Package 'tadar'

November 12, 2026

Title Transcriptome Analysis of Differential Allelic Representation

Version 1.10.0

Description This package provides functions to standardise the analysis of Differential Allelic Representation (DAR). DAR compromises the integrity of Differential Expression analysis results as it can bias expression, influencing the classification of genes (or transcripts) as being differentially expressed. DAR analysis results in an easy-to-interpret value between 0 and 1 for each genetic feature of interest, where 0 represents identical allelic representation and 1 represents complete diversity. This metric can be used to identify features prone to false-positive calls in Differential Expression analysis, and can be leveraged with statistical methods to alleviate the impact of such artefacts on RNA-seq data.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

BugReports https://github.com/baerlachlan/tadar/issues

URL https://github.com/baerlachlan/tadar

BiocType Software

biocViews Sequencing, RNASeq, SNP, GenomicVariation, VariantAnnotation, DifferentialExpression

VignetteBuilder knitr

Depends GenomicRanges, ggplot2, R (>= 4.4.0)

Imports BiocGenerics, Seqinfo, Gviz, IRanges, lifecycle, MatrixGenerics, methods, rlang, Rsamtools, S4Vectors, stats, VariantAnnotation

Suggests BiocStyle, covr, knitr, limma, rmarkdown, testthat (>= 3.0.0), tidyverse

git_url https://git.bioconductor.org/packages/tadar

git_branch RELEASE_3_23

git_last_commit b9700fe

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Lachlan Baer [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5213-3401>),
Stevie Pederson [aut] (ORCID: <https://orcid.org/0000-0001-8197-3303>)

Maintainer Lachlan Baer <baerlachlan@gmail.com>

Contents

tadar-package 2
assignFeatureDar 3
chr1_genes 5
chr1_tt 5
countAlleles 6
countsToProps 6
dar 7
filterLoci 9
flipRanges 10
modP 11
plotChrDar 12
plotDarECDF 15
readGenotypes 16
unphaseGT 17

Index **18**

`tadar-package` *tadar: A package for Differential Allelic Representation (DAR) analysis*

Description

This package enables DAR analysis by providing functions that address discrete steps of the analysis workflow.

DAR analysis is intended to be performed using functions in the following order:

1. `readGenotypes()` parses a multi-sample VCF file and returns a `GRanges` object containing only the data that is required for DAR analysis.
2. `countAlleles()` summarises the alleles from genotype data at each range for each sample group.

3. `filterLoci()` removes ranges that do not match a specified criterion.
4. `countsToProps()` normalises the allele counts to account for missing data and sample groups of different sizes.
5. `dar()` calculates the DAR between two sample groups.
6. `flipRanges()` is an optional step that enables the conversion of ranges output by `dar()` from origins to regions, or vice versa.
7. `assignFeatureDar()` assigns DAR values to features of interest.

`tadar` also provides visualisation functions that allow quick inspection of DAR within the dataset:

- `plotChrDar()` produces a `Gviz` plot that displays the trend in DAR across a chromosome.
- `plotDarECDF()` produces a `ggplot2` figure comparing DAR between chromosomes.

Author(s)

Lachlan Baer, Stevie Pederson

See Also

Useful links:

- https://github.com/baerlachlan/tadar
- Report bugs at https://github.com/baerlachlan/tadar/issues

`assignFeatureDar` *Assign DAR values to genomic features*

Description

Assign DAR values to genomic features of interest by averaging the DAR values of ranges that overlap the feature range.

Usage

```
assignFeatureDar(
  dar,
  features,
  dar_val = c("origin", "region"),
  fill_missing = NA
)

## S4 method for signature 'GRangesList,GRanges'
assignFeatureDar(
  dar,
  features,
  dar_val = c("origin", "region"),
  fill_missing = NA
)
```

Arguments

dar	GRangesList with DAR values of the associated ranges contained in metadata columns. Ranges that represent DAR regions are recommended to assign the greatest number of features with DAR values. This results in an assigned estimate of DAR in the region surrounding the feature. Alternatively, the use of DAR origin ranges results in an assigned average of DAR solely within the feature. Ranges can be converted between origins and regions with flipRanges.
features	GRanges object specifying the features of interest.
dar_val	Deprecated. character(1) specifying the whether to use origin or region DAR values for the chosen ranges. Please use the default "origin" to avoid averaging already averaged values.
fill_missing	The DAR value to assign features with no overlaps. Defaults to NA.

Value

GRangesList with ranges representing features of interest that overlap at least one DAR range. Feature metadata columns are retained and an additional column is added for the assigned DAR value.

Examples

```
data("chr1_genes")
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
counts_filt <- filterLoci(counts)
props <- countsToProps(counts_filt)
contrasts <- matrix(
    data = c(1, -1),
    dimnames = list(
        Levels = c("group1", "group2"),
        Contrasts = c("group1v2")
    )
)
dar <- dar(props, contrasts, region_loci = 5)
assignFeatureDar(dar, chr1_genes)

dar_regions <- flipRanges(dar, extend_edges = TRUE)
assignFeatureDar(dar_regions, chr1_genes)
```

chr1_genes *Genomic feature example data*

Description

Gene features for example usage. Generation of this data is documented in `system.file("data-raw/chr1_genes.R", package = "tadar")`.

Usage

```
data(chr1_genes)
```

Format

An object of class `GRanges` of length 1456.

Value

GRanges object with 1456 ranges and 2 metadata columns.

chr1_genes Ranges represent gene features for chromosome 1 of zebrafish GRCz11 genome.

Source

https://www.ensembl.org

chr1_tt *Differential expression example data*

Description

p-values from differential expressiong testing for example usage.

Usage

```
data(chr1_tt)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 716 rows and 5 columns.

Value

A 716 x 5 tibble object.

countAlleles *Count alleles within each experimental group*

Description

Summarise the alleles from genotype calls at each single nucleotide locus within each sample group.

Usage

```
countAlleles(genotypes, groups)

## S4 method for signature 'GRanges,list'
countAlleles(genotypes, groups)
```

Arguments

genotypes	GRanges object with metadata columns containing genotype information for all samples.
groups	Named list specifying the sample grouping structure, where each element contains a character vector of sample names.

Value

GRangesList containing a summary of allele counts at each range. Each element of the list represents a distinct sample group.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
countAlleles(genotypes, groups)
```

countsToProps *Convert allele counts to proportions*

Description

Normalise allele-level counts across samples by converting to a proportion of total alleles in all samples.

Usage

```
countsToProps(counts)

## S4 method for signature 'GRangesList'
countsToProps(counts)
```

Arguments

counts	GRangesList containing a summary of allele counts at each range. Each element of the list represents a distinct sample group.

Value

GRangesList containing a summary of normalised allele counts (i.e. as proportions) at each range. Each element of the list represents a distinct sample group.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
counts_filt <- filterLoci(counts)
countsToProps(counts_filt)
```

---

dar *Calculate Differential Allelic Representation (DAR)*

Description

Calculate DAR between two sample groups.

Usage

```
dar(props, contrasts, region_fixed = NULL, region_loci = NULL)

## S4 method for signature 'GRangesList,matrix'
dar(props, contrasts, region_fixed = NULL, region_loci = NULL)
```

Arguments

`props`	`GRangesList` containing a summary of normalised allele counts (i.e. as proportions) at each range. Each element of the list represents a distinct sample group.
`contrasts`	Contrast `matrix` specifying which sample groups to to calculate DAR between. Each column must represent a single contrast, and rows represent the levels (i.e. sample groups) to be contrasted. The two levels involved with each contrast should be specified with `1` and `-1`.
`region_fixed`	`integer(1)` specifying the width (in base pairs) of a fixed sliding window used for averaging DAR values within a region, which is centralised around the origin. Must be an integer greater than 1. This argument takes precedence over `region_loci`.
`region_loci`	`integer(1)` specifying the number of loci to include in an elastic sliding window used for averaging DAR values within a region. Must be an odd integer in order to incorporate the origin locus and an equal number of loci either side. Only used when `region_fixed` is NULL.

Details

DAR is calculated as the Euclidean distance between the allelic proportions (i.e. proportion of As, Cs, Gs and Ts) of two sample groups at a single nucleotide locus, scaled such that all values range inclusively between 0 and 1. A DAR value of 0 represents identical allelic representation between the two sample groups, while a DAR value of 1 represents complete diversity.

Value

`GRangesList` containing DAR values at each overlapping range between the contrasted sample groups. Two types of DAR values are reported in the metadata columns of each GRanges object:

- `dar_origin`: The raw DAR values calculated at single nucleotide positions (the origin) between sample groups. These values represent DAR estimates at a precise locus.
- `dar_region`: The mean of raw DAR values in a specified region surrounding the origin. This is optionally returned using either of the `region_fixed` or `region_loci` arguments, which control the strategy and size for establishing regions (more information below). This option exists because eQTLs don't necessarily confer their effects on genes in close proximity. Therefore, DAR estimates that are representative of regions may be more suitable for downstream assignment DAR values to genomic features.

Each element of the list represents a single contrast defined in the input contrast matrix.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
```

```
counts_filt <- filterLoci(counts)
props <- countsToProps(counts_filt)
contrasts <- matrix(
    data = c(1, -1),
    dimnames = list(
        Levels = c("group1", "group2"),
        Contrasts = c("group1v2")
    )
)
dar(props, contrasts, region_loci = 5)
```

`filterLoci` *Filter loci*

Description

Filter loci based on allele count criteria.

Usage

```
filterLoci(counts, filter = n_called > n_missing)

## S4 method for signature 'GRangesList'
filterLoci(counts, filter = n_called > n_missing)
```

Arguments

`counts`	`GRangesList` containing a summary of allele counts at each range. Each element of the list represents a distinct sample group.
`filter`	A logical expression indicating which rows to keep. Possible values include: • `n_called` The number of total alleles called. • `n_missing` The number of total alleles not reported. • `n_0`, `n_1`, `n_2`, `n_3` The number of ref, alt1, alt2 and alt3 alleles respectively. All values represent the sum of counts across all samples within the group. Defaults to return loci where the number of samples containing allele information is greater than number samples with missing information.

Value

`GRangesList` containing a summary of allele counts at each range passing the filter criteria. Each element of the list represents a distinct sample group.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
filterLoci(counts)
```

`flipRanges` *Convert DAR origin ranges to DAR region ranges, or vice versa*

Description

Convert the ranges element associated with origin DAR values to ranges associated with the region DAR values. This function can also be used to revert back to the original object containing origin ranges if desired.

Usage

```
flipRanges(dar, extend_edges = FALSE)

## S4 method for signature 'GRangesList'
flipRanges(dar, extend_edges = FALSE)
```

Arguments

`dar`	`GRangesList` with ranges representing single nucleotide (origin) positions.
`extend_edges`	`logical(1)` specifying if region DAR ranges at the edges of each chromosome should be extended to cover the entire chromosome when converting from origin ranges to region ranges. This argument is only considered if `region_loci` was used to construct regions in the `dar()` function. Useful for downstream assignment of DAR values to genomic features that exist at the 5' or 3' edges of the chromosome, which would have otherwise been missed.

Value

`GRangesList` with ranges that represent either DAR regions or DAR origins, depending on the ranges of the input object.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
counts_filt <- filterLoci(counts)
props <- countsToProps(counts_filt)
contrasts <- matrix(
    data = c(1, -1),
    dimnames = list(
        Levels = c("group1", "group2"),
        Contrasts = c("group1v2")
    )
)

## Establish regions using an elastic sliding window
dar <- dar(props, contrasts, region_loci = 5)
## Convert ranges to regions associated with dar_region values
dar_regions <- flipRanges(dar)
## Optionally extend the outer regions to completely cover chromosomes
dar_regions <- flipRanges(dar, extend_edges = TRUE)
## Convert back to origin ranges associated with dar_origin values
flipRanges(dar_regions)

## Establish regions using a fixed sliding window
dar <- dar(props, contrasts, region_fixed = 1001)
## Convert ranges to regions associated with dar_region values
dar_regions <- flipRanges(dar)
## Convert back to origin ranges associated with dar_origin values
flipRanges(dar_regions)
```

---

modP *DAR-moderated* p-*values*

Description

Moderate *p*-values from DE testing using assigned DAR values.

Usage

```
modP(pvals, dar, slope = -1.8)

## S4 method for signature 'numeric,numeric'
modP(pvals, dar, slope = -1.8)
```

Arguments

pvals	numeric of feature-level *p*-values from differential expression testing.
dar	numeric of DAR values assigned to corresponding features tested for differential expression.
slope	numeric(1) specifying the slope of alpha fit.

Value

numeric of DAR-moderated *p*-values of same length as input *p*-values.

Examples

```
data("chr1_genes")
data("chr1_tt")
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
counts_filt <- filterLoci(counts)
props <- countsToProps(counts_filt)
contrasts <- matrix(
    data = c(1, -1),
    dimnames = list(
        Levels = c("group1", "group2"),
        Contrasts = c("group1v2")
    )
)
dar <- dar(props, contrasts, region_loci = 5)
dar_regions <- flipRanges(dar, extend_edges = TRUE)
gene_dar <- assignFeatureDar(dar_regions, chr1_genes, dar_val = "region")
chr1_tt <- merge(chr1_tt, mcols(gene_dar$group1v2), sort = FALSE)
chr1_tt$darP <- modP(chr1_tt$PValue, chr1_tt$dar)
```

---

plotChrDar *Plot DAR across a chromosome*

Description

Use Gviz to plot the trend in DAR across a chromosomal region with the option to add features of interest as separate tracks.

Usage

```
plotChrDar(
  dar,
  dar_val = c("origin", "region"),
  chr,
  foi,
  foi_anno,
  foi_highlight = TRUE,
  features,
  features_anno,
  features_highlight = TRUE,
  title = ""
)

## S4 method for signature 'GRanges'
plotChrDar(
  dar,
  dar_val = c("origin", "region"),
  chr,
  foi,
  foi_anno,
  foi_highlight = TRUE,
  features,
  features_anno,
  features_highlight = TRUE,
  title = ""
)
```

Arguments

dar	GRanges with DAR values of associated ranges contained in metadata columns. Used to build the DataTrack showing the trend in DAR across the chromosome. If ranges of the input object span regions (i.e. post application of flipRanges()), data points are plotted at the midpoint of the region.
dar_val	character(1) specifying the whether to use origin or region DAR values for the chosen ranges. Options are "origin" and "region". The default ("region") represents averaged DAR values across a region and produces a smoother graph.
chr	Optional. character(1) specifying the chromosome to subset all GRanges objects. Plotting is only possible across a single chromosome and is therefore required if supplying GRanges objects spanning multiple chromosomes. Also controls the track title for the GenomeAxisTrack.
foi	Optional. GRanges object of features of interest (foi) to be highlighted and labelled along the GenomeAxisTrack.
foi_anno	Optional. character(1) specifying the name of the mcol containing labels associated with feature ranges in foi.
foi_highlight	logical(1) specifying if the positions of foi should be overlayed on the DataTrack showing the trend in DAR. Useful for visually inspecting DAR at the

location of chosen features. Default is TRUE.

features — Optional. GRanges object of features to be plotted on a separate AnnotationTrack.

features_anno — Optional. character(1) specifying the name of the mcol containing labels associated with feature ranges in features.

features_highlight — logical(1) specifying if the positions of features should be overlayed on the DataTrack showing the trend in DAR. Useful for visually inspecting DAR at the location of chosen features. Default is TRUE.

title — character(1). The main plot title.

Value

A Gviz object

Examples

```
set.seed(230822)
data("chr1_genes")
foi <- sample(chr1_genes, 1)
features <- sample(chr1_genes, 20)
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
genotypes <- readGenotypes(fl)
groups <- list(
    group1 = paste0("sample", 1:6),
    group2 = paste0("sample", 7:13)
)
counts <- countAlleles(genotypes, groups)
counts_filt <- filterLoci(counts)
props <- countsToProps(counts_filt)
contrasts <- matrix(
    data = c(1, -1),
    dimnames = list(
        Levels = c("group1", "group2"),
        Contrasts = c("group1v2")
    )
)
dar <- dar(props, contrasts, region_loci = 5)
plotChrDar(
    dar = dar$group1v2, dar_val = "region", chr = "1",
    foi = foi, foi_anno = "gene_name", foi_highlight = TRUE,
    features = features, features_anno = "gene_name",
    features_highlight = TRUE,
    title = "Example plot of DAR along Chromosome 1"
)
```

plotDarECDF *Plot the Empirical Cumulative Distribution Function of DAR*

Description

Plot the ECDF of DAR for each chromosome.

Usage

```
plotDarECDF(dar, dar_val = c("origin", "region"), highlight = NULL)

## S4 method for signature 'GRanges'
plotDarECDF(dar, dar_val = c("origin", "region"), highlight = NULL)
```

Arguments

dar	GRanges object with metadata columns containing the desired DAR values to plot.
dar_val	character(1) specifying the whether to plot dar_origin or dar_region values. Options are "origin" and "region".
highlight	character(1) specifying the chromosome to highlight with a different colour.

Value

A ggplot2 object.

Examples

```
set.seed(230704)
## Use simulated data that illustrates a commonly encountered scenario
simulate_dar <- function(n, mean) {
    vapply(
        rnorm(n = n, mean = mean),
        function(x) exp(x) / (1 + exp(x)),
        numeric(1)
    )
}
gr <- GRanges(
    paste0(rep(seq(1,25), each = 100), ":", seq(1,100)),
    dar_origin = c(simulate_dar(2400, -2), simulate_dar(100, 0.5))
)
## No highlighting, all chromosomes will be given individual colours
plotDarECDF(gr, dar_val = "origin") +
    theme_bw()

## With highlighting
plotDarECDF(gr, dar_val = "origin", highlight = "25") +
    scale_colour_manual(values = c("TRUE" = "red", "FALSE" = "grey")) +
```

```
theme_bw()
```

`readGenotypes` *Read genotypes from a VCF file*

Description

Extract genotypes from a VCF file into a GRanges object for downstream DAR analysis.

Usage

```
readGenotypes(file, unphase = TRUE, ...)

## S4 method for signature 'character'
readGenotypes(file, unphase = TRUE, ...)

## S4 method for signature 'TabixFile'
readGenotypes(file, unphase = TRUE, ...)
```

Arguments

`file`	The file path of a VCF file containing genotype data. Alternatively, a `TabixFile` as described in readVcf.
`unphase`	A `logical` specifying if phasing information should be removed from genotypes. This is required if proceeding with DAR analysis.
`...`	Passed to readVcf.

Details

Extract genotypes from a VCF file with the option to remove phasing information for DAR analysis.

Value

A `GRanges` object constructed from the CHROM, POS, ID and REF fields of the supplied `VCF` file. Genotype data for each sample present in the `VCF` file is added to the metadata columns.

Examples

```
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
readGenotypes(fl)
```

unphaseGT *Unphase genotypes*

Description

Remove phasing information from genotype calls.

Usage

```
unphaseGT(gt)

## S4 method for signature 'matrix'
unphaseGT(gt)

## S4 method for signature 'data.frame'
unphaseGT(gt)
```

Arguments

gt	matrix or data.frame containing sample genotype information.

Details

Phasing information is not required for a simple DAR analysis. Removing this enables easy counting of alleles from genotype calls.

Value

matrix containing unphased genotype calls.

Examples

```
library(VariantAnnotation)
fl <- system.file("extdata", "chr1.vcf.bgz", package="tadar")
vcf <- readVcf(fl)
gt <- geno(vcf)$GT
unphaseGT(gt)
```

Index

∗ **datasets**
chr1_genes, 5
chr1_tt, 5
∗ **internal**
tadar-package, 2

assignFeatureDar, 3
assignFeatureDar(), *3*
assignFeatureDar,GRangesList,GRanges-method *(*assignFeatureDar*)*, 3

chr1_genes, 5
chr1_tt, 5
countAlleles, 6
countAlleles(), *2*
countAlleles,GRanges,list-method *(*countAlleles*)*, 6
countsToProps, 6
countsToProps(), *3*
countsToProps,GRangesList-method *(*countsToProps*)*, 6

dar, 7
dar(), *3*, *10*
dar,GRangesList,matrix-method *(*dar*)*, 7

filterLoci, 9
filterLoci(), *3*
filterLoci,GRangesList-method *(*filterLoci*)*, 9
flipRanges, *4*, 10
flipRanges(), *3*, *13*
flipRanges,GRangesList-method *(*flipRanges*)*, 10

modP, 11
modP,numeric,numeric-method *(*modP*)*, 11

plotChrDar, 12
plotChrDar(), *3*
plotChrDar,GRanges-method *(*plotChrDar*)*, 12
plotDarECDF, 15
plotDarECDF(), *3*
plotDarECDF,GRanges-method *(*plotDarECDF*)*, 15

readGenotypes, 16
readGenotypes(), *2*
readGenotypes,character-method *(*readGenotypes*)*, 16
readGenotypes,TabixFile-method *(*readGenotypes*)*, 16
readVcf, *16*

tadar *(*tadar-package*)*, 2
tadar-package, 2

unphaseGT, 17
unphaseGT,data.frame-method *(*unphaseGT*)*, 17
unphaseGT,matrix-method *(*unphaseGT*)*, 17