

# PCAone - Principal Component Analysis All in One

Zilong Li\*

October 6, 2023

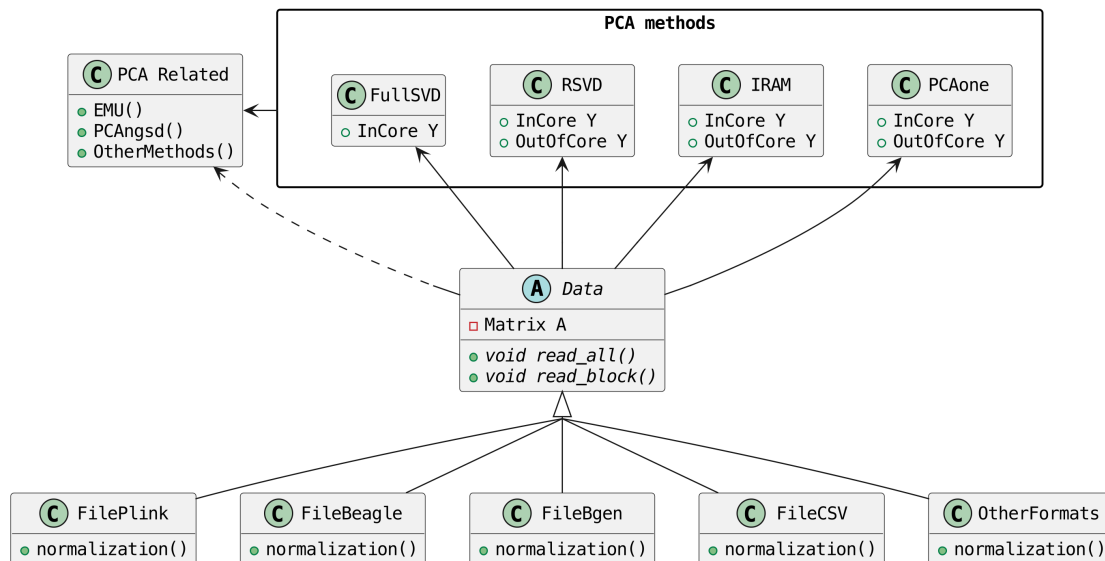
## Contents

<b>1</b>	<b>Quickstart</b>	<b>3</b>
1.1	Download PCAone . . . . .	3
1.2	Download example dataset . . . . .	3
<b>2</b>	<b>Features</b>	<b>4</b>
<b>3</b>	<b>Installation</b>	<b>4</b>
3.1	Download compiled binary . . . . .	4
3.2	Via Conda . . . . .	4
3.3	Build from source . . . . .	4
<b>4</b>	<b>Documentation</b>	<b>6</b>
4.1	Options . . . . .	6
4.2	Input formats . . . . .	6
4.3	Output files . . . . .	6
4.4	Running mode . . . . .	7
4.5	Normalization . . . . .	7
4.6	LD prune . . . . .	8
4.7	Examples . . . . .	8
<b>5</b>	<b>Citation</b>	<b>9</b>
<b>6</b>	<b>Acknowledgements</b>	<b>9</b>

---

\*zilong.dk@gmail.com

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, PCAoneArnoldi), **single pass Randomized SVD** but with power iteration scheme (RSVD, Algorithm1 in paper) and our own RSVD method with window based power iteration scheme (PCAone, Algorithm2 in paper). All have both in-core and out-of-core implementation. There is also an **R** package here but without out-of-core implementation. 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.



# 1 Quickstart

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-core mode if having enough RAM
./PCAone --bfile example/plink -k 10
R -s -e 'df=read.table("pcaone.eigvecs", h=F);plot(df[,1:2], xlab="PC1", ylab="PC2");'
```

We will find those files in your 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.perm.bed # the permuted bed file, only for PCAone
pcaone.perm.bim # the permuted bim file, only for PCAone
pcaone.perm.fam # the permuted fam file, only for PCAone
pcaone.log      # log file
```

## 1.1 Download PCAone

### 1.1.1 Linux

```
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, one can download the following 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
```

### 1.1.2 Mac

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

## 1.2 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
```

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

## 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 prune method for admixed population.

## 3 Installation

There are 3 ways to install PCAone.

### 3.1 Download compiled binary

There are compiled binaries provided for both Linux and Mac platform. Check [the releases page](#) to download one.

### 3.2 Via Conda

PCAone is also available from [bioconda](#).

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

### 3.3 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
- llvm or libomp for MacOS

We **recommend** building the software from source with MKL as backend to maximize the performance. For MacOS users, we recommend using `llvm` or `gcc` by `brew install llvm gcc` instead of the default `clang` shipped with MacOS. One should run `export CC=$(find $(brew --prefix)/bin/ -name "gcc-[0-9]*" | tail -1); export CXX=$(find $(brew --prefix)/bin/ -name "g++-[0-9]*" | tail -1)` and check the similar [mac workflow](#).

### 3.3.1 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.

### 3.3.2 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.

## 4 Documentation

### 4.1 Options

run `./PCAone --help` to see all options. Below are some useful and important options.

Main options:

<code>-h, --help</code>	print list of all options including hidden advanced options
<code>-d, --svd arg (=2)</code>	svd method to be applied. 0 is the recommended for big data 0: the implicitly restarted arnoldi method 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.
<code>-b, --bfile arg</code>	prefix to PLINK .bed/.bim/.fam files
<code>-B, --binary arg</code>	path of binary file
<code>-c, --csv arg</code>	path of comma seperated CSV file compressed by zstd
<code>-g, --bgen arg</code>	path of BGEN file
<code>-G, --beagle arg</code>	path of BEAGLE file
<code>-k, --pc arg (=10)</code>	top k components to be calculated
<code>-m, --memory arg (=0)</code>	specify the RAM usage in GB unit. default [0] uses all RAM
<code>-n, --threads arg (=10)</code>	number of threads for multithreading
<code>-o, --out arg (=pcaone)</code>	prefix to output files. default [pcaone]
<code>-p, --maxp arg (=40)</code>	maximum number of power iterations for RSVD algorithm
<code>-S, --no-shuffle</code>	do not shuffle the data if it is already permuted
<code>-v, --verbose</code>	verbose message output
<code>-w, --batches arg (=64)</code>	number of mini-batches to be used by PCAone (algorithm2)
<code>-C, --scale arg (=0)</code>	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
<code>--emu</code>	use EMU algorithm for genotype data with missingness
<code>--pcangsd</code>	use PCAngsd algorithm for genotype likelihood input
<code>--maf arg (=0)</code>	skip variants with minor allele frequency below maf
<code>-V, --printv</code>	output the right eigen vectors with suffix .loadings

### 4.2 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.

### 4.3 Output files

#### 4.3.1 Eigen vectors

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

#### 4.3.2 Eigen values

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

### 4.3.3 Features Loadings

Features Loadings are saved in file with suffix `.loadings`. Each row represents a feature and each col represents a PC. need to use `--printv` option to print it.

## 4.4 Running mode

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 in total 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.

### 4.4.1 run the window-based RSVD method (algorithm2) with in-core mode

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

### 4.4.2 run the window-based RSVD method (algorithm2) with out-of-core mode

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

### 4.4.3 run the Yu+Halko RSVD method (algorithm1) with in-core mode

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

### 4.4.4 run the Yu+Halko RSVD method (algorithm1) with out-of-core mode

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

### 4.4.5 run the IRAM method with in-core mode

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

### 4.4.6 run the IRAM method with out-of-core mode

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

### 4.4.7 run the Full SVD method with in-core mode

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

## 4.5 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.

## 4.6 LD prune

This is a novel LD prune method on admixed population. For more details, see our paper.

```
PCAone -b plink -k 3 -n 20 --svd 0 --ld-r2 0.8 --ld-window 1000000 --maf 0.05
```

## 4.7 Examples

Let's download the example data first.

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

### 4.7.1 Genotype data (PLINK)

We want to compute the top 10 PCs for this genotype dataset using 4 threads and only 2GB memory. We will use the proposed window-based RSVD algorithm with default setting `--svd 2`.

```
./PCAone --bfile example/plink -k 10 -n 4 -m 2
```

Then, we can make a PCA plot in R.

```
pcs <- read.table("pcaone.eigvecs",h=F)
fam <- read.table("example/plink.fam",h=F)
pop <- fam[,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)
```

### 4.7.2 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)
fam <- read.table("example/plink.fam",h=F)
pop <- fam[,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)
```

### 4.7.3 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.



## 5 Citation

- If you find PCAone helpful, please cite our paper on genome reseach [Fast and accurate out-of-core PCA framework for large scale biobank data](#).
- If using EMU algorithm, please also cite [Large-scale inference of population structure in presence of missingness using PCA](#).
- If using PCAngsd algorithm, please also cite [Inferring Population Structure and Admixture Proportions in Low-Depth NGS Data](#).

## 6 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.