Package: Rpgen (via r-universe)

July 10, 2026

Title Vendored 'PLINK 2' pgenlib with a C-Callable Genotype Reader API

Version 0.1.0

Description Vendors the read subset of 'PLINK 2's pgenlib library (<https://github.com/chrchang/plink-ng>), the same subset the CRAN package 'pgenlibr' vendors and builds, as a static library and exposes it through 'R_RegisterCCallable' C-callable entry points. Also vendors plink2's own import closure ('VcfToPgen', 'BcfToPgen', 'OxBgenToPgen', 'OxGenToPgen', 'OxHapslegendToPgen', 'Plink1DosageToPgen') so a VCF, BCF, BGEN, Oxford .gen, Oxford .haps/.legend, or legacy PLINK 1 --import-dosage file can be converted to a '.pgen' and read back through this package's own reader, without a separate 'htslib'-based parser. This is a carrier package: it has no high-level R modeling API of its own beyond a milestone smoke test that opens a '.pgen' file and reports its sample and variant counts. Other R packages link to it ('LinkingTo') to read 'PLINK 2' genotype files natively from their own C or C++ code, which 'pgenlibr' does not expose, since it only offers an R-level interface.

License GPL-3

Encoding UTF-8

Depends R (>= 4.4.0), Rfmalloc

Suggests tinytest

Remotes sounkou-bioinfo/Rfmalloc/packages/Rfmalloc

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/sounkou-bioinfo/Rfmalloc, https://sounkou-bioinfo.github.io/Rfmalloc/Rpgen/

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

Config/pak/sysreqs make

Repository https://sounkou-bioinfo.r-universe.dev

Date/Publication 2026-07-10 06:12:39 UTC

RemoteUrl https://github.com/sounkou-bioinfo/Rfmalloc

RemoteRef HEAD

RemoteSha 921d01f6786a2db38d90a2629c39f56a244b7875

RemoteSubdir packages/Rpgen

Contents

`rpgen_bed` *Read a .pgen or PLINK 1 .bed fileset into an Rfmalloc bed tensor*

Description

Reads hardcalls and packs them into fmalloc-backed, 2-bit `.bed` storage with `Rfmalloc::fmalloc_bed()`. `path` selects the reader: a path ending in `.bed` is read with `rpgen_read_bed_hardcalls()` (PLINK 1, counts from the companion `.bim`/`.fam`); anything else is read with `rpgen_read_hardcalls()` (PLINK 2 `.pgen`). The genotype matrix itself is materialized once, in R, as the integer matrix pgenlib decodes to; the fmalloc tensor it packs into is what downstream matrix products (a genotype PCA, a GRM) stream out-of-core, so the genotypes make one trip through an ordinary R matrix on the way into fmalloc storage, never a second one back out as doubles.

Usage

```
rpgen_bed(path, pvar = NULL, bim = NULL, fam = NULL, runtime = NULL)
```

Arguments

path	Path to a .pgen file, or a PLINK 1 .bed file.
pvar	Path to the companion .pvar/.pvar.zst file; see rpgen_read_hardcalls(). Only used when path is a .pgen. Defaults to NULL.
bim, fam	Paths to the companion .bim/.fam files; see rpgen_read_bed_hardcalls(). Only used when path is a .bed. Default to NULL (inferred from path).
runtime	Runtime handle from Rfmalloc::open_fmalloc(); defaults to the runtime established by Rfmalloc::init_fmalloc().

Value

An fmalloc_tensor of dtype "bed", n_sample x n_variant.

See Also

rpgen_dosage(), rpgen_read_hardcalls(), rpgen_read_bed_hardcalls()

Examples

```
pgen <- system.file("extdata", "chr21_phase3_start.pgen", package = "Rpgen")
rt <- Rfmalloc::open_fmalloc(tempfile(), size_gb = 0.5)
tn <- rpgen_bed(pgen, runtime = rt)
dim(tn)
Rfmalloc::cleanup_fmalloc(rt)
```

---

rpgen_bed_info *Report a PLINK 1 .bed fileset's sample and variant counts*

Description

A PLINK 1 .bed file carries no header of its own (unlike a .pgen's - see rpgen_info()): its sample and variant counts come from the companion .fam (one line per sample) and .bim (one line per variant) instead. This reads exactly those two line counts, with no pgenlib involvement.

Usage

```
rpgen_bed_info(bed)
```

Arguments

bed	Path to a .bed file. The companion .bim/.fam are found by swapping the .bed extension.

Value

A list with n_sample and n_variant, both integers.

See Also

rpgen_info(), rpgen_read_bed_hardcalls()

---

rpgen_dosage *Read a .pgen file into an Rfmalloc dosage tensor*

Description

Reads dosages from a PLINK 2 .pgen file with rpgen_read_dosages() and packs them into fmalloc-backed, 1-byte fixed-point storage with Rfmalloc::fmalloc_dosage(). As with rpgen_bed(), the genotype matrix is materialized once, in R, as the numeric matrix pgenlib decodes to; the fmalloc tensor it packs into is what a downstream standardized product (via Rfmalloc::fmalloc_dosage_standardize()) streams out-of-core.

Usage

```
rpgen_dosage(path, pvar = NULL, runtime = NULL)
```

Arguments

path	Path to a .pgen file, or a PLINK 1 .bed file.
pvar	Path to the companion .pvar/.pvar.zst file; see rpgen_read_hardcalls(). Only used when path is a .pgen. Defaults to NULL.
runtime	Runtime handle from Rfmalloc::open_fmalloc(); defaults to the runtime established by Rfmalloc::init_fmalloc().

Value

An fmalloc_tensor of dtype "dosage", n_sample x n_variant.

See Also

rpgen_bed(), rpgen_read_dosages(), Rfmalloc::fmalloc_dosage_standardize()

Examples

```
pgen <- system.file("extdata", "chr21_phase3_start.pgen", package = "Rpgen")
rt <- Rfmalloc::open_fmalloc(tempfile(), size_gb = 0.5)
tn <- rpgen_dosage(pgen, runtime = rt)
dim(tn)
Rfmalloc::cleanup_fmalloc(rt)
```

`rpgen_import_bcf` *Convert a BCF to a .pgen using plink2's own importer*

Description

Converts `bcf` to a `.pgen` by calling plink2's own `BcfToPgen()` importer

- BCF's binary-sibling counterpart to `rpgen_import_vcf()`'s `VcfToPgen()`, living in the same vendored closure - with the defaults a plain `plink2 --bcf <bcf> --make-pgen` (no other flags) would use.

Usage

```
rpgen_import_bcf(bcf, out = tempfile(fileext = ".pgen"))
```

Arguments

`bcf`	Path to a BCF file.
`out`	Path to the `.pgen` to produce; must end in `.pgen`. Defaults to a fresh `tempfile()`.

Details

The produced `.pgen` can be read back with any of Rpgen's existing `.pgen` readers, exactly as `rpgen_import_vcf()`'s can.

Value

`out`, invisibly on success; an R error is raised on failure.

See Also

`rpgen_import_vcf()`, `rpgen_read_hardcalls()`

Examples

```
vcf <- system.file("extdata", "tiny.vcf", package = "Rpgen")
if (nzchar(vcf) && nzchar(Sys.which("bcftools"))) {
    bcf <- tempfile(fileext = ".bcf")
    system2("bcftools", c("view", shQuote(vcf), "-Ob", "-o", shQuote(bcf)))
    pgen <- rpgen_import_bcf(bcf)
    rpgen_info(pgen)$n_sample
    unlink(c(bcf, pgen, sub("\\.pgen$", ".pvar", pgen), sub("\\.pgen$", ".psam", pgen)))
}
```

rpgen_import_bed *Convert a VCF straight into an Rfmalloc bed tensor*

Description

Convenience wrapper chaining `rpgen_import_vcf()` and `rpgen_bed()`: the VCF is imported to a (by default temporary) `.pgen` via plink2's own importer, immediately read back through Rpgen's existing `.pgen` reader, and packed into fmalloc-backed 2-bit `.bed` storage. The intermediate `.pgen`/`.pvar`/`.psam` files are removed afterward unless `keep = TRUE`.

Usage

```
rpgen_import_bed(
  vcf,
  out = tempfile(fileext = ".pgen"),
  keep = FALSE,
  runtime = NULL
)
```

Arguments

`vcf`	Path to a VCF file; see `rpgen_import_vcf()`.
`out`	Path to the intermediate `.pgen`; see `rpgen_import_vcf()`. Defaults to a fresh `tempfile()`.
`keep`	Keep the intermediate `.pgen`/`.pvar`/`.psam` files instead of deleting them after the read. Defaults to `FALSE`.
`runtime`	Runtime handle from `Rfmalloc::open_fmalloc()`; see `rpgen_bed()`. Defaults to the runtime established by `Rfmalloc::init_fmalloc()`.

Value

An `fmalloc_tensor` of dtype `"bed"`, `n_sample x n_variant`; see `rpgen_bed()`.

See Also

`rpgen_import_vcf()`, `rpgen_bed()`

Examples

```
vcf <- system.file("extdata", "tiny.vcf", package = "Rpgen")
if (nzchar(vcf)) {
    rt <- Rfmalloc::open_fmalloc(tempfile(), size_gb = 0.5)
    tn <- rpgen_import_bed(vcf, runtime = rt)
    dim(tn)
    Rfmalloc::cleanup_fmalloc(rt)
}
```

rpgen_import_bgen *Convert a BGEN file to a .pgen using plink2's own importer*

Description

Converts bgen (any of v1.1/v1.2/v1.3) to a .pgen by calling plink2's own OxBgenToPgen() importer, with the defaults a plain plink2 --bgen <bgen> --sample <sample> --make-pgen (or, if sample is NULL, plink2 --bgen <bgen> --make-pgen alone) would use. Unlike rpgen_import_gen()'s .gen, a BGEN v1.2/v1.3 file may carry its own sample identifier block, so sample may be omitted when the file has one

- plink2's own importer raises a clear error if it does not.

Usage

```
rpgen_import_bgen(bgen, sample = NULL, out = tempfile(fileext = ".pgen"))
```

Arguments

bgen	Path to a BGEN file (v1.1, v1.2, or v1.3).
sample	Path to a companion .sample file, or NULL (the default) to use the BGEN's own embedded sample identifiers, if present.
out	Path to the .pgen to produce; must end in .pgen. Defaults to a fresh tempfile().

Details

The produced .pgen can be read back with any of Rpgen's existing .pgen readers, exactly as rpgen_import_vcf()'s can.

Value

out, invisibly on success; an R error is raised on failure.

See Also

rpgen_import_gen(), rpgen_import_haps()

Examples

```
bgen <- system.file("extdata", "tiny.bgen", package = "Rpgen")
samp <- system.file("extdata", "tiny.sample", package = "Rpgen")
if (nzchar(bgen) && nzchar(samp)) {
    pgen <- rpgen_import_bgen(bgen, sample = samp)
    rpgen_info(pgen)$n_sample
    unlink(c(pgen, sub("\\.pgen$", ".pvar", pgen), sub("\\.pgen$", ".psam", pgen)))
}
```

rpgen_import_gen *Convert an Oxford-format .gen + .sample to a .pgen using plink2's own importer*

Description

Converts the Oxford-format gen/sample pair to a .pgen by calling plink2's own OxGenToPgen() importer, with the defaults a plain plink2 --gen <gen> --sample <sample> --make-pgen would use. Unlike rpgen_import_vcf()'s VCF/BCF, a .gen file carries no sample IDs or pedigree of its own, so sample is required, not optional.

Usage

```
rpgen_import_gen(gen, sample, out = tempfile(fileext = ".pgen"))
```

Arguments

gen	Path to an Oxford-format .gen file (original 5-column layout, with a leading chromosome column).
sample	Path to the companion .sample file. Required.
out	Path to the .pgen to produce; must end in .pgen. Defaults to a fresh tempfile().

Details

gen's rows must carry an explicit leading chromosome column (plink2's original 5-column .gen layout: chr id pos a1 a2 <probabilities>...); this function does not expose an --oxford-single-chr equivalent.

The produced .pgen can be read back with any of Rpgen's existing .pgen readers, exactly as rpgen_import_vcf()'s can.

Value

out, invisibly on success; an R error is raised on failure.

See Also

rpgen_import_bgen(), rpgen_import_haps()

`rpgen_import_haps` *Convert Oxford-format phased haplotypes (.haps/.legend/.sample) to a .pgen*

Description

Converts the Oxford-format haps/legend/sample triple to a `.pgen` by calling plink2's own `OxHapslegendToPgen()` importer, with the defaults a plain `plink2 --haps <haps> --legend <legend> <chr> --sample <sample> --make-pgen` would use.

Usage

```
rpgen_import_haps(haps, legend, sample, chr, out = tempfile(fileext = ".pgen"))
```

Arguments

`haps`	Path to a `.haps` file.
`legend`	Path to the companion `.legend` file.
`sample`	Path to the companion `.sample` file.
`chr`	Chromosome code for every variant in `legend` (e.g. `"1"`). Required; see this function's Details.
`out`	Path to the `.pgen` to produce; must end in `.pgen`. Defaults to a fresh `tempfile()`.

Details

A `.haps`/`.legend` pair encodes *phased* haplotypes, and the produced `.pgen` faithfully carries that phase - but Rpgen's current readers (`rpgen_read_hardcalls()`, `rpgen_read_dosages()`) only read the collapsed, unphased hardcall/dosage view; reading phase back is a future milestone. The genotypes themselves (0/1/2 copies of the alternate allele) still round-trip correctly through the existing readers - only which haplotype carries which allele is not yet recoverable.

`chr` is required: the classic IMPUTE2 `.legend` format (`id position a0 a1`, no chromosome column) does not carry its own chromosome, so plink2 itself requires one whenever `--legend` is used (confirmed via `plink2 --help legend`).

Value

`out`, invisibly on success; an R error is raised on failure.

See Also

`rpgen_import_gen()`, `rpgen_import_bgen()`, `rpgen_read_hardcalls()`

`rpgen_import_plink1_dosage` *Convert a legacy PLINK 1 –import-dosage file to a .pgen*

Description

Converts the legacy PLINK 1.x `--import-dosage` text format (`dosage`, plus companion `fam`/`map` files) to a `.pgen` by calling plink2's own `Plink1DosageToPgen()` importer, with the defaults a plain `plink2 --import-dosage <dosage> --fam <fam> --map <map> --make-pgen` (no `--import-dosage` modifiers) would use - in particular, the per-sample column format (a single dosage value, or a double/triple-probability layout) is auto-inferred from `dosage`'s own column count, the same as the plain command would do.

Usage

```
rpgen_import_plink1_dosage(dosage, fam, map, out = tempfile(fileext = ".pgen"))
```

Arguments

`dosage`	Path to a PLINK 1 `--import-dosage`-format text file (a header line of space-separated `FID IID` pairs, then one row per variant of `ID A1 A2 <per-sample value(s)>`).
`fam`	Path to the companion `.fam` file.
`map`	Path to the companion `.map` file.
`out`	Path to the `.pgen` to produce; must end in `.pgen`. Defaults to a fresh `tempfile()`.

Details

The produced `.pgen` can be read back with any of Rpgen's existing `.pgen` readers, exactly as `rpgen_import_vcf()`'s can.

Value

`out`, invisibly on success; an R error is raised on failure.

See Also

`rpgen_import_vcf()`, `rpgen_read_dosages()`

rpgen_import_vcf *Convert a VCF to a .pgen using plink2's own importer*

Description

Converts `vcf` to a `.pgen` by calling plink2's own `VcfToPgen()` importer (vendored by `tools/vendor-plink2-import/`, see its `PROVENANCE.md`) with the defaults a plain `plink2 --vcf <vcf> --make-pgen` (no other flags) would use - no argv parsing happens; every default matches what plink2.cc itself computes for that combination (see `src/rpgen_import.cpp`'s comments for exactly which). The design choice this embodies: reuse plink2's own import code rather than a from-scratch `htslib`-based VCF reader, because the identical vendored closure also covers BCF/BGEN/ Oxford for a later milestone.

Usage

```
rpgen_import_vcf(vcf, out = tempfile(fileext = ".pgen"))
```

Arguments

`vcf`	Path to a VCF file (may be gzip/bgzip-compressed; plink2's own importer detects that from the file's magic bytes, not its extension).
`out`	Path to the `.pgen` to produce; must end in `.pgen`. Defaults to a fresh `tempfile()`. The companion `.pvar`/`.psam` are written next to it, with the same base name and their own conventional extensions.

Details

The produced `.pgen` (plus its companion `.pvar`/`.psam`, written next to it with the same base name) can be read back with any of Rpgen's existing `.pgen` readers - `rpgen_info()`, `rpgen_read_hardcalls()`, `rpgen_read_dosages()`, or `rpgen_bed()`/`rpgen_dosage()` for the `Rfmalloc` tensor surface - since it is an ordinary `.pgen`, not a special format of its own.

Value

`out`, invisibly on success (matching the input, since plink2's importer writes exactly there); an R error is raised on failure, with plink2's own diagnostic already relayed to the R console (see `src/rpgen_import.cpp`'s top comment for why the error path has two halves).

See Also

`rpgen_bed()`, `rpgen_read_hardcalls()`

Examples

```
vcf <- system.file("extdata", "tiny.vcf", package = "Rpgen")
if (nzchar(vcf)) {
    pgen <- rpgen_import_vcf(vcf)
    info <- rpgen_info(pgen)
```

```
    info$n_sample
    unlink(c(pgen, sub("\\.pgen$", ".pvar", pgen), sub("\\.pgen$", ".psam", pgen)))
}
```

`rpgen_info` *Open a .pgen file and report its sample and variant counts*

Description

Opens a PLINK 2 `.pgen` file through Rpgen's vendored pgenlib (`PgfiInitPhase1()` / `PgfiInitPhase2()` / `PgrInit()`), reads its header counts, and closes it again. This is a thin wrapper around the `RC_rpgen_info` `.Call` entry point, itself a thin wrapper around the `Rpgen_open_info` C-callable registered for other packages to link against (see `inst/include/Rpgen.h`).

Usage

```
rpgen_info(path)
```

Arguments

`path`	Path to a `.pgen` file.

Value

A list with `n_sample` and `n_variant`, both integers.

Examples

```
pgen <- system.file("extdata", "chr21_phase3_start.pgen", package = "Rpgen")
rpgen_info(pgen)
```

`rpgen_read_bed_hardcalls` *Read genotypes from a PLINK 1 .bed fileset as a dense R matrix*

Description

Reads every sample and every variant from a PLINK 1 `.bed`/`.bim`/`.fam` fileset through Rpgen's vendored pgenlib. `PgfiInitPhase1()` opens a `.bed` transparently, in the exact same code path a `.pgen` takes (its `vrtypes` simply come back `NULL`) - the one real difference is that a `.bed` has no header to read its sample/variant counts back from, so they are counted from the companion `.fam`/`.bim` first (see `rpgen_bed_info()`) and passed in explicitly. Genotypes are then read via the same `plink2::PgrGet()` `rpgen_read_hardcalls()` uses; a `.bed` is biallelic hardcalls only, so there is no dosage counterpart to this function. This is a thin wrapper around the `RC_rpgen_read_bed_hardcalls` `.Call` entry point, itself a thin wrapper around the `Rpgen_read_bed_hardcalls` C-callable (see `inst/include/Rpgen.h`) - the lower-level counterpart to `rpgen_bed()` for callers who want the plain R matrix rather than an `Rfmalloc` tensor.

Usage

```
rpgen_read_bed_hardcalls(bed, bim = NULL, fam = NULL)
```

Arguments

bed	Path to a .bed file.
bim	Path to the companion .bim file. Defaults to bed with its extension swapped for .bim.
fam	Path to the companion .fam file. Defaults to bed with its extension swapped for .fam.

Value

An integer matrix of 0, 1, 2, or NA hardcall dosages, n_sample x n_variant, samples in rows and variants in columns.

See Also

rpgen_bed(), rpgen_bed_info(), rpgen_read_hardcalls()

---

rpgen_read_hardcalls *Read genotypes from a .pgen file as a dense R matrix*

Description

Reads every sample and every variant from a PLINK 2 .pgen file through Rpgen's vendored pgen-lib, via plink2::PgrGet() (rpgen_read_hardcalls()) or plink2::PgrGetD() (rpgen_read_dosages()). Both return a dense n_sample x n_variant matrix, samples in rows and variants in columns, matching PLINK's own .bed orientation. These are thin wrappers around the RC_rpgen_read_hardcalls/RC_rpgen_read_d .Call entry points, themselves thin wrappers around the Rpgen_read_hardcalls/ Rpgen_read_dosages C-callables (see inst/include/Rpgen.h) - the lower-level counterpart to rpgen_bed()/rpgen_dosage(), for callers who want the plain R matrix rather than an Rfmalloc tensor.

Usage

```
rpgen_read_hardcalls(path, pvar = NULL)

rpgen_read_dosages(path, pvar = NULL)
```

Arguments

path	Path to a .pgen file.
pvar	Path to the companion .pvar/.pvar.zst file. Accepted for API symmetry with a future allele-specific reader; not read yet (see Details) - the collapsed-ALT encoding these functions produce does not need it. Defaults to NULL.

Details

For a multiallelic variant, `plink2::PgrGet()`/`PgrGetD()` collapse every ALT allele into a single non-reference count - the same encoding `pgenlibr::ReadIntList()`/`ReadList()` return by calling the identical pgenlib entry points. Allele identity (which ALT allele is present) is not needed to produce that collapsed encoding, only for an allele-specific read (`plink2::PgrGet1()`/`PgrGet1D()`, not exposed here), which is the only thing pgenlibr's own `.pvar` requirement (`NewPgen(..., pvar = )`) at this fixture's multiallelic-plus-dosage combination actually guards against - so unlike `pgenlibr::NewPgen()`, `rpgen_read_hardcalls()`/ `rpgen_read_dosages()` do not require a `.pvar` for this file at all.

Value

`rpgen_read_hardcalls()` returns an integer matrix of `0`, `1`, `2`, or `NA` hardcall dosages. `rpgen_read_dosages()` returns a numeric matrix of dosages in `[0, 2]`, or `NA` for missing.

See Also

`rpgen_bed()`, `rpgen_dosage()`

Examples

```
pgen <- system.file("extdata", "chr21_phase3_start.pgen", package = "Rpgen")
hc <- rpgen_read_hardcalls(pgen)
dim(hc)
ds <- rpgen_read_dosages(pgen)
dim(ds)
```

Index