Second-generation PLINK: Rising to the challenge of larger and richer datasets

Christopher C. Chang
Complete Genomics
chrchang@alumni.caltech.edu

Background, motivation

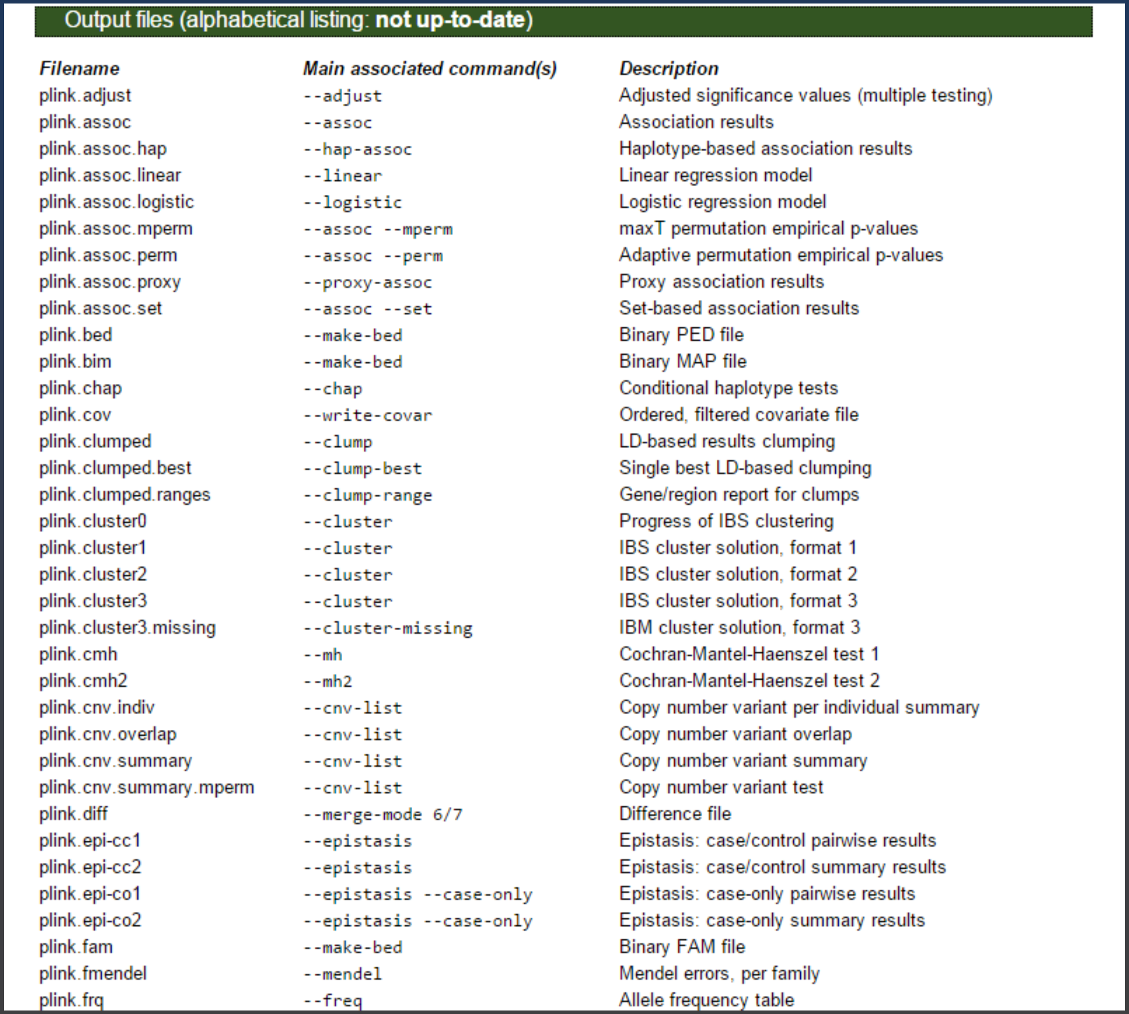
- PLINK 1 is a widely used program for managing and analyzing genomic datasets
  - Core data format limited in scope…

    `00` = hom. minor `01` = missing call `10` = heterozygous `11` = hom. major
  - but it's **very efficient for what it does**; perfect for "big data"
- A modernized PLINK is an excellent complement to more versatile, but slower, VCF-based tools

Roadmap

- PLINK 1.9 (2014): new algorithms
  - Biggest win: use bit population count everywhere
  - Dataset no longer has to fit in RAM
- PLINK 2.0 (2015): new data format
  - Current format is *too* restricted for modern GWAS
  - Low-MAF variant data should be compressed
    - SNPack (Sambo et al., 2014) has excellent ideas
  - (Also, current .bed file extension is confusing…)

Non-goal: many more methods

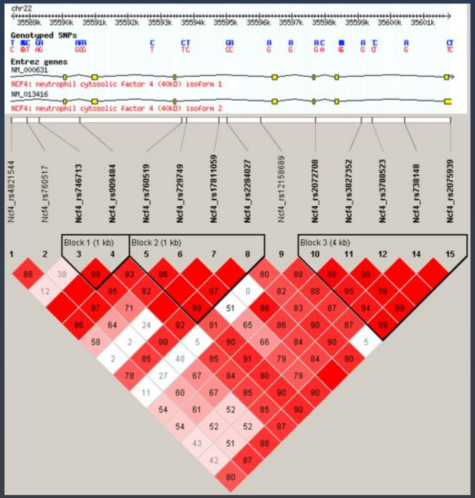


Output files (alphabetical listing: not up-to-date)

Filename	*Main associated command(s)*	*Description*
plink.adjust	`--adjust`	Adjusted significance values (multiple testing)
plink.assoc	`--assoc`	Association results
plink.assoc.hap	`--hap-assoc`	Haplotype-based association results
plink.assoc.linear	`--linear`	Linear regression model
plink.assoc.logistic	`--logistic`	Logistic regression model
plink.assoc.mperm	`--assoc --mperm`	maxT permutation empirical p-values
plink.assoc.perm	`--assoc --perm`	Adaptive permutation empirical p-values
plink.assoc.proxy	`--proxy-assoc`	Proxy association results
plink.assoc.set	`--assoc --set`	Set-based association results
plink.bed	`--make-bed`	Binary PED file
plink.bim	`--make-bed`	Binary MAP file
plink.chap	`--chap`	Conditional haplotype tests
plink.cov	`--write-covar`	Ordered, filtered covariate file
plink.clumped	`--clump`	LD-based results clumping
plink.clumped.best	`--clump-best`	Single best LD-based clumping
plink.clumped.ranges	`--clump-range`	Gene/region report for clumps
plink.cluster0	`--cluster`	Progress of IBS clustering
plink.cluster1	`--cluster`	IBS cluster solution, format 1
plink.cluster2	`--cluster`	IBS cluster solution, format 2
plink.cluster3	`--cluster`	IBS cluster solution, format 3
plink.cluster3.missing	`--cluster-missing`	IBM cluster solution, format 3
plink.cmh	`--mh`	Cochran-Mantel-Haenszel test 1
plink.cmh2	`--mh2`	Cochran-Mantel-Haenszel test 2
plink.cnv.indiv	`--cnv-list`	Copy number variant per individual summary
plink.cnv.overlap	`--cnv-list`	Copy number variant overlap
plink.cnv.summary	`--cnv-list`	Copy number variant summary
plink.cnv.summary.mperm	`--cnv-list`	Copy number variant test
plink.diff	`--merge-mode 6/7`	Difference file
plink.epi-cc1	`--epistasis`	Epistasis: case/control pairwise results
plink.epi-cc2	`--epistasis`	Epistasis: case/control summary results
plink.epi-co1	`--epistasis --case-only`	Epistasis: case-only pairwise results
plink.epi-co2	`--epistasis --case-only`	Epistasis: case-only summary results
plink.fam	`--make-bed`	Binary FAM file
plink.fmendel	`--mendel`	Mendel errors, per family
plink.frq	`--freq`	Allele frequency table

Speedup example: --blocks

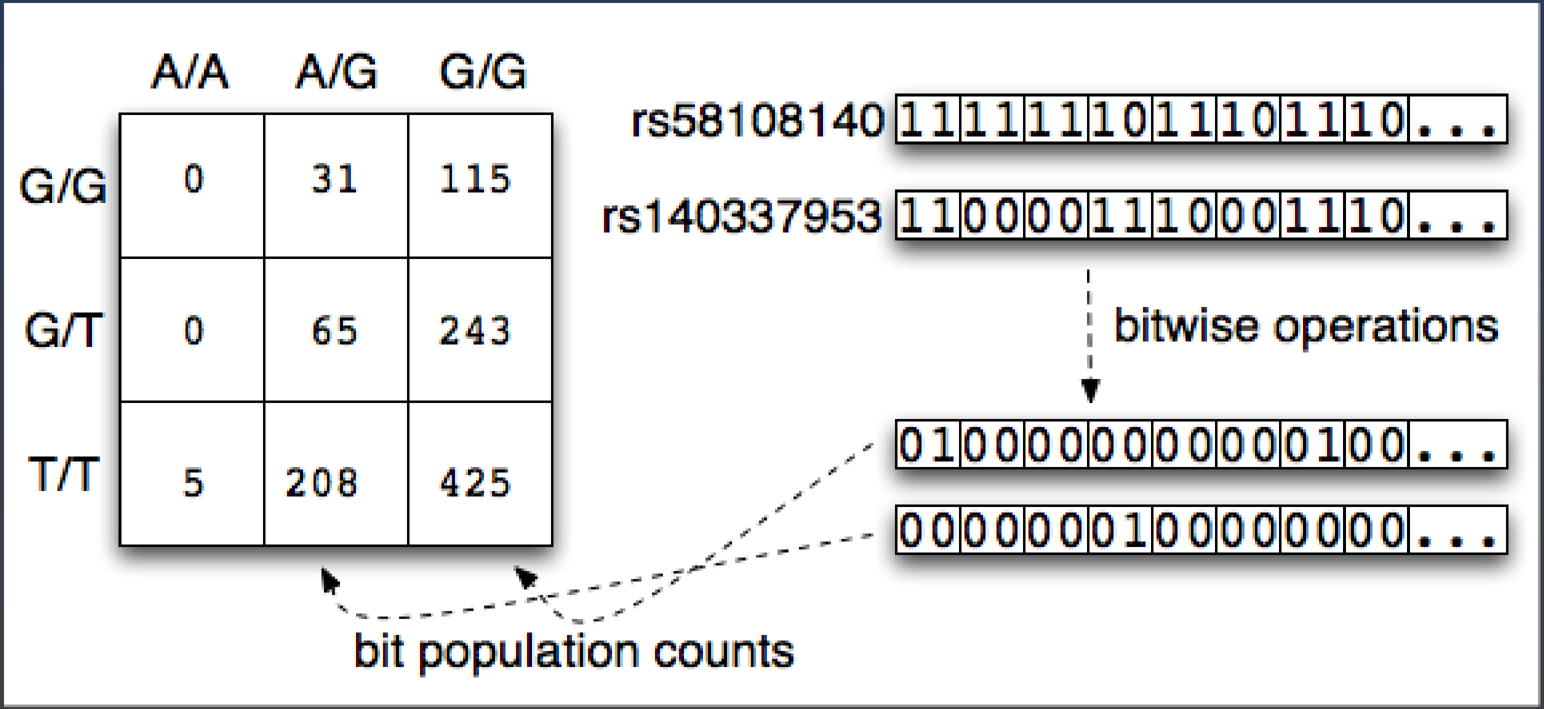
- Partitions genome into haplotype blocks, using Haploview's method (Gabriel et al., 2002; Wall and Pritchard, 2003) (picture from Olsson, 2007)



- Three key steps:
  - Compute 3x3 contingency tables for variant pairs
  - Classify D' confidence interval
  - Use classifications to determine block boundaries

--blocks contingency tables

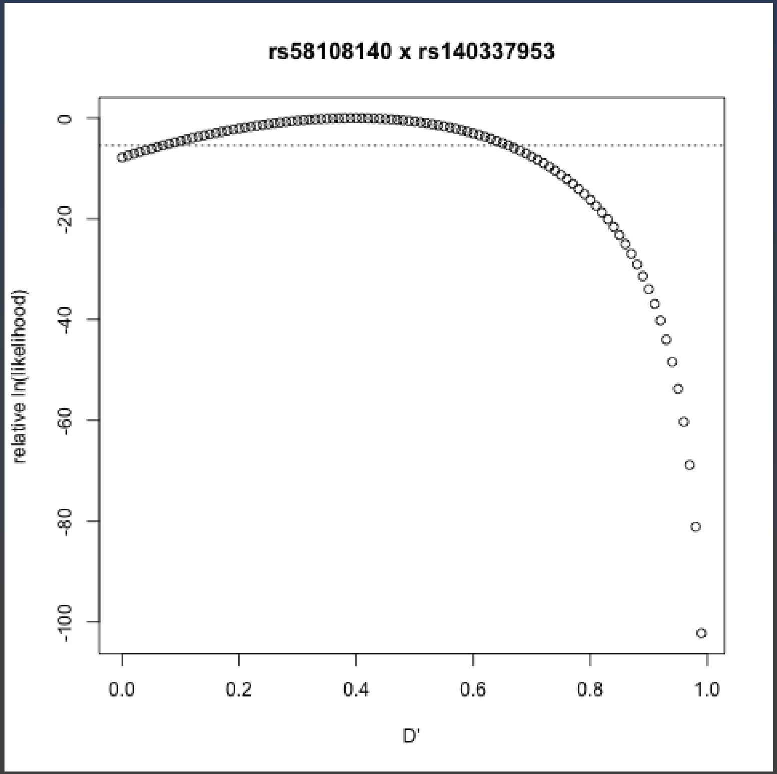
- Use bit population counts

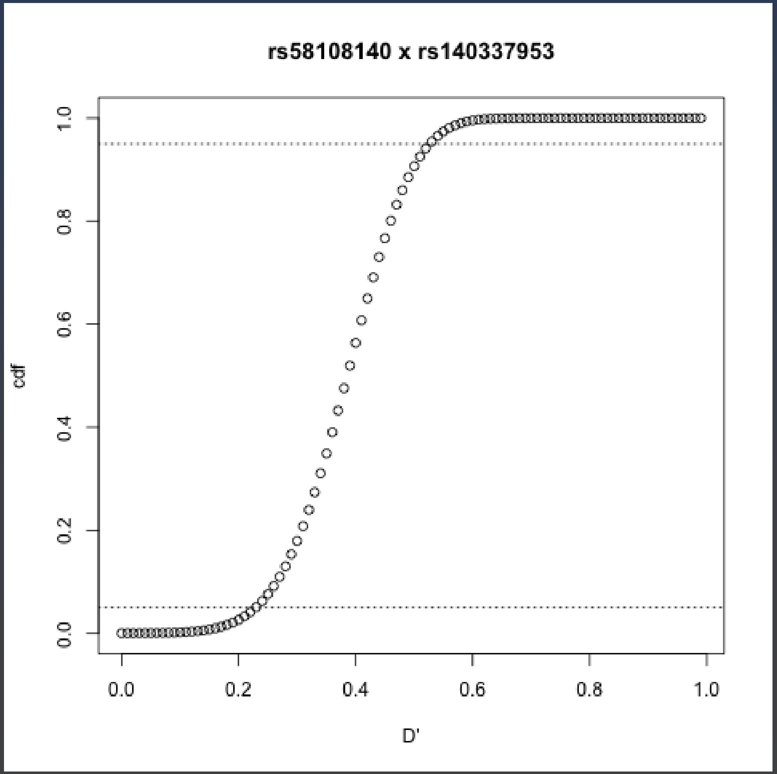


	A/A	A/G	G/G
G/G	0	31	115
G/T	0	65	243
T/T	5	208	425

--blocks CI classification

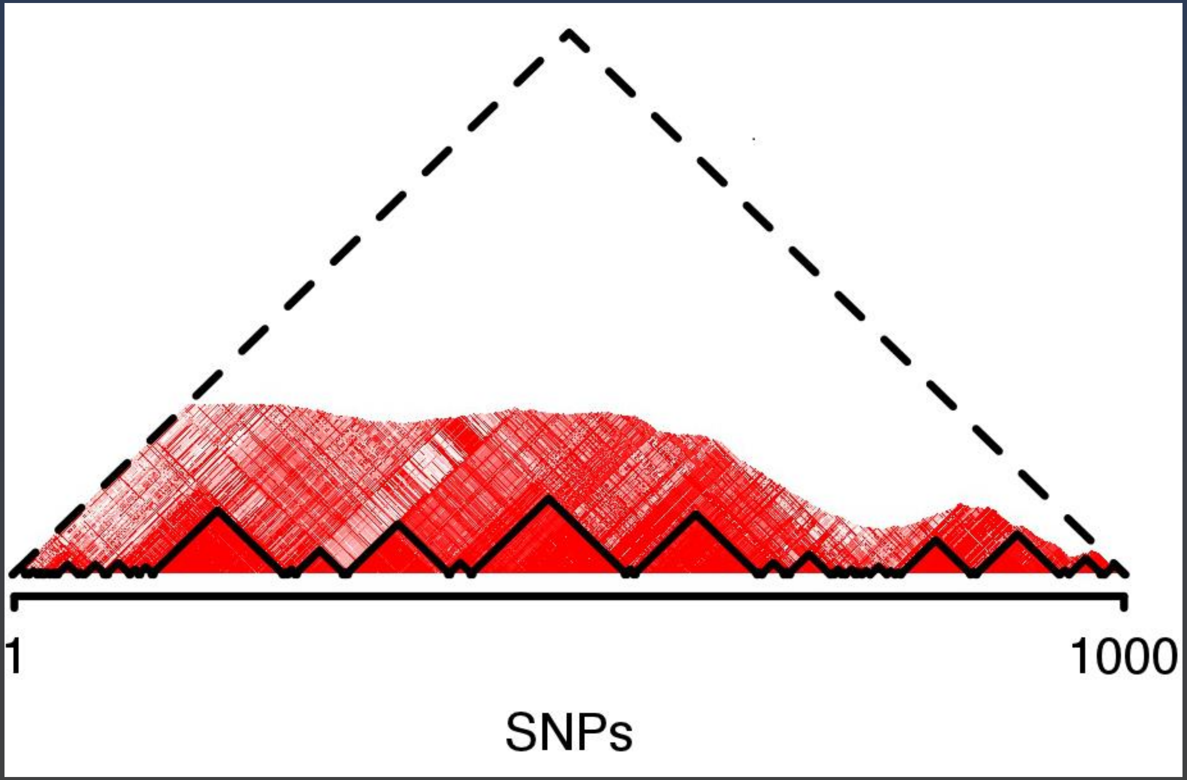
- Solve Hill's cubic equation, exploit unimodality when present





--blocks final partitioning

- Apply optimizations from LDExplorer (Taliun, 2014)



Combined effect: >1000x

1000 Genomes phase 1 chr 1 runtimes (seconds) using a 500kb scanning window

Machine	PLINK 1.07	PLINK 1.90
Mac-2	~2.7m	550.9
Mac-12	~3.6m	426.0
Linux32-2	~4.3m	1288.4
Linux64-512	~2.6m	1119.7
Win32-2	~17m	4535.8
Win64-2	~5.7m	1037.2

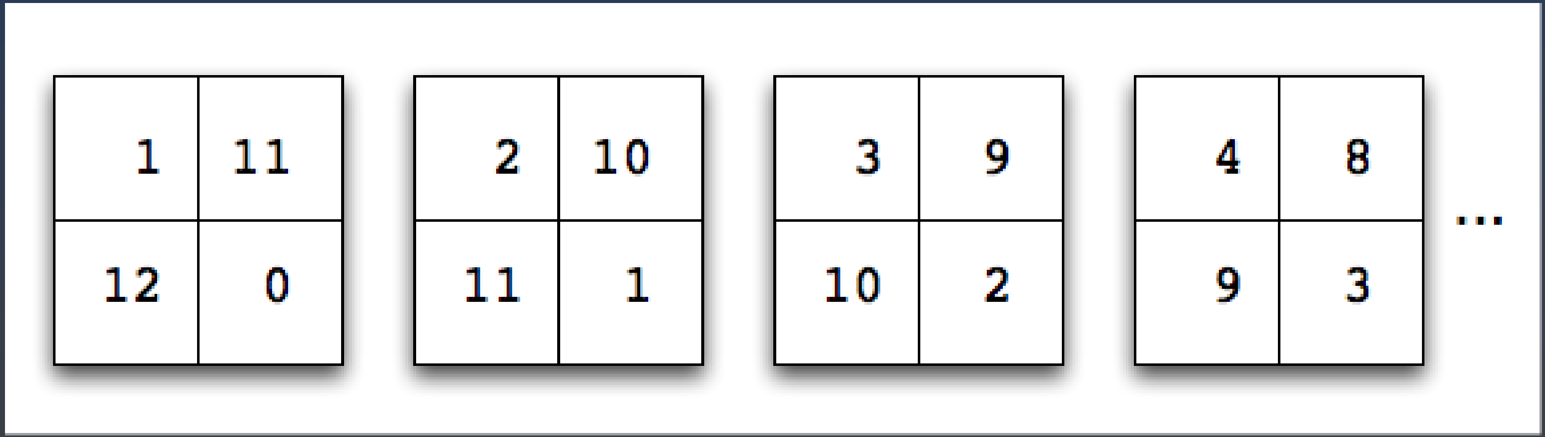
Other integrated algorithms

- Likelihood ratio-based epistasis test from BOOST (Wan et al., 2010), variance correction and joint-effects epistasis test from (Ueki & Cordell, 2012)
- PERMORY LD-exploiting permutation test (Steiß et al., 2012)
- “GWASSpeedup” TopCoder logistic regression contest (Loh et al., manuscript in preparation)
- `pigz` parallel compression (Adler, 2007)
- Number-to-string encoder discussed in Alexandrescu’s “Three Optimization Tips for C++”

Speedup example: Fisher's exact test

- p-value =

$$\frac{\text{[\# of lower/equal-multiplicity tables]}}{\text{[total \# of tables with same row and column sums]}}$$



1	11
12	0

2	10
11	1

3	9
10	2

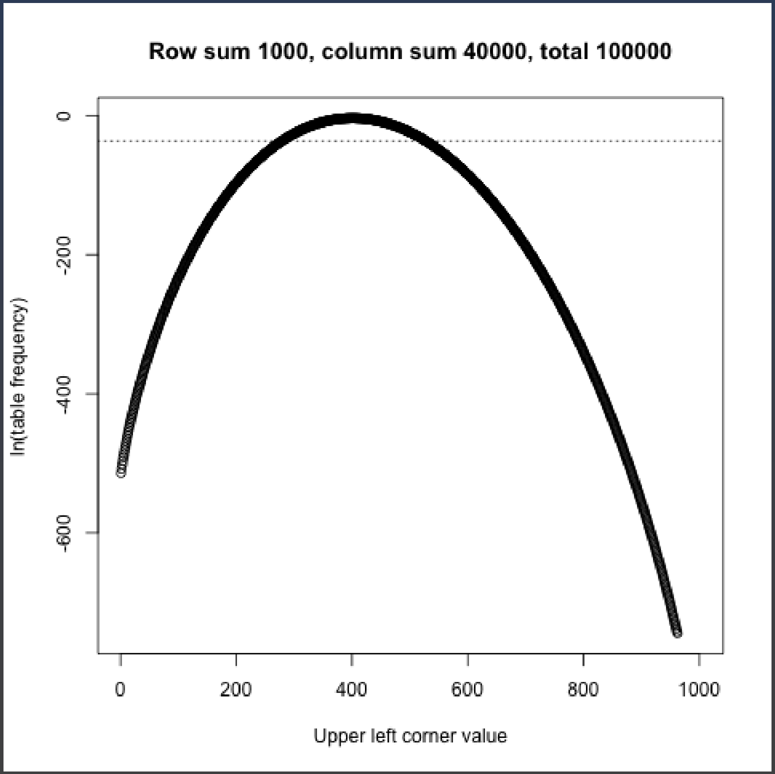
4	8
9	3

...

- SNP-HWE (Wigginton et al., 2005): no need to directly compute or estimate factorials; instead, scale starting point to 1 and compute relative likelihoods, since adjacent table multiplicities are related by simple ratios

Speedup example: Fisher's exact test

- PLINK 1.9: terminate computation as soon as floating-point precision limit is reached



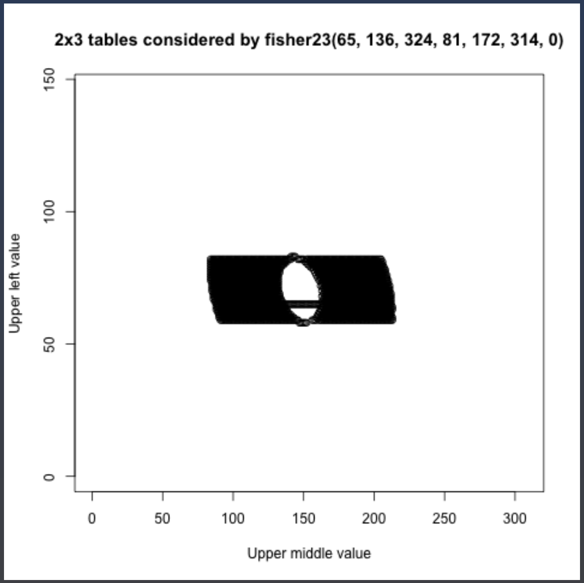
--fisher max(T) permutation test times

10000 permutations; synthetic dataset with 88025 markers, 4000 cases, and 6000 controls

Machine	PLINK 1.07	PLINK 1.90
Mac-2	~890k	49.8
Mac-12	~690k	7.6
Linux32-2	~1300k	393.7
Linux64-512	~720k	13.0
Win32-2	~3600k	208.3
Win64-2	~1700k	35.6

Speedup example: Fisher's exact test

- This approach can be extended to 2x3 and larger tables

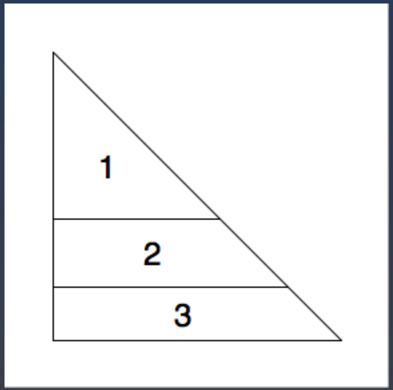


- Javascript tool and standalone source code at

`https://www.cog-genomics.org/software/stats`

Scale-up example: --parallel

- Divides matrix computation into (roughly) equal-size pieces, for later concatenation



```
plink --bfile my_data --make-grm-bin --parallel 1 3
plink --bfile my_data --make-grm-bin --parallel 2 3
plink --bfile my_data --make-grm-bin --parallel 3 3

cat plink.grm.bin.1 plink.grm.bin.2 plink.grm.bin.3 > plink.grm.bin
cat plink.grm.N.bin.1 plink.grm.N.bin.2 plink.grm.N.bin.3 > plink.grm.N.bin
```

Other new features

- Direct VCF/BCF2 import

```
plink --bcf my_data.bcf --out my_plink_data
plink --vcf my_data.vcf --vcf-min-gp 0.9 --out
my_plink_data
```

- Nonstandard chromosome/contig name support

```
plink --bfile mydata --allow-extra-chr …
plink --bfile mydata --aec …
```

Other new features

- LASSO regression

```
plink –bfile my_data --lasso 0.5
```

(why might you want to do this? See poster 1461S, “Applying compressed sensing to genome-wide association studies”, 2-2:30pm.)

Other new features

- Improved command-line help

```
c:\>plink --help indep-pairwise
PLINK v1.90b2m 64-bit (15 Oct 2014)        https://www.cog-genomics.org/plink2
(C) 2005-2014 Shaun Purcell, Christopher Chang   GNU General Public License v3

--indep [window size]<kb> [step size (locus ct)] [VIF threshold]
--indep-pairwise [window size]<kb> [step size (locus ct)] [r^2 threshold]
--indep-pairphase [window size]<kb> [step size (locus ct)] [r^2 threshold]
  Generate a list of markers in approximate linkage equilibrium.  With the
  ‘kb’ modifier, the window size is in kilobase instead of locus count units.
  (Pre-’kb’ space is optional, i.e. ‘--indep-pairwise 500 kb 5 0.5’ and
  ‘--indep-pairwise 500kb 5 0.5’ have the same effect.)
  Note that you need to rerun PLINK using --extract or --exclude on the
  .prune.in/.prune.out file to apply the list to another computation.

--ld-xchr [code]   : Set Xchr model for –indep{-pairwise}, --r/--r2,
...
```

Acknowledgements

Thanks to:

- **Shaun Purcell** for open-sourcing the original program and supporting this evolution
- **Carson Chow**, **Laurent Tellier**, **Shashaank Vattikuti**, and **James Lee** for initial testing and resources
- Numerous alpha and beta testers who've contributed bug reports

Slides, software, preprint, and additional credits at

`https://www.cog-genomics.org/plink2`