GGtools

October 25, 2011

`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

`X2chunk` *compute numerical matrix of chisq statistics in a genomic interval;*

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` or eqtlTestsManager 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

```
## Not run:
# 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 = eqtlTests(h20, ~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")

## End(Not run)
```

`cisProxScores-class` *Class "cisProxScores"*

Description

extends list to manage collections of eQTL test scores

Objects from the Class

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

Slots

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

`call`: Object of class `"call"` ~~

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 = "cisProxScores")`: concise report

Examples

```
showClass("cisProxScores")
```

cisProxScores *create, combine, and harvest eqtlTestsManager instances to collect all*

Description

create, combine, and harvest eqtlTestsManager instances to collect all eQTL tests satisfying certain gene proximity conditions

Usage

```
cisProxScores(smlSet, fmla, dradset, direc = NULL, folder, runname, geneApply =
 geneGRL=NULL, snpannopack="SNPlocs.Hsapiens.dbSNP.20100427", ffind=NULL, ...)

mcisProxScores (listOfSmlSets, listOfFmlas, dradset, direc = NULL,
    folder, runname, geneApply = mclapply, saveDirector = TRUE,
    makeCommonSNPs = FALSE, snpGRL=NULL,
    geneGRL=NULL, snpannopack="SNPlocs.Hsapiens.dbSNP.20100427", ffind=NULL, ...

interleave2cis( cisp, permcisp )
```

Arguments

`smlSet`	instance of `smlSet-class`
`fmla`	the right-hand side of a standard modeling formula – no dependent variable; the expression values in the `smlSet` will be used successively as dependent variables
`dradset`	a numeric vector indicating the boundaries within which test scores will be tabulated. For example, if `dradset` is `c(5000,10000,25000)` then scores will be tabulated for SNP in the regions (0-5kb) from start or end of gene, (5-10kb), (10-25kb).
`direc`	an instance of `multiCisDirector-class`; if non-null, `eqtlTests` will not be run, but the tests managed by managers in the `direc` instance will be used
`folder`	used to set `targdir` parameter when `eqtlTests` is run; ignored if `direc` is non-null
`runname`	used to set `runname` parameter when `eqtlTests` is run; some mangling will be applied. Ignored if `direc` is non-null
`geneApply`	iteration function (like `lapply`) to be used for each expression probe (gene); passed to `eqtlTests`; the setting is also used for some annotation-based iterations; if multicore package is present, setting this parameter to `mclapply` is advised
`saveDirector`	logical; since it is expensive to compute the `multiCisDirector` that will be harvested, we may want to serialize it; if so set `saveDirector` to TRUE. If set to true the function stores an object with name `paste(folder,"_director",".rda",sep='` in the current working folder.
`...`	arguments passed to `eqtlTests`
`listOfSmlSets`	for `mcisProxScores`, a list of smlSets that are to be sources for eQTL test scores that will be summed

`listOfFmlas`	for `mcisProxScores`, a list of formulas to be used with `snp.rhs.tests`, assumed to be ordered to correspond to elements of `listOfSmlSets`
`makeCommonSNPs`	for `mcisProxScores`, a logical telling whether the sets of SNPs elements of the `listOfSmlSets` should be reduced to their intersection; this can be slow, and can be done externally using the function of the same name.
`snpGRL`	named list of GRanges instances with SNP locations; list element names must coincide with names of smList entries in smlSet
`geneGRL`	named list of GRanges instances with gene extents; list element names must coincide with names of smList entries in smlSet
`snpannopack`	string naming package with SNPlocs information
`cisp`	result of `cisProxScores`
`permcisp`	result of `cisProxScores`
`ffind`	usually 1 for cis applications where one chromosome of SNP is selected at a time

Details

This function computes tests for all same-chromosome eQTL up to the maximum distance given in `dradset` and returns a named list with chi-squared statistics computed by `snp.rhs.tests`

The `interleave2cis` function helps with general comparison of distributions of real scores to distributions obtained after permutation of expression values against genotypes. See the example.

Value

a list with one component per 'radius' derived from `dradset`

each radius-associated component includes a list with one element per chromosome of the SNP data in the `smlSet`

each chromosome-associated sublist includes a list for each gene mapped to the chromosome, with contents a column-vector of test results for all SNP within the radius of the enclosing component; see the example for further concreteness

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

See Also

`eqtlTests`

Examples

```
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
hm = hmceuB36.2021
td = tempdir()
cd = getwd()
on.exit(setwd(cd))
setwd(td)
library(illuminaHumanv1.db)
g20 = intersect(get("20", revmap(illuminaHumanv1CHR)),
   featureNames(hm))[1:10]
```

```
g21 = intersect(get("21", revmap(illuminaHumanv1CHR)),
   featureNames(hm))[1:10]
hm = hm[probeId(c(g20,g21)), ] # restrict to small number of genes
try(unlink("man", recursive=TRUE))  # in tempdir
set.seed(1234)  # necessary for dealing with null imputation of missing
f1 = cisProxScores( hm, ~male, c(5000,10000,25000), folder="man",
   runname="man", geneApply=lapply, ffind=1 )
length(f1)  # number of proximity regions specified in dradset
length(f1[[1]]) # number of chromosomes of SNP data in smlSet
length(f1[[1]][[1]]) # number of genes in smlSet
                     # mapping to first chromosome in smlSet
                     # SNP data
length(f1[[1]][[2]]) # number of genes mapping to second chr...
sapply(f1, function(x)max(unlist(x)))
sapply(f1, function(x)length(x[[1]]))
lapply(f1, function(x)names(x[[1]]))
lapply(f1, function(x)rownames(x[[1]][[1]][[1]]))
set.seed(1234)
try(unlink("pman", recursive=TRUE))  # in tempdir
pf1 = cisProxScores( permEx(hm), ~male, c(5000, 10000, 25000), folder="pman",
  runname="pman", geneApply=lapply, ffind=1)
i1o = interleave2cis( f1, pf1 )
opar = par(no.readonly=TRUE)
par(las=2, mar=c(12, 5, 5, 5))
boxplot(lapply(i1o, unlist), range=0, main="compare observed to expr-permuted eQTL test s
par(opar)
load("man_director.rda")
man_director
## Not run:
set.seed(1234)  # necessary for dealing with null imputation of missing
mm = mcisProxScores( list(hm,hm), list(~male,~male),
  dradset=c(5000,10000,25000), folder="mmm", runname="MMM", ffind=1)

## End(Not run)

setwd(cd)
```

`clipPCs` *simple approach to removal of principal components from smlSet*

Description

simple approach to removal of principal components from smlSet

Usage

```
clipPCs(smlSet, inds2drop, center=TRUE)
```

Arguments

`smlSet`	instance of `smlSet-class`
`inds2drop`	numeric vector of PCs to be eliminated
`center`	logical passed to `prcomp`.

Details

uses SVD and zeroes out selected eigenvalues before reassembly

Value

an smlSet instance with transformed expression data

Examples

```
data(hmceuB36.2021)
library(illuminaHumanv1.db)
g20 = get("20", revmap(illuminaHumanv1CHR))
g20 = intersect(g20, featureNames(hmceuB36.2021))[1:25]
hmc = clipPCs(hmceuB36.2021, 1:4)
hmc = hmc[probeId(g20),]
pcs = prcomp(t(exprs(hmceuB36.2021)))$x
hmr = hmceuB36.2021[ probeId(g20), ]
pData(hmr) = data.frame(pData(hmr), pcs[,1:4])
hmc
f1 = eqtlTests(hmc[chrnum("20"),], ~male, targdir="clipdem")
f2 = eqtlTests(hmr[chrnum("20"),], ~male+PC1+PC2+PC3+PC4, targdir="clipfmla")
f3 = eqtlTests(hmr[chrnum("20"),], ~male, targdir="clipfmlaNOPC")
```

`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) ...
```

---

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

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, genegran=5
```

Arguments

`smlSet`	instance of `smlSet-class`
`rhs`	standard formula without dependent variable; predictors must be found in `pData(smlSet)`
`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

`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`
`genegran`	numeric value of frequency at which gene names will be catted to stdout in case `options()$verbose == TRUE`

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 `ceuhm3`.

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

Note

We are using `ff` to manage the extremely voluminous results of comprehensive eqtl searches with one short int per test. We do not have an approach to handling NA in this framework, so for any nonexistent test result (due for example to monomorphy or total missingness) we impute a value from the null distribution of the test statistic being computed – chisq of one d.f.. There is no practical risk of misinterpreting such results in contexts of interest, but this saves us the complication of dealing with artificial masses of test statistic distributions at zero, for example.

The `topFeats` methods have `minMAF` and `minGTF` parameters to assist in filtering results to SNPs with certain properties; the metadata used for these is stored in a summary ff structure.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
library(ceuhm3)
hm = getSS("ceuhm3", c("chr20", "chr21"))
library(illuminaHumanv1.db)
cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
indc = which(featureNames(hm) == cptag[1])
```

```
#
# get a set of additional genes on chr20
all20 = get("20", revmap(illuminaHumanv1CHR))
g20 = unique(c(all20[1:10], cptag))
#
hm = hm[probeId(g20),]  # reduce problem
hm = hm[ chrnum(c("chr20", "chr21")), ]
td = tempdir()
curd = getwd()
setwd(td)
time.lapply = unix.time(e1 <- eqtlTests( hm, ~male ))
e1
topFeats(probeId("GI_23397697-A"), mgr=e1, ffind=1)
dir("foo")
setwd(curd)
#
# additional examples are in the 'extras' folder, extrExt.txt
#
```

---

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

`summaryList`: Object of class `"list"` that includes references to ff files with per-chromosome MAF and genotype frequency (GTF) statistics per SNP. These summary statistics can be used with the `topFeats` methods.

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")
```

`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 SnpMatrix 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
```

`externalize` *create R package with decomposable smlSet representation*

Description

create R package with decomposable smlSet representation

Usage

```
externalize(smlSet,
  packname,
  author = "Replace Me <auth@a.b.com>",
  maintainer = "Replace Me <repl@a.b.com>")
getSS( packname, chrs )
```

Arguments

`smlSet`	instance of `smlSet-class`
`packname`	arbitrary string naming the package that will hold the externalized representation – this should not coincide with the name of any installed package, as such would be overwritten
`author`	string that should be a valid Author: entry for a DESCRIPTION file
`maintainer`	string that should be a valid Maintainer: entry for a DESCRIPTION file
`chrs`	vector of strings naming chromosomes to be included in the `smlSet-class` instance created by `getSS`

Details

Each `SnpMatrix-class` instance in the `smlEnv` slot of `smlSet` is written to disk in a folder inst/parts of the source package generated by this function. The `ExpressionSet-class` instance in the `smlSet` is isolated and saved as `eset.rda` to the data folder of the source package generated by this function.

`getSS` will construct an `smlSet-class` instance with the expression data and selected chromosomes

Value

instance of `smlSet-class`

Note

The purpose is to avoid loading very large objects as SNP panels grow into the millions. With this approach in-memory images can be chromosome-size, or smaller if desired, depending on the structure of `smList(smlSet)`.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
## Not run:
 data(hmceuB36.2021)
 owd = getwd()
 setwd(tempdir())
 externalize(hmceuB36.2021, "hmdemo")
 system("tar zcvf hmdemo.tar.gz hmdemo")
 install.packages("hmdemo.tar.gz", repos=NULL)
 library(hmdemo)
 getSS("hmdemo", "20")
 setwd(owd)

## End(Not run)
```

`gwSnpTests` *methods for iterating association tests (expression vs SNP) across*

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 `snpStats` 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) -- no longer supported, need to supply location data -- consider eqtlTests/manh
# show how to avoid adjusted fit
f1b = gwSnpTests(genesym("CPNE1")~1-1, hmFou, chrnum(20))
# show gene set modeling on chromosome
## Not run:
library(GSEABase)  # functionality abandoned
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

## End(Not run)
# 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
#
## Not run:   # nice but out of scope
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)

## End(Not run)
```

---

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*

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` *snpStats-generated rules from imputing from HapMap phase III loci to*

Description

snpStats-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 "snpStats"] 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 snpStats 1.1.1, 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` *snpStats (1.1.1) with imputed genotypes for 110 HapMap phase III*

Description

snpStats (1.1.1) with imputed genotypes for 110 HapMap phase III samples from CEU population

Usage

```
data(m20)
```

Format

The format is: Formal class 'SnpMatrix' [package "snpStats"] 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*

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
`suppressGeneLoc`	logical; if true, will refrain from trying to indicate gene location on plot. Important to have TRUE when a trans association is being plotted.
`...`	passed to plotting function

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
if (require(SNPlocs.Hsapiens.dbSNP.20100427)) {
 library(ceuhm3)
 hm = getSS("ceuhm3", "chr20")
 library(illuminaHumanv1.db)
 cptag = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
 indc = which(featureNames(hm) == cptag[1])
 hm = hm[c(indc,1:19),]  # reduce problem
 hm = hm[chrnum("chr20"),] # reduce snp set
 td = tempdir()
 curd = getwd()
 setwd(td)
 e1 <- eqtlTests( hm, ~male, targdir="mplex" )
 c20  = getSNPlocs("ch20", as.GRanges=TRUE)
 sr = ranges(c20)
 sr = GRanges(seqnames="chr20", sr)
 elementMetadata(sr) = elementMetadata(c20)
 names(sr) = paste("rs", elementMetadata(sr)$RefSNP_id, sep="")
# use ffind=1 below because you have confined attention to chr20
 manhPlot( cptag, e1, ffind=1, locGRanges=sr, cex=3)
 setwd(curd)
 }
```

`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

```
## Not run:
# 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 = multffCT( list(smlSet.example, smlSet.example), list(gs~male, gs~male), 1:3, runna
 set.seed(1234)
 dems = multffCT( 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)
}
}

## End(Not run)
```

`multiCisDirector-class` *Class "multiCisDirector"*

Description

manage multiple eqtlTestsManager instances, typically as interim results from a run of cisProxScores

Objects from the Class

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

Slots

`mgrs`: Object of class `"list"` ~~

Methods

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

Note

`makeDiagDirector` is a tool that will generate all same-chromosome eqtlTests from an smlSet instance or package and will create a director of this type.

See Also

`cisProxScores`

Examples

```
showClass("multiCisDirector")
```

`pcChooser` *utility to assist in choosing number of PCs to remove owing to*

Description

utility to assist in choosing number of PCs to remove owing to expression heterogeneity – only cis testing as of jan 2011

Usage

```
pcChooser(sms, cand = c(1, 10, 15, 20, 25, 30, 40), fmla, radius = c(1e+05), chr
 ffind=1, ...)
```

Arguments

`sms`	instance of `smlSet-class`
`cand`	number of PCs to be excluded in successive runs
`fmla`	formula to be used by `cisProxScores`
`radius`	number of basepairs up and downstream from gene boundaries to be checked for eQTL
`chr`	chromosome for current run, for use in space selection for GRanges-associated SNP addressing
`smlc`	name of chromosome in `names(smList(sms))` for this run
`geneApply`	iterator to be used for genes
`pvals`	upper bounds on p-values to declare eQTL present
`ncore`	if set to numeric value, options(cores=ncore) will be executed by this function, useful if geneApply=mclapply
`ffind`	chrom selector passed to cisProxScores, typically default is appropriate choice
`...`	passed to `cisProxScores`

Details

The idea is that we want to maximize the number of eQTL declared, and that there will be diminishing returns as the number of PCs included grows.

Value

matrix with columns corresponding to `cands` and rows corresponding to `pvals` – the row names are the chi-squared threshold values for `snp.rhs.tests` results

Examples

```
data(hmceuB36.2021)
library(illuminaHumanv1.db)
g20 = get("20", revmap(illuminaHumanv1CHR))
g20 = intersect(g20, featureNames(hmceuB36.2021))[1:40]
pcChooser( hmceuB36.2021[probeId(g20),], cand=c(7,9,11), fmla=~male,
  radius=1e6, chr="20", smlc="20", geneApply=lapply, pvals=10^(-c(3:5)))
```

`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

`relocate` *assist in the transport between systems of ff data managed by GGtools*

Description

assist in the transport between systems of ff data managed by GGtools

Usage

```
relocate(old, new, obj, ffind = 1)
```

Arguments

`old`	string to be replaced in the physical filename attribute on old system
`new`	string to be substituted for `old` in the physical filename attribute on old system
`obj`	manager object
`ffind`	index of file in fflist to be altered

Value

a new manager instance

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

Description

serialization of a table from Stranger'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

vcf2sm *generate a SnpMatrix instance on the basis of a VCF (4.0) file*

Description

generate a SnpMatrix instance on the basis of a VCF (4.0) file. NOTE: the tabix utility must be installed and be invocable via system().

Usage

```
vcf2sm(gzpath, chrom, tabixcmd = "tabix", nmetacol = 9, verbose = FALSE, gran=10
   metamax=100, makelocpref="chr")
```

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
`metamax`	number of lines to be sniffed for metadata before real data encountered, could be liberal
`makelocpref`	string telling what to use to construct a locus identifier when the id field is .; sometimes loc field is adequate and this should be set to "". set to "" if you see loc names with chrchr prefix.

Details

This function is relevant only for diallelic SNP. If any base call is denoted '.', the associated genotype is set to missing (raw 0), even if the nonmissing call is ALT, implying at least one ALT.

Value

an instance of SnpMatrix-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
chkTabix = try(system("tabix 2>&1", intern=TRUE))
if (!inherits(chkTabix, "try-error") && length(grep("Option", chkTabix))>0) {
 vref = system.file("vcf/CEU.exon.2010_09.genotypes.vcf.gz", package="GGtools")
 vcf2sm( vref, "1" )
 }
```

Index