Principal Component Analysis All in One

Version 0.4.9

Zilong Li[*]

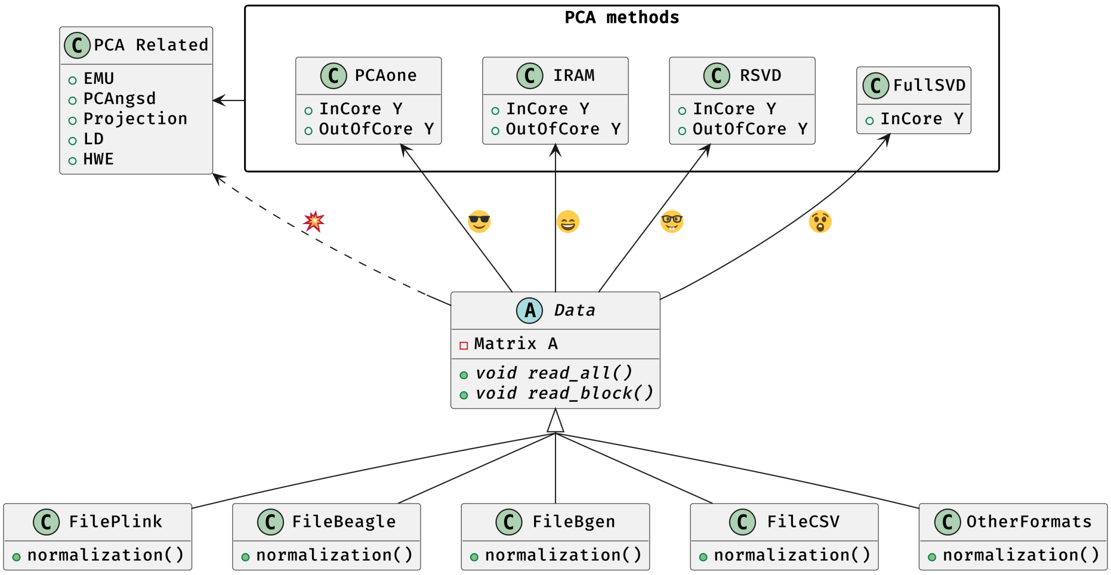
December 26, 2024

Contents

[*] zilong.dk@gmail.com

1 Introduction

PCAone is a fast and memory efficient PCA tool implemented in C++ aiming at providing comprehensive features and algorithms for different scenarios. PCAone implements 3 fast PCA algorithms for finding the top eigenvectors of large datasets, which are Implicitly Restarted Arnoldi Method (IRAM, –svd 0), single pass Randomized SVD but with power iteration scheme (RSVD, –svd 1, Algorithm1 in paper) and **our own RSVD with window based power iteration scheme** (PCAone, –svd 2, Algorithm2 in paper). All have both in-core and out-of-core implementation. Addtionally, full SVD (–svd 3) is supported via in-core mode only. There is also an R package here. PCAone supports multiple different input formats, such as PLINK, BGEN, Beagle genetic data formats and a general comma separated CSV format for other data, such as scRNAs and bulk RNAs. For genetics data, PCAone also implements EMU and PCAngsd algorithm for data with missingness and uncertainty. The PDF manual can be downloaded here.



2 Features

See change log here.

- Has both Implicitly Restarted Arnoldi Method (IRAM) and Randomized SVD (RSVD) with **out-of-core** implementation.
- Implements our new fast window based Randomized SVD algorithm for tera-scale dataset.
- Quite fast with multi-threading support using high performance library MKL or OpenBLAS as backend.
- Supports the PLINK, BGEN, Beagle genetic data formats.
- Supports a general comma separated CSV format for single cell RNA-seq or bulk RNA-seq data compressed by zstd.
- Supports EMU algorithm for scenario with large proportion of missingness.
- Supports PCAngsd algorithm for low coverage sequencing scenario with genotype likelihood as input.
- Novel LD prunning and clumping method that accounts for population structure in the data.

- Projection support for data with missingness.
- HWE test taking population structure into account.

3 Cite the work

- If you use PCAone, please first cite our paper on genome reseach Fast and accurate out-of-core PCA framework for large scale biobank data.
- If using the EMU algorithm, please also cite Large-scale inference of population structure in presence of missingness using PCA.
- If using the PCAngsd algorithm, please also cite Inferring Population Structure and Admixture Proportions in Low-Depth NGS Data.
- If using the ancestry ajusted LD statistics for pruning and clumping, please also cite Measuring linkage disequilibrium and improvement of pruning and clumping in structured populations.

4 Quick start

```
pkg=https://github.com/Zilong-Li/PCAone/releases/latest/download/PCAone-avx2-Linux.zip
wget $pkg
unzip -o PCAone-avx2-Linux.zip
wget http://popgen.dk/zilong/datahub/pca/example.tar.gz
tar -xzf example.tar.gz && rm -f example.tar.gz
# in default calculating top 10 PCs with in-memory mode
./PCAone -b example/plink
R -s -e 'd=read.table("pcaone.eigvecs2", h=F);
plot(d[,1:2+2], col=factor(d[,1]), xlab="PC1", ylab="PC2");
legend("topright", legend=unique(d[,1]), col=1:4, pch = 21, cex=1.2);'
```

You will find these files in current directory.

```
.
 PCAone            # program
 Rplots.pdf        # pca plot
 example           # folder of example data
 pcaone.eigvals    # eigenvalues
 pcaone.eigvecs    # eigenvectors, the PCs you need to plot
 pcaone.eigvecs2   # eigenvectors with header line
 pcaone.log        # log file
```

5 Installation

There are 3 ways to install PCAone.

5.1 Download compiled binary

There are compiled binaries provided for both Linux and Mac platform. Check the releases page to download one. For most modern CPUs and Linux systems, download the one named with avx2.

```
pkg=https://github.com/Zilong-Li/PCAone/releases/latest/download/PCAone-avx2-Linux.zip
wget $pkg || curl -LO $pkg
unzip -o PCAone-avx2-Linux.zip
```

If the server is too old to support avx2 instruction, download the alternative version.

```
pkg=https://github.com/Zilong-Li/PCAone/releases/latest/download/PCAone-x64-Linux.zip
wget $pkg || curl -LO $pkg
unzip -o PCAone-x64-Linux.zip
```

5.2 Via Conda

PCAone is also available from bioconda.

```
conda config --add channels bioconda
conda install pcaone
PCAone --help
```

5.3 Build from source

PCAone can be running on a normal computer/laptop with x86-64 instruction set architecture. PCAone has been tested on both Linux and MacOS system. To build PCAone from the source code, the following dependencies are required:

- GCC/Clang compiler with C++17 support
- GNU make
- zlib

We **recommend** building the software from source with MKL as backend to maximize the performance. For MacOS users, we recommend using llvm by brew install llvm instead of the default clang shipped with MacOS. Check out the mac workflow.

5.3.1 With MKL or OpenBLAS as backend

Build PCAone dynamically with MKL can maximize the performance since the faster threading layer libiomp5 will be linked at runtime. One can obtain the MKL by one of the following option:

- install mkl by conda

```
conda install -c conda-forge -c anaconda -y mkl mkl-include intel-openmp
git clone https://github.com/Zilong-Li/PCAone.git
cd PCAone
# if mkl is installed by conda then use ${CONDA_PREFIX} as mklroot
make -j4 MKLROOT=${CONDA_PREFIX}
./PCAone -h
```

- download mkl from the website

After having mkl installed, find the mkl root path and replace the path below with your own.

```
# if libiomp5 is not in the mklroot path, please link it to $MKLROOT/lib folder
make -j4 MKLROOT=/path/to/mklroot
```

Alternatively, for advanced user, modify variables directly in `Makefile` and run `make` to use MKL or OpenBlas as backend.

5.3.2 Without MKL or OpenBLAS dependency

If you don't want any optimized math library as backend, just run:

```
git clone https://github.com/Zilong-Li/PCAone.git
cd PCAone
make -j4
./PCAone -h
```

If this doesn't work because the server is too outdated, run `make clean && make -j4 AVX=0` instead.

6 Documentation

6.1 Options

Run `PCAone --groff > pcaone.1 && man ./pcaone.1` or `PCAone --help` to read the manual. Here are common options.

```
General options:
  -h, --help                      print all options including hidden advanced options
  -m, --memory arg (=0)           RAM usage in GB unit for out-of-core mode. default is in-core
  ↪  mode
  -n, --threads arg (=12)         the number of threads to be used
  -v, --verbose arg (=1)          verbosity level for logs. any level x includes messages for all
  ↪  levels (1...x). Options are
                                  0: silent, no messages on screen;
                                  1: concise messages to screen;
                                  2: more verbose information;
                                  3: enable debug information.

PCA algorithms:
  -d, --svd arg (=2)              SVD method to be applied. default 2 is recommended for big data.
  ↪  Options are
                                  0: the Implicitly Restarted Arnoldi Method (IRAM);
                                  1: the Yu's single-pass Randomized SVD with power iterations;
                                  2: the accurate window-based Randomized SVD method (PCAone);
                                  3: the full Singular Value Decomposition.
  -k, --pc arg (=10)              top k principal components (PCs) to be calculated
  -C, --scale arg (=0)            do scaling for input file. Options are
                                  0: do just centering;
                                  1: do log transformation eg. log(x+0.01) for RNA-seq data;
                                  2: do count per median log transformation (CPMED) for scRNAs.
  -p, --maxp arg (=40)            maximum number of power iterations for RSVD algorithm.
  -S, --no-shuffle                do not shuffle columns of data for --svd 2 (if not locally
  ↪  correlated).
  --emu                           use EMU algorithm for genotype input with missingness.
  --pcangsd                       use PCAngsd algorithm for genotype likelihood input.

Input options:
  -b, --bfile arg                 prefix of PLINK .bed/.bim/.fam files.
  -B, --binary arg                path of binary file.
  -c, --csv arg                   path of comma seperated CSV file compressed by zstd.
  -g, --bgen arg                  path of BGEN file compressed by gzip/zstd.
  -G, --beagle arg                path of BEAGLE file compressed by gzip.
  -f, --match-bim arg             the .mbim file to be matched, where the 7th column is allele
  ↪  frequency.
  --USV arg                       prefix of PCAone .eigvecs/.eigvals/.loadings/.mbim.

Output options:
  -o, --out arg (=pcaone)         prefix of output files. default [pcaone].
  -V, --printv                    output the right eigenvectors with suffix .loadings.
  -D, --ld                        output a binary matrix for downstream LD related analysis.
  -R, --print-r2                  print LD R2 to *.ld.gz file for pairwise SNPs within a window
  ↪  controlled by --ld-bp.

Misc options:
  --maf arg (=0)                  exclude variants with MAF lower than this value
  --project arg (=0)              project the new samples onto the existing PCs. Options are
                                  0: disabled;
                                  1: by multiplying the loadings with mean imputation for missing
                                  ↪  genotypes;
                                  2: by solving the least squares system Vx=g. skip sites with
                                  ↪  missingness;
```

```
                                3: by Augmentation, Decomposition and Procrusters
                                ↪  transformation.
  --inbreed arg (=0)            compute the inbreeding coefficient accounting for population
  ↪  structure. Options are
                                0: disabled;
                                1: compute per-site inbreeding coefficient and HWE test.
  --ld-r2 arg (=0)              R2 cutoff for LD-based pruning (usually 0.2).
  --ld-bp arg (=0)              physical distance threshold in bases for LD window (usually
  ↪  1000000).
  --ld-stats arg (=0)           statistics to compute LD R2 for pairwise SNPs. Options are
                                0: the ancestry adjusted, i.e. correlation between residuals;
                                1: the standard, i.e. correlation between two alleles.
  --clump arg                   assoc-like file with target variants and pvalues for clumping.
  --clump-names arg (=CHR,BP,P)  column names in assoc-like file for locating chr, pos and
  ↪  pvalue.
  --clump-p1 arg (=0.0001)      significance threshold for index SNPs.
  --clump-p2 arg (=0.01)        secondary significance threshold for clumped SNPs.
  --clump-r2 arg (=0.5)         r2 cutoff for LD-based clumping.
  --clump-bp arg (=250000)      physical distance threshold in bases for clumping.
```

6.2 Which SVD method to use

This depends on your datasets, particularlly the relationship between number of samples (`N`) and the number of variants / features (`M`) and the top PCs (`k`). Here is an overview and the recommendation.

Method	Accuracy	Scenario
IRAM (-d 0)	Very high	`N < 1000`
Window-Based RSVD (-d 2)	Very high	`M > 1,000,000`
RSVD (-d 1)	High	accuracy insensitive
Full SVD (-d 3)	Exact	cost insensitive

6.3 Input formats

PCAone is designed to be extensible to accept many different formats. Currently, PCAone can work with SNP major genetic formats to study population structure. such as PLINK, BGEN and Beagle. Also, PCAone supports a comma delimited CSV format compressed by zstd, which is useful for other datasets requiring specific normalization such as single cell RNAs data.

6.4 Output formats

6.4.1 Eigen vectors

Eigen vectors are saved in file with suffix `.eigvecs`. Each row represents a sample and each col represents a PC.

6.4.2 Eigen values

Eigen values are saved in file with suffix `.eigvals`. Each row represents the eigenvalue of corresponding PC.

6.4.3 Features loadings

Features Loadings are saved in file with suffix `.loadings`. Each row represents a feature and each column represents a corresponding PC. Use `--printv` option to output it.

6.4.4 Variants infomation

A plink-like bim file named with `.mbim` is used to store the variants list with extra infomation. Currently, the `mbim` file has 7 columns with the 7th being the allele frequency. And PCAone only outputs this file whenever it's necessary to downstrean analyses.

6.4.5 LD matrix

The matrix for calculating the ancestry-adjusted LD is saved in a file with suffix `.residuals`, and its associated variants information is stored in `mbim` file. For the binary file, the first 4-bytes stores the number of variants/SNPs, and the second 4-bytes stores the number of samples in the matrix. Then, the rest of the file is a sequence of **M** blocks of **N x 4** bytes each, where **M** is the number of variants and **N** is the number of samples. The first block corresponds to the first marker in the `.mbim` file, etc.

6.4.6 LD R2

The LD R2 for pairwise SNPs within a window can be outputted to a file with suffix `ld.gz` via `--print-r2` option. This file uses the same long format as the one plink used.

6.5 Memory-efficient modes

PCAone has both **in-core** and **out-of-core** mode for 3 different partial SVD algorithms, which are IRAM (`--svd 0`), Yu+Halko RSVD (`--svd 1`) and PCAone window-based RSVD (`--svd 2`). Also, PCAone supports full SVD (`--svd 3`) but with only **in-core** mode. Therefore, there are **7** ways for doing PCA in PCAone. In default PCAone uses **in-core** mode with `--memory 0`, which is the fastest way to do calculation. However, in case the server runs out of memory with `in-core` mode, the user can trigger `out-of-core mode` by specifying the amount of memory using `--memory` option with a value greater than 0.

6.5.1 Run PCAone window-based RSVD method (default) with in-core mode

```
./PCAone --bfile example/plink
```

6.5.2 Run PCAone window-based RSVD method (default) with out-of-core mode

```
./PCAone --bfile example/plink -m 2
```

6.5.3 Run Yu+Halko RSVD method with in-core mode

```
./PCAone --bfile example/plink --svd 1
```

6.5.4 Run Yu+Halko RSVD method with out-of-core mode

```
./PCAone --bfile example/plink --svd 1 -m 2
```

6.5.5 Run IRAM method with in-core mode

```
./PCAone --bfile example/plink --svd 0 -m 2
```

6.5.6 Run IRAM method with out-of-core mode

```
./PCAone --bfile example/plink --svd 0 -m 2
```

6.5.7 Run Full SVD method with in-core mode

```
./PCAone --bfile example/plink --svd 3
```

6.6 Data Normalization

PCAone will automatically apply the standard normalization for genetic data. Additionally, there are 3 different normalization method implemented with `--scale` option.

- 0: do just centering by substracting the mean
- 1: do log transformation (usually for count data, such as bulk RNA-seq data)
- 2: do count per median log transformation (usually for single cell RNA-seq data)

One should choose proper normalization method for specific type of data.

6.7 Projection

Project new samples onto existing PCs is supported with `--project` option. First, we run PCAone on a set of reference samples and output the loadings:

```
PCAone -b ref_samples -k 10 --printv -o ref
```

Then, we need to read in the SNPs loadings from the ref set (`--read-V`) and its scaling factors (`--read-S`) as well as the allele frequencies form the `.mbim` file via `--match-bim`. **Note:** one can use the `--USV` option instead to simplify the usage since v0.4.8 Here is the example command to project new target samples and get the PCs coordinates.

```
PCAone -b new_samples \
       --USV ref \  ## prefix to .eigvecs, .eigvals, .loadings
       --project 2 \  ## check the manual on projection methods
       -o new
```

6.8 HWE accounting for population structure

To test Hardy-Weinberg equilibrium in presence of population structure, we need to work on the so-called individual allele frequencies matrix Π, which can be reconstructed via the output of PCAone, i.e the `.eigvecs,.eigvals,.loadings and .mbim` files, generated by

```
PCAone -b example/plink -k 3 -V -o pcaone
```

Then we apply `--inbreed 1` option to obtain the P value of HWE and inbreeding coefficient per-site. The output file is named with suffix `.hwe`.

```
PCAone -b example/plink \
       --USV pcaone \
       --inbreed 1 \
       -o inbreed
```

6.9 Ancestry-Adjusted LD matrix

LD patterns vary across diverse ancestry and structured groups, and conventional LD statistics, e.g. the implementation in `plink --ld`, failed to model the LD in admixed populations. Thus, we can use the so-called ancestry-adjusted LD statistics to account for population structure in LD. See our paper for more details.

To calculate the ancestry-adjusted LD matrix, we first figure out the number of principal components (`-k/--pc`) that capture population structure. In this example, assuming that 3 PCs can accout for population structure, we enable `--ld` option to calculate and output the ancestry adjusted LD matrix in a file with suffix `.residuals`.

```
./PCAone -b example/plink -k 3 --ld -o adj
```

6.10 Report LD statistics

Currently, the LD R2 for pairwise SNPs within a window can be outputted via `--print-r2` option.

```
./PCAone -B adj.residuals \
        --match-bim adj.mbim \
        --ld-bp 1000000 \
        --print-r2 \
        -o adj
```

We provide the calc_decay_bin.R script to parse the output file `.ld.gz` and calculate the average R2 for each bin as well as plotting. We also provide the nextflow ld.nf for benchmarking the LD statistics.

6.11 Prunning based on Ancestry-Adjusted LD

Given the LD binary file `.residuals` and its associated variant file `.mbim`, we can do pruning based on user-defined thresholds and windows

```
./PCAone -B adj.residuals \
         --match-bim adj.mbim \
         --ld-r2 0.8 \
         --ld-bp 1000000 \
         -o adj
```

6.12 Clumping based on Ancestry-Adjusted LD

Likewise, we can do clumping based on the Ancestry-Adjusted LD matrix and user-defined association results

```
./PCAone -B adj_ld.residuals \
         --match-bim adj.mbim \
         --clump example/plink.pheno0.assoc,example/plink.pheno1.assoc  \
         --clump-p1 0.05 \
         --clump-p2 0.01 \
         --clump-r2 0.1 \
         --clump-bp 10000000 \
         -o adj
```

7 More tutorials

Let's download the example data first if you haven't done so.

```
wget http://popgen.dk/zilong/datahub/pca/example.tar.gz
tar -xzf example.tar.gz && rm -f example.tar.gz
```

7.1 Genotype data (PLINK)

We want to compute the top 40 PCs for this genotype dataset using 20 threads and only 2 GBs memory. We will use the proposed window-based RSVD algorithm with default setting `--svd 2`.

```
./PCAone --bfile example/plink -k 40 -m 2 -n 20
```

Then, we can make a PCA plot in R.

```
pcs <- read.table("pcaone.eigvecs2",h=F)
plot(pcs[,1:2+2], col=factor(pcs[,1]), xlab = "PC1", ylab = "PC2")
legend("topright", legend=unique(pcs[,1]), col=1:4, pch = 21, cex=1.2)
```

7.2 Genotype dosage (BGEN)

Imputation tools usually generate the genotype probabilities or dosages in BGEN format. To do PCA with the imputed genotype probabilities, we can work on BGEN file with `--bgen` option instead.

```
./PCAone --bgen example/test.bgen -k 10 -n 4 -m 2
```

Then, we can make a PCA plot in R.

```
pcs <- read.table("pcaone.eigvecs",h=F)
pop <- read.table("example/plink.fam",h=F)[,1]
plot(pcs[,1:2], col=factor(pop), xlab = "PC1", ylab = "PC2")
legend("topright", legend=unique(pop), col=factor(unique(pop)), pch = 21, cex=1.2)
```

7.3 Single cell RNA-seq data (CSV)

In this example, we run PCA for the single cell RNAs-seq data using a different input format with a normalization method called count per median log transformation (CPMED).

```
./PCAone --csv example/BrainSpinalCord.csv.zst -k 10 -n 20 -m 4 --scale 2 --svd 1
```

It should take around 5 minutes.

8 Acknowledgements

PCAone use Eigen for linear algebra operation. The IRAM method is based on yixuan/spectra. The bgen lib is ported from jeremymcrae/bgen. The EMU and PCAngsd algorithms are modified from @Jonas packages.