Package 'phers'

December 15, 2022

Type Package

Title Calculate Phenotype Risk Scores

Version 1.0.0

Description Use phenotype risk scores based on linked clinical and genetic data to study Mendelian disease and rare genetic variants. See Bastarache et al. 2018 <doi:10.1126/science.aal4043>.

URL https://phers.hugheylab.org, https://github.com/hugheylab/phers

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.5)

Imports BEDMatrix (>= 2.0.3), checkmate (>= 2.0.0), data.table (>= 1.5.0), foreach (>= 1.5.2), iterators (>= 1.0.14), survival (>= 3.3.1), speedglm (>= 0.3.4)

Suggests doParallel (>= 1.0.17), knitr, rmarkdown, testthat (>= 3.1.0), qs (>= 0.25.2)

Config/testthat/edition 3

NeedsCompilation no

Author Jake Hughey [aut, cre], Layla Aref [aut]

Maintainer Jake Hughey <jakejhughey@gmail.com>

Repository CRAN

Date/Publication 2022-12-14 23:20:02 UTC

R topics documented:

`demoSample` *Sample table of demographic information*

Description

The data are artificial and do not correspond to real patients.

Usage

```
demoSample
```

Format

A data table with the following columns:

- `person_id`: Character vector of the identifier for each person in the cohort
- `sex`: Character vector indicating biological sex

See Also

`getWeights()`, `getScores()`

diseaseDxIcdMap *Mapping of diseases and diagnostic ICD codes*

Description

This table provides a mapping between 27 Mendelian diseases and the corresponding ICD-9 and ICD-10 codes that indicate a genetic diagnosis.

Usage

```
diseaseDxIcdMap
```

Format

A data.table with the following columns:

- `disease_id`: Numeric vector of OMIM disease identifiers
- `disease_name`: Character vector of disease names
- `icd`: Character vector of ICD codes indicating a genetic diagnosis
- `flag`: Integer vector of the vocabulary of the ICD code (**9**: ICD-9-CM, **10**: ICD-10-CM)
- `icd_name`: Character vector containing the description of each ICD code

See Also

`getPhecodeOccurrences()`, `getDxStatus()`

diseaseHpoMap *Mapping of Mendelian diseases and their clinical features*

Description

This table provides a mapping between Mendelian diseases and their clinical features, represented as Human Phenotype Ontology (HPO) terms. The mapping is based on annotations from Online Mendelian Inheritance in Man (OMIM).

Usage

```
diseaseHpoMap
```

Format

A data.table with the following columns:

- `disease_id`: Numeric vector of OMIM disease identifiers
- `disease_name`: Character vector of disease names
- `hpo_term_id`: Character vector of HPO identifiers corresponding to each disease's clinical features
- `hpo_term_name`: Character vector of HPO term descriptions

Source

https://hpo.jax.org/app/download/annotation

See Also

mapDiseaseToPhecode()

---

getDxStatus *Identify cases and controls for Mendelian diseases*

Description

This function is useful for verifying that raw or residual phenotype risk scores of diagnosed individuals (cases) tend to be higher than scores of undiagnosed individuals (controls).

Usage

```
getDxStatus(
  demos,
  icdOccurrences,
  minUniqueAges = 2L,
  diseaseDxIcdMap = phers::diseaseDxIcdMap
)
```

Arguments

demos	A data.table having one row per person in the cohort. Must have a column person_id.
icdOccurrences	A data.table of occurrences of ICD codes for each person in the cohort. Must have columns person_id, icd, flag, and occurrence_age.
minUniqueAges	Integer indicating the minimum number of unique ICD code entry ages required to classify a person as a case. Persons with at least one, but fewer than minUniqueAges entry ages, are assigned as neither cases nor controls.
diseaseDxIcdMap	A data.table of the mapping between diseases and the corresponding ICD codes that indicate a diagnosis. Must have columns disease_id, icd, and flag. Default is diseaseDxIcdMap.

Value

A data.table with columns person_id, disease_id, and dx_status (1 indicates a case, 0 indicates a control, -1 indicates neither).

Examples

```
library('data.table')


icdSample1 = merge(icdSample, demoSample[, .(person_id, dob)], by = 'person_id')
icdSample1[, occurrence_age := as.numeric((entry_date - dob)/365.25)]
icdSample1[, `:=`(entry_date = NULL, dob = NULL)]

dxStatus = getDxStatus(demoSample, icdSample1)
```

---

getGeneticAssociations

Perform association tests between phenotype risk scores and genotypes

---

Description

The association test for each disease-variant pair is based on a linear model, with the phenotype risk score as the dependent variable.

Usage

```
getGeneticAssociations(
  scores,
  genotypes,
  demos,
  diseaseVariantMap,
  lmFormula,
  modelType = c("genotypic", "additive", "dominant", "recessive"),
  level = 0.95,
  dopar = FALSE
)
```

Arguments

`scores`	A data.table of phenotype risk scores. Must have columns `person_id`, `disease_id`, `score`.
`genotypes`	A matrix or 'BEDMatrix' object containing genetic data, with rownames corresponding to `person_id`s in `demos` and `scores`, and colnames corresponding to `variant_id`s in `diseaseVariantMap`.
`demos`	A data.table of characteristics for each person in the cohort. Must have column `person_id`.
`diseaseVariantMap`	A data.table indicating which genetic variants to test for association with phenotype risk scores for which diseases. Must have columns `disease_id` and `variant_id`.

lmFormula	A formula representing the linear model (excluding the term for genotype) to use for the association tests. All terms in the formula must correspond to columns in demos.
modelType	A string indicating how to encode genotype in the model.
level	A number indicating the level of the confidence interval. Default is 0.95.
dopar	Logical indicating whether to run calculations in parallel if a parallel backend is already set up, e.g., using doParallel::registerDoParallel(). Recommended to minimize runtime.

Value

A data.table of statistics for the association tests (if a model fails to converge, NAs will be reported):

- disease_id: Disease identifier
- variant_id: Variant identifier
- n_total: Number of persons with non-missing genotype data for the given variant.
- n_wt: Number of persons homozygous for the wild-type allele.
- n_het: Number of persons having one copy of the alternate allele.
- n_hom: Number of persons homozygous for the alternate allele.
- beta: Coefficient for the association of genotype with score
- se: Standard error for beta
- pval: P-value for beta being non-zero
- ci_lower: Lower bound of the confidence interval for beta
- ci_upper: Upper bound of the confidence interval for beta

If modelType is "genotypic", the data.table will include separate statistics for heterozygous and homozygous genotypes.

See Also

stats::lm(), stats::confint(), getScores()

Examples

```
library('data.table')
library('BEDMatrix')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights
weights = getWeights(demoSample, phecodeOccurrences)

# OMIM disease IDs for which to calculate phenotype risk scores
diseaseId = 154700

# map diseases to phecodes
```

```
diseasePhecodeMap = mapDiseaseToPhecode()

# calculate scores
scores = getScores(weights, diseasePhecodeMap[disease_id == diseaseId])

# map diseases to genetic variants
nvar = 10
diseaseVariantMap = data.table(disease_id = diseaseId, variant_id = paste0('snp', 1:nvar))

# load sample genetic data
npop = 50
genoSample = BEDMatrix(system.file('extdata', 'geno_sample.bed', package = 'phers'))
colnames(genoSample) = paste0('snp', 1:nvar)
rownames(genoSample) = 1:npop

# run genetic association tests
genoStats = getGeneticAssociations(
  scores, genoSample, demoSample, diseaseVariantMap, lmFormula = ~ sex,
  modelType = 'additive')
```

`getPhecodeOccurrences` *Map ICD code occurrences to phecode occurrences*

Description

This is typically the first step of an analysis using phenotype risk scores, the next is `getWeights()`.

Usage

```
getPhecodeOccurrences(
  icdOccurrences,
  icdPhecodeMap = phers::icdPhecodeMap,
  dxIcd = phers::diseaseDxIcdMap
)
```

Arguments

`icdOccurrences`	A data.table of occurrences of ICD codes for each person in the cohort. Must have columns `person_id`, `icd`, and `flag`.
`icdPhecodeMap`	A data.table of the mapping between ICD codes and phecodes. Must have columns `icd`, `phecode`, and `flag`. Default is the map included in this package.
`dxIcd`	A data.table of ICD codes to exclude from mapping to phecodes. Must have columns `icd` and `flag`. Default is the table of Mendelian diseases and the corresponding ICD codes that indicate a genetic diagnosis. If `NULL`, no ICD codes will be excluded.

Value

A data.table of phecode occurrences for each person.

See Also

getWeights(), getScores()

Examples

```
library('data.table')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights (using the prevalence method)
weights = getWeights(demoSample, phecodeOccurrences)

# OMIM disease IDs for which to calculate phenotype risk scores
diseaseId = 154700

# map diseases to phecodes
diseasePhecodeMap = mapDiseaseToPhecode()

# calculate scores
scores = getScores(weights, diseasePhecodeMap[disease_id == diseaseId])

# calculate residual scores
rscores = getResidualScores(demoSample, scores, lmFormula = ~ sex)
```

---

getResidualScores *Calculate residual phenotype risk scores*

Description

The residual score indicates to what extent a person's phenotype risk score for a given disease deviates from the expected score, after adjusting for the person's characteristics in a linear model.

Usage

```
getResidualScores(demos, scores, lmFormula)
```

Arguments

demos	A data.table of characteristics for each person in the cohort. Must have column person_id.
scores	A data.table containing the phenotype risk score for each person for each disease. Must have columns person_id, disease_id, and score.
lmFormula	A formula representing the linear model to use for calculating residual scores. All terms in the formula must correspond to columns in demos.

Value

A data.table, based on `scores`, with an additional column `resid_score`. Residual scores for each disease are standardized to have unit variance.

See Also

`stats::rstandard()`, `getScores()`

Examples

```
library('data.table')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights (using the prevalence method)
weights = getWeights(demoSample, phecodeOccurrences)

# OMIM disease IDs for which to calculate phenotype risk scores
diseaseId = 154700

# map diseases to phecodes
diseasePhecodeMap = mapDiseaseToPhecode()

# calculate scores
scores = getScores(weights, diseasePhecodeMap[disease_id == diseaseId])

# calculate residual scores
rscores = getResidualScores(demoSample, scores, lmFormula = ~ sex)
```

---

getScores *Calculate phenotype risk scores*

Description

A person's phenotype risk score for a given disease corresponds to the sum of the weights of the disease-relevant phecodes that the person has received.

Usage

```
getScores(weights, diseasePhecodeMap)
```

Arguments

`weights`	A data.table of phecodes and their corresponding weights. Must have columns `person_id`, `phecode` and `w`.
`diseasePhecodeMap`	A data.table of the mapping between diseases and phecodes. Must have columns `disease_id` and `phecode`.

Value

A data.table containing the phenotype risk score for each person for each disease.

See Also

mapDiseaseToPhecode(), getPhecodeOccurrences(), getWeights(), getResidualScores()

Examples

```
library('data.table')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights (using the prevalence method)
weights = getWeights(demoSample, phecodeOccurrences)

# OMIM disease IDs for which to calculate phenotype risk scores
diseaseId = 154700

# map diseases to phecodes
diseasePhecodeMap = mapDiseaseToPhecode()

# calculate scores
scores = getScores(weights, diseasePhecodeMap[disease_id == diseaseId])

# calculate residual scores
rscores = getResidualScores(demoSample, scores, lmFormula = ~ sex)
```

---

getWeights *Calculate phecode-specific weights for phenotype risk scores*

Description

This is typically the second step of an analysis using phenotype risk scores, the next is getScores().

Usage

```
getWeights(
  demos,
  phecodeOccurrences,
  method = c("prevalence", "logistic", "cox", "loglinear", "prevalence_precalc"),
  methodFormula = NULL,
  negativeWeights = FALSE,
  dopar = FALSE
)
```

Arguments

demos	A data.table having one row per person in the cohort. Must have a column person_id. When the cox method is used, demos must have columns first_age and last_age corresponding to first and last age of visit (in years).
phecodeOccurrences	A data.table of phecode occurrences for each person in the cohort. Must have columns person_id and phecode under the "prevalence" or "logistic" methods, columns person_id, phecode, and num_occurrences under the "loglinear" method, and columns person_id, phecode, and occurrence_age under the "cox" method. num_occurrences refers to the number of unique dates a phecode was recorded for a person. occurrence_age refers to the first age (in years) a person acquired a phecode.
method	A string indicating the statistical model for calculating weights.
methodFormula	A formula representing the right-hand side of the model corresponding to method. All terms in the formula must correspond to columns in demos. A method formula is not required for the "prevalence" and "prevalence_precalc" methods. Do not use age-related covariates with the "cox" method.
negativeWeights	Logical indicating whether to allow negative weights for individuals with no occurrences of a phecode. This option is not required for the "loglinear" method since under this method, individuals with a nonzero phecode occurrence can also have negative weights.
dopar	Logical indicating whether to run calculations in parallel if a parallel backend is already set up, e.g., using doParallel::registerDoParallel(). Recommended to minimize runtime.

Value

A data.table with columns person_id, phecode, pred, and w. The column pred represents a different quantity depending on method. Under the "prevalence" method, it is fraction of the cohort that has at least one occurrence of the given phecode. The "prevalence_precalc" method is similar to the "prevalence" method but pred is calculated based on EHR data from the Vanderbilt University Medical Center. Under "logistic" or "cox" method, it is the predicted probability of given individual having a given phecode based on methodFormula. Under the "loglinear" method, it is the predicted log2(num_occurrences + 1) of a given phecode for a given individual based on methodFormula. For the "prevalence", "prevalence_precalc", "cox", and "logistic" methods, weight is calculated as -log10(pred) when an individual has non-zero phecode occurrence and log10(1 - pred) when an individual has zero phecode occurrence. For the "loglinear" method weight is calculated as the difference between the observed log2(num_occurrences + 1) and pred.

See Also

getPhecodeOccurrences(), getScores()

Examples

```
library('data.table')
```

```
library('survival')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights using the prevalence method
weightsPrev = getWeights(demoSample, phecodeOccurrences)

# calculate weights using the prevalence method
# (assign negative weights to those with zero phecode occurrence)
weightsPrevNeg = getWeights(
  demoSample, phecodeOccurrences, negativeWeights = TRUE)

# calculate weights using the logistic method
weightsLogistic = getWeights(
  demoSample, phecodeOccurrences, method = 'logistic', methodFormula = ~ sex)

# calculate weights using the loglinear method
phecodeOccurrences2 = phecodeOccurrences[, .(
  num_occurrences = uniqueN(entry_date)), by = .(person_id, phecode)]
weightsLoglinear = getWeights(
  demoSample, phecodeOccurrences2, method = 'loglinear', methodFormula = ~ sex)

# calculate weights using the cox method
phecodeOccurrences3 = phecodeOccurrences[, .(
  first_occurrence_date = min(entry_date)) , by = .(person_id, phecode)]
phecodeOccurrences3 = merge(
  phecodeOccurrences3, demoSample[, .(person_id, dob)], by = 'person_id')
phecodeOccurrences3[,
  occurrence_age := as.numeric((first_occurrence_date - dob)/365.25)]
phecodeOccurrences3[, `:=`(first_occurrence_date = NULL, dob = NULL)]
demoSample3 = demoSample[, .(
  person_id, sex,
  first_age = as.numeric((first_visit_date - dob)/365.25),
  last_age = as.numeric((last_visit_date - dob)/365.25))]
weightsCox = getWeights(
  demoSample3, phecodeOccurrences3, method = 'cox', methodFormula = ~ sex)

# calculate weights using pre-calculated weights based on data from
# Vanderbilt University Medical Center
weightsPreCalc = getWeights(
  demoSample, phecodeOccurrences, method = 'prevalence_precalc')
```

---

hpoPhecodeMap *Mapping of HPO terms and phecodes*

Description

This table provides a mapping between Human Phenotype Ontology (HPO) terms and phecodes, and is useful for using phecodes to represent the clinical features of Mendelian diseases (version 1.2).

Usage

```
hpoPhecodeMap
```

Format

A data.table with the following columns:

- `hpo_term_id`: Character vector of HPO term identifiers
- `hpo_term_name`: Character vector of HPO term descriptions
- `phecode`: Character vector of phecodes
- `phecode_name`: Character vector of phecode descriptions

See Also

`mapDiseaseToPhecode()`

`icdPhecodeMap` *Mapping of ICD codes and phecodes*

Description

This table provides a mapping between International Classification of Diseases 9th and 10th revisions (ICD-9-CM and ICD-10-CM) and phecodes (version 1.2).

Usage

```
icdPhecodeMap
```

Format

A data.table with the following columns:

- `icd`: Character vector of ICD codes
- `flag`: Integer vector of the vocabulary of the ICD code (**9**: ICD-9-CM, **10**: ICD-10-CM)
- `icd_name`: Character vector of ICD code descriptions
- `phecode`: Character vector of phecodes
- `phecode_name`: Character vector of phecode descriptions

Source

https://phewascatalog.org/phecodes

See Also

`getPhecodeOccurrences()`

icdSample *Sample table of ICD occurrences*

Description

The data are artificial and do not correspond to real patients.

Usage

```
icdSample
```

Format

A data.table with the following columns:

- `person_id`: Character vector of the identifier for each person
- `icd`: Character vector of the ICD codes recorded for each person
- `flag`: Integer vector of the vocabulary of the ICD code (**9**: ICD-9-CM, **10**: ICD-10-CM)
- `entry_date`: Vector of type `Date` indicating the date each ICD code was recorded.

See Also

`getPhecodeOccurrences()`, `getWeights()`, `getScores()`

mapDiseaseToPhecode *Map diseases to phecodes via HPO terms*

Description

A mapping of diseases to their clinical features, represented as phecodes, is required for calculating phenotype risk scores.

Usage

```
mapDiseaseToPhecode(
  diseaseHpoMap = phers::diseaseHpoMap,
  hpoPhecodeMap = phers::hpoPhecodeMap
)
```

Arguments

`diseaseHpoMap`	A data.table containing the mapping between diseases and HPO terms. Must have columns `disease_id` and `term_id`. Default is the map included in this package.
`hpoPhecodeMap`	A data.table containing the mapping between HPO terms and phecodes. Must have columns `term_id` and `phecode`. Default is the map included in this package.

Value

A data.table with columns `disease_id` and `phecode`.

See Also

`getScores()`

Examples

```
library('data.table')
library('survival')

# map ICD codes to phecodes
phecodeOccurrences = getPhecodeOccurrences(icdSample)

# calculate weights using the prevalence method
weightsPrev = getWeights(demoSample, phecodeOccurrences)

# calculate weights using the prevalence method
# (assign negative weights to those with zero phecode occurrence)
weightsPrevNeg = getWeights(
  demoSample, phecodeOccurrences, negativeWeights = TRUE)

# calculate weights using the logistic method
weightsLogistic = getWeights(
  demoSample, phecodeOccurrences, method = 'logistic', methodFormula = ~ sex)

# calculate weights using the loglinear method
phecodeOccurrences2 = phecodeOccurrences[, .(
  num_occurrences = uniqueN(entry_date)), by = .(person_id, phecode)]
weightsLoglinear = getWeights(
  demoSample, phecodeOccurrences2, method = 'loglinear', methodFormula = ~ sex)

# calculate weights using the cox method
phecodeOccurrences3 = phecodeOccurrences[, .(
  first_occurrence_date = min(entry_date)) , by = .(person_id, phecode)]
phecodeOccurrences3 = merge(
  phecodeOccurrences3, demoSample[, .(person_id, dob)], by = 'person_id')
phecodeOccurrences3[,
  occurrence_age := as.numeric((first_occurrence_date - dob)/365.25)]
phecodeOccurrences3[, `:=`(first_occurrence_date = NULL, dob = NULL)]
demoSample3 = demoSample[, .(
  person_id, sex,
  first_age = as.numeric((first_visit_date - dob)/365.25),
  last_age = as.numeric((last_visit_date - dob)/365.25))]
weightsCox = getWeights(
  demoSample3, phecodeOccurrences3, method = 'cox', methodFormula = ~ sex)

# calculate weights using pre-calculated weights based on data from
# Vanderbilt University Medical Center
weightsPreCalc = getWeights(
  demoSample, phecodeOccurrences, method = 'prevalence_precalc')
```

`preCalcWeights` *Pre-calculated weights for calculating phenotype risk scores*

Description

The weights are based on EHR data from the Vanderbilt University Medical Center Synthetic Derivative (SD) and ICD-phecode map version 1.2 and are calculated using the "prevalence" method.

Usage

```
preCalcWeights
```

Format

A data.table with the following columns:

- `phecode`: Character vector of phecodes
- `prev`: Numeric vector of prevalences, i.e., fraction of subjects in the SD that have at least one occurrence of the given phecode
- `w`: Numeric vector of weights, calculated as `-log10(prev)`

See Also

`getWeights()`

Index