Package ‘skitools’

April 13, 2017

Title Various mskilab R utilties

Version 0.0.0.9000

Maintainer Marcin Imielinski <mimielinski@nygenome.org>

Author Marcin Imielinski <mimielinski@nygenome.org>

Description R miscellaneous utilities for basic data manipulation, debugging, viz, lsf management, and common mskilab tasks.

Imports gplots,
hwriter,
plotrix,
S4Vectors,
RColorBrewer,
methods,
reshape2,
IRanges,
tools,
Biostrings,
DT,
GenomeInfoDb

Depends R (>= 3.1.0),
GenomicRanges,
VariantAnnotation,
htmlwidgets,
devtools,
gUtils,
stringr,
plotly,
ggplot2,
data.table,
reshape2,

Suggests testthat,
parallel,
BSgenome.Hsapiens.UCSC.hg19

License MIT

LazyData true

RoxygenNote 6.0.1

R topics documented:

`.igv_host` *.igv_host*

Description

several settings for igv host

Usage

```
.igv_host(h)
```

Author(s)

Marcin Imielinski

`af` *af*

Description

computes prob density of af over a set of n mutation calls using alt allele count, total counts, cn, and provided grid.size

as per landau, carter et al

altc, totc, and cn are of length n, purity is length 1

Usage

```
af(altc, totc, grid.size = 0.01, verbose = F)
```

Author(s)

Marcin Imielinski

af2 *af2*

Description

computes 2D probability density af over a set of n mutation counts two samples, output columns correspond to x and rows correspond to y and dimnames correspond to amounts

as per landau, carter et al

altc1, totc1, altc2, totc2 are of length n

Usage

```
af2(altc1, totc1, altc2, totc2, grid.size = 0.01, verbose = F,
  animate = NA)
```

Author(s)

Marcin Imielinski

allele_multiplicity *allele_multiplicity*

Description

Given individuals file (with either Purity value or a bsp_participant_id or Tumor_scan_name columns) and maf file annotated with absolut copy number (maf$cn.tot) and column $patient for firehose id.

You can provide maf without cn.tot annotations latter case will pull absolute segs using bsp_participant_id in ind and annotate, but may take a few minutes to pull.

Usage

```
allele_multiplicity(ind, maf, abs.seg = NULL, verbose = TRUE)
```

Author(s)

Marcin Imielinski

`bisort` *bisort*

Description

"bisorts" matrix according to rows and columns (and optionally removes empty rows, ie with no nonzero)

Usage

```
bisort(A, drop = F)
```

Arguments

`A`	matrix to sort
`drop`	logical flag whether to drop empty rows (=FALSE)

Author(s)

Marcin Imielinski

`border` *border*

Description

orders rows of a logical / binary matrix treating each row as binary number with digits encoded as TRUE / FALSE values of entries

Usage

```
border(B, na.rm = TRUE)
```

Arguments

`B`	input matrix logical format, or castable to logical
`na.rm`	removes NA

Value

B resorted using "binary" order

`brewer.master` *brewer.master*

Description

Makes a lot of brewer colors using an "inexhaustible" brewer palette ie will not complain if number of colors requested is too high.

Yes - this technically violates the "grammar of graphics", but meant for quick and dirty use.

Usage

```
brewer.master(n, palette = "Accent")
```

Arguments

`n` TODO

`palette` character specifyign pallette to start with (options are: Blues, BuGn, BuPu, GnBu, Greens Greys, Oranges, OrRd, PuBu, PuBuGn, PuRd, Purples, RdPu, Reds, YlFn, YlFnBu, YlOrBr, YlOrRd, BrBg, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral, Accent, Dark2, Paired, Pastel1, Pastel2, Set2, Set3)

Value

length(n) character vector of colors

Author(s)

Marcin Imielinski

`bsub_cmd` *bsub_cmd*

Description

Makes bsub command that wraps shell command "cmd" to send to queue "queue" redirebmccting output / error etc streams to path prefixed by "jname", optional_args: maximum memory requirements "mem", "jlabel" job label b

Usage

```
bsub_cmd(cmd, queue, jname = NULL, jlabel = NULL, jgroup = NULL,
  mem = NULL, group = "cgafolk", cwd = NULL, mc.cores = NULL,
  deadline = F)
```

Arguments

`cmd`	length n vector of shell commands, optionally named, one per job
`queue`	optional length n or length 1 character specifying queue to send jobs to (default hour)
`jname`	optional length n character specifying names of jobs, this will be the root of the output files generated by the job
`jlabel`	optional length n character specifying labels of jobs, this the string
`jgroup`	optional length n character specifying job group name
`mem`	length n or length 1 integer specifying GB of memory to be used by jobs
`group`	character specifying job group (default cgafolk)
`cwd`	character specifying which working directory to launch jobs from (default is current working directory of R session)
`mc.cores`	length n or 1 integer specifying how many cores to assign to each job
`deadline`	logical flag whether to send jobs to deadline queue

Value

character vector of bsub commands, which can run using system or dumped to a shell script

Author(s)

Marcin Imielinski

camerplot *cameraplot*

Description

plots the results of CAMERA in limma package

Usage

```
cameraplot(camera.res, gene.sets, voom.res, design, contrast = ncol(design),
  title = "Camera Gene set notch plot", cex.space = 1,
  col.axis = alpha("gray20", 0.8), col.ramp = c("blue", "red"),
  cex.slab = 1, cex.glab = 1, lwd.notch = 1, tick.w = 0.1,
  max.genes = 10, text.shift = 0.5, height.wf = 0.1, min.corr = 0.1,
  min.dist = 10, max.gene.sets = 20, gtext.shift = 0.2)
```

Arguments

`camera.res`	output of camera from limma
`gene.sets`	gene set input to camera (named list of indices into the voom.res gene expression matrix)
`voom.res`	output of voom from limma
`design`	design matrix input to camera
`cex.space`	label spacing expansion factor (use if labels get too crowded

`col.axis` axis color character

`col.ramp` ramp from lowest to highest expression to phenotype correlation (default blue, red)

`cex.slab` set label cex

`cex.glab` gene label cex

`lwd.notch` notch thickness

`max.genes` max genes to draw in "leading edge" of gene set

`text.shift` amount to shift text from notches (>0, <1)

`height.wf` height of the topmost correlation waterfall plot

`min.corr` minimal abs(correlation) value for leading edge definition

`text.shift` minimal distance between labels

Author(s)

Marcin Imielinski

`capitalize` *capitalize*

Description

Capitalize first letter of each character element of vector "string"

Usage

```
capitalize(string, un = FALSE)
```

Arguments

`string` character vector to capitalize

`un` logical flag whether to uncapitalize (=FALSE)

Value

character vector of strings with capitalized values

`ccf` *ccf*

Description

computes fuzzy histogram of ccf across a set of n mutation calls using alt allele count, total counts, cn, and purity, and provided grid.size

as per landau, carter et al

altc, totc, and cn are of length n, purity is length

Usage

```
ccf(altc, totc, cn, purity, grid.size = 0.01, verbose = F)
```

Author(s)

Marcin Imielinski

`charToDec` *charToDec*

Description

converts character vector to byte vector in decimal representation

Usage

```
charToDec(c)
```

Arguments

`c`	character vector

Value

length(c) integer vector of byte representation of c

Author(s)

Marcin Imielinski

`chr2num` *Convert from chrXX to numeric format*

Description

Convert from chrXX to numeric format

Usage

```
chr2num(x, xy = FALSE)
```

Arguments

`x`	factor, Rle or character vector with chromosome names
`xy`	Flag to convert M to 25, Y to 24 and X to 23. Default FALSE

Value

character vector with xy=FALSE, or numeric vector with xy=TRUE

`chron` *chron*

Description

Repeat a command periodically, e.g. every 10 seconds

Usage

```
chron(expr, period = 5)
```

`chunk` *chunk*

Description

takes same input as seq (from, to, by, length.out) and outputs a 2 column matrix of indices corresponding to "chunks"

Usage

```
chunk(from, to = NULL, by = 1, length.out = NULL)
```

Arguments

`from`	integer where to begin sequence
`to`	integer to end sequence
`by`	interval to space sequence
`length.out`	number of desired chunks, i.e. nrows of output matrix

Value

2 column matrix of indices, each row representing chunk

Author(s)

Marcin Imielinski

`clock` *clock times expression*

Usage

```
clock(expr)
```

Arguments

`expr`	R code to eval while suppressing all errors

Author(s)

Marcin Imielinski

`clone_cluster` *clone_cluster*

Description

determines "clone membership" using CCF threshold for n variants, given alt read count, total read count, and purity

first fits k component mixture model (k pre-specified) to CCF histogram .. mixture model can also be given as input returns cluster centers and membership probabilities for each mutation

$mu k vector of means $sigma k vector of sigma $p.cluster n x k matrix of cluster probabilities $lambda k vector of cluster membership frequencies return n x k matrix of probabilities that CCF>ccf.thresh for each variant

altc, totc, and cn are of length n, purity is length 1

Usage

```
clone_cluster(altc, totc, cn, purity, thresh = 0.95, k = 2,
  mix.model = NULL, grid.size = 0.01, verbose = F, nsamp = 10000)
```

Author(s)

Marcin Imielinski

`col.slice` *col.slice*

Description

Hacked operator for subsetting columns of data.frames, DataFrames, data.tables, GRanges using a vector of regexps

df

Usage

```
df %!% ...
```

Arguments

`df`	data.frame
`regex`	string to match or number in which case that column is returned (same behavior for data.table)

Value

slices of data.frame matching regex

Author(s)

Marcin Imielinski

`coloredData` *coloredData*

Description

S4 class for data with colors used by heatmap.plus

simple object with data (e.g. vector or matrix of categorical, real numbers) + a colormap

colormap is a (named) vector mapping factor levels / unique values in data into colors, or otherwise assigning a color range to numeric data.

Instantiate coloredData

Usage

```
coloredData(data, colormap, upright = T)
```

Author(s)

Marcin Imielinski

dcast2 *dcast.data.table but allows vector arguments for value.var,*

Description

if value.var is a vector then will combine the right hand side column names with each element of value.var in a merged cast table

Usage

```
dcast2(data, formula, ..., value.var = NULL, fun.aggregate = function(x) if
  (length(x) <= 1) x[1] else paste(x, collapse = ","), sep = "_")
```

Author(s)

Marcin Imielinski

ddd *ddd*

Description

shortcut to gr2dt

Usage

```
ddd(x)
```

dedup *dedup*

Description

relabels duplicates in a character vector with .1, .2, .3 (where "." can be replaced by any user specified suffix)

relabels duplicates in a character vector with .1, .2, .3 (where "." can be replaced by any user specified suffix)

Usage

```
dedup(x, suffix = ".")

dedup(x, suffix = ".")
```

Arguments

`x`	input vector to dedup
`suffix`	suffix separator to use before adding integer for dups in x
`x`	input vector to dedup
`suffix`	suffix separator to use before adding integer for dups in x

Value

length(x) vector of input + suffix separator + integer for dups and no suffix for "originals"

length(x) vector of input + suffix separator + integer for dups and no suffix for "originals"

Author(s)

Marcin Imielinski

Marcin Imielinski

`dev.all.off` *dev.all.off*

Description

kills all plot devices

Usage

```
dev.all.off()
```

Author(s)

Marcin Imielinski

`dfstring` *dfstring*

Description

"tuple" style chraacter representation of a table, key name1 = value1, name2 = value2 either as a single line or many lines useful for quick eyeballing of tabular data

Usage

```
dfstring(df, oneline = TRUE, binary = FALSE, sep1 = "; ", sep2 = ", ")
```

Arguments

`df`	data.frame input
`oneline`	logical flag whether to print on one line (=TRUE)
`sep1`	first level separator (=;) i.e. between rows
`sep2`	second level separator (=,) i.e. between columns

Value

character vector of string representation

Author(s)

Marcin Imielinski

`dirr` *dirr*

Description

a variant of dir that gsubs pattern from normal output of dir to name output vector

eg dirr(path, '.txt') will return dir output with .txt removed eg dirr(path, '.txt', '.rds') will return dir output with .txt subbed with .rds

Usage

```
dirr(x, pattern = NULL, rep = "", full = TRUE, ...)
```

Arguments

`x`	character of path to run dir on
`pattern`	character pattern to limit files to and to replace with rep
`rep`	character pattern to replace filenames with
`full`	whether to return full path
`...`	additional arguments to dir

Value

named vector of file paths, named by file names in dir gsub-stripped with pattern

Author(s)

Marcin Imielinski

`discordant.pairs` *Label discordant read pairs*

Description

Labels read pairs discordant based on whether they have (1) ++ or – strand orientation (2) "-" strand read start is not greater than dmin or less than dmin ahead of "+" strand read on same chromosome

Usage

```
discordant.pairs(pairs, inter.only = F, dmin = 50, dmax = 500)
```

Note

need to merge with gr.isdisc

`dmix` *dmix*

Description

generates data frame of density points in a provided range for a provided mix of k densities of a singlen family useful for plugging into downstream plotting (eg ggplot 2)

"..." variables depend on density function, arguments should be provided as they would to the corresponding R function (ie with respect to vectorization)

if collapse = TRUE then the density will be summed according to the mixing parameter yielding a single density (ie a fuzzy histogram) summarizing the mixing distribution

Usage

```
dmix(dens = "dnorm", xlim = NULL, n = 500, alpha = NULL, plot = F,
  fill = T, collapse = F, ...)
```

Arguments

`dens`	character specifying R density function, the possibilities include (with ... arguments shown alongside the density names) dnorm: mean, sd dbinom: size, prob dmultinom: size, prob dgamma: shape, rate dbeta: shape1, shape2
`xlim`	length 2 vector specifying plot bounds (=NULL)
`n`	integer number of points to draw distribution over (=500)
`alpha`	length(k) numeric vector specifying mixing probability
`plot`	logical flag specifying whether to draw the plot (=FALSE)
`fill`	logical flag specifying whether to fill the colored plots (=TRUE)
`collapse`	collapse logical flag whether to collapse the mixture components into a single mixture)
`...`	additional density specific arguments each with vectorized values of length k, where k is the number of desired mixture componetns, see dens arugment)

Value

if plot == TRUE then ggplot2 object of plot, otherwise data.frame of data points with fields $id specifying thee mixture id, $x = data value,

Author(s)

Marcin Imielinski

dplot *dplot*

Description

Plots dotplot of grouped data

Usage

```
dplot(y, group, ylab = "", xlab = "", log = F, dotsize = NULL,
  binwidth = 0.02, title = NULL, ylim = NULL, text.size = NULL)
```

Arguments

y	numeric vector of data
group	length(y) vector of catageories
ylab	y axis label (=”)
xlab	x axis label (=”)
log	logical flag whether to plot y axis in log format (=FALSE)
dotsize	integer dot size to plot with, as function of 0.02 category width plot real estate (= NULL)
binwidth	numeric binwidth of histogram in units of data quantiles (= NULL)
title	character title of plot (=”)
ylim	y limits of plot (= NULL)
text.size	text size of legend (= NULL)

Author(s)

Marcin Imielinski

dtgr *Convert data.table to GRanges*

Description

Takes as input a data.table which must have the fields: start, end, strand, seqnames. All of the remaining fields are added as meta data to the GRanges

Usage

```
dtgr(dt)
```

Arguments

dt	data.table to convert to GRanges

Value

GRanges object of length = nrow(dt)

Examples

```
## Not run: r <- dtgr(data.table(start=1, seqnames="X", end=2, strand='+'))
```

elcycles *elcycles*

Description

enumerates all elementary cycles in a graph via igraph library

A is either an adjacency matrix or igraph object

Usage

```
elcycles(A)
```

Arguments

A	adjacency matrix

Value

list with fields: $cycles = list of vertices in elementary cycles $cycles.eix = list of edges in elementary cycles, where edges are numbered according to the 1D index of adj matrix A

Author(s)

Marcin Imielinski

file.dir *file.dir*

Description

grabs file.dirs from list of paths

Usage

```
file.dir(paths)
```

Arguments

paths	character vector of full paths

Value

character vector of just file.names

Author(s)

Marcin Imielinski

`file.name` *file.name*

Description

parses filenames from character vector of paths

Usage

```
file.name(paths)
```

Arguments

`paths`	character vector of full paths

Value

character vector of just file names

Author(s)

Marcin Imielinski

`fisher.combined` *fisher.combined*

Description

Computes fisher combined p value for a matrix of p values where the columns correspond to individual (independent) tests rows correspond to hypotheses.

Usage

```
fisher.combined(Ps)
```

Arguments

`Ps`	n x k matrix of p values from k different independent tests

Value

length n numeric vector of p values

Author(s)

Marcin Imielinski

`fisher.pairwise` *fisher.pairwise*

Description

Performs fisher test on cols of matrix / df x vs cols of matrix / df y

returns list with ncol(x) by ncol(y) matrices $p and $or denoting the p value and odds ratio of the result of the fisher test on col i of x and col j of y

If y is not provided, will correlate rows of x with themselves.

Usage

```
fisher.pairwise(x, y = x)
```

Arguments

`x`	n x k1 data frame of categorical data on k1 variables
`y`	n x k2 optional data frame of categorical data on k2 variables (= x)

Value

list with field $p and $or correspodning to k1 x k2 matrices of p values and odds ratios for each pair of tests

Author(s)

Marcin Imielinski

`fisher.plot` *Plots fisher contingency table with p value*

Description

Plots fisher contingency table with p value

Usage

```
fisher.plot(O)
```

Arguments

`O`	observed matrix of counts

`fready` *fread with name cleaning*

Description

calls fread while cleaning names using provided or default pattern and sub

Usage

```
fready(..., pattern = "\\W+", sub = "_")
```

Arguments

`pattern`	character (default \W)
`sub`	character to sub in names (default _)

Value

data.table

`fuckr` *fuckr*

Description

... what you feel when R is getting on your nerves. Toggles options(error =) to enable / disable debugging mode.

toggles options error recover / NULL

Usage

```
fuckr()
```

Author(s)

Marcin Imielinski

`func_code` *func_code*

Description

Produces (simple) R code calling function named "func" with args in list "argv", prepending with source() call to directories in the vector "sources" if specified.

NOTE: args in ... can be lists or vectors consisting of numerical values or characters. Lists can have named fields. These will be assigned in a "hard coded" way in the Rcode, so these should be ideally scalars or pretty short vectors / lists.

For code to run properly, the names of "argv" must correspond to argument names of "func", or if the list has unnamed fields then they must be ordered in the order of the function args.

Useful for dumping tmp code files for farming when there are many arguments being passed

Usage

```
func_code(func, sources = c(), ...)
```

Arguments

`func`	function to call
`sources`	files to source
`...`	additional arguments to function which should contain numerical or character vectors or lists of such vectors

Value

character data of source file containing call to function with arguments hard coded

Author(s)

Marcin Imielinski

`gatk_callvariants` *gatk_callvariants*

Description

Outputs variant calling pipeline on a set of whole genome (or whole exome) bams to shell script and prints instructions on how to execute from command line.

Usage

```
gatk_callvariants(bams, outdir = "./", run = gsub("[^\\w+]", "_",
  as.character(Sys.time()), perl = T), queue = "hour", memlimit = 2,
  sep = "/", skipclean = T, chunk = 250000, runtype = "wg",
  dry = FALSE, dcov = 200, bsub = FALSE, vqsr.memlimit = memlimit)
```

Arguments

`bams`	bams to run variant calling on
`outdir`	out directory to output to
`run`	name to give run (=timestamp)
`queue`	LSF queue to run on (='week')
`memlimit`	memory limit in GB
`skipclean`	logical flag whether to skipclean (=TRUE)
`chunk`	integer chunk of variants to run each section on (=250e3)
`runtype`	character run type (='wg')
`dry`	logical flag whether to do dry run (=FALSE)
`dcov`	dcov parameter to GATK (=200)
`bsub`	whether to run on LSF vs local (=FALSE)

Author(s)

Marcin Imielinski

`gatk_haplotypecaller` *gatk_haplotypecaller*

Description

calls haplotype caller on a set of input bams and a given set of targets, outputting to target.dir intervals are given as GRanges

Usage

```
gatk_haplotypecaller(outdir, bams, intervals = NULL,
  dbsnp = Sys.getenv("GATK.DBSNP"), hg = Sys.getenv("GATK.FASTA"),
  genome = "hg19", outroot = "out", stand_call_conf = 30,
  stand_emit_conf = 2, min_mapq = 20, other.args = "", run = F,
  verbose = T, write_bam = F, oncotate = T)
```

Arguments

`outdir`	out directory to output to
`bams`	bams to run variant calling on
`intervals`	GRanges intervals to run haplotype caller on
`dbsnp`	dbSNP path (=Sys.getenv('GATK.DBSNP'))
`hg`	genome fasta location (=Sys.getenv('GATK.FASTA'))
`genome`	name of genome build (='hg19')
`outroot`	prefix to give output files (='out')
`stand_call_conf`	confidence for calls (=30)

`stand_emit_conf`
confience for emission (=30)

`min_mapq` minimum mapping quality (=20)

`run` logical flag whether to run immediately or just return character vector of command (= FALSE)

`verbose` logical flag (=TRUE)

`write_bam` logical flag whether to write the bam (=FALSE)

`oncotate` logical flag whether to oncotate output into MAF files (=TRUE)

`other.arg` other arguments to give to haplotype caller (=”)

Value

character vector of command(s) (only if run = FALSE), otherwise just runs command with system call.

Author(s)

Marcin Imielinski

gatk_oncotate *gatk_oncotate*

Description

Makes shell script to oncotate variants outputted from GATK UG run in directory "gatk.dir", outputs instructions how to run the script.

Usage

```
gatk_oncotate(gatk.dir, jname = "gatk.oncotate", mem = 3, queue = "week",
  ...)
```

Arguments

`gatk.dir` output directory containing GATK UG output

`jname` job name to run jobs with

`mem` max memory in GB (=3)

`queue` queue to run on

Author(s)

Marcin Imielinski

`gc_content` *Get GC content from reference genome*

Description

Uses BSgenome package to compute gc content for a collection of segments in seg data frame ($chr, $start, $end or $chr, $pos1, $pos2 or $chr, $begin, $end) Returns vector of gc content of length nrow(segs).

Usage

```
gc_content(segs, bs_genome)
```

Arguments

`segs`	Segment data frame to pull gc from
`bs_genome`	A `BSgenome` object. Perhaps `BSgenome.Hsapiens.UCSC.hg19::Hsapiens`

`gr.all` *gr.all*

Description

Return a GRanges that holds interavals for all of HG19

Usage

```
gr.all(unmap = FALSE, M = FALSE, Y = TRUE)
```

Arguments

`unmap`	[default F] Optinally add a "unmapped" chr
`M`	[default F] Include mitochondrial chr
`Y`	[default T] Include Y chr

Value

`GRanges` object with one element per chromosome

gr.isclip *Check if reads are clipped*

Description

Returns a logical vector of length of the input GRanges that that classifies a read as clipped or not. The user can specify a cutoff value for how many bases need to be clipped.

Usage

```
gr.isclip(gr, clip.cutoff = 10)
```

Arguments

`gr`	Granges OR data.table that has `cigar` field and `qname` field
`clip.cutoff`	Minimum number of bases that are clipped to call the reads as clipped

Value

logical of length of input, denoting whether that read is part of a clipped read pair.

gr.isdisc *Checks if reads are discordant*

Description

Returns a logical vector denoting if a read is discordant. There is only a minimum absolute isize, and any read below this is not considered discordant. This will return logicals based on read pairs

Usage

```
gr.isdisc(gr, isize = 1000, unmap.only = FALSE)
```

Arguments

`gr`	Granges OR data.table that has `isize` field and `qname` field
`isize`	Minimum insert size required to call dis<cordant. Default 1000
`unmap.only`	Find only pairs with an unmapped read

Value

logical vector of length of input, denoting each read as discordant or not

`gr.mincov` *Return windows with minimal coverage*

Description

Takes a set of GRanges and removes any ranges that don't have a minimal coverage value. If you give it a GRangesList, you will get back an unlisted GRanges.

Usage

```
gr.mincov(gr, min.cov = 2, buffer = 0, ignore.strand = TRUE,
  pintersect = FALSE)
```

Arguments

`gr`	`GRanges` to filter
`min.cov`	Minimum number of overlaps to keep. Default 2
`buffer`	Add a buffer to the ranges when computing overlaps. Default 0
`ignore.strand`	Ignore the strand when comparing overlaps. Default TRUE
`pintersect`	Force the pintersect option for gr.findoverlaps

Value

GRanges

`gr.peaks` *Find peaks in a* `GRanges` *over a given meta-data field*

Description

Finds "peaks" in an input GRanges with value field y. first piles up ranges according to field score (default = 1 for each range) then finds peaks. If peel > 0, then recursively peels segments contributing to top peak, and recomputes nextpeak "peel" times if peel>0, bootstrap controls whether to bootstrap peak interval nbootstrap times if id.field is not NULL will peel off with respect to unique (sample) id of segment and not purely according to width if FUN preovided then will complex aggregating function of piled up values in dijoint intervals prior to computing "coverage" (FUN must take in a single argument and return a scalar) if id.field is not NULL, AGG.FUN is a second fun to aggregate values from id.field to output interval

Usage

```
gr.peaks(gr, field = "score", minima = FALSE, peel = 0, id.field = NULL,
  bootstrap = TRUE, na.rm = TRUE, pbootstrap = 0.95, nbootstrap = 10000,
  FUN = NULL, AGG.FUN = sum, peel.gr = NULL, score.only = FALSE,
  verbose = peel > 0)
```

Arguments

`gr` GRanges with some meta-data field to find peaks on

`field` character field specifying metadata to find peaks on, default "score, can be NULL in which case the count is computed

`minima` logical flag whether to find minima or maxima

`id.field` character denoting field whose values specifyx individual tracks (e.g. samples)

`bootstrap` logical flag specifying whether to bootstrap "peel off" to statistically determine peak boundaries

`na.rm` remove NA from data

`pbootstrap` quantile of bootstrap boundaries to include in the robust peak boundary estimate (i.e. essentially specifies confidence interval)

`FUN` function to apply to compute score for a single individual

`AGG.FUN` function to aggregate scores across individuals

`nboostrap` number of bootstraps to run

Examples

```
## outputs example gene rich hotspots from example_genes GRanges
pk = gr.peaks(example_genes)

## now add a numeric quantity to example_genes and compute
## peaks with respect to a numeric scores, e.g. "exon_density"
example_genes$exon_density = example_genes$exonCount / width(example_genes)
pk = gr.peaks(example_genes, field = 'exon_density')

## can quickly find out what genes lie in the top peaks by agggregating back with
## original example_genes
pk[1:10] %$% example_genes[, 'name']
```

`gr.readfilter` *Filter reads by average PHRED score Defines a cutoff score for the mean PHRED quality of a read in a GRanges.*

Description

Filter reads by average PHRED score Defines a cutoff score for the mean PHRED quality of a read in a GRanges.

Usage

```
gr.readfilter(gr, cutoff = "+")
```

Arguments

`gr` GRanges or data.table of reads that has a `qname` and `qual` field

`cutoff` cutoff score for mean PHRED quality. Default "+"

Value

GRanges or data.table where reads have mean quality score >= cutoff

---

`gr.reduce` *Minimal overlaps for GRanges/GRangesList*

Description

Takes any number of GRanges or GRangesList and reduces them to the minimal set of overlapping windows, ignoring strand (optional). Can also collapse only within levels of a meta data field "by"

Usage

```
gr.reduce(..., by = NULL, ignore.strand = TRUE, span = FALSE)
```

Arguments

`...`	`GRanges` or `GRangesList`

Details

Will populate output with metadata of first row of input contributing the reduced output range.

Value

GRanges

`gr.refactor` *gr.refactor*

Description

Takes a pile of ranges gr and new seqnames "sn" (either of length 1 or of length(gr)) and returns a gr object with the new seqnames and same widths and new start coordinates. These coordinates are determined by placing each gr on the corresponding chromosome at 1 + gap after previous gr (or at 1)

Usage

```
gr.refactor(gr, sn, gap = 0, rev = FALSE)
```

Arguments

`gr`	`GRanges` to refactor
`sn`	character vector of new seqnames
`gap`	Default 0
`rev`	Default FALSE

`gr.round` *Round a set of GRanges to another set "rounds" a set of query ranges Q to a given interval set S using the following rule: 1) If q in Q is partially / fully within S then return intersection of q with S. 2) If q intersects multiple ranges in S and `up = F` then return the "first" range, otherwise the last range 3) If q in Q is fully outside of S (ie fully inside not S) then return the `start-1` (if `up = T`) or `end+1` (if `up = F`) of the matching range in not S*

Description

Round a set of GRanges to another set "rounds" a set of query ranges Q to a given interval set S using the following rule: 1) If q in Q is partially / fully within S then return intersection of q with S. 2) If q intersects multiple ranges in S and `up = F` then return the "first" range, otherwise the last range 3) If q in Q is fully outside of S (ie fully inside not S) then return the `start-1` (if `up = T`) or `end+1` (if `up = F`) of the matching range in not S

Usage

```
gr.round(Q, S, up = TRUE, parallel = FALSE)
```

Arguments

`Q` Query `GRanges` (strand is ignored)

`S` Subject `GRanges` (strand is ignored)

`up` [default TRUE] See description.

`parallel` [default FALSE] If `TRUE`, assumes Q and S are same length and this analysis is only performed between the corresponding Q and S pairs.

Value

Rounded `GRanges`

Examples

```
## Not run: query  <- GRanges(1, IRanges(c(100,110),width=201), seqinfo=Seqinfo("1", 500))
subject <- GRanges(1, IRanges(c(160,170),width=201), seqinfo=Seqinfo("1", 500))
gr.round(query, subject)
## End(Not run)
```

`gr.setdiff` *More robust and faster implementation of GenomicRangs::setdiff*

Description

Robust to common edge cases of setdiff(gr1, gr2) where gr2 ranges are contained inside gr1's (yieldings setdiffs yield two output ranges for some of the input gr1 intervals.

Arguments

`query`	`GRanges` object as query
`subject`	`GRanges` object as subject
`max.slice`	Default Inf. If query is bigger than this, chunk into smaller on different cores
`verbose`	Default FALSE
`mc.cores`	Default 1. Only works if exceeded max.slice
`...`	arguments to be passed to gr.findoverlaps

Value

returns indices of query in subject or NA if none found

`gr.tostring` *gr.tostring*

Description

dumps out a quick text representation of a gr object (ie a character vector)

Usage

```
gr.tostring(gr, places = 2, interval = 1e+06, unit = "MB",
  prefix = "chr")
```

Arguments

`gr`	`GRanges`
`places`	Number of decimal places. Default 2
`interval`	Default 1e6
`unit`	Default "MB"
`prefix`	Default "chr"

Value

text representation of input

gr2gatk *Dump GRanges to GATK file*

Description

Dumps gr object into gatk intervals in file path "file"

Usage

```
gr2gatk(gr, file, add.chr = F)
```

Arguments

`gr`	GRanges
`file`	file
`add.chr`	Flag to add "chr" to seqnames. Default FALSE

Value

returns 0 if completed

gr2grl *gr2grl Quick way to make grl from list of indices into a GRanges gr*

Description

gr2grl Quick way to make grl from list of indices into a GRanges gr

Usage

```
gr2grl(gr, ix)
```

Arguments

`gr`	`GRanges` to split
`ix`	vector to split on

grl.filter *Filters GRangesList to only include ranges in the specified window*

Description

(this is different from

Usage

```
grl.filter(grl, windows)
```

Details

does not return list in necessarily same order

grl.span *grl.span*

Description

Returns GRanges object representing the left / right extent of each GRL item. In case of "chimeric" GRL items (ie that map to two chromosomes) there are two options: (1) specify "chr" chromosome as argument to subset GRL's that are on that chromosome, and compute GRL extents from this, any GRL full outside of that chromosome will get a 0 width GRL (2) (default) allow chimeric GRL items to get an extent that is with respect to the first chromosome in that GRL

Usage

```
grl.span(grl, chr = NULL, ir = FALSE, keep.strand = TRUE)
```

Arguments

`grl`	GRangesList to query
`chr`	[Default NULL]
`ir`	[Default FALSE]
`keep.strand`	[Default TRUE]

Details

If a grl item contains ranges that lie on different chromosomes, then corresponding grange will have chromosome "NA" and IRange(0, 0)

grl.split *grl.split*

Description

splits GRL's with respect to their seqnames and strand (default), returning new grl whose items only contain ranges with a single seqname / strand

Usage

```
grl.split(grl, seqname = TRUE, strand = TRUE, values = c())
```

Arguments

`grl`	`GRangesList` to split
`seqname`	Default TRUE
`strand`	Default TRUE
`values`	columns of values field in grl

Details

can also split by arbitrary (specified) genomic ranges value fields

`gsea_leading_edge` *gsea_leading_edge*

Description

Draws gsea plot for an input gene set and outputs leading edge

Usage

```
gsea_leading_edge(gene.set, sig, draw.plot = T, cex = 1, asp = 2,
  eps = 1e-16, name = "")
```

Author(s)

Marcin Imielinski

`gstring` *gstring*

Description

quick function to parse gr from character vector IGV / UCSC style strings of format gr1;gr2;gr3 where each gr is of format chr:start-end[+/-]

Usage

```
gstring(...)
```

`headf` *headf*

Description

"head" +/- grep vector of files

Usage

```
headf(x, n = 5, grep = NULL)
```

Arguments

`x`	vector of iles
`grep`	string to grep in files (=NULL)

Author(s)

Marcin Imielinski

heatmap.plus *heatmap.plus*

Description

Additional features: allows several label tracks on top, bottom, left, and right, with separate top and bottom legend frames tohouse each allows use of coloredData in tracks

Usage

```
heatmap.plus(x, Rowv = NULL, Colv = if (symm) "Rowv" else NULL,
  bar = FALSE, show.rdend = TRUE, show.cdend = TRUE, topColAttr = NULL,
  bottomColAttr = NULL, leftRowAttr = NULL, rightRowAttr = NULL,
  leg.args = NULL, dim.heatmap = c(4, 4), distfun = dist,
  hclustfun = hclust, reorderfun = function(d, w) reorder(d, w), add.expr,
  symm = FALSE, revC = identical(Colv, "Rowv"), scale = c("row", "column",
  "none"), na.rm = TRUE, margins = c(1, 1), cexRow = 0.2 + 1/log10(nr),
  cexCol = 0.2 + 1/log10(nc), labRow = NULL, add.grid = F,
  col.grid = "gray", lwd.grid = 1, size.legend.panel = 0.4,
  size.feature.panel = 0.2, col = topo.colors(100), optimal.leaf = T,
  return.clust = F, labCol = NULL, main = NULL, xlab = NULL,
  ylab = NULL, keep.dendro = FALSE, las.col = 2,
  verbose = getOption("verbose"), ...)
```

hets *hets*

Description

generates allele fraction at all possible hets at sites specified by vcf (eg hapmap) input for tumor and normal

Usage

```
hets(tum.bam, norm.bam = NULL, out.file,
  vcf.file = "/cga/meyerson/home/marcin/DB/dbSNP/hapmap_3.3.b37.vcf",
  chunk.size1 = 1000, chunk.size2 = 100, mc.cores = 1, verbose = T,
  na.rm = TRUE, filt.norm = T)
```

html_link *html_link*

Description

returns text with html link

Usage

```
html_link(href, text = NULL)
```

Arguments

href	character vector of paths to link
text	text to display

Value

character vector of html link text

Author(s)

Marcin Imielinski

html_tag *html_tag*

Description

makes a open and close html tag with optional text inside and optional (named) vector of name=value pairs contained inside of opening tag

Usage

```
html_tag(tag, text = NULL, collapse = " ", ...)
```

Arguments

tag	character vector of tags (without brackets)
text	text to put inside tags
collapse	how to collapse tags (=newline)

Value

character vector of html

Author(s)

Marcin Imielinski

igv *igv*

Description

Controls IGV on localhost (or specified host, separate from where R session is running). Igv application must be running and listening to a specified port. Then if you configure this port via environment variables (IGV_HOST, IGV_PORT) in the current R session then you can apply the following usages

igv(fn) ## sends any given file(s) into igv (eg .bam, .wig, .bed) igv(loci = cool.loci) ## plots the windows specified as GRanges or IGV-parsable strings (eg gene name) igv(gr = cool.gr) ## sends granges object to igv session, Note: currently requires the ability to write to a public_html that is web viewable by computer running IGV igv(snapshot = fn) ## sends current screen to file igv(new = TRUE) ## refreshes current session igv(reset = TRUE) ## resets connections, sometimes useful if IGV not responding

If alternate file paths are present on server (where R is runinng) and computer running IGV, then can specify gsub.paths variable which is a list of length 2 vectors specifying how to convert file paths from arguments given to igv to ones that can be loaded locally.

Usage

```
igv(paths = NULL, gr = NULL, loci = NULL, snapshot = NULL,
  track.view = NULL, new = FALSE, reset = FALSE, wkspace = "PanLungWGS",
  host = Sys.getenv("IGV_HOST"), mac = !grepl("(^cga)|(node\\d+)", host),
  rawpaths = FALSE, sort.locus = NULL, gsub.paths = list(),
  port = Sys.getenv("IGV_PORT"))
```

Arguments

`paths` file paths to display on current igv session (=NULL)

`gr` GRanges or GRAngesList of numeric genomic data or interval genomic annotations to send to IGV session, if gr has field $score then data will be dumped to .bw otherwise to .bed or .gff (=NULL)

`loci` GRanges or IGV parsable string specifying what window(s) on genome to view (=NULL)

`snapshot` file path to store snapshot in (has to be interpretable on file system where IGV is running)

`track.view` command for setting the track display mode ("expand","squish" or "collapse")

`new` logical flag whether to start new IGV session

`reset` logical flag whether to reset connection between R and IGV (useful if IGV non responsive)

`host` character specifying host where IGV is running

`mac` logical flag specifying whether host is a local "mac" (ie then apply gsub.paths) otherwise keep paths the same

`gsub.paths` list of length 2 vectors specifying gsub args to apply to filenames when mac = TRUE

`port` integer specifying port where IGV is running

Author(s)

Marcin Imielinski

`igv.loci` *igv.loci Wrapper for igv function to dump table + screenshots for individual GRanges loci that have sample column (default Tumor_Sample_Barcode) that is a key into ind data.table where columns matching col.string are fetched and plotted IGV host and port are taken from environment variables*

Description

Wrapper for igv function to visualize a given (single) mutation in a maf file

Usage

```
igv.loci(mut, ind, out.path, sample.key = "Tumor_Sample_Barcode",
  sleep = 30, window = 400, host = Sys.getenv("IGV_HOST"),
  port = Sys.getenv("IGV_PORT"), overwrite = FALSE, snapshots = TRUE,
  verbose = TRUE)
```

Author(s)

Marcin Imielinski

`img.html` *img.html*

Description

takes img.paths and dumps out html with imgs +/- names

can be dumped into a file for showing many images into a single page alternative to img_link for "embedding images"

Usage

```
img.html(paths, text = names(paths), height = 1024, width = 768,
  header = 1)
```

Arguments

`paths`	vector of (relative) paths to embed in html
`text`	optional text label to put above embedded images (default = names(paths))

Value

character vector of img tags

Author(s)

Marcin Imielinski

img_link *img_link*

Description

Returns vector of html image links to files "file" with text "caption"

if embed = T, then will make img link, and additional arguments may be supplied to image tag (eg height, width)

Usage

```
img_link(file, caption = NULL, embed = F, ...)
```

Arguments

`file`	vector of (relative) image file paths to link to
`caption`	character vector of captions to add (= ”)
`embed`	logical flag whether to imbed images instead of returning links (=FALSE)
`...`	additional parameters to embed in tag (e.g. height and width)

Value

character vector of links (<a> tags) or image tags (<img> or <embed) to dump into an html document

Author(s)

Marcin Imielinski

import.ucsc *import.ucsc*

Description

wrapper around rtracklayer import that (1) handles "as" formats (2) has additional flag chrsub to sub in 'chr' in selection, and then sub it out of the output

Usage

```
import.ucsc(con, selection = NULL, text, chrsub = TRUE, verbose = FALSE,
  as = NULL, ...)
```

`ind2sub` *ind2sub*

Description

MATLAB style ind2sub function in R physical essence. Provides the 2D row / column index for a given 1D query

Usage

```
ind2sub(dim, ind, byrow = F)
```

Arguments

`dim`	dimensions of matrix to query
`ind`	1D index
`byrow`	whether to calculate indices by row or column (= FALSE)

Value

length(ind) x 2 matrix of row and column index pairs corresponding to input ind in dim "dim" matrix

Author(s)

Marcin Imielinski

`install.packages.bioc` *install.packages.bioc*

Description

shortcut to install bioconductor packages

Usage

```
install.packages.bioc(pkg)
```

Arguments

`pkg`	character vector of package names to install

Author(s)

Marcin Imielinski

`install.packages.github` *install.packages.github*

Description

shortcut to install github packages

Usage

```
install.packages.github(pkg, username, branch)
```

Arguments

`pkg`	character vector of package names to install

Author(s)

Marcin Imielinski

`is.dup` *is.dup*

Description

labels which vectors in x are part of a dup returns logical TRUE if vector is part of a dup

Note: this is a twist on "duplicated" which only returns TRUE if a given element is a duplicate (i.e. duplicated () is FALSE for the original version for the duplicate, while is.dup() will be TRUE for that element)

x can be vector or matrix

Usage

```
is.dup(x)
```

Arguments

`x`	vector or matrix to check

Value

logical vector of length(x) or nrow(x)

Author(s)

Marcin Imielinski

`ldim` *ldim*

Description

returns dimensions of all objects contained in list

Usage

```
ldim(l)
```

Arguments

`l`	list

Author(s)

Marcin Imielinski

`levapply` *levapply*

Description

Applies FUN locally to levels of x and returns vector of length() (eg can do a "local" order within levels)

Usage

```
levapply(x, by, FUN = "order")
```

Arguments

`x`	input vector of data
`by`	length(x) vector of categorical labels
`FUN`	function that takes a length k vector and outputs a length k vector, used for processing each "level" of by

Value

length(x) vector of outputs, the results of applying FUN to each "by" defined level of x

Author(s)

Marcin Imielinski

`list.expr` *list.expr*

Description

Takes a character or numeric vector and makes an expression for re-creating that character in source code

Usage

```
list.expr(x)
```

Arguments

`x`	input vector

Value

character vector of command to create the input vector

Author(s)

Marcin Imielinski

`loud` *loud*

Description

Runs a system command but prints a message with the output

Usage

```
loud(x)
```

`maf2vcf` *maf2vcf*

Description

Dumps maf data frame to VCF file "fn"

Usage

```
maf2vcf(maf, fn)
```

Arguments

`maf`	maf data.frame
`fn`	output file

Author(s)

Marcin Imielinski

`mafcount` *mafcount*

Description

Returns base counts for reference and alternative allele for an input tum and norm bam and maf data frame or GRAnges specifying substitutions

Usage

```
mafcount(tum.bam, norm.bam = NULL, maf, chunk.size = 100, verbose = T,
  mc.cores = 1, ...)
```

Details

maf is a single width GRanges describing variants and field 'ref' (or 'Reference_Allele'), 'alt' (or 'Tum_Seq_Allele1') specifying reference and alt allele. maf is assumed to have width 1 and strand is ignored.

`maflite` *maflite*

Description

take maf data frame and returns columns corresponding to "maflite" format https://confluence.broadinstitute.org/display/C
Mafliteformat

Usage

```
maflite(maf)
```

Arguments

`maf`	maf.data.frame

Value

data.frame in maf.lite format

Author(s)

Marcin Imielinski

`maf_classify` *maf_classify*

Description

Re-classifies oncotated variants

Usage

```
maf_classify(maf)
```

Arguments

`maf`	maf data.frame

Value

variant categories

Author(s)

Marcin Imielinski

`maf_coding` *maf_coding*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is in protein coding region (includes synonymous)

Usage

```
maf_coding(maf, inclusive = T)
```

Arguments

`maf`	maf data.frame

Value

logical vector specifying whether row satisfies the criterion

Author(s)

Marcin Imielinski

maf_disp *maf_disp*

Description

Quick display of rows of data.frame holding contents of Oncotated maf file

Usage

```
maf_disp(maf, flavor = NULL, sorted = F, show.pat = TRUE,
  extra_cols = NULL, gene = NULL, pat = NULL)
```

Arguments

maf	data.frame with Oncotated maf columns
flavor	character specifying 'functional' or 'validation' flavors, which correspond to special column slices of maf data.frame (= NULL)
sorted	logical flag whether to return output sorted by gene, variant classificaiton, uniprot site, then patient (=FALSE)
show.pat	logical flag whether to show patient (=TRUE)
gene	character vector of Hugo_Symbol to subset on (=NULL)
pat	character vector of Tumor_Sample_Barcodes to subset on (=NULL)
extra_col	character vector of additional columns to include (=NULL)

Value

character vector or sliced data.frame

Author(s)

Marcin Imielinski

maf_exonic *maf_exonic*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is exonic

Usage

```
maf_exonic(maf, inclusive = T)
```

Arguments

maf	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

`maf_genic` *maf_genic*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is genic

Usage

```
maf_genic(maf, inclusive = T)
```

Arguments

`maf`	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

`maf_indel` *maf_indel*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is a indel

Usage

```
maf_indel(maf, inclusive = T)
```

Arguments

`maf`	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

maf_nonyn *maf_nonyn*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is non-synonymous

Usage

```
maf_nonsyn(maf, inclusive = T)
```

Arguments

maf	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

maf_ns *maf_ns*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is non-synonymous

Usage

```
maf_ns(maf, inclusive = T)
```

Arguments

maf	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

maf_sub *maf_sub*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is a substitution

Usage

```
maf_sub(maf, inclusive = T)
```

Arguments

maf	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

maf_syn *maf_syn*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is synonymous

Usage

```
maf_syn(maf, inclusive = T)
```

Arguments

maf	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

maf_to_simple *maf_to_simple*

Description

Dumps maf files to "simple" format for input into oncotator, adds dummy ref_allele and tum_allele1 cols if not provided

Usage

```
maf_to_simple(maf, filename, genome)
```

Arguments

`maf`	maf data.frame
`filename`	output file
`genome`	An BSgenome object (was "build genome build (='hg19')")

Author(s)

Marcin Imielinski

maf_truncating *maf_truncating*

Description

Scans "Variant_Classification" field in maf and outputs TRUE if variant is truncating

Usage

```
maf_truncating(maf, inclusive = T)
```

Arguments

`maf`	maf data.frame

Value

logical vector specifying whether maf row satisfies the criterion

Author(s)

Marcin Imielinski

match.bs — *Identify matches between query and dictionary*

Description

Wrapper around matchPdict to identify matches between a query string query and dictionary dict (both BString objects or subclasses)

Usage

```
match.bs(query, dict, midpoint = FALSE)
```

Arguments

`query`	Query
`dict`	Dictionary
`midpoint`	Flag for output the coordinates of the match as the location, where the midpoint of the dict string matches the given query. Default FALSE

Value

a vector of indices of length width(query) that contains indices of the (starting) dictionary in the query string

mmatch — *mmatch*

Description

match rows of matrix A to matrix B

Usage

```
mmatch(A, B, dir = 1)
```

Arguments

`A`	query matrix k1 x n
`B`	subject matrix k2 x n
`dir`	1

Value

length k1 vector specifying first row of B matching row i of A

Author(s)

Marcin Imielinski

modix *modix*

Description

Takes integer input ix and projects on to 1-based modulus over base l

ie modix(1, 5) -> 1, modix(5, 5) -> 5, modix(6, 5) -> 1

Takes integer input ix and projects on to 1-based modulus over base l

ie modix(1, 5) -> 1, modix(5, 5) -> 5, modix(6, 5) -> 1

Usage

```
modix(ix, l)
```

```
modix(ix, l)
```

Arguments

`ix`	input indices to apply module
`l`	base of ix
`ix`	input indices to apply module
`l`	base of ix

Value

((ix-1) mod l) - 1

((ix-1) mod l) - 1

Author(s)

Marcin Imielinski

Marcin Imielinski

morder *morder*

Description

matrix order wrt columns .. ie ordering rows matrix based on left to right ordering of columns (if MARGIN = 1) OR ordering columns of matrix based on top to bottom ordering of rows (if MARGIN = 2)

Usage

```
morder(A, orient = 1)
```

Arguments

`A`	matrix of values
`orient`	integer orientation, if 1 will do row-wise ordering, otherwise column ordering (=1)

Value

input matrix with rows and columns ordered

Author(s)

Marcin Imielinski

`more` *more*

Description

"more" +/- grep vector of files

Usage

```
more(x, grep = NULL)
```

Arguments

`x`	vector of iles
`grep`	string to grep in files (=NULL)

Author(s)

Marcin Imielinski

`mtable` *mtable*

Description

tabulates unique rows values for matrix / data frame

Usage

```
mtable(mat)
```

Arguments

`mat`	input matrix

Value

unique rows of mat, with additional column $count on the left

Author(s)

Marcin Imielinski

muffle *muffle Runs code returning NULL is there is any error*

Usage

```
muffle(code, ...)
```

Arguments

`code`	R code to eval while suppressing all errors
`...`	additional tryCatch arguments

Value

output of evaluated R code or NULL if error

Author(s)

Marcin Imielinski

munlist *munlist*

Description

unlists a list of vectors, matrices, data frames into a n x k matrix whose first column specifies the list item index of the entry and second column specifies the sublist item index of the entry and the remaining columns specifies the value(s) of the vector or matrices.

force.cbind = T will force concatenation via 'cbind' force.rbind = T will force concatenation via 'rbind'

Usage

```
munlist(x, force.rbind = F, force.cbind = F, force.list = F)
```

Arguments

`x`	list of vectors, matrices, or data frames
`force.rbind`	logical flag to force concatenation via rbind (=FALSE), otherwise will guess
`force.cbind`	logical flag to force concatenation via cbind (=FALSE), otherwise will guess
`force.list`	logical flag to force concatenation via unlist (=FALSE), otherwise will guess

Value

data.frame of concatenated input data with additional fields $ix and $iix specifying the list item and within-list index from which the given row originated from

Author(s)

Marcin Imielinski9

mutclusters	*mutclusters*

Description

Returns genes with a degree of mutaiton clustering (e.g. ranked by how many k>k.thresh clusters with d<d.thresh pairwise distance, or the largest cluster with those characteristics)

if max.cluster = TRUE returns maximum size cluster in gene where either all (method == complete) or at least one (method single) mutation pair is within distance d

if max.cluster = F then returns number of clusters of mutations of count greater than k within a distance d per gene

eg d = 0, k = 1, will give the number of unique sites with more than 1 perfectly recurrent mutation per gene

Clustering is by default using single-linkage agglomerative clustering, but any method that is input to hclust can be used

Usage

```
mutclusters(maf, d = 0, k = 1, method = "single", max.cluster = TRUE)
```

Arguments

`maf`	maf data.frame
`d`	max distance threshold in amino acid space
`k`	minimum number of mutations in returned clusters
`method`	character specifying "single" or "complete" linkage clustering of mutations
`max.cluster`	logical flag whether to return the gene with the largest cluster (if TRUE) or the most number of clusters (if FALSE) (=TRUE)

Value

genes ranked by numbers of cluster or max.cluster size

Author(s)

Marcin Imielinski

mutpairsd *mutpairsd*

Description

Takes maf data.frame and outputs table that lists how many pairs there are <= distance d in amino acid space

Usage

```
mutpairsd(maf, d = 0)
```

Arguments

maf	maf data.frame
d	distance threshold in amino acid space

Value

clusters of mutations with how many pairs of variants supporting

Author(s)

Marcin Imielinski

mutrate_window *mutrate_window*

Description

Computes mutation rates along k gene "windows" along an ordered list "genes" of genes.

Usage

```
mutrate_window(genes, maf, cov, window = 100)
```

Author(s)

Marcin Imielinski

mut_genecluster *mut_genecluster*

Description

Greedy divisive clustering of genes based on mutation rates along a provided order (eg order of gene expression)

Outputs a list of gene clusters (each list a character vector of gene symbols)

Usage

```
mut_genecluster(genes, maf, cov, p.thresh = 0.05, min.cluster.size = 2,
  bonferonni = TRUE)
```

Author(s)

Marcin Imielinski

nz *nz*

Description

outputs the nonzero entries of a vector or array

Usage

```
nz(x, zero = 0, full = FALSE, matrix = TRUE)
```

Arguments

x	length(x)
zero	integer specifying what to use as the "zero" value in the input (=0)

Value

data.frame of row id col id value pairs

Author(s)

Marcin Imielinski

pad *pad*

Description

pads an (integer) vector with k places below and above its lowest and highest value

(by default, clips at 0)

useful for querying around specific entires of vector, matrix, data.frame, GRanges ewtc

Usage

```
pad(x, k, clip = T)
```

Arguments

x	integer vector to pad
k	window around each entry to pad
clip	logical flag whether to clip elements below 0 (=TRUE)

Value

"padded" integer vector of unique entires with entries in k window around each input included

Author(s)

Marcin Imielinski

parsesnpeff *parsesnpeff*

Description

parses vcf file containing SnpEff annotations on Strelka calls

Usage

```
parsesnpeff(vcf, id = NULL)
```

Arguments

vcf	path to vcf
id	id of case

Value

GRanges object of all variants and annotations

Author(s)

Kevin Hadi

phist *phist*

Description

Quick plotlyhistogram

Usage

```
phist(expr, data = data.frame(), ...)
```

Author(s)

Marcin Imielinski

pindel *pindel*

Description

calls pindel on a set of input bams and a given set of targets, outputting to target.dir

intervals are given as GRanges

Usage

```
pindel(outdir, bams, intervals = NULL, isizes = NULL,
  hg = Sys.getenv("GATK.FASTA"), genome = "hg19", outroot = "out",
  run = F, verbose = T, write_bam = F, oncotate = F, threads = 1,
  window.size = 10, other.args = "")
```

Arguments

`outdir`	out directory to output to
`bams`	vector of input bams
`intervals`	GRanges of to run on (=NULL)
`isizes`	integer insert.size to use
`hg`	genome fasta location (=Sys.getenv('GATK.HG19'))
`genome`	genome build (='hg19')
`outroot`	prefix to give output files (='out')
`run`	logical flag whether to run immediately or just return character vector of command (= FALSE)
`verbose`	logical flag
`write_bam`	logical flag whether to write the bam (=FALSE)
`oncotate`	logical flag whether to oncotate output into MAF files (=TRUE)
`threads`	number of threads to use
`window.size`	integer window size to use (=10)
`other.args`	other args to add (=")

Value

character vector of command(s) (only if run = FALSE), otherwise just runs command with system call.

Author(s)

Marcin Imielinski

`plop` *plop*

Description

grabs file and plops into public_html (or Sys.getenv('PLOP.DIR') if defined)

prefix will be added to left of file name (can include firectory)

if fn is list then prefix is expanded to unlisted fn

Useful for inspecting a specific subset of analysis files eg when debugging.

Usage

```
plop(fn, prefix = NULL, force = NULL)
```

Arguments

`fn`	character vector of filenames to "plop" into ~public_html/prefix (Sys.getenv('PLOP.DIR') is used as alternative if defined)
`prefix`	character prefix to add to filenames after plopping (can include subdirectories which can be inspected)

Author(s)

Marcin Imielinski

`plot_multiplicity` *plot_multiplicity*

Description

Plots allele fractions and colors in an individual based on multiplicity call stratified by total copy number value "individual" is a list or data frame with field $Individual_Id, $Purity

maf is a maf file annotated with multiplicity (ie the output of allele_multiplicity) with additional fields $cn.tot, $mult, $mult.p

Usage

```
plot_multiplicity(individual, maf, plot.reads = F)
```

Author(s)

Marcin Imielinski

`pmGSEA` *poor mans GSEA*

Description

pmGSEA "poor man's GSEA " ***

Given a gene.set (character vector) or gene.sets (list of character vectors) and given a named vector of significance values or table of significant genes (sig.table) (if table then significance column is $p or first column) identifies gene sets that have significant negative deviation of a "signed K-S" statistic vs uniform distribution (ie have p values significantly clustering towards zero) ie are significantly enriched in genes showing positive selection.

if positive.selection = F, will identify sets with significantly positive deviation of a "signed K-S" statistic (ie have p values significantly clustering towards 1) these are sets showig significant negative selection.

All p-values are computed against a distribution of signed K-S statistic obtained through permutation using random gene sets of the same size chosen from sig.table

Will adaptively perform permutations between minperms and maxperms using following rule of thumb: if there are <PERM.THRESH permutations with greater than (lower.tail = F) or less than (lower.tail = T) score than observed score, then will compute additional perms

*** actually not much poorer than the original GSEA, basically a reimplementation of Mootha et al Nat Gen 2002

Usage

```
pmGSEA(gene.sets, sig.table, min.perms = 100, max.perms = 1e+05,
  positive.selection = T, length.filter = F, length.range = c(5, 50),
  plot.hist = F, verbose = F, bootstrap = T, rank.test = F)
```

Arguments

`gene.sets`	a named list of character vectors, each list item is a gene set, i.e. a character vector of genes
`sig.table`	named vector of p values from an analysis e.g. mutSig, the names of the genes are
`min.perms`	minimum number of permutations to do in the adaptive permutation test
`length.range`	length 2 integer vector specifying min and max gene set size to score after intersection with genes in sig.table default: c(5,50)
`min.perms`	maximum number of permutations to do in the adaptive permutation test

Author(s)

Marcin Imielinski

ppdf *ppdf*

Description

sends quick plot to ~public_html/plot.pdf. If PPDF.DIR env variable defined, then will send to that directory (i.e. instead of public_html) Useful for doing quick standard plots to a static location which one views through tabs in Chrome or other web browser.

Usage

```
ppdf(expr, filename = "plot.pdf", height = 10, width = 10, cex = 1,
  title = NULL, byrow = TRUE, dim = NULL, cex.title = 1, ...)
```

Arguments

expr	Plotting expression eg plot(runif(1000), runif(1000))
filename	filename under ~/public_html/ or Sys.getenv('PPDF.DIR') to dump plots to (='plot.pdf')
height	integer pixel height of plot (=1000)
width	integer pixel width of plot (=1000)
cex	expansion factor of plot from "default size" either length 1 scalar or length 2 vector specifying height and width expansion (=1)
title	title to add to plot (=")
dim	length 2 integer vector, if expr contains multiple plot calls then will output to matrix of plots with specified dim (=NULL)
cex.title	character expansion factor to title)
...	additional arguments to pdf()

Author(s)

Marcin Imielinski

ppng *ppng*

Description

sends quick plot to ~public_html/plot.png. If PPNG.DIR env variable defined, then will send to that directory (i.e. instead of public_html) Useful for doing quick standard plots to a static location which one views through tabs in Chrome or other web browser.

Usage

```
ppng(expr, filename = "plot.png", height = 1000, width = 1000,
  dim = NULL, cex = 1, title = NULL, cex.pointsize = min(cex),
  cex.title = 1, ...)
```

Arguments

`expr`	Plotting expression eg plot(runif(1000), runif(1000))
`filename`	filename under ~/public_html/ or Sys.getenv('PPNG.DIR') to dump plots to (='plot.png')
`height`	integer pixel height of plot (=1000)
`width`	integer pixel width of plot (=1000)
`dim`	length 2 integer vector, if expr contains multiple plot calls then will output to matrix of plots with specified dim (=NULL)
`cex`	expansion factor of plot from "default size" either length 1 scalar or length 2 vector specifying height and width expansion (=1)
`title`	title to add to plot (='')
`cex.title`	character expansion factor to title)
`...`	additional arguments to png()

Author(s)

Marcin Imielinski

qhost *qstat*

Description

Tabulates per host cluster load

Usage

```
qhost(full = FALSE, numslots = TRUE)
```

qq_pval *qq plot given input p values*

Usage

```
qq_pval(obs, highlight = c(), exp = NULL, lwd = 1, bestfit = T,
  col = NULL, col.bg = "black", pch = 18, cex = 1, conf.lines = T,
  max = NULL, max.x = NULL, max.y = NULL, qvalues = NULL,
  label = NULL, plotly = FALSE, annotations = list(), gradient = list(),
  titleText = "", subsample = NA, ...)
```

Arguments

`obs` vector of pvalues to plot, names of obs can be intepreted as labels

`highlight` optional arg specifying indices of data points to highlight (ie color red)

`lwd` integer, optional, specifying thickness of line fit to data

`pch` integer dot type for scatter plot

`cex` integer dot size for scatter plot

`conf.lines` logical, optional, whether to draw 95 percent confidence interval lines around x-y line

`max` numeric, optional, threshold to max the input p values

`label` character vector, optional specifying which data points to label (obs vector has to be named, for this to work)

`plotly` toggles between creating a pdf (FALSE) or an interactive html widget (TRUE)

`annotations` named list of vectors containing information to present as hover text (html widget), must be in same order as obs input

`gradient` named list that contains one vector that color codes points based on value, must bein same order as obs input

`titleText` title for plotly (html) graph only

`samp` integer, optional specifying how many samples to draw from input data (default NULL)

Author(s)

Marcin Imielinski, Eran Hodis, Zoran Gajic

`qstat` *qstat*

Description

Tabulates cluster usage (qstat()) or if full = TRUE flag given will dump out all running jobs in a data.table

Usage

```
qstat(full = FALSE, numslots = TRUE)
```

`query_lsf_out` *query_lsf_out*

Description

parses "out" and "err" files of jobs with jname root to identify exit status and error codes of jobs

Usage

```
lsf_query(dir = NULL, jname = NULL, detailed = F, mc.cores = 1)
```

Arguments

`dir`	character specifyhing directory where .out and .err files are located
`jname`	character vector names of jobs (as specified in bsub_cmd
`detailed`	logical flag specifying whether to return "detailed" information
`mc.cores`	integer specifying how many cores to use to parse the output data

Value

data.frame of job info

Author(s)

Marcin Imielinski

`queues` *queues*

Description

Lists all available queues

Usage

```
queues()
```

quickSig *quickSig*

Description

Quick implementation of Mike / Gaddy binomial / poisson model. Requires category based coverage output (*.per_gene.coverage.txt file) computed during mutsig preprocess step. (run mutsig_preprocess setting the following additional flags: P.output_per_gene_coverage = true; P.output_per_gene_mutation_c = true; P.simplified_gene_sample_coverage_table = false; P.simplified_gene_sample_mutation_counts_table = false;)

Computes context-category specific mutation rates either across whole cohort or within strata of gene-patient categories.

Outputs a significance table with the following columns for each category (if analyze.categories = TRUE) and/or category "tot" which is across all categories o.k = observed mutations of category k e.k = expected mutations of category k given background model eff.k = log(o.k / e.k) for category k p.k = p value of deviation from expectation under poisson model q.k = q value of deviation

Usage

```
quickSig(maf, cov, patients = NULL, genes = NULL, analyze.categories = F,
  remove.silent = TRUE, limit.cat = NULL, two.tailed = TRUE)
```

Author(s)

Marcin Imielinski

ra.overlaps *identifies events that are in ra1 that do not exist in ra2. Aside from ra1 and ra2, all arguments are sent to ra.overlaps*

Description

identifies events that are in ra1 that do not exist in ra2. Aside from ra1 and ra2, all arguments are sent to ra.overlaps

Usage

```
ra.setdiff(ra1, ra2, ...)
```

ra.union *ra.union*

Description

returns events in ra1 that are in ra2 also

Usage

```
ra.union(ra1, ra2, ...)
```

`ra_breaks` *ra_breaks*

Description

takes in either file or data frame from dranger or snowman or path to BND / SV type vcf file and returns junctions in VCF format.

Usage

```
ra_breaks(rafile, keep.features = T, seqlengths = hg_seqlengths(),
  chr.convert = T, snowman = FALSE, swap.header = NULL,
  breakpointer = FALSE, seqlevels = NULL, force.bnd = FALSE, skip = NA,
  get.loose = FALSE)
```

Details

The default output is GRangesList each with a length two GRanges whose strands point AWAY from the break. If get.loose = TRUE (only relevant for VCF)

`read.delim.cat` *read.delim.cat*

Description

takes a vector of tab delimited file paths and concatenates them into a single data frame (takin union of identically named / numbered columns as a default)

Usage

```
read.delim.cat(paths, skip = NULL, cols = NULL, include.paths = T,
  include.index = TRUE, cores = NULL, ...)
```

Arguments

`paths`	length n character vector of paths to tsv files
`skip`	optional length n or length 1 integer specifying how many lines to skip
`cols`	optional character vector of which cols to keep (by default union of all columns)
`include.paths`	optional logical flag whether to include paths to files as column $source.path column
`include.index`	optional logical flag whether to include source rownames if exist as $source.id column
`cores`	optional integer specifying number of cores to use (def 1)
`...`	additional args to read.delim

Author(s)

Marcin Imielinski

readRDA *readRDA*

Description

loads Rdata environment into a list variable and returns (to mirror RDS functionality)

Usage

```
readRDA(fn)
```

Arguments

fn	file name of .rda or .RData file

Value

object containing all the elements of the environment stored in fn

Author(s)

Marcin Imielinski

read_hg *Wrapper around BSgenome call*

Description

Retreives either the BSgenome hg18 or hg19 genome by default. Requires packages BSgenome.Hsapiens.UCSC.hg19 for hg19 and BSgenome.Hsapiens.UCSC.hg19 for hg18.

Usage

```
read_hg(hg19 = T, fft = F)
```

Arguments

hg19	Logical whether to return hg18 or hg19 BSgenome. Default TRUE
fft	Logical whether to return an ffTrack. Default FALSE

Details

If fft = TRUE, can also also return the hg19 ffTrack (requires that the file exists) Requires the existence of environment variable HG.FFT pointing to ffTrack .rds file..

Value

BSgenome or ffTrack of the genome

`relib` *relib*

Description

Reload library

Usage

```
relib(lib = "Flow")
```

`reorder_maf` *reorder_maf*

Description

Re-orders maf to comply with the TCGA MAF specifications (v2.2), tacking on all "non-standard" columns after the initial 32

Usage

```
reorder_maf(maf, include.other = TRUE)
```

Arguments

`maf`	data.frame of MAF
`include.other`	logical flag whether to include non-standard maf columns after the standard ones (=TRUE)

Value

data.frame representing MAF in standard column order

Author(s)

Marcin Imielinski

rmix *rmix*

Description

sample N points from a mixture of k densities of a single functional form (eg norm, beta, multinomial) where n is either an integer vector of length k denoting how many samples to be drawn from each density (in which case N = sum(n)) or n is a scalar, in which case n points are drawn from each density and N = n*k.

p = params data frame whose named columns correspond to arguments to rdens (eg $n, $shape1, $shape2 for rbeta or $n, $mean, $sd for rnorm) rdens = function encoding random number generator for given density, that takes as input named columns of params n = either an nrow(p) integer vector or scalar denoting how many samples to draw from each density

n can also be just be a column of p

useful for plotting "smears" of points

Output is the rbind-ed output of individual rdens calls

Usage

```
rmix(p, rdens, n = NULL)
```

Arguments

p	k x p data frame of k parameter sets of rdens density functions, each column is a parameter value, each row is a parameter setting for a mixture component
rdens	R density specific random number generator function object (eg rnorm)
n	length k or legnth 1 integer specifying number of samples to draw from each mixture component

Author(s)

Marcin Imielinski

row.slice *row.slice*

Description

Hacked operator for subsetting rows of data.frames, DataFrames, data.tables, GRanges

df

df

Usage

```
df %~% ...
```

Arguments

`df`	data.frame
`regex`	string to match or number in which case that column is returned (same behavior for data.table)

Value

slices of data.frame matching regex

Author(s)

Marcin Imielinski

`rpipe` *read pipe readsLines from pipe and then closes the pipe*

Usage

```
rpipe(cmd)
```

`rrbind2` *Improved rbidn for intersecting columns of data.frames or data.tables*

Description

like rbind, but takes the intersecting columns of the dfs rrbind = function(df1, df2, [df3 ... etc],)

Usage

```
rrbind2(..., union = T, as.data.table = FALSE)
```

Arguments

`...`	list of data frames to concatenate
`union`	if union flag is used then will take union of columns (and put NA's for columns of df1 not in df2 and vice versa). Default TRUE
`as.data.table`	[Default FALSE] return as a data.table

seqinfo2gr *Create GRanges from Seqinfo*

Description

Creates a genomic ranges from seqinfo object ie a pile of ranges spanning the genome

Usage

```
seqinfo2gr(si, strip.empty = FALSE)
```

Arguments

si	Seqinfo object
strip.empty	Don't know. Default FALSE

Examples

```
## Not run: si <- Seqinfo(names(hg_seqlength(), hg_seqlengths()))
seqinfo2gr(si)
## End(Not run)
```

set.comp *set.comp*

Description

Compares two sets and outputs data frame with "left", "middle", "right" members

Usage

```
set.comp(s1, s2)
```

Arguments

s1	vector corresponding to "set 1"
s2	vector corresponding to "set 2"

Value

list with fields $left, $middle, and $right corresponding to vectors that are in the left setdiff, intersection, right setdiff respectively

Author(s)

Bryan Hernandez

`setxor` *setxor*

Usage

```
setxor(A, B)
```

Arguments

`A`	vector specifying set A
`B`	vector specifying set B

Value

elements in A or B that are not in the intersection of A and B

Author(s)

Marcin Imielinski

`sortable` *sortable*

Description

dumps sortable list for manual sorting into list.html (in public_html by default)

Usage

```
sortable(x, filename = "list.html", title = NULL)
```

`sparse_subset` *sparse_subset*

Description

given k1 x n matrix A and k2 x n matrix B returns k1 x k2 matrix C whose entries ij = 1 if the set of nonzero components of row i of A is a (+/- strict) subset of the nonzero components of row j of B

Usage

```
sparse_subset(A, B, strict = FALSE, chunksize = 100, quiet = FALSE)
```

Arguments

`A`	k1 x n matrix
`B`	k2 x n matrix
`strict`	logical flag whether to return strict subset (=FALSE)
`chunksize`	integer size of rows to process from each matrix at a single iteration (=100)
`quiet`	logical flag (=FALSE)

Value

k1 x k2 matrix C whose entries ij = 1 if the set of nonzero components of row i of A is a (+/- strict) subset of the nonzero components of row j of B

Author(s)

Marcin Imielinski

---

splot *splot*

Description

convenient formatted scatter plot with additional features as defaults, useful for fast interactive data inspection / exploration of large datasets (eg 1000s of points): - autoamtic setting of solid dots (pch = 19) - transparent colors for over plotting - automatic setting of x and y limits parametrized by "p.outlier" - quick setting of jiggle / jitter on plot - automatic fitting and plotting of regression line (fit = FALSE)

Usage

```
splot(x, y, cex = 0.4, poutlier = 0.01, col = alpha("black", 0.3),
  xlim = quantile(x, na.rm = T, prob = c(poutlier[1], 1 -
  poutlier[length(poutlier)])), ylim = quantile(y, na.rm = T, prob =
  c(poutlier[1], 1 - poutlier[length(poutlier)])), label = NULL,
  cex.label = 1, adj.label = c(1, 0.5), col.label = "black", log = "",
  jiggle = NULL, fit = FALSE, col.fit = "blue", cex.fit = 1,
  square = FALSE, pch = 19, ...)
```

Arguments

`x`	numeric vector of x data
`y`	numeric vector y data
`cex`	character inspection
`poutlier`	numeric value between 0 and 1 specifying quantile threshold of outliers to remove (=0.01)
`col`	character vector color (=alpha('black', 0.3))
`xlim`	length 2 numeric vector specifying x limits (=quantile(x, na.rm = T, prob = c(poutlier[1], 1-poutlier[length(poutlier)])))
`ylim`	length 2 numeric vector specifying y limits (=quantile(y, na.rm = T, prob = c(poutlier[1], 1-poutlier[length(poutlier)])))
`log`	standard plot log string
`jiggle`	numeric value between 0 and 1 specifying what percentage of plot area to jiggle each point (useful for overplotting) (= NULL)
`fit`	logical flag whether to fit a linear regression line to the data (=FALSE)
`col.fit`	character specifying color of linear regression fit (='blue')
`cex.fit`	character specifying size of text associated with linear regerssion line (=1)
`square`	logical flag whether to make square plot
`pch`	pch
`...`	...

Author(s)

Marcin Imielinski

`standardize_segs` *standardize_segs*

Description

(data frame seg function)

Usage

```
standardize_segs(seg, chr = FALSE)
```

Details

Takes and returns segs data frame standardized to a single format (ie $chr, $pos1, $pos2)

if chr = TRUE will ensure "chr" prefix is added to chromossome(if does not exist)#'

`strsplit.fwf` *splits a single string according to fixed widths contained in fw (ie each components i of fw denotes the width of field i in string str*

Description

splits a single string according to fixed widths contained in fw (ie each components i of fw denotes the width of field i in string str

Usage

```
strsplit.fwf(str, fw)
```

`strsplit2` *strsplit2*

Description

Strsplit when there are two layers of separators (sep1, sep2) and one needs to extract a collapsed vector of subitem j for all items i.

Takes in a character vector and outputs a list of "separated" items

Usage

```
strsplit2(x, sep1 = ",", sep2 = " ", j = 1)
```

Arguments

`x`	character vector
`sep1`	character specifying first level separator (=',')
`sep2`	character specifying second level separator (=' ')
`j`	integer specifying which subitem to keep (=1)

Value

vector of values for subitem j

Author(s)

Marcin Imielinski

`sub2ind` *sub2ind*

Description

MATLAB style sub2ind function in R physical essence. Provides the one dim matrix index of row-column locations in matrix

(RIP matlab)

Usage

```
sub2ind(dim, r, c, byrow = F)
```

Arguments

`dim`	dimension of matrix to return index for
`r`	integer vector of row index to look up
`c`	length(r) integer vector of column index to look up
`byrow`	whether to calculate indices by row or column (= FALSE)

Value

length(r) vector of 1D indices into matrix with dim "dim"

Author(s)

Marcin Imielinski

system.call — *Wrapper to base system function to call system (e.g. perl) from R. The only benefit to this wrapper is a more controlled verbose argument.*

Description

Wrapper to base system function to call system (e.g. perl) from R. The only benefit to this wrapper is a more controlled verbose argument.

Usage

```
system.call(syscall, verbose = T)
```

Arguments

syscall	string containing the system call
verbose	print the syscall to screen, and it's stdout

Author(s)

Jeremiah Wala <jwala@broadinstitute.org>

Examples

```
# system.call('perl s/[0-9]+//g file1 > file2')
```

tabstring — *tabstring*

Description

string representation of a named vector (ie the result of tab = table(x) ie name1 (value1), name2 (value2), name3 (value3)

Usage

```
tabstring(tab, sep = ", ", sep2 = "_", dt = FALSE)
```

Arguments

tab	"table" or any named(vector)
sep	separator to use between table elements

Value

character representation of table

Author(s)

Marcin Imielinski

tailf *tailf*

Description

"tail -f" +/- grep vector of files

Usage

```
tailf(x, n = NULL, grep = NULL)
```

Arguments

x	vector of iles
grep	string to grep in files (=NULL)

Author(s)

Marcin Imielinski

timestamp *timestamp*

Description

returns character time stamp

Usage

```
timestamp()
```

Author(s)

Marcin Imielinski

toggle_grfo *toggle data.table vs IRanges find overlaps*

Description

toggles global setting of whether to use data.table vs IRanges find overlaps machinery

Usage

```
.toggle_grfo()
```

Author(s)

Marcin Imielinski

`ucount` *ucount*

Description

returns vector of same length as input with number of counts of each value in the whole list

Usage

```
ucount(x)
```

Arguments

`x`	vector

Value

length(x) vector with number of instances of each item in x

Author(s)

Marcin Imielinski

`vaggregate` *vaggregate*

Description

same as aggregate except returns named vector with names as first column of output and values as second

Note: there is no need to ever use aggregate or vaggregate, just switch to data.table

Usage

```
vaggregate(...)
```

Arguments

`...`	arguments to aggregate

Value

named vector indexed by levels of "by"

Author(s)

Marcin Imielinski

varcount *varcount*

Description

Wrapper around applyPileups

Usage

```
varcount(bams, gr, min.mapq = 0, min.baseq = 20, max.depth = 500,
  indel = F, ...)
```

Details

takes in vector of bam paths, GRanges corresponding to sites / territories to query, and outputs a list with fields $counts = 3D matrix of base counts (A, C, G, T, N) x sites x bams subject to mapq and baseq thresholds #'

(uses varbase)

... = other args go to read.bam

vplot *vplot*

Description

Quick violin plot

Usage

```
vplot(y, group = "x", facet1 = NULL, facet2 = NULL, transpose = FALSE,
  mapping = NULL, stat = "ydensity", position = "dodge", trim = TRUE,
  scale = "area", log = FALSE, count = TRUE, xlab = NULL, ylim = NULL,
  ylab = NULL, minsup = NA, scatter = FALSE, text = NULL,
  cex.scatter = 1, col.scatter = NULL, alpha = 0.3, title = NULL,
  legend.ncol = NULL, legend.nrow = NULL, vfilter = TRUE, vplot = TRUE,
  dot = FALSE, stackratio = 1, binwidth = 0.1, plotly = FALSE,
  print = TRUE)
```

Arguments

`y`	numeric data vector
`group`	length(y) vector of categories
`facet1`	optional length(y) vector of row categories to facet on (=NULL)
`facet2`	optional length(y) vector of column categories to facet on (=NULL)
`transpose`	logical vector whether flip row / column orientation of facets (=FALSE)
`mapping`	mapping of groups to colors (=NULL)
`scale`	scale parameter to geom_vplot (=area)

`log`	logical flag whether to log transform (=FALSE)
`count`	logical flag whether to include counts in ylabels (=TRUE)
`xlab`	character xlabel (=NULL)
`ylab`	character ylabel (=NULL)
`minsup`	minimum support to include in a group (=NA)
`scatter`	logical flag whether to include scatter of points (=FALSE)
`alpha`	numeric vector between 0 and 1 to specify alpha transparency of points if scatter is TRUE (0.3)
`title`	character specifying plot title (=NULL)

Author(s)

Marcin Imielinski

`wfplot` *Quick waterfall plot*

Description

Quick waterfall plot

data is a numeric vector labels are text labels of the same length as data col is either (1) an unamed list of unique colors (2) a named list mapping unique labels to colors

Usage

```
wfplot(data, labels = NULL, names.arg = NULL, col = NULL, las = 2,
  cex = 1, leg.pos = NULL, ...)
```

Arguments

`data`	length n numeric vector to be drawn and sorted on y axis
`labels`	length n character vector categorical labels of data
`names.arg`	length n character vector, optional, of individual labels to be drawn verticallyon x axis
`col`	optional named character vector mapping unique category labels to colors
`las`	optional integer vector specifying orientation of labels on barplot
`cex`	numeric value specifying size of names.arg data labels
`leg.pos`	NULL
`...`	additional arguments to barplot

Value

plot

Author(s)

Marcin Imielinski

which.char *which.char*

Description

finds the index of the character in subject (length 1 character vector) matching nchar = 1 single character query eg which.char('a', 'cat') = 2

if query has more than one char (or has length>1) then will return indices matching <any one> of the characters in any element of query

Usage

```
which.char(subject, query)
```

Arguments

subject	length 1 character vector
query	length 1, nchar 1 character

Value

indices in subject that query appears

Author(s)

Marcin Imielinski

wij *wij*

Description

Evaluates output of htmlwidget generating expression (e.g. via highcharter) and send to filename in predefined WIDGET.DIR by default plot.html

Usage

```
wij(expr, filename = "plot.html", zoom = NULL, cex = 1, force = FALSE,
  quiet = FALSE, embed = FALSE)
```

wijj *wijj*

Description

Embeds widget in jupyter notebook

Usage

```
wijj(x, width = NULL, height = NULL, file = paste0("plotlyJupyterHTML/",
  digest::digest(Sys.time()), ".html"))
```

write.htab *write.htab*

Description

writes data frame (or anything castable to data frame) to pretty HTML formatted table to a static location. Useful for quick table inspection via Chrome or other web browser.

Very similar syntax to write.tab

Usage

```
write.htab(tab, file = NULL, title = NULL, footer = NULL,
  highlight = NULL, row.names = TRUE, col.names = TRUE,
  high.color = "yellow", row.colors = c("lightgray", "white"),
  header.colors = c("#4A4A4A", "white"), data.size = 15, dt = TRUE,
  force = FALSE, embed = FALSE, title.size = 15, footer.size = 20,
  header.size = round(1.1 * data.size))
```

Arguments

`tab`	data.frame, data.table, or GRanges
`file`	optional output .html file (by default ~/public_html/htab.html but directory can be set using env variable HTAB.PATH)
`title`	title to the page (=NULL)
`footer`	footer to add to page (=NULL)
`highlight`	optional integer vector specifying what rows to highlight
`row.names`	logical flag whether to include row labels (=TRUE)
`col.names`	logical flag whether to include col labels (=TRUE)
`high.color`	chaoracter highlight color (= 'yellow')
`row.colors`	length 2 character vector to shade data rows (= c('lightgray', 'white'))
`header.colors`	length 2 character vector specifygin background and text for header row (= c('#4A4A4A', 'white'))
`data.size`	integer font size in px for data, title, and footer (= 15)
`title.size`	integer font size in px for title (= 15)
`footer.size`	integer font size in px for footer (= 15)
`header.size`	integer font size in px for header (= 15)

Author(s)

Marcin Imielinski

`write.tab`	*write.tab writes tab delimited no quotes without row names table (passes remaining arguments to write.table) equivalent to write.table(sep = TAB.DELIM, quote = F, row.names = F)*

Usage

```
write.tab(x, ..., sep = "\t", quote = F, row.names = F)
```

Arguments

`x`	data.frame to dump
`...`	additional arguments to write.table

Index