glactools: a suite of utilities for the management of allele frequency information

Gabriel Renaud
gabriel [dot] reno [at] gmail.com

Abstract

glactools is a set of command-line utilities to store genotype likelihoods and allele count data. It can import data from various formats, merge/-filter/split datasets, compute summary statistics and exporting data to various formats.

Contents

1 Introduction

glactools aims at storing either :

- genotype likelihoods for a single diploid individual for autosomal data (GLF format).
- allele count information for either a given individual or population (ACF format)

imported from various input sources (VCF, BAM etc), creating intersections, querying the data and exporting it to various formats used by current software tools.

1.1 Installation

Please refer to the README for downloading and installing the software.

Program	Use
Data import	
vcf2acf	Convert single sample VCF to acf
vcf2glf	Convert single sample VCF to glf
vcfm2acf	Convert multi sample VCF to acf
bam2acf	Convert single sample BAM to acf
axt2acf	Convert AXT alignment to acf
23andme2acf	Convert 23andme data to acf
Filtering	
noundef	No undefined sites for populations
bedfilter	Filter ACF/GLF file using sorted bedfile
segsite	Just retain segregating sites (or trans./transi)
sharing	Retain sites that share alleles between populations
nosharing	Retain sites that do NOT share alleles between populations
snosharing	Retain sites that STRICKLY do NOT share alleles between populations
Computations	
freqspec	Compute the frequency spectrum
closest	Return the distance between records
compute	Compute summary statistics
stats	Provide very basic stats
File transformations	
cat	Concatenate (GL—AC)f files
intersect	Intersection of (GL—AC)f files
union	Union of (GL—AC)f files
reheader	Replace header of (GL—AC)f files
Population transformations	
meld	Merge multiple populations as one for ACF files
popsub	Keep a subset of the populations
removepop	Remove a subset of the populations
replaceanc	Use ancestral/root information from another file
usepopsrootanc	Use 2 specified pops as ancestral/root information
rename	Rename populations
Data export	
glac2bed	Convert a (GL—AC)f file to BED
acf2bplink	Convert an ACF file to binary PLINK
acf2fasta	Convert an ACF file to fasta
acf2gphocs	Convert an ACF file to G-PhoCs
acf2nexus	Convert an ACF file to Nexus
acf2treemix	Convert an ACF file to treemix
acf2eigenstrat	Convert an ACF file to EIGENSTRAT
GLF/ACF conversion	
glf2acf	Convert glf to acf
Indexing	
index	Index acf/glf file
idxstats	Basic statistics using the index of a acf/glf file
Viewing	
view	View all or a region of a ACF/GLF file

2 File format

A glactools file can be 2 types of files:

- **GLF**: genotype likelihoods file for a single diploid individual individual
- **ACF**: allele count file for 1 or more indivuals

Each one of these files can either be:

- **binary and bgzipped**: Default output. Recommended as it saves space and can be indexed
- **binary**: only used for UNIX pipes into another program
- **text and (bg)zipped**:
- **text**: not recommend as it waste disk space

In addition to containing the fields for the various populations/individuals, the format has also 2 special fields to contain a root population and an ancestral population. The route population (labeled "root") corresponds to some outgroup to all other individuals (e.g. chimp for humans). The ancestral population (labeled "anc") corresponds to the most recent common ancestor of all the individuals in the ACF or GLF file and the root population (e.g. chimp/human ancestor for humans).

2.1 Text format

When stored as a text file (zipped or not), a glactools file a composed of a header and the data. Each line in the header starts with a # and has different slightly formats for a GLF/ACF files.

2.1.1 GLF

The header has the following format

```
#GLF
#PG:[command line used]
#GITVERSION: [github revision]
#DATE: YYYY-MM-DD
#[PROGRAM TAG]
#SQ    SN:[FIRST CHROMOSOME]   LN:[FIRST CHROMOSOME LENGTH]
#SQ    SN:[SECOND CHROMOSOME]  LN:[SECOND CHROMOSOME LENGTH]
...
#SQ    SN:[LAST  CHROMOSOME]   LN:[LAST  CHROMOSOME LENGTH]
#chr   coord  REF,ALT root  anc    pop1    pop2    ...
```

The lines starting with **#SQ** define the chromosomes of the reference and their lengths. The line starting with **#chr** is called the defline as it contains the name of the individuals. The following lines:

```
#SQ    SN:[LAST  CHROMOSOME]   LN:[LAST  CHROMOSOME LENGTH]
#chr   coord  REF,ALT root  anc    pop1    pop2    ...
```

use a [tab] to separate the fields.
Genotype likelihoods The remaining lines contain the actual data have the following format:

```
chr   coord  REF,ALT [root info]   [anc info]    [pop1 info]    [pop2 info
    ]   ...
```

Each field is [TAB] delimited. The ALT allele is set to N if there were no alternative allele found. The info for GLF has the following format:

```
[ref,ref GL],[ref,alt GL],[alt,alt GL]:[CPG flag 0=no,1=yes]
```

where GL=genotype likelihoods and is on a PHRED scale and between 0..255. Here is an example of valid lines:

```
1      1689574 A,N   0,255,255:0   0,255,255:0   0,45,120:0
1      1689575 T,C   255,0,255:0   0,255,255:0   0,25,90:0
```

2.1.2 ACF

The header has the following format:

```
#ACF
#PG:[command line used]
#GITVERSION: [github revision]
#DATE: YYYY-MM-DD
#[PROGRAM TAG]
#SQ    SN:[FIRST CHROMOSOME]   LN:[FIRST CHROMOSOME LENGTH]
#SQ    SN:[SECOND CHROMOSOME]  LN:[SECOND CHROMOSOME LENGTH]
...
#SQ    SN:[LAST  CHROMOSOME]   LN:[LAST  CHROMOSOME LENGTH]
#chr   coord  REF,ALT root  anc    pop1    pop2    ...
```

The lines starting with SQ define the chromosomes of the reference and their lengths. The line starting with chr is called the defline as it contains the name of the individuals. The following lines:

```
#SQ    SN:[LAST  CHROMOSOME]   LN:[LAST  CHROMOSOME LENGTH]
#chr   coord  REF,ALT root  anc    pop1    pop2    ...
```

use a [tab] to separate the fields.
Allele counts The remaining lines contain the actual data have the following format:

```
chr   coord  REF,ALT [root info]   [anc info]    [pop1 info]    [pop2 info
    ]   ...
```

The ALT allele is set to N if there were no alternative allele found. The info for GLF has the following format:

```
[REF allele count],[ALT allele count]:[CPG flag 0=no,1=yes]
```

The allele counts are raw counts and between 0..65535. Here is an example of valid lines:

```
1      1689574 A,N   1,0:0  1,0:0  2,0:0
1      1689575 T,C   0,1:0  1,0:0  2,0:0
```

2.2 Binary

The binary format has the following specifications. We recommend to store the format as a binary bgzip file as it can be indexed easily. However for UNIX piping we recommend to use the uncompressed version (option -u). An ACF/GLF file in binary format has the following format:

Field	**Description**	**Type**	**Value**
bammagicstr	BAM magic string. This field has no purporse but to make htslib happy	char [4]	BAM\1
magicstr	ACF/GLF magic string. This field indicates whether this file is in GLF or ACF format It is either ACF or GLF followed by the number of bytes per record (1 for GLF 2 for ACF) Currently, the code support up to 65535 alleles but could be extended for higher numbers	char [5]	ACF2\1 or GLF1\1
sizeheader	size of the header in bytes	uint32_t	
header	The header (see section 2.1.1 and 2.1.2) in bytes	char [sizeheader]	
sizePops	The number of populations **excluding** the root and anc	uint32_t	
Begin list of records			
chri	The index of the chromosome	uint16_t	
coordinate	The coordinate on the chromosome	uint32_t	
REF|ALT	The first 4 bits are for the REF, the last for the ALT N=0000, A=0001, C=0010, G=0011, T=0100	uint8_t	
Begin list of individuals/populations (see below)			

2.2.1 GLF

Each record contains the following for (sizePops+2) times:

Field	**Description**	**Type**	**Value**
rrGL	Genotype likelihood on a PHRED scale for the ref,ref genotype	uint8_t	
raGL	Genotype likelihood on a PHRED scale for the ref,alt genotype	uint8_t	
aaGL	Genotype likelihood on a PHRED scale for the alt,alt genotype	uint8_t	
CpG	Flag to know whether the actual individual is a CpG or not	uint8_t	

2.2.2 ACF

Each record contains the following for (sizePops+2) times:

Field	Description	Type	Value
refCount	Raw count of the number of reference alleles found	uint16_t	
altCount	Raw count of the number of alternative alleles found	uint16_t	
CpG	Flag to know whether the actual individual is a CpG or not	uint8_t	

3 General considerations

Each subprogram contained in glactools will always print compressed binary and printed the STDOUT. if you wish to redirect the output to a file, simply use the Unix redirect:

```
glactools cmd input.acf.gz > output.acf.gz
```

For several commands, the command line option –fai is mandatory as this provides the program with information about the reference genome, the name and order of the chromosomes. To visualize either an ACF or a GLF file, simply use glactools view:

```
glactools view input.acf.gz
```

you can combine several programs using Unix pipes where the output of a program becomes the input of another one:

```
glactools cmd1 -u input.acf.gz | glactools cmd2 -u --option example /dev/
    stdin | glactools cmd3 /dev/stdin > output.acf.gz
```

In general when piping to a program, we highly recommend to use the -u option which will disable the block zipped output. However the last program part of this pipeline should write compressed binary to save disk space (default output mode).

4 Data import

This program produces an ACF file from a BAM file:

4.1 From BAM

```
glactools bam2acf [options] <fasta file> <bam file> <name sample>
```

4.2 From VCF

This program convert VCF files into GLF (prints to the STDOUT):

```
glactools vcf2glf <vcf file> <name sample>
```

However, to call ACF directly from VCF, use this program:

```
glactools vcf2acf <vcf file> <name sample>
```

If it is a VCF file with multiple individuals (e.g. 1000 Genomes), use:

```
vcfm2acf [options] <vcf file>
```

4.3 From 23andme

This program convert 23andme files into ACF (prints to the stdout)

```
glactools 23andme2ACF <23andme file> <name sample>
```

4.4 From AXT alignment

This program will parse a multispecies AXT alignment (from USCS) and print a ACF file

```
glactools axt2acf <chr name> <name sample> <axt file>
```

5 Viewing and indexing

5.1 Viewing and subsampling ACF/GLF files

The basic command for viewing compressed ACF/GLF files is:

```
glactools view <ACF/GLF file>
```

It has options to view the defile (-h) and the entire header (-H).

It can also produce binary compressed or uncompressed from ACF/GLF files in raw text (which has to include the header) using:

```
cat myACFfile.txt | glactools view -b - > myACFfile.acf.gz
```

Works as well with GLF. A subset of records can be produces using -s.

5.2 Indexing for fast retrieval

If a GLF/ACF is index, you can retrieve rapidly a specific site or range or chromosome. If the ACF/GLF is in binary format and zipped using bgzip, the following command can be used to index it:

```
glactools index <ACF/GLF file
```

for example:

```
glactools index input.acf.gz
```

This will create a .bai file. The index format used is exactly the same as the one used by htslib so please refer to do their documentation. Once finished, you can retrieve data the following way:

```
glactools view <ACF/GLF file> chr:start-end
```

example:

```
glactools view myfile.acf.gz 21:20000-20010
```

or even just the chromosome:

```
glactools view myfile.acf.gz 21
```

6 File transformation

This section described operations that tranform one or more ACF/GLF files into one or more ACF/GLF files.

6.1 Tranform GLF to ACF

This program will transform GLF files to ACF using a cutoffs which can be changed on PHRED likelihoods.

```
glactools glf2acf <glf file>
```

6.2 Filtering

The filters can be one of the following:

mode	use
noundef	No undefined sites for populations
bedfilter	Filter glactools file using sorted bedfile
segsite	Just retain segregating sites (or trans./transi)
sharing	Retain sites that share alleles between populations
nosharing	Retain sites that do not share alleles between populations
znosharing	Retain sites that strickly do not share alleles between populations

Here is a description of the different filter modes:

6.2.1 No undefined sites

This will filter out any site where the allele count is null (0,0) for both reference and alternative

```
glactools noundef <ACF file>
```

6.2.2 Filtering using BED file

This will keep only the positions in the bed file

```
glactools noundef [options] <ACF or GLF file>
```

6.2.3 Filter for segregating sites

This will retain sites where the allele count is greater than 0 for either the reference or alternative for at least one individual. It has options to retain only transitions or transversions.

```
glactools segsite [options] <ACF file>
```

6.2.4 Allele sharing between 2 populations

This will only retain sites where every individuals in population group 1 share the same allele(s) as every individual in population group 2. It requires that the allele count for every individual for both groups be non-zero. A random allele is picked (biased for allele count) for heterozygous position so do not be surprised if you get different outputs every time.

```
glactools sharing <ACF file> <comma separated group 1> <comma separated
    group 2>
```

6.2.5 No allele sharing between 2 populations

This will filter sites where individuals in population group 1 do not share at least one allele with individual in population group 2. It requires that the allele count for every individual for both groups be non-zero. A random allele is picked (biased for allele count) for heterozygous position so do not be surprised if you get different outputs every time. In other words, the individuals in the first group have to be all reference and the second all alternative or vice-versa.

```
glactools nosharing <ACF file> <comma separated group 1> <comma separated
     group 2>
```

6.2.6 Strictly no allele sharing between 2 populations

This will filter sites where individuals in population group 1 strickly do not share any allele with individual in population group 2. It requires that the allele count for every individual for both groups be non-zero. Please remember that this will exclude any hetezygous sites

```
glactools snosharing <ACF file> <comma separated group 1> <comma
    separated group 2>
```

6.3 File operations

6.3.1 Concatenate

This program concatenates many files where the header is found in the first file and does not use the headers from the remaining ones. It prints to the /dev/stdout.

```
glactools cat [options] <glf file1> <glf file2> ..
```

6.3.2 Intersect 2 or more files on chromosome/coordinate

This program will print the intersection (when all sites are defined) of the ACF/GLF files to stdout, it will skip triallelic sites.

```
glactools intersect <ACF/GLF file 1> <ACF/GLF file 2> ...
```

6.3.3 Unite 2 or more files on chromosome/coordinate

This program will print the union (when any site is defined) of the ACF/GLF files to stdout, it will skip triallelic sites.

```
glactools union [options] <ACF/GLF file 1> <ACF/GLF file 2> ...
```

6.3.4 Substitute header

This program will used the header the user specifies and prints to the /dev/stdout. This is more for advanced users as you change the behavior of the program.

```
glactools reheader [options] <glac file> <text file>
```

6.4 Populations operation

6.4.1 Take a subset of populations

This will keep only the population specified in the list. Please note that it will set the alternative allele to 'N' if no population has the alternative allele

```
glactools  popsub <ACF/GLF file> <comma separated group to keep>
```

6.4.2 Remove populations

This will remove the population specified in the list. Please note that it will set the alternative allele to 'N' if no population has the alternative allele

```
glactools removepop <ACF/GLF file> <comma separated group to remove>
```

6.4.3 Merge two populations as one

This program will merge/meld different specified populations into a single one. You

```
glacmeld [options] <glac file> "popToMerge1_to_1,popToMerge2_to_1,...." "
    newid1" "popToMerge1_to_2,popToMerge2_to_2,...." "newid2"
```

Example of usage:

```
glactools meld data.acf.gz "Papuan,Austalian" "Oceanians" > dataOceanians
    .acf.gz
```

6.4.4 Rename populations

This program will rename different populations.

```
glactools rename [options] <ACF file> "oldpopname1,oldpopname2,..." "
    newpopname1,newpopname2,..."
```

Example of usage:

```
glactools rename data.acf.gz "Papuan,Austalian" "Oceanians1,Oceanians2"
```

6.4.5 Use population as root and ancestor

This program will use specified populations as root and ancestor and produce records with only those two populations. This program is especialyl useful with "replaceanc".

```
glactools usepopsrootanc [options] <glac file> <pop to use as root> <pop
    to use as anc>
```

6.4.6 Replace ancestor

This program will print the first ACF/GLF file but with the ancestral information from the second one to STDOUT. Can be combined with usepopsrootanc.

```
glactools replaceanc [options] <GLAC file1> <GLAC file2>
```

7 Computations

This section describes computation operations that can be done on ACF/GLF files and will produce some type of information on the console.

7.1 Basic file stats

This program will compute very basic stats on the file (# of seg sites, TS/TV)

```
glactools stats <ACF|GLF file>
```

7.2 Stats from the index

This program will provide stats using the index file as to how many records are present on each chromosome:

```
glactools idxstats <ACF|GLF file>
```

7.3 Population genetics/genomics

7.3.1 Site frequency spectrum

This program will print the number of observed alleles for the reference and alternative alleles.

```
glactools freqSpec [options] <ACF file>
```

7.4 Closest sites

This program will print to STDOUT the distance to the closest site for each record.

```
glactools closest [options] <ACF/GLF file>
```

7.5 Compute summary stats

This program will compute summary stats for population genetics:

```
glactools compute -p <program> <ACF file>
```

As of now ¡program¿ can be: “paircoacompute” and “dstat”. See further details in the subsections below.

These programs usually separate results into the following categories:

category	**meaning**
all	all sites
onlyCpg	restricted to CpG sites
noCpg	remove CpG sites from the calculations
transitions	restrict to transitions
transversions	restrict to transversions
noDamage	remove sites the ancestral-¿derived is either C-¿T or G-¿A

7.5.1 Pairwise coalescence

"glaccompute" offers the possibility to compute the average coalescence (see Prüfer et al. [2010]) for all pairs of individuals. For any bi-allelic segregating site for which the ancestral allele is available and where one or both individuals of a pair carry the derived variant (DD), this derived change can be traced back to a given lineage (see Supplemental Figure 1). If either one of the individuals carries the derived allele but the other still carries the ancestral (DA or AD), it is more likely to have arisen in the lineage specific to that individual. For the first individual, the average coalescence is computed as such:

$$\frac{DA}{DA + DD}$$

and for the second individual:

$$\frac{AD}{AD + DD}$$

Average coalescence for an individual measures the fraction of sites that coalesce after the split from the other individual. An average coalescence of 100% indicates a complete absence of shared alleles between two samples whereas a coalescence of 0% means that both individuals are identical at segregating sites. Confidence intervals for the average coalescence for a given window are obtained with a Wilson score interval (see Wilson [1927]). Let $\hat{p}$ be the computed average coalescence and n the denominator from either of the average coalescence expressions above and let z be error percentile score. The confidence on the measure is expressed by:

$$\frac{\hat{p} + \frac{z^2}{2n} \pm z \cdot \sqrt{\frac{\hat{p}*(1-\hat{p}) + \frac{z^2}{4n}}{n}}}{1 + \frac{z^2}{n}}$$

The results are reported for each window. The last "window" are all the data with the jacknifing. Each window contains the following category: all, onlyCpg, noCpg, transitions, transversions, noDamage.

Each category contains: absence of mutation (AA), mutation in the common branch (DD), mutation in branch for individual #1 (DA), mutation in branch for individual #2 (AD), $\frac{DA}{DA+DD}$, lower bound for $\frac{DA}{DA+DD}$, higher bound for $\frac{DA}{DA+DD}$, $\frac{AD}{AD+DD}$, lower bound for $\frac{AD}{AD+DD}$, higher bound for $\frac{AD}{AD+DD}$. If this is the last window with all the data with jacknifing, it also contains the mean from the jacknifing, the lower bound using the jacknifing and upper bound using the jacknifing.

To plot the output, 2 R scripts are available, one to make barplots:

```
paircoacompute2barplot.R [output pairwise coa] [samples to include, comma
    delim] [ancient samples to exclude, comma delim] [pdf out]
```

another to make a heatmap:

```
paircoacompute2heatmap.R [output pairwise coa] [samples to include, comma
    delim] [pdf out prefix] [pdf size]
```

You might have to tweek the fonts and margins. If you get a margin problem, use a large pdf size (e.g. 20).

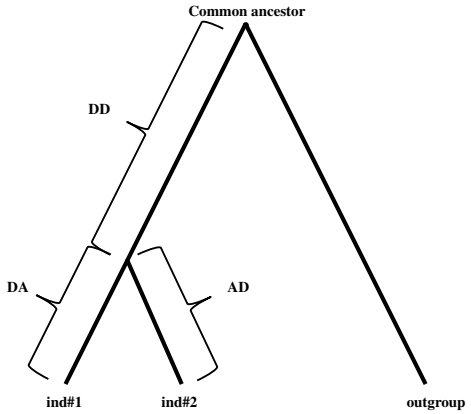

Figure 1: While conditioning on bi-allelic segregating sites, glactools can compute the fraction of such sites that coalesce in the common lineage. If both individuals carry the derived allele, it is likely to coalesce in the common branch. The average coalescence for a given individual attempts to measure the fraction of sites that coalesce in the lineage specific to that individual.

7.5.2 D-statistics

In the previous section, we focused on sites where an allele could be parsimoniously traced to a given lineage. However, given three individuals where a third falls outside of the variation of the first two, there are cases where derived mutations in this third individual cannot be parsimoniously traced to a specific lineage 2 (see Figure 2). More precisely, there are two possibilities: if the ind#1 carries the derived allele but the other does not (DADA) or vice-versa (ADDA). In practice, such cases occur due to the existence of both alleles in both populations but where one was not sampled. However, in the absence of admixture or substructure, both cases are expected to occur with equal frequency. If admixture occurred between the third individual (labeled "source" in Figure 2) and one of the first two (either ind#1 or ind#2), more derived alleles will be seen in that individual. The D-statistic (see Patterson et al. [2012]) attempts to measure imbalance between both cases by computing the following ratio:

$$D = \frac{ADDA - DADA}{ADDA + DADA}$$

If the result is close to 0, this can indicate a lack of greater proximity of one of the two individuals to the third one.

The results are reported for each window. The last "window" are all the data with jacknifing. Each window contains the following category: all, onlyCpg, noCpg, transitions, transversions, noDamage.

Each category contains: absence of mutation in both individuals (AA), individual #1 has the ancestral allele but #2 has the derived (AD), individual #1 has the derived allele but #2 has the ancestral (DA), absence of mutation in both individuals (DD), $\frac{ADDA-DADA}{ADDA+DADA}$. If this is the last window (all the data plus jacknifing), it will also report next to the D-stats, the mean from the

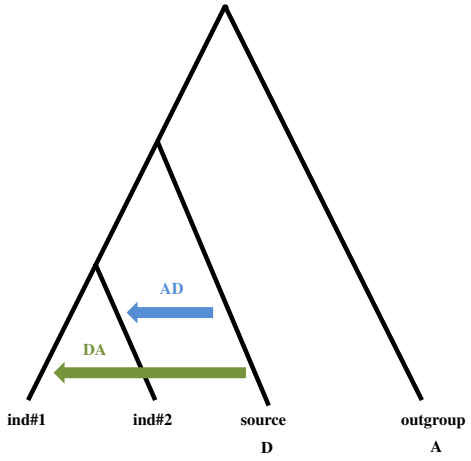

Figure 2: Given gene flow, a certain individual will carry a greater proportion of derived alleles from the potential source.

jackknifing, the lower bound from the jackknifing and the upper bound from the jackknifing.

To plot the output, 2 R scripts are available, one to make barplots:

```
dstats2pdf.R [dstat file] [output prefix pdf]
```

another to make a heatmap:

```
dstats2heatmap.R [dstat file] [sample source] [pdf out prefix] [pdf size]
```

You might have to tweek the fonts and margins. If you get a margin problem, use a large pdf size (e.g. 20).

8 Exporting data

8.1 To BED

To print an GLF/ACF as BED but where contiguous regions have been merged.

```
glactools glac2bed [options] <ACF/GLF file>
```

8.2 To binary PLINK

This program takes an ACF file and prints the genotype and SNP file in PLINK format

```
glactools acf2bplink [options] <ACF file> [out prefix]
```

8.3 To FASTA

This program takes an ACF file and prints a FASTA file using the allele information with one record per population. Each site generates one base pair.

```
glactools acf2fasta [options] <ACF file>
```

8.4 To G-PhoCS

This program takes an ACF file and prints G-Phocs output given a certain range specified in a BED file.

```
glactools acf2ghocs [options] <ACF file> <bedfile>
```

We recommend using glac2gphocsWrapper.pl is this automates the process and calls "glac2bed" for you.

8.5 To NEXUS

This program takes an ACF matrix and prints the alleles in NEXUS format.

```
glactools acf2nexus [options] <ACF file>
```

8.6 To EIGENSTRAT

This program takes an ACF file and exports the data in EIGENSTRAT.

```
glactools acf2eigenstrat [options] <ACF file> [out prefix]
```

8.7 To Treemix

To print Treemix input.

```
glactools acf2treemix [options] <ACF file>
```

References

Nick Patterson, Priya Moorjani, Yontao Luo, Swapan Mallick, Nadin Rohland, Yiping Zhan, Teri Genschoreck, Teresa Webster, and David Reich. Ancient admixture in human history. *Genetics*, 192(3):1065–1093, 2012.

Kay Prüfer, Udo Stenzel, Michael Hofreiter, Svante Pääbo, Janet Kelso, and Richard E Green. Computational challenges in the analysis of ancient DNA. *Genome Biology*, 11(5):R47, 2010.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.