

# Package ‘UMI4Cats’

March 30, 2021

**Title** UMI4Cats: Processing, analysis and visualization of UMI-4C chromatin contact data

**Version** 1.0.1

**URL** <https://github.com/Pasquali-lab/UMI4Cats>

**Description** UMI-4C is a technique that allows characterization of 3D chromatin interactions with a bait of interest, taking advantage of a sonication step to produce unique molecular identifiers (UMIs) that help remove duplication bias, thus allowing a better differential comparison of chromatin interactions between conditions. This package allows processing of UMI-4C data, starting from FastQ files provided by the sequencing facility. It provides two statistical methods for detecting differential contacts and includes a visualization function to plot integrated information from a UMI-4C assay.

**BugReports** <https://github.com/Pasquali-lab/UMI4Cats/issues>

**License** Artistic-2.0

**Encoding** UTF-8

**Depends** R (>= 4.0.0), SummarizedExperiment

**Imports** magick, cowplot, scales, GenomicRanges, ShortRead, zoo, ggplot2, reshape2, regioneR, IRanges, S4Vectors, magrittr, dplyr, BSgenome, Biostrings, DESeq2, R.utils, Rsamtools, stringr, Rbowtie2, methods, GenomeInfoDb, GenomicAlignments, RColorBrewer, utils, grDevices, stats, org.Hs.eg.db, annotate, TxDb.Hsapiens.UCSC.hg19.knownGene, rlang, GenomicFeatures, BiocFileCache, rappdirs, fda

**Suggests** knitr, rmarkdown, BiocStyle, BSgenome.Hsapiens.UCSC.hg19, tidyr, testthat

**biocViews** QualityControl, Preprocessing, Alignment, Normalization, Visualization, Sequencing, Coverage

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**git\_url** <https://git.bioconductor.org/packages/UMI4Cats>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** 1f6f3d8

**git\_last\_commit\_date** 2020-12-29

**Date/Publication** 2021-03-29

**Author** Mireia Ramos-Rodriguez [aut, cre]  
 (<<https://orcid.org/0000-0001-8083-2445>>),  
 Marc Subirana-Granes [aut]

**Maintainer** Mireia Ramos-Rodriguez <[mireiarr9@gmail.com](mailto:mireiarr9@gmail.com)>

## R topics documented:

.getCache	3
.getSummaryBam	3
.singleAlignmentUMI4C	4
.singleCounterUMI4C	4
.singlePrepUMI4C	5
.singleSplitUMI4C	6
.smoothMonotone	7
addGrouping	7
addStepping	8
alignmentUMI4C	9
calculateAdaptativeTrend	10
calculateDomainogram	10
contactsUMI4C	11
counterUMI4C	12
createGeneAnnotation	14
createStatsTable	15
darken	15
demultiplexFastq	16
deseq2UMI4C	17
dgram	17
differentialNbinomWaldTestUMI4C	19
digestGenome	21
downloadUMI4CexampleData	22
ex_ciita_umi4c	22
fisherUMI4C	23
formatPlotsUMI4C	24
geoMeanCoordinates	25
getColors	25
getFactors	26
getNormalizationMatrix	26
getViewpointCoordinates	27
groupSamplesUMI4C	28
nbinomWaldTestUMI4C	28
plotDifferential	29
plotDomainogram	29
plotGenes	30
plotTrend	31
plotUMI4C	32
prepUMI4C	33
smoothMonotoneUMI4C	34
splitUMI4C	35
statsUMI4C	35
theme	36
themeXblank	37

<code>.getCache</code>	3
<code>themeXYblank</code>	37
<code>themeYblank</code>	38
<code>UMI4C</code>	38
<code>UMI4C2dds</code>	41
<code>UMI4Cats</code>	42
<code>vstUMI4C</code>	43
<b>Index</b>	<b>44</b>

---

<code>.getCache</code>	<i>Get BiocFileCache object</i>
------------------------	---------------------------------

---

### Description

Get BiocFileCache object

### Usage

```
.getCache()
```

### Value

Returns BFC object with the cache for the UMI4Cats package

---

<code>.getSummaryBam</code>	<i>Summarize BAM file</i>
-----------------------------	---------------------------

---

### Description

Get summary of interesting bam statistics

### Usage

```
.getSummaryBam(bam_file, mapped = TRUE, secondary = FALSE)
```

### Arguments

<code>bam_file</code>	Path for the bam file.
<code>mapped</code>	Logical indicating whether to extract mapped reads.
<code>secondary</code>	Logical indicating whether to extract secondary aligned reads.

### Value

Returns a numeric containing the number of reads in `bam_file` that has the specified mapped and secondary status.

---

*.singleAlignmentUMI4C* *Align split fastq file*

---

**Description**

Align split fastq file

**Usage**

```
.singleAlignmentUMI4C(
  split_file,
  align_dir,
  threads = 1,
  bowtie_index,
  pos_viewpoint,
  filter_bp = 1e+07
)
```

**Arguments**

<code>split_file</code>	Split fastq file to align.
<code>align_dir</code>	Directory where to save aligned files.
<code>threads</code>	Number of threads to use in the analysis. Default=1.
<code>bowtie_index</code>	Path and prefix of the bowtie index to use for the alignment.
<code>pos_viewpoint</code>	GRanges object containing the genomic position of the viewpoint.
<code>filter_bp</code>	Integer indicating the bp upstream and downstream of the viewpoint to select for further analysis. Default=10e6

**Value**

Creates a BAM file in `wk_dir/align` named "`basename(fastq)_filtered.bam`", containing the aligned filtered reads. A data.frame object with the statistics is also returned.

---

*.singleCounterUMI4C* *Count UMIs for a given bam file.*

---

**Description**

Count UMIs for a given bam file.

**Usage**

```
.singleCounterUMI4C(
  filtered_bam_R1,
  filtered_bam_R2,
  digested_genome_gr,
  pos_viewpoint,
  res_enz,
  count_dir,
  filter_bp = 1e+07
)
```

### Arguments

filtered_bam_R1	R1 bam file.
filtered_bam_R2	R2 bam file.
digested_genome_gr	GRanges object containing the coordinates for the digested genome.
pos_viewpoint	Vector consist of chromosome, start and end position of the viewpoint.
res_enz	Character containing the restriction enzyme sequence.
count_dir	Counter directory.
filter_bp	Integer indicating the bp upstream and downstream of the viewpoint to select for further analysis. Default=10e6

### Value

Creates a tab-delimited file in `wk_dir/count` named "`basename(fastq) _counts.tsv`", containing the coordinates for the viewpoint fragment, contact fragment and the number of UMIs detected in the ligation.

---

`.singlePrepUMI4C`      *Prepar fastq files at a given barcode.*

---

### Description

Prepar fastq files at a given barcode.

### Usage

```
.singlePrepUMI4C(  
  fq_R1,  
  fq_R2,  
  bait_seq,  
  bait_pad,  
  res_enz,  
  prep_dir,  
  numb_reads = 1e+11  
)
```

### Arguments

fq_R1	Fastq file R1.
fq_R2	Fastq file R2.
bait_seq	Character containing the bait primer sequence.
bait_pad	Character containing the pad sequence (sequence between the bait primer and the restriction enzyme sequence).
res_enz	Character containing the restriction enzyme sequence.
prep_dir	Prep directory.
numb_reads	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.

**Value**

Creates a compressed FASTQ file in `wk_dir/prep` named `basename(Fastq)).fq.gz`, containing the filtered reads with the UMI sequence in the header. A `data.frame` object with the statistics is also returned.

---

`.singleSplitUMI4C`      *Split fastq files at a given restriction site.*

---

**Description**

Split fastq files at a given restriction site.

**Usage**

```
.singleSplitUMI4C(
  fastq_file,
  res_enz,
  cut_pos,
  split_dir,
  min_flen = 20,
  numb_reads
)
```

**Arguments**

<code>fastq_file</code>	Fastq file path.
<code>res_enz</code>	Character containing the restriction enzyme sequence.
<code>cut_pos</code>	Numeric indicating the nucleotide position where restriction enzyme cuts (zero-based) (for example, for DpnII is 0).
<code>split_dir</code>	Directory where to save split files.
<code>min_flen</code>	Minimal fragment length to use for selecting the fragments. Default=20
<code>numb_reads</code>	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.

**Value**

Creates a compressed FASTQ file in `wk_dir/split` named `basename(fastq)).fq.gz`, containing the split reads based on the restriction enzyme used.

---

.smoothMonotone      *Monotone smoothing of the VST counts*

---

### Description

Takes the variance stabilized count values and calculates a symmetric monotone fit for the distance dependency. The signal trend is fitted using the `fda` package.

### Usage

```
.smoothMonotone(trafo_counts, alpha = 20, penalty = 0.1, frag_data)
```

### Arguments

trafo_counts	Variance stabilized count values assay from DDS object.
alpha	Approximate number of fragments desired for every basis function of the B-spline basis. $\text{floor}((\text{max}(\text{number of fragments})) / \text{alpha})$ is passed to <code>create.bspline.basis</code> as <code>nbasis</code> argument. 4 is the minimum allowed value. Default: 20.
penalty	Amount of smoothing to be applied to the estimated functional parameter. Default: 0.1.
frag_data	Data frame with all the information on restriction fragments and the interval around the viewpoint.

### Details

This function computes the smoothing function for the VST values, based on `fda` package, and calculates a symmetric monotone fit counts for the distance dependency

### Value

A dataframe with monotone smoothed fit counts.

---

addGrouping      *Add grouping of UMI-4C samples*

---

### Description

This function can be used to add specific groupings to UMI4C objects.

### Usage

```
addGrouping(  
  umi4c,  
  grouping = "sampleID",  
  scales = 5:150,  
  normalized = TRUE,  
  sd = 2  
)
```

**Arguments**

umi4c	UMI4C object as generated by <code>makeUMI4C</code> .
grouping	Name of the column in <code>colData</code> used to merge the samples or replicates. Set to NULL for skipping grouping. Default: "condition".
scales	Numeric vector containing the scales for calculating the domainogram.
normalized	Logical indicating whether UMI-4C profiles should be normalized to the <code>ref_umi4c</code> sample/group. Default: TRUE
sd	Standard deviation for adaptive trend.

**Value**

Adds a new UMI4C object into the `groupsUMI4C` slot with samples grouped according to grouping variable.

**Examples**

```
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")
```

---

addStepping	<i>Add stepping for plotting genes</i>
-------------	--

---

**Description**

Given a [GRanges](#) dataset representing genes, will add an arbitrary value for them to be plotted in the Y axis without overlapping each other.

**Usage**

```
addStepping(genesDat, coordinates, mcol.name)
```

**Arguments**

genesDat	GRanges object containing gene information.
coordinates	GRanges object with coordinates you want to plot.
mcol.name	Integer containing the column number that contains the gene name.

**Value**

Calculates the stepping position to avoid overlap between genes.

---

alignmentUMI4C	<i>UMI4C alignment</i>
----------------	------------------------

---

### Description

Align split UMI-4C reads to a reference genome using Bowtie2.

### Usage

```
alignmentUMI4C(  
  wk_dir,  
  pos_viewpoint,  
  bowtie_index,  
  threads = 1,  
  filter_bp = 1e+07  
)
```

### Arguments

wk_dir	Working directory where to save the outputs generated by the UMI-4c analysis.
pos_viewpoint	GRanges object containing the genomic position of the viewpoint. It can be generated by getViewpointCoordinates function.
bowtie_index	Path and prefix of the bowtie index to use for the alignment.
threads	Number of threads to use in the analysis. Default=1.
filter_bp	Integer indicating the bp upstream and downstream of the viewpoint to select for further analysis. Default=10e6

### Value

Creates a BAM file in wk\_dir/align named "basename(fastq))\_filtered.bam", containing the aligned filtered reads. The alignment log is also generated in wk\_dir/logs named "umi4c\_alignment\_stats.txt".

### Examples

```
if (interactive()){  
  path <- downloadUMI4CexampleData(reduced = TRUE)  
  alignmentUMI4C(  
    wk_dir = file.path(path, "CIITA"),  
    pos_viewpoint = GenomicRanges::GRanges("chr16:10972515-10972548"),  
    bowtie_index = file.path(path, "ref_genome", "ucsc.hg19.chr16")  
  )  
}
```

---

```
calculateAdaptativeTrend
```

*Adaptative smoothing of normalized trend*

---

### Description

Will perform adaptative smoothing will scaling one profile to the reference UMI-4C profile.

### Usage

```
calculateAdaptativeTrend(umi4c, sd = 2, normalized = TRUE)
```

### Arguments

umi4c	UMI4C object as generated by <a href="#">makeUMI4C</a> .
sd	Stantard deviation for adaptative trend.
normalized	Logical indicating whether UMI-4C profiles should be normalized to the ref_umi4c sample/group. Default: TRUE

### Value

Calculates the adaptative trend considering the minimum number of molecules to use for merging different restriction fragments. It also calculates the geometric mean of the coordinates of the merged restriction fragments.

---

```
calculateDomainogram Create Domainogram
```

---

### Description

Using as input the raw UMIs, this function creates a domainogram for the supplied scales.

### Usage

```
calculateDomainogram(umi4c, scales = 5:150, normalized = TRUE)
```

### Arguments

umi4c	UMI4C object as generated by <a href="#">makeUMI4C</a> .
scales	Integer vector indicating the number of scales to use for the domainogram creation. Default: 5:150.
normalized	Logical whether the the resulting domainograms should be normalized or not. Default: TRUE.

### Value

A matrix where the first column represents the fragment end coordinates (start) and the rest represent the number of UMIs found when using a specific scale.

---

 contactsUMI4C

*UMI4C Contacts Processing*


---

### Description

Using demultiplexed FastQ files as input, performs all necessary steps to end up with a tsv file summarizing the restriction enzyme fragments and the number of UMIs supporting that specific contact with the viewpoint (bait) of interest.

### Usage

```
contactsUMI4C(
  fastq_dir,
  wk_dir,
  file_pattern = NULL,
  bait_seq,
  bait_pad,
  res_enz,
  cut_pos,
  digested_genome,
  bowtie_index,
  threads = 1,
  numb_reads = 1e+11,
  rm_tmp = TRUE,
  min_flen = 20,
  filter_bp = 1e+07,
  ref_gen,
  sel_seqname = NULL
)
```

### Arguments

fastq_dir	Path of the directory containing the FastQ files (compressed or uncompressed).
wk_dir	Working directory where to save the outputs generated by the UMI-4c analysis.
file_pattern	Character that can be used to filter the files you want to analyze in the fastq_dir.
bait_seq	Character containing the bait primer sequence.
bait_pad	Character containing the pad sequence (sequence between the bait primer and the restriction enzyme sequence).
res_enz	Character containing the restriction enzyme sequence.
cut_pos	Numeric indicating the nucleotide position where restriction enzyme cuts (zero-based) (for example, for DpnII is 0).
digested_genome	Path for the digested genome file generated using the <a href="#">digestGenome</a> function.
bowtie_index	Path and prefix of the bowtie index to use for the alignment.
threads	Number of threads to use in the analysis. Default=1.
numb_reads	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.

rm_tmp	Logical indicating whether to remove temporary files (sam and intermediate bams). TRUE or FALSE. Default=TRUE.
min_flen	Minimal fragment length to use for selecting the fragments. Default=20
filter_bp	Integer indicating the bp upstream and downstream of the viewpoint to select for further analysis. Default=10e6
ref_gen	A BSgenome object of the reference genome.
sel_seqname	A character with the chromosome name to focus the search for the viewpoint sequence.

### Value

This function is a combination of calls to other functions that perform the necessary steps for processing UMI-4C data.

### Examples

```

if (interactive()) {
  path <- downloadUMI4CexampleData()

  hg19_dpnii <- digestGenome(
    cut_pos = 0,
    res_enz = "GATC",
    name_RE = "DpnII",
    ref_gen = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19,
    out_path = file.path(path, "digested_genome")
  )

  raw_dir <- file.path(path, "CIITA", "fastq")

  contactsUMI4C(
    fastq_dir = raw_dir,
    wk_dir = file.path(path, "CIITA"),
    bait_seq = "GGACAAGCTCCCTGCAACTCA",
    bait_pad = "GGACTTGCA",
    res_enz = "GATC",
    cut_pos = 0,
    digested_genome = hg19_dpnii,
    bowtie_index = file.path(path, "ref_genome", "ucsc.hg19.chr16"),
    threads = 1,
    numb_reads = 10e10,
    ref_gen = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19,
    sel_seqname = "chr16"
  )

  unlink(path, recursive=TRUE)
}

```

**Description**

Algorithm for counting and collapsing the number of UMIs supporting a specific ligation.

**Usage**

```
counterUMI4C(
  wk_dir,
  pos_viewpoint,
  res_enz,
  digested_genome,
  filter_bp = 1e+07
)
```

**Arguments**

<code>wk_dir</code>	Working directory where to save the outputs generated by the UMI-4c analysis.
<code>pos_viewpoint</code>	GRanges object containing the genomic position of the viewpoint.
<code>res_enz</code>	Character containing the restriction enzyme sequence.
<code>digested_genome</code>	Path for the digested genome file generated using the <a href="#">digestGenome</a> function.
<code>filter_bp</code>	Integer indicating the bp upstream and downstream of the viewpoint to select for further analysis. Default=10e6.

**Details**

For collapsing different molecules into the same UMI, takes into account the ligation position and the number of UMI sequence mismatches.

**Value**

Creates a compressed tab-delimited file in `wk_dir/count` named `"basename(fastq)_counts.tsv.gz"`, containing the coordinates for the viewpoint fragment, contact fragment and the number of UMIs detected in the ligation.

**Examples**

```
if (interactive()) {
  path <- downloadUMI4CexampleData(reduced = TRUE)

  hg19_dpni <- digestGenome(
    cut_pos = 0,
    res_enz = "GATC",
    name_RE = "DpnII",
    sel_chr = "chr16", # digest only chr16 to make example faster
    ref_gen = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19,
    out_path = file.path(path, "digested_genome")
  )

  viewpoint <- GenomicRanges::GRanges("chr16:10972515-10972548")

  counterUMI4C(
    wk_dir = file.path(path, "CIITA"),
    pos_viewpoint = viewpoint,
```

```
    res_enz = "GATC",
    digested_genome = hg19_dpni
  )
}
```

---

createGeneAnnotation *Create gene annotation object*

---

## Description

Create gene annotation object

## Usage

```
createGeneAnnotation(
  window,
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene : TxDb.Hsapiens.UCSC.hg19.knownGene,
  longest = TRUE
)
```

## Arguments

window	<a href="#">GRanges</a> object with coordinates to use for selecting the genes to plot.
TxDb	TxDb object to use for drawing the genomic annotation.
longest	Logical indicating whether to plot only the longest transcripts for each gene in the gene annotation plot.

## Value

GRanges object with the gene annotation in the window.

## Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
window <- GRanges("chr16:11298262-11400036")
gene_anno <- createGeneAnnotation(
  window = window,
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene
)
```

---

createStatsTable	<i>Create stats table</i>
------------------	---------------------------

---

**Description**

Create a statistical summary of the UMI-4C experiments analyzed with contactsUMI4C.

**Usage**

```
createStatsTable(wk_dir)
```

**Arguments**

wk\_dir            Working directory where to save the outputs generated by the UMI-4c analysis.

**Value**

Returns a data.frame summarizing all the different statistics for each sample analyzed in wk\_dir.

---

darken	<i>Darken colors</i>
--------	----------------------

---

**Description**

Darken colors

**Usage**

```
darken(color, factor = 1.4)
```

**Arguments**

color            Character containing the name or hex value of a color.  
factor           Numeric representing a factor by which darken the specified color.

**Value**

Darkens the provided color by the provided factor.

**Examples**

```
darken("blue", factor = 1.4)
```

---

demultiplexFastq      *Demultiplex FASTQ files using fastq-multx*

---

### Description

Demultiplex FASTQ files containng different bait information

### Usage

```
demultiplexFastq(barcodes, fastq, out_path = "raw_fastq", numb_reads = 1e+11)
```

### Arguments

barcodes	Dataframe with "name of sample" and "barcode" for every sample to demultiplex.
fastq	Fastq to demultiplex containing mate 1s. Different pairs should be named as "_R1" or "_R2". Allowed formats: _R1.fastq.gz, _R1.fq.gz, _R1.fastq or _R1.fq.
out_path	Path where to save the demultiplex output. Defaults to a path named raw_fastq in your working directory.
numb_reads	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.

### Value

Paired-end FastQ files demultiplexed in a compressed format. A log file with the statistics is also generated in out\_path named barcode\_umi4cats\_demultiplexFastq\_stats.txt.

### Examples

```
## Not run:
path <- downloadUMI4CexampleData(use_sample = TRUE)
fastq <- file.path(path, "CIITA", "fastq", "sub_ctr1_hi19_CIIITA_R1.fastq.gz")
barcodes <- data.frame(
  sample = c("CIITA"),
  barcode = c("GGACAAGCTCCCTGCAACTCA")
)

demultiplexFastq(
  barcodes = barcodes,
  fastq = fastq,
  out_path = path
)

## End(Not run)
```

---

deseq2UMI4C	<i>DDS object to UMI4Cats object.</i>
-------------	---------------------------------------

---

### Description

Transforms an DDS object to a UMI4C object after applying `nbinomWaldTestUMI4C`.

### Usage

```
deseq2UMI4C(
  umi4c,
  dds,
  design = ~condition,
  normalized = TRUE,
  padj_method = "fdr",
  padj_threshold = 0.05
)
```

### Arguments

<code>umi4c</code>	UMI4C object as generated by <code>makeUMI4C</code> or the UMI4C constructor.
<code>dds</code>	DDS object as generated by <code>nbinomWaldTestUMI4C</code> with the DESeq2 Wald Test results
<code>design</code>	A formula or matrix. The formula expresses how the counts for each fragment end depend on the variables in <code>colData</code> . See <a href="#">DESeqDataSet</a> .
<code>normalized</code>	Logical indicating if the function should return normalized or raw UMI counts. Default: TRUE.
<code>padj_method</code>	The method to use for adjusting p-values, see <a href="#">p.adjust</a> . Default: <code>fdr</code> .
<code>padj_threshold</code>	Numeric indicating the adjusted p-value threshold to use to define significant differential contacts. Default: 0.05.

### Value

UMI4C object with the DESeq2 Wald Test results.

---

dgram	<i>UMI4C class methods</i>
-------	----------------------------

---

### Description

This page contains a summary of the different methods used to access the information contained inside the `UMI4C-class` object. See the details section for more information on the different accessors.

**Usage**

```

dgram(object)

dgram(object) <- value

groupsUMI4C(object, value)

groupsUMI4C(object) <- value

bait(object)

trend(object)

resultsUMI4C(object, format = "GRanges", counts = TRUE, ordered = FALSE)

## S4 method for signature 'UMI4C'
dgram(object)

## S4 replacement method for signature 'UMI4C'
dgram(object) <- value

## S4 method for signature 'UMI4C'
groupsUMI4C(object)

## S4 replacement method for signature 'UMI4C'
groupsUMI4C(object) <- value

## S4 method for signature 'UMI4C'
bait(object)

## S4 method for signature 'UMI4C'
trend(object)

## S4 method for signature 'UMI4C'
resultsUMI4C(object, format = "GRanges", counts = FALSE, ordered = FALSE)

```

**Arguments**

object	a UMI4C-class object.
value	Alternative list of dgrams to replace the current slot.
format	Either "GRanges" (default) or "data.frame", indicating the format output of the results.
counts	Logical indicating whether counts for the different region should be provided. Default: FALSE.
ordered	Logical indicating whether to sort output by significance (adjusted p-value). Default: FALSE.

**Value**

There are several accessors to easily retrieve information from a UMI4C-class object:

- dgram: Returns a named list with the output domainograms for each sample.

- `bait`: Returns a [GRanges](#) object with the position of the bait.
- `trend`: Returns a data.frame in long format with the values of the adaptive smoothen trend.
- `resultsUMI4C`: Returns a [GRanges](#) or data.frame with the results of the differential analysis.

### See Also

UMI4C, UMI4C-class

### Examples

```
# Access the different information inside the UMI4C object
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")

dgram(ex_ciita_umi4c)
bait(ex_ciita_umi4c)
head(trend(ex_ciita_umi4c))

# Perform differential test
enh <- GRanges(c("chr16:10925006-10928900", "chr16:11102721-11103700"))
umi_dif <- fisherUMI4C(ex_ciita_umi4c, query_regions = enh,
                      filter_low = 20, resize = 5e3)
resultsUMI4C(umi_dif)
```

---

differentialNbinomWaldTestUMI4C

*Differential UMI4C contacts using DESeq2 Wald Test*

---

### Description

Using a UMI4C object, infers the differences between conditions specified in design of the smooth monotone fitted values using a Wald Test from DESeq2 package.

### Usage

```
differentialNbinomWaldTestUMI4C(
  umi4c,
  design = ~condition,
  normalized = TRUE,
  padj_method = "fdr",
  query_regions = NULL,
  padj_threshold = 0.05,
  penalty = 0.1,
  alpha = 20
)
```

### Arguments

<code>umi4c</code>	UMI4C object as generated by <code>makeUMI4C</code> or the UMI4C constructor.
<code>design</code>	A formula or matrix. The formula expresses how the counts for each fragment end depend on the variables in <code>colData</code> . See <a href="#">DESeqDataSet</a> .

normalized	Logical indicating if the function should return normalized or raw UMI counts. Default: TRUE.
padj_method	The method to use for adjusting p-values, see <a href="#">p.adjust</a> . Default: fdr.
query_regions	GRanges object or data.frame containing the coordinates of the genomic regions you want to use to perform the analysis in specific genomic intervals. Default: NULL.
padj_threshold	Numeric indicating the adjusted p-value threshold to use to define significant differential contacts. Default: 0.05.
penalty	Amount of smoothing to be applied to the estimated functional parameter. Default: 0.1.
alpha	Approximate number of fragments desired for every basis function of the B-spline basis. $\text{floor}((\max(\text{number of fragments})) / \alpha)$ is passed to <code>create.bspline.basis</code> as <code>nbasis</code> argument. 4 is the minimum allowed value. Default: 20.

### Details

This function fits the signal trend of a variance stabilized count values using a symmetric monotone fit for the distance dependency. Then scales the raw counts across the samples to obtain normalized factors. Finally, it detects differences between conditions applying the DESeq2 Wald Test.

### Value

UMI4C object with the DESeq2 Wald Test results.

### Examples

```
files <- list.files(system.file("extdata", "CIITA", "count", package="UMI4Cats"),
                  pattern = "*_counts.tsv.gz",
                  full.names = TRUE
)
# Create colData including all relevant information
colData <- data.frame(
  sampleID = gsub("_counts.tsv.gz", "", basename(files)),
  file = files,
  stringsAsFactors = FALSE
)

library(tidyr)
colData <- colData %>%
  separate(sampleID,
           into = c("condition", "replicate", "viewpoint"),
           remove = FALSE
)

# Make UMI-4C object including grouping by condition
umi <- makeUMI4C(
  colData = colData,
  viewpoint_name = "CIITA",
  grouping = NULL,
  bait_expansion = 2e6
)

umi_wald <- differentialNbinomWaldTestUMI4C(umi4c=umi,
```

```
design=~condition,
alpha = 100)
```

---

```
digestGenome          Digest reference genome
```

---

### Description

Performs an *in silico* digestion of a given reference genome using a given restriction enzyme sequence.

### Usage

```
digestGenome(
  res_enz,
  cut_pos,
  name_RE,
  ref_gen,
  sel_chr = paste0("chr", c(seq_len(22), "X", "Y")),
  out_path = "digested_genome/"
)
```

### Arguments

res_enz	Character containing the restriction enzyme sequence.
cut_pos	Numeric indicating the nucleotide position where restriction enzyme cuts (zero-based) (for example, for DpnII is 0).
name_RE	Restriction enzyme name.
ref_gen	A BSgenome object of the reference genome.
sel_chr	Character vector indicating which chromosomes to select for the digestion. Default: chr1-22, chrX, chrY.
out_path	Output path where to save the genomic track. The default is a directory named digested_genome/ created in your working directory. The rda objects are saved in folder named by the ref_gene_name_RE in the out_path folder.

### Value

Creates a rda file for every chromosome defined in sel\_chr.

### Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
ref_gen <- BSgenome.Hsapiens.UCSC.hg19

hg19_dpnii <- digestGenome(
  res_enz = "GATC",
  cut_pos = 0,
  name_RE = "dpnII",
  sel_chr = "chr16", # Only in chr16 to reduce example running time
  ref_gen = ref_gen,
  out_path = file.path(tempdir(), "digested_genome/")
)
```

---

`downloadUMI4CexampleData`*Download UMI4Cats example datasets*

---

**Description**

Downloads the required UMI4Cats example datasets.

**Usage**

```
downloadUMI4CexampleData(out_dir = tempdir(), verbose = TRUE, reduced = FALSE)
```

**Arguments**

<code>out_dir</code>	Output directory for the datasets, defaults to <code>tempdir()</code> .
<code>verbose</code>	Whether to print verbose messages or not. Default: <code>TRUE</code> .
<code>reduced</code>	Whether to use a reduced dataset to make test functions run faster.

**Value**

It creates the `output_dir` with the example UMI-4C files used by the vignette and examples. Takes advantage of the `BiocFileCache` package to make sure that the file has not been previously downloaded by the user.

**Examples**

```
if (interactive()) {  
  # Using reduced data data to make example faster.  
  # Remove reduced=TRUE or set to FALSE to  
  # download the full dataset.  
  
  path <- downloadUMI4CexampleData(reduced = TRUE)  
}
```

---

`ex_ciita_umi4c`*Contacs with CIITA promoter*

---

**Description**

An example UMI4C object showing the contacts with a viewpoint located at the CIITA gene promoter.

**Usage**

```
ex_ciita_umi4c
```

**Format**

A UMI4C object from this package.

**Source**

See `inst/script/CIITA_process_example.R` to see the code use for generating the UMI4C object.

---

fisherUMI4C

*Differential UMI4C contacts using Fisher's Exact test*


---

**Description**

Using the UMIs inside `query_regions` performs Fisher's Exact test to calculate significant differences between contact intensities.

**Usage**

```
fisherUMI4C(
  umi4c,
  grouping = "condition",
  query_regions,
  resize = NULL,
  window_size = 5000,
  filter_low = 50,
  padj_method = "fdr",
  padj_threshold = 0.05
)
```

**Arguments**

<code>umi4c</code>	UMI4C object as generated by <code>makeUMI4C</code> or the UMI4C constructor.
<code>grouping</code>	Name of the column in <code>colData</code> used to merge the samples or replicates. If none available or want to add new groupings, run <code>addGrouping</code> . Default: "condition".
<code>query_regions</code>	<code>GenomicRanges</code> object or <code>data.frame</code> containing the region coordinates used to perform the differential analysis.
<code>resize</code>	Width in base pairs for resizing the <code>query_regions</code> . Default: no resizing.
<code>window_size</code>	If <code>query_regions</code> are not defined, will bin region in <code>window_size</code> bp and perform the analysis using this windows.
<code>filter_low</code>	Either the minimum median UMIs required to perform Fisher's Exact test or <code>FALSE</code> for performing the test in all windows.
<code>padj_method</code>	Method for adjusting p-values. See <code>p.adjust</code> for the different methods.
<code>padj_threshold</code>	Numeric indicating the adjusted p-value threshold to use to define significant differential contacts.

**Details**

This function calculates the overlap of fragment ends with either the provided `query_regions` or the binned region using `window_size`. The resulting number of UMIs in each `query_region` will be the *sum* of UMIs in all overlapping fragment ends. As a default, will filter out those regions whose median UMIs are lower than `filter_low`.

Finally, a contingency table for each `query_reegions` or window that passed the `filter_low` filter is created as follows:

	<i>query_region</i>	<i>region</i>
<i>Reference</i>	n1	N1-n1
<i>Condition</i>	n2	N2-n2

and the Fisher's Exact test is performed. Obtained p-values are then converted to adjusted p-values using `padj_method` and the results list is added to the UMI4C object.

### Value

Calculates statistical differences between UMI-4C experiments.

### Examples

```
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")

# Perform differential test
enh <- GRanges(c("chr16:10925006-10928900", "chr16:11102721-11103700"))
umi_dif <- fisherUMI4C(ex_ciita_umi4c, query_regions = enh,
                      filter_low = 20, resize = 5e3)
resultsUMI4C(umi_dif)
```

---

formatPlotsUMI4C      *Format plots for UMI4C*

---

### Description

Format plots for UMI4C

### Usage

```
formatPlotsUMI4C(plot_list, font_size)
```

### Arguments

`plot_list`      List of plots generated by [plotUMI4C](#)

`font_size`      Base font size to use for the UMI4C plot. Default: 14.

### Value

Given a named `plot_list` and considering the number and type of included plots, formats their axes accordingly to show the final UMI4C plot.

---

geoMeanCoordinates      *Get geometric mean of given coordinates*

---

**Description**

Get geometric mean of given coordinates

**Usage**

```
geoMeanCoordinates(coords, scale, bait_start)
```

**Arguments**

coords	Vector of integers representing the coordinates from which to obtain the geometric mean.
scale	Vector of scales indicating how many fragment where merged.
bait_start	Integer indicating the coordinates for the bait start.

**Value**

Calculates geometric mean of the provided coordinates, taking into account the distance to the viewpoint and how many restriction fragments are being merged.

---

getColors      *Get default colors*

---

**Description**

Get default colors

**Usage**

```
getColors(factors)
```

**Arguments**

factors	Name of the factors that will be used for grouping variables.
---------	---

**Value**

Depending on the number of factors it creates different color palettes.

---

getFactors *Get factors fro plotting*

---

### Description

Get factors fro plotting

### Usage

```
getFactors(umi4c, grouping = NULL)
```

### Arguments

umi4c	UMI4C object
grouping	Grouping used for the different samples. If none available or want to add new groupings, run <a href="#">addGrouping</a> .

### Value

Factor vector where the first element is the reference factor.

---

getNormalizationMatrix  
*Get normalization matrix*

---

### Description

Will return a normalization matrix.

### Usage

```
getNormalizationMatrix(
  umi4c,
  norm_bins = 10^(3:6),
  post_smooth_win = 50,
  r_expand = 1.2
)
```

### Arguments

umi4c	<a href="#">UMI4C</a> object as generated by <a href="#">makeUMI4C</a> .
norm_bins	Numeric vector with the genomic bins to use for normalization. Default: 1K, 10K, 100K, 1Mb.
post_smooth_win	Numeric indicating the smoothing window to use. Default: 50.
r_expand	Numeric indicanting the expansion value for normalization. Default: 1.2.

### Value

Creates a matrix of normalization factors using as a reference the profile specified in the [UMI4C](#) object.

---

```
getViewpointCoordinates
    Get viewpoint coordinates
```

---

### Description

Finds the viewpoint coordinates for a given reference genome and sequence.

### Usage

```
getViewpointCoordinates(  
    bait_seq,  
    bait_pad,  
    res_enz,  
    ref_gen,  
    sel_seqname = NULL  
)
```

### Arguments

bait_seq	Character containing the bait primer sequence.
bait_pad	Character containing the pad sequence (sequence between the bait primer and the restriction enzyme sequence).
res_enz	Character containing the restriction enzyme sequence.
ref_gen	A BSgenome object of the reference genome.
sel_seqname	A character with the chromosome name to focus the search for the viewpoint sequence.

### Value

Creates a GRanges object containing the genomic position of the viewpoint.

### Examples

```
getViewpointCoordinates(  
    bait_seq = "GGACAAGCTCCCTGCAACTCA",  
    bait_pad = "GGACTTGCA",  
    res_enz = "GATC",  
    ref_gen = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19,  
    sel_seqname = "chr16" # Look only in chr16  
)
```

---

groupSamplesUMI4C      *Group UMI4C samples*

---

**Description**

Combines UMI4C samples by adding UMIs from assay(umi4c) to represent the levels in grouping.

**Usage**

```
groupSamplesUMI4C(umi4c, grouping = "condition")
```

**Arguments**

umi4c	UMI4C object as generated by <code>makeUMI4C</code> .
grouping	Name of the column in colData used to merge the samples or replicates. Set to NULL for skipping grouping. Default: "condition".

**Value**

A grouped UMI4C object.

---

nbinomWaldTestUMI4C      *Differential UMI4C contacts using DESeq2 Wald Test*

---

**Description**

Takes the smooth monotone fit count values and infers the differential UMI4C contacts using DESeq2 Wald Test from DESeq2 package.

**Usage**

```
nbinomWaldTestUMI4C(dds)
```

**Arguments**

dds	DDS object as generated by <code>smoothMonotoneUMI4C</code> with the smooth monotone fit counts
-----	---

**Details**

This function back-transform fitted values to the scale of raw counts and scale them across the samples to obtain normalized factors using `normalizationFactors` from DESeq2 package. To detect differences between conditions, the DESeq2 Wald Test is applied to fitted counts with the normalization factors.

**Value**

DDS object with the DESeq2 Wald Test results, with results columns accessible with the `results` function.

---

plotDifferential      *Plot differential contacts*

---

**Description**

Plot differential contacts

**Usage**

```
plotDifferential(umi4c, grouping = NULL, colors = NULL, xlim = NULL)
```

**Arguments**

umi4c	UMI4C object as generated by <a href="#">makeUMI4C</a> .
grouping	Grouping used for the different samples. If none available or want to add new groupings, run <a href="#">addGrouping</a> .
colors	Named vector of colors to use for plotting for each group.
xlim	Limits for the plot x axis (genomic coordinates).

**Value**

Produces a plot of the fold changes at the differential regions analyzed that are contained in the [UMI4C](#) object.

**Examples**

```
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")

enh <- GRanges(c("chr16:10925006-10928900", "chr16:11102721-11103700"))
umi_dif <- fisherUMI4C(ex_ciita_umi4c, query_regions = enh,
                      filter_low = 20, resize = 5e3)
plotDifferential(umi_dif)
```

---

plotDomainogram      *Plot domainogram*

---

**Description**

Plot domainogram

**Usage**

```
plotDomainogram(
  umi4c,
  dgram_function = "quotient",
  grouping = NULL,
  colors = NULL,
  xlim = NULL
)
```

**Arguments**

umi4c	UMI4C object as generated by <code>makeUMI4C</code> .
dgram_function	Function used for calculating the fold-change in the domainogram plot, either "difference" or "quotient". Default: "quotient".
grouping	Grouping used for the different samples. If none available or want to add new groupings, run <code>addGrouping</code> .
colors	Named vector of colors to use for plotting for each group.
xlim	Limits for the plot x axis (genomic coordinates).

**Value**

Produces the domainogram plot, summarizing the merged number of UMIs at the different scales analyzed (y axis).

**Examples**

```
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")

plotDomainogram(ex_ciita_umi4c, grouping = "condition")
```

---

plotGenes

*Plot genes*

---

**Description**

Plot genes in a window of interest.

**Usage**

```
plotGenes(
  window,
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  longest = TRUE,
  xlim = NULL,
  font_size = 14
)
```

**Arguments**

window	<b>GRanges</b> object with coordinates to use for selecting the genes to plot.
TxDb	TxDb object to use for drawing the genomic annotation.
longest	Logical indicating whether to plot only the longest transcripts for each gene in the gene annotation plot.
xlim	Limits for the plot x axis (genomic coordinates).
font_size	Base font size to use for the UMI4C plot. Default: 14.

**Value**

Produces a plot with the genes found in the provided window.

**Examples**

```
window <- GRanges("chr16:11348649-11349648")
plotGenes(
  window = window,
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
)
```

---

plotTrend

*Plot adaptative smoothen trend*

---

**Description**

Plot adaptative smoothen trend

**Usage**

```
plotTrend(umi4c, grouping = NULL, colors = NULL, xlim = NULL, ylim = NULL)
```

**Arguments**

umi4c	UMI4C object as generated by <a href="#">makeUMI4C</a> .
grouping	Grouping used for the different samples. If none available or want to add new groupings, run <a href="#">addGroupings</a> .
colors	Named vector of colors to use for plotting for each group.
xlim	Limits for the plot x axis (genomic coordinates).
ylim	Limits of the trend y axis.

**Value**

Produces the adaptative trend plot, showing average UMIs at each position taking into account the minimum number of molecules used to merge restriction fragments.

**Examples**

```
data("ex_ciita_umi4c")
plotTrend(ex_ciita_umi4c)
```

plotUMI4C

*Plot UMI4C data***Description**

Produce a UMI-4C data plot containing the genes in the region, the adaptive smoothen trend and the domainogram.

**Usage**

```
plotUMI4C(
  umi4c,
  grouping = "condition",
  dgram_function = "quotient",
  dgram_plot = TRUE,
  colors = NULL,
  xlim = NULL,
  ylim = NULL,
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  longest = TRUE,
  rel_heights = c(0.25, 0.4, 0.12, 0.23),
  font_size = 14
)
```

**Arguments**

umi4c	UMI4C object as generated by <a href="#">makeUMI4C</a> .
grouping	Grouping used for the different samples. If none available or want to add new groupings, run <a href="#">addGrouping</a> .
dgram_function	Function used for calculating the fold-change in the domainogram plot, either "difference" or "quotient". Default: "quotient".
dgram_plot	Logical indicating whether to plot the domainogram. If the UMI4C object only contains one sample will be automatically set to FALSE. Default: TRUE.
colors	Named vector of colors to use for plotting for each group.
xlim	Limits for the plot x axis (genomic coordinates).
ylim	Limits of the trend y axis.
TxDb	TxDb object to use for drawing the genomic annotation.
longest	Logical indicating whether to plot only the longest transcripts for each gene in the gene annotation plot.
rel_heights	Numeric vector of length 3 or 4 (if differential plot) indicating the relative heights of each part of the UMI4C plot.
font_size	Base font size to use for the UMI4C plot. Default: 14.

**Value**

Produces a summary plot with all the information contained in the UMI4C object.

**Examples**

```
data("ex_ciita_umi4c")
ex_ciita_umi4c <- addGrouping(ex_ciita_umi4c, grouping="condition")

plotUMI4C(ex_ciita_umi4c,
  dgram_plot = FALSE
)
```

---

```
prepUMI4C
```

---

```
Prepare UMI4C data
```

---

**Description**

Prepare the FastQ files for the further analysis by selecting reads with bait and adding the respective UMI identifier for each read in its header.

**Usage**

```
prepUMI4C(
  fastq_dir,
  wk_dir,
  file_pattern = NULL,
  bait_seq,
  bait_pad,
  res_enz,
  numb_reads = 1e+11
)
```

**Arguments**

fastq_dir	Path of the directory containing the FastQ files (compressed or uncompressed).
wk_dir	Working directory where to save the outputs generated by the UMI-4c analysis.
file_pattern	Character that can be used to filter the files you want to analyze in the fastq_dir.
bait_seq	Character containing the bait primer sequence.
bait_pad	Character containing the pad sequence (sequence between the bait primer and the restriction enzyme sequence).
res_enz	Character containing the restriction enzyme sequence.
numb_reads	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.

**Value**

Creates a compressed FASTQ file in wk\_dir/prep named `basename(fastq).fq.gz`, containing the filtered reads with the UMI sequence in the header. A log file with the statistics is also generated in wk\_dir/logs named `umi4c_stats.txt`.

**See Also**

[contactsUMI4C](#).

**Examples**

```

if (interactive()) {
  path <- downloadUMI4CexampleData(reduced = TRUE)
  raw_dir <- file.path(path, "CIITA", "fastq")

  prepUMI4C(
    fastq_dir = raw_dir,
    wk_dir = file.path(path, "CIITA"),
    bait_seq = "GGACAAGCTCCCTGCAACTCA",
    bait_pad = "GGACTTGCA",
    res_enz = "GATC"
  )
}

```

---

smoothMonotoneUMI4C     *Monotone smoothing of the DDS object VST counts*

---

**Description**

Takes the variance stabilized count values and calculates a symmetric monotone fit for the distance dependency. The signal trend is fitted using the `fda` package. The position information about the viewpoint have to be stored in the metadata as `metadata(dds)[['bait']]`.

**Usage**

```
smoothMonotoneUMI4C(dds, alpha = 20, penalty = 0.1)
```

**Arguments**

<code>dds</code>	DDS object as generated by <code>vstUMI4C</code> with the variance stabilized count values
<code>alpha</code>	Approximate number of fragments desired for every basis function of the B-spline basis. <code>floor((max(number of fragments)) / alpha)</code> is passed to <code>create.bspline.basis</code> as <code>nbasis</code> argument. 4 is the minimum allowed value. Default: 20.
<code>penalty</code>	Amount of smoothing to be applied to the estimated functional parameter. Default: 0.1.

**Details**

This function computes the smoothing function for the VST values, based on `fda` package, and calculates a symmetric monotone fit counts for the distance dependency

**Value**

DDS object with monotone smoothed fit counts.

---

splitUMI4C	<i>Split UMI4C reads</i>
------------	--------------------------

---

**Description**

Split the prepared reads using the restriction enzyme information.

**Usage**

```
splitUMI4C(wk_dir, res_enz, cut_pos, numb_reads = 1e+11, min_flen = 20)
```

**Arguments**

wk_dir	Working directory where to save the outputs generated by the UMI-4c analysis.
res_enz	Character containing the restriction enzyme sequence.
cut_pos	Numeric indicating the nucleotide position where restriction enzyme cuts (zero-based) (for example, for DpnII is 0).
numb_reads	Number of lines from the FastQ file to load in each loop. If having memory size problems, change it to a smaller number. Default=10e10.
min_flen	Minimal fragment length to use for selecting the fragments. Default=20

**Value**

Creates a compressed FASTQ file in `wk_dir/split` named `basename(fastq).fq.gz`, containing the split reads based on the restriction enzyme used.

**Examples**

```
if (interactive()) {
  path <- downloadUMI4CexampleData(reduced = TRUE)

  splitUMI4C(
    wk_dir = file.path(path, "CIITA"),
    res_enz = "GATC",
    cut_pos = 0
  )
}
```

---

statsUMI4C	<i>Statistics UMI4C</i>
------------	-------------------------

---

**Description**

Creates a stats summary file and generates a summary plot describing statistics for processed UMI-4C samples.

**Usage**

```
statsUMI4C(wk_dir)
```

**Arguments**

`wk_dir` Working directory where to save the outputs generated by the UMI-4c analysis.

**Value**

Returns a plot summarizing the main statistics of the processed UMI-4C experiments found in `wk_dir`. Also, writes a file named `stats_summary.txt` in `wk_dir/logs` that summarizes all the values represented in the plot.

**Examples**

```
statsUMI4C(wk_dir = system.file("extdata", "CIITA",
  package = "UMI4Cats"
))
stats <- read.delim(system.file("extdata", "CIITA", "logs", "stats_summary.txt",
  package = "UMI4Cats"
))
head(stats)
```

---

 theme

*Theme*


---

**Description**

Theme

**Usage**

```
theme(...)
```

**Arguments**

`...` Additional arguments to pass to the theme call from `ggplot2`.

**Value**

`ggplot2` theme.

**Examples**

```
library(ggplot2)

ggplot(
  iris,
  aes(Sepal.Length, Sepal.Width)
) +
  geom_point() +
  theme()
```

---

themeXblank	<i>Theme X blank</i>
-------------	----------------------

---

**Description**

Theme X blank

**Usage**

```
themeXblank(...)
```

**Arguments**

... Additional arguments to pass to the theme call from ggplot2.

**Value**

ggplot2 theme with a blank X axis.

**Examples**

```
library(ggplot2)

ggplot(
  iris,
  aes(Sepal.Length, Sepal.Width)
) +
  geom_point() +
  themeXblank()
```

---

themeXYblank	<i>Theme Y blank</i>
--------------	----------------------

---

**Description**

Theme Y blank

**Usage**

```
themeXYblank(...)
```

**Arguments**

... Additional arguments to pass to the theme call from ggplot2.

**Value**

ggplot2 theme with a blank X and Y axis.

**Examples**

```
library(ggplot2)

ggplot(
  iris,
  aes(Sepal.Length, Sepal.Width)
) +
  geom_point() +
  themeXYblank()
```

---

themeYblank	<i>Theme Y blank</i>
-------------	----------------------

---

**Description**

Theme Y blank

**Usage**

```
themeYblank(...)
```

**Arguments**

... Additional arguments to pass to the theme call from ggplot2.

**Value**

ggplot2 theme with a blank Y axis.

**Examples**

```
library(ggplot2)

ggplot(
  iris,
  aes(Sepal.Length, Sepal.Width)
) +
  geom_point() +
  themeYblank()
```

---

UMI4C	<i>UMI4C-class</i>
-------	--------------------

---

**Description**

The **UMI4C** constructor is the function `makeUMI4C`. By using the arguments listed below, performs the necessary steps to analyze UMI-4C data and summarize it in an object of class **UMI4C**.

**Usage**

```
makeUMI4C(
  colData,
  viewpoint_name = "Unknown",
  grouping = "condition",
  normalized = TRUE,
  ref_umi4c = NULL,
  bait_exclusion = 3000,
  bait_expansion = 1e+06,
  scales = 5:150,
  min_win_factor = 0.02,
  sd = 2
)
```

**Arguments**

<code>colData</code>	Data.frame containing the information for constructing the UMI4C experiment object. Needs to contain the following columns: <ul style="list-style-type: none"> <li>• <code>sampleID</code>. Unique identifier for the sample.</li> <li>• <code>condition</code>. Condition for performing differential analysis. Can be control and treatment, two different cell types, etc.</li> <li>• <code>replicate</code>. Number for identifying replicates.</li> <li>• <code>file</code>. File as outputed by <code>umi4CatsContacts</code> function.</li> </ul>
<code>viewpoint_name</code>	Character indicating the name for the used viewpoint.
<code>grouping</code>	Name of the column in <code>colData</code> used to merge the samples or replicates. Set to <code>NULL</code> for skipping grouping. Default: "condition".
<code>normalized</code>	Logical indicating whether UMI-4C profiles should be normalized to the <code>ref_umi4c</code> sample/group. Default: <code>TRUE</code>
<code>ref_umi4c</code>	Name of the sample or group to use as reference for normalization. By default is <code>NULL</code> , which means it will use the sample with less UMIs in the analyzed region. It should be a named vector, where the name corresponds to the grouping column from <code>colData</code> and the value represents the level to use as reference.
<code>bait_exclusion</code>	Region around the bait (in bp) to be excluded from the analysis. Default: 3000bp.
<code>bait_expansion</code>	Number of bp upstream and downstream of the bait to use for the analysis (region centered in bait). Default: 1Mb.
<code>scales</code>	Numeric vector containing the scales for calculating the domainogram.
<code>min_win_factor</code>	Proportion of UMIs that need to be found in a specific window for adaptative trend calculation
<code>sd</code>	Stantard deviation for adaptative trend.

**Value**

It returns an object of the class [UMI4C](#).

**Slots**

`colData` Data.frame containing the information for constructing the UMI4C experiment object. Needs to contain the following columns:

- `sampleID`: Unique identifier for the sample.

- `condition`: Condition for performing differential analysis. Can be control and treatment, two different cell types, etc.
- `replicate`: Number or ID for identifying different replicates.
- `file`: Path to the files outputted by `contactsUMI4C`.

`rowRanges` `GRanges` object with the coordinates for the restriction fragment ends, their IDs and other additional annotation columns.

`metadata` List containing the following elements:

1. `bait`: `GRanges` object representing the position of the bait used for the analysis.
2. `scales`: Numeric vector containing the scales used for calculating the domainogram.
3. `min_win_factor`: Factor for calculating the minimum molecules required in a window for not merging it with the next one when calculating the adaptative smoothing trend.
4. `grouping`: Columns in `colData` used for the different sample groupings, accessible through `groupsUMI4C`.
5. `normalized`: Logical indicating whether samples/groups are normalized or not.
6. `region`: `GRanges` with the coordinates of the genomic window used for analyzing UMI4C data.
7. `ref_umi4c`: Name of the sample or group used as reference for normalization.

`assays` Matrix where each row represents a restriction fragment site and columns represent each sample or group defined in `grouping`. After running the `makeUMI4C` function, it will contain the following data:

1. `umis`: Raw number of UMIs detected by `contactsUMI4C`.
2. `norm_mat`: Normalization factors for each sample/group and fragment end.
3. `trend`: Adaptative smoothing trend of UMIs.
4. `geo_coords`: Geometric coordinates obtained when performing the adaptative smoothing.
5. `scale`: Scale selected for the adaptative smoothing.
6. `sd`: Stantard deviation for the adaptative smoothing trend.

`dgram` List containing the domainograms for each sample. A domainogram is matrix where columns are different scales selected for merging UMI counts and rows are the restriction fragments.

`groupsUMI4C` List of UMI4C objects with the specific groupings.

`results` List containing the results for the differential analysis ran using `fisherUMI4C`.

## Note

The UMI4C class extends the `SummarizedExperiment` class.

## See Also

`UMI4C-methods`

## Examples

```
# Load sample processed file paths
files <- list.files(system.file("extdata", "CIITA", "count",
  package = "UMI4Cats"
),
pattern = "*_counts.tsv",
full.names = TRUE
)
```

```

# Create colData including all relevant information
colData <- data.frame(
  sampleID = gsub("_counts.tsv.gz", "", basename(files)),
  file = files,
  stringsAsFactors = FALSE
)

library(tidyr)
colData <- colData %>%
  separate(sampleID,
    into = c("condition", "replicate", "viewpoint"),
    remove = FALSE
  )

# Load UMI-4C data and generate UMI4C object
umi <- makeUMI4C(
  colData = colData,
  viewpoint_name = "CIITA",
  grouping = "condition"
)

```

---

UMI4C2dds

*UMI4Cats object to DDS object.*


---

## Description

Transforms an UMI4C object to a DDS object

## Usage

```
UMI4C2dds(umi4c, design = ~condition, query_regions = NULL, ...)
```

## Arguments

umi4c	UMI4C object as generated by makeUMI4C or the UMI4C constructor.
design	A formula or matrix. The formula expresses how the counts for each fragment end depend on the variables in colData. See <a href="#">DESeqDataSet</a> .
query_regions	GRanges object or data.frame containing the coordinates of the genomic regions you want to use to perform the analysis in specific genomic intervals. Default: NULL.
...	Other arguments to be passed to <a href="#">DESeq</a> function.

## Value

DDS object.

## Description

The UMI4Cats package provides functions for the pre-processing, analysis and visualization of UMI-4C chromatin contact data.

## File preparation

There are two different functions that can be used to prepare the files for analyzing them with UMI4Cats:

1. `demultiplexFastq`. Demultiplex reads belonging to different viewpoints from a paired-end FastQ file.
2. `digestGenome`. Digest the reference genome of choice using a given restriction sequence.

## Processing

The pre-processing functions are wrapped in the `contactsUMI4C` main function. This function will sequentially run the following steps:

1. `prepUMI4C`. Filter specific and high quality reads.
2. `splitUMI4C`. Split reads by the restriction sequence.
3. `alignmentUMI4C`. Align reads to the reference genome.
4. `counterUMI4C`. Apply UMI counting algorithm to quantify the interactions with the viewpoint.

The statistics from the samples analyzed with the `contactsUMI4C` function can be extracted and visualized with the function `statsUMI4C`.

## Analysis

The analysis of UMI-4C data is wrapped in the construction of an object of `UMI4C` class by the creator function `makeUMI4C`. This function will group your samples according to the variable you provided in the `grouping` argument (default: "condition") and then normalize it to `ref_umi4c`.

The differential analysis can be performed with `fisherUMI4C` or `differentialNbinomWaldTestUMI4C` functions. Both will return a `UMI4C` object containing the results of the differential test. You can access these results with the method `resultsUMI4C`.

## Visualization

An integrative plot showing the results stored inside the `UMI4C` object can be generated with the function `plotUMI4C`.

---

`vstUMI4C`*Variance stabilizing transformation*

---

**Description**

Using a DDS object performs a variance stabilizing transformation from DESeq2 package to the UMI4C counts

**Usage**

```
vstUMI4C(dds)
```

**Arguments**

`dds` DDS object generated by UMI4C2dds

**Details**

This function estimate the size factors and dispersions of the counts base on [DESeq](#) for infering the VST distribution and transform raw UMI4C counts.

**Value**

DDS object with variance stabilizing transformation counts

# Index

## \* datasets

- ex\_ciita\_umi4c, 22
- .UMI4C (UMI4C), 38
- .getCache, 3
- .getSummaryBam, 3
- .singleAlignmentUMI4C, 4
- .singleCounterUMI4C, 4
- .singlePrepUMI4C, 5
- .singleSplitUMI4C, 6
- .smoothMonotone, 7
  
- addGrouping, 7, 23, 26, 29–32
- addStepping, 8
- alignmentUMI4C, 9, 42
  
- bait (dgram), 17
- bait, UMI4C-method (dgram), 17
  
- calculateAdaptativeTrend, 10
- calculateDomainogram, 10
- contactsUMI4C, 11, 33, 40, 42
- counterUMI4C, 12, 42
- createGeneAnnotation, 14
- createStatsTable, 15
  
- darken, 15
- demultiplexFastq, 16, 42
- DESeq, 41, 43
- deseq2UMI4C, 17
- DESeqDataSet, 17, 19, 41
- dgram, 17
- dgram, UMI4C-method (dgram), 17
- dgram<- (dgram), 17
- dgram<-, UMI4C-method (dgram), 17
- differentialNbinomWaldTestUMI4C, 19, 42
- digestGenome, 11, 13, 21, 42
- downloadUMI4CexampleData, 22
  
- ex\_ciita\_umi4c, 22
  
- fisherUMI4C, 23, 40, 42
- formatPlotsUMI4C, 24
  
- geoMeanCoordinates, 25
- getColors, 25
- getFactors, 26
- getNormalizationMatrix, 26
- getViewpointCoordinates, 27
- GRanges, 8, 14, 19, 30, 40
- groupSamplesUMI4C, 28
- groupsUMI4C (dgram), 17
- groupsUMI4C, UMI4C-method (dgram), 17
- groupsUMI4C<- (dgram), 17
- groupsUMI4C<-, UMI4C-method (dgram), 17
  
- makeUMI4C, 8, 10, 26, 28–32, 38, 40, 42
- makeUMI4C (UMI4C), 38
  
- nbinomWaldTestUMI4C, 28
  
- p.adjust, 17, 20, 23
- plotDifferential, 29
- plotDomainogram, 29
- plotGenes, 30
- plotTrend, 31
- plotUMI4C, 24, 32, 42
- prepUMI4C, 33, 42
  
- resultsUMI4C, 42
- resultsUMI4C (dgram), 17
- resultsUMI4C, UMI4C-method (dgram), 17
  
- smoothMonotoneUMI4C, 34
- splitUMI4C, 35, 42
- statsUMI4C, 35, 42
- SummarizedExperiment, 40
  
- theme, 36
- themeXblank, 37
- themeXYblank, 37
- themeYblank, 38
- trend (dgram), 17
- trend, UMI4C-method (dgram), 17
  
- UMI4C, 8, 10, 26, 28–32, 38, 38, 39, 42
- UMI4C-class (UMI4C), 38
- UMI4C-methods (dgram), 17
- UMI4C2dds, 41
- UMI4Cats, 42
  
- vstUMI4C, 43