

Package: GenomicDataStream (via r-universe)

September 30, 2026

Type Package

Title Read Genomic Data Files (VCF, BCF, BGEN, PGEN, BED, H5AD) into R/Rcpp in Chunks

Version 0.99.28

Date 2026-09-29

Description Read genomic data files (i.e. VCF, BCF, BGEN, PGEN, BED, H5AD, DelayedArray) into R/Rcpp in chunks for analysis with Armadillo / Eigen / Rcpp libraries. Modern datasets are often too big to fit into memory, and many analyses operate on a small chunk features at a time. GenomicDataStream provides interfaces at both the C++ and R levels for out-of-core analysis.

URL <https://github.com/GabrielHoffman/GenomicDataStream>

BugReports <https://github.com/GabrielHoffman/GenomicDataStream/issues>

License Artistic-2.0

Suggests BiocStyle, RUnit, muscat, VariantAnnotation, knitr, rmarkdown

Depends R (>= 4.5.0), methods, stats, utils

Imports Rcpp, beachmat, hdf5, DelayedArray, RcppParallel, HDF5Array, rhdf5filters, SingleCellExperiment, anndataR, stringr, dplyr, h5mread, S4Vectors, withr, magrittr

LinkingTo Rcpp, RcppArmadillo, RcppEigen (>= 0.3.4.0.2), RcppParallel, beachmat (>= 2.21.9), assorthead, BH (>= 1.87.0), vcfpR (>= 0.8.1)

biocViews DataImport, Infrastructure

Encoding UTF-8

SystemRequirements GNU make, Intel TBB, libcurl: libcurl-devel (rpm) or libcurl4-openssl-dev (deb)

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libicu-dev libpng-dev libssl-dev python3 zlib1g-dev

Repository <https://biocstaging.r-universe.dev>
Date/Publication 2026-09-29 20:25:28 UTC
RemoteUrl <https://github.com/BiocStaging/GenomicDataStream>
RemoteRef HEAD
RemoteSha 834e39c3206d892b1023a3cc2f3a514b25b8a18a

Contents

atEndOfStream	2
chopChroms	3
countChunks	4
dashSVD	5
featuresRead	6
GenomicDataStream	7
GenomicDataStream-class	8
getChromRanges	8
getNextChunk	9
getSampleNames	10
getVariantLocations	11
initializeStream	11
isInitialized	12
normPC	13
PCA-class	13
PCAone	14
perfMetric	16
plot,PCA,ANY-method	17
print,GenomicDataStream-method	18
readH5AD	19
rownames,GenomicDataStream-method	20
setChunkSize	20
setRegion	21
show,GenomicDataStream-method	22
standardize_in_place	22
summarizeChunks	23
Index	24

atEndOfStream	<i>Detected if end of stream is reached</i>
---------------	---

Description

Detected if end of stream is reached

Usage

```
atEndOfStream(x)
```

Arguments

x GenomicDataStream

Value

if end of stream has been reached, return TRUE. Else FALSE

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

# loop until break
while (1) {
  # get data chunk
  # data$X matrix with features as columns
  # data$info information about each feature as rows
  dat <- getNextChunk(obj)

  if (atEndOfStream(obj)) break

  print(dat$info)
}
```

chopChroms

Get genomic intervals

Description

Get genomic intervals chopping up regions into smaller chunks

Usage

```
chopChroms(x, nchunks = 10)
```

Arguments

x data.frame with columns chrom, start, end
nchunks number of chunks to create

Value

vector genome intervals

Examples

```
file <- system.file("extdata", "test.bed", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, chunkSize = 5, initialize = TRUE)
df <- getChromRanges(obj)
chopChroms( df )
```

countChunks

Get chunk sizes

Description

Get array of chunk sizes

Usage

```
countChunks(x)
```

Arguments

x GenomicDataStream

Value

array of chunk sizes

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

countChunks(obj)
```

dashSVD

*Dynamic Shifts Based Randomized SVD***Description**

DashSVD for fast randomized SVD

Usage

```
dashSVD(x, k, p = 3, s = 20, rand = 1, byrow = FALSE, verbose = FALSE)
```

```
## S4 method for signature 'ANY'
```

```
dashSVD(x, k, p = 3, s = 20, rand = 1, byrow = FALSE, verbose = FALSE)
```

Arguments

x	data matrix
k	specifies the target rank of the low-rank decomposition. k should satisfy $k \ll \min(m, n)$.
p	number of additional power iterations (by default $p = 7$).
s	oversampling parameter (by default $s = 20$).
rand	randomization method. 1 indicates Gaussian sampling and 2 indicates uniform sampling
byrow	default FALSE. Set to true if accessing rows is faster than accessing columns
verbose	print messages

Value

PCA class storing d, u and v

Author(s)

Zilong Li <zilong.dk@gmail.com>, Gabriel Hoffman

References

Feng, X., Yu, W., Xie, Y. and Tang, J., 2024. Algorithm 1043: Faster randomized svd with dynamic shifts. ACM Transactions on Mathematical Software, 50(2), pp.1-27.

Examples

```
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
X <- hilbert(9)[, 1:6]

dcmp1 <- svd(X)
dcmp2 <- dashSVD(X, k=5)
```

```
# compare singular values
dcmp1$d
dcmp2$d
```

featuresRead*Get number of features read from GenomicDataStream*

Description

Get number of total features read from GenomicDataStream

Usage

```
featuresRead(x)
```

Arguments

x GenomicDataStream

Value

total number of features read from the stream

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

# loop until break
while (1) {
  # get data chunk
  # data$X matrix with features as columns
  # data$info information about each feature as rows
  dat <- getNextChunk(obj)

  if (atEndOfStream(obj)) break

  print(dat$info)
}

featuresRead(obj)
```

GenomicDataStream	<i>GenomicDataStream</i>
-------------------	--------------------------

Description

Read genomic data files (VCF, BCF, BGEN, h5ad) into R/Rcpp in chunks for analysis with Armadillo or Eigen libraries

Interface to GenomicDataStream C++ code

Usage

```
GenomicDataStream(
  file,
  field = "",
  region = "",
  samples = "-",
  MAF = 0,
  minVariance = 0,
  chunkSize = 10000,
  missingToMean = TRUE,
  initialize = FALSE
)
```

Arguments

file	file in VCF/BCF/BGEN/PGEN format with index
field	field of VCF/BCF to read. Ignored for other file types
region	target in the format chr2:1-12345. Multiple regions can be separated by one of ", \n\t", for example "chr2:1-12345, chr3:1000-8000". Setting region to "" includes all variants
samples	string of comma separated sample IDs to extract: "ID1, ID2, ID3". "-" indicates all samples
MAF	minor allele frequency filter applied to variants with max value <= 2. Use NaN to retain all variants
minVariance	minimum variance filter applied to variants with max value > 2
chunkSize	number of variants to return per chunk
missingToMean	if true, set missing values to the mean dosage value. if false, set to NaN
initialize	default FALSE. If TRUE, file info is read from path, otherwise store path until GenomicDataStream is initialized later

Details

Variants are filtered using MAF if the max value is <=2, or minVariance otherwise

Value

object of class GenomicDataStream

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

obj

# loop until break
while (1) {
  # get data chunk
  # data$X matrix with features as columns
  # data$info information about each feature as rows
  dat <- getNextChunk(obj)

  if (atEndOfStream(obj)) break

  print(dat$info)
}
```

GenomicDataStream-class

Interface to genomic data files

Description

Read genomic data files (VCF, BCF, BGEN, h5ad) into R/Rcpp in chunks for analysis with Armadillo or Eigen libraries

getChromRanges

Get ranges for each chromosome

Description

Get max and max postion for each chromosome

Usage

getChromRanges(x)

Arguments

x GenomicDataStream

Value

data.frame of chrom, start, end

Examples

```
file <- system.file("extdata", "test.bed", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, chunkSize = 5, initialize = TRUE)
getChromRanges( obj )
```

getNextChunk	<i>Get data chunk from GenomicDataStream</i>
--------------	--

Description

Get data chunk from GenomicDataStream

Usage

```
getNextChunk(x)
```

Arguments

x GenomicDataStream

Value

get data chunk as list with entries X storing a matrix with features as columns, and info storing information about each feature as rows

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

# loop until break
while (1) {
  # get data chunk
  # data$X matrix with features as columns
  # data$info information about each feature as rows
  dat <- getNextChunk(obj)
```

```
    if (atEndOfStream(obj)) break
  print(dat$info)
}
```

getSampleNames	<i>Get sample names</i>
----------------	-------------------------

Description

BGEN uses sample order from the query, but VCF/BCF/PGEN uses order in file

Usage

```
getSampleNames(x)
```

Arguments

x	GenomicDataStream
---	-------------------

Details

Get sample names in order that the genotypes are extracted

Value

array of string names

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")
obj <- GenomicDataStream(file, "DS", init=TRUE)
getSampleNames(obj)
```

getVariantLocations	<i>Get location of each feature</i>
---------------------	-------------------------------------

Description

Get location of each feature in GenomicDataStream

Usage

```
getVariantLocations(x)
```

Arguments

x	GenomicDataStream
---	-------------------

Value

get data chunk as list with entries X storing a matrix with features as columns, and info storing information about each feature as rows

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize = TRUE)

getVariantLocations(obj)
```

initializeStream	<i>Initialize GenomicDataStream</i>
------------------	-------------------------------------

Description

Read file info from path to initialise stream. If already initialized, return to the beginning of the stream

Usage

```
initializeStream(x, region = NULL)
```

Arguments

x	GenomicDataStream
region	target in the format chr2:1-12345. Multiple regions can be separated by one of ",\n\t", for example "chr2:1-12345, chr3:1000-8000". Setting region to "" includes all variants

Value

initialized GenomicDataStream

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

obj <- GenomicDataStream(file, "DS", chunkSize = 5)

# by default, GenomicDataStream is not initialized
isInitialized(obj)

# initialize
obj <- initializeStream(obj)

isInitialized(obj)
```

isInitialized	<i>Get status of GenomicDataStream</i>
---------------	--

Description

If initialized, return TRUE, else FALSE

Usage

```
isInitialized(x)
```

Arguments

x	GenomicDataStream
---	-------------------

Value

initialization status

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

obj <- GenomicDataStream(file, "DS", chunkSize = 5)

# by default, GenomicDataStream is not initialized
isInitialized(obj)

# initialize
obj <- initializeStream(obj)
```

isInitialized(obj)

normPC	<i>Normalize principal components</i>
--------	---------------------------------------

Description

Modify signs of principal components so diagonal are always positive

Usage

normPC(U)

Arguments

U matrix of principal components

Value

matrix of principal components with positive diagonal values

Examples

```
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
X <- hilbert(9)[, 1:6]
k <- 4

dcmp <- svd(X, k, k)

normPC( dcmp$u )
```

PCA-class	<i>PCA result</i>
-----------	-------------------

Description

PCA result

PCAone

*Window-based Randomized SVD with PCAone***Description**

Window-based Randomized SVD with PCAone

Usage

```

PCAone(
  x,
  k,
  ...,
  p = 7,
  s = 20,
  B = 64,
  threads = 4,
  threads2 = 4,
  scaleAndCenter = TRUE,
  shuffle = TRUE,
  verbose = TRUE
)

## S4 method for signature 'GenomicDataStream'
PCAone(
  x,
  k,
  ...,
  p = 7,
  s = 20,
  B = 64,
  threads = 4,
  threads2 = 1,
  scaleAndCenter = TRUE,
  shuffle = TRUE,
  verbose = TRUE
)

```

Arguments

x	GenomicDataStream, or matrix as matrix, DelayedArray.
k	integer; specifies the target rank of the low-rank decomposition. k should satisfy $k \ll \min(m, n)$.
...	other argument to control streaming
p	integer, optional; number of additional power iterations (by default $p = 7$).

<code>s</code>	integer, optional; oversampling parameter (by default $s = 20$).
<code>B</code>	integer, optional; number of windows (by default $B = 64$).
<code>threads</code>	integer, optional; number of threads (by default $threads = 4$) to read data
<code>threads2</code>	integer, optional; number of threads (by default $threads = 4$), used for linear algebra operations
<code>scaleAndCenter</code>	bool, optional; if TRUE, scale and center features
<code>shuffle</code>	bool, optional; if TRUE (default) shuffle genomic regions, the next chunk is not in LD with the previous chunk
<code>verbose</code>	string, optional; if TRUE (default), print details

Details

PCAone implements the window-based Randomized SVD proposed by Li, et al. (2023).

If `scaleAndCenter`, the data matrix is scaled so that each feature has a mean of zero and a cross-product of 1. In this case the sum of squares of all eigen values equals the number of features (i.e $\sum(d^2) = p$).

Computational time is spent on two steps:

- 1) Reading data from disk and and processing data. Multiple chunks can be read and processed in parallel. This is controlled by setting `threads`
- 2) Updating PCA with current data chunk. Only one chunk can be processed at a time, but linear algebra operations can be parallelized. This is controlled by setting `threads2`

If `x` is a matrix, PCA is performed using rows as features. Note: this is the `_transpose_` of what `svd()` does.

Value

PCAone returns a list containing the following three components:

- d** array_like;
singular values; vector of length (k) .
- u** array_like;
left singular vectors; (m, k) or (m, nu) dimensional array.
- v** array_like;
right singular vectors; (n, k) or (n, nv) dimensional array.

Note

The singular vectors are not unique and only defined up to sign. If a left singular vector has its sign changed, changing the sign of the corresponding right vector gives an equivalent decomposition.

Author(s)

Zilong Li <zilong.dk@gmail.com>, Gabriel Hoffman

References

- Li, Z., Meisner, J., & Albrechtsen, A. (2023). Fast and accurate out-of-core PCA framework for large scale biobank data. *Genome Research*, 33(9), 1599-1608. doi:10.1101/gr.277525.122.

Examples

```
# Example: PCA on genotype data
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

obj <- GenomicDataStream(file, "DS", chunkSize = 3)

res <- PCAone(obj, k=5, threads=1)

res

par(pty="s")
plot(res)
```

perfMetric	<i>Evaluate performance of PC estimates</i>
------------	---

Description

Evaluate performance of PC estimates compared to true PC values

Usage

```
perfMetric(U, U_est, k = ncol(U_est), metric = c("MEV", "minSSE"))
```

Arguments

U	true eigen values
U_est	estimated eigen-values
k	number of PCs to evaluate
metric	evaluate the accuracy of the estimated PCs compared to the true PCs using mean explained variance ("MEV") or minimum of sum of squared errors ("minSSE")

Details

See performance metrics described by Li, et al. (2023)

Value

performance metric

References

- Li, Z., Meisner, J., & Albrechtsen, A. (2023). Fast and accurate out-of-core PCA framework for large scale biobank data. *Genome Research*, 33(9), 1599-1608. doi:10.1101/gr.277525.122.

Examples

```
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
X <- hilbert(9)[, 1:6]
k <- 4

# Compute SVD using two methods
dcmp1 <- svd(X, k, k)
dcmp2 <- dashSVD( X, k=k)

# Mean variance explained is 1
perfMetric(dcmp1$u, dcmp2$u, metric = "MEV")

# minimum of sum of squared errors zero
perfMetric(dcmp1$u, dcmp2$u, metric = "minSSE")
```

plot,PCA,ANY-method *Plot PCAone*

Description

Plot PCAone

Usage

```
## S4 method for signature 'PCA,ANY'
plot(x, ..., main = "scree plot")
```

Arguments

x	PCA object
...	other arguments
main	title

Value

plot

Examples

```
# Example: PCA on genotype data
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

obj <- GenomicDataStream(file, "DS", chunkSize = 3)

res <- PCAone(obj, k=5, threads=1)

plot(res)
```

```
print, GenomicDataStream-method
      Print object
```

Description

Print object

Print object

Usage

```
## S4 method for signature 'GenomicDataStream'
print(x, ...)

## S4 method for signature 'PCA'
print(x, ...)
```

Arguments

x	PCA object
...	other arguments

Value

printed strings

readH5AD	<i>Read H5AD as SingleCellExperiment</i>
----------	--

Description

Read H5AD as SingleCellExperiment where counts is a file-backed DelayedArray

Usage

```
readH5AD(file, layer = NULL, ondisk = TRUE, verbose = FALSE, raw = FALSE)
```

Arguments

file	H5AD file
layer	NULL (the default) or the name of a matrix in the /layers group. By default (i.e. when layer is not specified) returns the central matrix (X).
ondisk	if TRUE (default), only stream count data into memory when needed. If FALSE, read count data into memory now as a sparseMatrix
verbose	print messages
raw	if TRUE, read counts from /raw/X. Cannot be used with layer.

Details

Uses `HDF5Array::H5ADMatrix()` to read counts as a file-backed DelayedArray, and `anndataR::read_h5ad()` to read all other data from H5AD.

Value

SingleCellExperiment

Examples

```
file <- system.file("extdata", "example.h5ad", package = "anndataR")
sce <- readH5AD(file)
```

rownames, GenomicDataStream-method
<i>Get rownames</i>

Description

BGEN uses sample order from the query, but VCF/BCF/PGEN uses order in file

Usage

```
## S4 method for signature 'GenomicDataStream'
rownames(x, do.NULL = TRUE, prefix = "row")
```

Arguments

x	GenomicDataStream
do.NULL	not used
prefix	not used

Details

Get rownames (i.e. sample names) in order that the samples are extracted

Value

array of string names

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")
obj <- GenomicDataStream(file, "DS", init=TRUE)
rownames(obj)
```

setChunkSize	<i>Set Chunk Size</i>
--------------	-----------------------

Description

Set chunk size for existing GenomicDataStream

Usage

```
setChunkSize(x, chunkSize)
```

Arguments

x GenomicDataStream
 chunkSize positive integer

Value

none

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")
obj <- GenomicDataStream(file, "DS", chunkSize = 5, initialize=TRUE)
obj <- setChunkSize(obj, 200)
```

setRegion	<i>Set regions of GenomicDataStream</i>
-----------	---

Description

If GenomicDataStream is already initialized, set new query in C++ backend. Otherwise substitute region values

Usage

```
setRegion(x, region)
```

Arguments

x GenomicDataStream
 region target in the format chr2:1-12345. Multiple regions can be separated by one of ", \n\t", for example "chr2:1-12345, chr3:1000-8000". Setting region to "" includes all variants

Details

Set regions of GenomicDataStream

Value

GenomicDataStream with region set

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

obj <- GenomicDataStream(file, "DS", chunkSize = 5)

# by default, GenomicDataStream is not initialized
setRegion(obj, "1:10000-12000")
```

```
show,GenomicDataStream-method
      Show object
```

Description

Show object
 Show object

Usage

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

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

Arguments

object PCA object

Value

printed strings

```
standardize_in_place    Standardize matrix columns in place
```

Description

Standardize mean and variance of matrix columns in place

Usage

```
standardize_in_place(X, center = TRUE, scale = TRUE)
```

Arguments

X	matrix
center	boolean, TRUE indices center columns
scale	boolean, TRUE indices scale columns

Value

none, matrix is standardized in place

Examples

```
X <- matrix(runif(12), 3, 4)
standardize_in_place(X)

scale(X)
```

summarizeChunks	<i>Get information about chunks</i>
-----------------	-------------------------------------

Description

Read through stream to get size of each chunk, IDs of variants and samples

Usage

```
summarizeChunks(x, threads = 4)
```

Arguments

x	GenomicDataStream
threads	number of threads

Value

list of info about stream

Examples

```
file <- system.file("extdata", "test.vcf.gz", package = "GenomicDataStream")

# initialize
obj <- GenomicDataStream(file, "DS", chunkSize = 3, initialize = TRUE)
summarizeChunks(obj)
```

Index

atEndOfStream, [2](#)

chopChroms, [3](#)

countChunks, [4](#)

dashSVD, [5](#)

dashSVD, ANY-method (dashSVD), [5](#)

featuresRead, [6](#)

GenomicDataStream, [7](#)

GenomicDataStream-class, [8](#)

getChromRanges, [8](#)

getNextChunk, [9](#)

getSampleNames, [10](#)

getVariantLocations, [11](#)

initializeStream, [11](#)

isInitialized, [12](#)

normPC, [13](#)

PCA-class, [13](#)

PCAone, [14](#)

PCAone, GenomicDataStream-method
(PCAone), [14](#)

perfMetric, [16](#)

plot, PCA, ANY-method, [17](#)

plot, PCA-method (plot, PCA, ANY-method),
[17](#)

print, GenomicDataStream, GenomicDataStream-method
(print, GenomicDataStream-method),
[18](#)

print, GenomicDataStream-method, [18](#)

print, PCA-method
(print, GenomicDataStream-method),
[18](#)

readH5AD, [19](#)

rownames, GenomicDataStream-method, [20](#)

setChunkSize, [20](#)

setRegion, [21](#)

show, GenomicDataStream, GenomicDataStream-method
(show, GenomicDataStream-method),
[22](#)

show, GenomicDataStream-method, [22](#)

show, PCA, PCA-method
(show, GenomicDataStream-method),
[22](#)

show, PCA-method
(show, GenomicDataStream-method),
[22](#)

standardize_in_place, [22](#)

summarizeChunks, [23](#)