IBD_Haplo R Tools

Marshall Brown, updated by E. A. Thompson, Fiona Grimson

March 5, 2013

1 Introduction

`IBDhaploRtools` consists of several functions to store, analyze, and plot the output of the `IBD_Haplo`. More information regarding `IBD_Haplo` can be found at www.stat.washington.edu/thompson/Genepi/pangaea.shtml, including a tutorial on how to use the software to generate outputs like the example data sets used in this tutorial.

2 Installing the Package

To begin, install and load the package into `R` if you have not already done so. This tutorial is also included in the package distribution as a vignette.

```
> ## to install the package
> ## install.packages("IBDhaploRtools", repos="http://R-Forge.R-project.org")
>
> ## to load the library at the start of R sesssion
> library("IBDhaploRtools")
>
> ## to bring up this tutorial
> ## vignette("IBDhaploRtools_tutorial")
```

Once the package is loaded, help files for specific functions can be accessed by typing `?[function name]` (for example `?ibdhap.make.states`.)

It is also possible to view a list of all functions and data sets in the package

```
## list everything in the package
ls("package:IBDhaploRtools", all=TRUE)
## list all the functions and their arguments
lsf.str("package:IBDhaploRtools")
## list datasets
data(package = "IBDhaploRtools")$results
```

This tutorial provides a description of the main functions in this package, and demonstrates how to implement these functions on IBDhaplo outputs.

3 Reading in Data

IBDhaplo output consists of a qibd file and an ids file. There is the option to supply each file as either a matrix which the user has already loaded into R, or as a string specifying the location of the file.

For this tutorial we will use examples of ids and qibd files for phased (haplotypic) and unphased (genotypic) data that are included in the package. First we need to load the example data sets:

```
> ## to use the tutorial data
> data(qibd_phased)
> data(ids_phased)
> data(ids_unphased)
> data(qibd_unphased)
>
> ## to use you own data, e.g.
> ## qibd.filename <- '~/Documents/qibd_unphased_2011.gold'
> ## ids.filename <- '~/Documents/ids_unphased_2011.gold'
```

The function `ibdhap.makes.states` will read the files and creates output that is used by all the other functions in the package. Thus, it is imperative to run this `ibdhap.make.states` first before any other function.

ibdhap.make.states: stores and simplifies the main output files (called "qibd_h.out" in the Gold examples) created by IBD Haplo. This is acheived by "calling" a marker to be in an ibd state if the probability of the state for the marker (conditional on the data and the model) meets the "cutoff" value. If the cuttoff is met, the value of the marker

is set to the ibd state (integer value from 1–15 or 1–9), otherwise value of the marker is set to 0 (which means this marker is a "no call".) `ibdhap.make.states` outputs a `R data.frame` where each row is a marker, and each column is a set of four haplotypes. The value at a marker for a set of haplotypes is as described above. The `R data.frame` that this function creates is expected by other functions in this package.

To run this function on the tutorial data sets, run:

```
> phased.gold <- ibdhap.make.states( qibd.file  = qibd_phased,
+                    ids.file= ids_phased, cutoff = 0.8)

[1] "qibd matrix accepted"
[1] "ids matrix accepted"

> unphased.gold <- ibdhap.make.states( qibd.file  = qibd_unphased,
+                    ids.file= ids_unphased, cutoff = 0.8)

[1] "qibd matrix accepted"
[1] "ids matrix accepted"

>
> ## or if you specified a file location
> ## rather than supplying pre-loaded data:
> ## phased.gold <- ibdhap.make.states( qibd.filename = qibd.filename,
> ##                                    ids.filename = ids.filename, cutoff =
```

Both data frames have four columns (one for each chromosome) and 2,000 rows (one for each marker). For the phased data set the first 20 markers for each haplotype are:

```
> phased.gold[1:20,]

   V1 V2 V3 V4
1   5  0 15 15
2   5  0 15 15
3   5  0 15 15
4   5  0 15 15
5   5  0 15 15
6   5  0 15 15
```

```
7   5  0 15 15
8   5 15 15 15
9   5 15 15 15
10  5 15 15 15
11  5 15 15 15
12  5 15 15 15
13  0 15 15 15
14 15 15 15 15
15 15 15 15 15
16 15 15 15 15
17 15 15 15 15
18 15 15 15 15
19 15 15 15 15
20 15 15 15 15
```

For marker eight, we see that `IBD_Haplo` inferred with greater than 0.8 probability that the first set of haplotypes were in IBD state 5 while the other three sets of haplotypes were in IBD state 15 (no IBD shared).

4 Data Analysis

In this section the functions that anaylse the output of `ibdhap.make.states` are demonstrated.

ibdhap.summary : summarizes the data created by `ibdhap.make.states` by calculating mean lengths of ibd segments, mean proportions of ibd shared, and counts on ibd segments. Averages are taken over all markers and all sets of haplotypes. Note that we need to specify whether we have haplotypic (h) or gentypic (g) data.

```
> ## summary statistics for phased data:
> summary.phased <- ibdhap.summary( phased.gold, data.type="h")
> summary.phased

$mean.prop
any.ibd not.ibd no.call
 0.2155  0.7035  0.0810
```

```
$mean.length
    len.ibd len.not.ibd  len.nocall
   61.57143   152.02703    10.60656

$seg.counts
    ibd  no.ibd no.call
     28      37      61
```

```
> ## and for unphased data:
> summary.unphased <- ibdhap.summary( unphased.gold, data.type="g")
> summary.unphased
```

```
$mean.prop
any.ibd not.ibd no.call
0.25175 0.61950 0.12875

$mean.length
    len.ibd len.not.ibd  len.nocall
   74.59259   176.92857    20.15686

$seg.counts
    ibd  no.ibd no.call
     27      28      51
```

This shows, for example, that for the phased data on average 70.35 % of the chromosome is in IBD state 15 (no ibd shared among the four haplotypes), and that the mean length (number of markers) of a continuous segment of the chromosome with no ibd shared is 152.02703 markers.

ibdhap.seg.lengths : given the ibd states from a set of haplotypes/pair of genotypes (taken from a column of the output of ibdhap.make.states), this function creates a `data.frame` consisting of all segments of differing ibd state, paired with their respective length. This function is called by many of the other functions, and provides equivalent information held in a column of "qibd.gold" but just in a different form.

```
> seg.lengths.phased <- ibdhap.seg.lengths(phased.gold[,4])
> seg.lengths.phased
```

```
   ibd.state seg.lengths
1         15          35
2          0          26
3         15         729
4          0           1
5         12         517
6          0          24
7         12         165
8          0          51
9         15         117
10         0           3
11        13          21
12         0           3
13        15          12
14         0           5
15        12         202
16         0           2
17        15          86
```

```
> seg.lengths.unphased <- ibdhap.seg.lengths(unphased.gold[,4])
> seg.lengths.unphased
```

```
   ibd.state seg.lengths
1          9         791
2          0           4
3          8         465
4          0          11
5          8         231
6          0          48
7          9         114
8          0          24
9          9          30
10         0           6
11         8         187
12         0          12
13         9          76
```

In the phased data, for the fourth set of haplotypes, this shows that there are 30 segments of differing ibd state as inferred by `IBD_Haplo`.

The seventh segment is in ibd state 12 and consists of 165 markers, while the ninth segment is in ibd state 15 and consists of 117 markers. If a vector of positions is given, segment lengths will be given in its respective units.

ibdhap.transitions: creates a matrix of transition counts from when ibd state switches along the chromosome. This function also needs to know whether the data are genotypic or haplotypic

```
> ##set the display so the matrix displays nicely
> options(width=100)
> ## transition matrix
> transitions.phased <- ibdhap.transitions(phased.gold, data.type="h")
> transitions.unphased <- ibdhap.transitions(unphased.gold, data.type="g")
> ## print the transitions matrix into two parts so it fits on the
> ## document
> transitions.phased[,1:8]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
 [1,]    0    0    0    0    0    0    0    0
 [2,]    0    0    0    0    0    0    0    0
 [3,]    0    0    0    0    0    0    0    0
 [4,]    0    0    0    0    0    0    0    0
 [5,]    0    0    0    0    0    0    0    0
 [6,]    0    0    0    0    0    0    0    0
 [7,]    0    0    0    0    0    0    0    0
 [8,]    0    0    0    0    0    0    0    0
 [9,]    0    0    0    0    0    0    0    0
[10,]    0    0    0    0    0    0    0    0
[11,]    0    0    0    0    0    0    0    0
[12,]    0    0    0    0    0    0    0    0
[13,]    0    0    0    0    0    0    0    0
[14,]    0    0    0    0    0    0    0    0
[15,]    0    0    1    0    2    0    0    4
```

```
> transitions.phased[,9:15]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]    0    0    0    0    0    0    0
```

```
 [2,]    0    0    0    0    0    0    0
 [3,]    0    0    0    0    0    0    1
 [4,]    0    0    0    0    0    0    0
 [5,]    0    0    0    0    0    0    3
 [6,]    0    0    0    0    0    0    0
 [7,]    0    0    0    0    0    0    0
 [8,]    0    0    0    0    1    0    3
 [9,]    0    0    0    0    0    0    0
[10,]    0    0    0    0    1    0    0
[11,]    0    0    0    0    0    0    6
[12,]    0    0    0    0    0    0    4
[13,]    0    0    1    0    0    0    3
[14,]    0    0    0    0    0    0    3
[15,]    0    1    5    4    2    3    0
```

```
> transitions.unphased
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    0    0    0    0    0    0    0    0    0
 [2,]    0    0    0    0    0    0    0    0    0
 [3,]    0    0    0    0    0    0    0    0    1
 [4,]    0    0    0    0    0    0    0    1    2
 [5,]    0    0    0    0    0    0    0    0    0
 [6,]    0    0    0    0    0    0    0    0    3
 [7,]    0    0    0    0    0    0    0    1    0
 [8,]    0    0    1    1    0    1    1    0   11
 [9,]    0    0    0    2    0    2    0   13    0
```

Hence, in the unphased transitions, we saw 13 instances of ibd state 9 transitioning to ibd state 8, for example.

ibdhap.barplot : Graphically displays regions of any ibd sharing, no ibd sharing and no calls along a chromosome for a set of haplotypes / pair of genotypes. The default colors are red, white, and grey for any state with ibd sharing, no ibd sharing and no calls, respectively.

For the phased data:

```
> par(mfrow=c(4,1))
> ibdhap.barplot(phased.gold[,1], data.type="h", xlab="", ylab="")
```

```
> ibdhap.barplot(phased.gold[,2], data.type="h", xlab="", ylab="")
> ibdhap.barplot(phased.gold[,3], data.type="h", xlab="", ylab="")
> ibdhap.barplot(phased.gold[,4], data.type="h", xlab="", ylab="")
```

For the unphased data:

```
> par(mfrow=c(4,1))
> ibdhap.barplot(unphased.gold[,1], data.type="g", xlab="", ylab="")
> ibdhap.barplot(unphased.gold[,2], data.type="g", xlab="", ylab="")
> ibdhap.barplot(unphased.gold[,3], data.type="g", xlab="", ylab="")
> ibdhap.barplot(unphased.gold[,4], data.type="g", xlab="", ylab="")
```

We can see that the first set of haplotypes does not have much (inferred) ibd shared among the four haplotypes, while the fourth set has a large portion of the chromosome in an ibd state where ibd is shared among the four haplotypes. Since we simulated these haplotypes, and therefore know the true ibd states for these four sets of haplotypes, it is interesting to compare the previous barplots to similar barplots created from using the known ibd states.

5 Comparison to True IBD

For the tutorial data we have the true IBD states for the phased data. We can transform this data to the true IBD states for unphased data.

h.to.g : This function is to transform the true ibd states in haplotypic (phased) format to gentoypic (unphased) format. Recall that the phased data has 15 possible IBD states, and the phased data has 9 possible ibd states. The function can be applied to any matrix with entries that are integers between 0 and 15.

```
> ## load the true IBD states data set
> data(trueibd_phased)
> ## map phased to unphased labels
> trueibd_unphased <- h.to.g( trueibd_phased )
> ## view the first few rows of each data set
> head(trueibd_phased)
```

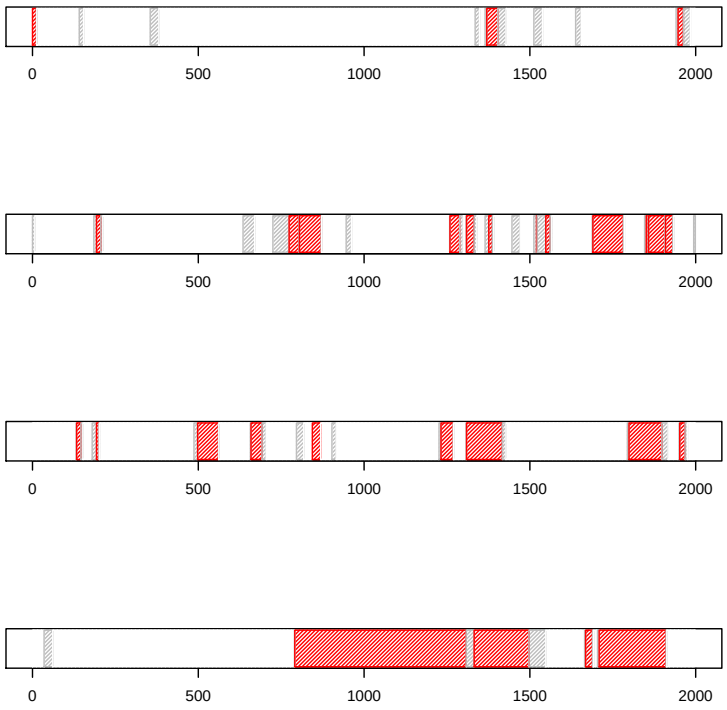

Figure 1: Barplots for four sets of four PHASED haplotypes using IBD_Haplo output. Red means some ibd shared, white is no ibd shared and grey means "no call."

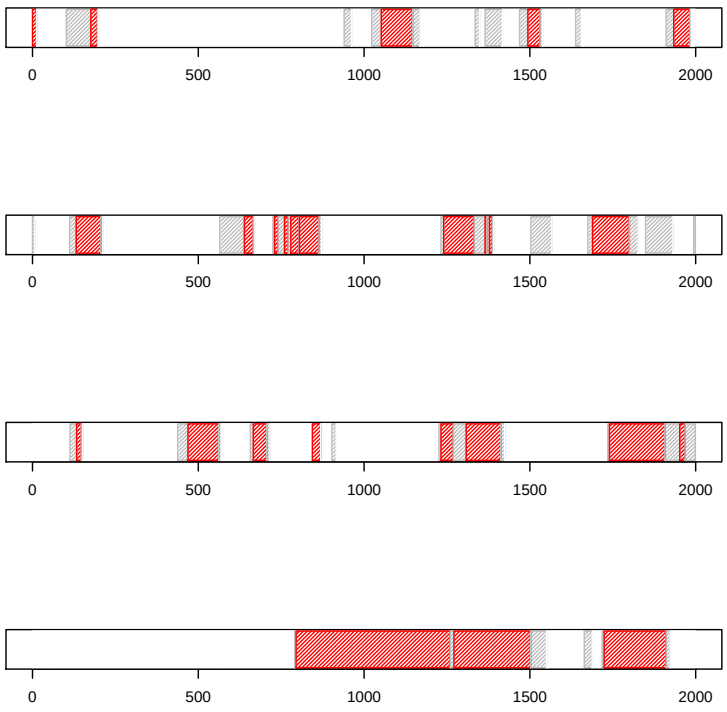

Figure 2: Barplots for four sets of four UNPHASED haplotypes using IBD_Haplo output. Red means some ibd shared, white is no ibd shared and grey means "no call."

```
   [,1] [,2] [,3] [,4]
V3   15   15   15   15
V4   15   15   15   15
V5   15   15   15   15
V6   15   15   15   15
V7   15   15   15   15
V8   15   15   15   15
```

```
> head(trueibd_unphased)
```

```
   [,1] [,2] [,3] [,4]
V3    9    9    9    9
V4    9    9    9    9
V5    9    9    9    9
V6    9    9    9    9
V7    9    9    9    9
V8    9    9    9    9
```

Now we can examine the true ibd states graphically using ibdhap.barplot. Phased data:

```
> par(mfrow=c(4,1))
> ibdhap.barplot(trueibd_phased[,1], data.type="h", xlab="", ylab="")
> ibdhap.barplot(trueibd_phased[,2], data.type="h", xlab="", ylab="")
> ibdhap.barplot(trueibd_phased[,3], data.type="h", xlab="", ylab="")
> ibdhap.barplot(trueibd_phased[,4], data.type="h", xlab="", ylab="")
```

Unphased data:

```
> par(mfrow=c(4,1))
> ibdhap.barplot(trueibd_unphased[,1], data.type="g", xlab="", ylab="")
> ibdhap.barplot(trueibd_unphased[,2], data.type="g", xlab="", ylab="")
> ibdhap.barplot(trueibd_unphased[,3], data.type="g", xlab="", ylab="")
> ibdhap.barplot(trueibd_unphased[,4], data.type="g", xlab="", ylab="")
```

Comparing these to the previous barplots, we see that `IBD_Haplo` does well in inferring ibd in this example.

ibdhap.compare : Compares inferred ibd state with simulated "true" states. Calculates the proportion of markers called in the correct state, false

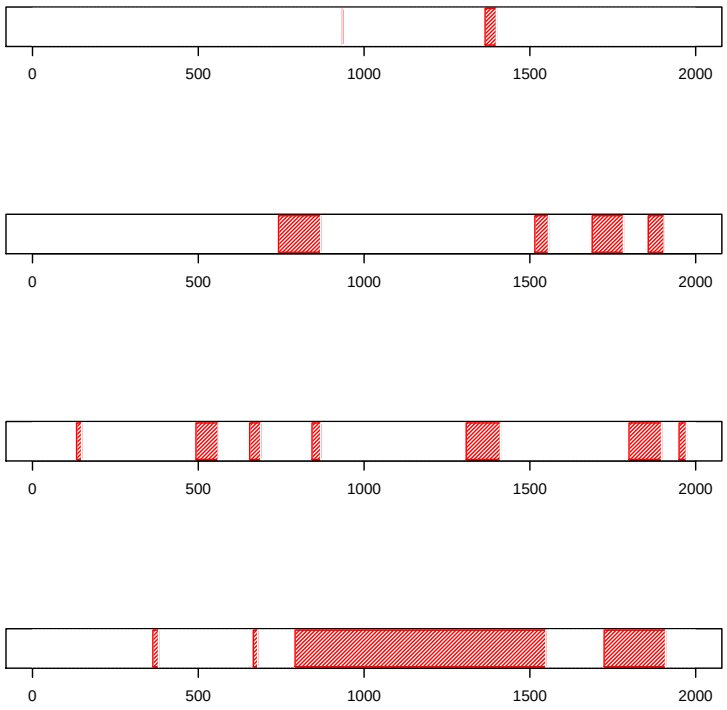

Figure 3: Barplots for four sets of four PHASED haplotypes using true IBD states of simulated data. Red means some ibd shared, and white is no ibd shared

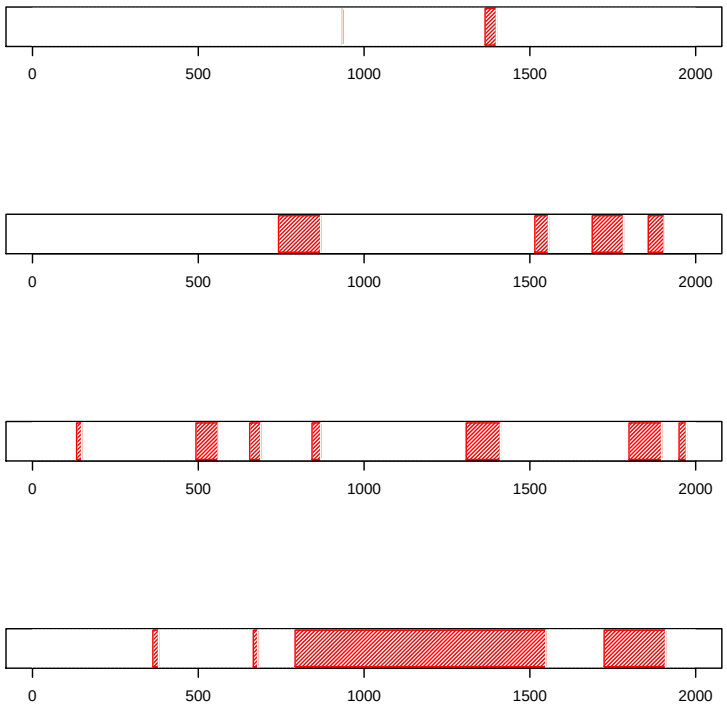

Figure 4: Barplots for four sets of four UNPHASED haplotypes using true IBD states of simulated data. Red means some ibd shared, and white is no ibd shared

positives (i.e. inferring ibd shared when none is shared), false negatives (i.e. inferring no ibd shared when ibd sharing is present) and the proportion of no calls.

```
> ## phased data
> ibdhap.compare(phased.gold, trueibd_phased, "h")

$prop.correct
[1] 0.884

$false.positives
[1] 0.030125

$false.negatives
[1] 0.00425

$no.calls
[1] 0.081

> ## unphased data
> ibdhap.compare(unphased.gold, trueibd_unphased, "g")

$prop.correct
[1] 0.79

$false.positives
[1] 0.075

$false.negatives
[1] 0.00425

$no.calls
[1] 0.12875
```

ibdhap.correct.bylen : if simulated data (true ibd state) is available, this function creates a dataframe summarizing how well `IBD_haplo` infers segments based on segment length. Each row in the dataframe represents a segment of true (simulated) ibd (shared or not shared). If

`ibd.only` is true, then only segments where ibd is shared are returned. The first column of resulting dataframe gives the length of the segment (as specified by the position vector), the second column gives the proportion of markers inferred correctly by ibdhaplo, and the third column gives the ibd state of the segment.

```
> corr.bylen.phased <-ibdhap.correct.bylen( phased.gold,
+                       trueibd_phased, data.type="h",
+                       ibd.only=FALSE, position=NA)
> corr.bylen.unphased <- ibdhap.correct.bylen( unphased.gold,
+                          trueibd_unphased, data.type="g",
+                          ibd.only=FALSE, position=NA)
```

We can use the output of `ibdhap.correct.bylen` to create Figures 5 and 6. The x axis is the length of the segment, and the y axis is proportion of SNPs called correctly within the true segment.

```
> plot( corr.bylen.phased[,1], corr.bylen.phased[,2],
+        xlab="segment length", ylab="proportion correct",
+        main = "Phased Data")
>  lines(lowess(corr.bylen.phased[,1],
+        corr.bylen.phased[,2]), col="blue")

> plot( corr.bylen.unphased[,1], corr.bylen.unphased[,2],
+        xlab="segment length", ylab="proportion correct",
+        main = "Unphased Data")
> lines(lowess(corr.bylen.unphased[,1],
+       corr.bylen.unphased[,2]), col="blue")
```

We can see, for example, in figure 5 that `IBD\_Haplo` does very well with ibd segments that are above 25 to 50 snps in length. In these data, this corresponds with ibd segments that are roughly one centimorgan in length.

This completes our quick run through of the main functions of `IBDHaploRtools`. Please see the man pages for more information regarding function arguments, descriptions, and examples.

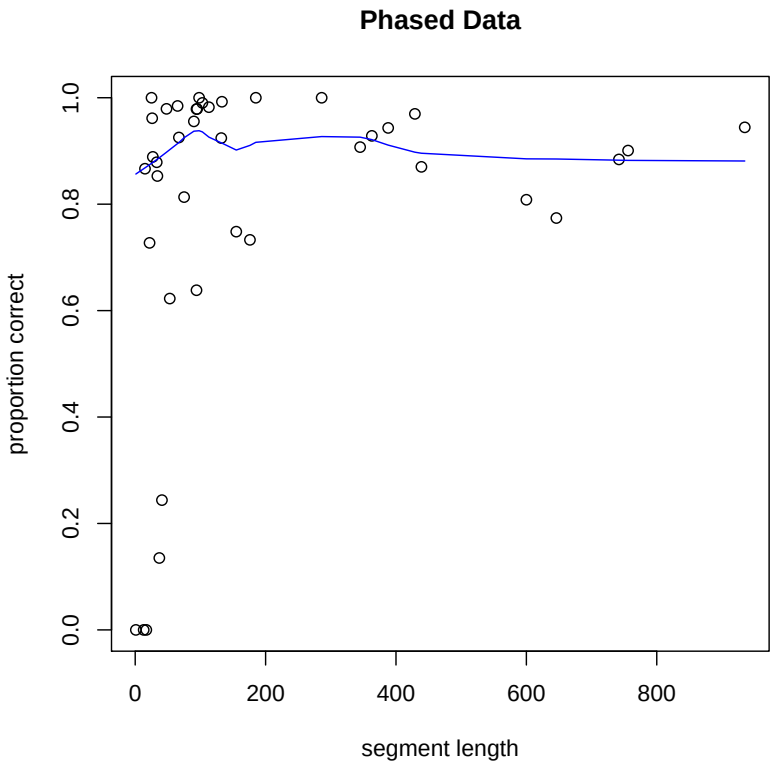

Figure 5: Each point in the plot is a segment of true (simulated) ibd (shared or not shared).The x axis is the length of the segment, and the y axis is proportion snps called correctly within the true segment

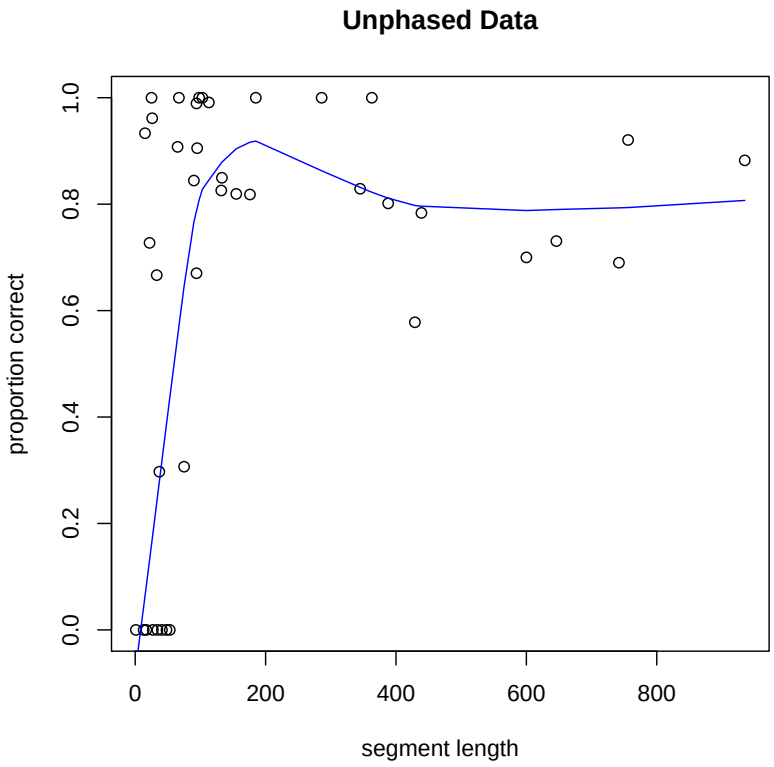

Figure 6: Each point in the plot is a segment of true (simulated) ibd (shared or not shared).The x axis is the length of the segment, and the y axis is proportion snps called correctly within the true segment