

# Principal Component Analysis All in One

Zilong Li\*

September 27, 2024

## Contents

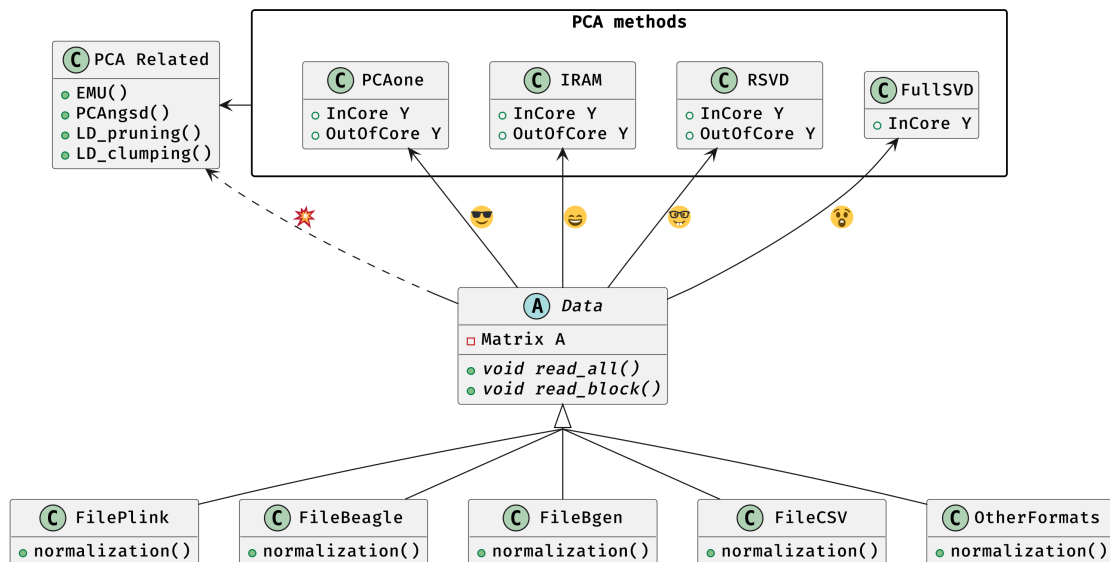
1	Introduction	2
2	Cite the work	2
3	Features	2
4	Quick start	3
4.1	Download PCAone	3
4.2	Download example dataset	4
5	Installation	4
5.1	Download compiled binary	4
5.2	Via Conda	4
5.3	Build from source	4
6	Documentation	6
6.1	Options	6
6.2	Which SVD method to use	7
6.3	Input formats	7
6.4	Output formats	7
6.5	Memory-efficient modes	8
6.6	Data Normalization	9
6.7	Ancestry-Adjusted LD matrix	9
6.8	Report LD statistics	9
6.9	Prunning based on Ancestry-Adjusted LD	9
6.10	Clumping based on Ancestry-Adjusted LD	10
6.11	More Examples	10
7	Acknowledgements	11

---

\*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 method with window based power iteration scheme (PCAone, -svd 2, Algorithm2 in paper). All have both in-core and out-of-core implementation. Additionally, 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 Cite the work

- If you use PCAone, please first cite our paper on genome research **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 adjusted LD statistics for pruning and clumping, please also cite **Measuring linkage disequilibrium and improvement of pruning and clumping in structured populations**.

## 3 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** pruning and clumping method for admixed population.

## 4 Quick start

We can run the following on Linux to have a 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);'
```

We will find those 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
```

### 4.1 Download PCAone

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
```

## 4.2 Download example dataset

```
pkg=http://popgen.dk/zilong/datahub/pca/example.tar.gz
wget $pkg || curl -LO $pkg
tar -xzf example.tar.gz && rm -f example.tar.gz
```

You should find a fold named `example` with some example data.

# 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.

## 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++11 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 --help` to see all options including hidden advanced options. The below are some useful options.

```
Main options:
-h, --help                print all options including hidden advanced options
-d, --svd arg (=2)        svd method to be applied. default 2 is recommended for big data.
                           0: the Implicitly Restarted Arnoldi Method (IRAM)
                           1: the Yu's single-pass Randomized SVD with power iterations
                           2: the proposed window-based Randomized SVD method
                           3: the full Singular Value Decomposition.
-b, --bfile arg           prefix to PLINK .bed/.bim/.fam files
-B, --binary arg          path of binary file
-c, --csv arg             path of comma separated 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
-k, --pc arg (=10)        top k principal components (PCs) to be calculated
-m, --memory arg (=0)     desired RAM usage in GB unit for out-of-core mode. 0 is for
    ↪ in-core mode
-n, --threads arg (=10)   number of threads to be used
-o, --out arg (=pcaone)   prefix to output files. default [pcaone]
-p, --maxp arg (=40)      maximum number of power iterations for RSVD algorithm
-S, --no-shuffle          do not shuffle the data for --svd 2 if it is already permuted
-v, --verbose             verbose message output
-w, --batches arg (=64)   number of mini-batches to be used by PCAone --svd 2
-C, --scale arg (=0)      do scaling for input file.
                           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
--emu                    uses EMU algorithm for genotype input with missingness
--pcangsd                uses PCAngsd algorithm for genotype likelihood input
--maf arg (=0)           exclude variants with MAF lower than this value (in-core mode)
    ↪ only
-V, --printv             output the right eigenvectors with suffix .loadings
--ld                     output a binary matrix for downstream LD related analysis
--ld-bim arg             variants information in plink bim file related to LD matrix
--ld-r2 arg (=0)         r2 cutoff for LD-based pruning.
--ld-bp arg (=1000000)   physical distance threshold in bases for ld pruning
--ld-stats arg (=0)      statistics to get r2 for LD. (0: the ancestry adjusted, 1: the
    ↪ standard)
--print-r2               print LD r2 for pairwise SNPs in a ld-window (saved in
    ↪ <prefix>.ld.gz)
--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, particularly 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 print it.

### 6.4.4 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 a file named with `.kept.bim`. 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 `.kept.bim` file, etc.

#### 6.4.5 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 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 subtracting 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 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 account 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 -m 4
```

## 6.8 Report LD statistics

Currently, the LD  $r^2$  for pairwise SNPs within a window can be outputted via `--print-r2` option.

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

## 6.9 Pruning based on Ancestry-Adjusted LD

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

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

## 6.10 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 \  
  --ld-bim adj.kept.bim \  
  --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
```

## 6.11 More Examples

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
```

### 6.11.1 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)
```

### 6.11.2 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)
```

### 6.11.3 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.

## 7 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.