GGtools

October 5, 2010

`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

`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")
```

`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]])))
```

`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

```
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)
```

`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$OS.type != "windows") {
example(multffCT)
dems
g1 = colnames(dems$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.
}
```

`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)
```

`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)
```

`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")`: `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
```

`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") {
 data(smlSet.example)
 td = tempdir()
 od = getwd()
 on.exit(unlink(td))
 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()
}
```

`multffManager-class` *Class "multffManager"*

Description

coordinates access to and interrogation of multipopulation eQTL searches

Objects from the Class

Objects can be created by calls of the form `new("multffManager", ...)`. 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 = "multffManager")`: concise report that provides an excerpt from the ff image

[`signature(x = "multffManager"), 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
  }
```

`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

---

`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

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

Index