),
  temp_dir = NULL,
  normalize_method = c("quantile", "SWAN", "funnorm", "noob"),
  detection_pval = 0.01,
  remove_failed_samples = TRUE,
  n_cores = NULL
)
```

Arguments

zip_file	Path to ZIP file containing IDAT files and sample sheet
sample_sheet_name	Name of the sample sheet file in the ZIP (default: "samplesheet.csv")
array_type	Array type for annotation ("450k", "EPIC", "EPICv2")
temp_dir	Temporary directory for extraction (default: creates temporary directory)
normalize_method	Normalization method for minfi ("quantile", "SWAN", "funnorm", "noob")
detection_pval	P-value threshold for detection (default: 0.01)
remove_failed_samples	Remove samples with >10 percent failed probes (default: TRUE)
n_cores	Number of CPU cores to use for parallel processing (default: NULL for sequential processing)

Value

A list containing:

beta_matrix	Beta value matrix ready for GIMP analysis
sample_info	Sample information from the sample sheet
qc_metrics	Quality control metrics
failed_samples	Names of samples that failed QC

Examples

```
# Example showing function structure
# Note: Requires an actual ZIP file containing IDAT files and sample sheet
# idat_data <- read_idat_zip("my_data.zip", array_type = "EPIC")
# str(idat_data)

# Check if function is available
if (exists("read_idat_zip")) {
  print("IDAT ZIP reading function is available")
}


# Read IDAT files from ZIP
idat_data <- read_idat_zip("my_methylation_data.zip", array_type = "EPIC")
beta_matrix <- idat_data$beta_matrix

# Use parallel processing with 4 cores
idat_data <- read_idat_zip("my_methylation_data.zip", array_type = "EPIC", n_cores = 4)

# Use with GIMP functions
ICRcpg <- make_cpgs(Bmatrix = beta_matrix, bedmeth = "v1")
```

validate_geo_dataset *Check if GEO ID exists and has IDAT data*

Description

Validates a GEO accession ID and checks if it contains raw IDAT files for methylation array analysis.

Usage

```
validate_geo_dataset(geo_id)
```

Arguments

geo_id	Character string of GEO accession (e.g., "GSE12345")

Value

List with validation results and metadata

Examples

```
# Example showing function usage structure
# Note: Requires internet connection and GEOquery package
# result <- validate_geo_dataset("GSE68777")
# print(result$valid)

# Check available function
if (exists("validate_geo_dataset")) {
  print("GEO validation function is available")
}


# Check a GEO dataset
result <- validate_geo_dataset("GSE68777")
if (result$valid && result$has_idats) {
    message("Dataset is suitable for GIMP analysis")
}
```

Index

∗ **datasets**
bed450k, 2
bedEPICv1, 3
bedEPICv2, 4
DMRs.hg19, 9
DMRs.hg38, 9

bed450k, 2
bedEPICv1, 3
bedEPICv2, 4

calculate_detection_pvalues, 5
check_minfi_functions, 5
create_bedmeth, 6
create_sample_sheet_template, 7

diagnose_geo_dataset, 8
DMRs.hg19, 9
DMRs.hg38, 9

get_geo_phenotype_data, 10
GIMP_app, 11

ICRs_heatmap, 12
iDMPs, 13

make_cpgs, 15
make_ICRs, 16

plot_cpgs_coverage, 17
plot_line_ICR, 18
preview_idat_zip, 20
process_geo_dataset, 20
process_geo_with_mappings, 22

read_idat_zip, 23

validate_geo_dataset, 25