imputeR Documentation

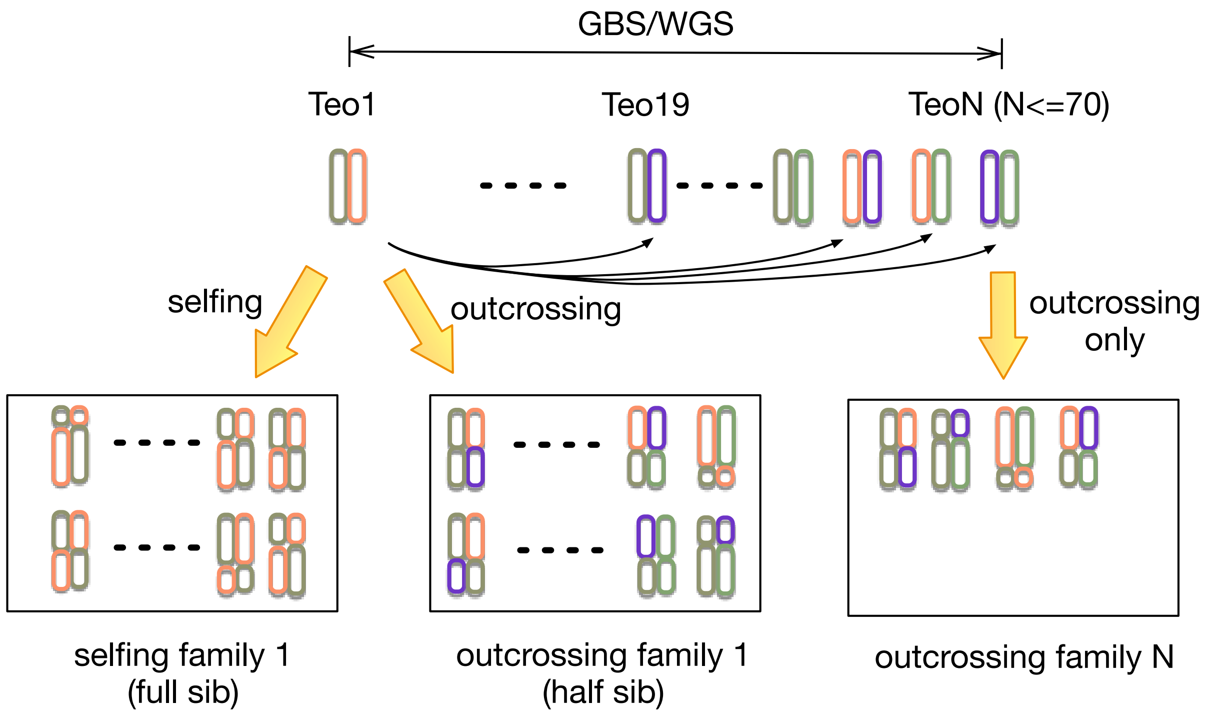
Jinliang Yang

October 8, 2015

Contents

1 Introduction

1.1 Crossing Scheme and Experimental Design



In this experiment, we selfed and outcrossed a set of ~70 teosinte landraces to get a progeny array composed of 4,875 individuals. The ~70 founders and all the progeny were genotyped using GBS. We also re-sequenced 20/70 founder lines (the others will be re-sequenced soon). Because of the high error rate of the GBS data, especially problematic for calling heterozygous sites, we employed a phasing and imputation strategy to infer the expected genotypes by combining parentage and GBS information.

This file is to document the `R` package we developed to solve the problems step by step.

1.2 Install and Usage

Install devtools first, and then use devtools to install imputeR from github.

```r
# install and load devtools
devtools::install_github("hadley/devtools")
library(devtools)
# install and load imputeR
install_github("yangjl/imputeR")
library(imputeR)
```

1.3 How to find help.

Within "R" console, type `?impute_mom` or `help(impute_mom)` to find help information about the function.

```
?impute_mom
```

```
## No documentation for 'impute_mom' in specified packages and libraries:
## you could try '??impute_mom'
```

2 Infer mom's genotype from GBS data

If we got mom's WGS data, this step can be skipped.

We have observed mom and observed (selfed) kids. We want to know $P(G|\theta)$, or the probability of mom's genotype given observed data θ. And according to Bayes' theorem,

$P(G|\theta) \propto P(\theta|G) \times P(G)$,

where $P(G)$ is the probability of the genotype according to the Hardy-Weinberg equilibrium estimated from the population. This consists of observed genotypes (G') of both mom and kids. So:

$P(G|\theta) \propto \left(\prod_{i=1}^{k} P(G'_k|G)\right) \times P(G'_{mom}|G) \times P(G)$,

where $P(G'_{mom}|G)$ is the probability of mom's observed genotype given a true genotype G by considering error rates, i.e. GBS homozygote error = 0.02 and heterozygote error = 0.8.
And $P(G'_k|G)$ is the probability of the k th kid's observed genotype given genotype G by considering error rates and Mendelian segregation rate. The function `impute_mom` was implemented to compute mom's genotype probabilities.

2.1 Toy example

In the below toy example, we simulated a full sib family with 12 kids for 3 loci. Mom GBS genotype is `0 0 0` and kids are all `0 0 0`. In the resulting table, the first three columns are the probabilities of genotype `0, 1, 2`. The 4th column is the odd ratio of the highest divided by the 2nd highest probability. `gmax` mom's genotype with the highest probability. `gor` mom's genotype with the highest probability and `OR` bigger than your threshould.

```
library(devtools)
library(imputeR)
# mom's GBS data is a vector
obs_mom <- c(0, 0, 0)
# kids GBS data is a list of vectors
obs_kids <- list(c(2, 0, 0), c(0, 0, 0), c(2, 0, 0), c(0, 0, 0), c(2, 0, 0), c(1, 0, 0),
c(0, 0, 0), c(2, 0, 0), c(0, 0, 0), c(1, 1, 0), c(0, 0, 0), c(0, 0, 0))

# run impute_mom to get the probabilities of mom's genotype of all loci
geno <- impute_mom(obs_mom, obs_kids, hom.error=0.02, het.error=0.8, p=NULL)
```

```
## ###>>> impute mom's genotype using [ 12 ] selfed kids ...
## ###>>> no allele frequencies provided. generating random allele frequencies from a neutral SFS
```

```
# find the most likely genotype of mom
momgeno(geno, oddratio=0.5, returnall=TRUE)
```

```
##            g0        g1        g2        OR gmax gor
## 1 -14.690171 -14.16787 -21.30116 0.5223057    1   1
## 2 -15.566039 -15.93703 -24.31126 0.3709867    0   3
## 3  -5.668641 -11.26843 -25.21913 5.5997923    0   0
```

2.2 Simulated Data

```
set.seed(123456)
sim <- SimSelfer(size.array=10, het.error=0.8, hom.error=0.002, numloci=100, rec=1.2, imiss=0.3)
```

2.3 Real Data

To load `hdf5` file, you need to install Vince's tasselr and ProgenyArray packages. If you fail to install them, please follow the above links and install the required dependencies.

```
# install devtools and then install the devlopmental version of tasselr and ProgenyArray using devtools
devtools::install_github("hadley/devtools")
library(devtools)
install_github("vsbuffalo/tasselr")
install_github("vsbuffalo/ProgenyArray")
install_github("yangjl/imputeR")
```

Load the required packages. Note you have to specify the locations of the packages if they were not in your searching path.

```
# load packages
library(parallel)
library(devtools)
options(mc.cores=NULL)
# you need to specify the location where the packages were installed.
load_all("~/bin/tasselr")
load_all("~/bin/ProgenyArray")
load_all("~/Documents/Github/imputeR")
```

The following several lines help you to reformat the `*.h5` HDF5 file into an R object. You need to specify the path of the HDF5 file.

```
# Note: at least 100G memory will be needed to load the hdf5 file
# load h5file
teo <- initTasselHDF5("largedata/teo.h5", version="5")
teo <- loadBiallelicGenotypes(teo, verbose = TRUE)
# reformat to imputeR object
ob <- imputeRob(teo)
save(file="largedata/teo.RData", list="ob")
```

Then, for each mom, run as above in the toy example using only the selfed offspring. Note that unlike the toy example, you will need to supply a vector of allele frequencies at each locus estimated from the parents. If you do not supply this or leave `p=NULL`, a random allele frequency drawn from the neutral SFS will be used instead. Since parents are coded as 0,1, or 2 for N parents the allele frequency p at a locus can be calculated as $\frac{\sum_{i=1}^{N} p_i}{2N}$.

3 Phasing Founder Genotypes

$P(H|\theta) \propto P(\theta|H) \times P(H)$
$P(H|\theta) \propto \left(\prod_{i=1}^{k} P(H'_k|H)\right) \times P(H)$
$P(H|\theta) \propto \left(\prod_{i=1}^{k}\prod_{l=1}^{n} P(G'_{i,l}|H)\right) \times P(H)$

- Where θ denotes observed data.
- $P(H)$ is the probability of the haplotype for a given window size of n.
- $P(G'_{i,l}|H)$ is the probability of kid i at locus l for a given haplotype H.
- The prior $P(H)$ is that all possible haplotypes of a given window size are equally likely.

---

4 Imputing and Phasing Kids

$P(H_k|\theta) \propto P(\theta|H_k) \times P(H_k)$
$P(H_k|\theta) \propto \left(\prod_{i=k} P(H'_k|H_k)\right) \times P(H_k)$
$P(H_k|\theta) \propto \left(\prod_{i=k}\prod_{l=1}^{n} P(G'_{i,l}|H_k)\right) \times P(H_k)$

- Where θ denotes observed data.
- $P(H)$ is the probability of the haplotype for a given window size of n.
- $P(G'_{i,l}|H)$ is the probability of kid i at locus l for a given haplotype H.
- The prior $P(H)$ is that all possible haplotypes of a given window size are equally likely.

```r
phase <- read.csv("../data/sim_phasing_res.csv")

hist(phase$er, breaks=30, main="Simulation (N=100)",col="#faebd7", xlab="Phasing Error Rate")
abline(v=mean(phase$er), col="red", lwd=2)
abline(v=median(phase$er), col="darkblue", lwd=2)
```

---

5 Phasing Dad of outcrossing progeny array

$P(H_d|\theta) \propto P(\theta|H_d) \times P(H_d)$
$P(H_d|\theta) \propto \left(\prod_{i=1}^{k} P(H'_k|H_d, H_m)\right) \times P(H_m) \times P(H_d)$
$P(H|\theta) \propto \left(\prod_{i=1}^{k}\prod_{l=1}^{n} P(G'_{i,l}|H)\right) \times P(H)$

- Where θ denotes observed data.
- $P(H_d)$ is the probability of the dad's haplotype for a given window size of n.
- $P(G'_{i,l}|H)$ is the probability of kid i at locus l for a given haplotype H.
- The prior $P(H)$ is that all possible haplotypes of a given window size are equally likely.
