PCA-seq Package Specification

JL Kirk

May 2, 2015

1 Package Purpose

The goal of this package is to implement the EIGENSTRAT and PCA-seq method of inferring population structure for both rare and common variants. Specific aims are:

- Calculate the GRM using EIGENSTRAT or PCA-seq on any subset of the data (rare or common variants)
- Make use of `SNPRelate` functions as much as possible for convenience
- Use `SNPRelate` syntax as much as possible for user convenience

1.1 Input Data Types

- Genotype data: `SNPGDSFileClass`, SNP GDS file containing all of the relevant genotype data
- Sample IDs: `vector`, samples to use in the analysis (NULL implies use all)
- SNP IDs: `vector`, SNP ids to use in the analysis (NULL implies use all)
- MAF filter: `numeric` or `vector`, if numeric, assumed to be the minimum MAF value; if vector, assumed to give (min, max) maf values. MAF values are in 0-0.5 & if two are given, they should not be equal. The first value should be less than the second value.
- Number of Eigenvectors: `numeric`, the number of eigenvectors & values to return. This is should be less than or equal to the number of subjects.
- Verbose: `logical`, indicates to show all information printed out from the analysis

- Remove Monomorphic SNPs: `logical`, indicates to remove monomorphic SNPs
- Autosome Only: `logical` or `numeric` or `character`, indicates to use the autosomal chromosomes or the named chromosomes
- Method: `string`, indicates which method to use for calculating the GRM

1.2 Output Data Types

The output data type will be an object of the class `snpgdsPCAClass`, which is a list with the following elements:

- Sample IDs: `vector`, the sample ids passed into the function & used to perform the analysis
- SNP IDs: `vector`, the SNP ids passed into the function & used to perform the analysis
- Eigenvalues: `vector`, eigenvalues
- Eigenvectors: `matrix`, the eigenvectors in columns
- Variance Proportion: `vector`, the proportion of the variance explained by each eigenvector
- GRM Trace: `numeric`, the trace of the GRM
- GRM: `matrix`, the GRM
- method: `string`, the method used to find the GRM
- maf: `numeric` or `vector`, the MAF filtering values

2 Wishlist

- `seqPCA`: function that performs PCA on sequences (analogous to `snpgdsPCA` in `SNPRelate`)
- `seqGRM`: function to find the GRM only
- `check.bool`: function to check that the input argument is a `logical`
- `check.ecnt`: function to check that the `eigen.cnt` argument is valid

- `check.maf`: function to check that the `maf` argument is valid
- `check.miss`: function to check that the `missing.rate` argument is valid
- `check.vect`: function to check if the `snp.id` and `samp.id` vectors have length of at least 1 and no duplicates
- `run.grm`: function to compute the appropriate GRM
- `grm.pcaseq`: function to calculate the GRM using PCA-seq
- `grm.eigenstrat`: function to calculate the GRM using EIGENSTRAT
- `get.snps`: function to get the snp ids for one block of snps
- `filter.snps`: function to filter the genotype data based on SNPs
- `filter.monosnps`: function to remove monomorphic SNPs
- `filter.missing`: function to remove SNPs with too much missingness
- `filter.auto`: function to remove sex chromosome SNPs
- `filter.maf`: function to remove SNPs by MAF
- `make.snpgdsPCAClass`: function to construct an object from the `snpPCAClass`

2.1 Functions from Other Packages

- `SNPRelate`
  - `print.snpgdsPCA`
  - `snpgdsPCAcorr`
- `matrixcalc`
  - `matrix.trace`
- `base`
  - `eigen`

2.2 Function Compatibility with `SNPRelate`

- `snpgdsPCALoading`
- `snpgdsPCASampLoading`

3 Function Specification

3.1 `seqPCA`

This function takes in all of the variables specified in Section 1.1 and returns an object of the class specified in Section 1.2. Test cases for this function include:

- A few small cases to check the math
- Various levels of MAF to check
- SNP ID filtering
- Sample ID filtering
- Monomorphic SNP filtering
- Autosomal Filtering
- Try both methods

3.2 `seqGRM`

This function takes in all of the variables specified in Section 1.1 and returns sample ids, SNP ids, GRM trace, and the GRM. Test cases for this function include:

- A few small cases to check the math
- Various levels of MAF to check
- SNP ID filtering
- Sample ID filtering
- Monomorphic SNP filtering
- Autosomal Filtering
- Try both methods

3.3 `check.bool`

This function checks if an input parameter is `logical`. It takes in a `value` of any class, and returns a `logical` or an `error`. Test cases include:

- Results for a logical (both TRUE, FALSE)
- Results for non-logical
- Results for a vector
- Results for NA
- Results for NULL
- Results for NaN

3.4 `check.ecnt`

This function checks if an input parameter is a valid number of eigenvalues/eigenvectors. It takes in a `value` of any class and `sample.id`, and returns a `logial` and a `number` or `alogical` and an `error`. If the `value` is zero, then a warning is returned; if the value is greater than the number of subjects, it is set to the number of subjects and a warning is returned. Test cases include:

- Results for correct input
- Results for non-integer
- Results for negative number
- Results for vector
- Results for number greater than # of subjects
- Results for 0
- Results for NaN
- Results for NA
- Results for NULL

3.5 `check.maf`

This function checks if an input parameter is a valid maf value or range of values. It takes in a `value` of any class, and returns a `logial` or an `error`. Test cases include:

- Results for correct input–1 value
- Results for correct input–2 values
- Results for more than two values
- Results for non-numeric values
- Results for number outside of $[0, 0.5]$
- Results for maf.min $\geq$ maf.max
- Results for NaN
- Results for NA
- Results for NULL

3.6 `check.miss`

This function checks if an input parameter is a valid missingness proportion. It takes in a `value` of any class, and returns a `logial` or an `error`. Test cases include:

- Results for correct input
- Results for more than one value
- Results for non-numeric values
- Results for number outside of $[0, 1]$
- Results for NaN
- Results for NA
- Results for NULL

3.7 `run.grm`

This function calculates the GRM matrix using the appropriate subset of the data and the requested method. It takes in

- `gdsobj`: `SNPGDSFileClass`, a SNP GDS file
- `sample.id`: `vector`, the samples to keep for the analysis
- `snp.id`: `vector`, the SNPs to keep for the anlaysis
- `autosome.only`: `logical`
- `remove.monosnp`: `logical`
- `maf`: `numeric` or `vector`, if numeric, assumed to be the minimum MAF value; if vector, assumed to give (min, max) maf values. MAF values are in 0-0.5 & if two are given, they should not be equal
- `missing.rate`: `numeric`
- `method`: `string`

This function returns:

- grm: `matrix`

Test cases for this function include:

- Results for autosome.only is TRUE
- Results for autosome.only is FALSE
- Results for sample.id removes some samples
- Results for snp.id removes some SNPs
- Results for remove.monosnp is TRUE
- Results for remove.monosnp is FALSE
- Results under various MAF
- Results for missing.rate is given
- Results for both methods

3.8 grm.eigenstrat

This function calculates the GRM in Pritchard et al 2006. It takes a

- genodat: matrix, the genotype data, SNPs in columns
- maf: numeric or vector, the MAF to filter on
- snp.id: vector, the snps to filter out

This function returns:

- grm: matrix

3.9 grm.pcaseq

This function calculates the GRM using PCA-seq. It takes a

- genodat: matrix, the genotype data, SNPs in columns
- maf: numeric or vector, the MAF to filter on
- snp.id" vector, the snps to filter out

This function returns:

- grm: matrix

3.10 get.snps

This function calculates the indices of the ith block of nblock snps. This function takes:

- nblock: numeric, a constant that is the number of SNPs to work with at one time
- i: numeric, an integer that indicates which block the snps are in
- nsnps: numeric, the total number of snps

The function returns a vector of integers of length nblock or less that can be used to subset the snp.id vector.

3.11 filter.snps

This function filters the genotype data based on criteria related to the SNPs, after the data has been subset to the SNPs in snp.id. It takes:

- snp.dat: a matrix of SNP data
- autosome.only: logical
- remove.monosnp: logical
- maf: numeric or vector, if numeric, assumed to be the minimum MAF value; if vector, assumed to give (min, max) maf values. MAF values are in 0-0.5 & if two are given, they should not be equal
- missing.rate: numeric

This function returns a matrix of genotype data.

3.12 filter.monosnps

This function filters out monomorphic SNPs. It takes:

- snp.dat: a matrix of SNP data

This function returns a matrix of genotype data.

3.13 filter.missing

This function filters out SNPs with too many missing values. It takes:

- snp.dat: a matrix of SNP data
- missing.rate: numeric

This function returns a matrix of genotype data.

3.14 filter.auto

This function filters out snps from non-autosomal (sex) chromosomes. It takes:

- snp.dat: a matrix of SNP data

This function returns a matrix of genotype data.

3.15 `filter.maf`

This function filters the genotype data based on MAF. It takes:

- `snp.dat`: a matrix of SNP data
- `maf`: `numeric` or `vector`, if numeric, assumed to be the minimum MAF value; if vector, assumed to give (min, max) maf values. MAF values are in 0-0.5 & if two are given, they should not be equal

This function returns a `matrix` of genotype data.

3.16 `make.snpgdsPCAClass`

Need to figure out how to calculate the variance proportion. This function creates an object of the class `snpgdsPCACLass`. It takes the inputs:

- `eigen.res`: `list` of the eigenvalues and eigenvectors
- `grm`: `matrix`, the GRM
- `sample.id`: `vector`
- `snp.id`: `vector`
- `need.genmat`: `logical`

This function returns an object of the class `snpgdsPCAClass`, which is a list with the fields:

- `sample.id`: `vector`, a list of sample ids used for the calculation
- `snp.id`: `vector`, a list of SNPs used for the calculation
- `eigenval`: `vector`, a list of eigenvalues
- `eigenvect`: `matrix`, a matrix of eigenvectors
- `varprop`: `vector`, a vector of the proportion of the variance explained by each eigenvector
- `TraceXTX`: `numeric`, the trace of the GRM
- `Bayesian`: `logical`, set to FALSE always
- `genmat`: `matrix`, the GRM

Test cases for this function include:

- Correct results for including the GRM
- Correct results for not including the GRM