

# Package: RfmallocStatgen (via r-universe)

October 10, 2026

**Title** Statistical Genetics on Rfmalloc Genotype Tensors

**Version** 0.1.0

**Description** Runs genome-scale statistical genetics methods over Rfmalloc's file-backed, compressed genotype tensors ('bed' at 2 bits per genotype, 'dosage' at 1 byte per genotype), closing over the analysis stack popularized by 'bigsnpr' and 'bigstatsr' on denser storage with native readers, and adding haplotype methods that a genotype-only stack cannot express. Methods are implemented from published algorithms and validated against 'bigsnpr' and 'bigstatsr' as numerical oracles. The exception is the randomized-SVD workhorse, which is pinned from GPL-3 'PCAone' (Li 2023) with a deterministic dependency trim and R-console adaptation. The current surface includes streamed univariate regression and principal components, banded linkage disequilibrium, LDpred2, and an all-pairs colocalisation expressed as a single matrix multiply over any registered compute backend.

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 4.0.0), Rfmalloc

**Imports** stats

**LinkingTo** RcppEigen, Rfmalloc

**Suggests** Rggml, Rpgen, tinytest, bigsnpr, bigstatsr, bigsparser, Matrix, pcaone

**Remotes** Rfmalloc=sounkou-bioinfo/Rfmalloc/packages/Rfmalloc,  
Rggml=sounkou-bioinfo/Rfmalloc/packages/Rggml

**SystemRequirements** C++17

**Roxygen** list(markdown = TRUE)

**URL** <https://github.com/sounkou-bioinfo/Rfmalloc>,  
<https://sounkou-bioinfo.github.io/Rfmalloc/RfmallocStatgen/>

**BugReports** <https://github.com/sounkou-bioinfo/Rfmalloc/issues>

**Config/roxygen2/version** 8.0.0  
**Config/pak/sysreqs** make  
**Repository** https://rgenomicsetl.r-universe.dev  
**Date/Publication** 2026-09-09 23:54:20 UTC  
**RemoteUrl** https://github.com/RGenomicsETL/Rfmalloc  
**RemoteRef** HEAD  
**RemoteSha** 156efa6ec133a512d8969fded81aaa630ff9ba16  
**RemoteSubdir** packages/RfmallocStatgen

Contents

|                            |   |
|----------------------------|---|
| statgen_coloc_bf . . . . . | 2 |
| statgen_gwas_lin . . . . . | 4 |
| statgen_ldpred2 . . . . .  | 5 |
| statgen_pca . . . . .      | 7 |
| statgen_snp_cor . . . . .  | 9 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>11</b> |
|--------------|-----------|

---

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| statgen_coloc_bf | <i>Colocalisation of two traits' signals by Bayes factors (all pairs at once)</i> |
|------------------|-----------------------------------------------------------------------------------|

---

Description

A matrix reinterpretation of coloc's coloc.bf\_bf: instead of testing one signal of trait 1 against one signal of trait 2 with vector operations (and looping), the shared-causal (H4) evidence for *every* pair of signals is a matrix product accumulated over bounded SNP panels. The consecutive per-pair calculation and this matrix form are algebraically identical (see Details); the payoff is that the hot kernel is a GEMM, so it rides any backend Rggml offers - CPU, OpenBLAS, or a Vulkan GPU (including via dzn) - and, over Rfmalloc's out-of-core panel-GEMM, signal sets larger than device/host memory. This is the same idea as gpu-coloc (Torch), minus the CUDA/Metal-only lock-in.

Usage

```
statgen_coloc_bf(  
  bf1,  
  bf2,  
  p1 = 1e-04,  
  p2 = 1e-04,  
  p12 = 1e-05,  
  backend = c("cpu", "blas", "vulkan"),  
  trim = FALSE,  
  overlap_min = 0.5,
```

```
    block_size = NULL
  )
```

### Arguments

|             |                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bf1, bf2    | numeric matrices of per-SNP log ABFs, rows = signals, columns = SNPs. Both must have the same number of columns and their columns must be the same, aligned SNPs.                                                                                                                                                                                                          |
| p1, p2, p12 | coloc priors: P(SNP causal for trait 1 only / trait 2 only / both).                                                                                                                                                                                                                                                                                                        |
| backend     | one of "cpu" (base R, double precision), "blas" or "vulkan" (dispatch the GEMM to Rggml's backend; single precision). Non-CPU backends fall back to "cpu" with a message when Rggml or a device is unavailable.                                                                                                                                                            |
| trim        | logical; if TRUE, also return the per-pair posterior overlap ( $\sum_i \min(pp_{1,a,i}, pp_{2,b,i})$ , one minus the total variation distance of the two signals' within-signal posteriors) and mark pairs below overlap_min. This is the analogue of coloc's trim_by_posterior: pairs whose strongly-associated SNPs barely overlap get an unreliable H4 and are flagged. |
| overlap_min | overlap threshold for trim.                                                                                                                                                                                                                                                                                                                                                |
| block_size  | Number of SNP columns transformed and multiplied at once. NULL chooses a block whose two exponentiated input panels occupy at most roughly 64 MiB. The dense signal-pair result is intrinsic, but no complete signal-by-SNP exponential matrix is materialized.                                                                                                            |

### Details

For per-SNP log approximate Bayes factors  $l_1$  (a signal of trait 1) and  $l_2$  (a signal of trait 2) over the same  $n$  aligned SNPs, coloc's `combine.abf` gives the five hypotheses' evidence from three summaries:  $s_1 = \text{lse}(l_1)$ ,  $s_2 = \text{lse}(l_2)$  and  $c = \text{lse}(l_1 + l_2)$  ( $\text{lse} = \text{logSumExp}$ ):

$$lH_1 = \log p_1 + s_1, \quad lH_2 = \log p_2 + s_2,$$

$$lH_3 = \log p_1 + \log p_2 + \text{logdiff}(s_1 + s_2, c), \quad lH_4 = \log p_{12} + c,$$

and  $PP_k = \exp(lH_k - \text{lse}(0, lH_1, \dots, lH_4))$ . Only  $c$  couples the two signals. Stacking trait 1's signals as rows of bf1 ( $S_1 \times n$ ) and trait 2's as bf2 ( $S_2 \times n$ ), the whole  $S_1 \times S_2$  matrix of  $c$  is, with per-row maxima  $m_1, m_2$  and  $E_1 = \exp(bf1 - m_1)$ ,  $E_2 = \exp(bf2 - m_2)$ :

$$c[a, b] = m_1[a] + m_2[b] + \log(E_1 E_2^\top)[a, b].$$

The  $E_1 E_2^\top$  is the GEMM; the per-row max subtraction is the log-sum-exp stabilisation, so it is exact in exact arithmetic and safe in floating point.  $s_1, s_2$  are per-row log-sum-exps (cheap); everything after  $c$  is elementwise over the  $S_1 \times S_2$  grid.

### Value

a data.frame, one row per signal pair, with columns signal1, signal2, nsnps, PP0..PP4, and (when trim) overlap and keep.

statgen\_gwas\_lin

*Univariate linear-regression GWAS over an fmalloc genotype tensor***Description**

Regresses a continuous phenotype on every variant in a bed or dosage `Rfmalloc::fmalloc_tensor`, one variant at a time, adjusting for an optional covariate design. This is the fused form `bigstatsr::big_univLinReg()` uses (Prive et al. 2018, [doi:10.1093/bioinformatics/bty185](https://doi.org/10.1093/bioinformatics/bty185)), not a loop of per-variant `lm()` calls: the phenotype and every genotype column are residualized on the covariate design `[1, covar]` once (the Frisch-Waugh-Lovell theorem), and every variant's slope, standard error, t-statistic, and p-value then follow from a handful of crossprod reductions shared across all variants.

**Usage**

```
statgen_gwas_lin(genotypes, y, covar = NULL, block_size = NULL)
```

**Arguments**

|            |                                                                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| genotypes  | A "bed" or "dosage" <code>Rfmalloc::fmalloc_tensor</code> (samples x variants; see <code>Rfmalloc::fmalloc_bed()</code> / <code>Rfmalloc::fmalloc_dosage()</code> ).  |
| y          | A numeric phenotype vector, one value per sample ( <code>nrow(genotypes)</code> ).                                                                                    |
| covar      | NULL (the default), or a numeric vector/matrix/data frame of covariates, one row per sample. An intercept column is prepended automatically; do not include one.      |
| block_size | Number of variant columns decoded at once. NULL chooses a block holding at most roughly 64 MiB of decoded doubles. This affects memory use and speed, not the result. |

**Details**

Missing genotype calls are mean-imputed per variant before regression, matching the convention `Rfmalloc::fmalloc_bed_standardize()` and `Rfmalloc::fmalloc_dosage_standardize()` use to bake mean-imputation into their codecs' own decode. `y` and `covar` must not contain missing values: drop or impute those samples before calling this function.

**Value**

A data frame with one row per variant (in tensor column order) and columns `beta` (the regression slope on the variant), `se` (its standard error), `t` (`beta / se`), `p` (the two-sided p-value from the `t` distribution on `n - rank([1, covar]) - 1` residual degrees of freedom), and `n` (samples used; currently the same for every variant, since missing genotypes are imputed rather than dropped).

**See Also**

`Rfmalloc::fmalloc_bed()`, `Rfmalloc::fmalloc_dosage()`

## Examples

```
rt <- Rfmalloc::open_fmalloc(tempfile(fileext = ".bin"), mode = "scratch", size_gb = 0.1)
set.seed(1)
n <- 200L
m <- 10L
g <- matrix(sample(0:2, n * m, replace = TRUE), nrow = n, ncol = m)
storage.mode(g) <- "integer"
y <- g[, 1L] * 0.5 + rnorm(n)
tn <- Rfmalloc::fmalloc_bed(g, runtime = rt)
res <- statgen_gwas_lin(tn, y)
head(res)
Rfmalloc::cleanup_fmalloc(rt)
```

---

statgen\_ldpred2

*LDpred2: Bayesian polygenic scores over a banded LD matrix*


---

## Description

LDpred2 (Prive, Arbel & Vilhjalmsón 2020, [doi:10.1093/bioinformatics/btaa1029](https://doi.org/10.1093/bioinformatics/btaa1029)) reweights GWAS summary statistics using an LD matrix and a point-normal (spike-and-slab) prior on the effect sizes. These functions are reimplemented clean-room from the paper and run over an [Rfmalloc::fmalloc\\_ld](#) banded LD store (e.g. from [statgen\\_snp\\_cor\(\)](#)): they read the LD matrix one column's neighbour run at a time and never touch the genotypes.

## Usage

```
statgen_ldpred2_inf(corr, df_beta, h2)
```

```
statgen_ldpred2(
  corr,
  df_beta,
  h2,
  p,
  sparse = FALSE,
  burn_in = 50L,
  num_iter = 100L
)
```

```
statgen_ldpred2_auto(
  corr,
  df_beta,
  h2_init,
  p_init = 0.1,
  burn_in = 500L,
  num_iter = 200L,
  p_bounds = c(1e-05, 1)
)
```



```

m <- 20L
## a simple tridiagonal LD matrix
i <- j <- x <- numeric(0)
for (col in seq_len(m)) {
  rows <- max(1L, col - 1L):min(m, col + 1L)
  i <- c(i, rows); j <- c(j, rep(col, length(rows)))
  x <- c(x, ifelse(rows == col, 1, 0.3))
}
corr <- Rfmalloc::fmalloc_ld(i, j, x, n_variants = m, bits = 16L, runtime = rt)
df_beta <- data.frame(beta = rnorm(m, sd = 0.02), beta_se = rep(0.01, m),
  n_eff = rep(10000, m))
beta_inf <- statgen_ldpred2_inf(corr, df_beta, h2 = 0.1)
head(beta_inf)
Rfmalloc::cleanup_fmalloc(rt)

```

statgen\_pca

*Fast principal-component analysis by PCAone randomized SVD*

## Description

Computes the top-k singular triplet of a matrix with the randomized SVD of Li (2023, [doi:10.1101/2022.05.25.493261](https://doi.org/10.1101/2022.05.25.493261)), vendored verbatim from 'PCAone' (src/pcaone/, GPL-3) and driven here over an in-memory matrix. This is the "be the pointer, don't reimplement" path: the RSVD workhorse (RsvdOpData::computeUSV) is PCAone's own block-wise power-iteration code, not a reimplementation, so results track PCAone/pcaone bit-for-bit given the same seed and parameters.

## Usage

```

statgen_pca(
  X,
  k,
  p = 7L,
  s = 10L,
  center = FALSE,
  scale = FALSE,
  tol = 1e-04,
  seed = 112L,
  block_size = NULL
)

```

## Arguments

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X | A numeric matrix (samples x variants), or anything <code>base::as.matrix()</code> turns into one; or a two-dimensional bed/dosage <code>Rfmalloc::fmalloc_tensor</code> (samples x variants), which is decomposed <i>out of core</i> : the tensor is streamed one variant-column block at a time and the dense matrix is never materialized. To get a standardized PCA of a tensor, standardize the tensor itself first ( <code>Rfmalloc::fmalloc_bed_standardize()</code> / <code>Rfmalloc::fmalloc_dosage_standardize()</code> ) - standardization is a property |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|               |                                                                                                                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | of the tensor, fused into its decode, so center/scale do not apply to a tensor input.                                                                                                                                 |
| k             | Number of principal components to return (a positive integer, $k \geq 1$ ).                                                                                                                                           |
| p             | Number of power iterations (default 7). More iterations sharpen the top-k subspace at the cost of extra passes over X.                                                                                                |
| s             | Oversampling size (default 10): the random test matrix has $k + s$ columns, and $k + s \leq \min(\dim(X))$ is required.                                                                                               |
| center, scale | Logical, whether to centre / scale the columns of X before decomposing (as <code>base::scale()</code> does). Defaults FALSE (a raw SVD of X, matching <code>base::svd()</code> ).                                     |
| tol           | Subspace-convergence tolerance for the power-iteration loop (default $1e-4$ ); the iteration stops early once successive estimates of the top-k subspace agree to tol.                                                |
| seed          | Integer seed for the Gaussian random test matrix (default 112, PCAone's own default), so runs are reproducible.                                                                                                       |
| block_size    | For a tensor input, the number of variant columns decoded per streamed block (bounds the resident memory). NULL (default) picks a size that splits the variants into a handful of blocks. Ignored for a matrix input. |

## Details

For a matrix X (samples x variants) the result satisfies  $X \sim u \% \% \text{diag}(d) \% \% t(v)$ , matching the convention of `base::svd()`: d are the top-k singular values, u (samples x k) the left singular vectors, v (variants x k) the right singular vectors. With center/scale, the decomposition is of the centred/scaled matrix, so d relates to `stats::prcomp()`'s sdev by  $d = \text{sdev} * \sqrt{\text{nrow}(X) - 1}$  and v is its rotation.

Randomized SVD is exact only in the limit; accuracy on the top k components improves with the number of power iterations p and the oversampling s, and is excellent when the spectrum decays (as genotype data's does).  $k + s$  must not exceed  $\min(\text{nrow}(X), \text{ncol}(X))$ .

## Value

A list with elements d (length-k singular values), u ( $\text{nrow}(X) \times k$  left singular vectors) and v ( $\text{ncol}(X) \times k$  right singular vectors).

## See Also

`base::svd()`, `stats::prcomp()`

## Examples

```
set.seed(1)
## a matrix with a decaying spectrum, where RSVD is near-exact
U <- qr.Q(qr(matrix(rnorm(200 * 10), 200)))
V <- qr.Q(qr(matrix(rnorm(60 * 10), 60)))
X <- U \% \% diag(5 * 0.6^(0:9)) \% \% t(V)
pc <- statgen_pca(X, k = 3)
pc$d
svd(X)$d[1:3]
```

statgen\_snp\_cor

*Windowed LD (correlation) matrix from a genotype tensor*

## Description

Computes the (Pearson) correlations between nearby variants of a bed or dosage [Rfmalloc::fmalloc\\_tensor](#) and stores them as a compressed, banded [Rfmalloc::fmalloc\\_ld](#) LD matrix. This is the fused form `bigsnpr::snp_cor()` uses (Prive et al. 2018, [doi:10.1093/bioinformatics/bty185](https://doi.org/10.1093/bioinformatics/bty185)): the genotype tensor is streamed one variant column at a time through [Rfmalloc::fmalloc\\_tensor](#)'s decode (never materializing the dense samples x variants matrix), each column is mean-imputed, centred and normalized to unit L2 norm, and the correlation of two nearby variants is then the dot product of their unit columns. Each band column is written into the store as soon as no later variant can add to it, so what stays resident is a sliding window of unit columns and of in-flight band columns, both bounded by the window width rather than by the variant count; the band itself lives in the store's own `fmalloc` storage.

## Usage

```
statgen_snp_cor(genotypes, size, thr_r2 = 0, infos_pos = NULL, bits = 8L)
```

## Arguments

|                        |                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>genotypes</code> | A "bed" or "dosage" <a href="#">Rfmalloc::fmalloc_tensor</a> (samples x variants), the variants position-sorted.                                                                                                                                                                                                                                                                                         |
| <code>size</code>      | Window half-width. With <code>infos_pos = NULL</code> (the default) this is a window in <b>number of variants</b> each side (variant <code>j</code> is correlated with variants <code>j - size .. j + size</code> ). When <code>infos_pos</code> is supplied, it is a window in <b>kilobases</b> of physical distance (bigsnpr's convention: the window in position units is <code>size * 1000</code> ). |
| <code>thr_r2</code>    | Squared-correlation threshold: correlations with $r^2 < \text{thr\_r2}$ are set to 0 within the band. Default 0 (keep every in-window correlation).                                                                                                                                                                                                                                                      |
| <code>infos_pos</code> | NULL (window by variant index) or a numeric vector of physical positions, one per variant, <b>sorted</b> (window by genomic distance).                                                                                                                                                                                                                                                                   |
| <code>bits</code>      | Quantization width for the stored correlations, 8 (int8, the default; resolution $\sim 1/127$ ) or 16 (int16; resolution $\sim 3e-5$ ).                                                                                                                                                                                                                                                                  |

## Details

Because the variants are assumed position-sorted, each variant's in-window neighbours are a contiguous index range, so the full symmetric band is kept per column with no explicit neighbour indices (see [Rfmalloc::fmalloc\\_ld](#)). The diagonal is 1. The result is storage-agnostic: read it with [Rfmalloc::ld\\_col\(\)](#) / [Rfmalloc::ld\\_pair\(\)](#), or hand it to `LDpred2`.

Missing genotype calls are mean-imputed per variant before the correlation (matching [Rfmalloc::fmalloc\\_bed\\_standardi](#)). On complete data this is the same estimator as `bigsnpr::snp_cor()`; with missingness the two differ (bigsnpr deletes pairwise-incomplete samples instead of imputing).

**Value**

An `Rfmalloc::fmalloc_ld` banded LD matrix (`ncol(genotypes)` x `ncol(genotypes)`).

**See Also**

`Rfmalloc::fmalloc_ld`, `Rfmalloc::ld_col()`, `Rfmalloc::ld_pair()`, `statgen_ldpred2_inf()`

**Examples**

```
rt <- Rfmalloc::open_fmmalloc(tempfile(fileext = ".bin"), mode = "scratch",
                              size_gb = 0.1)

set.seed(1)
n <- 100L; m <- 20L
g <- matrix(sample(0:2, n * m, replace = TRUE), nrow = n, ncol = m)
storage.mode(g) <- "integer"
tn <- Rfmalloc::fmalloc_bed(g, runtime = rt)
corr <- statgen_snp_cor(tn, size = 3)
Rfmalloc::ld_pair(corr, 1, 2)
Rfmalloc::cleanup_fmmalloc(rt)
```

# Index

`base::as.matrix()`, [7](#)  
`base::scale()`, [8](#)  
`base::svd()`, [8](#)  
  
`Rfmalloc::fmalloc_bed()`, [4](#)  
`Rfmalloc::fmalloc_bed_standardize()`, [4](#),  
[7](#), [9](#)  
`Rfmalloc::fmalloc_dosage()`, [4](#)  
`Rfmalloc::fmalloc_dosage_standardize()`,  
[4](#), [7](#)  
`Rfmalloc::fmalloc_ld`, [5](#), [6](#), [9](#), [10](#)  
`Rfmalloc::fmalloc_tensor`, [4](#), [7](#), [9](#)  
`Rfmalloc::ld_col()`, [9](#), [10](#)  
`Rfmalloc::ld_pair()`, [9](#), [10](#)  
  
`statgen_coloc_bf`, [2](#)  
`statgen_gwas_lin`, [4](#)  
`statgen_ldpred2`, [5](#)  
`statgen_ldpred2_auto` (`statgen_ldpred2`),  
[5](#)  
`statgen_ldpred2_inf` (`statgen_ldpred2`), [5](#)  
`statgen_ldpred2_inf()`, [10](#)  
`statgen_pca`, [7](#)  
`statgen_snp_cor`, [9](#)  
`statgen_snp_cor()`, [5](#), [6](#)  
`stats::prcomp()`, [8](#)