Package ‘umiAnalyzer’

October 12, 2022

Type Package

Title Tools for Analyzing Sequencing Data with Unique Molecular Identifiers

Version 1.0.0

Date 2021-11-23

Maintainer Stefan Filges <stefan.filges@gu.se>


License GPL-3

URL https://github.com/sfilges/umiAnalyzer

BugReports https://github.com/sfilges/umiAnalyzer/issues

Depends R (>= 4.1.0)

Imports BiocManager, dplyr (>= 0.7.5), DT (>= 0.19), forcats (>= 0.5.0), ggplot2 (>= 2.2.1), graphics, grDevices, gridExtra (>= 2.3), magrittr (>= 1.5), methods, pheatmap (>= 1.0.12), plotly (>= 4.9.2.1), readr (>= 1.1.1), Rsamtools (>= 1.32.3), scales (>= 1.1.0), shiny (>= 1.7.1), shinydashboard (>= 1.2), shinyFiles (>= 0.9.0), shinyWidgets (>= 0.6.2), stats, stringr (>= 1.4.0), tibble (>= 1.4.2), tidyverse (>= 0.8.1), utils, viridis (>= 0.5.1)

Suggests knitr (>= 1.27), rmarkdown (>= 2.1)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

NeedsCompilation no

Author Stefan Filges [aut, cre] (<https://orcid.org/0000-0002-5994-6699>), Gustav Johansson [ctb]
addMetaData

Description

Add metaData
addUmiSample

Usage

addMetaData(object, attributeName, attributeValue)

Arguments

object R object to which meta data should be added
attributeName Name of the meta data attribute.
attributeValue Meta data to be saved.

Value

A UMIexperiment object

Examples

library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
metaData <- system.file("extdata", "metadata.txt", package = "umiAnalyzer")
simsen <- addMetaData(simsen, 'metaData', metaData)

addUmiSample

Add UMI sample to an existing experiment object

Description

Add UMI sample to an existing experiment object

Usage

addUmiSample(object, sampleName, sampleDir, clearData = FALSE)

Arguments

object UMIexperiment object
sampleName Name of new sample
sampleDir Directory to new sample
clearData Should other data in UMIexperiment be cleared

Value

A UMIexperiment object
AmpliconHeatmap

Description

Generates a heatmap of mutations with sample clustering using pheatmap.

Usage

AmpliconHeatmap(
  object,
  filter.name = "default",
  cut.off = 5,
  left.side = "columns",
  amplicons = NULL,
  samples = NULL,
  abs.count = FALSE,
  font.size = 10
)

Arguments

object
  Requires a UMI sample or UMI experiment object
filter.name
  Name of the filter to be plotted.
cut.off
  How many variant reads are necessary to consider a variant above background?
  Default is 5 reads.
left.side
  Show assays or sample on the left side of the heatmap. Default is assays
amplicons
  (Optional) character vector of amplicons to be plotted.
samples
  (Optional) character vector of samples to be plotted.
abs.count
  Logical. Should absolute counts be used instead of frequencies?
font.size
  Font size to use for sample labels

Value

A graphics object

Examples

## Not run:
library(umiAnalyzer)

main = system.file("extdata", package = 'umiAnalyzer')
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
simsen <- createUmiExperiment(experimentName = 'example',mainDir = main,sampleNames = samples)
simsen <- filterUmiObject(simsen)
AmpliconPlot

hmap <- AmpliconHeatmap(simsen)

## End(Not run)

---

**AmpliconPlot**  
*Generate Amplicon plots*

**Description**

Plots variant allele frequencies or alternate allele counts for chosen samples and assays.

**Usage**

```r
AmpliconPlot(
  object,
  filter.name = "default",
  cut.off = 5,
  min.count = 0,
  min.vaf = 0,
  amplicons = NULL,
  samples = NULL,
  abs.count = FALSE,
  y_min = 0,
  y_max = NULL,
  theme = "classic",
  option = "default",
  direction = "default",
  plot.text = FALSE,
  plot.ref = TRUE,
  stack.plot = FALSE,
  classic.plot = FALSE,
  fdr = 0.05,
  font.size = 6,
  angle = 45,
  use.caller = FALSE,
  use.plotly = TRUE
)
```

**Arguments**

- **object**  
  Requires a UMI sample or UMI experiment object

- **filter.name**  
  Name of the filter to be plotted.

- **cut.off**  
  How many variant reads are necessary to consider a variant above background? Default is 5 reads.

- **min.count**  
  Minimum variants counts to plot, default is 0.
min.vaf  Minimum variants allele frequency to plot, default is 0.
amplicons (Optional) character vector of amplicons to be plotted.
samples (Optional) character vector of samples to be plotted.
abs.count Should absolute counts be plotted instead of frequencies? Default is FALSE.
y_min Minimum y-axis value, default is 0
y_max Maximum y-axis value, default is NULL (autoscale)
theme Plotting theme to use, default is classic.
option Color palette to use.
direction Orientation of the color palette.
plot.text Should non-references bases be indicated above the bar?
plot.ref If true show reference base instead of position on x-axis.
stack.plot Show all variant alleles in a stacked bar plot.
classic.plot Show classical debarcer amplicon plot with raw error.
fdr False-discovery-rate cut-off for variants.
font.size Font size
angle Font angle
use.caller Should data from variant caller be used? Default is FALSE
use.plotly Should plotly be used instead of the regular ggplot device? Default is TRUE

Value

A UMIexperiment object containing a ggplot object with the amplicon plot.

Examples

library(umiAnalyzer)

main = system.file('extdata', package = 'umiAnalyzer')
samples <- list.files(path = main, full.names = FALSE, recursive = FALSE)
simsen <- createUmiExperiment(experimentName = 'example', mainDir = main, sampleNames = samples)
simsen <- filterUmiObject(simsen)

amplicon_plot <- AmpliconPlot(simsen)
BarcodeFamilyHistogram

Consensus depth histograms

Description

Generate histograms for the frequency of barcode family depths.

Usage

BarcodeFamilyHistogram(
  object,
  xMin = 0,
  xMax = 100,
  samples = NULL,
  option = "viridis",
  direction = 1,
  theme = "classic"
)

Arguments

object  Requires a UMI sample or UMI experiment object
xMin    Minimum consensus family size to plot, default is 0.
xMax    Maximum consensus family size to plot. Default is 100.
samples List of samples to be shown.
option  Color scheme to use
direction If using viridis colors sets the orientation of color scale.
theme   ggplot theme to use. Defaults to classic.

Value

A ggplot object

Examples

library(umiAnalyzer)
main = system.file("extdata", package = 'umiAnalyzer')
simsen <- createUmiExperiment(main, importBam = TRUE)
barcode_dist <- BarcodeFamilyHistogram(simsen)
beta_binom

**Description**

Code was obtained from VGAM package function VGAM::rbetabinom.ab. The VGAM package is available under the GPL-3 license and maintained by Thomas Yee <t.yee at auckland.ac.nz>. Source code of the function is identical to rbetabinom.ab, but the function name was changed to beta_binom.

**Usage**

```
beta_binom(n, size, shape1, shape2, limit.prob = 0.5, .dontuse.prob = NULL)
```

**Arguments**

- `n`
- `size`
- `shape1`
- `shape2`
- `limit.prob`
- `.dontuse.prob`

**Value**

Numeric

**References**


**Examples**

```
beta_binom(10, 5, 0.5, 1)
beta_binom(10, 2, 0.5, 1)
```
**callVariants**

callVariants using beta binomial distribution

**Description**

Calculate variant p-values using permutation-based testing. A prior is fitted to model the background error using maximum likelihood estimation of a beta distribution. The maximum likelihood estimate of the beta distribution is then used to define the shape of a beta-binomial distribution used to estimate variant P-Values. This can be interpreted as a probability for a variant to not have arisen by chance.

**Usage**

callVariants(object, minDepth = 3, minCoverage = 100, computePrior = FALSE)

**Arguments**

- **object**
  - A UMIErrorCorrect object.
- **minDepth**
  - Minimum consensus depth required default is 3
- **minCoverage**
  - Minimum Coverage to use, default is 100 reads.
- **computePrior**
  - Should a new distribution be derived from data? Default is FALSE.

**Value**

Object containing raw and FDR-adjusted P-Values

**See Also**

- `filterVariants` on how to filter variants.

**Examples**

```r
library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")

simsen <- createUmiExperiment(main)
simsen <- filterUmiObject(simsen)
simsen <- callVariants(simsen, computePrior = FALSE)
```
createUmiExperiment  

Method for creating a UMI experiment object

Description

Method for creating a UMI experiment object

Usage

```r
createUmiExperiment(
  mainDir,
  experimentName = NULL,
  sampleNames = NULL,
  importBam = FALSE,
  as.shiny = FALSE
)
```

Arguments

- `mainDir`  Main experiment directory
- `experimentName`  Name of the experiment
- `sampleNames`  List of sample names. Can be either NULL or list. If NULL all subdirectories of mainDir will be searched.
- `importBam`  Logical. Should bam files be imported on creation? Default is False.
- `as.shiny`  Set to TRUE if run within a shiny::withProgress environment

Value

An object of class UMIexperiment

Examples

```r
library(umiAnalyzer)

main = system.file('extdata', package = 'umiAnalyzer')

samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)

exp1 <- createUmiExperiment(experimentName = 'exp1', mainDir = main, sampleNames = samples)
```
Method for creating a UMI experiment object

**Description**

Method for creating a UMI experiment object

**Usage**

```r
createUMIexperiment_Debarcer(experiment.name, main.dir, dir.names)
```

**Arguments**

- `experiment.name`  
  Name of the experiment
- `main.dir`  
  Main experiment directory
- `dir.names`  
  List of sample names

**Value**

A UMIexperiment object

---

Method for creating a UMI sample from UMIErrorCorrect output.

**Description**

Method for creating a UMI sample from UMIErrorCorrect output.

**Usage**

```r
createUmiSample(sampleName, sampleDir, importBam = FALSE)
```

**Arguments**

- `sampleName`  
  UMI sample object name
- `sampleDir`  
  Path to UMI sample folders. Must be a folder generated by UMIErrorCorrect
- `importBam`  
  Logical. Should BAM files be imported at object initialization? Default is False.

**Value**

An object of class UMIsample
Examples

```r
library(umiAnalyzer)

main = system.file('extdata', package = 'umiAnalyzer')
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
s1 <- createUmiSample('s1', sampleDir = paste(main,"/",samples[1],sep=""))
```

createUmiSample_Debarcer

*Method for creating a UMI sample object*

Description

Method for creating a UMI sample object

Usage

```r
createUmiSample_Debarcer(sample.name, sample.dir, cons = "10")
```

Arguments

- `sample.name`: UMI sample object name
- `sample.dir`: Path to UMI sample
- `cons`: Consensus depth. Needs to be string; default is 10.

Value

A UMI sample object

---

download_template

*Download meta data template*

Description

Function for downloading a template file containing metadata.

Usage

```r
download_template(object)
```

Arguments

- `object`: A UMI experiment object
filterUmiObject

Value
A tibble containing a metadata template

Examples
library(umiAnalyzer)

main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
download_template(simsen)

filterUmiObject

Method for filtering UMIexperiment and sample objects

Description
Method for filtering UMIexperiment and sample objects

Usage
filterUmiObject(
  object,
  name = "default",
  minDepth = 3,
  minCoverage = 100,
  minFreq = 0,
  minCount = 0
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Requires a UMI sample or UMI experiment object.</td>
</tr>
<tr>
<td>name</td>
<td>String. Name of the filter. Default is &quot;default&quot;.</td>
</tr>
<tr>
<td>minDepth</td>
<td>Consensus depth to analyze. Default is 3.</td>
</tr>
<tr>
<td>minCoverage</td>
<td>Minimum coverage required for amplicons. Default is 1.</td>
</tr>
<tr>
<td>minFreq</td>
<td>Minimum variant allele frequency to keep. Default is 0.</td>
</tr>
<tr>
<td>minCount</td>
<td>Minimum variant allele count to keep. Default is 3.</td>
</tr>
</tbody>
</table>

Value
A UMI sample or UMI experiment object.
Examples

```r
library(umiAnalyzer)
main = system.file('extdata', package = 'umiAnalyzer')
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
simsen <- createUmiExperiment(experimentName = 'simsen', mainDir = main, sampleNames = samples)
simsen <- filterUmiObject(simsen)
```

---

**filterVariants**

*Filter variants based on p values or depth*

### Description

You can filter variants called with the "callVariants" function based on adjusted p-value, minimum variant allele count and supply a list of assays and samples to plot.

### Usage

```r
filterVariants(
  object,
  p.adjust = 0.2,
  minVarCount = 5,
  amplicons = NULL,
  samples = NULL
)
```

### Arguments

- `object`: A UMIexperiment object  
- `p.adjust`: Numeric. Adjusted p value (FDR). Default is 0.2.  
- `minVarCount`: Integer. Minimum variant allele count. Default is 5.  
- `amplicons`: NULL or list of assays to plot. NULL uses all.  
- `samples`: NULL or list of samples to plot. NULL uses all.

### Value

A UMIexperiment object with filtered variants. Can be used to generate VCF files.

### See Also

- `callVariants` on how to call variants.
findConsensusReads

Examples

```r
## Not run:
library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")

simsen <- createUmiExperiment(main)
simsen <- filterUmiObject(simsen)
simsen <- callVariants(simsen, computePrior = FALSE)
simsen <- filterVariants(simsen, p.adjust = 0.05)

## End(Not run)
```

findConsensusReads  
Find consensus reads A function to analyze consensus read tables generated with parseBamFiles or a UMIexperiment object containing reads.

Description

Find consensus reads A function to analyze consensus read tables generated with parseBamFiles or a UMIexperiment object containing reads.

Usage

```r
findConsensusReads(
  object,
  consDepth = 0,
  groupBy = c("none", "sample", "position", "both"),
  pattern = NULL
)
```

Arguments

- **object**: Either a tibble generated with parseBamFiles or a UMIexperiment object
- **consDepth**: Minimum consensus depth to keep. Default is 0.
- **groupBy**: Should data be grouped by position, sample, both or not at all.
- **pattern**: Regular expression

Value

A data table
generateVCF

Examples

```r
library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main, importBam = TRUE)

reads <- findConsensusReads(simsen)
reads
```

---

**generateVCF**  
*Generate VCF file from UMI sample or UMI experiment object*

**Description**

Generate VCF file from UMI sample or UMI experiment object

**Usage**

```r
generateVCF(object, outDir = getwd(), outFile, printAll = FALSE)
```

**Arguments**

- **object**: Requires a UMI sample or UMI experiment object
- **outDir**: String. Output directory, defaults to working directory.
- **outFile**: String. Name of the output file
- **printAll**: Logical. Should all or only trusted variant be printed?

**Value**

A VCF file

**Examples**

```r
## Not run:
library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
simsen <- filterUmiObject(simsen)
generateVCF(simsen, 'simsen.vcf', printAll = FALSE, save = FALSE)
## End(Not run)
```
Method for retrieving filtered data

Description

Method for retrieving filtered data

Usage

getFilteredData(
  object,
  name = "default",
  save = FALSE,
  outDir = getwd(),
  fileName = NULL,
  delim = ",;"
)

Arguments

object: Requires a UMI sample or UMI experiment object.
name: String. Name of the filter. Default is "default".
save: Logical, should data be saved as csv file? Default is FALSE.
outDir: Output directory
fileName: Filename to be used, default is the same as 'name'
delim: Character string denoting delimiter to be used, default is ';'.

Value

A filtered consensus table, as a tibble.

Examples

library(umiAnalyzer)

main = system.file('extdata', package = 'umiAnalyzer')

samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)

simsen <- createUmiExperiment(experimentName = 'simsen',mainDir = main,sampleNames = samples)
simsen <- filterUmiObject(simsen)

myfilter <- getFilteredData(simsen)

myfilter
getMetaData

Retrieves metadata by name.

Description

Retrieve metadata by name.

Usage

getMetaData(object, attributeName)

Arguments

object R object from which to get metadata.
attributeName Name of the metadata attribute.

Value

Metadata

Examples

library(umiAnalyzer)

main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
metaData <- system.file("extdata", "metadata.txt", package = "umiAnalyzer")
simsen <- importDesign(object = simsen, file = metaData)
design <- getMetaData(object = simsen, attributeName = "design")
design

importBedFile

Import bed file

Description

Import bed file

Usage

importBedFile(path)
importDesign

Arguments

path
path to bed file

Value
A table containing genome positions

importDesign Import experimental design meta data such as replicates, treatments, categorical variables.

Description
Import experimental design meta data such as replicates, treatments, categorical variables.

Usage

importDesign(object, file, delim = NULL)

Arguments

object UMI.experiment to which to add metadata
file File containing meta data
delim Column separator. Default is NULL (automatically determine delimiter)

Value
A UMIexperiment object

Examples

library(umiAnalyzer)

main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)

metaData <- system.file("extdata", "metadata.txt", package = "umiAnalyzer")
simsen <- importDesign(object = simsen, file = metaData)

# Retrieve meta data
design <- getMetaData(object = simsen, attributeName = "design")
design
mergeAssays

**Description**

Merge assays together by name. Requires a name of the new assay and a list of assays that will be merged.

**Usage**

```r
mergeAssays(object, name, assay.list)
```

**Arguments**

- `object`: A UMIexperiment object
- `name`: Name of the new assay
- `assay.list`: List of assays to merge

**Value**

merged consensus data

**Examples**

```r
library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
simsen <- mergeAssays(object = simsen, name = "new", assay.list = c("PIK3CA_123", "PIK3CA_234"))
```

---

parseBamFiles

**Description**

Function to parse bam files

**Usage**

```r
parseBamFiles(mainDir, sampleNames = NULL, consDepth = 0, as.shiny = FALSE)
```
QCplot

Arguments

mainDir Directory containing UMIErrorCorrect output folders.
sampleNames A list of sample names.
consDepth Only retain consensus reads of at least cons.depth. Default is 0.
as.shiny Set to TRUE if run within a shiny::withProgress environment

Value

A data table

Examples

library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)

reads <- parseBamFiles(main, consDepth = 10)

QCplot

Generate QC plots

Description

Visualize the UMI count for each selected assay and sample for a given consensus depth. This is useful to detect differences in coverage, especially for multiplexed assays.

Usage

QCplot(
  object,
  group.by = "sample",
  plotDepth = 3,
  assays = NULL,
  samples = NULL,
  theme = "classic",
  option = "viridis",
  direction = "default",
  toggle_mean = TRUE,
  center = "mean",
  line_col = "blue",
  angle = 0,
  plotly = FALSE
)

runUmiVisualizer

Arguments

- **object**: Requires a UMI sample or UMI experiment object
- **group.by**: String. Which variable should be used as a factor on the x-axis. Default is `sample`
- **plotDepth**: Which consensus depth to plot
- **assays**: (Optional) user-supplied list of assays to plot. Default is all.
- **samples**: (Optional) user-supplied list of samples to plot. Default is all.
- **theme**: ggplot theme to use.
- **option**: Color palette to use, either ggplot default or viridis colors.
- **direction**: If viridis colors are used, choose orientation of color scale.
- **toggle_mean**: Show mean or median
- **center**: Choose mean or median
- **line_col**: Choose color for mean/median line
- **angle**: Angle of labels on x-axis.
- **plotly**: Should plotly be used for rendering?

Value

A ggplot object

Examples

```r
library(umiAnalyzer)

main = system.file(‘extdata’, package = ‘umiAnalyzer’)
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
simsen <- createUmiExperiment(experimentName = ‘example’, mainDir = main, sampleNames = samples)

depth_plot <- QCplot(simsen)
```

Description

Function to run the umiVisualizer shiny app

Usage

```r
runUmiVisualizer()
```

Value

Opens the umiVisualizer app
saveConsData

Examples

```r
## Not run:
library(umiAnalyzer)
runUmiVisualizer()

## End(Not run)
```

---

saveConsData  
Save consensus data

Description

If `save` is set to TRUE data will be written to a csv file otherwise consensus data will be returned as a tibble.

Usage

```r
saveConsData(
  object,
  save = FALSE,
  fileName = "consensus_data.csv",
  outDir = getwd(),
  delim = ";"
)
```

Arguments

- `object`  
  UMIexperiment object
- `save`  
  Logical. Should data be saved to file? Default is FALSE.
- `fileName`  
  String. Name of the file to be saved. Default is 'consensus_data.csv'
- `outDir`  
  output directory, defaults to working directory
- `delim`  
  Single character string, either ';' or ',' or tab

Value

A data table

Examples

```r
library(umiAnalyzer)

main = system.file("extdata", package = "umiAnalyzer")
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
```
example <- createUmiExperiment(experimentName = 'example', mainDir = main, sampleNames = samples)

consensus_data <- saveConsData(object = example)

consensus_data

simsen

**UMIexperiment data generated with SiMSen-Seq**

**Description**

UMIexperiment data generated with SiMSen-Seq

**Format**

An object of class "UMIexperiment"

timeSeriesGrid

**Plot time series data**

**Description**

Function for plotting time series or other meta data. Uses facet wrap to display user-provided categorical variables.

**Usage**

timeSeriesGrid(
    object,
    filter.name = "default",
    cut.off = 5,
    min.count = 0,
    min.vaf = 0,
    amplicons = NULL,
    samples = NULL,
    x_variable = NULL,
    y_variable = "Max Non-ref Allele Frequency",
    columns = "Sample Name",
    rows = "Name",
    color_by = "Name",
    fdr = 0.05,
    use.caller = TRUE,
    bed_positions = NULL
)
Arguments

object  A consensus data table
filter.name  "default"
cut.off  5
min.count  0
min.vaf  0
amplicons  NULL
samples  NULL
x_variable  NULL
y_variable  "Max Non-ref Allele Frequency"
columns  "Sample Name"
rows  "Name"
color_by  "Name"
fdr  0.05
use.caller  TRUE
bed_positions  NULL

Value

A ggplot object.

Examples

library(umiAnalyzer)
main <- system.file("extdata", package = "umiAnalyzer")
simsen <- createUmiExperiment(main)
simsen <- filterUmiObject(simsen)
metaData <- system.file("extdata", "metadata.txt", package = "umiAnalyzer")
simsen <- importDesign(object = simsen, file = metaData)
bed_dir <- system.file("extdata", "simple.bed", package = "umiAnalyzer")
bed <- importBedFile(path = bed_dir)
time_plot <- timeSeriesGrid(simsen, x_variable = "time", bed_positions = bed)
UmiCountsPlot

Plot UMI counts

Description
Visualize the number detected UMI for each consensus depth cut-off. This may be helpful in choosing the right consensus depth for your analysis, by checking the number of reads still available for each assay and sample for your chosen cut-off.

Usage
UmiCountsPlot(
  object,
  amplicons = NULL,
  samples = NULL,
  theme = "classic",
  option = "viridis",
  direction = 1
)

Arguments

- **object**: Requires a UMI sample or UMI experiment object
- **amplicons**: (Optional) user-supplied list of assays to plot. Default is all.
- **samples**: (Optional) user-supplied list of samples to plot. Default is all.
- **theme**: Plotting theme, default is classic
- **option**: Color palette. Default uses ggplot standard, otherwise viridis options.
- **direction**: If using viridis colors should the scale be inverted or default?

Value
A UMIexperiment object

Examples
library(umiAnalyzer)

main = system.file('extdata', package = 'umiAnalyzer')
samples <- list.dirs(path = main, full.names = FALSE, recursive = FALSE)
simsen <- createUmiExperiment(experimentName = 'example', mainDir = main, sampleNames = samples)
simsen <- filterUmiObject(simsen)

count_plot <- UmiCountsPlot(simsen)
**UMIexperiment-class**  

**UMIexperiment class**

**Description**

The UMIexperiment is the core data object, storing all data and relevant analysis data associated with your experiment. Each object has number of slots storing raw data, graphs and processed data.

**Value**

An object of class UMIexperiment

**Slots**

- **name**  Optional project name for record keeping.
- **cons.data**  The raw consensus data supplied by the user.
- **summary.data**  Summary data from UMIErrorCorrect
- **raw.error**  Cons0 error profile
- **reads**  Consensus reads imported using the parseBamFiles function.
- **meta.data**  Sample data optionally supplied by the user.
- **filters**  A list of filtered cons.data, which can be accessed separately.
- **plots**  A list of generated plots.
- **variants**  Consensus table generated with the umiAnalyzer variant caller.
- **merged.data**  Data generated using the mergeTechnicalReplicates function.

---

**UMIsample-class**  

**UMIsample class**

**Description**

UMIsample class

**Value**

An object of class UMIsample

**Slots**

- **name**  Sample name
- **cons.data**  Raw consensus data
- **summary.data**  Summary data from UMIErrorCorrect
- **reads**  Consensus reads imported from a bam file.
Index

* datasets
  - simsen, 24

addMetaData, 2
addUmiSample, 3
AmpliconHeatmap, 4
AmpliconPlot, 5
BarcodeFamilyHistogram, 7
beta_binom, 8
callVariants, 9, 14
createUmiExperiment, 10
createUmiExperiment_Debarcer, 11
createUmiSample, 11
createUMIsample_Debarcer, 12
download_template, 12
filterUmiObject, 13
filterVariants, 9, 14
findConsensusReads, 15
generateVCF, 16
getFilteredData, 17
getMetaData, 18
importBedFile, 18
importDesign, 19
mergeAssays, 20
parseBamFiles, 20
QCplot, 21
runUmiVisualizer, 22
saveConsData, 23
simsen, 24
timeSeriesGrid, 24

UmiCountsPlot, 26
UMIexperiment (UMIexperiment-class), 27
UMIexperiment-class, 27
UMIsample (UMIsample-class), 27
UMIsample-class, 27