Manuscript Title

This manuscript (permalink) was automatically generated from related-sciences/gwas-analysis-manuscript@4202750 on June 2, 2020.

Authors

- **John Doe**
  XXXX-XXXX-XXXX-XXXX · johndoe · johndoe
  Department of Something, University of Whatever · Funded by Grant XXXXXXXX

- **Jane Roe**
  XXXX-XXXX-XXXX-XXXX · janeroe
  Department of Something, University of Whatever; Department of Whatever, University of Something

Abstract

Outline

- Background
  - Scale of current genomic datasets
    - Stats on number of Biobanks
    - Stats on UKBB
    - Stats on growth from HapMap and 1KG for comparison
  - Focus of paper
    - Genotyping array data QC, control flow, and association modeling
    - Scalability problems with single-node, in-memory tools
    - Spark
  - Tutorial
    - Explain necessary GWAS toolkit operations
      - Data management (filtering, merging, etc.)
      - Liftover
      - Summary statistics (HWE, AF, call rates, heterozygosity, etc.)
      - Population stratification (IBS + MDS)
      - Association analysis (logreg, fisher/chisq)
      - Genetic relatedness
    - Introduce Marees 2018 paper and explain how tutorial is re-implemented with other tools
  - Tools
    - Primary: PLINK, Glow, Hail
    - Secondary: dask, bigsnpr, scikit-allel, pysnptools/fastlmm
      - Modin may be worth mentioning, even though out-of-core isn't really supported
  - Datasets
    - HapMap
    - 1KG
    - 3K rice genome
  - Data Formats
    - plink, bgen, parquet, Hail MT, vcf, hdf5/npz/zarr
    - Explain encoding and compression concerns
- Results
  - Code
    - Figure: flow chart of Marees analysis
      - Explain what the resulting code for this project does
    - Figure: side-by-side comparison of code examples
      - Primary differences:
        - Hail is an API over opaque data structures and implementations
        - Glow is simply a convention for representing genetic data in a Spark Dataset, with accompanying methods
        - PLINK is a gigantic list of parameterized commands for a single-core, single-node CLI
      - Operation differences
        - Operations that are similarly easy with all 3 toolkits:
          - call rate filtering
          - heterozygosity rate filtering
          - HWE filtering
          - AF filtering
        - LD Pruning: non-existent in Glow, very slow in Hail
          - However, Glow does support inline calls to PLINK (albeit very awkwardly since PLINK is not streaming software)
        - Gender imputation: PLINK=automatic, Hail=automatic, Glow=manual
          - PLINK and Hail both use inbreeding coefficient on X chromosome data
          - Glow approach (and Marees 2018) look at homozygosity rate on X instead
        - Liftover
          - Not available in PLINK
          - Hail supports coordinate liftover only (not variant liftover)
            - Requires a chain file for the destination reference genome
          - Glow supports both coordinate liftover and variant liftover
            - Requires a chain file and a reference fasta for the destination genome
            - see the notebook at https://glow.readthedocs.io/en/latest/etl/lift-over.html for specification of chain and reference files
        - PCA for population stratification: simple in PLINK and Hail, non-existent in Glow
    - Usability:
      - Extending and learning algorithms from Glow source code is the easiest of all tools

        - Hail documentation is fairly thorough, though essentially no examples or answers are available outside of that documentation (or the discourse)
        - PLINK examples and documentation are both very extensive and easily found, however few PLINK workflows don't also include the need for some scripting language in the same pipeline for interpreting/visualizing results (those outputs are then often used to parameterize other PLINK commands, as exemplified by the tutorials in the project)
        - There are many ways to do the same thing in Hail and it is difficult to know which method to choose (cf. https://discuss.hail.is/t/issues-with-sample-and-variant-qc-by-group/1286/5)
    - Data
        - Figure: File Format Comparison
            - Show comparison of file sizes by dataset and format
    - Performance
        - Figure: Times for operations by dataset and toolkit
        - Touch on vectorization support in breeze (some ops use jni to LAPACK, but simpler ones like sums do not) as compared to numpy
        - Explain benefits of bitpacking (modify 1KG dask nb to have step without GeneticBitPacking filter and compare to original)
- Discussion
    - Computational operations needed in GWAS analyses (see "Computational Operations" in notes)
        - This may be a good place to characterize all operations and what matrix functions support them

Notes

UKBB

- UKB Genotyping and Imputation Data Release March 2018 – FAQ
    - Imputed call set is 2.1T bgen
    - Measured call set is 92G bed
- DETAILS AND CONSIDERATIONS OF THE UK BIOBANK GWAS
    - "Starting from the 487,409 individuals with phased and imputed genotype data, we filtered down to 337,199 QC positive individuals"
    - "We have run 1513 unique phenotypes"
    - "Over 92 million imputed autosomal SNPs were available for analysis. As of this writing, over half of these SNPs were undergoing re-imputation by the UK Biobank team due to mapping issues, leaving the ~40 autosomal million SNPs imputed from the Haplotype Reference Consortium as eligible for analysis. We further restricted to SNPs with minor allele frequency (MAF) > 0.1% and HWE p-value > 1e-10 in the 337,199 QC positive individuals, an INFO score > 0.8 (directly from UK Biobank), leaving 10.8 million SNPs for analysis."
- http://biobank.ctsu.ox.ac.uk/crystal/label.cgi?id=100314
    - Genotypes and imputation therefrom for 488,000 participants
    - Exome sequences for 50,000 participants
    - Whole genome sequences for 50 participants
- Quality control guide: http://www.ukbiobank.ac.uk/wp-content/uploads/2014/04/UKBiobank_genotyping_QC_documentation-web-1.pdf
    - UK Biobank team uses PLINK (for 1KG mostly), shellfish, and flashPCA for QC that is then used by NealeLab
    - This guide is about QC for interm release of 150k samples
        - Results in 152k samples by 806k variants
    - SNP QC is first performed only on European cohort
        - Using a homogeneous cohort is ideal just for initial SNP filters
    - SNPs are filtered based on heterozygosity (adjusted by regression using cohort since heterozygosity differs so much), missingness, and HWE
    - QC is done for each SNP in 4.8k sample batches (matching Affymetrix batches) so a SNP can fail QC in one batch and become all missing
    - SNP QC also done using ROH which is sequential heterozygous call rates
    - Sample QC adjusts for gender (finds .1% are wrong – has interesting aneuploidy figure)
- GPU Acceleration
- Epistatic Interactions
    - SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder
    - 1000× faster than PLINK: Combined FPGA and GPU accelerators for logistic regression-based detection of epistasis
        - Uses reparameterization of Newton's method in PLINK to make interaction model much faster to solve
- IBS/IBD
    - PlinkGPU: A Framework for GPU Acceleration of Whole Genome Data Analysis
        - Ports pairwise IBS distance calculation in PLINK to GPU using block-wise calculations
        - Summary of PLINK capabilities:
            - Data management (filtering, merging, etc.)
            - Summary statistics (HWE, AF, call rates, heterozygosity, etc.)
            - Population stratification (IBS + MDS)

        - Association analysis (logreg, fisher/chisq)
        - Genetic relatedness
- Exome Sequencing Stats
    - The ~50k individuals were called using GATK in joint analysis workflow, which produces gVCF for each sample for a cohort-wide calling to create the final "joint" result (source)
        - The joint results ("project level variant data" as they call it) is 100G PLINK
        - Sample level variant call data is ~5TB gVCF
        - Sample level aligned sequence data is ~50TB CRAM
        - The cited paper within states that there are ~9.6M variants in the final call set, with 4.7M in targeted regions (Table 2)

LMM

- Dealing with related samples in GWAS via lasso (much like LMMs do):
    - A LASSO penalized regression approach for genome-wide association analyses using related individuals: application to the Genetic Analysis Workshop 19 simulated data (2016)
        - "Ignoring relatedness between study participants can have significant impact on the study results and increase false positive rates"
        - "Because of the computational intensity involved in the estimation of the parameters of the LMM, most methods perform single marker analysis"
        - "Lately, least absolute shrinkage and selection operator (LASSO) regression [6] has attracted attention as an alternative tool for selecting the most promising SNPs in GWAS"
        - "Currently, all LASSO methods used in GWAS assume that the sample members are unrelated to each other."
        - One approach to dealing with this in lasso multivariant SNP regression is to:
            - Fit null model with GRM as covariance and then fit lasso to residuals
            - This makes the samples independent
        - The method requires a GRM
        - The model adds an L1 penalty to the standard LMM likelihood with a 0 mean and GRM based covariance random effect
        - Shows that SNPs way more SNPs are identified (non-zero beta) than via univariate methods but that those identified are **NOT** accurate

Ecosystem Tools

- bigsnpr
    - https://privefl.github.io/bigsnpr/articles/demo.html
    - Supports PLINK and UKBB BGEN reading
    - Efficient analysis of large-scale genome-wide data with two R packages: bigstatsr and bigsnpr
    - bigstatsr is related but more generic library for out-of-core matrix ops, PCA, and linear models
    - Supports imputation using custom XGBoost model as well as wrappers for calls out to PLINK + Beagle
- FaST-LMM
    - From 2011 Nature Methods paper FaST linear mixed models for genome-wide association studies
    - PySnpTools
        - https://fastlmm.github.io/PySnpTools/#snpreader-snpreader
        - Can read and write PLINK bed/ped data (it appears to be designed specifically for working efficiently with PLINK data)
        - Supports "DistributedBed" files with chunked PLINK datasets
        - Supports slicing of PLINK datasets (i.e. random IO)
        - Supports IO with npz, hdf5, and memmap files
        - Has efficient c++ reader/writer implementations
- scikit-allel
    - In memory EDA for genotyping data
    - Supports vcf, plink,
    - Has vcf_to_{npz,zarr,recarray} methods
    - Represents genotype calls as 3D uint8 arrays
        - GenotypeArray > This class represents data on discrete genotype calls as a 3-dimensional numpy array of integers. By convention the first dimension corresponds to the variants genotyped, the second dimension corresponds to the samples genotyped, and the third dimension corresponds to the ploidy of the samples.

          Each integer within the array corresponds to an allele index, where 0 is the reference allele, 1 is the first alternate allele, 2 is the second alternate allele, … and -1 (or any other negative integer) is a missing allele call. A single byte integer dtype (int8) can represent up to 127 distinct alleles, which is usually sufficient. The actual alleles (i.e., the alternate nucleotide sequences) and the physical positions of the variants within the genome of an organism are stored in separate arrays, discussed elsewhere.

In many cases the number of distinct alleles for each variant is small, e.g., less than 10, or even 2 (all variants are biallelic). In these cases a genotype array is not the most compact way of storing genotype data in memory. This class defines functions for bit-packing diploid genotype calls into single bytes, and for transforming genotype arrays into sparse matrices, which can assist in cases where memory usage needs to be minimised. Note however that these more compact representations do not allow the same flexibility in terms of using numpy universal functions to access and manipulate data.

- Supports bitpacking by collapsing the ploidy dimension (axis=2) into a single byte using 4 bits for each uint8
- Phasing is supported by assuming the ordering in the ploidy dimension has meaning: > If the genotype calls are unphased then the ordering of alleles along the third (ploidy) dimension is arbitrary
- http://alimanfoo.github.io/2016/06/10/scikit-allel-tour.html > The scikit-allel genotype array convention is flexible, allowing for multiallelic and polyploid genotype calls. However, it is not very compact, requiring 2 bytes of memory for each call. A set of calls for 10,000,000 SNPs in 1,000 samples thus requires 20G of memory.

  One option to work with large arrays is to use bit-packing, i.e., to pack two or more items of data into a single byte. E.g., this is what the plink BED format does. If you have have diploid calls that are only ever biallelic, then it is possible to fit 4 genotype calls into a single byte. This is 8 times smaller than the NumPy unpacked representation.

  However, coding against bit-packed data is not very convenient. Also, there are several libraries available for Python which allow N-dimensional arrays to be stored using compression: h5py, bcolz and zarr. Genotype data is usually extremely compressible due to sparsity - most calls are homozygous ref, i.e., (0, 0), so there are a lot of zeros.

- Dependent Projects:
  - https://github.com/kern-lab/ReLERNN
    - https://github.com/kern-lab/ReLERNN
    - Predicts recombination rate along the genome using mutation rates (in vcf) or allele frequencies
    - https://academic.oup.com/mbe/advance-article/doi/10.1093/molbev/msaa038/5741419
    - Uses vcf_to_hdf5 (source)
    - Uses VariantChunkedTable and GenotypeChunkedArray as out-of-core inputs for RNN prediction (source)
  - https://github.com/kern-lab/locator
    - Predict geographic origin from genomic sequence
    - Uses scikit-allel to construct model inputs, nothing more
  - https://github.com/hardingnj/xpclr
    - Using rogers_huff_r for LD estimation
    - Using utility methods like “is_non_segregating”, “is_singleton”, “count_alleles”, and “compress” (source)
  - https://github.com/ornl-oxford/genben
    - scikit-allel benchmarks library
    - Using scikit-allel PCA (source)
  - https://github.com/Gregor-Mendel-Institute/SNPmatch
    - Using allel.sequence_diversity
  - https://github.com/SaundersLab/FieldPathogenomics
    - All by https://github.com/dnlbunting
    - https://fieldpathogenomics.readthedocs.io/en/latest/
    - Using ChunkedDaskGenotypeArray and allel VariantTable as part of phylogenetic tree pipeline (source)
    - Uses locate_unlinked as part of STRUCTURE implementation (source)
    - Using rogers_huff_r LD estimation

- SNPRelate
  - List of functions
  - Authors also created gdsfmt, which is the data structure they use within the library
    - This is basically an R version of zarr, not a genetics-specific file format
  - They also created SeqArray
    - This is for whole exome/genome sequencing data
    - It is an extension to GDS
  - Benchmarks show the uniprocessor implementations of PCA and identity-by-descent are ~8–50 times faster than the implementations provided in the popular EIGENSTRAT (v3.0) and PLINK (v1.07) programs, respectively, and can be sped up to 30–300-fold by using eight cores
  - tutorials: http://bioconductor.org/packages/release/bioc/vignettes/SNPRelate/inst/doc/SNPRelate.html
    - These cover:
      - LD pruning
      - PCA
      - Fst estimation (fixation index)
        - This is some statistic that summarizes population structure (given two or more populations as inputs)
        - Higher Fst means that the population is more differentiated than one with lower Fst (source)

        - Compare to scikit-allele Fst Estimation
        - From wikipedia:
            - "The values range from 0 to 1. A zero value implies complete panmixis; that is, that the two populations are interbreeding freely. A value of one implies that all genetic variation is explained by the population structure, and that the two populations do not share any genetic diversity"
    - Kinship estimation
        - IBD PLINK
        - IBD Maximum Likelihood
        - KING method of moments
    - IBS
        - Uses MDS over IBS matrix to show population clustering
- http://bioconductor.org/packages/release/bioc/html/SNPRelate.html
- KING
    - https://github.com/zhengxwen/SNPRelate/blob/master/src/genKING.cpp
        - Has c++ implementations of King homo (population homogeneity) and King robust (provides robust relationship inference in the presence of population substructure)
- IBD (PLINK port it seems)
    - https://www.rdocumentation.org/packages/SNPRelate/versions/1.6.4/topics/snpgdsPairIBD
    - https://github.com/zhengxwen/SNPRelate/blob/master/R/IBD.R
    - https://github.com/zhengxwen/SNPRelate/blob/ac01cbaca760228def7342261d7eed5e8bdbcd20/src/genIBS.cpp#L556
- LD pruning
    - https://github.com/zhengxwen/SNPRelate/blob/master/src/genLD.cpp

LD pruning

- https://hail.is/docs/0.1/hail.VariantDataset.html#hail.VariantDataset.ld_prune
- Hail discussions on ld prune implementation:
    - https://discuss.hail.is/t/ld-prune-implementation-in-0-1/244
    - https://dev.hail.is/t/seeking-some-input-on-current-implementation-of-ld-prune-method/78/11
- Hail 0.2 implementation (https://hail.is/docs/0.2/_modules/hail/methods/statgen.html#ld_prune)
    - Local LD prune implementation: https://github.com/hail-is/hail/blob/master/hail/src/main/scala/is/hail/methods/LocalLDPrune.scala#L227
    - Summary of Hail 0.2 implementation:
        - A greedy strategy first prunes using double loop for all variants in a single partition and considers only R2 (no MAF) so the choice is arbitrary
            - This is what is ultimately returned, though everything that follows is intended to identify correlated variants spanning partitions that need to be removed
        - Next, a giant BlockMatrix multiplication with its transpose used to get R2 across all partitions
        - This matrix is than sparsified based on row intervals (see sparsify_row_intervals)
        - The matrix is collapsed to entries in processed further as (variant i, variant j, r2)
        - Variants *above* the r2 threshold are kept are kept and fed into a maximally independent set finder
            - MIS is done in memory on a single machine (see Graph.scala)
        - The final set of variants to remove is filtered out of the result from step 1
    - Operations needed to recreate this:
        - Matrix transpose + multiplication
            - This is how the LD calculation works
        - Matrix + vector subtraction with broadcasting and element-wise matrix multiplication
            - The sparsification of the LD matrix is done by determining which variants are within some number of bp from each other
            - This is done in a custom Scala routine, but it could also be expressed as a subtraction of genomic positions to create a mask matrix
            - The mask matrix could then be multiplied by the LD matrix to get the correct banding
        - Matrix enumeration (rows, cols, or entries)
            - In the Hail case, enumeration is used to collapse a sparse LD matrix, filter it per entry, build a list of graph edges, and then run a custom scala routine to find maximal independent sets
            - If Privé's clumping was used instead, there is a way to express everything with per calculations on the sparsified LD matrix along with distributed sorting and filtering (having an upfront "importance" metric to sort by makes it much easier to do without any single-node algorithm bottlenecks)
- https://www.cog-genomics.org/plink/1.9/ld
- Actual pseudo code from C Chang on how ld pruning works (2015):
    - https://groups.google.com/d/msg/plink2-users/w5TuZo2fgsQ/WbNnE16_xDIJ
- location of ld_prune for PLINK 1.9: https://github.com/chrchang/plink-ng/blob/master/1.9/plink_ld.c
- An earlier conversation (2014) on the ld pruning implementation:
    - https://groups.google.com/forum/#!topic/plink2-users/wlSG51SdbFE
        - This suggests the old algorithm was different and marked snps for some kind of greedy selection
- In PLINK:
    - (PLINK 1.07) https://wlz0726.github.io/2017/05/27/LD-prune-with-plink/

        - a. consider a window of 50 SNPs
        - b. calculate LD between each pair of SNPs in the window
        - b. remove one of a pair of SNPs if the LD is greater than 0.5
        - c. shift the window 5 SNPs forward and repeat the procedure
    - How to make PLINK use scores other than MAF: see here
        - `--read-freq` is the opportune param
- Other methods:
    - SNPrune
        - Fast, very domain-specific technique for finding highly correlated variants (LD >= .99)
- Maximal independent set notes
    - From paper on Luby algorithm (https://www.cs.utah.edu/~hari/teaching/bigdata/SICOM86-LubyM-Parallel.Algorithm.Graph.Maximal.Independent.Set.pdf)
        - This is commonly cited
        - "The obvious sequential algorithm for the MIS problem can be simply stated as: Initialize I to the empty set; for 1, , n, if vertex is not adjacent to any vertex in I then add vertex, to I. The MIS output by this algorithm is called the lexicographically first maximal independent set (LFMIS)."
            - This is the PLINK algorithm (if all MAFs are the same)
        - "Valiant [Va] noted that the MIS problem, which has such an easy sequential algorithm, may be one of the problems for which there is no fast parallel algorithm."
    - This paper in Algorithm 1 describes the LFMIS algorithm well
        - http://people.cs.georgetown.edu/~jfineman/papers/greedymis.pdf
    - Blelloch's iterative algorithm (parallel) describes LD pruning well where the dependency structure in the sequential selection problem is very shallow (due to the windows)
        - https://arxiv.org/pdf/1202.3205.pdf
        - "Luby's randomized algorithm [17] ... can be converted to run in linear work. The problem, however, is that on a modest number of processors it is very hard for these parallel algorithms to outperform the very simple and fast sequential greedy algorithm. Furthermore the parallel algorithms give different results than the sequential algorithm"
    - Blelloch citers:
        - https://www.semanticscholar.org/paper/A-High-Quality-and-Fast-Maximal-Independent-Set-for-Burtscher-Devale/5e4866a895eb947eaf6f0ecd0556e83745630b54
        - Tight Analysis of Parallel Randomized Greedy MIS mentioned by Blelloch in Theoretically Efficient Parallel Graph Algorithms Can Be Fast and Scalable as the most recent work improving on his bounds that the number of iterations is O(log n) rather than O(log^2 n)
    - Parallel algorithms for the maximal independent set problem in graphs
        - "The greedy sequential algorithm to compute a MIS loops over the vertices according to the given order, adding them to the resulting set only if no previous neighboring vertex has been added."
        - "If this structure is shallow (has a polylogarithmic depth), then running each of the iterates in parallel, while respecting the dependencies,leads to an efficient parallel implementation that mimics the sequential algorithm. The depth of this dependance structure is called the dependence length"
    - MIS on GPU presentation
        - Paper: A High-Quality and Fast Maximal Independent Set Implementation for GPUs
    - Effects of LD pruning on downstream analysis
        - Sensitivity analysis with LD pruning for epistasis screening: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6558841/

PCA

- Projecting samples using pre-computed PCs
    - Hail does not have this but McArthur lab has examples (see discourse post and linked code)
- On how loadings can be used to identify poor PCA results
    - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2912642/
    - "Correlated SNPs will therefore have high loadings, because correlated random variables can generate linear combinations with high variability. As we demonstrate, the net effect is to give higher weight to groups of correlated SNPs, although there is little reason to believe that such SNPs will perform well in differentiating among subpopulations. An intermediate goal, therefore, is to eliminate the distorting effect of the redundant information provided by groups of highly correlated SNP genotypes."
- HWE normalization
    - https://doc.goldenhelix.com/SVS/latest/svsmanual/ftParts/pca.html#formulas-for-pca-normalization-of-genotypic-data
    - Explained in Population Structure and Eigenanalysis
- Identifying outlier SNPs in loadings
    - Peaks in loadings may indicate LD structure captured by PCA
    - A Practical Approach to Adjusting for Population Stratification in Genome-wide Association Studies: Principal Components And Propensity Scores (PCAPS)
        - "Most importantly, patterns of local linkage disequilibrium (LD) may cause PCA to create "nuisance axes", which may be interpreted as the existence of subpopulations that reflect localized LD phenomena rather than plausible PS (Zou et al. 2010)"
    - They may also indicate natural selection underway within multiple populations
        - pcadapt: an R package to perform genome scans for selection based on principal component analysis

        - Regresses PCs against each SNP to see which SNPs aren't explained by PCA
        - "The first statistic based on PCA was the communality statistic, which measures the percentage of variation of a single nucleotide polymorphism (SNP) explained by the first K principal components"
    - One interpretation appears to be that SNPs not captured by PCs may represent selection
        - Detecting Genomic Signatures of Natural Selection with Principal Component Analysis: Application to the 1000 Genomes Data
- From Population Structure and Eigenanalysis: - Defines a test for stating whether population structure exists in a bi-allelic dataset - Also gives a test to answer the question "Does the data show evidence of additional population structure over and above what has already been detected?"
- Types of population structure (Li et al. 2010)
    - Discrete
    - Admixed
    - Hierarchical
- PCA Loadings
    - Genome-wide SNP and haplotype analyses reveal a rich history underlying dog domestication (vonHoldt 2010)
        - Uses small numbers of highly ranked SNPs (by loading) to uniquely define genetic differences between dogs and grey wolves (supplemental table 2)
- Phylogenetic Analysis
    - See Jombart et al. 2010 (ppca which uses PCA and clustering to identify ancestry
- Scaling
    - Approximate truncated SVD: O(mnk) (k = rank)
    - Randomized truncated SVD: (mn log(k))
    - From https://arxiv.org/pdf/0909.4061.pdf (via https://discourse.related.vc/t/pca-implementations/224)
        - "A standard deterministic technique for computing an approximate SVD is to perform a rank-revealing QR factorization of the matrix, and then to manipulate the factors to obtain the final decomposition. The cost of this approach is typically O(kmn)"
        - "For a dense input matrix, randomized algorithms require O(mn log(k)) floating-point operations (flops) in contrast with O(mnk) for classical algorithms"

Kinship Estimation

- The methods below are all essentially different estimators for the exact same statistic, a kinship coefficient (often denoted as phi_hat)
    - KING PropIBD
    - Hail pc_relate "kin"
    - Hail identity_by_descent "PI_HAT"
    - PLINK –genome PI_HAT (as in KING, this is P(IBD2) + .5 * P(IBD1))
    - PLINK –make-rel (for GRM), Hail GRM/RRM, and GCTA
        - According to the PC-Relate paper and Goudet 2018, these are all actually estimating the same thing as summed IBD probabilities
        - PC-Relate also defines the the GRM as: "The entries in this GRM measure the genotype correlations for pairs of individuals"
        - **Side note**: This assertion, that the GRM estimator (which is a correlation in [-1, 1]) is asymptotically equal to IBD probability sharing (a probability in [0, 1]) was unexpected for me, but definitely helpful to keep in mind when trying to parse the intention of all these different methods.
            - Appendix A (Empirical Genetic Relationship Matrix) shows this proof along with the disclaimer that it is only true in a homogeneous population
- Summary of PC-AiR and PC-Relate from GENESIS package: https://www.bioconductor.org/packages/release/bioc/vignettes/GENESIS/inst/doc/pcair.html#principal-components-analysis-in-related-samples-pc-air
- Summary table of methods with the population characteristics they support unbiased estimation with:

Estimator	LD	Pop Structure	Recent Admixture
GRM (Hail/PLINK)			
IBD (Hail/PLINK)			
KING	X	X	
PC-Relate (Hail)	X	X	X

- These characteristics above are difficult to account for because they create so many circular dependences; for example:
    - GRM/IBD require homogeneous populations but to get these from a sample with mixed ancestry a method like PCA must be used which is most effective when related individuals are removed (which you can't do without GRM/IBD)
    - PC-Relate also has this issue, and tries to overcome it by running KING as a first pass before running PCA and then producing structure-adjusted kinship estimates (though the Hail implementation seems to just ignore that initial step)
- The kinship coefficient is sometimes referred to as the coefficient of ancestry; this should not be confused with coancestry coefficient which increases when there are more distant common ancestors

    - The GRM estimator of coefficient is equal (asymptotically) to the real kinship coefficient + coefficient of ancestry - some function of recent coancestry (see PC-Relate paper)
        - This helps to understand the relationship between the GRM estimator and population structure – values are inflated when such structure exists
- Related quantities with trivial transformations to and from kinship coefficient (phi_hat)
    - Coefficient of inbreeding
        - Equal to 2 * phi_hat - 1
    - Coefficient of relationship
        - Equal to 2 * phi_hat
- KING
    - Should be run with rare and correlated variants
    - “Please do not prune or filter any”good" SNPs that pass QC prior to any KING inference, unless the number of variants is too many to fit the computer memory, e.g., > 100,000,000 as in a WGS study, in which case rare variants can be filtered out. LD pruning is not recommended in KING."
- Hail pc_relate
- Gnomad v2 QC
    - “Relatedness filtering for this release was done very similarly to the 2.0.2 release. The only difference is that instead of using KING, we used the pc_relate method in Hail to infer relatedness”
- Gnomad v3 QC
    - “We used Hail pc_relate to compute relatedness, followed by Hail maximal_independent_set in order to select as many samples as possible, while still avoiding including any pairs of first and second degree relatives”
    - Application of pc_relate is here
- PC-Relate
    - Summary
        - Many initial methods for relatedness do not account for population structure including:
            - Kinship coefficients
            - The –make-rel method in PLINK
            - PLINK cites GCTA as having the definition for relatedness, so presumably that also suffers from this
            - The method used in Hail GRM and RRM (which uses the same GCTA definition)
        - KING can handle distinct populations but breaks down when there is recent admixture in the immediate ancestry of a pair of samples
            - i.e. If you have perfectly pure subpopulations there are no problems, but if two siblings have a European and Asian parent, it stops working
        - PC-Relate can handle recent and distant relatedness
        - The same authors developed PC-AiR
            - PC-Relate uses this to get an initial set of PCs that are then regressed against genotype calls for individuals
            - Predictions from this model are then used to create allele frequencies used in the usual GRM formula
    - “Existing approaches for the estimation of frequently used measures of recent genetic relatedness, such as kinship coefficients and identity by descent (IBD) sharing probabilities, have limitations in the presence of population structure”
    - “For example, a variety of maximum likelihood and method of moments estimators have been developed for relatedness inference from genotype data under a strong assumption of sampling from a single population with no underlying ancestral diversity. In samples with population stratification, these methods that assume population homogeneity have been shown to give extremely biased estimates of recent genetic relatedness.”
    - “The widely used KING-robust method11 has been developed for inference on close pedigree relationships under an assumption of sampling from ancestrally distinct subpopulations with no admixture. However, KING-robust gives biased relatedness estimates for pairs of individuals who have different ancestry, which can result in incorrect relationship inference for relatives with admixed ancestry.”
    - The section “Convolution of Recent and Distant Genetic Relatedness” defines the GRM formula (same as used in Hail/PLINK)
    - “A widely used empirical genetic relationship matrix (GRM) has been proposed for inference on population structure (distant genetic relatedness) in samples without close relatives, as well as inference on recent kinship and heritability estimation of complex traits in samples derived from a single population”
    - Recent vs Distant genetic relatedness
        - From PC-Relate Presentation
            - Distinguishing familial relatedness from population structure using genotype data is difficult, as both manifest as genetic similarity through the sharing of alleles.
            - It is important to note that relatedness and ancestry are a continuum
            - On KING: “A limitation of the method is that it gives biased kinship estimates for individuals with different ancestry, including close relatives who are admixed.”
    - The GRM calculation is asymptotically equal to kinship coefficient (see right after Equation 2)
- PC-Relate in Hail
    - https://github.com/hail-is/hail/issues/3490
        - This thread talks the PC relate implementation and issues with it
            - PC-AiR uses KING to get an initial unrelated set but “Hail’s version of pc-relate does not identify an initial set of related and unrelated individuals. The R pcrelate implementation (the official / reference implementation by the authors of the paper) does this to identify a set of individuals on which to run the principal components analysis. It is not entirely clear to me why this is necessary, and we don’t currently have a mechanism for doing so (since

pc_relate is our mechanism for determining related and unrelated individuals when there is population structure in the data set). If you have prior knowledge about related samples, you might try filtering to an known unrelated set and computing the scores from that set. I'm curious if that makes any difference in the results."
        - "Specifically, PCA is better run after first removing related individuals which is problematic as this is the method that should determine if they're related in the first place"
        - "A longer term solution is to simply implement PC-AiR in hail. I skimmed the implementation section of the paper earlier this week and it looks very straightforward. It seems to boil down to using the KING estimator to estimate relatedness, compute PCA on unrelated individuals, project related individuals into unrelated PC space. Finally, we can use pc_relate to improve on our original estimates of relatedness from KING."
    - https://dev.hail.is/t/linear-algebra-syntax-example-pc-relate/131
        - This has a helpful discussion on matrix operations needed for pc_relate
- PLINK
    - See https://www.cog-genomics.org/plink/1.9/distance (Relationship/covariance)
        - `--make-rel` is realized relationship matrix (presumably same as Hail RRM) and cites GCTA: A Tool for Genome-wide Complex Trait Analysis as the reference implementation (this paper has the formula in the Hail docs)
    - PI_HAT from `--genome` is kinship coefficient estimator
    - Homogeneous population requirement
        - From original PLINK paper:
            - "In homogeneous samples, PLINK provides options to estimate genomewide IBD-sharing coefficients between seemingly unrelated individuals from whole-genome data"
        - From v1.07 docs:
            - "As mentioned above, the IBD estimation part of this analysis relies on the sample being reasonably homogeneous – otherwise, the estimates will be biased (i.e. individuals within the same strata will show too much apparent IBD)"
    - Description of PLINK IBS (not IBD) implementation:
        - https://arxiv.org/pdf/1410.4803.pdf
        - This has a good pseudocode description in "Improvements in PLINK 1.9"
            - The calculation loops through every possible sample pair and for each variant for that pair, increments IBS{0,1,2} based on comparison of calls
    - Description of PLINK IBD implementation:
        - https://www.biorxiv.org/content/10.1101/103325v1.full.pdf
        - "PLINK computes for each pair of individuals an IBS score and uses a hidden Markov Model (HMM) method to find IBDs. The hidden IBD state is estimated by the computed IBS sharing and genome-wide level of relatedness. PLINK is slow because even the first step of computing all IBS scores requires O(n^2) operations for n samples."
- Papers with good intro surveys of kinship estimation
    - How to estimate kinship
        - Mentions VanRaden 2008 as first publication with GRM estimator
        - "The kinship, or coancestry, coefficient θ jj' for individuals j and j' is defined here as the average of the four ibd probabilities for one allele from each individual"
    - Quickly identifying identical and closely related subjects in large databases using genotype data
        - "The maximum-likelihood estimators always generate biologically meaningful probabilities and are usually more accurate than the method-of-moment estimators. However, maximum-likelihood approaches are usually slower and sometimes more biased than the method-of-moment ones. Method-of-moment methods usually use the observed numbers of IBS sharing loci instead of the predicted numbers when calculating IBD sharing probabilities, and hence may yield estimates that cannot be interpreted as probabilities, in which case, researchers truncate the estimates into the meaningful range [0, 1]. The truncation of the results introduces artificial effects and biases."
        - This paper cites PLINK as one of the method-of-moments estimators (rather than maximum likelihood)
    - Efficient Estimation of Realized Kinship from Single Nucleotide Polymorphism Genotypes
        - 2017
        - Says that "pedigree kinship" is what all "realized kinship" methods are trying to estimate (the former is a deterministic function of the true pedigree)
        - Explains "realized kinship" estimation methods like PLINK, GRM, and many variants in paragraph 4 of intro
        - The paper simulates data from 1KG using ibd_create of MORGAN software
        - Table 1 is a terrific summary of why GRM estimates are so much worse than IBD via method of moments or MLE
            - Interestingly, it also includes the self-reported pedigree data
- Comparisons of methods
    - GRM vs RRM
        - The only difference is that variants are divided by binomial variance (which assumes HWE) in the GRM calculation while they are divided instead by the emprical variance in the RRM
    - RRM vs Pearson correlation
        - They are the same except that the denominator in the RRM standardization is multiplied by the number of variants (not sure why)
    - GRM vs IBD from method of moments (PLINK) or MLE
        - First, they are the same in expectation
        - However, the GRM estimates have much higher variance (Wang et al. 2017)
- Using imputed variants to inform kinship estimation
    - https://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1007021

- UK Biobank
  - According to Simultaneous SNP selection and adjustment for population structure in high dimensional prediction models:
    - 147k people are related in UKB and only 18k have a documented relationship

Ascertainment bias

- SNP arrays are designed using an "ascertainment sample" and it is this sample that determines what SNPs make it onto the array
- As these SNPs are often filtered to only this with AF in a certain range, it is possible for many rarer variants in such a small sample to be ignored (ascertainment bias)
- There are ways to try to adjust for this, but it is not possible when the "ascertainment sample" does not match the ancestry of the "typed sample" (the one an experiment is being run on)
  - From Population genetic analysis of ascertained SNP data (Nielson 2004):
    - "In cases where there is little or no overlap between the ethnicities of the individuals included in the typed sample and the ascertainment samples, however, corrections can only be made in parametric models describing the genetic relationship between the populations. In such cases, it will typically be difficult or impossible to use classical non-parametric methods for statistical inference."

Positive Selection Statistics

- Selective Sweep
  - Occurs when a beneficial mutation becomes fixed (AF of 1) and causes nearby mutations to become fixed as well even if they don't confer some benefit
- Generally, speaking these statistics and methods characterize the selective pressure related to an allele in a population
- Homozygosity and linkage disequilibrium
  - Haplotype homozygosity - the probability of selecting two identical haplotypes at random from a population
  - This is different from genotype homozygosity in that:
    - It is defined by two loci instead of one (though you could still think of it in terms of two haplotypes from the same person at the same loci)
    - It seems to only consider when the allele is present (i.e. "homozygous alternate" is what this term refers to)
    - For example, for two SNPs with some number of alleles each (not necessarily 2), the probability of of identical haplotypes selected at random is the product of the AF for each (not the product of AF for both present + product of 1-AF for each)
      - See figure 1 for this definition
- Haplotype Homozygosity and Derived Alleles in the Human Genome
  - Common LD statistics like D' and r2 do not differentiate between the alleles at a single locus
    - This means that it doesn't matter if the allele is present or absent, only that it correlates with others
  - On gauging the age of an allele in a population subject to **POSITIVE** selection:
    - A new allele from a mutation will start on a single haplotype
    - Over time, LD between this allele and those nearby will decay (from recombination)
    - This "decay in LD" is "stopwatch by which its age can be estimated"
  - NOTE: This kind of method may be useful for characterizing when a protective allele for a disease arose
- A Map of Recent Positive Selection in the Human Genome
  - Defines iHS (integrated haplotype score)
  - Authors state that a **"selection map" of ongoing sweeps** in the human genome are an important genome annotation that should be considered in GWAS studies
  - in scikit-allele: allel.ihs
  - Builds on extended haplotype homozygosity (EHH)
    - "The EHH measures the decay of identity, as a function of distance, of haplotypes that carry a specified "core" allele at one end"
    - Figure 1B is the best explanation of why haplotype homozygosity is more informative than LD estimates
      - It allows comparing an ancestral and derived allele in terms of LD around them
    - This works by starting with a "core" allele (one of interest) and moving to adjacent alleles one at a time, with increasing distance, and determining haplotype homozygosity with each
      - For a derived allele undergoing positive selection, you should see that the homozygosity decreases more slowly (i.e. LD is higher) as distance increases
      - This is because presumably the derived allele is causing the others to be inherited together more often
  - Properties of iHS:
    - "The iHS at each SNP measures the strength of evidence for selection acting at or near that SNP"
    - "The iHS is constructed to have an approximately standard normal distribution and hence the sizes of iHS signals from different SNPs are directly comparable regardless of the allele frequencies at those SNPs"
    - "it does not provide a formal significance test"
    - Controls for recombination rate using "LD patterns and the estimated genetic distances"
    - Has better power than "Fay and Wu's H, and Tajima's D" (cf. allel.moving_delta_tajima_d)
  - Calculation:
    - "iHS is computed for every SNP with minor allele frequency > 5%, treating each SNP in turn as a core SNP"

- Inference:
  - As an example, the authors mention that alleles favoring lighter skin in Europeans are undergoing positive selection
  - Similar examples are shown for reproductive cell fitness, metabolism, and neurological attributes

WGS/WES Operations

- Hail
  - Splitting multi-allelic variants using common HTS (high-throughput seqeuencing) fields: split_multi_hts
    - Requires the following entry fields, some of which are transformed in the downcoded biallelic variant results:

```
struct {
  GT: call,
  AD: array<int32>,
  DP: int32,
  GQ: int32,
  PL: array<int32>,
  PGT: call,
  PID: str
}
```

Long Range LD

- From bigsnpr.clumping.R: Long-Range LD Can Confound Genome Scans in Admixed Populations.

File Formats

- BGEN
  - BGEN: a binary file format for imputed genotype and haplotype data
  - https://www.cog-genomics.org/plink/2.0/formats#bgen
  - https://www.well.ox.ac.uk/~gav/bgen_format/
  - https://www.well.ox.ac.uk/~gav/bgen_format/spec/latest.html

SIMD and Native Libraries

- Hail uses breeze (https://github.com/hail-is/hail/blob/de2968e0bf9215e058b1fbace4ae618cc93463fe/hail/src/main/scala/is/hail/expr/ir/BlockMatrixIR.scala)
- Breeze uses netlib-java which is a wrapper for BLAS/LAPACK/ARPACK (https://github.com/scalanlp/breeze/wiki/Breeze-Linear-Algebra#performance)
  - Breeze uses LAPACK for some ops like svd but not others like sum
  - Sums along an exist apparently don't exist in LAPACK (see this post)
  - Matrix multiplication (DenseMatrixOps) seems to use BLAS though
- Hail says atlas should be installed for native linear algebra: https://hail.is/docs/0.2/getting_started.html#blas-and-lapack
- SIMD support doesn't really seem to exist in spark 2.x, or at least this jira ticket states that companies are hacking this functionality in now (3.x might have it)
- A decent explanation of the relationship between breeze and LAPACK/BLAS: http://izmailoff.github.io/ml/neural-network-from-scratch-in-scala/
- On Atlas vs BLAS vs LAPACK: "ATLAS is a portable reasonably good implementation of the BLAS interfaces, that also implements a few of the most commonly used LAPACK operations."
- An OpenJDK vector API seems like it would provide SIMD op access, though it is still incubating as of JDK13 (example)
- Good overview of vectorization in Spark: https://www.waitingforcode.com/apache-spark-sql/vectorized-operations-apache-spark-sql/read
- Atlas vs OpenBLAS vs MKL
  - default linux blas & lapack are significantly slower than the others, with MKL being slightly better than openBlas and Atlas
- Checking numpy/scipy blas
  - See: https://stackoverflow.com/questions/37184618/find-out-if-which-blas-library-is-used-by-numpy/37190672

```
ldd /opt/conda/envs/hail/lib/python3.7/site-packages/numpy/core/_multiarray_umath.cpython-37m-x86_64-linux-gnu.so
> libcblas.so.3 => /opt/conda/envs/hail/lib/python3.7/site-packages/numpy/core/../../../../libcblas.so.3 (0x00007f0b737e1000)
readlink -e /opt/conda/envs/hail/lib/python3.7/site-packages/numpy/core/../../../../libcblas.so.3
> /opt/conda/envs/hail/lib/libopenblasp-r0.3.7.so
```

- np.__config__.show() gives paths to MKL used at build, but actual binaries are linked to openblas
- This may fix it: https://github.com/conda-forge/numpy-feedstock/issues/153
  - Add `blas=*=openblas` and `conda-forge::numpy` to environment.yaml
- For scipy:

```
ldd /opt/conda/envs/hail/lib/python3.7/site-packages/scipy/linalg/_fblas.cpython-37m-x86_64-linux-gnu.so
>> libopenblasp-r0-2ecf47d5.3.7.dev.so => /opt/conda/envs/hail/lib/python3.7/site-packages/scipy/linalg/../.libs/libopenblasp-r0-2ecf47d5.3.7.dev.so (0x00007f13d0f56000)
```

C/G and A/T SNPs

- This QC step is common because the probe sequence on genotyping arrays is not specific to a reference genome, by design. This means that the genotyping data can tell you that an "A" nucleotide was present at a locus but it doesn't actually know if this nucleotide represents some kind of "variant" with respect to a larger population. The probes are chosen such that they capture individual sites of common variation but deciding which nucleotides comprise heterozygous, homozygous ref, homozygous alt genotypes (i.e. make a call) is up to the user. For any given site, the arrays capture multiple individual nucleotides so one way to do this independent of a reference genome, for a single dataset, is to simply assume that whatever nucleotide is less common is the minor (aka alternate) allele and the other is the major (aka reference) allele. This is an acceptable (and very common) method for analyzing a single dataset but causes obvious problems when trying to compare calls for the same variants between datasets (a nucleotide may have been the alternate in one and reference in the other). Two strategies for making datasets comparable then are:
  1. For each dataset, use knowledge of the probe sequences to determine what strand each nucleotide is on. This appears to be the only completely unambiguous way to ensure that all calls correspond to the same reference and alternate nucleotides.
  2. Use the fact that the SNP arrays at least tell you which nucleotides were measured for each locus to infer, in a fairly quick and dirty way, which strand the probes measured in each dataset. Here are some examples to make this more clear. Let Dataset 1 = D1 and Dataset 2 = D2 in each of these and assume each determine major/minor aka ref/alt alleles based on frequency (where "AC" implies A was designated as the major allele and C as the minor):
     - **Example 1 (AC vs CA)**: D1 says a variant has A as the major allele and C as the minor allele. D2 says C is major and A is minor
       - Correction: For all calls in D2 for this variant, switch the homozygous/heterozygous interpretation (presumably the C allele was more common in D2 but not D1)
     - **Example 2 (AC vs TG)**: D1 says variant has A = major, C = minor and D2 says T = major, G = minor
       - Correction: Nothing for the calls. The probes in this case were for different strands but ultimately captured the same nucleotides (since A is complementary with T and C with G) AND assumed the same major/minor relationship. The allele nucleotides in D2 should be complemented so that the variant is known as AC, but that's it.
     - **Example 3 (AC vs GT)**
       - Correction: As a combination of example 1 and 2, the call interpretation and allele nucleotides should both be swapped in D2 to align with D1.
     - **Example 4 (AT vs TA)**
       - This is where things get tricky. In example 2, we knew that the probes measured different strands simply because A or C nucleotides would be on one strand while T and G nucleotides would be on the other. This definitive knowledge of a strand swap is key. In the AT vs TA case, it could be that the probe measured different strands or it could be that the same strand was used but alleles occurred at different frequencies in both datasets. We could now say something like, "If the A allele has a frequency of 5% in D1 and it has a frequency of ~5% in D2, then we can safely assume that the same strand was used for the probe". This, however, becomes problematic as the allele frequencies 50%. The same is true for cases like CG vs GC or even AT vs AT – you simply can't tell which strand the probes corresponded too without knowledge of the probe seqeuences. These sequences could be compared between the two datasets to determine if they were for the same strand, but they appear to be difficult to come by. This is the main reason why many analyses simply through out A/T and C/G SNPs.
- Here are some helpful discussions/papers on why this step is necessary (and on strand ambiguity in general):
  - StrandScript: evaluation of Illumina genotyping array design and strand correction > Additionally, the strand issue can be resolved by comparing the alleles to a reference genome. Yet, when two alleles of the SNPs are complementary (A/T or C/G), the true strand remains undetermined. The only absolute solution to determine the strand is to compare the probe sequences to a reference genome, providing the probe sequences is correct
  - Genotype harmonizer: automatic strand alignment and format conversion for genotype data integration > However, there are two major challenges to be resolved: ... 2) the ambiguous A/T and G/C single nucleotide polymorphisms (SNPs) for which the strand is not obvious. For many statistical analyses, such as meta-analyses of GWAS and genotype imputation, it is vital that the datasets to be used are aligned to the same genomic strand.
  - Is 'forward' the same as 'plus'?... and other adventures in SNP allele nomenclature

bigsnpr/bigstatsr

- In trying to understand how the loadings plots were generated in https://privefl.github.io/bigsnpr/articles/how-to-PCA.html, I found these useful source links:

- loadings plot function: https://github.com/privefl/bigstatsr/blob/810347dfdbc7b625f0c2907e45e111764c47da8c/R/plot.R#L158
- big_randomSVD: https://github.com/privefl/bigstatsr/blob/master/R/randomSVD.R#L189
- what big_randomSVD calls: https://github.com/privefl/bigstatsr/blob/master/R/randomSVD.R#L105
- where big_SVD class is defined: https://github.com/privefl/bigstatsr/blob/c859ad28d9c6f8c8cc365c3315e3abbb81e128a8/R/SVD.R#L92

Association Analysis

- Nealelab original gwas uses linear regression for ALL phenotypes (binary, ordinal, or continuous)
  - See: http://www.nealelab.is/blog/2017/9/11/details-and-considerations-of-the-uk-biobank-gwas
  - "Model misspecification: While normally distributed quantitative traits are suited for linear regression models, binary traits are better suited to a logistic model, and the linear assumptions can create biases in the beta coefficients and significance, which we address below."

Dask

- Arrays created from numpy arrays in memory have all their contents hashed (so this is unexpectedly slow)
  - See: https://github.com/dask/dask/issues/3946#issuecomment-418568246
  - Arrays from files are assigned a name based on filename, modification time, etc.

Canines

- From Boxer bares all (2005)
  - "All domestic dogs are descended from grey wolves (C. lupus) that were tamed about 15,000 years ago"
  - "Today, researchers published the full genetic code of a 12-year-old boxer named Tasha"
  - The actual publication on the first full genome is Genome sequence, comparative analysis and haplotype structure of the domestic dog.
    - Nature Link
- From Genome sequence, comparative analysis and haplotype structure of the domestic dog
  - On choosing a boxer: "This particular animal was chosen for sequencing because it had the lowest heterozygosity rate among ~120 dogs tested at a limited set of loci; subsequent analysis showed that the genome-wide heterozygosity rate in this boxer is not substantially different from other breeds (Parker 2004)"
    - The "subsequent analysis" is Parker, H. G. et al. Genetic structure of the purebred domestic dog

TODO

- Consider using OpenBLAS and MKL for perf testing since Chris Chang mentioned that it can really improve PCA performance

References