Package 'FieldSimR'

April 12, 2023

Title Simulation of Plot Errors and Phenotypes in Plant Breeding Field Trials

Version 1.1.0

Date 2023-04-10

Maintainer Christian Werner <werner.christian@proton.me>

Description Simulates plot data in field trials for multiple traits across multiple environments. Its core function generates plot errors comprising 1) a spatially correlated error term, 2) a random error term, and 3) an extraneous error term. Spatially correlated errors are simulated using either bivariate interpolation or a two-dimensional autoregressive process of order one (AR1:AR1). The three error terms are combined at a user-defined ratio. Plot phenotypes can be generated by combining the simulated errors with genetic values (e.g. true, simulated or predicted). 'FieldSimR' provides wrapper functions to simulate genetic values for multiple traits across multiple environments using the 'R' package 'AlphaSimR'.

License GPL (>= 3)

URL https://github.com/crWerner/fieldsimr, https://crwerner.github.io/fieldsimr/

Encoding UTF-8

LazyData true

Imports ggplot2, interp, lattice, matrixcalc, mbend

Suggests AlphaSimR, knitr, rmarkdown

RoxygenNote 7.2.3

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation no

Author Christian Werner [aut, cre] (<https://orcid.org/0000-0001-9400-5061>), Daniel Tolhurst [aut] (<https://orcid.org/0000-0002-4787-080X>), Chris Gaynor [ctb] (<https://orcid.org/0000-0003-0558-6656>), Giovanny Covarrubias-Pazaran [ctb] (<https://orcid.org/0000-0002-7194-3837>), Lorena Batista [ctb] (<https://orcid.org/0000-0001-8472-8776>)

Repository CRAN

Date/Publication 2023-04-12 16:10:02 UTC

R topics documented:

compsym_asr_input 2
compsym_asr_output 5
df_error_bivar 7
df_gv_unstr 8
field_trial_error 9
make_phenotypes 12
plot_effects 13
qq_plot 14
rand_cor_mat 15
sample_variogram 16
theoretical_variogram 17
unstr_asr_input 18
unstr_asr_output 22

Index **25**

`compsym_asr_input` *Simulate genetic values based on a compound symmetry model for GxE interaction - 'AlphaSimR' input parameters*

Description

Creates a list of input parameters for 'AlphaSimR' to simulate genetic values for multiple traits across multiple environments based on a compound symmetry model for genotype-by-environment (GxE) interaction.
By default, 'AlphaSimR' does not support complex models for GxE interaction. However, its functionality to simulate correlated genetic values can be utilised for this purpose by providing the required variance structures. `compsym_asr_input` is a wrapper function to construct the variance structures required to simulate GxE interaction in 'AlphaSimR' based on a multi-trait compound symmetry model. This function assumes a separable structure between traits and environments. After simulating the genetic values, the wrapper function compsym_asr_output can be used to obtain data frames with the values.

Usage

```
compsym_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = 0,
  var = 1,
  rel_main_eff_A = 0.5,
  cor_A = NULL,
```

```
  mean_DD = NULL,
  var_DD = NULL,
  rel_main_eff_DD = NULL,
  cor_DD = NULL,
  rel_AA = NULL,
  rel_main_eff_AA = NULL,
  cor_AA = NULL
)
```

Arguments

n_envs	Number of environments to be simulated. A minimum of two environments is required.
n_traits	Number of traits to be simulated.
mean	A vector of mean genetic values for each environment-within-trait combination. If only one value is specified, all environment-within-trait combinations will be assigned the same mean.
var	A vector of genetic variances for each trait. Simulated traits are restricted by the compound symmetry model to having the same variance for each environment (i.e., main effect variance + GxE interaction variance) and the same covariance between each pair of environments (main effect variance). **Note:** When `useVarA = TRUE` is specified in 'AlphaSimR' (default) the values in `var` represent the `additive` genetic variances, otherwise they will represent the `total` (additive + non-additive) genetic variances.
rel_main_eff_A	A vector defining the magnitude of the additive main effect variance relative to the additive main effect + GxE interaction variance for each trait. If only one value is specified, all traits will be assigned the same relative magnitude. **Note:** `0 < rel_main_eff_A < 1`.
cor_A	A matrix of additive genetic correlations between traits. By default, a diagonal matrix is constructed.
mean_DD	A vector of mean dominance degrees for each environment-within-trait combination (similar to `mean`). If only one value is specified, all environment-within-trait combinations will be assigned the same mean. By default, `mean_DD = NULL` and dominance is not simulated.
var_DD	A vector of dominance degree variances for each trait. Simulated traits have the same dominance degree variance for each environment and the same dominance degree covariance between each pair of environments (similar to `var`).
rel_main_eff_DD	A vector defining the magnitude of the dominance degree main effect variance relative to the main effect + GxE interaction variance for each trait (similar to `rel_main_eff_A`). If only one value is specified, all traits will be assigned the same relative magnitude. **Note:** `0 < rel_main_eff_DD < 1`.
cor_DD	A matrix of dominance degree correlations between traits (similar to `cor_A`). If not specified and dominance is simulated, a diagonal matrix is constructed.

`rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each trait, that is in a diploid organism with allele frequency 0.5. Simulated traits have the same epistatic variance for each environment and the same epistatic covariance between each pair of environments (similar to `var`). If only one value is specified, all traits will be assigned the same relative magnitude.

`rel_main_eff_AA` A vector defining the magnitude of the epistatic main effect variance relative to the main effect + GxE interaction variance for each trait (similar to `rel_main_eff_A`). If only one value is specified, all traits will be assigned the same relative magnitude.
Note: `0 < rel_main_eff_AA < 1`.

`cor_AA` A matrix of epistatic correlations between traits (similar to `cor_A`). If not specified and epistasis is simulated, a diagonal matrix is constructed.

Details

Note: 'AlphaSimR' can simulate different biological effects (see: SimParam).

- For additive traits use `addTraitA()`.
- For additive + dominance traits use `addTraitAD()`.
- For additive + epistatic traits use `addTraitAE()`.
- For additive + dominance + epistatic traits use `addTraitADE()`.

If non-additive effects are to be simulated, check the `useVarA` argument of these functions.

Value

A list containing input parameters for 'AlphaSimR', which is used to simulate correlated genetic effects based on a compound symmetry model.

Examples

```
# Simulate genetic values in 'AlphaSimR' for two additive + dominance traits across
# three environments based on a compound symmetry model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees.
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # Trait 1 and 2, same values in the three environments.

# Additive genetic variances and dominance degree variances.
var <- c(0.08, 13) # Different values set for traits 1 and 2.
var_DD <- c(0.2, 0.2) # The same value set for traits 1 and 2.

# Relative magnitude of the additive and dominance degree main effect variances.
rel_main_eff_A <- c(0.4, 0.6) # Different values set for traits 1 and 2.
rel_main_eff_DD <- 0.4 # The same value set for traits 1 and 2.

# Additive and dominance degree correlations between the two simulated traits.
```

```
cor_A <- matrix( # Additive correlation matrix.
  c(
    1.0, 0.5,
    0.5, 1.0
  ),
  ncol = 2
)
cor_DD <- diag(2) # Dominance correlation matrix - assume independence.

input_asr <- compsym_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  var = var,
  rel_main_eff_A = rel_main_eff_A,
  cor_A = cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  rel_main_eff_DD = rel_main_eff_DD,
  cor_DD = cor_DD
)
```

`compsym_asr_output` *Simulate genetic values based on a compound symmetry model for GxE interaction - Simulation using 'AlphaSimR'*

Description

Creates a data frame of simulated genetic values for multiple traits across multiple environments based on a compound symmetry model for genotype-by-environment (GxE) interaction. This function requires an 'AlphaSimR' population object generated using the function compsym_asr_input.

Usage

```
compsym_asr_output(pop, n_envs, n_traits, n_reps, effects = FALSE)
```

Arguments

`pop`	An 'AlphaSimR' population object (Pop-class or HybridPop-class) generated using compsym_asr_input.
`n_envs`	Number of simulated environments (same number used in compsym_asr_input).
`n_traits`	Number of simulated traits (same number used in compsym_asr_input).
`n_reps`	A vector defining the number of complete replicates (blocks) in each environment. If only one value is specified, all environments will be assigned the same number.
`effects`	When `TRUE`, a list is returned with additional entries containing the total (additive + dominance + epistatic) main effects and GxE interaction effects for each environment-within-trait combination. By default, `effects = FALSE`.

Value

A data frame with columns 'env', 'rep', genotype 'id', and the simulated genetic values for each trait. When `effects = TRUE`, a list is returned with additional entries containing the total (additive + dominance + epistatic) main effects and GxE interaction effects for each environment-within-trait combination.

Examples

```
# Simulate genetic values in 'AlphaSimR' for two additive + dominance traits across
# three environments based on a compound symmetry model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees.
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # Trait 1 and 2, same values in the three environments.

# Additive genetic variances and dominance degree variances.
var <- c(0.08, 13) # Different values set for traits 1 and 2.
var_DD <- c(0.2, 0.2) # The same value set for traits 1 and 2.

# Relative magnitude of the additive and dominance degree main effect variances.
rel_main_eff_A <- c(0.4, 0.6) # Different values set for traits 1 and 2.
rel_main_eff_DD <- 0.4 # The same value set for traits 1 and 2.

# Additive and dominance degree correlations between the two simulated traits.
cor_A <- matrix( # Additive correlation matrix.
  c(
    1.0, 0.5,
    0.5, 1.0
  ),
  ncol = 2
)
cor_DD <- diag(2) # Dominance correlation matrix - assume independence.

input_asr <- compsym_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  var = var,
  rel_main_eff_A = rel_main_eff_A,
  cor_A = cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  rel_main_eff_DD = rel_main_eff_DD,
  cor_DD = cor_DD
)


# 2. Use input_asr to simulate genetic values in 'AlphaSimR' based on a compound symmetry model
# for GxE interaction.

library("AlphaSimR")
```

```
FOUNDERPOP <- quickHaplo(
  nInd = 100,
  nChr = 6,
  segSites = 100
)

SP <- SimParam$new(FOUNDERPOP)

SP$addTraitAD(
  nQtlPerChr = 100,
  mean = input_asr$mean,
  var = input_asr$var,
  meanDD = input_asr$mean_DD,
  varDD = input_asr$var_DD,
  corA = input_asr$cor_A,
  corDD = input_asr$cor_DD,
  useVarA = TRUE
)

# By default, the value provided in 'var' represents the additive variance.
# If useVarA = FALSE, 'var' represents the total genetic variance.

pop <- newPop(FOUNDERPOP)


# 3. Create a data frame containing the simulated genetic values for the two traits across the
# three environments.

n_reps <- c(3, 3, 2) # Vector containing the number of complete replicates in each environment.

gv_df <- compsym_asr_output(
  pop = pop,
  n_envs = 3,
  n_traits = 2,
  n_reps = n_reps,
  effects = TRUE
)
```

`df_error_bivar` *Plot errors - example data frame*

Description

An example data frame of simulated plot errors for two traits across three locations. The first two locations include three replicates (blocks), and the third location includes two replicates. Each replicate comprises 10 rows and 10 columns. The data frame was generated using the function field_trial_error with bivariate interpolation. The simulation of the plot errors is shown in the vignette on the Simulation of plot errors and phenotypes for a multi-environment plant breeding trial.

Usage

```
df_error_bivar
```

Format

A data frame with 800 rows and 6 columns:

env Environment number

block Block (replicate) number

col Column number

row Row number

e.Trait.1 Simulated plot error for trait 1

e.Trait.2 Simulated plot error for trait 2

---

df_gv_unstr *Genetic values - example data frame*

Description

An example data frame of simulated genetic values for 100 genotypes with two traits across three environments. The data frame was generated using the wrapper functions unstr_asr_input and unstr_asr_output to simulate correlated genetic values based on an unstructured model for genotype-by-environment (GxE) interaction with 'AlphaSimR'. The simulation of the genetic values is shown in the vignette on the Simulation of genetic values using an unstructured GxE interaction model.

Usage

```
df_gv_unstr
```

Format

A data frame with 800 rows and 5 columns:

env Environment number

rep Replicate number

id Genotype id

gv.Trait.1 Simulated genetic values for trait 1

gv.Trait.2 Simulated genetic values for trait 2

`field_trial_error` *Simulate plot errors in plant breeding field trials*

Description

Creates a data frame of simulated field trial plot errors for one or more traits across one or more environments. The plot errors can consist of a spatial error, a random error and extraneous error component. The spatial error is simulated according to either 1) bivariate interpolation using the `interp` function of the package 'interp', or 2) a separable first-order autoregressive process (AR1:AR1). The random error is simulated using an independent process. The extraneous error is simulated as the sum of column and/or row terms, where the user can choose from an independent or a correlated processes. The spatial, random and extraneous errors are combined according to a user-defined ratio.

For multiple traits, correlated errors can be simulated assuming 1) correlated spatial error between traits, 2) correlated random error between traits, 3) correlated extraneous error between traits, or 4) some combination of 1-3.

A separable correlation structure is assumed between traits and environments, but different variances can be assigned to different environment-within-trait combinations.

Usage

```
field_trial_error(
  n_envs = 1,
  n_traits = 1,
  n_reps = 2,
  n_cols = 10,
  n_rows = 20,
  rep_dir = "col",
  var_R = 1,
  S_cor_R = NULL,
  R_cor_R = NULL,
  E_cor_R = NULL,
  spatial_model = "Bivariate",
  complexity = NULL,
  plot_length = 5,
  plot_width = 2,
  col_cor = 0.4,
  row_cor = 0.6,
  prop_spatial = 0.5,
  prop_ext = 0,
  ext_dir = "row",
  ext_col_cor = 0,
  ext_row_cor = 0,
  return_effects = FALSE
)
```

Arguments

n_envs	Number of environments to be simulated (same number used for compsym_asr_input or unstr_asr_output, where applicable).
n_traits	Number of traits to be simulated.
n_reps	A vector specifying the number of complete replicates (blocks) in each environment. If only one value is provided, all environments will be assigned the same number.
n_cols	A vector specifying the total number of columns in each environment. If only one value is provided, all environments will be assigned the same number.
n_rows	A vector specifying the total number of rows in each environment. If only one value is provided, all environments will be assigned the same number.
rep_dir	A vector specifying the direction of replicates (blocks) in each environment. Use 'col' for a side-by-side arrangement (default), 'row' for an above-and-below arrangement, or NA if only one replicate block is simulated. If only one value is provided, all environments will be assigned the same replicate block direction (where applicable).
var_R	A vector of error variances for each environment-within-trait combination. If only one value is provided, all environment-within-trait combinations will be assigned the same error variance.
S_cor_R	A matrix of spatial error correlations between traits. If not specified and a spatial error is simulated, a diagonal matrix is constructed.
R_cor_R	A matrix of random error correlations between traits. If not specified and a random error is simulated, a diagonal matrix is constructed.
E_cor_R	A matrix of extraneous error correlations between traits. If not specified and an extraneous error is simulated, a diagonal matrix is constructed. **Note:** the same correlation between traits is used for the column and row errors (where applicable).
spatial_model	A character string specifying the model used to simulate the two-dimensional spatial error term. One of either 'Bivariate' (bivariate interpolation, the default) or 'AR1:AR1' (separable first-order autoregressive process).
complexity	A vector specifying the complexity of the bivariate interpolation in each environment. If only one value is provided, all environments will be assigned the same complexity. If not specified and spatial_model = "Bivariate", the complexity is set to the maximum number of columns and rows in each environment. This generally provides good results. See 'interp' for further details.
plot_length	A vector of plot lengths (column direction, usually longer side) for each environment. If only one value is provided, all environments will be assigned the same plot length. Only required when spatial_model = "Bivariate".
plot_width	A vector of plot widths (row direction, usually shorter side) for each environment. If only one value is provided, all environments will be assigned the same plot width. Only required when spatial_model = "Bivariate".
col_cor	A vector of column autocorrelations for each environment used in the AR1:AR1 spatial error model. If only one value is provided, all environments will be assigned the same column autocorrelation. Only required when spatial_model = "AR1:AR1".

`row_cor` A vector of row autocorrelations for each environment used in the AR1:AR1 spatial error model. If only one value is provided, all environments will be assigned the same row autocorrelation. Only required when `spatial_model = "AR1:AR1"`.

`prop_spatial` A vector specifying the proportion of spatial error variance to total error variance (spatial + random + extraneous) for each environment-within-trait combination. If only one value is provided, all environment-within-trait combinations will be assigned the proportion of spatial error variance.

`prop_ext` A vector specifying the proportion of extraneous error variance to total error variance (spatial + random + extraneous) for each environment-within-trait combination. If only one value is provided, all environment-within-trait combinations will be assigned the proportion of extraneous error variance.

`ext_dir` A vector specifying the direction of extraneous variation for each environment. Use 'col' to simulate variation in the column direction, 'row' (default) for variation in the row direction, 'both' for variation in both directions, or NA if zero extraneous variation is simulated. When `ext_dir = "both"`, half the variance is assigned to the columns and half is assigned to the rows. If only one value is provided, all environments will be assigned the same direction (where applicable).

`ext_col_cor` A vector of column autocorrelations for each environment used in the extraneous error model. If only one value is provided, all environments will be assigned the same column autocorrelation.

`ext_row_cor` A vector of row autocorrelations for each environment used in the extraneous error model. If only one value is provided, all environments will be assigned the same row autocorrelation.

`return_effects` When `TRUE`, a list is returned with additional entries for each trait containing the spatial, random and extraneous errors. By default, `return_effects = FALSE`.

Value

A data frame with columns 'env', 'block', 'col' and 'row', as well as the simulated error for each trait. When `return_effects = TRUE`, a list is returned with additional columns for each trait providing the spatial, random and extraneous errors.

Examples

```
# Simulate plot errors for two traits across three environments using a bivariate
# interpolation model for spatial variation.

n_envs <- 3
n_traits <- 2

# Field layout
n_cols <- 10
n_rows <- c(30, 30, 20)
rep_dir <- "col"
plot_length <- 5
plot_width <- 2
```

```
n_reps <- c(3, 3, 2)

# Error variances for all six environment-within-trait combinations.
var_R <- c(0.2, 0.4, 0.6, 10, 15, 20) # Trait 1 x 3 environments, trait 2 x 3 environments.

# Spatial error correlations between the two simulated traits.
S_cor_R <- matrix(
  c(
    1.0, 0.2,
    0.2, 1.0
  ),
  ncol = 2
)

# Structure of simulated error.
prop_spatial <- 0.4
prop_ext <- 0.2
ext_dir <- "row"
ext_row_cor <- -0.6

error_df <- field_trial_error(
  n_envs = n_envs,
  n_traits = n_traits,
  n_reps = n_reps,
  n_cols = n_cols,
  n_rows = n_rows,
  rep_dir = "row",
  var_R = var_R,
  S_cor_R = S_cor_R,
  spatial_model = "Bivariate",
  plot_length = plot_length,
  plot_width = plot_width,
  prop_spatial = prop_spatial,
  prop_ext = prop_ext,
  ext_dir = ext_dir,
  ext_row_cor = ext_row_cor,
  return_effects = TRUE
)
```

---

make_phenotypes *Simulate phenotypes - Combine simulated genetic values and plot errors*

Description

Creates a data frame of simulated phenotypes for one or more traits by combining simulated plot errors with genetic values (e.g. true, simulated or predicted). The genetic values can be generated externally, but note that they must be stored in a data frame as described below.

Usage

```
make_phenotypes(gv_df, error_df, randomise = FALSE)
```

Arguments

gv_df	A data frame of genetic values. Must contain the columns 'env', 'rep', and 'id', followed by the genetic values for each trait.
error_df	A data frame of plot errors. Must contain the columns 'env', 'block', 'col', and 'row', followed by the plot errors for each trait.
randomise	When TRUE, genotypes are randomly allocated to plots within blocks to generate a randomized complete block design (RCBD). **Note:** Other experimental designs must be generated externally.

Value

A data frame with columns 'env', 'block', 'column', 'row' and 'genotype', followed by the phenotypes for each trait.

Examples

```
# Simulate phenotypes by combining the genetic values and plot errors provided in
# the two example data frames 'df_gv_unstr' and 'df_error_bivar'.

gv_df <- df_gv_unstr
error_df <- df_error_bivar

pheno_df <- make_phenotypes(
  gv_df,
  error_df,
  randomise = TRUE
)
```

---

plot_effects *Graphics for plot effects*

Description

Graphically displays plot effects (e.g., phenotypes, genetic values, plot errors) onto a field array, in which the colour gradient ranges from red (low value) to green (high value). The function requires a data frame generated with field_trial_error as an input, or any data frame with columns 'col', 'row', and the effect to be displayed. When the data frame contains a 'block' column, the field array is split into blocks if `blocks = TRUE`.

Usage

```
plot_effects(df, effect, blocks = TRUE)
```

Arguments

df	A data frame with columns 'col', 'row', and the effect to be plotted. When df also contains a 'block' column, the field array is split into blocks if blocks = TRUE.
effect	The effect to be plotted.
blocks	When TRUE (default), the field array is split into blocks.

Value

A graphical field array, in which the colour gradient ranges from red (low value) to green (high value).

Examples

```
# Plot the simulated plot errors for trait 2 in environment 2 provided in the example data
# frame 'df_error_bivar'.

error_df <- df_error_bivar[df_error_bivar$env == 2, ]

plot_effects(
  df = error_df,
  effect = "e.Trait.2"
)
```

---

qq_plot *Q-Q plot*

Description

Creates a quantile-quantile (Q-Q) plot which compares the theoretical quantiles of a normal distribution with the sample quantiles of the distribution of user effects.

Usage

```
qq_plot(df, effect, labels = FALSE)
```

Arguments

df	A data frame containing the effect to be plotted.
effect	The name of the effect to be plotted.
labels	When FALSE (default), data points without labels are plotted. When TRUE, column and row labels are shown in the Q-Q plot. This requires additional columns 'col' and 'row' in the data frame.

Value

A Q-Q plot with the x- and y-axes displaying the theoretical and sample quantiles of the effect to be plotted, respectively.

Examples

```
# Q-Q plot of the simulated plot errors for trait 2 in environment 2 provided in the example
# data frame 'df_error_bivar'.

error_df <- df_error_bivar[df_error_bivar$env == 2, ]

qq <- qq_plot(
  df = error_df,
  effect = "e.Trait.2",
  labels = TRUE
)

# Q-Q plot
qq

# Extraction of a data frame containing the theoretical and sample quantiles of
# the effect to be plotted.
qq_df <- qq$data
```

---

rand_cor_mat *Random correlation matrix*

Description

Creates a symmetric p x p correlation matrix with user-defined minimum and maximum correlation values.

Usage

```
rand_cor_mat(p, min_cor = -1, max_cor = 1, pos_def = FALSE)
```

Arguments

p	A scalar defining the dimensions of the correlation matrix.
min_cor	A scalar defining the minimum potential value.
max_cor	A scalar defining the maximum potential value.
pos_def	When TRUE, the function 'bend' of the package 'mbend' is used with default arguments to bend a non-positive-definite correlation matrix to a positive-definite matrix (when appropriate). By default, pos_def = FALSE.

Value

A symmetric p x p correlation matrix.

Examples

```
cor_A <- rand_cor_mat(10, min_cor = -0.2, max_cor = 0.8, pos_def = TRUE)
```

`sample_variogram` *Sample variogram*

Description

Creates a sample variogram. The x- and y-axes display the row and column displacements, respectively. The z-axis displays the semi-variance (variogram ordinates).

Usage

```
sample_variogram(df, effect, min_np = 30)
```

Arguments

`df`	A data frame containing the columns 'col', 'row', and the effect to be plotted.
`effect`	The name of the effect to be plotted.
`min_np`	Only semi variances based on at least `min_np` pairs of plots will be displayed. By default, `min_np = 30`.

Value

Graphic of the sample variogram, where the x- and y-axes display the row and column displacements and the z-axis displays the semi-variance (variogram ordinates).

Examples

```
# Sample variogram of the simulated plot errors for trait 2 in environment 2 provided in the
# example data frame 'df_error_bivar'.

error_df <- df_error_bivar[df_error_bivar$env == 2, ]

vario <- sample_variogram(
  df = error_df,
  effect = "e.Trait.2",
)

# Sample variogram
vario

# Extraction of a data frame containing the column and row displacements as well as the
# semi-variances (sample variogram ordinates).

sample_df <- vario$data
```

`theoretical_variogram` *Theoretical variogram*

Description

Creates a theoretical variogram. The x- and y-axes display the row and column displacements, respectively. The z-axis displays the semi-variance (variogram ordinates).

Usage

```
theoretical_variogram(
  n_cols,
  n_rows,
  var_R = 1,
  prop_spatial = 0.5,
  col_cor,
  row_cor
)
```

Arguments

`n_cols`	A scalar defining the number of columns.
`n_rows`	A scalar defining the number of rows.
`var_R`	A scalar defining the total error variance.
`prop_spatial`	A scalar defining the proportion of spatial error variance to total error variance (spatial + random).
`col_cor`	A scalar defining the column autocorrelation value.
`row_cor`	A scalar defining the row autocorrelation value.

Value

Graphic of the theoretical variogram, where the x- and y- axes display the row and column displacements and the z-axis displays the semi-variance (variogram ordinates).

Examples

```
# Theoretical variogram for a field with 10 columns and 20 rows, using column and row
# autocorrelations of 0.4 and 0.8.

vario <- theoretical_variogram(
  n_cols = 10,
  n_rows = 20,
  var_R = 1,
  prop_spatial = 0.5,
  col_cor = 0.4,
  row_cor = 0.8
)
```

```
# Theoretical variogram
vario

# Extraction of a data frame containing the column and row displacements as well as the
# semi-variances (theoretical variogram ordinates).

theoretical_df <- vario$data
```

`unstr_asr_input` *Simulate genetic values based on an unstructured model for GxE interaction - 'AlphaSimR' input parameters*

Description

Creates a list of input parameters for 'AlphaSimR' to simulate genetic values for multiple traits across multiple environments based on an unstructured model for genotype-by-environment (GxE) interaction.
By default, 'AlphaSimR' does not support complex models for GxE interaction. However, its functionality to simulate correlated genetic values can be utilised for this purpose by providing the required variance structures. `unstr_asr_input` is a wrapper function to construct the variance structures required to simulate GxE interaction in 'AlphaSimR' based on a multi-trait unstructured model. This function can handle separable and non-separable structures between traits and environments (see below). After simulating the genetic values, the wrapper function unstr_asr_output can be used to obtain a data frame with the relevant values.

Usage

```
unstr_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = 0,
  var = 1,
  T_var = NULL,
  E_var = NULL,
  cor_A = NULL,
  T_cor_A = NULL,
  E_cor_A = NULL,
  mean_DD = NULL,
  var_DD = NULL,
  T_var_DD = NULL,
  E_var_DD = NULL,
  cor_DD = NULL,
  T_cor_DD = NULL,
  E_cor_DD = NULL,
  rel_AA = NULL,
  T_rel_AA = NULL,
```

```
  E_rel_AA = NULL,
  cor_AA = NULL,
  T_cor_AA = NULL,
  E_cor_AA = NULL
)
```

Arguments

n_envs	Number of environments to be simulated. A minimum of two environments is required.
n_traits	Number of traits to be simulated.
mean	A vector of mean genetic values for each environment-within-trait combination. If only one value is specified, all environment-within-trait combinations will be assigned the same mean.
var	A vector of genetic variances for each environment-within-trait combination. If only one value is specified, all environment-within-trait combinations will be assigned the same mean. **Alternatively**, if a separable structure between traits and environments is desired, T_var and E_var can be specified.
T_var	A vector of genetic variances for each trait. Must be provided in combination with E_var. **Alternatively**, var can be specified.
E_var	A vector of genetic variances for each environment. Must be provided in combination with T_var. **Alternatively**, var can be specified.
cor_A	A matrix of additive genetic correlations between all environment-within-trait combinations. By default, a diagonal matrix is constructed. **Alternatively**, T_cor_A and E_cor_A can be specified.
T_cor_A	A matrix of additive genetic correlations between traits. Must be provided in combination with E_cor_A. **Alternatively**, cor_A can be specified.
E_cor_A	A matrix of additive genetic correlations between environments. Must be provided in combination with T_cor_A. **Alternatively**, cor_A can be specified.
mean_DD	A vector of mean dominance degrees for each environment-within-trait combination (similar to mean). If only one value is specified, all environment-within-trait combinations will be assigned the same mean. By default, mean_DD = NULL and dominance is not simulated.
var_DD	A vector of dominance degree variances for each environment-within-trait combination (similar to var). If only one value is specified, all environment-within-trait combinations will be assigned the same variance. **Alternatively**, if a separable structure between traits and environments is desired, T_var_DD and E_var_DD can be specified.
T_var_DD	A vector of dominance degree variances for each trait (similar to T_var). Must be provided in combination with E_var_DD. **Alternatively**, var_DD can be specified.

`E_var_DD` A vector of dominance degree genetic variances for each environment (similar to `E_var`). Must be provided in combination with `T_var_DD`.
Alternatively, `var_DD` can be specified.

`cor_DD` A matrix of dominance degree correlations between all environment-within-trait combinations (similar to `cor_A`). If not specified and dominance is simulated, a diagonal matrix is constructed.
Alternatively, `T_cor_DD` and `E_cor_DD` can be specified.

`T_cor_DD` A matrix of dominance degree correlations between traits (similar to `T_cor_A`). Must be provided in combination with `E_cor_DD`.
Alternatively, `cor_DD` can be specified.

`E_cor_DD` A matrix of dominance degree correlations between environments (similar to `E_cor_A`). Must be provided in combination with `T_cor_DD`.
Alternatively, `cor_DD` can be specified.

`rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to additive genetic variance for each environment-within-trait combination, that is in a diploid organism with allele frequency 0.5. If only one value is specified, all environment-within-trait combinations will be assigned the same value. By default, `rel_AA = NULL` and epistasis is not simulated.
Alternatively, if a separable structure between traits and environments is desired, `T_rel_AA` and `E_rel_AA` can be specified.

`T_rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each trait. Must be provided in combination with `E_rel_AA`.
Alternatively, `rel_AA` can be specified.

`E_rel_AA` A vector defining the magnitude of additive-by-additive (epistatic) variance relative to the additive genetic variance for each environment. Must be provided in combination with `T_rel_AA`.
Alternatively, `rel_AA` can be specified.

`cor_AA` A matrix of epistatic correlations between all environment-within-trait combinations (similar to `cor_A`). If not specified and epistasis is simulated, a diagonal matrix is constructed.
Alternatively, `T_cor_AA` and `E_cor_AA` can be specified.

`T_cor_AA` A matrix of epistatic correlations between traits (similar to `T_cor_A`). Must be provided in combination with `E_cor_AA`.
Alternatively, `cor_AA` can be specified.

`E_cor_AA` A matrix of epistatic correlations between environments (similar to `E_cor_A`). Must be provided in combination with `T_cor_AA`.
Alternatively, `cor_AA` can be specified.

Details

`unstr_asr_input` can handle separable and non-separable structures between traits and environments.

- For separable structures, provide (1) `T_var` & `E_var`, and (2) `T_cor_A` & `E_cor_A`.
- For non-separable structures, provide (1) `var`, and (2) `cor_A`.

Note: 'AlphaSimR' can simulate different biological effects (see: SimParam).

- For additive traits use `addTraitA()`.
- For additive + dominance traits use `addTraitAD()`.
- For additive + epistatic traits use `addTraitAE()`.
- For additive + dominance + epistatic traits use `addTraitADE()`.

If non-additive effects are to be simulated, check the `useVarA` argument of these functions.

Value

A list containing input parameters for 'AlphaSimR', which is used to simulate correlated genetic effects based on an unstructured model.

Examples

```
# Simulate genetic values in 'AlphaSimR' for two additive + dominance traits across
# three environments based on an unstructured model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees.
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # Trait 1 and 2, same values in the three environments.

# Additive genetic variances and dominance degree variances.
var <- c(0.085, 0.12, 0.06, 15.1, 8.5, 11.7) # Trait 1 x 3 environments, trait 2 x 3 environments.
var_DD <- rep(0.2, 6) # The same value set for traits 1 and 2.

# Additive genetic correlations between the two simulated traits.
T_cor_A <- matrix(
  c(
    1.0, 0.6,
    0.6, 1.0
  ),
  ncol = 2
)

# Additive genetic correlations between the three simulated environments.
E_cor_A <- matrix(
  c( # Matrix of additive genetic correlations between the three environments.
    1.0, 0.4, 0.6,
    0.4, 1.0, 0.5,
    0.6, 0.5, 1.0
  ),
  ncol = 3
)

# Dominance degree correlations between all six environment-within-trait combinations.
cor_DD <- diag(6) # Assuming independence between traits

input_asr <- unstr_asr_input(
  n_envs = 3,
```

```
  n_traits = 2,
  mean = mean,
  var = var,
  T_cor_A = T_cor_A,
  E_cor_A = E_cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  cor_DD = cor_DD
)
```

`unstr_asr_output` *Simulate genetic values based on an unstructured model for GxE interaction - Simulation using 'AlphaSimR'*

Description

Creates a data frame of simulated genetic values for multiple traits across multiple environments based on an unstructured model for genotype-by-environment (GxE) interaction. This function requires an 'AlphaSimR' population object generated using the unstr_asr_input function.

Usage

```
unstr_asr_output(pop, n_envs, n_traits, n_reps)
```

Arguments

`pop`	An 'AlphaSimR' population object (Pop-class or HybridPop-class) generated using unstr_asr_input.
`n_envs`	Number of simulated environments (same number used in unstr_asr_input).
`n_traits`	Number of simulated traits (same number used in unstr_asr_input).
`n_reps`	A vector defining the number of complete replicates (blocks) in each environment. If only one value is specified, all environments will be assigned the same number.

Value

A data frame with columns 'env', 'rep', genotype 'id', and the simulated genetic values for each trait.

Examples

```
# Simulate genetic values in 'AlphaSimR' for two additive + dominance traits across
# three environments based on an unstructured model for GxE interaction.

# 1. Define the genetic architecture of the simulated traits.
# Mean genetic values and mean dominance degrees.
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # Trait 1 x 3 environments, trait 2 x 3 environments.
mean_DD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # Trait 1 and 2, same values in the three environments.
```

```
# Additive genetic variances and dominance degree variances.
var <- c(0.085, 0.12, 0.06, 15.1, 8.5, 11.7) # Trait 1 x 3 environments, trait 2 x 3 environments.
var_DD <- rep(0.2, 6) # The same value set for traits 1 and 2.

# Additive genetic correlations between the two simulated traits.
T_cor_A <- matrix(
  c(
    1.0, 0.6,
    0.6, 1.0
  ),
  ncol = 2
)

# Additive genetic correlations between the three simulated environments.
E_cor_A <- matrix(
  c( # Matrix of additive genetic correlations between the three environments.
    1.0, 0.4, 0.6,
    0.4, 1.0, 0.5,
    0.6, 0.5, 1.0
  ),
  ncol = 3
)

# Dominance degree correlations between all six environment-within-trait combinations.
cor_DD <- diag(6) # Assuming independence between traits

input_asr <- unstr_asr_input(
  n_envs = 3,
  n_traits = 2,
  mean = mean,
  var = var,
  T_cor_A = T_cor_A,
  E_cor_A = E_cor_A,
  mean_DD = mean_DD,
  var_DD = var_DD,
  cor_DD = cor_DD
)


# 2. Use input_asr to simulate genetic values in 'AlphaSimR' based on an unstructured model for
# GxE interaction.

library("AlphaSimR")
FOUNDERPOP <- quickHaplo(
  nInd = 100,
  nChr = 6,
  segSites = 100
)

SP <- SimParam$new(FOUNDERPOP)

SP$addTraitAD(
```

```
  nQtlPerChr = 100,
  mean = input_asr$mean,
  var = input_asr$var,
  meanDD = input_asr$mean_DD,
  varDD = input_asr$var_DD,
  corA = input_asr$cor_A,
  corDD = input_asr$cor_DD,
  useVarA = TRUE
)

# By default, the value provided in 'var' represents the additive variance.
# If useVarA=FALSE, 'var' represents the total genetic variance.

pop <- newPop(FOUNDERPOP)


# 3. Create a data frame containing the simulated genetic values for the two traits across the
# three environments.

n_reps <- c(3, 3, 2) # Vector containing the number of complete replicates in each environment.

gv_df <- unstr_asr_output(
  pop = pop,
  n_envs = 3,
  n_traits = 2,
  n_reps = n_reps
)
```

Index

∗ **datasets**
    df_error_bivar, 7
    df_gv_unstr, 8

compsym_asr_input, 2, *5*
compsym_asr_output, *2*, 5

df_error_bivar, 7
df_gv_unstr, 8

field_trial_error, *7*, 9

make_phenotypes, 12

plot_effects, 13

qq_plot, 14

rand_cor_mat, 15

sample_variogram, 16

theoretical_variogram, 17

unstr_asr_input, *8*, 18, *22*
unstr_asr_output, *8*, *18*, 22