

Package: EMMREML (via r-universe)

September 30, 2026

Type Package

Version 5.2

Date 2026-09-29

Title Fitting Mixed Models with Known Covariance Structures and GWAS

Author Deniz Akdemir, Okeke Uche Godfrey

Maintainer Deniz Akdemir <deniz.akdemir.work@gmail.com>

Description An improved mixed model solver that handles a variety of covariance structures. The main functions are 'emmreml', 'emmremlMultiKernel', and 'emmremlMultivariate'. The 'emmreml' function solves mixed models with known covariance structure using the 'EMMA' algorithm. 'emmremlMultiKernel' allows for fitting models with multiple random components with known covariance structures. The 'emmremlMultivariate' function solves multivariate Gaussian mixed models with known covariance structures using the 'ECM' algorithm. The package now offers both the original matrix interface (fully backward compatible with v3.1) and a modern formula interface. The 'emmremlMultivariate' function restores 'Gpred' and all test outputs from v3.1 while fixing ECM convergence. The new 'gwas' function provides genome-wide association studies with mixed models, including an accelerated algorithm and visualization options.

Depends R (>= 3.5.0), Matrix, stats

Imports methods

Suggests knitr, rmarkdown, testthat, ggplot2, MASS, data.table, viridis, impute

License GPL-2

NeedsCompilation no

Encoding UTF-8

RoxygenNote 7.3.3

Additional_repositories <https://bioconductor.org/packages/release/bioc>

VignetteBuilder knitr

URL <https://github.com/dakdemir/EMMREML>

BugReports <https://github.com/dakdemir/EMMREML/issues>

Repository <https://denizakdemir.r-universe.dev>

Date/Publication 2026-09-30 08:39:51 UTC

RemoteUrl <https://github.com/denizakdemir/emmreml>

RemoteRef HEAD

RemoteSha 876aa2b4d4e21c62d5491c38faf21858400c7ec4

Contents

calculate_allele_frequencies	3
calculate_kinship	3
compare_kinship	5
convert_marker_coding	5
demo_gwas	6
demo_K_optimization	7
demo_kinship	7
demo_multivariate	8
emmreml	8
EMMREML	10
emmremlMultiKernel	11
emmremlMultivariate	13
emmremlMultivariate_core	15
filter_markers	17
generate_example_data	18
gwas	19
identify_marker_coding	21
impute_markers	22
kinship	23
marker_statistics	25
plot.emmreml_gwas	26
plot_marker_stats	26
predict.emmreml	27
print.emmreml	28
print.emmreml_gwas	28
print.emmreml_multi	29
print.summary.emmreml	29
print.summary.emmreml_gwas	30
print.summary.emmreml_multi	30
subset_markers_by_maf	31
summarize_marker_data	31
summary.emmreml	32
summary.emmreml_gwas	33
summary.emmreml_multi	33

Usage

```
calculate_kinship(
  M,
  method = "vanraden1",
  coding = NULL,
  scale = TRUE,
  na.rm = TRUE,
  na.method = "mean",
  return_scaled_M = FALSE
)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
method	Character, method for calculating kinship: - "vanraden1": First method from VanRaden (2008) - "vanraden2": Second method from VanRaden (2008) - "astle": Astle & Balding method - "ibs": Identity by state (IBS) - "ibd": Identity by descent (IBD) - "centered": Simple centered cross-product - "correlation": Correlation-based
coding	Character, the marker coding type ("01", "012", "-101", or "custom")
scale	Logical, whether to scale the kinship matrix (default: TRUE)
na.rm	Logical, whether to remove missing values (default: TRUE)
na.method	Character, how to handle missing data ("omit", "mean", "zero")
return_scaled_M	Logical, whether to also return the scaled marker matrix

Value

A kinship matrix or a list with the kinship matrix and scaled markers

References

VanRaden, P. M. (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, 91(11), 4414-4423.

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2), 100*1000, replace=TRUE), 100, 1000)
K <- calculate_kinship(M_012, method="vanraden1", coding="012")

## End(Not run)
```

compare_kinship	<i>Compare different kinship matrices</i>
-----------------	---

Description

Compare different kinship matrices

Usage

```
compare_kinship(K_list, names = NULL, plot = TRUE)
```

Arguments

K_list	List of kinship matrices
names	Character vector of names for the matrices
plot	Logical, whether to create a plot (default: TRUE)

Value

If plot=TRUE, a ggplot2 object, otherwise correlation matrix

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2), 100*1000, replace=TRUE), 100, 1000)
K1 <- calculate_kinship(M_012, method="vanraden1", coding="012")
K2 <- calculate_kinship(M_012, method="vanraden2", coding="012")
K3 <- calculate_kinship(M_012, method="correlation", coding="012")
compare_kinship(list(K1, K2, K3), names=c("VanRaden1", "VanRaden2", "Correlation"))

## End(Not run)
```

convert_marker_coding	<i>Convert marker coding between different types</i>
-----------------------	--

Description

Convert marker coding between different types

Usage

```
convert_marker_coding(M, from, to)
```

Arguments

<code>M</code>	A marker matrix with samples in rows and markers in columns
<code>from</code>	Character, the current coding type ("01", "012", "-101", or "custom")
<code>to</code>	Character, the desired coding type ("01", "012", "-101")

Value

A marker matrix with the converted coding

Examples

```
## Not run:
# Convert 0,1,2 coding to -1,0,1
M_012 <- matrix(sample(c(0,1,2), 100*10, replace=TRUE), 100, 10)
M_n101 <- convert_marker_coding(M_012, from="012", to="-101")

## End(Not run)
```

demo_gwas

Example of using GWAS functionality

Description

This function demonstrates how to use the GWAS functionality in EMMREML.

Usage

```
demo_gwas()
```

Value

GWAS result object

Examples

```
## Not run:
gwas_result <- demo_gwas()
plot(gwas_result)

## End(Not run)
```

demo_K_optimization	<i>Example of using EMMREML with K optimization</i>
---------------------	---

Description

This function demonstrates how EMMREML can automatically detect when K is an identity matrix and optimize calculations accordingly.

Usage

```
demo_K_optimization()
```

Value

Nothing, runs examples and prints timing results

Examples

```
## Not run:  
demo_K_optimization()  
  
## End(Not run)
```

demo_kinship	<i>Demonstrate the kinship matrix calculation functionality</i>
--------------	---

Description

This function demonstrates how to use the kinship module to process marker data, calculate different types of kinship matrices, and visualize the results.

Usage

```
demo_kinship()
```

Value

A list containing the example marker data, processed data, and kinship matrices

Examples

```
## Not run:  
demo_results <- demo_kinship()  
str(demo_results)  
  
## End(Not run)
```

demo_multivariate	<i>Example of using the multivariate EMMREML</i>
-------------------	--

Description

This function demonstrates how to use the multivariate EMMREML functionality.

Usage

```
demo_multivariate()
```

Value

Multivariate EMMREML result object

Examples

```
## Not run:
mv_result <- demo_multivariate()
summary(mv_result)

## End(Not run)
```

emmreml	<i>Solver for Gaussian mixed model with known covariance structure.</i>
---------	---

Description

This function estimates the parameters of the model

$$y = X\beta + Zu + e$$

where y is the n vector of response variable, X is a $n \times q$ known design matrix of fixed effects, Z is a $n \times l$ known design matrix of random effects, β is $q \times 1$ vector of fixed effects coefficients and u and e are independent variables with $N_l(0, \sigma_u^2 K)$ and $N_n(0, \sigma_e^2 I_n)$ correspondingly. It also produces the BLUPs and some other useful statistics like large sample estimates of variances and PEV.

Usage

```
emmreml(y, X, Z, K, varbetahat=FALSE, varuhat=FALSE, PEVuhat=FALSE, test=FALSE)
```

Arguments

<code>y</code>	$nx1$ numeric vector
<code>X</code>	nxq matrix
<code>Z</code>	nxl matrix
<code>K</code>	lxl matrix of known relationships
<code>varbetahat</code>	TRUE or FALSE
<code>varuhat</code>	TRUE or FALSE
<code>PEVuhat</code>	TRUE or FALSE
<code>test</code>	TRUE or FALSE

Value

<code>Vu</code>	Estimate of σ_u^2
<code>Ve</code>	Estimate of σ_e^2
<code>betahat</code>	BLUES for β
<code>uhat</code>	BLUPs for u
<code>Xsqtestbeta</code>	χ^2 test statistics for testing whether the fixed effect coefficients are equal to zero.
<code>pvalbeta</code>	pvalues obtained from large sample theory for the fixed effects. We report the pvalues adjusted by the "padjust" function for all fixed effect coefficients.
<code>Xsqtestu</code>	χ^2 test statistic values for testing whether the BLUPs are equal to zero.
<code>pvalu</code>	pvalues obtained from large sample theory for the BLUPs. We report the pvalues adjusted by the "padjust" function.
<code>varuhat</code>	Large sample variance for the BLUPs.
<code>varbetahat</code>	Large sample variance for the β 's.
<code>PEVuhat</code>	Prediction error variance estimates for the BLUPs.
<code>loglik</code>	loglikelihood for the model.

Examples

```

n=200
M1<-matrix(rnorm(n*300), nrow=n)
K1<-cov(t(M1))
K1=K1/mean(diag(K1))

covY<-2*K1+1*diag(n)

Y<-10+crossprod(chol(covY),rnorm(n))

#training set
Trainset<-sample(1:n, 150)

funout<-emmreml(y=Y[Trainset], X=matrix(rep(1, n)[Trainset], ncol=1),
  Z=diag(n)[Trainset,], K=K1)

```

```
cor(Y[-Trainset], funout$uhat[-Trainset])
```

EMMREML

Fitting mixed models with known covariance structures.

Description

The main functions are `emmreml`, and `emmremlMultiKernel`. `emmreml` solves a mixed model with known covariance structure using the EMMA algorithm in Kang et.al. (2008). `emmremlMultiKernel` is a wrapper for `emmreml` to handle multiple random components with known covariance structures. The function `emmremlMultivariate` solves a multivariate gaussian mixed model with known covariance structure using the ECM algorithm in Zhou and Stephens (2012).

Details

Package: EMMREML
Type: Package
Version: 3.1
Date: 2015-07-20
License: GPL-2

Author(s)

Deniz Akdemir, Okeke Uche Godfrey

Maintainer: Deniz Akdemir <deniz.akdemir.work@gmail.com>

References

Efficient control of population structure in model organism association mapping. Kang, Hyun Min and Zaitlen, Noah A and Wade, Claire M and Kirby, Andrew and Heckerman, David and Daly, Mark J and Eskin, Eleazar. Genetics, 2008.

Genome-wide efficient mixed-model analysis for association studies. Zhou, Xiang and Stephens, Matthew. Nature genetics, 2012.

emmremlMultiKernel	<i>Function to fit Gaussian mixed model with multiple mixed effects with known covariances.</i>
--------------------	---

Description

This function is a wrapper for the emmreml to fit Gaussian mixed model with multiple mixed effects with known covariances. The model fitted is $y = X\beta + Z_1u_1 + Z_2u_2 + \dots Z_ku_k + e$ where y is the n vector of response variable, X is a nxq known design matrix of fixed effects, Z_j is a nxl_j known design matrix of random effects for $j = 1, 2, \dots, k$, β is $nx1$ vector of fixed effects coefficients and $U = (u_1^t, u_2^t, \dots, u_k^t)^t$ and e are independent variables with $N_L(0, \text{blockdiag}(\sigma_{u_1}^2 K_1, \sigma_{u_2}^2 K_2, \dots, \sigma_{u_k}^2 K_k))$ and $N_n(0, \sigma_e^2 I_n)$ correspondingly. The function produces the BLUPs for the $L = l_1 + l_2 + \dots + l_k$ dimensional random effect U . The variance parameters for random effects are estimated as $(\hat{w}_1, \hat{w}_2, \dots, \hat{w}_k) * \hat{\sigma}_u^2$ where $w = (w_1, w_2, \dots, w_k)$ are the kernel weights. The function also provides some useful statistics like large sample estimates of variances and PEV.

Usage

```
emmremlMultiKernel(y, X, Zlist, Klist,
varbetahat=FALSE, varuhat=FALSE, PEVuhat=FALSE, test=FALSE)
```

Arguments

<code>y</code>	$nx1$ numeric vector
<code>X</code>	nxq matrix
<code>Zlist</code>	list of random effects design matrices of dimensions $nxl_1, \dots, nxl_k$
<code>Klist</code>	list of known relationship matrices of dimensions $l_1xl_1, \dots, l_kxl_k$
<code>varbetahat</code>	TRUE or FALSE
<code>varuhat</code>	TRUE or FALSE
<code>PEVuhat</code>	TRUE or FALSE
<code>test</code>	TRUE or FALSE

Value

<code>Vu</code>	Estimate of σ_u^2
<code>Ve</code>	Estimate of σ_e^2
<code>betahat</code>	BLUES for β
<code>uhat</code>	BLUPs for u
<code>weights</code>	Estimates of kernel weights
<code>Xsqtestbeta</code>	A χ^2 test statistic based for testing whether the fixed effect coefficients are equal to zero.

pvalbeta	pvalues obtained from large sample theory for the fixed effects. We report the pvalues adjusted by the "padjust" function for all fixed effect coefficients.
Xsqtestu	A χ^2 test statistic based for testing whether the BLUPs are equal to zero.
pvalu	pvalues obtained from large sample theory for the BLUPs. We report the pvalues adjusted by the "padjust" function.
varuhat	Large sample variance for the BLUPs.
varbetahat	Large sample variance for the β 's.
PEVuhat	Prediction error variance estimates for the BLUPs.
loglik	loglikelihood for the model.

Examples

```
####example
#Data from Gaussian process with three
#(total four, including residuals) independent
#sources of variation

n=80
M1<-matrix(rnorm(n*10), nrow=n)

M2<-matrix(rnorm(n*20), nrow=n)

M3<-matrix(rnorm(n*5), nrow=n)

#Relationship matrices
K1<-cov(t(M1))
K2<-cov(t(M2))
K3<-cov(t(M3))
K1=K1/mean(diag(K1))
K2=K2/mean(diag(K2))
K3=K3/mean(diag(K3))

#Generate data
covY<-2*(.2*K1+.7*K2+.1*K3)+diag(n)

Y<-10+crossprod(chol(covY),rnorm(n))

#training set
Trainsamp<-sample(1:80, 60)

funout<-emmremlMultiKernel(y=Y[Trainsamp], X=matrix(rep(1, n)[Trainsamp], ncol=1),
Zlist=list(diag(n)[Trainsamp,], diag(n)[Trainsamp,], diag(n)[Trainsamp,]),
Klist=list(K1,K2, K3),
varbetahat=FALSE,varuhat=FALSE, PEVuhat=FALSE, test=FALSE)
#weights
funout$weights
#Correlation of predictions with true values in test set
uhatmat<-matrix(funout$uhat, ncol=3)
```

```

uhatvec<-rowSums(uhatmat)

cor(Y[-Trainsamp], uhatvec[-Trainsamp])

```

emmremlMultivariate	<i>Fit a Multivariate Gaussian Mixed Model with Known Covariance Structure</i>
---------------------	--

Description

Fits a multivariate Gaussian mixed model of the form $Y = XB + ZG + E$, where $\text{vec}(G) \sim N(0, K \otimes V_g)$ and $\text{vec}(E) \sim N(0, I_n \otimes V_e)$, using the ECM algorithm.

Two calling conventions are supported:

Matrix interface (v3.1 compatible):

```

emmremlMultivariate(Y, X, Z, K, varBhat, varGhat, PEVGhat,
                    test, tolpar, tolparinv)

```

where Y and X are (traits x individuals) matrices and Z is an (individuals x levels) incidence matrix.

Formula interface (v5.x):

```

emmremlMultivariate(formula, random, data, K, verbose,
                    tolParConv, tolParInv, maxIter)

```

Usage

```

emmremlMultivariate(Y_or_formula, X_or_random = NULL, Z_or_data, K,
                    varBhat = FALSE, varGhat = FALSE, PEVGhat = FALSE,
                    test = FALSE, tolpar = 1e-6, tolparinv = 1e-6,
                    verbose = TRUE, tolParConv = 1e-6, tolParInv = 1e-6,
                    maxIter = 100, ...)

```

Arguments

Y_or_formula	Either a (d x n) response matrix (traits in rows, individuals in columns) for the matrix interface, or a formula such as <code>cbind(y1, y2) ~ x</code> for the formula interface.
X_or_random	Either a (m x n) fixed-effects design matrix for the matrix interface, or an optional random-effects formula (or NULL) for the formula interface.
Z_or_data	Either a (n x N) incidence matrix for the matrix interface, or a data frame for the formula interface.
K	(N x N) covariance/kernel matrix for random effects.
varBhat	Logical, whether to return variance of B estimates. Default FALSE.
varGhat	Logical, whether to return variance of G predictions. Default FALSE.
PEVGhat	Logical, whether to return prediction error variance of G. Default FALSE.

test	Logical, whether to perform chi-square significance tests. Default FALSE.
tolpar	Convergence tolerance for variance components (matrix interface). Default 1e-6.
tolparinv	Regularisation added to matrices before inversion (matrix interface). Default 1e-6.
verbose	Logical, whether to print progress (formula interface). Default TRUE.
tolParConv	Convergence tolerance (formula interface). Default 1e-6.
tolParInv	Regularisation tolerance (formula interface). Default 1e-6.
maxIter	Maximum number of ECM iterations. Default 100.
...	Ignored (for forward compatibility).

Details

The ECM algorithm projects Y and X into the eigenvector space of ZKZ' , then iterates per-eigenvalue E-steps and M-steps to update the genetic (V_g) and residual (V_e) covariance matrices. This is the same algorithm as in v3.1, with the addition of a `maxIter` guard and a robust convergence check. See [emmremlMultivariate_core](#) for the underlying implementation.

Value

A list of class `emmreml_mv` containing:

Bhat	Fixed effects estimates in eigenvector space (m x n matrix).
Vg	Estimated genetic covariance matrix (d x d).
Ve	Estimated residual covariance matrix (d x d).
Gpred	BLUPs for G (d x N matrix).
iterations	Number of ECM iterations run.
convergence	Logical: did the algorithm converge?
varBhat	(if requested) Diagonal of variance matrix for <code>vec(B)</code> .
varGhat	(if requested) Diagonal of variance matrix for <code>vec(G)</code> .
PEVGhat	(if requested) Diagonal of prediction error variance for <code>vec(G)</code> .
XsqtestB	(if <code>test=TRUE</code>) Chi-square statistics for fixed effects.
pvalB	(if <code>test=TRUE</code>) Adjusted p-values for fixed effects.
XsqtestG	(if <code>test=TRUE</code>) Chi-square statistics for G predictions.
pvalG	(if <code>test=TRUE</code>) Adjusted p-values for G predictions.

See Also

[emmremlMultivariate_core](#), [emmreml](#), [emmremlMultiKernel](#)

Examples

```

set.seed(123)
n <- 15 # individuals
d <- 2  # traits

# --- matrix interface (v3.1 style) ---
Y <- matrix(rnorm(d * n), d, n) # traits x individuals
X <- rbind(rep(1, n), rnorm(n)) # fixed effects x individuals
Z <- diag(n)
K_raw <- matrix(rnorm(n * n), n, n)
K <- tcrossprod(K_raw)
diag(K) <- diag(K) + 1e-4
K <- K / mean(diag(K))

result <- emmremlMultivariate(Y, X, Z, K)
print(result$Vg)
print(result$Ve)
dim(result$Gpred) # d x n

## Not run:
# --- formula interface (v5.x style) ---
df <- data.frame(y1 = rnorm(n), y2 = rnorm(n), x1 = rnorm(n))
result2 <- emmremlMultivariate(
  formula = cbind(y1, y2) ~ x1,
  data = df,
  K = K,
  verbose = FALSE
)
print(result2$Vg)

## End(Not run)

```

emmremlMultivariate_core

Fit a Multivariate Gaussian Mixed Model with Known Covariance Structure

Description

Fits a multivariate Gaussian mixed model of the form $Y = XB + ZG + E$, where $\text{vec}(G) \sim N(0, K \otimes V_g)$ and $\text{vec}(E) \sim N(0, I_n \otimes V_e)$, using the ECM algorithm.

This is the primary (matrix-based) interface, preserved for backward compatibility with v3.1. The formula-based wrapper `emmremlMultivariate` calls this function internally.

Usage

```

emmremlMultivariate_core(
  Y,
  X,

```

```

Z,
K,
varBhat = FALSE,
varGhat = FALSE,
PEVGhat = FALSE,
test = FALSE,
tolpar = 1e-06,
tolparinv = 1e-06,
maxIter = 100
)

```

Arguments

Y	Response matrix (d x n): d traits, n individuals. NOTE: traits in rows, individuals in columns — same convention as v3.1.
X	Fixed effects design matrix (m x n): m effects, n individuals.
Z	Random effects incidence matrix (n x N): n individuals, N levels.
K	Covariance/kernel matrix (N x N) for random effects.
varBhat	Logical, whether to return variance of B estimates.
varGhat	Logical, whether to return variance of G predictions.
PEVGhat	Logical, whether to return prediction error variance of G.
test	Logical, whether to perform chi-square significance tests.
tolpar	Tolerance for convergence of variance components.
tolparinv	Regularisation added to matrices before inversion.
maxIter	Maximum number of ECM iterations.

Value

A list (class `emmreml_mv`) containing:

Bhat	Fixed effects estimates (m x n eigenvector-space; same as v3.1 Bt)
Vg	Estimated genetic covariance matrix (d x d)
Ve	Estimated residual covariance matrix (d x d)
Gpred	BLUPs for G (d x N matrix, same as v3.1)
iterations	Number of ECM iterations run
convergence	Logical: did the algorithm converge?
varBhat	(optional) variance of <code>vec(B)</code>
varGhat	(optional) variance of <code>vec(G)</code>
PEVGhat	(optional) prediction error variance of <code>vec(G)</code>
XsqtestB, pvalB, XsqtestG, pvalG	(optional) test statistics

filter_markers	<i>Filter markers based on various criteria</i>
----------------	---

Description

Filter markers based on various criteria

Usage

```
filter_markers(  
  M,  
  min_maf = 0.01,  
  max_missing = 0.2,  
  min_samples = NULL,  
  coding = NULL  
)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
min_maf	Numeric, minimum minor allele frequency (default: 0.01)
max_missing	Numeric, maximum proportion of missing values (default: 0.2)
min_samples	Integer, minimum number of samples with non-missing values
coding	Character, the marker coding type ("01", "012", "-101", or "custom")

Value

A filtered marker matrix

Examples

```
## Not run:  
M_012 <- matrix(sample(c(0,1,2, NA), 100*1000, replace=TRUE), 100, 1000)  
M_filtered <- filter_markers(M_012, min_maf=0.05, max_missing=0.1)  
  
## End(Not run)
```

generate_example_data *Generate example data for EMMREML demonstrations*

Description

Generate example data for EMMREML demonstrations

Usage

```
generate_example_data(  
  n = 100,  
  p = 1000,  
  with_identity_K = TRUE,  
  with_gwas = TRUE  
)
```

Arguments

n	Number of samples
p	Number of markers (for GWAS examples)
with_identity_K	Whether to create an identity K matrix example
with_gwas	Whether to generate GWAS example data

Value

A list with example data

Examples

```
## Not run:  
# Basic mixed model  
example_data <- generate_example_data(n = 100)  
  
# Fit model  
mod <- emmreml(y ~ x, random = ~ id, data = example_data$data, K = example_data$K)  
summary(mod)  
  
## End(Not run)
```

Description

Performs genome-wide association analysis using mixed models with variance components estimated by REML. This function can handle both accelerated mode (where variance components are estimated once) and regular mode (where the full model is fitted for each marker).

Usage

```
gwas(
  formula,
  random = NULL,
  data,
  markers,
  K = NULL,
  accelerated = TRUE,
  pc.count = 0,
  method = c("mixed", "lm", "score"),
  marker.select = NULL,
  verbose = TRUE,
  plots = TRUE,
  plot.type = c("manhattan", "qq", "pca", "effect"),
  tolParConv = 1e-06,
  ...
)
```

Arguments

<code>formula</code>	Formula for fixed effects excluding markers, e.g., $y \sim x_1 + x_2$
<code>random</code>	Formula for random effects, e.g., $\sim \text{id}$ (for kinship)
<code>data</code>	A data frame containing phenotypes and fixed effect covariates
<code>markers</code>	Matrix of marker genotypes (samples in rows, markers in columns)
<code>K</code>	Kinship matrix or NULL (if no kinship correction is needed)
<code>accelerated</code>	Logical, whether to use accelerated mode (default: TRUE)
<code>pc.count</code>	Number of principal components to include as covariates
<code>method</code>	Method for analysis: "mixed" (default), "lm", or "score"
<code>marker.select</code>	Optional index vector for testing only a subset of markers
<code>verbose</code>	Logical, whether to print progress information
<code>plots</code>	Logical, whether to generate plots
<code>plot.type</code>	Types of plots to generate: "manhattan", "qq", "pca", "effect"
<code>tolParConv</code>	Tolerance for parameter convergence
<code>...</code>	Additional arguments

Value

An object of class 'emmmreml_gwas' containing:

results	Data frame with marker results (effects, p-values, etc.)
manhattan	Manhattan plot (if plots=TRUE)
qq	QQ plot (if plots=TRUE)
base.model	Base model (if accelerated=TRUE)
method	Method used
call	The matched call

Examples

```
## Not run:
# Simulate data
n <- 200 # number of samples
m <- 1000 # number of markers

# Create some markers
markers <- matrix(sample(c(0, 1, 2), n*m, replace=TRUE), nrow=n, ncol=m)
colnames(markers) <- paste0("SNP", 1:m)

# Create a kinship matrix
K <- tcrossprod(scale(markers)) / m

# Simulate phenotypes with causal variants
causal_idx <- c(100, 300, 700)
betas <- rep(0, m)
betas[causal_idx] <- c(0.5, 0.4, 0.3)

# Create genetic values
g <- markers %*% betas

# Add random effect from kinship
u <- MASS::mvrnorm(1, rep(0, n), 2 * K)

# Add residual error
e <- rnorm(n, 0, 1)

# Create phenotype
y <- 10 + g + u + e

# Prepare data
df <- data.frame(y = y, id = 1:n)

# Run GWAS
gwas_result <- gwas(y ~ 1, random = ~ id, data = df,
                    markers = markers, K = K)

# Plot results
plot(gwas_result)
```

```
## End(Not run)
```

```
identify_marker_coding
```

Kinship Matrix Calculation and Marker Data Processing

Description

Functions for calculating relationship matrices from marker data, marker data preprocessing, filtering, imputation, and visualization. Identify marker data coding type

Usage

```
identify_marker_coding(M, check_samples = 100, tolerance = 1e-08)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
check_samples	Integer, number of markers to sample when checking (default: 100)
tolerance	Numeric, tolerance for comparing values (default: 1e-8)

Value

A list with elements:

coding_type	Character, one of "01", "012", "-101", or "custom"
min_val	Minimum value in the matrix
max_val	Maximum value in the matrix
unique_values	Vector of unique values found
ploidy	Estimated ploidy level based on unique values

Examples

```
## Not run:
# Binary markers (0,1)
M_01 <- matrix(sample(c(0,1), 100*10, replace=TRUE), 100, 10)
identify_marker_coding(M_01)

# SNP markers (0,1,2)
M_012 <- matrix(sample(c(0,1,2), 100*10, replace=TRUE), 100, 10)
identify_marker_coding(M_012)

# Centered markers (-1,0,1)
M_n101 <- matrix(sample(c(-1,0,1), 100*10, replace=TRUE), 100, 10)
identify_marker_coding(M_n101)
```

```
## End(Not run)
```

impute_markers	<i>Impute missing marker data</i>
----------------	-----------------------------------

Description

Impute missing marker data

Usage

```
impute_markers(M, method = "mean", k = 5, coding = NULL)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
method	Character, imputation method: - "mean": Impute with the mean (most common for quantitative data) - "mode": Impute with the most frequent value (most common for discrete data) - "median": Impute with the median value - "knn": K-nearest neighbor imputation - "random": Random imputation from empirical distribution
k	Integer, number of neighbors for KNN imputation (default: 5)
coding	Character, the marker coding type ("01", "012", "-101", or "custom")

Value

A marker matrix with imputed values

Examples

```
## Not run:
M <- matrix(sample(c(0,1,2, NA), 100*10, replace=TRUE), 100, 10)
M_imp <- impute_markers(M, method="mean", coding="012")

## End(Not run)
```

Description

A collection of functions for kinship matrix calculation, marker data preprocessing, filtering, imputation, and visualization. These functions provide comprehensive tools for handling genomic marker data and calculating various types of relationship matrices.

Details

The functions in this module fall into several categories:

Marker Data Inspection:

- `identify_marker_coding` - Identify the coding system used in marker data (0/1, 0/1/2, -1/0/1, etc.)
- `marker_statistics` - Calculate various statistics for marker data (MAF, heterozygosity, etc.)
- `summarize_marker_data` - Create summary table of marker data properties

Marker Data Preprocessing:

- `convert_marker_coding` - Convert marker data between different coding systems
- `filter_markers` - Filter markers based on MAF, missing data, etc.
- `impute_markers` - Impute missing values in marker data
- `subset_markers_by_maf` - Select markers based on MAF distribution

Kinship Matrix Calculation:

- `calculate_allele_frequencies` - Calculate allele frequencies from marker data
- `calculate_kinship` - Calculate kinship matrices using various methods
- `compare_kinship` - Compare different kinship matrices

Visualization:

- `visualize_kinship` - Create heatmap visualizations of kinship matrices
- `plot_marker_stats` - Plot marker statistics (MAF, missing data, etc.)

Author(s)

Deniz Akdemir

References

- VanRaden, P. M. (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, 91(11), 4414-4423.
- Astle, W. & Balding, D. J. (2009). Population structure and cryptic relatedness in genetic association studies. *Statistical Science*, 24(4), 451-471.
- Endelman, J. B. & Jannink, J. L. (2012). Shrinkage estimation of the realized relationship matrix. *G3: Genes, Genomes, Genetics*, 2(11), 1405-1413.

See Also

[emmreml](#) for mixed model analysis using kinship matrices [emmremlMultiKernel](#) for mixed models with multiple kinship matrices [gwas](#) for genome-wide association studies using kinship matrices

Examples

```
## Not run:
# Generate example marker data
set.seed(123)
n_samples <- 100
n_markers <- 1000
M <- matrix(sample(c(0,1,2), n_samples*n_markers, replace=TRUE),
             nrow=n_samples, ncol=n_markers)
colnames(M) <- paste0("SNP", 1:n_markers)
rownames(M) <- paste0("Sample", 1:n_samples)

# Identify marker coding
coding_info <- identify_marker_coding(M)
print(coding_info)

# Calculate marker statistics
stats <- marker_statistics(M, coding="012")
head(stats)

# Filter markers by MAF
M_filtered <- filter_markers(M, min_maf=0.1, coding="012")

# Calculate VanRaden kinship matrix
K_vanraden <- calculate_kinship(M_filtered, method="vanraden1", coding="012")

# Calculate correlation-based kinship matrix
K_corr <- calculate_kinship(M_filtered, method="correlation", coding="012")

# Compare different kinship matrices
comparison <- compare_kinship(
  list(K_vanraden, K_corr),
  names=c("VanRaden", "Correlation")
)
print(comparison)

# Use in mixed model
y <- rnorm(n_samples) # Example phenotypes
X <- cbind(1, rnorm(n_samples)) # Example fixed effects
colnames(X) <- c("intercept", "covariate")

# Fit model with kinship
model <- emmreml(y ~ X, K=K_vanraden)
summary(model)

## End(Not run)
```

marker_statistics	<i>Calculate minor allele frequencies and return marker statistics</i>
-------------------	--

Description

Calculate minor allele frequencies and return marker statistics

Usage

```
marker_statistics(M, coding = NULL, na.rm = TRUE)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
coding	Character, the marker coding type ("01", "012", "-101", or "custom")
na.rm	Logical, whether to remove NA values when calculating frequencies

Value

A data frame with marker statistics:

marker	Marker name or index
freq	Allele frequency
maf	Minor allele frequency
missing_rate	Proportion of missing values
heterozygosity	Heterozygosity rate (for diploid markers)

Examples

```
## Not run:  
M_012 <- matrix(sample(c(0,1,2), 100*10, replace=TRUE), 100, 10)  
stats <- marker_statistics(M_012, coding="012")  
  
## End(Not run)
```

plot.emmreml_gwas	<i>Plot Method for emmreml_gwas Objects</i>
-------------------	---

Description

Plot Method for emmreml_gwas Objects

Usage

```
## S3 method for class 'emmreml_gwas'
plot(x, type = c("manhattan", "qq", "pca", "effect"), ...)
```

Arguments

- x An object of class 'emmreml_gwas'
- type Type of plot to display: "manhattan", "qq", "pca", or "effect"
- ... Additional arguments passed to plotting functions

Value

Invisibly returns the plot object

plot_marker_stats	<i>Plot marker data summary statistics</i>
-------------------	--

Description

Plot marker data summary statistics

Usage

```
plot_marker_stats(M, coding = NULL, plot_type = "both")
```

Arguments

- M A marker matrix with samples in rows and markers in columns
- coding Character, the marker coding type ("01", "012", "-101", or "custom")
- plot_type Character, type of plot to create ("maf", "missing", "both")

Value

A ggplot2 object or list of plots

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2, NA), 100*1000, replace=TRUE, prob=c(0.3, 0.4, 0.2, 0.1)), 100, 1000)
p <- plot_marker_stats(M_012, coding="012", plot_type="both")
print(p$maf)
print(p$missing)

## End(Not run)
```

predict.emmreml	<i>Predict Method for emmreml Objects</i>
-----------------	---

Description

Predict Method for emmreml Objects

Usage

```
## S3 method for class 'emmreml'
predict(object, newdata = NULL, type = "total", se.pred = FALSE, ...)
```

Arguments

object	An object of class 'emmreml'
newdata	Optional data frame for predictions
type	Type of prediction: "fixed" for fixed effects only, "random" for random effects only, or "total" for both
se.pred	Logical, whether to return standard errors
...	Additional arguments

Value

A list with predictions

print.emmreml	<i>Print Method for emmreml Objects</i>
---------------	---

Description

Print Method for emmreml Objects

Usage

```
## S3 method for class 'emmreml'  
print(x, ...)
```

Arguments

x	An object of class 'emmreml'
...	Additional arguments

Value

Invisibly returns the input object

print.emmreml_gwas	<i>Print Method for emmreml_gwas Objects</i>
--------------------	--

Description

Print Method for emmreml_gwas Objects

Usage

```
## S3 method for class 'emmreml_gwas'  
print(x, ...)
```

Arguments

x	An object of class 'emmreml_gwas'
...	Additional arguments

Value

Invisibly returns the input object

print.emmreml_multi *Print Method for emmreml_multi Objects*

Description

Print Method for emmreml_multi Objects

Usage

```
## S3 method for class 'emmreml_multi'  
print(x, ...)
```

Arguments

x	An object of class 'emmreml_multi'
...	Additional arguments

Value

Invisibly returns the input object

print.summary.emmreml *Print Method for summary.emmreml Objects*

Description

Print Method for summary.emmreml Objects

Usage

```
## S3 method for class 'summary.emmreml'  
print(x, ...)
```

Arguments

x	An object of class 'summary.emmreml'
...	Additional arguments

Value

Invisibly returns the input object

```
print.summary.emmreml_gwas
```

Print Method for summary.emmreml_gwas Objects

Description

Print Method for summary.emmreml_gwas Objects

Usage

```
## S3 method for class 'summary.emmreml_gwas'  
print(x, ...)
```

Arguments

x	An object of class 'summary.emmreml_gwas'
...	Additional arguments

Value

Invisibly returns the input object

```
print.summary.emmreml_multi
```

Print Method for summary.emmreml_multi Objects

Description

Print Method for summary.emmreml_multi Objects

Usage

```
## S3 method for class 'summary.emmreml_multi'  
print(x, ...)
```

Arguments

x	An object of class 'summary.emmreml_multi'
...	Additional arguments

Value

Invisibly returns the input object

subset_markers_by_maf *Subset a marker matrix based on MAF buckets*

Description

This function creates a stratified subset of markers based on MAF buckets, which can be useful for testing different marker selection strategies.

Usage

```
subset_markers_by_maf(M, n_markers, breaks = NULL, coding = NULL)
```

Arguments

M	A marker matrix with samples in rows and markers in columns
n_markers	Number of markers to select in total
breaks	Vector of MAF breakpoints for stratification (default: 5 equal buckets)
coding	Character, the marker coding type ("01", "012", "-101", or "custom")

Value

A subset of the marker matrix with selected markers

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2), 100*1000, replace=TRUE), 100, 1000)
M_sub <- subset_markers_by_maf(M_012, n_markers=100, coding="012")

## End(Not run)
```

summarize_marker_data *Create a summary table of marker data properties*

Description

Create a summary table of marker data properties

Usage

```
summarize_marker_data(M, coding = NULL)
```

Arguments

M A marker matrix with samples in rows and markers in columns

coding Character, the marker coding type ("01", "012", "-101", or "custom")

Value

A data.frame with summary statistics

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2, NA), 100*1000, replace=TRUE), 100, 1000)
summary_table <- summarize_marker_data(M_012, coding="012")
print(summary_table)

## End(Not run)
```

summary.emmreml

Summary Method for emmreml Objects

Description

Summary Method for emmreml Objects

Usage

```
## S3 method for class 'emmreml'
summary(object, ...)
```

Arguments

object An object of class 'emmreml'

... Additional arguments

Value

A summary object

summary.emmreml_gwas *Summary Method for emmreml_gwas Objects*

Description

Summary Method for emmreml_gwas Objects

Usage

```
## S3 method for class 'emmreml_gwas'  
summary(object, ...)
```

Arguments

object	An object of class 'emmreml_gwas'
...	Additional arguments

Value

A summary object

summary.emmreml_multi *Summary Method for emmreml_multi Objects*

Description

Summary Method for emmreml_multi Objects

Usage

```
## S3 method for class 'emmreml_multi'  
summary(object, ...)
```

Arguments

object	An object of class 'emmreml_multi'
...	Additional arguments

Value

A summary object

visualize_kinship	<i>Visualize a kinship matrix as a heatmap</i>
-------------------	--

Description

Visualize a kinship matrix as a heatmap

Usage

```
visualize_kinship(
  K,
  labels = NULL,
  clustering = TRUE,
  cluster_method = "complete",
  dist_method = "euclidean",
  palette = "viridis",
  title = "Kinship Matrix"
)
```

Arguments

K	A kinship (relationship) matrix
labels	Character vector of sample labels (optional)
clustering	Logical, whether to perform hierarchical clustering (default: TRUE)
cluster_method	Character, clustering method (default: "complete")
dist_method	Character, distance method for clustering (default: "euclidean")
palette	Character, color palette to use (default: "viridis")
title	Character, plot title (default: "Kinship Matrix")

Value

A ggplot2 object

Examples

```
## Not run:
M_012 <- matrix(sample(c(0,1,2), 100*1000, replace=TRUE), 100, 1000)
K <- calculate_kinship(M_012, method="vanraden1", coding="012")
visualize_kinship(K)

## End(Not run)
```

Index

calculate_allele_frequencies, [3](#), [23](#)
calculate_kinship, [3](#), [23](#)
compare_kinship, [5](#), [23](#)
convert_marker_coding, [5](#), [23](#)

demo_gwas, [6](#)
demo_K_optimization, [7](#)
demo_kinship, [7](#)
demo_multivariate, [8](#)

EMMREML, [10](#)
emmreml, [8](#), [14](#), [24](#)
emmremlMultiKernel, [11](#), [14](#), [24](#)
emmremlMultivariate, [13](#)
emmremlMultivariate_core, [14](#), [15](#)

filter_markers, [17](#), [23](#)

generate_example_data, [18](#)
gwas, [19](#), [24](#)

identify_marker_coding, [21](#), [23](#)
impute_markers, [22](#), [23](#)

kinship, [23](#)

marker_statistics, [23](#), [25](#)

plot.emmreml_gwas, [26](#)
plot_marker_stats, [23](#), [26](#)
predict.emmreml, [27](#)
print.emmreml, [28](#)
print.emmreml_gwas, [28](#)
print.emmreml_multi, [29](#)
print.summary.emmreml, [29](#)
print.summary.emmreml_gwas, [30](#)
print.summary.emmreml_multi, [30](#)

subset_markers_by_maf, [23](#), [31](#)
summarize_marker_data, [23](#), [31](#)
summary.emmreml, [32](#)

summary.emmreml_gwas, [33](#)
summary.emmreml_multi, [33](#)

visualize_kinship, [23](#), [34](#)