

Package: BreedingExperiment (via r-universe)

October 4, 2026

Title Container and Relationship Matrices for Breeding and
Quantitative Genomics

Version 0.99.4

Description Provides the BreedingExperiment class, a container that keeps marker genotypes, phenotypes, pedigree and marker coordinates together in one object, extending RangedSummarizedExperiment so that markers are addressable as genomic ranges. Relationship matrices central to quantitative genetics are computed as methods on the container: the pedigree numerator relationship matrix of Henderson (1976) <doi:10.2307/2529339>, the genomic relationship matrix of VanRaden (2008) <doi:10.3168/jds.2007-0980>, the dominance relationship matrix of Vitezica, Varona and Legarra (2013) <doi:10.1534/genetics.113.155176>, and the single-step matrix combining pedigree and genomic information of Legarra, Aguilar and Misztal (2009) <doi:10.3168/jds.2009-2061> and Christensen and Lund (2010) <doi:10.1186/1297-9686-42-2>, which allows genotyped and ungenotyped individuals to be analysed together as in Aguilar and others (2010) <doi:10.3168/jds.2009-2730>. Marker quality control, imputation, inbreeding coefficients and allele frequency summaries are included.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

Depends R (>= 4.5.0), SummarizedExperiment

Imports methods, stats, utils, S4Vectors, GenomicRanges, IRanges, BiocGenerics

Suggests testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

biocViews Software, Genetics, SNP, GenomicVariation, DataRepresentation, Infrastructure, Regression, QualityControl

URL <https://github.com/mqfarooqi1/BreedingExperiment>,
<https://mqfarooqi1.github.io/BreedingExperiment/>

BugReports <https://github.com/mqfarooqi1/BreedingExperiment/issues>
Config/testthat/edition 3
Roxygen list(markdown = TRUE)
Collate 'AllClasses.R' 'AllGenerics.R' 'BreedingExperiment-package.R'
 'constructor.R' 'accessors.R' 'pedigree.R' 'relationship.R'
 'singlestep.R' 'qc.R' 'simulate.R' 'data.R' 'show.R'
Config/roxygen2/version 8.1.0
Config/pak/sysreqs zlib1g-dev
Repository <https://mqfarooqi1.r-universe.dev>
Date/Publication 2026-09-04 14:58:11 UTC
RemoteUrl <https://github.com/mqfarooqi1/BreedingExperiment>
RemoteRef HEAD
RemoteSha c9e013ff73c2fe2b88ae22b5c0f6868393422a43

Contents

alleleFrequency	2
Amatrix	3
BreedingExperiment	4
BreedingExperiment-class	6
demoBreeding	7
Dmatrix	8
filterMarkers	9
genotyped	10
genotypes	11
Gmatrix	12
Hmatrix	13
imputeMarkers	14
inbreeding	15
pedigree	16
simulateBreeding	17
sortPedigree	18
Index	19

alleleFrequency	<i>Allele frequencies and marker summaries</i>
-----------------	--

Description

Computes, for every marker, the frequency of the counted allele, the minor allele frequency, the call rate and the observed heterozygosity.

Usage

```
alleleFrequency(x, ...)

## S4 method for signature 'BreedingExperiment'
alleleFrequency(x, ...)
```

Arguments

`x` A [BreedingExperiment](#) object.

`...` Unused.

Value

A [S4Vectors::DataFrame](#) with one row per marker and columns freq, maf, callRate and heterozygosity.

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 30, n_marker = 50)
head(alleleFrequency(be))
```

Amatrix	<i>Pedigree numerator relationship matrix</i>
---------	---

Description

Builds the additive relationship matrix A from the pedigree using the tabular method. Element A_{ij} is twice the coefficient of kinship between individuals i and j , so a diagonal entry is $1 + F_i$ where F_i is the inbreeding coefficient.

Usage

```
Amatrix(x, ...)

## S4 method for signature 'BreedingExperiment'
Amatrix(x, ids = NULL, ...)
```

Arguments

`x` A [BreedingExperiment](#) object.

`...` Unused.

`ids` Optional character vector of identifiers to return, in that order. Defaults to every individual in the pedigree.

Details

The matrix is built recursively over a pedigree sorted so that parents come first: an individual's relationship to an earlier individual is the average of its parents' relationships to that individual, and its diagonal is $1 + \frac{1}{2}A_{sire,dam}$.

Value

A symmetric numeric matrix with dimnames taken from the pedigree.

References

Henderson, C. R. (1976) "A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values." *Biometrics* 32, 69-83. doi:[10.2307/2529339](https://doi.org/10.2307/2529339)

See Also

[Gmatrix\(\)](#), [Hmatrix\(\)](#), [inbreeding\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 12, n_marker = 30)
A <- Amatrix(be)
round(A[1:5, 1:5], 3)
```

BreedingExperiment	<i>Create a breeding experiment</i>
--------------------	-------------------------------------

Description

Builds a [BreedingExperiment](#) from a marker genotype matrix, with optional phenotypes, pedigree and marker coordinates.

Usage

```
BreedingExperiment(
  genotypes,
  phenotypes = NULL,
  pedigree = NULL,
  rowRanges = NULL,
  ...
)
```

Arguments

genotypes	A numeric matrix of allele dosages, markers in rows and individuals in columns, with column names identifying the individuals.
phenotypes	Optional data frame or S4Vectors::DataFrame of individual-level variables, with one row per column of genotypes.
pedigree	Optional data frame with columns id, sire and dam. Unknown parents may be NA or $\emptyset$.
rowRanges	Optional GenomicRanges::GRanges giving the position of each marker, of the same length as <code>nrow(genotypes)</code> .
...	Further assays, passed to SummarizedExperiment::SummarizedExperiment() .

Details

Genotypes are allele dosages: usually the integers 0, 1 and 2, with NA for missing calls. Fractional values in between are accepted, so expected dosages from imputation can be stored directly. Markers are rows and individuals are columns, matching the Bioconductor convention that features are rows.

If `rowRanges` is not supplied, markers are given placeholder coordinates on a single sequence so that the object is still a valid ranged experiment; pass real coordinates whenever you have them, because that is what allows a data set to be subset by genomic region.

The pedigree may name ancestors that were never genotyped. Those extra individuals are exactly what [Hmatrix\(\)](#) needs in order to combine pedigree and genomic information.

Value

A [BreedingExperiment](#) object.

See Also

[Gmatrix\(\)](#), [Amatrix\(\)](#), [Hmatrix\(\)](#), [simulateBreeding\(\)](#)

Examples

```
set.seed(1)
geno <- matrix(rbinom(40 * 12, 2, 0.3), nrow = 40, ncol = 12,
              dimnames = list(paste0("snp", 1:40), paste0("ind", 1:12)))
pheno <- data.frame(yield = rnorm(12))
be <- BreedingExperiment(geno, phenotypes = pheno)
be
dim(be)
```

BreedingExperiment-class

Breeding experiment container

Description

A BreedingExperiment keeps everything a quantitative-genetic analysis needs in one object: marker genotypes, the phenotypes and design variables of the individuals, the pedigree, and the position of every marker on the genome.

Usage

```
## S4 method for signature 'BreedingExperiment'
show(object)
```

Arguments

object A [BreedingExperiment](#) object.

Details

The class extends [SummarizedExperiment::RangedSummarizedExperiment](#), so markers are rows and individuals are columns, and the familiar accessors work as usual: `assay()` returns the genotype matrix, `rowRanges()` the marker coordinates as a [GenomicRanges::GRanges](#), and `colData()` the phenotypes. Because marker positions are genomic ranges, a breeding data set can be subset by region, overlapped with annotation, and otherwise handled with the standard Bioconductor vocabulary.

One slot is added to the parent class:

pedigree A [S4Vectors::DataFrame](#) with columns `id`, `sire` and `dam`, giving the parents of each individual. Unknown parents are NA. It may contain ancestors that were never genotyped, which is what makes single-step analysis possible.

Genotypes are stored as allele dosages: 0, 1 or 2 copies of the counted allele, with NA for missing calls.

Value

A class definition, so nothing is returned by the class itself. The constructor [BreedingExperiment\(\)](#) returns a BreedingExperiment object, and the `show` method is called for the summary it prints, returning object invisibly.

Construction

Use [BreedingExperiment\(\)](#); see its help page for examples.

Display

The show method prints the size of the object, the assays it holds, the phenotype columns, the sequences the markers lie on, and a summary of the pedigree including how many individuals are genotyped.

See Also

[BreedingExperiment\(\)](#), [Amatrix\(\)](#), [Gmatrix\(\)](#), [Hmatrix\(\)](#)

Examples

```
data(demoBreeding)
demoBreeding
```

demoBreeding

A demonstration breeding population

Description

A small, five-generation breeding population supplied so that every function in the package can be tried on data that behaves like a real programme.

Usage

```
data(demoBreeding)
```

Format

A [BreedingExperiment](#) with 900 markers and 120 genotyped individuals, and a pedigree of 180. colData holds:

generation Generation number, 3 to 5 for genotyped individuals.

sex "F" or "M".

yield A moderately heritable trait ($h^2 \approx 0.35$), recorded on females only.

stature A more heritable trait ($h^2 \approx 0.60$), recorded on everyone.

trueBV_yield, trueBV_stature The simulated true breeding values.

Details

The data are **simulated**, not observed. They are built to have the awkward features that real breeding data has and that tidy textbook examples lack:

- **Only part of the population is genotyped.** The two earliest generations were born before genotyping began, so 60 of the 180 individuals in the pedigree have no markers. This is the situation [Hmatrix\(\)](#) exists for.
- **Unequal family sizes**, because a few sires are used heavily.

- **A trait recorded on one sex only.** yield is missing for males, as a milk trait would be.
- **Missing genotype calls**, at about one per cent, as any chip has.
- **Markers on ten chromosomes** with irregular spacing, so subsetting by genomic region is meaningful.

Because the data are simulated, the true breeding values are known and are kept in `colData`, which makes the object useful for checking that a prediction method recovers what it should.

Source

Simulated by `data-raw/make_demo.R` in the package sources.

See Also

`simulateBreeding()` to generate your own, `Hmatrix()` for the analysis this data set is shaped for.

Examples

```
data(demoBreeding)
demoBreeding

# 60 of the 180 individuals in the pedigree were never genotyped
nrow(pedigree(demoBreeding)) - ncol(demoBreeding)

# the single-step matrix covers all of them
H <- Hmatrix(demoBreeding)
dim(H)
```

Dmatrix

Dominance genomic relationship matrix

Description

Builds the dominance relationship matrix from marker dosages using the genotypic parameterisation of Vitezica, Varona and Legarra (2013). Heterozygotes and the two homozygotes are coded so that the resulting matrix captures dominance deviations, orthogonal to the additive relationships returned by `Gmatrix()`.

Usage

```
Dmatrix(x, ...)

## S4 method for signature 'BreedingExperiment'
Dmatrix(x, min_maf = 0, impute = TRUE, ...)
```

Arguments

x	A BreedingExperiment object.
...	Unused.
min_maf	Markers with a minor allele frequency below this are dropped.
impute	Replace missing dosages by twice the allele frequency before computing. If FALSE, missing values raise an error.

Details

Fitting additive and dominance effects together lets a model separate the part of the genetic variance that is inherited predictably from the part that is not, which matters for crossbreeding and for predicting the performance of specific crosses.

Value

A symmetric numeric matrix over the genotyped individuals.

References

Vitezica, Z. G., Varona, L. & Legarra, A. (2013) "On the additive and dominant variance and covariance of individuals within the genomic selection scope." *Genetics* 195, 1223-1230. doi:[10.1534/genetics.113.155176](#)

See Also

[Gmatrix\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 20, n_marker = 200)
D <- Dmatrix(be)
round(D[1:4, 1:4], 3)
```

filterMarkers

Filter markers on quality

Description

Removes markers that are uninformative or poorly genotyped: those below a minor allele frequency threshold, those called in too few individuals, and optionally those that are monomorphic.

Usage

```
filterMarkers(x, ...)

## S4 method for signature 'BreedingExperiment'
filterMarkers(
  x,
  min_maf = 0.01,
  min_call_rate = 0.9,
  drop_monomorphic = TRUE,
  ...
)
```

Arguments

x	A BreedingExperiment object.
...	Unused.
min_maf	Minimum minor allele frequency to keep.
min_call_rate	Minimum proportion of individuals with a call.
drop_monomorphic	Drop markers with no variation.

Value

A [BreedingExperiment](#) object with fewer rows.

See Also

[imputeMarkers\(\)](#), [alleleFrequency\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 30, n_marker = 100, missing = 0.05)
nrow(be)
nrow(filterMarkers(be, min_maf = 0.05, min_call_rate = 0.9))
```

genotyped

Which individuals are genotyped

Description

Returns the identifiers that appear in the genotype matrix. In a single-step analysis the pedigree usually names more individuals than this, and the difference is what [Hmatrix\(\)](#) reconciles.

Usage

```
genotyped(x, ...)

## S4 method for signature 'BreedingExperiment'
genotyped(x, ...)
```

Arguments

x A [BreedingExperiment](#) object.

... Unused.

Value

A character vector of individual identifiers.

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 8, n_marker = 20)
genotyped(be)
```

genotypes

Genotype dosages

Description

Gets or sets the matrix of allele dosages, a shorthand for the genotype assay. Markers are rows and individuals are columns.

Usage

```
genotypes(x, ...)

genotypes(x) <- value

## S4 method for signature 'BreedingExperiment'
genotypes(x, ...)

## S4 replacement method for signature 'BreedingExperiment'
genotypes(x) <- value
```

Arguments

x A [BreedingExperiment](#) object.

... Unused.

value A numeric matrix of dosages coded 0, 1 or 2.

Value

genotypes() returns a numeric matrix; the replacement form returns the updated object.

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 6, n_marker = 10)
genotypes(be)[1:5, 1:4]
```

Gmatrix	<i>Genomic relationship matrix</i>
---------	------------------------------------

Description

Builds the additive genomic relationship matrix G from marker dosages using VanRaden’s first method: with W the marker matrix centred by twice the allele frequency, $G = WW' / 2 \sum p(1 - p)$.

Usage

```
Gmatrix(x, ...)

## S4 method for signature 'BreedingExperiment'
Gmatrix(x, min_maf = 0, impute = TRUE, ...)
```

Arguments

x	A BreedingExperiment object.
...	Unused.
min_maf	Markers with a minor allele frequency below this are dropped.
impute	Replace missing dosages by twice the allele frequency before computing. If FALSE, missing values raise an error.

Details

G plays the same role as the pedigree matrix A but is estimated from markers, so it captures Mendelian sampling that a pedigree cannot: full sibs have an expected pedigree relationship of exactly 0.5, whereas their genomic relationship varies around it.

Value

A symmetric numeric matrix over the genotyped individuals.

References

VanRaden, P. M. (2008) "Efficient methods to compute genomic predictions." Journal of Dairy Science 91, 4414-4423. doi:[10.3168/jds.20070980](#)

See Also

[Amatrix\(\)](#), [Dmatrix\(\)](#), [Hmatrix\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 20, n_marker = 200)
G <- Gmatrix(be)
round(G[1:4, 1:4], 3)
```

Hmatrix

Single-step relationship matrix

Description

Builds the matrix H that combines pedigree and genomic information, so that genotyped and ungenotyped individuals can be analysed in a single model. This is the relationship structure behind single-step genomic prediction (ssGBLUP).

Usage

```
Hmatrix(x, ...)

## S4 method for signature 'BreedingExperiment'
Hmatrix(x, blend = 0.05, tune = TRUE, min_maf = 0, impute = TRUE, ...)
```

Arguments

<code>x</code>	A BreedingExperiment object with both a pedigree and genotypes.
<code>...</code>	Unused.
<code>blend</code>	Weight w given to A_{22} when blending, between 0 and 1. The default, 0.05, is a common choice.
<code>tune</code>	Rescale G to the scale of A_{22} before blending.
<code>min_maf, impute</code>	Passed to Gmatrix() .

Details

Real breeding programmes genotype only part of the population. Using only G throws away every ungenotyped relative; using only A throws away the markers. H keeps both: relationships among genotyped individuals come from the markers, relationships among ungenotyped individuals are those from the pedigree corrected by what the markers reveal about their genotyped relatives, and the two groups are connected through the pedigree.

Writing 1 for ungenotyped and 2 for genotyped individuals,

$$H_{22} = G^*$$

$$H_{12} = A_{12}A_{22}^{-1}G^*$$

$$H_{11} = A_{11} + A_{12}A_{22}^{-1}(G^* - A_{22})A_{22}^{-1}A_{21}$$

Because G and A_{22} are estimated on different scales, G is first made compatible with A_{22} in two standard steps. *Tuning* rescales G so that its mean diagonal and mean off-diagonal match those of A_{22} . *Blending* then mixes in a small share of A_{22} , $G^* = (1 - w)G + wA_{22}$, which also makes the result invertible when markers are fewer than individuals.

Value

A symmetric numeric matrix over every individual in the pedigree, ungenotyped individuals first, with an attribute genotyped giving the identifiers of the genotyped group.

References

- Legarra, A., Aguilar, I. & Misztal, I. (2009) "A relationship matrix including full pedigree and genomic information." *Journal of Dairy Science* 92, 4656-4663. doi:[10.3168/jds.20092061](https://doi.org/10.3168/jds.20092061)
- Christensen, O. F. & Lund, M. S. (2010) "Genomic prediction when some animals are not genotyped." *Genetics Selection Evolution* 42, 2. doi:[10.1186/12979686422](https://doi.org/10.1186/12979686422)
- Aguilar, I., Misztal, I., Johnson, D. L., Legarra, A., Tsuruta, S. & Lawlor, T. J. (2010) "Hot topic: A unified approach to utilize phenotypic, full pedigree, and genomic information for genetic evaluation of Holstein final score." *Journal of Dairy Science* 93, 743-752. doi:[10.3168/jds.20092730](https://doi.org/10.3168/jds.20092730)

See Also

`Amatrix()`, `Gmatrix()`

Examples

```
# only some of the individuals are genotyped
set.seed(1)
be <- simulateBreeding(n_ind = 30, n_marker = 300, genotyped = 20)
H <- Hmatrix(be)
dim(H)
attr(H, "genotyped")
```

imputeMarkers

Impute missing genotypes

Description

Fills missing dosages with twice the allele frequency of the marker, the simple mean imputation that relationship matrices assume. It is adequate for the low rates of missingness left after quality control; for anything more, impute with dedicated software before building the object.

Usage

```
imputeMarkers(x, ...)

## S4 method for signature 'BreedingExperiment'
imputeMarkers(x, ...)
```

Arguments

x A [BreedingExperiment](#) object.
 ... Unused.

Value

A [BreedingExperiment](#) object with no missing genotypes.

See Also

[filterMarkers\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 20, n_marker = 50, missing = 0.1)
anyNA(genotypes(be))
anyNA(genotypes(imputeMarkers(be)))
```

inbreeding	<i>Inbreeding coefficients</i>
------------	--------------------------------

Description

Returns the inbreeding coefficient of each individual, obtained from the diagonal of the pedigree relationship matrix as $F_i = A_{ii} - 1$.

Usage

```
inbreeding(x, ...)

## S4 method for signature 'BreedingExperiment'
inbreeding(x, ...)
```

Arguments

x A [BreedingExperiment](#) object.
 ... Unused.

Value

A named numeric vector of inbreeding coefficients.

See Also

[Amatrix\(\)](#)

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 12, n_marker = 30)
round(inbreeding(be), 4)
```

pedigree

Pedigree of a breeding experiment

Description

Gets or sets the pedigree, a table with columns id, sire and dam. The pedigree may include ancestors that were never genotyped; those individuals are what allow [Hmatrix\(\)](#) to combine pedigree and genomic information.

Usage

```
pedigree(x, ...)

pedigree(x) <- value

## S4 method for signature 'BreedingExperiment'
pedigree(x, ...)

## S4 replacement method for signature 'BreedingExperiment'
pedigree(x) <- value
```

Arguments

x	A BreedingExperiment object.
...	Unused.
value	A data frame with columns id, sire and dam.

Value

pedigree() returns a [S4Vectors::DataFrame](#); the replacement form returns the updated object.

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 8, n_marker = 20)
head(pedigree(be))
```

simulateBreeding

*Simulate a small breeding population***Description**

Generates a pedigreed population with markers and a heritable trait, for examples, tests and teaching. Founders are unrelated; later generations are produced by mating sampled parents, and each offspring inherits one allele from each parent at every marker, so the simulated genotypes and the pedigree agree with each other.

Usage

```
simulateBreeding(
  n_ind = 50,
  n_marker = 200,
  n_founder = 10,
  genotyped = NULL,
  n_qtl = 20,
  h2 = 0.4,
  missing = 0
)
```

Arguments

n_ind	Number of individuals in total.
n_marker	Number of markers.
n_founder	Number of unrelated founders.
genotyped	Number of individuals with genotypes, taken from the most recent ones. Defaults to all of them.
n_qtl	Number of markers with an effect on the trait.
h2	Narrow-sense heritability of the simulated trait.
missing	Proportion of genotype calls set to NA.

Details

Set genotyped below n_ind to leave some individuals ungenotyped, which is the situation [Hmatrix\(\)](#) is designed for.

The function draws random numbers but does not set the seed itself. Call [set.seed\(\)](#) beforehand if you need the same population twice.

Value

A [BreedingExperiment](#) object with a phenotype column and the true breeding value in colData.

Examples

```
set.seed(1)
be <- simulateBreeding(n_ind = 25, n_marker = 100)
be
head(as.data.frame(colData(be)))
```

sortPedigree*Order a pedigree so that parents precede their offspring*

Description

The tabular method for building a relationship matrix requires that every individual appears after both of its parents. This performs that topological sort and reports a cycle if the pedigree contains one.

Usage

```
sortPedigree(ped)
```

Arguments

ped A data frame or [S4Vectors::DataFrame](#) with columns id, sire and dam.

Value

A data frame with the same columns, reordered.

Examples

```
ped <- data.frame(id = c("c", "a", "b"),
                  sire = c("a", NA, NA),
                  dam = c("b", NA, NA))
sortPedigree(ped)
```

Index

* datasets

demoBreeding, 7

alleleFrequency, 2
alleleFrequency(), 10
alleleFrequency, BreedingExperiment-method
(alleleFrequency), 2

Amatrix, 3
Amatrix(), 5, 7, 13, 14, 16
Amatrix, BreedingExperiment-method
(Amatrix), 3

BreedingExperiment, 3, 4, 4–7, 9–13, 15–17
BreedingExperiment(), 6, 7
BreedingExperiment-class, 6

demoBreeding, 7
Dmatrix, 8
Dmatrix(), 13
Dmatrix, BreedingExperiment-method
(Dmatrix), 8

filterMarkers, 9
filterMarkers(), 15
filterMarkers, BreedingExperiment-method
(filterMarkers), 9

GenomicRanges::GRanges, 5, 6
genotyped, 10
genotyped, BreedingExperiment-method
(genotyped), 10
genotypes, 11
genotypes, BreedingExperiment-method
(genotypes), 11
genotypes<- (genotypes), 11
genotypes<- , BreedingExperiment-method
(genotypes), 11

Gmatrix, 12
Gmatrix(), 4, 5, 7–9, 13, 14
Gmatrix, BreedingExperiment-method
(Gmatrix), 12

Hmatrix, 13
Hmatrix(), 4, 5, 7, 8, 10, 13, 16, 17
Hmatrix, BreedingExperiment-method
(Hmatrix), 13

imputeMarkers, 14
imputeMarkers(), 10
imputeMarkers, BreedingExperiment-method
(imputeMarkers), 14

inbreeding, 15
inbreeding(), 4
inbreeding, BreedingExperiment-method
(inbreeding), 15

pedigree, 16
pedigree, BreedingExperiment-method
(pedigree), 16
pedigree<- (pedigree), 16
pedigree<- , BreedingExperiment-method
(pedigree), 16

S4Vectors::DataFrame, 3, 5, 6, 16, 18
set.seed(), 17
show, BreedingExperiment-method
(BreedingExperiment-class), 6
simulateBreeding, 17
simulateBreeding(), 5, 8
sortPedigree, 18
SummarizedExperiment::RangedSummarizedExperiment,
6
SummarizedExperiment::SummarizedExperiment(),
5