**CTPR v1.1 User Manual**

*Wonil Chung*

*January 29, 2019*

**1. Overview**

The CTPR (Cross-Trait / Cross-eThnic Penalized Regression) software was originally developed for multi-trait polygenic risk prediction in large cohorts and is subsequently being extended for multi-ethnic polygenic risk prediction. It utilizes multiple secondary traits (or ethnicities) based on individual-level genotypes and/or summary statistics from large-scale GWAS studies to improve prediction accuracy. Based on penalized least squares methods, we propose a novel cross trait penalty function with the Lasso and the minimax concave penalty (MCP) to incorporate the shared genetic effects across multiple traits (or ethnicities) and implement it for large-sample GWAS data. Our approach extracts information from the secondary traits (or ethnicities) that is beneficial for predicting the primary trait (or ethnicity) but tunes down information that is not. Our novel implementation of a distributed memory parallel computing algorithm makes it feasible to apply our methods to biobank-scale GWAS data. We compared our multi-trait methods with other existing methods such as MTGBLUP (multi-trait genomic best linear unbiased prediction method)[1](#_ENREF_1), wMT-SBLUP (weighted multi-trait summary statistic best linear unbiased prediction method)[2](#_ENREF_2), MTAG (multi-trait analysis of GWAS)[3](#_ENREF_3) and showed that our approach outperforms them in predictive performance.

**Citations**

The CTPR algorithm is described in the following reference[4](#_ENREF_4):

Wonil Chung, Jun Chen, Constance Turman, Sara Lindstrom, Zhaozhong Zhu, Po-Ru Loh, Peter Kraft and Liming Liang (2019), Efficient cross-trait penalized regression increases prediction accuracy in large cohorts using secondary phenotypes. Nature Communications, 10, 569.

Wonil Chung and Liming Liang, (2019), Improving the polygenic risk prediction by incorporating LD information from multi-ethnic Biobank data. In preparation.

**Questions and Requests**

If you have any questions on CTPR software, please email to Wonil Chung ([wchung@hsph.harvard.edu](mailto:wchung@hsph.harvard.edu)).

**2. Installing and compiling CTPR**

## You can download the latest version of the CTPR software at: <https://github.com/wonilchung/CTPR>.

**2.1. Change log**

Version 1.2 (expected in February, 2019):

* Will support MACH dosage file format (e.g. test.mldose, test.mlinfo).
* Will add options for data management to specify a list of individuals to be included or excluded and a list of SNPs to be included or exclude in the analysis.

Version 1.1 (January 29, 2019):

* Modified file type for genotype data from double to float to decrease the memory size in half.
* Added various options for data management.

Version 1.0 (March 3, 2017):

* Initial release of CTPR.

**2.2. Installation**

The *ctpr\_vx.x.tar.gz* download package contains a standalone (i.e., statically linked) 64-bit Linux executable, CTPR, which we have tested on several Linux systems. If you wish to compile your own version of the CTPR software from the source code, you will need to ensure that compiler requirements and library dependencies are fulfilled, and you will need to make appropriate modifications to the Makefile (MakefileSpp for a single node version or MakfileMpi for MPI version). We explain how to install required packages and compile CTPR software on linux system below.

(1) Install Armadillo package

Armadillo is a C++ library for linear algebra and scientific computing (see <http://arma.sourceforge.net/>). Before installing Armadillo package, you need to ensure CMake tool, LAPACK and BLAS (or preferably OpenBLAS) are installed on your system. If these tools are not installed, you can download them from <http://www.cmake.org> (CMake tool) and <http://www.openblas.net/> (OpenBLAS library). If you wish to install the library and headers for Armadillo package in a user’s own directory, please use the option CMAKE\_INSTALL\_PREFIX as follows.

wget http://sourceforge.net/projects/arma/files/armadillo-9.200.7.tar.xz

tar xf armadillo-9.200.7.tar.xz

cd armadillo-9.200.7

cmake . -DCMAKE\_INSTALL\_PREFIX:PATH=/xxxx/armadillo-9.200.7

make && make install

(2) Install openmpi

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners (see <https://www.open-mpi.org/>).

wget https://download.open-mpi.org/release/open-mpi/v2.0/openmpi-2.0.4.tar.gz

tar -xvf openmpi-2.0.4.tar.gz

(3) Compile CTPR

Before compiling CTPR, you need to modify 'make file' (MakefileSpp for a single node version and MakefileMpi for MPI version) using customized paths for Armadillo (including LAPACK, BLAS) and Open MPI.

make -f MakefileSpp

make -f MakefileMpi

**2.3. Running CTPR**

Before running CTPR, it is necessary to specify the path for the dynamic shared libraries (i.e. xxx.so) CTPR was linked to. If the module system is installed on your system, you can use the ‘module load’ command. If not, you need to add all paths for Armadillo and Open MPI to LD\_LIBRARY\_PATH as follows.

module load gcc/6.2.0

module load openblas/0.2.19

module load openmpi/2.0.1

module load armadillo/9.200.7

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/n/app/gcc/6.2.0/lib:/n/app/gcc/6.2.0/lib64"

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/n/app/openblas/0.2.19/lib"

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/n/app/openmpi/2.0.1/lib"

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/home/wc120/packages/armadillo-9.200.7/lib64"

To run CTPR, simply invoke ./ctpr or ./ctprmpi on the Linux command line (within the CTPR install directory). To obtain information on license of CTPR, run: ./ctpr –l or ./ctprmpi –l. To obtain full list of CTPR options, run: ./ctpr –h or ./ctprmpi –h. The example/ subdirectory contains example data and code, so you can learn how to execute CTPR software.

**3. Computing Requirements**

Basically, CTPR can run on any computing system including PC and Mac but with large-scale biobank-based GWAS data, we recommend using Linux-based high performance computing cluster. For distributed high performance computing, clusters utilize job schedulers such as LSF (Load Sharing Facility) and SLURM (Simple Linux Utility for Resource Management) to start, execute and monitor jobs on a set of allocated computing nodes. We will explain how to execute CTPR on cluster computer using SLURM.

**3.1. Operating system**

We have only compiled and tested CTPR on Linux computing environments including CenOS and Debian Linux. However, the source code is available and thus you can compile CTPR for a different OS.

**3.2. Memory**

For typical data sets, CTPR utilizes approximately N\*P\*4 bytes of memory, where N is the number of individuals and P is the number of SNPs. More precisely:

N = # of individuals in dosage file (MACH output format; mldose) that satisfy all of the conditions:

* listed in any --keep file
* not listed in any --remove file

P = # of SNPs in information file (MACH output format; mlinfo) that satisfy all of the conditions:

* listed in any --include file
* not listed in any --exclude file

**3.3 Running time**

To assess the computational feasibility of CTPR for biobank-based GWAS data, we tested with N=437K individuals and P=1M SNPs from UK Biobank, which required ~1.7TB of memory with float data type (i.e. 437K\*1M\*4B=~1.7TB)[4](#_ENREF_4). The CTPR ran on 40 cores (Intel Xeon CPU 2.1 GHz) with 48GB of memory for each core, total of ~1.9TB of memory, for up to 7 days to complete the analyses with 40 core-groups (exact solution). The running time of CTPR depends linearly not only on the sample size (*N*) and the number of SNPs (*P*) but also on the number of core-group (*q*), which represents *O(NPq)*. With 10 core-groups (approximate solution), the running time of CTPR dropped to ~1.75 days and it still generated almost the same predictive performance as exact solution due to good convergence. Even when sample size increases, the running time is able to remain similar because larger sample size increases likelihood of convergence and therefore less number of core-groups are needed. When you use Slurm scheduler, you can specify as

sbatch -n 40 --mem-per-cpu=48G -p mpi -t 5-00:00:00 --wrap="mpirun -np 40 ./ctprmpi --ng 40 …”  
sbatch -n 40 --mem-per-cpu=48G -p mpi -t 5-00:00:00 --wrap="mpirun -np 40 ./ctprmpi --ng 10 …”

**4. Input File Format**

CTPR requires three input files containing genotype, phenotype and summary data. Genotype files can be in dosage (.dose or .mldose) formats and phenotype and summary files can be in the text (txt) file formats.

**4.1. Genotype File Format**

**Dosage format:** The simple dosage file (.dose) contains only genotype information. The number of rows is equivalent to the number of individuals (N) and the number of columns is equal to the number of markers (P).

test.dose (no header line; columns are SNP1, SNP2, SNP3, …)

0.79304 0.54848 0.86099 ....

0.79304 1.51711 2.33368 ....

0.87451 1.51711 0.86099 ....

....

**MACH format:** The MACH dosage file (.mldose) contains individual IDs as well as genotypes. The MACH information file (.mlinfo) contains SNP IDs, two alleles, frequency and Imputation R2.

test.mldose (no header line; columns are Family ID, Individual ID, SNP1, SNP2, SNP3…)

FID1 IID1 0.79304 0.54848 0.86099 ....

FID1 IID2 0.79304 1.51711 2.33368 ....

FID2 IID3 0.87451 1.51711 0.86099 ....

....

test.mlinfo (header line; marker name, allele 1, allele 2, frequency of allele 1, probability for the most likely genotype, Imputation R2)

SNP Al1 Al2 Freq Quality Rsq

SNP1 A C 0.2467 0.6943 0.999

SNP2 C G 0.0583 0.4935 0.879

SNP3 G A 0.4829 0.5932 0.694

....

**4.2. Phenotype File Format**

This file contains phenotype information. Each line has numbers indicating multiple phenotypic values for each individual in turn, in the same order as in genotype file. Each column is related to each phenotype. The number of rows should be equal to the number of individuals (N) and the number of columns should be equal to the number of phenotype (K).

phenotype.txt (no header line; columns are 1st phenotype, 2nd phenotype, …)

-2.41873 -0.90627

0.74807 -0.20951

1.44223 1.26426

……

**4.3. Summary File Format**

This file contains summary statistics for all markers. The first column is marker id, the second column is its minor allele frequency (MAF), the third column is its maker effect (beta) and the fourth column is its standard error of marker effect (se). This file contains marker effects for multiple phenotypes. The number of rows is the same as the number of markers (P) and the number of columns is 2\*K+2 where K is the number of phenotypes.

summary.txt (no header line; columns are marker name, minor allele frequency, beta1, se1,, beta2, se2…)

1 0.43139 0.01826 0.01195 0.03643 0.01533

2 0.67261 -0.00978 0.01205 -0.02564 0.01536

3 0.67029 -0.00969 0.01195 0.03246 0.01364

4 0.33718 0.00069 0.01218 -0.02464 0.01467

……

**5. CTPR Options**

**5.1. Input and output**

--out or --output [prefix]: specify output file prefix

--dos or --dosage [filename]: specify dosage file name for training

--dos-ext or --dosage-extension [ext]: specify dosage file extension for training

--phe or --phenotype [filename]: specify phenotype file name for training

--dos-test or --dosage-test [filename]: specify dosage file name for testing

--dos-test-ext or --dosage-test-extension [ext]: specify dosage file extension for testing

--phe-test or --phenotype-test [filename]: specify phenotype file name for testing

--sum or --summary [filename]: specify summary file name

--sum-ext or --summary-extension [ext]: specify summary file extension

--num-phe or --number-phenotype [num]: specify the number of phenotypes to be analyzed (default 1)

--num-sum or --number-summary [num]: specify the number of phenotypes for summary file (default 1)

--separ-ind or --separate-individual [num,num,..]: specify the numbers of individuals for each phenotypes separated by comma (,) in case of multiple phenotypes

**5.2. Data management**

--keep [filename]: specify a list of individuals to be included in the analysis

--remove [filename]: specify a list of individuals to be excluded from the analysis

--include [filename]: specify a list of SNPs to be included in the analysis

--exclude [filename]: specify a list of SNPs to be excluded from the analysis

--scaling or --scaling-phenotype: specify for scaling secondary phenotypes using simple linear regression between phenotypes and genotypes

**5.3. Coordinate decent algorithm**

--penalty or --penalty-term [num]: specify the sparsity and cross-trait penalty terms

(default 1; 1: Lasso+CTPR; 2: MCP+CTPR)

--nfold or --number-fold [num]: specify the number of folds for coordinate decent algorithm (default 5)

--prop or --proportion [num]: specify proportion of maximum number of non-zero beta (default 0.25)

--lambda2 or --lambda2-option [num]: specify value for lambda2.

If negative value is specified, pre-specified values are used for lambda2 (default -3)

-1: =(0, 0.94230, 1.60280, 3.37931);

-2:=(0, 0.06109, 0.94230, 0.13920, 0.24257, 0.38582, 0.59756, 0.94230, 1.60280, 3.37931);

-3:=(0, 0.06109, 0.94230, 0.13920, 0.24257, 0.38582, 0.59756, 0.94230, 1.60280, 3.37931, 24.5);

-4:=(0, 0.06109, 0.94230, 0.13920, 0.24257, 0.38582, 0.59756, 0.94230, 1.60280, 3.37931, 8.5, 15.5, 24.5))

--ng or --number-group [num]: specify the number of group for MPI mode (default number of MPI nodes)

--st or --start-number [num]: specify starting number for MPI files (default 1)

--flamb1 or --first-lambda1 [num]: specify first lambda1 value for MPI mode (default 1)

--llamb1 or --last-lambda1 [num]: specify last lambda1 value for MPI mode (default 100)

**6. Running CTPR**

**6.1 Example codes for a single node version**

./ctpr \

--out ./res/test \

--dos final\_5000\_train.dose \

--phe final\_pheno\_5000\_train.phe \

--dos-test final\_5000\_test.dose \

--phe-test final\_5000\_test.phe \

--separ-ind 7400,7400 \

--penalty 1 \

--lambda2 0

**6.2 Example codes for MPI version**

mpirun -x LD\_PRELOAD=libmpi.so -np 2 ./ctprmpi \

--out ./res/testmpimcp \

--dos final\_one\_scaled\_5000\_train \

--dos-ext dose \

--phe final\_one\_summary\_pheno\_5000\_train.phe \

--dos-test final\_one\_scaled\_5000\_test \

--dos-test-ext dose \

--phe-test final\_one\_summary\_pheno \_5000\_test.phe \

--sum final\_one\_marginal\_beta \_part \

--sum-ext txt \

--num-phe 1 \

--penalty 2 \

--lambda2 0.13920

**References**

1. Maier, R. *et al.* Joint analysis of psychiatric disorders increases accuracy of risk prediction for schizophrenia, bipolar disorder, and major depressive disorder. *The American Journal of Human Genetics* **96**, 283-294 (2015).

2. Maier, R.M. *et al.* Improving genetic prediction by leveraging genetic correlations among human diseases and traits. *Nature communications* **9**, 989 (2018).

3. Turley, P. *et al.* Multi-trait analysis of genome-wide association summary statistics using MTAG. *Nature genetics* **50**, 229 (2018).

4. Chung, W. *et al.* Efficient cross-trait penalized regression increases prediction accuracy in large cohorts using secondary phenotypes. *Nature communications* **10**, 569 (2019).