Package 'gtx'

February 20, 2015

Version 0.0.8

Date 2012-09-20

Title Genetics ToolboX

Author Toby Johnson <Toby.x.Johnson@gsk.com>

Maintainer Toby Johnson <Toby.x.Johnson@gsk.com>

Depends R (>= 2.4.0), survival

Description Assorted tools for genetic association analyses. The current focus is on implementing (either exactly or approximately) regression analyses using summary statistics instead of using subject-specific data. So far, functions exist to support multi-SNP risk score analyses, multi-SNP conditional regression analyses, and multi-phenotype analyses, using summary statistics. There are helper functions for reading and manipulating subject-specific genotype data, which provide a platform for calculating the summary statistics, or for using R to conduct other analyses not supported by specific GWAS analysis tools.

License GPL (>= 2)

Repository CRAN

Date/Publication 2013-01-11 12:26:06

NeedsCompilation no

R topics documented:

gtx-package *Genetics ToolboX*

Description

This package implements assorted tools for genetic association analyses, which is viewed as being entirely an exercise in regressing a (possibly multivariate) phenotypic "response variable" onto one or more "explanatory variables" that include genetic variables.

The current focus of this package is on implementing (either exactly or approximately) regression analyses using summary statistics instead of using subject-specific genotype and phenotype data. So far, functions exist to support three applications detailed below: Multi-SNP risk score analyses; multi-SNP conditional regression analyses; and multi-phenotype analyses.

In addition, there are "helper" functions for reading and manipulating subject-specific genotype and phenotype data, which provide a platform for calculating the necessary summary statistics, and for performing "exact" analyses to validate some of the approximate summary statistic based methods.

The first application is multi-SNP risk score analyses, and the main functions provided for analysing summary statistics are `grs.summary`, `grs.plot` and `grs.filter.Qrs`. The summary statistics necessary for these analyses are single SNP association statistics, which can be calculated using a wide variety of existing tools for GWAS analysis and meta-analysis.

The second application is multi-SNP conditional or multiple regression analyses. The main functions provided for performing multiple regression using summary statistics are `combine.moments2`, `est.moments2`, `lm.moments2` and `stepup.moments2`. The summary statistics necessary for these analyses can be calculated from subject-specific genotype and phenotype data, using the function `make.moments2`.

The third application is multi-phenotype analyses. So far, a single function `multipheno.T2` is provided.

The helper functions for reading and manipulating subject-specific genotype and phenotype data provide a convenient interface from R to genotype data exported from PLINK, and imputed genotype data generated by MACH, minimac, or IMPUTE. The main functions provided are `read.snpdata.plink`, `read.snpdata.mach`, `read.snpdata.minimac`, and `read.snpdata.impute`.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

abf.normal *Calculate approximate Bayes factor (ABF) for normal prior.*

Description

Calculates an approximation to the Bayes Factor for an alternative model where the parameter beta is a priori normal, by approximating the likelihood function with a normal distribution.

Usage

```
abf.normal(beta, se, priorscale, gridrange = 3, griddensity = 20)
```

Arguments

beta	Vector of effect size estimates.
se	Vector of associated standard errors.
priorscale	Scalar specifying the scale (standard deviation) of the prior on true effect sizes.
gridrange	Parameter controlling range of grid for numerical integration.
griddensity	Parameter controlling density of points in grid for numerical integration.

Details

This uses the same normal approximation for the likelihood function as "Bayes factors for genome-wide association studies: comparison with P-values" by John Wakefield, 2009, Genetic Epidemiology 33(1):79-86 at http://dx.doi.org/10.1002/gepi.20359. In that work, an analytical expression for the approximate Bayes factor was derived, which is implemented in abf.Wakefield. This function uses a numerical algorithm has to be used to calculate the (approximate) Bayes factor, which may be a useful starting point if one wishes to change the assumptions so that the analytical expression of Wakefield (2009) no longer applies (as in abf.t).

Value

A vector of approximate Bayes factors.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(agtstats)
agtstats$pval <- with(agtstats, pchisq((beta/se.GC)^2, df = 1, lower.tail = FALSE))
max1 <- function(bf) return(bf/max(bf, na.rm = TRUE))
agtstats$BF.normal <- with(agtstats, max1(abf.Wakefield(beta, se.GC, 0.05)))
agtstats$BF.numeric <- with(agtstats, max1(abf.normal(beta, se.GC, 0.05)))
with(agtstats, plot(BF.normal, BF.numeric)) # excellent agreement
```

---

abf.t *Calculate approximate Bayes factor (ABF) for t distribution prior.*

Description

Calculates an approximation to the Bayes Factor for an alternative model where the parameter beta is a priori t distributed, by approximating the likelihood function with a normal distribution.

Usage

```
abf.t(beta, se, priorscale, df = 1, gridrange = 3, griddensity = 20)
```

Arguments

beta	Vector of effect size estimates.
se	Vector of associated standard errors.
priorscale	Scalar specifying the scale (standard deviation) of the prior on true effect sizes.
df	Degrees of freedom for t distribution prior.
gridrange	Parameter controlling range of grid for numerical integration.
griddensity	Parameter controlling density of points in grid for numerical integration.

Details

This uses the same normal approximation for the likelihood function as "Bayes factors for genome-wide association studies: comparison with P-values" by John Wakeley, 2009, Genetic Epidemiology 33(1):79-86 at http://dx.doi.org/10.1002/gepi.20359. However, in contrast to that work, a t distribution is used for the prior, which means it is necessary to use a numerical algorithm to calculate the (approximate) Bayes factor.

Value

A vector of approximate Bayes factors.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(agtstats)
agtstats$pval <- with(agtstats, pchisq((beta/se.GC)^2, df = 1, lower.tail = FALSE))
max1 <- function(bf) return(bf/max(bf, na.rm = TRUE))
agtstats$BF.normal <- with(agtstats, max1(abf.Wakefield(beta, se.GC, 0.05)))
agtstats$BF.t <- with(agtstats, max1(abf.t(beta, se.GC, 0.0208)))
with(agtstats, plot(-log10(pval), log(BF.normal)))
with(agtstats, plot(-log10(pval), log(BF.t)))
```

---

abf.Wakefield *Calculate approximate Bayes factor (ABF) using method of Wakefield (2009).*

Description

Calculates an approximation to the Bayes Factor for an alternative model where the parameter beta is a priori normal, by approximating the likelihood function with a normal distribution.

Usage

```
abf.Wakefield(beta, se, priorsd)
```

Arguments

beta	Vector of effect size estimates.
se	Vector of associated standard errors.
priorsd	Scalar specifying the standard deviation of the prior on true effect sizes.

Details

See "Bayes factors for genome-wide association studies: comparison with P-values" by John Wakefield, 2009, Genetic Epidemiology 33(1):79-86 at http://dx.doi.org/10.1002/gepi.20359.

Value

A vector of approximate Bayes factors.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(agtstats)
agtstats$pval <- with(agtstats, pchisq((beta/se.GC)^2, df = 1, lower.tail = FALSE))
max1 <- function(bf) return(bf/max(bf, na.rm = TRUE))
agtstats$BF.normal <- with(agtstats, max1(abf.Wakefield(beta, se.GC, 0.05)))
agtstats$BF.t <- with(agtstats, max1(abf.t(beta, se.GC, 0.0208)))
with(agtstats, plot(-log10(pval), log(BF.normal)))
with(agtstats, plot(-log10(pval), log(BF.t)))
```

---

agtstats *Hypertension association statistics for SNPs near the AGT gene.*

Description

Hypertension case/control association test statistics (effect size estimate and standard error after genomic control) for 21 single nucleotide polymorphisms (SNPs) near the angiotensinogen (AGT) gene, from a meta-analysis of 25118 subjects from 10 cohorts, genotyped using the Illumina HumanCVD BeadChip.

Usage

```
data(agtstats)
```

Format

agtstats is a data frame with three informative columns: dbSBNP is the name of the SNP, beta is the meta-analysis effect size estimate, and se.GC is the meta-analysis standard error after genomic control (GC).

Source

Taken directly from Supplemental Table S13 of Johnson et al. (2011), see http://dx.doi.org/10.1016/j.ajhg.2011.10.013.

---

align.snpdata.coding *Update genotype coding when there are coded allele designation flips.*

Description

The input parameterisation specify a desired coded allele for each SNP. This function examines the coded and noncoded alleles used in the input genotype data, and for each SNP where the the input genotype data are encoded as the dose of the opposite (desired noncoded allele) allele, an additional column is added to the output genotype data with the dose of the desired coded allele.

Usage

```
align.snpdata.coding(params, snpdata, ploidy = 2,
                     missing.snp = "fail")
```

Arguments

params	a data frame, see gtx.params.
snpdata	a list with snpinfo and data, see snpdata.
ploidy	if dosage for the noncoded allele is x, the dosage for the coded allele is calculated as ploidy-x.
missing.snp	character, either "fail" or "okay".

Details

The PLINK convention of calling the coded allele "0" for monomorphic SNPs is handled transparently, by assuming that the absent allele in the input genotype data matches whatever allele in the desired parameterisation does not match the present allele in the input genotype data. This behaviour should not cause inadvertent strand flips.

You should not need to call this function, unless you are intending to call grs.onesnp.apply without calling grs.make.scores first. Note that grs.onesnp.apply has no way to check whether columns for desired coded alleles are present and may return NA for codes it cannot find.

The ploidy argument defaults to 2, but should be set to 1 if the input genotype data are haplotypes (either phased or male X or Y chromosome).

The missing.snp argument controls how to handle SNPs in the desired paramterisation that are not present in the input genotype data. If "okay" then SNPs listed in the desired parameterisation but not present in the input genotype data are assumed to have dosage zero for all individuals.

This function is one of the main computational bottlenecks and should be aggresively optimised in future releases.

Value

List with $params and $snpdata slots, contain the input arguments with additional columns. The input params has an extra column data.coded.freq and the input snpdata$data has extra column(s) for doses of the specified coded alleles.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
"rs1537514_G" %in% names(mthfrex$data) # FALSE
mthfrex <- align.snpdata.coding(mthfr.params, mthfrex)$snpdata
"rs1537514_G" %in% names(mthfrex$data) # TRUE
```

---

allelesAB *Paste together (vectors of) A and B alleles after sorting alphabetically.*

Description

A tool to summarise and compare alleles when we do not care what order they are reported in, i.e. when A/G and G/A are considered the same.

Usage

```
allelesAB(A1, A2, sep = "/")
```

Arguments

A1	a vector of allele names.
A2	a vector of allele names, of same length as A1.
sep	string to use as a separator.

Value

A vector of combined allele names; elementwise A1 and A2 are pasted together in alphabetically increasing order.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(t2d.scores)
table(allelesAB(t2d.scores$coded.allele, t2d.scores$noncoded.allele))
```

---

bp.scores *Genetic risk scores for blood pressure.*

Description

Risk scores parameterised using GWAS meta-analysis results for 29 SNPs published by the International Consortium for Blood Pressure GWAS (ICBP-GWAS Nature 2011). There are scores for systolic blood pressure (SBP), diastolic blood pressure (DBP), and mean blood pressure (MBP).

Usage

```
data(bp.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

There are three scores are called SBP2011, DBP2011 and MBP2011. Mean blood pressure is defined $(\mathrm{SBP}+\mathrm{DBP})/2$ and is *not* the same as mean arterial pressure. These are parameterised using the "all data" effect size estimates from Table 1 and Supplementary Table 5 of the ICBP-GWAS manuscript. There are also two scores called SBP2011A and DBP2011A. There are parameterised using the "unbiased" effect size estimates from Supplementary Appendix A of the ICBP-GWAS manuscript. The coefficients are all in mmHg per coded allele dose after medication adjustment (which was +15mmHg SBP and +10mmHg DBP for medicated individuals).

Source

For the publication by ICBP-GWAS from which these data were extracted see http://dx.doi.org/10.1038/nature10405.

Examples

```
data(bp.scores)
head(subset(bp.scores, score == "SBP2011A"))
sbp <- subset(bp.scores, score == "SBP2011A")
dbp <- subset(bp.scores, score == "DBP2011A")
dbp <- dbp[match(sbp$locus, dbp$locus), ]
plot(sbp$coef/sign(sbp$coef), dbp$coef/sign(sbp$coef),
     xlim = c(0, max(sbp$coef/sign(sbp$coef))),
     ylim = c(0, max(dbp$coef/sign(sbp$coef))),
     xlab = "SBP effect [mmHg]", ylab = "DBP effect [mmHg]",
     las = 1)
```

cad.scores *Genetic risk scores for coronary artery disease risk*

Description

Risk score parameterised using GWAS meta-analysis results published by the C4D and CARDIoGRAM consortia (C4D Consortium Nature Genetics 2011, Schunkert et al. Nature Genetics 2011). The score is for risk of prevalent coronary artery disease (CAD).

Usage

```
data(cad.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

The score is called CAD2011, and has coefficients in ln(odds) for CAD per coded allele. For details of how results from the two papers were combined, see manuscript by T. Johnson.

Source

For the publications from which these data were extracted see http://dx.doi.org/10.1038/ng.782 and http://dx.doi.org/10.1038/ng.784.

Examples

```
data(cad.scores)
cad.scores$MAF <- pmin(cad.scores$coded.freq, 1-cad.scores$coded.freq)
cad.scores$OR <- exp(abs(cad.scores$coef))
plot(cad.scores$MAF, cad.scores$OR,
     xlim = c(0, 0.5), ylim = c(1, max(cad.scores$OR)),
     xlab = "Minor allele frequency",
     ylab = "Odds ratio effect", las = 1)
text(cad.scores$MAF, cad.scores$OR, cad.scores$name, pos = 1, cex = 0.5)
```

chi2ncp *Compute non-centrality parameter of chi squared distribution.*

Description

Computes the non-centrality parameter of chi squared distribution for specified alpha and beta, such that there is probability (power) beta of exceeding the critical value for probability (size) alpha for a central chi squared distribution

Usage

```
chi2ncp(alpha, beta, df = 1)
```

Arguments

alpha	Required tail area for central chi squared distribution.
beta	Required tail area for non-central chi squared distribution.
df	Degrees of freedom for both chi squared distributions.

Details

See the examples.

Value

The non-centrality parameter.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
## 0.80 power for 0.05 size test
chi2ncp(.05, .8)
## 0.80 power for genome-wide significance
chi2ncp(5e-08, .8)
## test
critval <- qchisq(5e-08, lower.tail = FALSE, df = 1)
pchisq(critval, ncp = chi2ncp(5e-08, .8), lower.tail = FALSE, df = 1)
```

---

coeff.extract *Coefficient extractor for fitted models.*

Description

Tool to extract coefficients and standard errors from different fitted model objects, and return them in a standard format.

Usage

```
coeff.extract(object)
```

Arguments

object	a fitted model object of class lm, glm, or coxph.

Details

coeff.extract detects the class of object and handles it appropriately to extract a two column data frame with columns "Estimate" and "Std Error".

This function is useful because there does not seem to otherwise be a uniform interface to all these classes.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

`combine.moments2` *Builds a matrix of second moments for a set of individuals, from the matrices of second moments for all constituent subsets of individuals.*

Description

Given matrices of non-central second moments between a set of variables, calculated for two or more subsets of individuals, the matrix of non-central second moments between that set of variables, in the combined set of individuals, is obtained simply by addition. This function provides a convenience interface that deals with different row and column orderings in the input matrices, and also allows a subset of variables to be treated as pertaining to specific subsets of individuals only.

Usage

```
combine.moments2(xtxlist, fixed)
```

Arguments

`xtxlist`	a named list of matrices of second moments.
`fixed`	names of variables for which regression coefficients are considered fixed across all studies.

Details

The names of the elements of `xtxlist` are used as identifiers for the subsets of individuals for which each matrix has been calculated.

Variables whose names are included in the `fixed` argument must be included in *all* matrices in `xtxlist`, and are assumed to have identical definitions in all individuals. The intended application is that a regression model will be fitted with a coefficient for each of these variables that takes a single fixed value for all individuals.

Any variables included in matrices in `xtxlist` whose names are not included in the `fixed` argument are assumed to have potentially different definitions in the subsets of individuals corresponding to the matrices in `xtxlist`. An example would be ancestry principal component covariates, that might have the same names ("PC1", "PC2", ...) but do not have the same definitions in different subsets of individuals. The intended application is that a regression model will be fitted with a subset-specific coefficient for each of these variables, and the variables are therefore renamed by prepending identifiers constucted using the names of the elements of `xtxlist`.

The combined matrix has no "ONE" intercept term, and instead has `length(xtxlist)` subset-specific intercepts (named by prepending identifiers constucted using the names of the elements of `xtxlist`). This allows the the full rank of the combined matrix to be maintained while treating all subsets symetrically.

Value

A numeric matrix of second moments.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
## artifical example with two datasets obtained by
## splitting mthfrex by the HTN variable

xtx.hi <- make.moments2(mthfr.params, c("SBP", "SexC", "Age"),
                        as.snpdata(list(snpinfo = mthfrex$snpinfo,
                                        data = subset(mthfrex$data, HTN == 1))))

xtx.lo <- make.moments2(mthfr.params, c("SBP", "SexC", "Age"),
                        as.snpdata(list(snpinfo = mthfrex$snpinfo,
                                        data = subset(mthfrex$data, HTN == 0))))

## make list of X'X matrices
xtx.list <- list(hi = xtx.hi, lo = xtx.lo)

## combine for outcome SBP and fixed effects for all SNPs
## other variables SexC and Age will be treated as study-specific
fixed <- paste(mthfr.params$snp, mthfr.params$coded.allele, sep = "_")
xtx.comb <- combine.moments2(xtx.list, c("SBP", fixed))

## fit regression model
n.comb <- sum(diag(xtx.comb)[1:length(xtx.list)])
lm.moments2(xtx.comb, "SBP", c("hi_ONE", "lo_ONE", "rs4846052_T"), n = n.comb)

## equivalent regression directly using subject-specific data
coeff.extract(lm(SBP ~ HTN + rs4846052_C, data = mthfrex$data))
```

---

contrasting.rainbow *Rainbow of colours permuted to maximise contrast.*

Description

Returns a rainbow of colours, permuted so that adjacent elements constrast each other as much as possible.

Usage

```
contrasting.rainbow(x, ...)
```

Arguments

x	number of colours in the rainbow.
...	other arguments to be passed to rainbow.

Details

The permutation is chosen by finding the largest coprime of x that is not greater than x/2, and using that coprime as an increment size to cycle through the normal rainbow.

Value

A vector of colours of length x.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
## contrasting colours suitable for Manhattan plot for 22 autosomes
plot(1:22, rep(1, 22), pch = 22, cex = 2, ann = FALSE,
     yaxt = "n", bg = contrasting.rainbow(22))
```

---

est.moments2 *Estimate regression coefficients using quadratic approximation to likelihood function.*

Description

To make a quadratic approximation to the likelihood function, the score and information are obtained from a pre-built matrix of weighted second moments. This allows a parameter estimate to be obtained by one iteration of weighted least squares, or equivalently a score test. Typically the weights used to construct the pre-built matrix correspond to the MLE under a chosen null model.

Usage

```
est.moments2(xtwx, leftvar, rightvars, n = NULL, vscale = NULL)
```

Arguments

xtwx	an object of class moments2, typically built using make.moments2 with the weightvar argument set, or a matrix of weighted second moments.
leftvar	name of the response variable (the left hand side of the formula).
rightvars	name(s) of the explanatory variables (the right hand side of the formula).
n	sample size, only needed for the normal linear model if there is not a single intercept "ONE" for all individuals.
vscale	parameter needed if xtwx is not of class moments2, set to NULL for normal linear model and 1 for logistic regression.

Details

Variables in `rightvars` with non-identifiable coefficients are removed, with preference for keeping variables that occur earlier rather than later in `rightvars`.

When the `vscale` attribute of `xtwx` (or the `vscale` function argument) is `NULL`, this function assumes that the `xtwx` argument was calculated with unit weights and therefore that a linear model fit is required with error variance estimated from the data. For this application it is preferred to call `lm.moments2`, which is a wrapper for this function with `vscale=NULL`.

When the `vscale` attribute of `xtwx` (or the `vscale` function argument) is set equal to 1, this function assumes that the `xtwx` argument was calculated with weights calculated such that a GLS problem has been correctly set up to approximate a likelihood function, and therefore that generalised linear model fit is required.

Values other than `NULL` or 1 for the `vscale` parameter may not be what you think. Do not use other values unless you are absolutely sure what you understand what are doing. See the source code for details.

Value

A list with slots for the effect size estimates, standard errors, and a precision matrix.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
mthfrex <- gls.approx.logistic(mthfrex, "HTN", c("SexC", "Age"))
xtwx <- make.moments2(mthfr.params, c("HTNstar", "SexC", "Age"), mthfrex,
                      weightvar = "weight")
myglm <- est.moments2(xtwx, "HTNstar", c("ONE", "rs6668659_T", "rs4846049_T",
                                "rs1801133_G", "SexC", "Age"), vscale=1)
myglm$z <- myglm$betahat/myglm$se
cbind(beta = myglm$betahat, se = myglm$se, z = myglm$z,
      pval = pnorm(-abs(myglm$z))*2)

## Compare against results from glm
## Note have to use coded alleles used in original data
mycheck <- glm(HTN ~ rs6668659_G+rs4846049_G+rs1801133_A+Sex+Age,
               family="binomial", data = mthfrex$data)
coef(summary(mycheck))
## Note in results Sex factor coded differently than SexC
## Coefficients for covariates used in null model are different,
##     because xtwx approximates around the fitted null model

## Look at pairwise correlations
cor(subset(mthfrex$data, select = c("rs6668659_G", "rs4846049_G",
                                    "rs1801133_A")))^2

## SNP coefficients well approximated (given very high
## inter-SNP correlations) but signs ALL inverted by coded allele flips
```

```
## check error less than 10percent
stopifnot(all(-1*myglm$z[2:4]/coef(summary(mycheck))[2:4,3] > 0.9))
stopifnot(all(-1*myglm$z[2:4]/coef(summary(mycheck))[2:4,3] < 1.1))
```

`fitmix` *Fit finite mixture of univariate Gaussian densities to data.*

Description

Implementation of EM algorithm to fit k component univariate Gaussian mixture to data, for user specified value of k. In constrast to general purpose (unconstrained) Gaussian mixture models, this function allows certain restrictions or parameter space reductions that make the model correspond more closely to a classical quantitative genetics model.

Usage

```
fitmix(x, k, tol = 1e-6, maxit = 100, restarts = 20,
       p.binomial = FALSE, mu.additive = FALSE, sigma.common = FALSE)
```

Arguments

`x`	Real vector of data to be fitted by the model.
`k`	Number of components in the mixture.
`tol`	Threshold for log-likelihood increase for convergence.
`maxit`	Maximum number of iterations for each run of the EM algorithm.
`restarts`	Number of times to restart EM algorithm at random initial points.
`p.binomial`	Assume mixture proportions correspond to binomial expansion (corresponds to Hardy-Weinberg for k=3).
`mu.additive`	Assume means follow additive model.
`sigma.common`	Assume variances are same for all components.

Details

Most likely applications are `k=2` with `sigma.common=TRUE` for a completely dominant or recessive genetic model, and `k=3` with `p.binomial=TRUE`, `mu.additive=TRUE` and `sigma.common=TRUE` for a codominant biallelic genetic model.

Note that the unconstrained model has many global optima that allow unbounded increase in the log likelihood as one (or more) mixture components have sigma tend to zero as mu tends to one of the data observed values. The EM algorithm rarely converges to these optima since they typically have very small domains of attraction.

Value

A list. The elements p, mu, and sigma, are each vectors of length k, and describe the mixture found that maximised the likelihood. An element loglik is a vector of length `restarts+1` and gives the log likelihood at the end of each run of the EM algorithm. Inspecting `loglik` may give some indication of whether a "good" local optimum has been found.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
xx <- fitmix.simulate(100, c(0.49, 0.42, 0.09), c(0, 1, 2), c(.3, .3, .3))

## additive model, common variance, Hardy--Weinberg
fit.a <- fitmix(xx, 3, maxit = 10, restarts = 3,
                sigma.common = TRUE, p.binomial = TRUE, mu.additive = TRUE)
fitmix.plot(xx, fit.a$p, fit.a$mu, fit.a$sigma)

## general (unrestricted) fit
fit.g <- fitmix(xx, 3, maxit = 10, restarts = 3)
fitmix.plot(xx, fit.g$p, fit.g$mu, fit.g$sigma)
```

---

fitmix.plot *Plot empirical density and components and total density for finite mixture of univariate Gaussian densities.*

Description

Draws a nice plot.

Usage

```
fitmix.plot(x, p, mu, sigma)
```

Arguments

x	Real vector of data.
p	Real vector of mixture proportions.
mu	Real vector of mixture component means.
sigma	Real vector of mixture component standard deviations.

Value

Returns an invisible null.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
xx <- fitmix.simulate(100, c(0.49, 0.42, 0.09), c(0, 1, 2), c(.3, .3, .3))

## additive model, common variance, Hardy--Weinberg
fit.a <- fitmix(xx, 3, maxit = 10, restarts = 3,
                sigma.common = TRUE, p.binomial = TRUE, mu.additive = TRUE)
fitmix.plot(xx, fit.a$p, fit.a$mu, fit.a$sigma)

## general (unrestricted) fit
fit.g <- fitmix(xx, 3, maxit = 10, restarts = 3)
fitmix.plot(xx, fit.g$p, fit.g$mu, fit.g$sigma)
```

---

fitmix.r2 *For finite mixture of univariate Gaussian densities, computes proportion of variance explained by the mixture labels.*

Description

Computes the true R-squared if the mixture labels were known without error. Currently only works if all components have the same variance.

Usage

```
fitmix.r2(p, mu, sigma)
```

Arguments

p	Real vector of mixture proportions.
mu	Real vector of mixture component means.
sigma	Mixture component standard deviation, same for all components.

Value

The R-squared value for a (hypothetical) regression of data values onto mixture label, as a k level factor.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
xx <- fitmix.simulate(100, c(0.49, 0.42, 0.09), c(0, 1, 2), c(.3, .3, .3))

## additive model, common variance, Hardy--Weinberg
fit.a <- fitmix(xx, 3, maxit = 10, restarts = 3,
                sigma.common = TRUE, p.binomial = TRUE, mu.additive = TRUE)
fitmix.plot(xx, fit.a$p, fit.a$mu, fit.a$sigma)
fitmix.r2(fit.a$p, fit.a$mu, unique(fit.a$sigma))
```

---

fitmix.simulate *Simulate from finite mixture of univariate Gaussian densities.*

Description

Simulates random variables from a user specified finite mixture of univariate Gaussian densities.

Usage

```
fitmix.simulate(n, p, mu, sigma)
```

Arguments

n	Number of draws from target distribution.
p	Real vector of mixture proportions.
mu	Real vector of mixture component means.
sigma	Real vector of mixture component standard deviations.

Value

A vector of simulated values.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
xx <- fitmix.simulate(100, c(0.49, 0.42, 0.09), c(0, 1, 2), c(.3, .3, .3))
plot(density(xx))
```

`gls.approx.logistic` *Calculate weights and transformed phenotype so that one iteration of generalised least squares approximates a logistic regression.*

Description

Logistic regression models are usually fitted by iteratively reweighted generalised least squares (GLS). This function formulates a GLS problem by calculating weights and rescaling the response variable such that a logistic regression analysis is approximated. This is equivalent to making a quadratic approximation to the likelihood and to performing a score test.

Usage

```
gls.approx.logistic(snpdata, leftvar, rightvars = NULL,
                    outvar = paste(leftvar, "star", sep = ""),
                    weightvar = "weight")
```

Arguments

`snpdata`	a list with snpinfo and data, see snpdata.
`leftvar`	the name (in `snpdata$data`) of the response variable for the logistic regression. Must be 0/1 or NA for all individuals.
`rightvars`	the names (in `snpdata$data`) of the explanatory variables for the logistic regression.
`outvar`	the name (to be added to `snpdata$data`) of the transformed phenotype that is the response variable in the GLS problem.
`weightvar`	the name (to be added to `snpdata$data`) of the weight variable in the GLS problem.

Details

An intercept term is *always* included and there is currently no way to override this.

The specified model is printed as a check. MLE parameter values are estimated under the specified model by calling `glm`. These are used to compute weights and a transformed phenotype such that one iteration of generalised least squares constitutes a score test for additional terms that might subsequently be added to the model.

This function works on the (usually phenotypic) columns in the `$data` slot of `snpdata`.

Value

Returns `snpdata` with additional columns for the weights and transformed phenotype.

The formula used to find an MLE is printed.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
mthfrex <- gls.approx.logistic(mthfrex, "HTN", c("SexC", "Age"))
xtwx <- make.moments2(mthfr.params, c("HTNstar", "SexC", "Age"), mthfrex,
                      weightvar = "weight")
est.moments2(xtwx, "HTNstar", c("ONE", "rs6668659_T", "rs4846049_T",
                                "rs1801133_G", "SexC", "Age"), vscale=1)
```

`grs.filter.Qrs` *Filter SNPs for inclusion in genetic risk score using heterogeneity test.*

Description

Performs a stepwise downward "model selection" in which SNPs are iteratively removed from the risk score until the heterogeneity test is no longer significant at the specified threshold.

Usage

```
grs.filter.Qrs(w, b, s, p.thresh = 0.05)
```

Arguments

`w`	coefficients for the risk score
`b`	aligned beta coefficients in the testing dataset
`s`	standard errors
`p.thresh`	P-value threshold

Details

When there are m SNPs in the risk score, the heterogeneity test is an (m-1) d.f. LRT comparing the 1 d.f. risk score model against the unconstrained m d.f. model, as reported by the Qrs element of `grs.summary`. At each iteration, if the risk score model has a significant heterogeneity test (at `p.thresh`), the SNP is removed that gives the greatest decrease in the heterogeneity test statistic.

No guarantee is given about the performance of this procedure. Intuitively, it is expected to work when the majority of SNPs in the risk score only affect the outcome with effects proportional to their weights in the score (including the possibility of zero effects), and a minority of SNPs affect the outcome with non-proportional effects.

When using a risk score to make causal inference, the application of this procedure (and indeed of any use of the heterogeneity test) should not be viewed as a replacement for detailed biological knowledge about the mechanisms of action of the causal genetic variants tagged by the SNPs used in the risk score.

Compared with iteratively calculating the heterogeneity test using `grs.summary` and removing SNPs based on inspection of `grs.plot`, the `grs.filter.Qrs` procedure offers only convenience and (a modicum of) objectivity, and nothing more.

Value

A logical vector of the same length as w, set TRUE or FALSE respectively for SNPs included or excluded at the end of the model selection procedure.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(magic.scores)
score1 <- subset(magic.scores, score == "FG2010")
score1 <- within(score1, okay <- grs.filter.Qrs(coef, beta_TG, se_TG))
with(score1, {grs.plot(coef, beta_TG, se_TG, locus);
  title(xlab = "FG effect", ylab = "TG effect")})
with(score1, locus[!okay]) # loci removed
with(subset(score1, okay), {grs.plot(coef, beta_TG, se_TG, locus);
  title(xlab = "FG effect", ylab = "TG effect")})
```

---

grs.make.scores *Make genetic risk scores from individual-level data.*

Description

Constructs the value of one or more genetic risk scores with specified parameterisation, from individual-level genotype data. Automatically handles coded allele designation.

Usage

```
grs.make.scores(params, snpdata, appendage = ".score")
```

Arguments

params	a data frame, see gtx.params.
snpdata	a list with snpinfo and data, see snpdata.
appendage	text to add to score name to make new column name in snpdata$data.

Details

This function computes subject-specific values of one or more additive multi-SNP genetic risk scores using a supplied parameterisation and subject-specific genotype and phenotype data. These risk scores are returned as additional columns on the input genotype data frame, making it straightforward to use them as explanatory variables for any of the model fitting functions available in the R language, such as lm, glm, or coxph.

Value

The snpdata list with additional column(s) added to data.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

`grs.onesnp.apply` *Convenience tool to fit a series of single-SNP models.*

Description

This function takes as input a fitted model object, typically created by the standard R functions lm, glm or coxph, and fitted to individual-level genotype and phenotype data. The model is augmented by adding a term for each single SNP in turn and the refitted. This provides the single SNP summary association statistics needed to apply the summry statistic methods.

Usage

```
grs.onesnp.apply(params, object, coeff.extract.fun = coeff.extract)
```

Arguments

`params`	a data frame, see `gtx.params`.
`object`	a fitted model object of class lm, glm, or coxph
`coeff.extract.fun`	function that extracts Estimate and Std.Err from fitted model objects, see `coeff.extract`.

Details

By default this uses `coeff.extract` to detect the class of object and handles coefficient extraction appropriately. Supply your own function if you have an object that works with update (supply your own update too).

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
library(survival)
data(t2d.scores)
data(t2dex)

mycoxph <- coxph(Surv(FollowupDays,FollowupT2D) ~ Overweight,
                 data = t2dex$data) # fit null model
assoc1 <- grs.onesnp.apply(t2d.scores, mycoxph) # single SNP association
## risk score fit from single SNPs
unlist(grs.summary(t2d.scores$coef, assoc1$beta, assoc1$se,
                   n = length(residuals(mycoxph))))

## compare direct analysis of subject-specific data
```

```
t2dex <- grs.make.scores(t2d.scores, t2dex)
coxph(Surv(FollowupDays,FollowupT2D) ~ Overweight + T2D2010.score,
      data = t2dex$data)
```

`grs.plot` *Diagnostic plot for genetic risk score calculation from summary statistics.*

Description

Each SNP is plotted by coefficient in the risk score (x axis) versus estimated effect size for trait of interest in the testing dataset (y axis). A solid red line shows the effect size estimate for the risk score on the trait of interest in the testing dataset.

Usage

```
grs.plot(w, b, s, text = NULL, textpos = NULL, textcex = 0.5,
         alpha = 0.05)
```

Arguments

`w`	coefficients for the risk score.
`b`	aligned beta coefficients in the testing dataset.
`s`	standard errors for `b`.
`text`	optional labels for SNPs
`textpos`	position for labels for SNPs, as `pos` argument to `text()`.
`textcex`	size for lables for SNPs, as `cex` argument to `text()`.
`alpha`	determines size of confidence limits plotted.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(t2dex)
library(survival)
mycoxph <- coxph(Surv(FollowupDays,FollowupT2D) ~ Overweight,
                 data = t2dex$data) # fit null model
data(t2d.scores)
assoc1 <- grs.onesnp.apply(t2d.scores, mycoxph) # single SNP association
## risk score fit from single SNPs
grs.plot(t2d.scores$coef, assoc1$beta, assoc1$se, t2d.scores$name)
title(xlab = "risk score weight", ylab = "estimated effect size")
```

`grs.summary` *Genetic risk score calculation from summary statistics.*

Description

Implements the summary statistic method of Johnson et al. for approximating the regression of a response variable onto an additive multi-SNP genetic risk score in a given testing dataset, using only single SNP association summary statistics.

Usage

```
grs.summary(w, b, s, n)
```

Arguments

`w`	coefficients for the risk score.
`b`	aligned beta coefficients in the testing dataset, of same length as `w`.
`s`	standard errors for `b`, of same length as `w` and `b`.
`n`	sample size of testing dataset.

Details

The risk score coefficients `w` are the "weights" used to construct the risk score, for a set of SNPs, in chosen units per dose of the coded allele. Typically these are single SNP regression coefficients estimated in a "discovery" dataset.

The aligned beta coefficients `b` are regression coefficients for the response variable of interest, for the same set of SNPs and per dose of the same coded allele as used to define `w`. Typically these are single SNP regression coefficients estimated in the "testing" dataset. The standard errors `s` are standard errors on `b`.

In applications to causal inference, a common objective is to estimate the causal effect of an intermediate trait or biomarker, on a response variable or outcome. In such applications, the `w` are the estimated effects on the intermediate trait or biomarker, and the `b` are estimated effects on the response variable or outcome, with standard errors `s`.

The sample size argument `n` is required only to compute the (pseudo) variance explained in the testing dataset from the likelihood ratio test statistic.

The method for calculating the regression of the response variable onto the risk score was first used for the work of the International Consortium for Blood Pressure Genome-Wide Association Studies (2011), and described in more detail in Dastani et al. (2012). The method is exact for uncorrelated SNPs and a quadratic log-likelihood, the latter being obtained under a normal linear model, or under any regression model with a large sample size.

The heterogeneity test is a test of whether the regression coefficients for the response variable are proportional to the risk score coefficients. It is described in detail in the "ashg2012" package vignette. In applications to causal inference, firstly note that the heterogeneity test often lacks power, and hence a non-significant heterogeneity test is not evidence of clean instruments. Secondly note that poor fit may be detected either when there are pleiotropic effects, or alternatively when

one or more of the coefficients used to parameterise the risk score have been estimated imprecisely or with bias, and therefore a significant heterogeneity test is not necessarily evidence of unclean instruments. Nonetheless, a significant heterogeneity test may indicate that underlying assumptions should be subjected to extra scrutiny before any inference is made about causality.

Value

A named list with the following elements: `m` is the number of SNPs used in the risk score. `n` is the input sample size. `X2m` is the chi squared test statistic for an m d.f. test in the testing dataset (all SNPs have independent effects). `R2m` is the (pseudo) variance explained by the m d.f. model in the testing dataset. `ahat` is the estimated coefficent for regressing the response onto the m SNP risk score. `aSE` is the standard error. `X2rs` is the chi squared test statistic for a 1 d.f. test for the risk score in the testing dataset. `R2rs` is the (pseudo) variance explained by the risk score model in the testing dataset. `pval` is the P-value for the 1 d.f. test. `Qrs` is the (m-1) d.f. heterogeneity test statistic. `phet` is the heterogeneity test P-value.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

References

International Consortium for Blood Pressure Genome-Wide Association Studies (2011 Nature) http://dx.doi.org/10.1038/nature10405

Dastani et al. (2012 PLoS Genetics) http://dx.doi.org/10.1371/journal.pgen.1002607

Johnson (2012 ASHG poster) see "ashg2012" package vignette.

---

gtx.params *Parameter format for multi-SNP analyses*

Description

Format used by functions in the gtx package for parameterisations for multi-SNP analyses. The parameterisation describes the SNPs to be used, choice of coded and noncoded alleles, and potentially allele frequency and effect size information.

Details

`params` must be a data frame with (at least) columns called "`snp`", "`coded.allele`" and "`noncoded.allele`". For some functions, additional columns called "`coded.freq`" and "`coef`" are also required.

Suitable parameterisations in the required format are provided for some genetic risk score analyses by `cad.scores`, `t2d.scores` etc.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
## Not run:
align.snpdata.coding(params, snpdata)
grs.make.scores(params, snpdata)
grs.onesnp.apply(params, object)
make.moments2(params, phenolist, snpdata)

## End(Not run)
```

`hapmap.read.haplotypes` *Read hapmap haplotypes.*

Description

Reads HapMap haplotypes and legend and stores in format expected by functions in gtx package.

Usage

```
hapmap.read.haplotypes(chr, path, sample = "CEU", release = 22)
```

Arguments

`chr`	chromosome to read
`path`	path to local mirror of ftp://ftp.ncbi.nlm.nih.gov/hapmap
`sample`	a HapMap sample, e.g. CEU or YRI
`release`	a HapMap release version, currently must be 21 or 22

Details

You need a local mirror of HapMap data. To build this use e.g. a shell command
`rsync ftp://ftp.ncbi.nlm.nih.gov/hapmap ~`
then use an argument `path="~/hapmap"` for this function call.

Technically the local mirror only needs the contents of
ftp://ftp.ncbi.nlm.nih.gov/hapmap/phasing/2006-07_phaseII/phased/
and
ftp://ftp.ncbi.nlm.nih.gov/hapmap/phasing/2007-08_rel22/phased/

Other gtx functions expect to be passed a list with 22 elements; one for each chromosome. See the examples.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
## Not run:
## chromosomes 1-22
hapmap <- lapply(1:22, hapmap.read.haplotypes, path = "~/hapmap")

## selected chromosomes only
hapmap <- lapply(1:22, function() return(NULL))
for (chr in c(1, 11, 17)) {
  hapmap[[chr]] <- hapmap.read.haplotypes(chr, path = "~/hapmap")
}

## End(Not run)
```

`hapmap.snpdata` *Extract individual level snp/haplotype data from HapMap*

Description

Does this.

Usage

```
hapmap.snpdata(params, hapmap)
```

Arguments

`params`	a parameterised risk score
`hapmap`	a hapmap list object

Details

See hapmap for how to build. Returns a snpdata object with special ploidy=1.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

height.scores — *Genetic risk score for height.*

Description

Risk score for height parameterised using GWAS meta-analysis results published by Langon Allen et al. (Nature 2010) as part of work by the GIANT consortium.

Usage

```
data(height.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

The score is called HEIGHT2010, and has coefficients in height Z score units (i.e. phenotypic standard deviations) per coded allele. One phenotypic standard deviation for height is 0.06-0.07m in most samples analysed by Langon Allen et al.

Effect size estimates are taken from the "STAGE 2" results in Supplementary Table 1 of Langon Allen et al.

Source

For the publication by Langon Allen et al. from which these data were extracted see http://dx.doi.org/10.1038/nature09410.

lipid.cad.scores — *Genetic risk scores for serum lipid levels and coronary artery disease outcome*

Description

Risk scores parameterised using GWAS meta-analysis results published by Waterworth et al. (Arteriosclerosis, Thrombosis, and Vascular Biology 2010). There are scores for three different lipid level phenotypes (High Density Lipoprotein HDL; Low Density Lipoprotein LDL; Triglycerides TG), all parameterised using SNPs reported in previous studies, with effect sizes estimated using the data of Waterworth et al. (2010 Table 1). For each SNP the outcome, estimated effect on risk of coronary artery disease (CAD), from a case/control collection of 9633 cases and 38684 controls, was reported by Waterworth et al. (2010 Table 4).

Usage

```
lipid.cad.scores
```

Format

A data frame suitable for use with other functions in this package.

These scores are called HDL, LDL and TG, and have coefficients in change in natural log transformed lipid level per coded allele.

The coefficients for outcome effect are in natural log odds ratio per coded allele.

Source

For the publication by Waterworth et al. from which these data were extracted see http://dx.doi.org/10.1161/ATVBAHA.109.201020.

---

lipid.scores *Genetic risk scores for serum lipid levels*

Description

Risk scores parameterised using GWAS meta-analysis results published by the Global Lipid Genetics Consortium (Teslovich et al. Nature 2010). There are scores for four different lipid level phenotypes (High Density Lipoprotein HDL; Low Density Lipoprotein LDL; Total Cholesterol TC; Triglycerides TG), all parameterised using genome wide significant SNPs reported in Supplementary Table 1 of Teslovich et al. (2010).

Usage

```
lipid.scores
```

Format

A data frame suitable for use with other functions in this package.

These scores are called HDL2010, LDL2010, TC2010 and TG2010, and have coefficients in mg/dl per coded allele.

Additional scores based on larger numbers of SNPs are available separately... LDL.prune.0.01 and there is an additional column in the data frame, p-value, that should be used to threshold on. The coefficients are back-calculates from Z scores and HapMap allele frequency data.

Source

For the publication by Teslovich et al. from which these data were extracted see http://dx.doi.org/10.1038/nature09270.

`liver.scores` *Genetic risk scores for liver enzyme traits.*

Description

Risk scores parameterised using GWAS meta-analysis results published by Chambers et al. (Nature Genetics 2011) for serum ALT, ALP and GGT levels.

Usage

```
data(liver.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

The scores are called ALT2011, ALP2011 and GGT2011, for serum concentrations of alanine transaminase (ALT), alkaline phosphatase (ALP) and gamma-glutamyl transferase (GGT). The first twelve columns of data were taken from Table 1 of Chambers et al. (2011). Effect sizes (`coef`) are on log10(concentration) scale, from inverse variance weighted meta-analyses over 17 studies, with analysed sample sizes 45596, 56415 and 61089, for ALT, ALP and GGT respectively. Two additional columns of data (`beta_replication` and `se_replication`) were taken from Supplementary Table 18 of Chambers et al. (2011), and give the effect size and SE (back calculated from the reported P-value) in an independent replication sample of size 8112.

Source

For the publication by Chambers et al. from which these data were extracted see http://dx.doi.org/10.1038/ng.970.

Examples

```
data(liver.scores)
with(subset(liver.scores, score == "ALP2011"), {
  plot(pmin(coded.freq, 1 - coded.freq), coef,
       xlab = "MAF", ylab = "ALP effect",
       xlim = c(0, 0.5), ylim = c(0, 10), las = 1)
  text(pmin(coded.freq, 1 - coded.freq), coef, name,
       pos = 3, cex = 0.5)})
## compare unweighted score used by Chambers et al. 2011
## with weighted score:
with(subset(liver.scores, score == "ALP2011"),
     rbind(unlist(grs.summary(rep(1, length(coef)), beta_replication,
                              se_replication, n = 8112)),
           unlist(grs.summary(coef, beta_replication,
                              se_replication, n = 8112))))
```

`lm.moments2` *Fit normal linear model using pre-built matrix of second moments.*

Description

Regression coefficients for a normal linear model are be calculated, using a pre-built sufficient summary statistic matrix that contains the second moments between all the variables of (potential) interest.

Usage

```
lm.moments2(xtx, leftvar, rightvars, n = NULL)
```

Arguments

`xtx`	a matrix of second moments, typically built using `make.moments2`.
`leftvar`	name of the response variable (the left hand side of the formula).
`rightvars`	name(s) of the explanatory variables (the right hand side of the formula).
`n`	sample size, only needed if there is not a single intercept for all individuals.

Details

Non-identifiable variables in `rightvars` are removed, with preference for keeping variables that occur earlier rather than later in `rightvars`.

Value

A list with slots for the effect size estimates, standard errors, and a precision matrix.

This function is just a wrapper for calling `est.moments2` with `scale = NULL`.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
xtx <- make.moments2(mthfr.params, c("SBP", "DBP", "SexC", "Age"), mthfrex)
lm.moments2(xtx, "SBP", c("ONE", "rs6668659_T", "rs4846049_T",
                                "rs1801133_G", "SexC", "Age"))
## Compare against results from lm
## Note have to use coded alleles in original data
lm(SBP ~ rs6668659_G+rs4846049_G+rs1801133_A+Sex+Age, data = mthfrex$data)
## Note in results Sex factor coded differently than SexC
```

`magic.scores` *Genetic risk scores for glucose/insulin traits.*

Description

Risk scores parameterised using GWAS meta-analysis results published by the MAGIC consortium (Dupuis et al. Nature Genetics 2010) for glucose and insulin traits.

Usage

```
data(magic.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

The scores are called FG2010 and FI2010, for fasting plasma glucose (FG, mmol/l) and fasting insulin (FI, pmol/l), and using the 16 and 2 genome wide significant loci from Table 1 of Dupuis et al. (2010) respectively. Effect size coefficients are taken from Table 2 of Dupuis et al. (2010) and hence use SNPs different from those reported in Table 1 at the SLC30A8 and TCF7L2 loci.

In addition, for all SNPs in both risk scores, there are effect size estimates (beta) and standard errors (se) for FG, HOMA-B index of beta cell function, FI, HOMA-IR index of insulin resistance, glycated hemoglobin (HbA1c, percent), glucose 2 hours after oral glucose tolerance test (2hG, mmol/l), insulin 2 hours after oral glucose tolerance test (2hI, pmol/l), type 2 diabetes risk (T2D, ln odds, converted from odds ratio), body mass index (BMI, kg/m2), diastolic blood pressure (DBP, mmHg), systolic blood pressure (SBP, mmHg), hypertension risk (HTN, ln odds), and serum lipid concentrations (HDL, LDL, TC, TG), all taken from Table 2 and Table 3 of Dupuis et al. (2010).

Source

For the publication by Dupuis et al. from which these data were extracted see http://dx.doi.org/10.1038/ng.520.

Examples

```
data(magic.scores)
with(subset(magic.scores, score = "FG2010"),
  grs.plot(beta_FG, beta_TG, se_TG, text = locus))
```

`make.moments2` *Build matrix of second moments from subject-specific data.*

Description

Performs input checking, flipping of coded and noncoded alleles, removal of individuals with missing data, constructs a subject-specific data matrix X, and returns X'X.

Usage

```
make.moments2(params, phenolist, snpdata, weightvar)
```

Arguments

`params`	a data frame, see gtx.params.
`phenolist`	a list of phenotypic response variables and covariates to be included in addition to genetic variables from `params`.
`snpdata`	a list with snpinfo and data, see snpdata.
`weightvar`	name of a variable in `snpdata$data` containing individual weights when approximating the likelihood for GLMs.

Details

After extracting the necessary columns, any rows (individuals) with *any* missing data are removed.

Value

A numeric matrix of second moments.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
xtx <- make.moments2(mthfr.params, c("SBP", "DBP", "SexC", "Age"), mthfrex)
```

mincover *Compute minimum size of cover of overlapping intervals.*

Description

Compute minimum size of cover of overlapping intervals.

Usage

```
mincover(x.begin, x.end)
```

Arguments

x.begin	a vector of interval begin positions.
x.end	a vector of interval end positions.

Value

An integer, the total size covered by the union of all intervals.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
mincover(c(1, 5, 10, 11, 22), c(8, 17, 13, 19, 25))
## first to fourth intervals all overlap
## third interval 10:13 entirely inside second interval 5:17
```

moments2 *Class for summary statistic matrix of second moments.*

Description

Objects of this class are a matrix of second moments between a set of variables to be used in regression analyses.

Usage

```
is.moments2(object)
```

Arguments

object	an object to test.

Details

An object of class `moments2` is square matrix, with an attribute "`vscale`" set to NULL or 1 depending on whether the moments are unweighted (for normal linear model analyses) or weighted (for GLS approximation of generalised linear model analysis).

Typically created by calling `make.moments2()` and used in calls to `lm.moments2()` and `est.moments2()`.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

`mthfrex` *Simulated example finemapping genotype and phenotype data.*

Description

A simulated example finemapping genotype and phenotype dataset, consisting of genotypes for 2000 individuals at 64 SNPs, and case/control and continuous phenotypes and covariates.

Usage

```
data(mthfrex)
```

Format

`mthfrex` is a list suitable for use with other functions in this package, see snpdata.

`mthfr.params` is a data frame suitable for use with other functions in this package, see gtx.params.

Details

This is a simulated dataset, whose sole purpose is to illustrate the use of functions in this package for multi-SNP regression analyses.

The dataset is provided so that the usage examples can actually be run, without burdening each example with many lines of code to generate an analysable dataset.

Source

The genotypes were simulated using Hudson's mksamples (ms) program, assuming an infinite sites standard neutral model with no recombination, with 1% genotypes missing at random. The simulated sites were crudely matched to real SNP names and hg18 map positions near the MTHFR locus (in an order to match rankwise with real allele frequencies). Some quasi-realistic case/control and continuous phenotypes and covariates were simulated conditional on the genotype data with some (unrealistically) large effect sizes (many OR>2).

multimatch *Match with multiple matching possible.*

Description

Function to represent results of matching when each query argument may match multiple elements in the target.

Usage

```
multimatch(query, target, values, sep = ",", use.unique = TRUE)
```

Arguments

query	Vector of values to be matched.
target	Vector of values that query is to be matched against.
values	Vector of values to be returned for matches in target.
sep	Character to separate values for multiple matches.
use.unique	Logical flag indicating whether to return unique set of values only.

Value

A character vector of length length(query).

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
bmidata <- data.frame(subject = c("A001", "A002", "A003", "A003"),
                      year = c(2001, 2001, 2001, 2005),
                      bmi = c(21.3, 29.7, 25.5, 22.3))
multimatch(c("A002", "A003"), bmidata$subject, bmidata$bmi)
```

multipheno.T2 *Multi-phenotype test for association*

Description

For a set of phenotypes, given summary results for association tests for each single phenotype, over a large fixed set of marker genotypes (e.g. GWAS results), this function calculates a multi-phenotype association test for each marker that is equivalent to using the subject-specific data to perform a multivariate analysis of variance for each marker.

Usage

```
multipheno.T2(z)
```

Arguments

z	a matrix of association Z statistics, with rows corresponding to markers and columns corresponding to phenotypes.

Details

The function is named for the close correspondence with Hotelling's T2 test.

It is the user's responsibility to ensure that the columns of "z" are marginally well calibrated, i.e. over a set of null markers, each column of "z" should be standard normal marginal to the other columns of "z".

Rank deficient situations are currently not handled.

Value

A list with two elements (both of length `nrow(z)`). "`T2`" contains the test statistics, which are chi-squared with `ncol(z)` d.f. under the null, and "`pval`" contains the P-values.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
## Not run:
nsubj <- 400 # number of subjects
nsnp <- 4000 # intended number of SNPs in GWAS

snps <- replicate(nsnp, rbinom(nsubj, 2, rbeta(1, 1.2, 1.2)))
## simulate with random allele frequencies
snps <- snps[ , apply(snps, 2, var) > 0]
nsnp <- ncol(snps) # number after filtering monomorphic

beta <- matrix(rnorm(30, 0, 0.1), ncol = 3)
## matrix of effects of 10 snps on 3 phenotypes

y1 <-  rnorm(nsubj)
y2 <- .1*y1 + rnorm(nsubj)
y3 <- .1*y1 + .3*y2 + rnorm(nsubj) # simulate correlated errors
y <- cbind(y1, y2, y3) + snps[ , 1:10] %*% beta
## wlog the truely associated snps are 1:10
rm(y1, y2, y3)

astats <- function(ii) {
  with(list(snp = snps[ , ii]),
       c(coef(summary(lm(y[ , 1] ~ snp)))["snp", 3],
         coef(summary(lm(y[ , 2] ~ snp)))["snp", 3],
```

```
        coef(summary(lm(y[ , 3] ~ snp)))["snp", 3],
        summary(manova(y ~ snp))$stats["snp", 6]))
}
system.time(gwas <- t(sapply(1:nsnp, astats)))
## columns 1-3 contain single phenotype Z statistics
## column 4 contains manova P-value
pv <- multipheno.T2(gwas[ , 1:3])$pval

plot(-log10(gwas[ , 4]), -log10(pv), type = "n",
     xlab = "MANOVA -log10(P)", ylab = "Summary statistic -log10(P)", las = 1)
points(-log10(gwas[-(1:10), 4]), -log10(pv[-(1:10)]))
points(-log10(gwas[1:10, 4]), -log10(pv[1:10]), cex = 2, pch = 21, bg = "red")
legend("topleft", pch = c(1, 21), pt.cex = c(1, 2), pt.bg = c("white", "red"),
       legend = c("null SNPs", "associated SNPs"))
## nb approximation gets better as nsnp becomes large, even with small nsubj

## End(Not run)
```

`parse.snps` *Parse text representation of a SNP embedded in flanking sequences.*

Description

Extracts the first occurence of a pattern like [A/T], along with the 5' and 3' flanking sequences.

Usage

```
parse.snps(seqs, seqnames = NULL)
```

Arguments

`seqs`	a (vector of) sequences.
`seqnames`	optional names.

Value

A data frame with seven columns, called seq5p, len5p, poly, alleleA, alleleB, seq3p, and len3p.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(snps.BRCA1)
parse.snps(snps.BRCA1$SourceSeq[1:10])
```

`read.snpdata.impute` *Read genotype dosages in the format output by IMPUTE.*

Description

Reads sample information and genotype data from paired sample and genotype files, as generated by the IMPUTE and IMPUTE2 genotype imputation programs and as used by the SNPTEST program. Returns the data in a standard format (see snpdata) that can be used by other functions in this package.

Usage

```
read.snpdata.impute(samplefile, genofile, phenotypes = NULL)
```

Arguments

`samplefile`	filename for samples, assumed 000Ps format.
`genofile`	filename for genotypes, assumed IMPUTE verbose format.
`phenotypes`	if not null, a data frame of phenotypes to be merged with the genotypes; first two columns will be used to match against first two columns of the samplefile.

Details

The sample file is assumed to have *two* header lines, of which the first header line is assumed to be column names and the second header line is assumed to be the "000Ps" line. The genotype file is assumed to have one line for each SNP, of which the first five columns contain information about the SNP, followed by triplets of numeric values giving the probabilities of genotypes 0, 1, 2, for each sample in turn.

This function will be slow for large input files. Best to use gtool or grep/awk out the relevant lines (SNPs) into a smaller file first.

Value

Returns a list with snpinfo and data slots, see `snpdata`.

`read.snpdata.mach` *Read genotype dosages in the format output by MACH*

Description

Reads snp coding information and genotype data from paired `.mlinfo` and `.mldose` files, as generated by the MACH and minimac genotype imputation programs, and returns the data in a standard format (see snpdata) that can be used by other functions in this package.

Usage

```
read.snpdata.mach(fileroot, tol.af = 0.01, phenotypes = NULL,
                  isuffix = ".mlinfo", dsuffix = ".mldose")
```

Arguments

fileroot	a filename root to which suffixes (my default .mlinfo and .mldose) will be appended.
tol.af	a tolerence for checking allele frequencies between the two files read.
phenotypes	if not NULL, a data frame of phenotypes to be merged with the genotypes; must contain a column called MACHID which is used to match against the first column of the mldose file.
isuffix	a suffix to add to the filename root to make the name of the info file
dsuffix	a suffix to add to the filename root to make the name of the dosage file

Details

This function will (just) work as is with HapMap imputed single chromosomes and 4000 individuals. Need to comment on how to extract large numbers of SNPs before feeding to R.

If isuffix or dsuffix end in .gz extensions, the files will be decompressed on the fly.

Value

Returns a list with snpinfo and data slots, see snpdata.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

References

Information about the MACH and minimac programs for genotype imputation, including their output formats, can be found at http://www.sph.umich.edu/csg/abecasis/MACH.

---

read.snpdata.minimac *Read genotype dosages in the format output by minimac*

Description

Reads snp coding information and genotype data from paired .info.gz and .dose.gz files, as generated by the MACH and minimac genotype imputation programs, and returns the data in a standard format (see snpdata) that can be used by other functions in this package.

Usage

```
read.snpdata.minimac(fileroot, tol.af = 0.01, phenotypes = NULL,
                  isuffix = ".info.gz", dsuffix = ".dose.gz")
```

Arguments

fileroot	a filename root to which suffixes (my default .info.gz and .dose.gz) will be appended.
tol.af	a tolerence for checking allele frequencies between the two files read.
phenotypes	if not NULL, a data frame of phenotypes to be merged with the genotypes; must contain a column called MACHID which is used to match against the first column of the mldose file.
isuffix	a suffix to add to the filename root to make the name of the info file
dsuffix	a suffix to add to the filename root to make the name of the dosage file

Details

This function is just a wrapper for read.snpdata.mach with different default isuffix and dsuffix arguments.

Value

Returns a list with snpinfo and data slots, see snpdata.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

References

Information about the MACH and minimac programs for genotype imputation, including their output formats, can be found at http://www.sph.umich.edu/csg/abecasis/MACH.

---

read.snpdata.plink *Read genotype dosages in the format output by PLINK.*

Description

Reads genotype dosages as output by PLINK using the --recodeA option, and combines these with information about coded *and noncoded* alleles from information output by PLINK using the --freq option.

Usage

```
read.snpdata.plink(fileroot, tol.af = 0.01, phenotypes = NULL)
```

Arguments

fileroot	a filename root to which .frq and .raw extensions will be appended
tol.af	a tolerence for checking allele frequencies between the two files read
phenotypes	if not null, a data frame of phenotypes to be merged with the genotypes; must contain columns called FID and IID

Details

Run PLINK twice, once with `--freq` and once with `--recodeA`, and with *otherwise identical options* for e.g. individual and SNP inclusions and exclusions, in order to make suitable files for this function to read.

Value

Returns a list with snpinfo and data slots, see `snpdata`.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

References

Information about PLINK and its input and output data formats can be found at http://pngu.mgh.harvard.edu/~purcell/plink.

Examples

```
## Not run:
mypheno <- read.table("MYPHENO.dat", header = TRUE,
                      as.is = TRUE, na.strings = c("NA", "-9"))
system("plink --bfile MYBED --freq --out MYEXPORT")
system("plink --bfile MYBED --recodeA --out MYEXPORT")
mydata <- read.snpdata.plink("MYEXPORT", phenotypes = mypheno)

## End(Not run)
```

---

remap.q2t — *Remap coordinates from BLAT query sequence to BLAT target sequence.*

Description

For a single BLAT match of a query sequence against a target sequence, a vector of nucleotide positions in the query sequence are remapped to the corresponding nucleotide positions in the target sequence.

Usage

```
remap.q2t(qpos, blatres)
```

Arguments

`qpos`	a vector of 0-offset positions in the query sequence.
`blatres`	a single match from BLAT.

Value

A vector of remapped positions in the target sequence.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

`sanitise.whitespace` *Remove leading and trailing spaces; convert double spaces to single spaces.*

Description

Convenience function for a series of gsub() calls.

Usage

```
sanitise.whitespace(tt)
```

Arguments

`tt`	a character vector.

Value

A character vector with leading and trailing spaces removed, and double spaces converted to single spaces.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
sanitise.whitespace(c("  A to G", "A to  G", "A to G", "A   to G   "))
```

snpdata *Class for SNP genotype and phenotype data.*

Description

Objects of this class contain subject-specific SNP genotype or allele dosage data along with subject-specific phenotypes data, and SNP coded/noncoded allele information.

Usage

```
is.snpdata(object)
as.snpdata(object)
## S3 method for class 'snpdata'
summary(object, ...)
```

Arguments

object	an object to test or coerce to snpdata class.
...	other arguments to summary are ignored.

Details

An object of class snpdata is a list with at least two elements, called "snpinfo" and "data". snpdata$snpinfo must be a data frame with (at least) columns called "snp", "coded.allele" and "noncoded.allele". snpdata$data must be a data frame with columns for each SNP in snpdata$snpinfo with column names formed by pasting together the snp and coded.allele with a separating underscore character, e.g. "rs12345_A".

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

See Also

Widely used formats for measured or imputed SNP genotype data are read by the read.snpdata.plink, read.snpdata.mach and read.snpdata.impute functions, which all return data in the format described above.

Examples

```
## Not run:
align.snpdata.coding(params, snpdata)
grs.make.scores(params, snpdata)
grs.onesnp.apply(params, object)
make.moments2(params, phenolist, snpdata)

## End(Not run)
```

snphwe *Exact test of Hardy–Weinberg.*

Description

Computes a p-value for the exact test of Hardy–Weinberg proportions, for genotype counts for a biallelic locus, as described by Wiggington et al. (2005)

Usage

```
snphwe(g)
```

Arguments

g	Vector of genotypes.

Details

This function is a wrapper for `snphweCounts`, which uses code by Wiggington et al. to compute a P-value for an exact test of Hardy–Weinberg proportions.

This function requires that when g is sorted, the order is hom/het/hom. For example g can be additively coded genotypes (0, 1, 2), or text coded genotypes like AA, AC, CC. For text coded genotypes the heterozygote MUST ALWAYS be coded the same way, e.g. a mixture of AC and CA are not allowed.

Value

The p-value.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
snphwe(rbinom(100, 2, 0.2))
```

snphweCounts *Exact test of Hardy–Weinberg.*

Description

Computes a p-value for the exact test of Hardy–Weinberg proportions, for genotype counts for a biallelic locus, as described by Wiggington et al. (2005).

Usage

```
snphweCounts(obs_hets, obs_hom1, obs_hom2)
```

Arguments

obs_hets	Observed count of the heterozygote genotype.
obs_hom1	Observed count of one homozygote genotype.
obs_hom2	Observed count of the other homozygote genotype.

Details

This is the original code of Wiggington et al. (see references below), with minor modifications. The original (and perhaps counterintuitive) order of arguments has been preserved. It is better to use the snphwe wrapper function.

Value

The p-value.

References

The method and algorithm used are described by Wiggington, Cutler and Abecasis (2005; A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics. 76(5):887-893) available at http://www.ncbi.nih.gov/pmc/articles/PMC1199378. The code of Wiggington et al. that is used here was downloaded from http://www.sph.umich.edu/csg/abecasis/Exact. With appropriate citation, this code is freely available for use and can be incorporated into other programs.

Examples

```
snphwe(rbinom(100, 2, 0.2))
```

snps.BRCA1 *Genotyping array annotation for SNPs near the BRCA1 gene.*

Description

Subset of the array annotation for the Illumina HumanOmniExpressExome array, at positions chr17:41,145,568-41,328,243, which covers (by a generous margin) all transcripts for the BRCA1 gene.

Usage

```
data(snps.BRCA1)
```

Format

`snps.BRCA1` is a data frame.

Source

http://www.illumina.com/support/array/array_kits/infinium_humanomniexpressexome_beadchip_kit/downloads.ilmn

stepdown.moments2 *Stepwise downward model selection using summary statistic matrix.*

Description

Terms are dropped iteratively from a regression model until the reduction of improvement in fit (judged by the P-value for a partial t-test or score test) is signicant at a specified threshold. The method implemented here makes use of a pre-built sufficient summary statistic matrix, which contains the (weighted) second moments between all the variables that are being assessed for inclusion in the regression model. The calculations are exact for a normal linear model and correspond to a score test for a generalised linear model.

Usage

```
stepdown.moments2(xtwx, leftvar, biggest, smallest,
                  p.thresh = 0.05, n = NULL, vscale = NULL)
```

Arguments

`xtwx`	a matrix of (weighted) second moments, typically built using `make.moments2`.
`leftvar`	name of the response variable (the left hand side of the formula).
`biggest`	name(s) of the explanatory variables in the biggest model to fit, i.e.\ variables to *consider* for inclusion.

smallest	name(s) of the explanatory variables in the smallest model to fit, i.e.\ variables that *must* be included.
p.thresh	P-value threshold for proceeding to add a term to the model.
n	sample size, only needed for the normal linear model if there is not a single intercept "ONE" for all individuals.
vscale	parameter, set to NULL for normal linear model and 1 for logistic regression.

Details

This performs stepwise downward model selection. Significance of terms considered for inclusion is determined using identical calculations to lm.moments2 and est.moments2.

When the vscale argument is NULL this function assumes that the xtwx argument was calculated with unit weights and therefore that a linear model fit is required with error variance estimated from the data.

When the vscale argument is set equal to 1 this function assumes that the xtwx argument was calculated with weights calculated such that a correct likelihood function can be recovered and therefore that a generalised linear model fit is required.

Values other than NULL or 1 for the vscale parameter may not be what you think. Do not use other values unless you are absolutely sure that you understand what are doing. See the manuscript for details.

Value

The fitted model, as returned by calling lm.moments2 or est.moments2. This is a list with slots for the effect size estimates, standard errors, and a precision matrix.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
xtx <- make.moments2(mthfr.params, c("SBP", "DBP", "SexC", "Age"), mthfrex)
allsnps <- paste(mthfr.params$snp, mthfr.params$coded.allele, sep = "_")
myfit <- stepdown.moments2(xtx, "SBP", allsnps, c("ONE", "SexC", "Age"))
## much faster than stepAIC but only steps to bigger models
cbind(beta = myfit$betahat, se = myfit$se, t = myfit$beta/myfit$se)
```

`stepup.moments2` *Stepwise upward model selection using summary statistic matrix.*

Description

Terms are added iteratively to a regression model until the improvement in fit (judged by the P-value for a partial t-test or score test) is no longer signicant at a specified threshold. The method implemented here makes use of a pre-built sufficient summary statistic matrix, which contains the (weighted) second moments between all the variables that are being assessed for inclusion in the regression model. The calculations are exact for a normal linear model and correspond to a score test for a generalised linear model.

Usage

```
stepup.moments2(xtwx, leftvar, biggest, smallest,
                p.thresh = 0.05, n = NULL, vscale = NULL)
```

Arguments

`xtwx`	a matrix of (weighted) second moments, typically built using `make.moments2`.
`leftvar`	name of the response variable (the left hand side of the formula).
`biggest`	name(s) of the explanatory variables in the biggest model to fit, i.e.\ variables to *consider* for inclusion.
`smallest`	name(s) of the explanatory variables in the smallest model to fit, i.e.\ variables that *must* be included.
`p.thresh`	P-value threshold for proceeding to add a term to the model.
`n`	sample size, only needed for the normal linear model if there is not a single intercept "`ONE`" for all individuals.
`vscale`	parameter, set to `NULL` for normal linear model and 1 for logistic regression.

Details

This performs stepwise upward model selection. Significance of terms considered for inclusion is determined using identical calculations to `lm.moments2` and `est.moments2`.

When the `vscale` argument is `NULL` this function assumes that the `xtwx` argument was calculated with unit weights and therefore that a linear model fit is required with error variance estimated from the data.

When the `vscale` argument is set equal to 1 this function assumes that the `xtwx` argument was calculated with weights calculated such that a correct likelihood function can be recovered and therefore that a generalised linear model fit is required.

Values other than `NULL` or 1 for the `vscale` parameter may not be what you think. Do not use other values unless you are absolutely sure that you understand what are doing. See the manuscript for details.

Value

The fitted model, as returned by calling lm.moments2 or est.moments2. This is a list with slots for the effect size estimates, standard errors, and a precision matrix.

Author(s)

Toby Johnson <Toby.x.Johnson@gsk.com>

Examples

```
data(mthfrex)
xtx <- make.moments2(mthfr.params, c("SBP", "DBP", "SexC", "Age"), mthfrex)
allsnps <- paste(mthfr.params$snp, mthfr.params$coded.allele, sep = "_")
myfit <- stepup.moments2(xtx, "SBP", allsnps, c("ONE", "SexC", "Age"))
## much faster than stepAIC but only steps to bigger models
cbind(beta = myfit$betahat, se = myfit$se, t = myfit$beta/myfit$se)
```

---

t2d.scores *Genetic risk scores for type 2 diabetes risk.*

Description

Risk score parameterised using GWAS meta-analysis results published by the DIAGRAM+ consortium (Voight et al. Nature Genetics 2011). The score is for risk of prevalent type 2 diabetes (T2D).

Usage

```
data(t2d.scores)
```

Format

A data frame suitable for use with other functions in this package, see gtx.params.

The score is called T2D2010, and has coefficients in ln(odds) for T2D per coded allele.

In future releases, additional risk scores used in earlier papers may be added for historical interest (WHII and Lin papers).

Source

For the publication by Voight et al. from which these data were extracted see http://dx.doi.org/10.1038/ng.609.

t2dex *Simulated example genotype and longitudinal phenotype data.*

Description

A simulated example genotype and phenotype dataset for genetic risk score analyses, consisting of genotypes for 2000 individuals at 31 SNPs, baseline case/control and covariate status, and a survival phenotype.

Usage

```
data(t2dex)
```

Format

`t2dex` is a list suitable for use with other functions in this package, see snpdata.

Details

This is a simulated dataset, whose sole purpose is to illustrate the use of functions in this package for genetic risk score analyses.

The dataset is provided so that the usage examples can actually be run, without burdening each example with many lines of code to generate an analysable dataset.

The genotype data were simulated assuming exact Hardy-Weinberg and linkage equilibrium. The phenotypes were simulated using a crude parametric model of a longitudinal study, intended to be used in regression models as if:

- `t2dex$data$Age` is age at recruitment into the study.
- `t2dex$data$Overweight` 0/1 indicator for BMI>25 at recruitment.
- `t2dex$data$T2D` 0/1 indicator for prevalent T2D at recruitment.
- `t2dex$data$FollowupDays` days subject followed up for, until incident T2D event or followup stopped.
- `t2dex$data$FollowupT2D` 0/1 indicator for incident T2D event.

Source

The genotypes were simulated using allele frequencies from t2d.scores.

Examples

```
data(t2dex)
summary(subset(t2dex$data, select = c("Age", "Overweight", "T2D",
                                      "FollowupDays", "FollowupT2D")))

library(survival)
plot(survfit(Surv(FollowupDays,FollowupT2D) ~ Overweight,
                 data = t2dex$data), col = c("green", "red"))
```

```
mycoxph <- coxph(Surv(FollowupDays,FollowupT2D) ~ Overweight,
                 data = t2dex$data) # fit null model
data(t2d.scores)
assoc1 <- grs.onesnp.apply(t2d.scores, mycoxph) # single SNP association
## risk score fit from single SNPs
unlist(grs.summary(t2d.scores$coef, assoc1$beta, assoc1$se,
                   n = length(residuals(mycoxph))))

## compare direct analysis of subject-specific data
t2dex <- grs.make.scores(t2d.scores, t2dex)
coxph(Surv(FollowupDays,FollowupT2D) ~ Overweight + T2D2010.score,
      data = t2dex$data)
```

Index