GGtools

April 20, 2011

`bestCis` *extract best (or all) cis-associated eQTL from a multffmgr instance*

Description

extract best (or all) cis-associated eQTL from a multffmgr instance

Usage

```
bestCis(ffmgr, slranges, radius = 1e+06, ffind = 1, anno, ncores = 10)
#allCisP_1sided(ffmgr, slranges, radius = 1e+06, ffind = 1, anno, ncores = 10)
```

Arguments

`ffmgr`	manager object, output of multffCT or diagffCC
`slranges`	snp locations RangedData instance
`radius`	number of bases up and down stream to declare cis
`ffind`	index into fflist component of manager for eQTL associations cores
`anno`	character atom naming annotation package for resolution of colnames of ffmgr matrix
`ncores`	number of cores for mclapply to use

Value

for `bestCis`, data frame with genes as rows, rsnum and chisq(df) scores, with df and gene and SNP locations as columns.

Author(s)

VJ Carey

Examples

```
example(diagffCC)
data(snpLocs20)
bestCis(ff, snpLocs20, anno="illuminaHumanv1.db")
```

cisRanges *create GRanges instance for intervals cis to a set of genes*

Description

create GRanges instance for intervals cis to a set of genes

Usage

```
cisRanges(probeids, chr, anno, radius = 5e+05, useEnd=FALSE)
```

Arguments

probeids	character vector of array probe identifiers for annotation mapped by package identified in anno
chr	a string to be used for seqnames component of GRanges result
anno	string identifying an annotation package from which locations will be derived
radius	count of basepairs upstream and downstream of (first) CHRLOC entry defining the range
useEnd	logical to request cis relative to radius past entire gene sequence as opposed to local to TSS

Value

a GRanges-class instance

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
library(illuminaHumanv1.db)
g20 = get("20", revmap(illuminaHumanv1CHR))
m1 = intersect(g20, mappedkeys(illuminaHumanv1CHRLOC))
cisRanges(m1[1:10], "chr20", "illuminaHumanv1.db")
```

cisSnpTests *perform tests for eQTL cis to specified genes*

Description

perform tests for eQTL cis to specified genes

Usage

```
cisSnpTests(fmla, smls, radius, ...)
```

Arguments

fmla	standard formula. LHS can be a GeneSet with AnnotationIdentifier geneIdType. RHS can be predictor formula component using variables in pData of smls
smls	instance of smlSet
radius	numeric value: number of bases up and downstream from probe CHRLOC to be examined for SNP
...	not in use

Value

a list of cwSnpScreen instances

Note

Getting SNP locations is slow for the first event while metadata are brought into scope. Subsequent calls are faster.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
library(GSEABase)
# two genes on chr 20
gs1 = GeneSet(c("CPNE1", "ADA"), geneIdType = SymbolIdentifier())
gs2 = gs1
organism(gs2) = "Homo sapiens"
geneIdType(gs2) = AnnotationIdentifier("illuminaHumanv1.db")
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
cc = cisSnpTests(gs2~male, hmceuB36.2021, radius=1e5)
lapply(cc, function(x) length(p.value(x@.Data[[1]])))
cc = cisSnpTests(gs2~male, hmceuB36.2021, radius=1e6)
lapply(cc, function(x) length(p.value(x@.Data[[1]])))
```

---

degnerASE01 *transcription of a table from a paper by Degner et al*

Description

transcription of a table from a paper by Degner et al, involving identification of genes with allele-specific expression discovered by RNA-seq

Usage

```
data(degnerASE01)
```

Format

A data frame with 55 observations on the following 10 variables.

`rsnum` a factor with levels `rs10266655` `rs1042448` `rs1046747` `rs1047469` `rs1059307` `rs1060915` `rs11009147` `rs1127326` `rs11376` `rs11570126` `rs11578` `rs1158` `rs13306758` `rs13309` `rs16952692` `rs17014852` `rs17459` `rs1879182` `rs2070924` `rs2071888` `rs2089910` `rs2234978` `rs2271920` `rs2530680` `rs3025040` `rs3170545` `rs325400` `rs368116` `rs3819946` `rs3871984` `rs4784800` `rs4982685` `rs558018` `rs6568` `rs6682136` `rs6890805` `rs7046` `rs705` `rs7121` `rs7141712` `rs7192` `rs7695` `rs7739387` `rs8023358` `rs8084` `rs8429` `rs8647` `rs8905` `rs9038` `rs916974`

`refreads` a numeric vector

`nonrefreads` a numeric vector

`miscall` a numeric vector

`chr` a factor with levels `chr1` `chr10` `chr11` `chr12` `chr14` `chr15` `chr16` `chr17` `chr18` `chr19` `chr2` `chr20` `chr22` `chr5` `chr6` `chr7` `chr8` `chr9`

`loc` a numeric vector

`gene` a factor with levels `ADAR` `ADPGK` `AKAP2` `AP4M1` `ATF5` `BIN1` `BRCA1` `C6orf106` `CCL22` `CD59` `CRYZ` `DFNA5` `ENSA` `FAS` `GNAS` `GYPC` `HLA-DPB1` `HLA-DRA` `HMMR` `ITGB1` `LSP1` `MADD` `MARK3` `ME2` `MEF2A` `MGAT1` `MRPL52` `MTMR2` `NF2` `NIN` `NUP62` `OAS2` `PALM2-AKAP2` `PIP4K2A` `PRKAR1A` `PTK2B` `SAR1A` `SEC22B` `SEMA4A` `SEPT9` `SLC2A1` `SNHG5` `SNURF/SNRPN` `STX16` `TAF6` `TAPBP` `VEGFA`

`indiv` a factor with levels `GM19238` `GM19239`

`eqtl` a factor with levels `Yes`

`imprint` a logical vector

Source

Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data. Jacob F. Degner 1,3,, John C. Marioni 1,, Athma A. Pai 1, Joseph K. Pickrell 1, Everlyne Nkadori 1,2, Yoav Gilad 1, and Jonathan K. Pritchard 1,2, Bioinformatics 2009.

Examples

```
data(degnerASE01)
degnerASE01[1:4,]
## maybe str(degnerASE01) ; plot(degnerASE01) ...
```

`diagffCC` *perform a 'diagonal' cis eQTL search (only check SNPs chromosomally coresident with genes)*

Description

perform a 'diagonal' cis eQTL search (only check SNPs chromosomally coresident with genes)

Usage

```
diagffCC(sms, gfmla, targdir = ".", runname = "foo", overwriteFF = TRUE, ncores
```

Arguments

sms	smlSet
gfmla	formula with right-hand side specifying covariates, dependent variable should be 'gs'
targdir	folder to hold results
runname	arbitrary distinguishing tag
overwriteFF	preserve preexisting FF files if FALSE
ncores	number of cores to use with multicore
vmode	can be "short" to use efficient space
shortfac	amount to scale short ints by to preserve some precision
mc.set.seed	as in multicore
fillNA	when test cannot be performed (eg due to monomorphy) fill in with chisq(1) if true
...	passed to snp.rhs.tests of snpMatrix

Details

uses annotation package specified in annotation slot of smlSet (which should have .db suffix) to get list of genes on each chromosome present in smlSet

Value

a multffManager instance

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
if (require(ff)){
data(hmceuB36.2021)
library(illuminaHumanv1.db)
g20 = get("20", revmap(illuminaHumanv1CHR))[1:10]
g21 = get("21", revmap(illuminaHumanv1CHR))[1:10]
cpn = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
g20 = c(g20,cpn)
hh = hmceuB36.2021[probeId(c(g20,g21)),]
owd = getwd()
setwd(ind <- tempdir())
print(ind)
ff = diagffCC( hh, gs~male, runname="test")
ff
# we know the following should have a score above 50
ff[ rsid("rs6060535"), probeId(cpn) ]
#
# now compute (minimum over genes, snp-specific) p-values associated with maximal chi-squ
mm = maxchisq(ff)
mm
pvraw = min_p_vals( mm, "none", "", 2 )
length(pvraw)
```

```
pvraw[[1]][1:10]
# pvadj = min_p_vals( mm, "BH", "chr_specific", 2 )
# pvadj[[1]][1:10]
mm2 = maxchisq(ff, type="perGene")
mm2
# min_p_vals(mm2, "BH", "global", sidedness=2)[[1]][1:5]
}
setwd(owd)
```

`eqtlTestsManager-class`
Class "eqtlTestsManager"

Description

interface to ff files that store results for large numbers of eQTL tests

Objects from the Class

Objects can be created by calls of the form `new("eqtlTestsManager", ...)`, or `new("cisTransDirector", ...)`. The `mkCisTransDirector` function should be used for the latter task.

A manager object collects metadata and reference information regarding tests relating a single set of expression measures (gene-oriented) and a collection of structural variants (snp-oriented).

A director object collects metadata and reference information for a specified set of managers.

Slots

`fflist`: Object of class `"list"` collection of serialized references to ff objects generated per chromosome

`call`: Object of class `"call"` call for auditing

`sess`: Object of class `"ANY"` sessionInfo() result

`exdate`: Object of class `"ANY"` execution date

`shortfac`: Object of class `"numeric"` factor by which short int data are inflated for increased resolution

`geneanno`: Object of class `"character"` name of annotation package documenting featureNames of expression data

`df`: Object of class `"numeric"` number of degrees of freedom of chi-square tests under null hypothesis

Methods

[`signature(x = "eqtlTestsManager", i = "rsid", j = "probeId", drop = "ANY")`: This gives matrix-like extraction idiom to retrieve chisquared statistics from the ff archives for eQTL searches

[`signature(x = "cisTransDirector", i = "character", j = "character", drop = "ANY")`: ...

show `signature(object = "eqtlTestsManager")`: ...

show `signature(object = "cisTransDirector")`: ...

probeNames `signature(object = "eqtlTestsManager")`: extract the probe names as a vector

probeNames `signature(object = "cisTransDirector")`: extract the probe names as a list with one element per manager

Note

Instances of this class can be coerced to instances of eqtlTestsManager to facilitate management by a cisTransDirector. Objects of class eqtlTestsManager include references to pathnames on the system on which the objects are created. These can be modified if serialized objects are moved along with the folder of ff-formatted outputs.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
# look at example(eqtlTests) for workout
showClass("eqtlTestsManager")
showClass("cisTransDirector")
```

`eqtlTests` *perform genome x transcriptome eQTL searches with high-performance options*

Description

perform genome x transcriptome eQTL searches with high-performance options

Usage

```
eqtlTests(smlSet, rhs = ~1 - 1, runname = "foo", targdir = "foo",
geneApply = lapply, chromApply = lapply, shortfac = 100, computeZ = FALSE,
        checkValid = TRUE, saveSummaries = TRUE, uncert=TRUE, family, ...)
ieqtlTests(smlSet, rhs = ~1 - 1, rules, runname = "ifoo", targdir = "ifoo",
geneApply = lapply, chromApply = lapply, shortfac = 100, computeZ = FALSE, uncer
meqtlTests(listOfSmls, rhslist, runname = "mfoo", targdir = "mfoo",
geneApply = lapply, chromApply = lapply, shortfac = 100, computeZ = FALSE, harmo
cisScores( mgr, ffind=1, chr, snpGR, radius=5e5, applier=lapply, minMAF = 0, min
```

Arguments

`smlSet`	instance of `smlSet-class`
`listOfSmls`	list of instances of `smlSet-class`
`rhs`	standard formula without dependent variable; predictors must be found in `pData(smlSet)`
`rhslist`	List of standard formulae without dependent variable; predictors for each formula must be found in `pData(smlSet)` associated with each rhslist element. In other words, the pData of the kth smlSet in listOfSmls provides the data to resolve the symbols in the kth formula of rhslist

rules	instance of snp.reg.imputation-class expressing rules by which unobserved SNP are 'imputed' (that is, the value used is the conditional expectation of B copy number, which is real-valued and may lie outside [0,2])
runname	arbitrary character string that will identify a serialized object storing references to results
targdir	arbitrary character string that will name a folder where results are stored as ff files
geneApply	lapply-like function for iterating over genes
chromApply	lapply-like function for iterating over chromosomes
shortfac	quantity by which chisquared tests will be inflated before coercion to short int
computeZ	logical to direct calculation of Zscore instead of X2
harmonizeSNPs	logical: it can be time consuming to harmonize SNPs across a long listOfSmls, so you can do this outside of the function and set harmonizeSNPs=FALSE; if TRUE, it will be done before statistical processing of the data in this function.
checkValid	logical: shall the function run validObject on input smlSet?
saveSummaries	logical: shall a set of ff files be stored that includes genotype and allele frequency data for downstream filtering?
uncert	setting for value of uncertain argument in snp.rhs.tests
family	specify the GLM family to use; defaults to 'gaussian' if left missing
...	parameters passed to snp.rhs.tests
mgr	an instance of eqtlTestsManager
ffind	index into the list of ff files managed by mgr to be used for obtaining scores
chr	token identifying chromosome of interest
snpGR	instance of GRanges-class with SNP location
radius	numeric: number of basepairs up and downstream of TSS to be used for cis filtering
minMAF	numeric: minimum minor allele frequency for inclusion of SNP in reporting
minGTF	numeric: minimum genotype frequency for inclusion of SNP in reporting
applier	function, typically either lapply or mclapply if multicore services are desired
useEnd	logical to request cis relative to radius past entire gene sequence as opposed to local to TSS

Details

snp.rhs.tests is run for all genes enumerated in featureNames(smlSet) individually as dependent variables, and all SNP in smList(smlSet) as predictors, one by one. Each model fitted for SNP genotype is additionally adjusted for elements in rhs. There are consequently G*S test results where G is the number of features in exprs(smlSet), and S is the total number of SNP in smlSet. These are stored in ff files in folder targdir.

imphm3_1KG_20_mA2 is a set of imputation rules for SNP on chromosome 20, where the 1000 genomes genotypes distributed in 'pilot1' VCF files are used to create imputations to loci not covered in the phase 3 hapmap data in hm3ceuSMS. Not useful in Bioconductor 2.7 as snpMatrix package lost its imputation facilities just prior to release; use snpMatrix2 in Bioconductor 2.8.

cisScores will fail if genes are present that are not on the chromosome for which scores are requested.

Value

(i,m)eqtlTests returns instance of `eqtlTestsManager`

cisScores returns list with elements for each gene consisting of chi-squared statistics for SNP cis to the genes according to settings of radius and useEnd

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
library(illuminaHumanv1.db)
cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
indc = which(featureNames(hmceuB36.2021) == cptag[1])
#
# get a set of additional genes on chr20
all20 = get("20", revmap(illuminaHumanv1CHR))
g20 = unique(c(all20[1:10], cptag))
#
hm = hmceuB36.2021[probeId(g20),]  # reduce problem
hm = hm[chrnum(c("20", "21")),]
td = tempdir()
curd = getwd()
setwd(td)
time.lapply = unix.time(e1 <- eqtlTests( hm, ~male ))
e1
dir("foo")
#
# now show how to extract scores cis to genes
#
 if (require(SNPlocs.Hsapiens.dbSNP.20100427)) {
 c20 = getSNPlocs("ch20")
 library(GenomicRanges)
 c20r = GRanges(seqnames="chr20", IRanges(c20$loc, width=1))
 names(c20r) = paste("rs", c20$RefSNP_id, sep="")
 chkcc = cisScores(e1, 1, "chr20", c20r, radius=2e6)
 sapply(chkcc, max)
}
setwd(curd)
#
# additional examples are in the 'extras' folder, extrExt.txt
#
```

---

ex6 *example exon region data*

Description

example exon region data

Usage

```
data(ex6)
```

Format

The format is: Formal class 'GRanges' [package "GenomicRanges"] with 7 slots ..@ seqnames :Formal class 'Rle' [package "IRanges"] with 5 slots .. .. ..@ values : Factor w/ 49 levels "chr1","chr1_random",..: 36 .. .. ..@ lengths : int 12974 .. .. ..@ elementMetadata: NULL .. .. ..@ elementType : chr "ANY" .. .. ..@ metadata : list() ..@ ranges :Formal class 'IRanges' [package "IRanges"] with 6 slots .. .. ..@ start : int [1:12974] 237101 249628 256880 280114 290854 293103 293769 293769 295822 336752 ... .@ width : int [1:12974] 460 34 83 50 75 172 73 2585 534 58 ... .@ NAMES : NULL .. .. ..@ elementMetadata: NULL .. .. ..@ elementType : chr "integer" .. .. ..@ metadata : list() ..@ strand :Formal class 'Rle' [package "IRanges"] with 5 slots .. .. ..@ values : Factor w/ 3 levels "+","-","*": 1 2 .. .. ..@ lengths : int [1:2] 6235 6739 .. .. ..@ elementMetadata: NULL .. .. ..@ elementType : chr "ANY" .. .. ..@ metadata : list() ..@ seqlengths : Named int [1:49] 247249719 1663265 135374737 113275 134452384 215294 132349534 114142980 186858 106368585 ... .- attr(*, "names")= chr [1:49] "chr1" "chr1_random" "chr10" "chr10_random" ... ..@ elementMetadata:Formal class 'DataFrame' [package "IRanges"] with 6 slots .. .. ..@ rownames : NULL .. .. ..@ nrows : int 12974 .. .. ..@ elementMetadata: NULL .. .. ..@ elementType : chr "ANY" .. .. ..@ metadata : list() .. .. ..@ listData :List of 1 .. ..$ exon_id: int [1:12974] 81518 81519 81520 81521 81522 81523 81524 81526 81525 81527 ... ..@ elementType : chr "ANY" ..@ metadata : list()

Examples

```
data(ex6)
ex6[1:4]
## maybe str(ex6) ; plot(ex6) ...
```

---

exome_minp *acquire minimum p-value for association between genotype and expression*

Description

acquire minimum p-value for association between genotype and expression in context of exome genotyping – where a list of SNPs associated with genes or exons governs organization of tests, and minimum p-value per gene or exon is all that is required

Usage

```
exome_minp(smlSet, fmla, targdir, runname, snpl, feat=NULL, mgr = NULL, scoreApp
```

Arguments

smlSet	basic genotype plus expression structure; this must have an smList() result of length 1 (all SNP in one snp.matrix regardless of number of chromosomes)
fmla	formula expressing covariates to be found in phenoData of smlSet and used in each association model
targdir	folder where ff files will be written
runname	prefix for names of ff files
snpl	a named list, with one element per gene or exon, each element is name of snps assayed for the associated gene or exon; names of list elements are the gene or exon names

feat	name of feature for focused reporting; important if names of features of original smlSet don't agree with names of snpl
mgr	if an eqtlTestsManager (with fflist of length 1) is already available, this can be used instead of constructing one from the smlSet
scoreApply	lapply-like function to be used to compute scores – use mclapply for multicore deployment
...	parameters passed to eqtlTests

Examples

```
data(hmceuB36.2021)
hmlit = hmceuB36.2021[ chrnum(20), ]
library(illuminaHumanv1.db)
cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
indc = which(featureNames(hmlit) == cptag[1])
hm = hmlit[c(indc,1:19),]  # reduce problem
curd = getwd()
td = tempdir()
setwd(td)
sl = colnames(smList(hm)[[1]])[1:80]
sl = split(sl, rep(1:20, each=4))
names(sl) = featureNames(hm)
e1 = exome_minp( hm, ~male, "ex1", "ex1", sl )
e1
```

---

geneRanges *construct a RangedData instance for genes enumerated according to an annotation .db package*

Description

construct a RangedData instance for genes enumerated according to an annotation .db package

Usage

```
geneRanges(ids, annopkg, extend = 0)
```

Arguments

ids	character vector
annopkg	package that includes CHR, CHRLOC and CHRLOCEND maps for tokens in ids
extend	atomic number of bases to extend ranges from start upstream and from end downstream

Details

if no location is available, start is set to 1 and end is set to 2, regardless of value of extend

Value

RangedData-class instance

Author(s)

VJ Carey

Examples

```
library(illuminaHumanv1.db)
gg = get(c("CPNE1", "BRCA2"), revmap(illuminaHumanv1SYMBOL))
geneRanges(gg, "illuminaHumanv1.db")
```

---

geneTrack — *create a RangedData structure with multffCT test results (as -log10 p values by default)*

Description

create a RangedData structure with multffCT test results (as -log10 p values by default)

Usage

```
geneTrack(mgr, gn, chrtag, locdata, dropDups = TRUE, mlog10p = TRUE, minchisq =
```

Arguments

mgr	an instance of multffManager-class.
gn	a character string naming a 'gene' (typically name for a microarray probe)
chrtag	the name of the chromosome for which SNP scores are desired, names of mgr[["fflist"]]
locdata	a RangedData instance with ranges defining SNP locations (0 width) and names giving rs numbers or any other SNP identifiers indexing rows in mgr[["fflist"]] ff matrices.
dropDups	logical: should duplicated SNP regions be dropped?
mlog10p	logical: should the score generated be -10 log p (if FALSE, the chi-squared variate with mgf[["df"]] degrees of freedom is used)
minchisq	ignore

Value

The structure provided as locdata is filtered for SNP that are tested in mgr and scores are added in score element

export using rtracklayer to visualize series of scores on genomic coordinates

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
# runs interactively but not in check on windows
if (.Platform$OS.type != "windows") {
sessionInfo()
example(multffCT)
dems
g1 = colnames(dems$fflist[[1]])[1]
data(snpLocs_21)
sco = geneTrack( dems, g1, "21", snpLocs_21 )
sco
library(rtracklayer)
export(sco, con=paste(g1, ".wig", sep=""))
readLines(paste(g1, ".wig", sep=""), n=10)
#
# now add to genome browser as a custom track
#
# if you want to modify aspects of the display as a track, use, e.g.,
# nsco = as(sco, "UCSCData")
# nsco@trackLine@name = "[genename]"  etc.
}
```

`getGRanges` *acquire a GRanges instance with eQTL scores for all SNP for a given gene*

Description

acquire a GRanges instance with eQTL scores for all SNP for a given gene

Usage

```
getGRanges(mgr, ffind, geneind, seqnames, namedlocs)
```

Arguments

`mgr`	instance of `eqtlTestsManager-class`
`ffind`	index into the list of `ff` files that are being managed, typically a sequential index into the list of chromosomes analyzed
`geneind`	identifier or index of gene of interest
`seqnames`	chromosome name
`namedlocs`	a named vector of integer locations, as generated by `getNamedLocs` for example

Examples

```
if (require("SNPlocs.Hsapiens.dbSNP.20100427")) {
 if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
 library(illuminaHumanv1.db)
 cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
 indc = which(featureNames(hmceuB36.2021) == cptag[1])
 hm = hmceuB36.2021[c(indc,1:19),]  # reduce problem
 td = tempdir()
```

```
 curd = getwd()
 setwd(td)
 time.lapply = unix.time(e1 <- eqtlTests( hm, ~male, targdir="ggr" ))
 ggg = getGRanges(e1, 1, cptag, "ch20",
   getNamedLocs(slpack="SNPlocs.Hsapiens.dbSNP.20100427", chr="ch20"))
 ggg
# now combine with cis ranges for a set of genes
 example(cisRanges)
 cc = cisRanges(m1[1:10], "chr20", "illuminaHumanv1.db")
 seqnames(cc) = "ch20"
 FF = findOverlaps(ggg, cc)
 GG = split(FF@matchMatrix[,1], FF@matchMatrix[,2])
 names(GG) = m1[1:10]
 sapply(GG,length)
 myl = lapply(GG, function(x) ggg[x, ])
 names(myl)
 sapply(myl, function(x) max(elementMetadata(x)$score))

 }
```

`getNamedLocs` *get a named vector of SNP locations*

Description

get a named vector of SNP locations

Usage

```
getNamedLocs(slpack = "SNPlocs.Hsapiens.dbSNP.20100427", chrtok)
```

Arguments

`slpack`	location package name
`chrtok`	string giving the token used to obtain a chromosome's worth of SNP locations

Note

should eventually give way to GRanges processing

Examples

```
nn = getNamedLocs(chr="ch20")
length(nn)
nn[1:5]
```

GGtools-package *GGtools Package Overview*

Description

GGtools Package Overview

Details

This package provides facilities for analyzing relationships between gene expression distributions (singly or in groups) and SNP genotype series (chromosome-specific or genome-wide). The gwSnpTests method is the primary interface.

Important data classes in use: smlSet-class, gwSnpScreenResult-class, defined in GGBase package.

Main data sets: hmceuB36.2021, an excerpt based on chromosomes 20 and 21, with genotypes for all phase II HapMap SNP and full expression data for 90 CEU HapMap cohort members.

Introductory information is available from vignettes, type openVignette().

Full listing of documented articles is available in HTML view by typing help.start() and selecting GGtools package from the Packages menu or via library(help="GGtools").

Author(s)

V. Carey

gwSnpTests *methods for iterating association tests (expression vs SNP) across genomes or chromosomes*

Description

methods for iterating association tests (expression vs SNP) across genomes or chromosomes

Usage

```
gwSnpTests(sym, sms, cnum, cs, ...)
```

Arguments

sym	genesym, probeId, or formula instance
sms	smlSet instance
cnum	chrnum instance or missing
cs	chunksize specification
...	...

Details

invokes snpMatrix package test procedures (e.g., snp.rhs.tests as appropriate

chunksize can be specified to divide task up into chunks of chromosomes; gc() will be run between each chunk – this may lead to some benefits when memory capacity is exceeded

The dependent variable in the formula can have class genesym (chip annotation package used for lookup), probeId (direct specification using chip annotation vocabulary), or phenoVar (here we use a phenoData variable as dependent variable). If you want to put expression values on the right-hand side of the model, add them to the phenoData and enter them in the formula.

Value

gwSnpScreenResult-class or cwSnpScreenResult-class instance

Author(s)

Vince Carey <stvjc@channing.harvard.edu>

Examples

```
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
# condense to founders only
hmFou = hmceuB36.2021[, which(hmceuB36.2021$isFounder)]
# show basic formula fit
f1 = gwSnpTests(genesym("CPNE1")~male, hmFou, chrnum(20))
f1
#The following code will create a view of the UCSC
#genome browser:
#if (interactive()) {
#library(rtracklayer)
#f1d = as(f1, "RangedData")
#s1 = browserSession("UCSC")
#s1[["CPNE1"]] = f1d
#v1 = browserView(s1, GenomicRanges(30e6, 40e6, "chr20"), full="CPNE1")
#}
# R-based visualization
plot(f1)
# show how to avoid adjusted fit
f1b = gwSnpTests(genesym("CPNE1")~1-1, hmFou, chrnum(20))
# show gene set modeling on chromosome
library(GSEABase)
gs1 = GeneSet(c("CPNE1", "ADA"))
geneIdType(gs1) = SymbolIdentifier()
f2 = gwSnpTests(gs1~male, hmFou, chrnum(20))
f2
names(f2)
plot(f2[["ADA"]])
# show 'smlSet-wide' fit
f3 = gwSnpTests(gs1~male, hmFou)
f3
# now use a phenoVar
f3b = gwSnpTests(phenoVar("persid")~male, hmFou, chrnum(20))
topSnps(f3b)
## Not run:
# in example() we run into a problem with sys.call(2); works
# in interpreter
```

```
f4 = gwSnpTests(gs1~male, hmFou, snpdepth(250), chunksize(1))
f4
#

## End(Not run)
# illustrate alternate approach to expression feature enumeration
#
data(smlSet.example)
esml = as(smlSet.example, "ExpressionSet")
library(genefilter)
annotation(esml) = "illuminaHumanv1" # drop .db
library(illuminaHumanv1.db)
fesml = nsFilter(esml)[[1]] # unique entrez ids + other filters
fn = featureNames(fesml)
eids = unlist(mget(fn, illuminaHumanv1ENTREZID))
featureNames(fesml) = as.character(eids)
fesml = make_smlSet( fesml, smList(smlSet.example) )
# now we have an smlSet with Entrez ID featureNames
annotation(fesml) = "org.Hs.eg"
mygs = GeneSet(c("ZNF253", "MRS2"), geneIdType = SymbolIdentifier())
geneIdType(mygs) = AnnotationIdentifier("org.Hs.eg")
tt = gwSnpTests(mygs~male, fesml)
lapply(tt, topSnps)
```

`hla2set` *a gene set of 9 genes from human HLA2 locus*

Description

a gene set of 9 genes from human HLA2 locus

Usage

```
data(hla2set)
```

Format

The format is: Formal class 'GeneSet' [package "GSEABase"] with 13 slots

..@ geneIdType :Formal class 'SymbolIdentifier' [package "GSEABase"] with 2 slots

.. .. ..@ type :Formal class 'ScalarCharacter' [package "Biobase"] with 1 slots

and so on.

See `GeneSet-class` for additional information.

Details

This set of 9 genes related to human HLA2 locus was used in the 2009 Bioinformatics Application Note by Carey, Davis et al.

Examples

```
data(hla2set)
if (require(GSEABase)) {
 geneIds(hla2set)
}
```

hmceuB36.2021 *two chromosomes of genotype data and full expression data for CEPH CEU hapmap data*

Description

two chromosomes of genotype data and full expression data for CEPH CEU hapmap data

Usage

```
data(hmceuB36.2021)
```

Format

The format is: Formal class 'smlSet' [package "GGBase"] with 9 slots

..@ smlEnv :<environment: 0x3902e98>

..@ annotation : chr "illuminaHumanv1.db"

..@ chromInds : num [1:2] 20 21

..@ organism : chr "Hs"

..@ assayData :<environment: 0x3c96504>

..@ phenoData :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots

..@ featureData :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots

..@ experimentData :Formal class 'MIAME' [package "Biobase"] with 13 slots

..@ ...classVersion..:Formal class 'Versions' [package "Biobase"] with 1 slots

Examples

```
data(hmceuB36.2021)
validObject(hmceuB36.2021)
```

imphm3_1KG_20 *snpMatrix-generated rules from imputing from HapMap phase III loci to 1000 genomes loci – for chromosome 20 only*

Description

snpMatrix-generated rules from imputing from HapMap phase III loci to 1000 genomes loci – for chromosome 20 only

Usage

```
data(imphm3_1KG_20_mA2)
```

Format

The format is: Formal class 'snp.reg.imputation' [package "snpMatrix"] with 1 slots
..@ .Data:List of 110511
.. ..$:List of 4
.. .. ..$ maf : num 0.2
.. .. ..$ r.squared : num 1
.. .. ..$ snps : chr "rs6139074"
.. .. ..$ coefficients: num [1:2] 0 1
.. ..$:List of 4
.. .. ..$ maf : num 0.117
.. .. ..$ r.squared: num 0.892
.. .. ..$ snps : chr [1:3] "rs13043000" "rs17685809" "rs1935386"
.. .. ..$ hap.probs: num [1:16] 3.01e-01 6.97e-22 1.56e-02 2.36e-20 8.49e-03 ...
.. ..$: NULL

Details

Generated with snpMatrix 1.13.3, rules that use the ceu1kg package to define loci and calls for 1000 genomes genotypes for CEU, to allow imputation from the hapmap phase III loci for CEU. The data object with suffix mA2 was generated with setting mA=2; for suffix mA5, mA was set at 5; see `snp.imputation` for details on this parameter, which sets the minimum number of observations required for an LD determination to be made for SNP tagging or haplotype modeling.

Source

ceuhm3 package was used to define the hapmap phase III loci; locations derived from SNPlocs.Hsapiens.dbSNP.2009050
ceu1kg package includes metadata and calls derived from the 1000 genomes pilot phase 1 VCF file for CEU.

Examples

```
data(imphm3_1KG_20_mA2)
imphm3_1KG_20_mA2[1:10]
```

---

m20 — *snpMatrix (1.3.13) with imputed genotypes for 110 HapMap phase III samples from CEU population*

Description

snpMatrix (1.3.13) with imputed genotypes for 110 HapMap phase III samples from CEU population

Usage

```
data(m20)
```

Format

The format is: Formal class 'snp.matrix' [package "snpMatrix"] with 1 slots
..@ .Data: raw [1:110, 1:190473] 03 03 03 03 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$: chr [1:110] "NA06984" "NA06989" "NA12340" "NA12341" ...
.. .. ..$: chr [1:190473] "rs6078030" "rs4814683" "rs34147676" "rs6139074" ...

Details

results of MACH applied by Blanca Himes of Channing Laboratory, leading to an mlprob file read with read.mach()

Source

The HapMap phase III genotypes were obtained as `hapmap3_r2_b36_fwd.CEU.qc.poly.[ped/map]` as distributed at hapmap.org

Examples

```
data(m20)
```

---

makeCommonSNPs — *confine the SNPs (in multiple chromosomes) in all elements of a list of smlSets to the largest shared subset per chromosome; test for satisfaction of this condition*

Description

confine the SNPs (in multiple chromosomes) in all elements of a list of smlSets to the largest shared subset per chromosome; test for satisfaction of this condition

Usage

```
makeCommonSNPs(listOfSms)
checkCommonSNPs(listOfSms)
```

Arguments

`listOfSms`	an R list with each element consisting of a `smlSet-class`

Details

intersection of set of rsids per chromosome is computed over all elements

Value

list of smlSet instances sharing all SNP on all chromosomes

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
data(smlSet.example)
tmp = smList(smlSet.example)[[1]]
tmp = tmp[,-c(20:40)]
newe = new.env()
assign("smList", list(`21`=tmp), newe)
ex2 = smlSet.example
ex2@smlEnv = newe
try(checkCommonSNPs(list(smlSet.example,ex2)))
list2 = makeCommonSNPs( list(smlSet.example, ex2) )
checkCommonSNPs(list2)
```

---

manhPlot *manhattan plot for an eqtlTests result*

Description

manhattan plot for an eqtlTests result

Usage

```
manhPlot(probeid, mgr, ffind, namedlocvec = NULL, locGRanges = NULL, plotter = s
```

Arguments

probeid	element of colnames of fflist(mgr)[[ffind]] – the gene of interest, typically
mgr	an instance of eqtlTestsManager
ffind	index of the ff file of interest – typically identifying a chromosome where SNP locations define the x-axis of the plot
namedlocvec	a vector with named elements, giving SNP locations
locGRanges	a GRanges instance with SNP locations
plotter	function to be used for rendering
tx	the numbers acquired from the manager are assumed to be chi-squared(1) – this function can be altered to define how the y axis is derived from manager contents
xlab	label for x axis
ylab	label for y axis
...	passed to plotting function

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
## Not run:
if (!exists("e1")) example(eqtlTests)  # creates e1, cptag, c20r
# use ffind=1 below because you have confined attention to chr20
manhPlot( cptag, e1, ffind=1, locGRanges=c20r, cex=3)
setwd(curd)

## End(Not run)
```

maxchisq-class *Class "maxchisq"*

Description

container for results of cis-trans eQTL searches, and a p-value extractor

Objects from the Class

Objects can be created by calls of the form `new("maxchisq", ...)`.

Slots

`.Data`: Object of class `"list"` currently representation is simple – a named list of named vectors of chisquared statistics corresponding to SNP, a value for the d.f. of the chisq stats, the gene for which chisq was maximized for each SNP, and some production metadata. Note that a type parameter allows computation of max chisq stats per SNP (over genes) or per gene (over SNP)

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2. Class `"AssayData"`, by class "list", distance 2. Class vectorORfactor, by class "list", distance 3.

Methods

min_p_vals `signature(mcs = "maxchisq", mtcorr = "character", type = "character", sidedness="numeric")`: Note: owing to a namespace complication with 'update' in multtest package, the mtcorr parameter is ignored; corrections to p-values generated with this tool need to be computed 'manually' at this time.

`mtcorr` is the `proc` token for `mt.rawp2adjp`. Specifically, if mtcorr is set to `"BH"`, the Benjamini-Hochberg FDR transformation is applied. If mtcorr is set to `"none"`, nothing is done.

`type` determines the scope of the corrections. Options are `""` which must be used if `mtcorr` is `"none"`, `"chr_specific"`, with which the testing corrections are made within chromosomes, or `"global"`, with which the testing corrections are made over all tests over the whole genome.

sidedness determines whether a 2 sided (2*(1-pchisq)) or 1 sided p-value is returned. supply the factor 1 or 2 as desired.

show `signature(object = "maxchisq")`: concise but informative report

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
showClass("maxchisq")
# also see example(diagffCC) for illustrations
```

mkCisTransDirector *Create an object that manages a collection of eqtlTestManagers*

Description

Create an object that manages a collection of eqtlTestManagers

Usage

```
mkCisTransDirector(dl, indexdbname, snptabname, probetabname, probeanno, commonS
```

Arguments

dl	list of eqtlManager instances
indexdbname	scalar character used to distinguish the director
snptabname	name to be used for the index of snp to chromosomes
probetabname	name to be used for the index of probes to managers
probeanno	platform annotation package name, e.g., "illuminaHumanv1.db"
commonSNPs	logical indicating whether all managers cover the same collection of SNPs

Details

Creates two ff files that serve as indexes: one for snp id to fflist element for managers, and one for gene id to manager.

Value

instance of cisTransDirector class

Author(s)

VJ Carey <stvjc@channing.harvard.edu?

Examples

```
# see example(eqtlTests)
```

multffCT *parallelized multipopulation cis-trans eQTL searches*

Description

run a parallelized cis-trans eQTL search

Usage

```
multffCT(listOfSms, gfmlaList, geneinds = 1:10, harmonizeSNPs = FALSE, targdir =
   ncores = 2, mc.set.seed=TRUE, vmode = "single", shortfac=100, ...)
```

Arguments

listOfSms	list of smlSet-class instances
gfmlaList	list of formulas (associated one to one with components of listOfSms) with dummy dependent variable and variables on right-hand side drawn from pData of listOfSms, to be passed to snp.rhs.tests
geneinds	object inheriting from numeric or probeId-class to enumerate genes for analysis
harmonizeSNPs	logical indicating whether to skip the call to makeCommonSNPs for the listOfSms
targdir	path to location where ff files will be written
runname	tag to be used in ff filenames and for ultimate control object to be serialized
overwriteFF	logical indicating whether preexisting ff files with names to be used in this run should be overwritten (by default they are)
fillNA	logical indicating whether array elements corresponding to missing tests should be filled with independent chisquared df 1. Note that concrete reproducibility of sets of scores that are randomly generated is not achieved if mc.set.seed=TRUE, which is the default value.
ncores	maximum number of cores to be used by mclapply
mc.set.seed	as passed to mclapply
vmode	mode for numeric storage in ff files, see vmode. If you use "short", the "shortfac" will multiply the chisquares so that integer storage retains some precision (if shortfac = 100, you have two digits beyond the decimal point; the short can only represent 0-32767.) More infrastructure is needed for downstream handling of the short representation, but it seems worthwhile.
shortfac	quantity by which short ints will be inflated for storage to allow more precision in usage
...	additional arguments for passage to snp.rhs.tests

Details

function constructs nchrom ff files holding sums of chisquared tests across smlSets supplied in listOfSms, and serializes metadata about them and the run in [runname].rda.

Value

a list for inspection, but key result is side effect of writing ff files and serializing their metadata

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
# runs interactively but not in check on windows
if (.Platform$OS.type != "windows") {
if (require(ff)) {
 data(smlSet.example)
 sessionInfo()
 td = tempdir()
 od = getwd()
```

```
setwd(td)
set.seed(1234)
dem = multtffCT( list(smlSet.example, smlSet.example), list(gs~male, gs~male), 1:3, runna
set.seed(1234)
dems = multtffCT( list(smlSet.example, smlSet.example), list(gs~male, gs~male),
   1:3, vmode="short", shortfac=100, runname="dem2" )
#
# note that chisq fillin of missing snps make strict numerical reproducibility
# nontrivial
dem
dems
dir()
setwd(od)
}
}
```

`multtffManager-class`
Class "multtffManager"

Description

coordinates access to and interrogation of multipopulation eQTL searches

Objects from the Class

Objects can be created by calls of the form `new("multtffManager", ...)`. These extend list during the experimental development phase.

Slots

`.Data`**:** Object of class `"list"` ~~

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2. Class `"AssayData"`, by class "list", distance 2. Class `vectorORfactor`, by class "list", distance 3.

Methods

show `signature(object = "multtffManager")`: concise report that provides an excerpt from the ff image

[`signature(x = "multtffManager"), i, j, ...`: you can extract results by rsid or probeId with customary bracket semantics, with the exception that if the SNP request spans multiple chromosomes, you will get a list of results

Note

> names(dd)

Currently components of .Data are

`fflist` a list of ff references, to tables holding sums of chi-squared statistics accumulated across populations

`call` for auditing, the initial call

`runname` an arbitrary user-supplied tag

`targdir` the folder used to write the ff files

`generangetag` a generated tag giving the scope of the gene set used for searches

`filenames` a character vector of the ff file paths

`df` numeric value of the number of populations summed

`vmode` ff specification of virtual mode of data values; if 'short', rescale using shortfac

`shortfac` factor by which chisquared deviates were multiplied so that a short int can represent without too much coarsening

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
#
# seems to throw file error in CMD check on windows
#
if (.Platform$OS.type != "windows") {
  example("multffCT")
  dem
  getClass(class(dem))
  dem$fflist[[1]]
  dem$df
  dem$filenames
  dem$vmode
  dem$call
  }
```

`permEx` *permute expression data against genotype data in an smlSet*

Description

permute expression data against genotype data in an smlSet

Usage

```
permEx(sms)
```

Arguments

`sms`	an instance of `smlSet-class`

Value

an instance of `smlSet-class`

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
library(illuminaHumanv1.db)
cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
indc = which(featureNames(hmceuB36.2021) == cptag[1])
hm = hmceuB36.2021[c(indc,1:19),]  # reduce problem
td = tempdir()
curd = getwd()
setwd(td)
time.lapply = unix.time(e1 <- eqtlTests( hm, ~male, targdir="pex" ))
e1
hmp = permEx(hm)
e1perm = eqtlTests(hmp, ~male, targdir="permfoo", runname="permrun")
topFeats(probeId(cptag), mgr=e1, ffind=1, anno="illuminaHumanv1.db", useSym=FALSE)
topFeats(probeId(cptag), mgr=e1perm, ffind=1, anno="illuminaHumanv1.db", useSym=FALSE)
```

---

plot-methods *Methods for Function plot in Package 'GGtools'*

Description

Methods for function `plot` in Package 'GGtools'

Methods

x = "cwSnpScreenResult", y = "missing" shows results of chromosome-wide screen for expression-associated SNP

x = "filteredGwSnpScreenResult", y = "ANY" shows results of genome-wide screen for expression-associated SNP

x = "filteredMultiGwSnpScreenResult", y = "ANY" fails, need to pick gene at this time

---

scoresInRanges *structured survey of eqtlTestsManager to retrieve scores cis to genes*

Description

structured survey of eqtlTestsManager to retrieve scores cis to genes defined through ranges

Usage

```
scoresInRanges(mgr, geneRanges, snpRanges, applier = lapply)
```

Arguments

`mgr`	instance of `eqtlTestsManager-class`
`geneRanges`	instance of `GRanges-class` representing gene-like regions within which SNPs will be sought for association with expression
`snpRanges`	instance of `GRanges-class` representing SNP locations
`applier`	lapply or mclapply if relevant

Value

a list of lists, one element per gene range, with matrix of chisq scores for SNP

Examples

```
gl = GGtools:::geneLimits( anno="illuminaHumanv1.db" )
egl = GGtools:::extendGR(gl, siz=5e5)
data(hmceuB36.2021)
gl20 = names(egl[ which(seqnames(egl) == "chr20") ] )[1:20]
p2 = intersect( gl20, featureNames(hmceuB36.2021) )
curd = getwd()
setwd(tempdir())
if (file.exists("foo"))system("rm -rf foo")
ee = eqtlTests( hmceuB36.2021[ probeId(p2), ], ~1, targdir="sir" )
library(ceu1kg)
data(ceu1kgMeta_20)
ceu1kgMeta_20 = updateObject(ceu1kgMeta_20)
cc = scoresInRanges( ee, egl[1:10], ceu1kgMeta_20 )
sapply(cc, sapply, length)
maxs = sapply(cc, sapply, max)
maxs
# this is impoverished from a metadata perspective, but
# it is good to keep the code of scoresInRange simple
# let's get the snp names of the best hits
rsnhits = sapply(cc, sapply, function(x) rownames(x)[which.max(x)])
# combine with gene names
data.frame( probe=names(egl[1:10]), bestSNP=rsnhits, maxchisq=maxs )
setwd(curd)
```

---

snp130locs *prototypical function for creation of IRanges-based SNP location data*

Description

prototypical function for creation of IRanges-based SNP location data

Usage

```
snp130locs(chr, start, end)
```

Arguments

chr	scalar string with prefix "chr"
start	numeric start value, typically 1
end	numeric end value, typically length in bases of chromosome

Value

a ucscTableQuery output

Examples

```
## The function is currently defined as
function (chr, start, end)
{
    sess = browserSession()
    quer = ucscTableQuery(sess, "snp130", GenomicRanges(start,
        end, chr))
    tableName(quer) = "snp130"
    track(quer)
  }
```

`snpLocs20` *prototype SNP location instance for use with GGtools*

Description

prototype SNP location instance for use with GGtools

Usage

```
data(snpLocs20)
```

Format

The format is: Formal class 'UCSCData' [package "rtracklayer"] with 6 slots ..@ trackLine :Formal class 'BasicTrackLine' [package "rtracklayer"] with 12 slots
.. .. ..@ itemRgb : logi(0)
.. .. ..@ useScore : logi(0)
.. .. ..@ group : chr(0)
.. .. ..@ db : chr(0)
.. .. ..@ offset : num(0)
.. .. ..@ url : Named chr " "
.. ..- attr(*, "names")= chr "url"
.. .. ..@ htmlUrl : chr(0)
.. .. ..@ name : Named chr "snp130"
.. ..- attr(*, "names")= chr "name"
.. .. ..@ description: Named chr "snp130"
.. ..- attr(*, "names")= chr "description"
.. .. ..@ visibility : Named chr "1"
.. ..- attr(*, "names")= chr "visibility"
.. .. ..@ color : int(0)
.. .. ..@ priority : num(0)
..@ ranges :Formal class 'CompressedIRangesList' [package "IRanges"] with 5 slots
.. .. ..@ elementMetadata: NULL
.. .. ..@ elementType : chr "IRanges"
.. .. ..@ metadata :List of 1
.. ..$ universe: chr "hg18"
.. .. ..@ partitioning :Formal class 'PartitioningByEnd' [package "IRanges"] with 5 slots
.. ..@ end : int 450693
.. ..@ NAMES : chr "chr20"
.. ..@ elementMetadata: NULL

.. ..@ elementType : chr "integer"
.. ..@ metadata : list()
.. .. ..@ unlistData :Formal class 'IRanges' [package "IRanges"] with 6 slots
.. ..@ start : int [1:450693] 60492 60572 60646 60705 61098 61605 61795 62100 62291 62731 ...
.. ..@ width : int [1:450693] 1 1 1 0 1 1 1 1 0 1 ...
.. ..@ NAMES : NULL
.. ..@ elementMetadata: NULL
.. ..@ elementType : chr "integer"
.. ..@ metadata : list()
..@ values :Formal class 'CompressedSplitDataFrameList' [package "IRanges"] with 5 slots
.. .. ..@ elementMetadata: NULL
.. .. ..@ elementType : chr "DataFrame"
.. .. ..@ metadata : list()
.. .. ..@ partitioning :Formal class 'PartitioningByEnd' [package "IRanges"] with 5 slots
.. ..@ end : int 450693
.. ..@ NAMES : chr "chr20"
.. ..@ elementMetadata: NULL
.. ..@ elementType : chr "integer"
.. ..@ metadata : list()
.. .. ..@ unlistData :Formal class 'DataFrame' [package "IRanges"] with 6 slots
.. ..@ rownames : NULL
.. ..@ nrows : int 450693
.. ..@ elementMetadata: NULL
.. ..@ elementType : chr "ANY"
.. ..@ metadata : list()
.. ..@ listData :List of 3
.. ..$ name : chr [1:450693] "rs35078228" "rs28753379" "rs28579812" "rs35616340" ...
.. ..$ score : num [1:450693] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ strand: chr [1:450693] "+" "+" "+" "+" ...
..@ elementMetadata: NULL
..@ elementType : chr "ANY"
..@ metadata : list()

Details

derived from UCSC table for snp130

snpLocs_21 is in a different format for chromosome 21

Source

snp130 table in hg19 UCSC table set

Examples

```
data(snpLocs20)
snpLocs20
```

`strMultPop` *serialization of a table from Stringer's multipopulation eQTL report*

Description

serialization of a table from Stringer's multipopulation eQTL report

Usage

```
data(strMultPop)
```

Format

A data frame with 39649 observations on the following 12 variables.

`rsid` a factor with levels rs...

`genesym` a factor with levels `37865` `39692` `ABC1` `ABCD2` `ABHD4` `ACAS2` ...

`illv1pid` a factor with levels `GI_10047105-S` `GI_10092611-A` `GI_10190705-S` `GI_10567821-S` `GI_10835118-S` `GI_10835186-S` ...

`snpChr` a numeric vector

`snpCoordB35` a numeric vector

`probeMidCoorB35` a numeric vector

`snp2probe` a numeric vector

`minuslog10p` a numeric vector

`adjR2` a numeric vector

`assocGrad` a numeric vector

`permThresh` a numeric vector

`popSet` a factor with levels `CEU-CHB-JPT` `CEU-CHB-JPT-YRI` `CHB-JPT`

Details

imported from the PDF(!) distributed by Stranger et al as supplement to PMID 17873874

Source

PMID 17873874 supplement

References

PMID 17873874 supplement

Examples

```
data(strMultPop)
strMultPop[1:2,]
```

topSnps-methods *report on most significant SNP with gwSnpTests results*

Description

report on most significant SNP with gwSnpTests results

Methods

x = "cwSnpScreenResult" also takes argument n for number to report

x = "gwSnpScreenResult" also takes argument n for number to report

GGtools-RangedData *Transform results of gwSnpTests to browser tracks*

Description

Create a browser track from a chromosome-wide SNP screen

Coercion

`as(object, "RangedData")`: Coerce a `cwSnpScreenResult`, `object`, to a `RangedData` instance, with the genomic coordinates -log10 p-values for each SNP

vcf2sm *generate a snp.matrix instance on the basis of a VCF (4.0) file*

Description

generate a snp.matrix instance on the basis of a VCF (4.0) file

Usage

```
vcf2sm(gzpath, chrom, tabixcmd = "tabix", nmetacol = 9, verbose = FALSE, gran=10
```

Arguments

`gzpath`	string: path to a gzipped vcf file
`chrom`	string: chromosome for processing; use tabix -l to obtain the list of tokens if necessary
`tabixcmd`	string: assumes tabix available as an executable utility; tells the absolute path for invoking the command
`nmetacol`	numeric: tells number of columns used in each record as locus-level metadata
`verbose`	logical: if TRUE, provide processing info
`gran`	numeric: a report is given once every gran snp are traversed to show progress

Value

an instance of snp.matrix-class

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

References

http://www.1000genomes.org/wiki/doku.php?id=1000_genomes:analysis:vcf4.0

Examples

```
# requires tabix
## Not run:
 vref = system.file("vcf/ex.vcf.gz", package="GGtools")
 vcf2sm( vref, "20" )

## End(Not run)
```

X2chunk *compute numerical matrix of chisq statistics in a genomic interval; extract features as requested*

Description

compute numerical matrix of chisq statistics in a genomic interval (rows are SNP, columns are genes), or extract features

Usage

```
X2chunk(mgr, ffind, start, end, snplocs, anno, useSym)
topFeats( x, ... )
# additional potential args include
#       mgrOrCTD, ffind, anno, n=10, useSym=TRUE, minMAF=0, minGTF=0 )
```

Arguments

x	for topFeats, an instance of probeId-class or rsid-class or genesym classes; this is an API change because of odd logic of old function; to use old behavior, call GGtools:::.topFeats
mgr	an instance of multffManager
mgrOrCTD	an instance of multffManager or a cisTransDirector instance
ffind	the index of the ff structure to use (typically chromosome number)
start	left end of interval of interest
end	right end of interval of interest
snplocs	location structure for SNP (RangedData instance)
n	for topFeats, the number of features to report

anno	name of a gene annotation package resolving the identifiers used in column names of ff matrix
useSym	logical indicating whether colnames of return should be gene symbols derived from anno
minMAF	numeric lower bound on minor allele frequency of SNPs to be considered
minGTF	numeric lower bound on minimum genotype frequency of SNPs to be considered
...	see comment in USAGE and entries above

Details

X2chunk will obtain RAM resources for material on disk, so use with caution

Note that gene symbols may map to multiple probes. The first hit is used by topFeats when used with sym=.

Author(s)

VJ Carey

Examples

```
# build an smlSet with a small set of neighboring genes
data(snpLocs20)
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
library(illuminaHumanv1.db)
gOn20 = get("20", revmap(illuminaHumanv1CHR))
gLocs = geneRanges(gOn20, "illuminaHumanv1.db")
start = 10000000
end = 13500000
g2use_inds = which(ranges(gLocs)$chr20 %in% IRanges(start,end))
g2use_names = gLocs[g2use_inds,]$name
h20 = hmceuB36.2021[ probeId(g2use_names), ]
h20 = h20[chrnum(20),]
sn2use_inds = which(ranges(snpLocs20)$chr20 %in% IRanges(start,end))
od = getwd()
setwd(tempdir())
# create the ff manager instance
library(ff)
dd = diagffCC(h20, gs~male)
# extract the matrix
fc = X2chunk(dd, 1, start, end, snpLocs20, "illuminaHumanv1.db")
dim(fc)
fc[1:4,1:5]
setwd(od)
heatmap(fc[1:50,], Rowv=NA, Colv=NA, scale="none")
topFeats( rsid("rs6094162"), mgr=dd, 1, "illuminaHumanv1.db")
topFeats( genesym("MKKS"), mgr=dd, 1, "illuminaHumanv1.db")
```

Index