Statistical phasing with SelectionTools

Matthias Frisch

1 Overview **2**

2 Reference-based phasing of an F2 population **2**
2.1 Graphical genotypes 3
2.2 A phased heterozygous reference individual 5
2.3 Phasing options 6

3 Reference-free cohort phasing **7**

4 Imputation **8**

1 Overview

Statistical phasing determines which alleles at heterozygous markers belong to the same chromosome copy. `st.phase()` can either use a reference panel with known haplotypes or infer phase from the cohort itself. The examples below use simulated populations so that the inferred phase can be compared with the known phase by `st.switch.error()`.

Simulation uses `rng.init(seed=901)` after setting the thread count; this seeds the internal simulation generators as well as R. Later `set.seed()` calls control phase destruction, while `seed=42` in phasing options controls phasing. Reproducing the simulations requires the same thread count.

The first example is an F2 population derived from two inbred parents. It shows reference-based phasing, graphical genotypes, and the use of one phased heterozygous F1 individual as reference. The second example phases a random-mated cohort without a reference population.

```
> library(SelectionTools)
> st.set.num.threads(8)
> rng.init(seed=901)
```

2 Reference-based phasing of an F2 population

The first two tropical-maize lines are used as parents. We first isolate each parent. A second restriction keeps markers with no missing genotype and only one observed allele, that is, markers at which the parent is observed and homozygous.

```
> data("v-tropmaize-vcf")
> st.load.vcf.data(v.tropmaize.vcf,data.set="tropical")
> st.copy.marker.data( "P1","tropical" )
> st.restrict.marker.data( ind.list="1", data.set="P1")
> st.restrict.marker.data( NoAll.MAX=1, MaMis.MAX=0, data.set="P1")
> st.copy.marker.data("P2","tropical")
> st.restrict.marker.data(ind.list="2",data.set="P2")
> st.restrict.marker.data( NoAll.MAX=1, MaMis.MAX=0, data.set="P2")
```

The markers retained in both single-parent data sets are observed and homozygous in both parents. These markers are copied from the original data set and the two parental alleles are recoded as 1 and 2. Recoding before the simulation makes the later graphical genotypes directly interpretable as parental origin.

```
> retained.markers <- intersect(
+     st.get.map(data.set="P1")$Name,
+     st.get.map(data.set="P2")$Name
+ )
> st.copy.marker.data("parents","tropical")
> st.restrict.marker.data(
+     ind.list=c("1","2"),
+     mar.list=retained.markers,
+     data.set="parents"
+ )
> st.recode.ref.2(r1=1, r2=2, data.set="parents")
```

The two parental lines are transferred to the simulation backend. Their cross produces an F1, and selfing the F1 produces 200 F2 individuals. The F1 is kept as a phased heterozygous reference for a later example; the F2 is kept with its true simulated phase.

```
> st.set.simpop( pop.name="Parents", data.set="parents" )
> population.divide( "P1", "Parents", 1 )
> population.rename( "Parents", "P2" )
> cross( "F1", "P1", "P2", NoPg=1 )
> st.get.simpop( "F1", data.set="f1.reference")
> cross( "F2", "F1", "F1", NoPg=200 )
> st.get.simpop( "F2", data.set="f2.true" )
```

For phasing, a copy of the F2 is made and its stored phase is randomized. The unordered genotypes are unchanged. Additional copies are retained for the plots and the heterozygous-reference example.

```
> st.copy.marker.data( "f2", "f2.true")
> set.seed(314159)
> st.destroy.phase( data.set="f2")
> st.copy.marker.data( "f2.unphased", "f2" )
> st.copy.marker.data( "f2.hetero",   "f2" )
```

The two homozygous parents supply fixed reference haplotypes. The `st.phase()` phases the F2 with its general copying model.

```
> ph <- list(seed=42)
> st.phase(data.set="f2", ref="parents", phasing.options=ph)
```

Because the true simulated phase is available, the phasing result can be evaluated directly with `st.switch.error()`. Its print method shows the switch-error rate, scoring coverage, and genotype mismatches. Use `print(se, complete=TRUE)` to display all statistics.

```
> se <- st.switch.error(true.data.set="f2.true", data.set="f2")
> print(se)
```

```
                   result
switch.error       0.0395
coverage           1.0000
genotype.mismatches 0.0000
```

2.1 Graphical genotypes

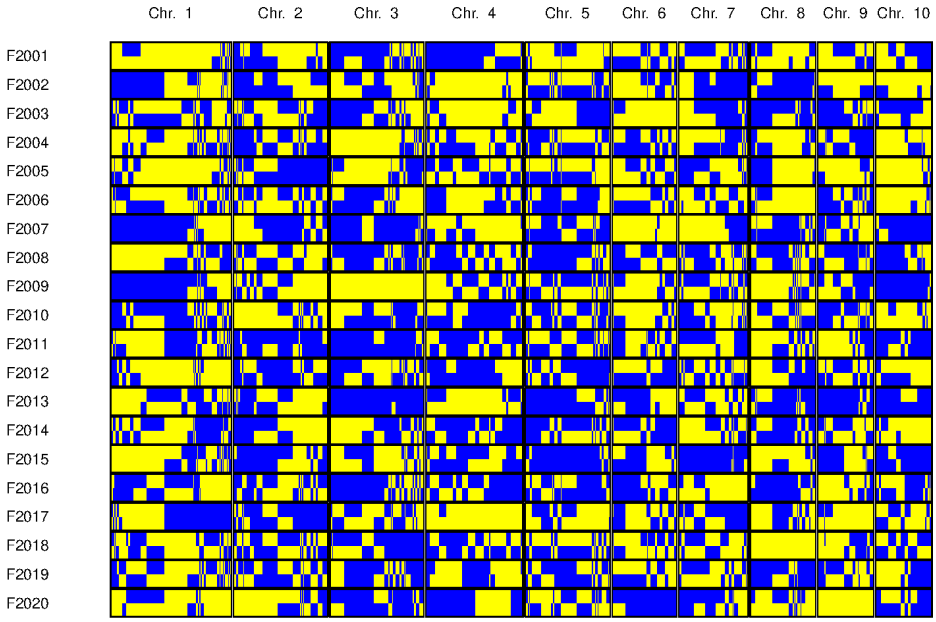
For the graphical genotypes we retain only markers that distinguish the two parents. After the parental recoding these markers have expected heterozygosity 0.5 in the two-parent data set.

```
> st.copy.marker.data("informative","parents")
> st.restrict.marker.data(ExHet.MIN=0.5, data.set="informative")
> informative.markers <- st.get.map(data.set="informative")$Name
> st.copy.marker.data("f2.plot.before","f2.unphased")
> st.restrict.marker.data(
+     mar.list=informative.markers,
+     data.set="f2.plot.before"
+ )
> st.copy.marker.data("f2.plot.after","f2")
> st.restrict.marker.data(
+     mar.list=informative.markers,
+     data.set="f2.plot.after"
+ )
> st.copy.marker.data("f2.plot.true","f2.true")
```

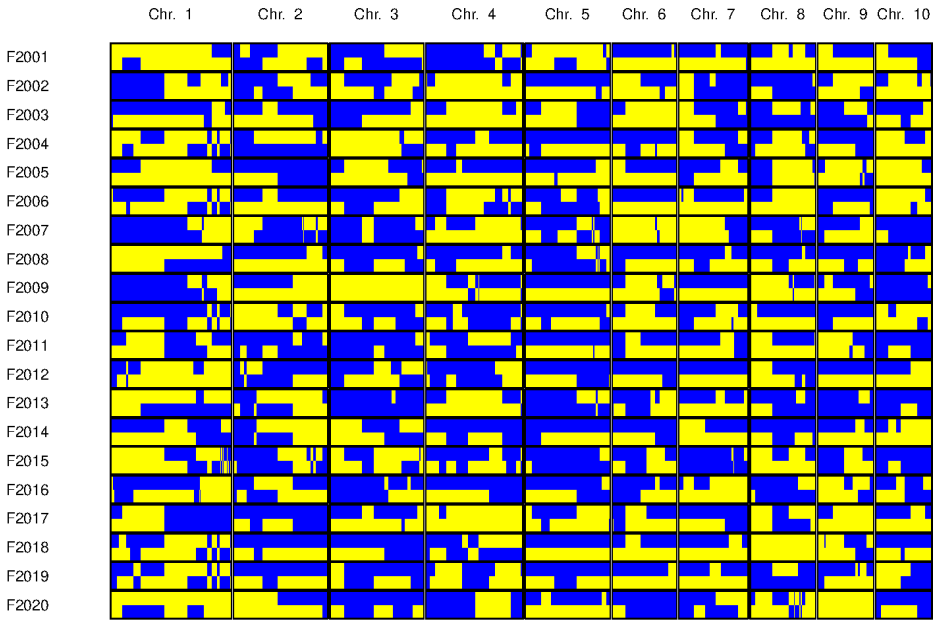
```
> st.restrict.marker.data(
+     mar.list=informative.markers,
+     data.set="f2.plot.true"
+ )
```

Because parental origin is already coded as 1 and 2, the three data sets can be plotted directly. The first plot shows the randomized input, the second the phased result, and the third the known simulated phase.

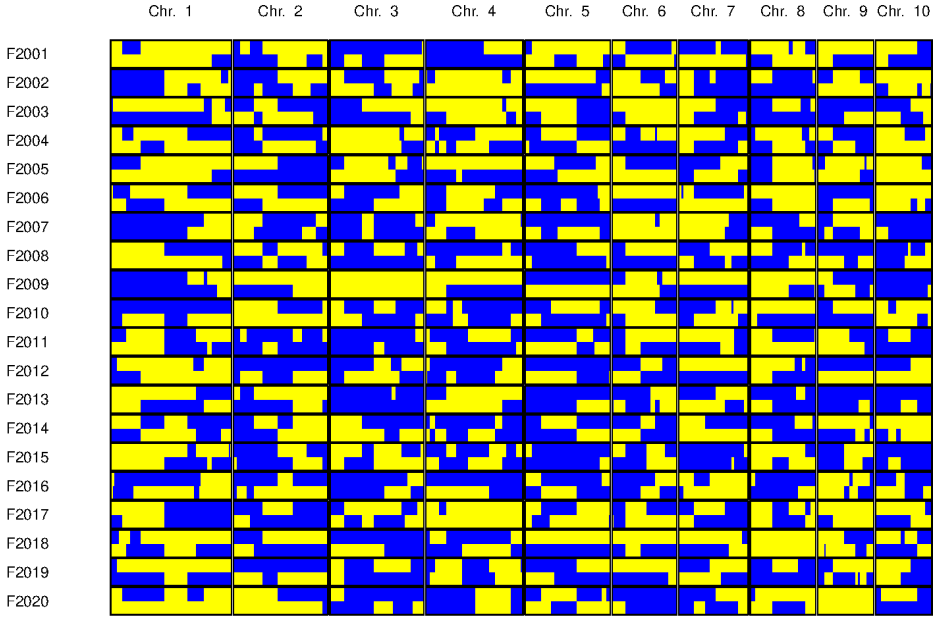
```
> st.plot.ggt(
+     data.set="f2.plot.before",
+     f.ind=1,
+     l.ind=20,
+     color=c("yellow","blue"),
+     z.min=1,
+     z.max=2
+ )
> st.plot.ggt(
+     data.set="f2.plot.after",
+     f.ind=1,
+     l.ind=20,
+     color=c("yellow","blue"),
+     z.min=1,
+     z.max=2
+ )
> st.plot.ggt(
+     data.set="f2.plot.true",
+     f.ind=1,
+     l.ind=20,
+     color=c("yellow","blue"),
+     z.min=1,
+     z.max=2
+ )
```



The randomized input contains frequent artificial changes between the two stored homologues. After phasing, long parental segments reappear.



They can be compared with the known simulated phase.



2.2 A phased heterozygous reference individual

Reference mode is not restricted to homozygous lines. The simulated F1 has one phased chromosome copy from each parent and can therefore be used directly as a single heterozygous reference individual. The copying model keeps its two reference haplotypes fixed; no population-type argument is needed.

```
> ph <- list(seed=42)
> st.phase(data.set="f2.hetero", ref="f1.reference",
 ↪  phasing.options=ph)
```

As in the two-parent reference example, the known simulated phase is used only to evaluate the result.

```
> se <- st.switch.error(true.data.set="f2.true", data.set="f2.hetero")
> print(se)
```

```
                    result
switch.error        0.0388
coverage            1.0000
genotype.mismatches 0.0000
```

2.3 Phasing options

Options are supplied in `phasing.options`; reference panels are supplied with the `ref` argument. Threads are set with `st.set.num.threads()`. Parallel execution requires an OpenMP-enabled build; a build without OpenMP executes serially.

Window selection is automatic by default. It chooses a whole chromosome when fewer than 800 common markers are available. This improves the amount of marker context without selecting settings against the known simulated phase. See `help("st.phase")` for every option, valid range and default. Observed unordered genotypes are retained, and the missing-call default is `impute=FALSE`. The function writes to the named data set only after a complete result has been validated. Use `commit=FALSE` to inspect diagnostics without changing the data set:

```
> ph <- list(seed=42)
> fit <- st.phase(
+     data.set="f2.unphased",
+     ref="parents",
+     phasing.options=ph,
+     commit=FALSE
+ )
> fit[c("windows", "maximum.worker.team", "chromosome.windows",
+       "filled.genotypes", "reference", "committed")]
```

```
$windows
[1] 10

$maximum.worker.team
[1] 8

$chromosome.windows
 1  2  3  4  5  6  7  8  9 10
 1  1  1  1  1  1  1  1  1  1

$filled.genotypes
[1] 0

$reference
$reference$samples
[1] 2

$reference$haplotypes
[1] 4

$reference$unchanged
[1] TRUE

$reference$ignored.markers
[1] 0
```

```
$committed
[1] FALSE
```

The dry run discards its proposed genotypes and returns diagnostics only. With `commit=TRUE`, the validated result is written to the data set and the function returns invisible `NULL`. The diagnostics report the number of windows, the largest worker team used, and the number of filled genotypes. The reference panel is unchanged, and `committed=FALSE` confirms that the target data set was not modified.

Reference calls must be complete at all target markers and heterozygous reference calls must be declared phased. Target and reference markers are matched by ID and their genetic maps must agree. The existing STvcf exporter normalizes allele labels separately within each data set: panels imported independently must already use compatible allele coding. The examples here use shared founder coding; no strand or allele-orientation inference is made.

3 Reference-free cohort phasing

When no reference data set is supplied, `st.phase()` phases the cohort itself. Here one random-mating generation is produced from the tropical-maize lines. We call this synthetic population `popSYN1` and retain its true simulated phase.

```
> st.set.simpop(pop.name="Base", data.set="tropical")
> m.s <- st.marker.data.statistics(
+     data.set="tropical",
+     mar=FALSE,
+     gen=FALSE
+ )$individual.list
> n.base <- nrow(m.s)
> cross("popSYN1","Base","Base",NoPg=n.base)
> st.get.simpop("popSYN1",data.set="popSYN1.true")
```

A copy is made and its stored phase is randomized in the same way as in the F2 example.

```
> st.copy.marker.data( "popSYN1", "popSYN1.true")
> set.seed(271828)
> st.destroy.phase( data.set="popSYN1")
```

Without a reference data set, `st.phase()` phases the cohort itself. The copying model estimates mismatch and recombination parameters from targets when its default `em=TRUE` setting is used. Imputation is disabled here so that the example focuses on phasing.

```
> ph <- list(seed=42)
> st.phase(data.set="popSYN1", phasing.options=ph, impute=FALSE)
```

The known simulated phase is used only for evaluating the result.

```
> se <- st.switch.error(
+     true.data.set="popSYN1.true",
+     data.set="popSYN1"
+ )
> print(se)
```

```
                   result
switch.error       0.0645
coverage           1.0000
genotype.mismatches 0.0000
```

4 Imputation

For a homozygous population, there is no phase ambiguity at an observed marker because the two chromosome copies carry the same allele. In this case, `st.phase()` can be used primarily to impute missing marker data. Enable filling of missing genotypes at typed markers explicitly with `impute=TRUE`. The inferred hard calls are retained without a posterior-confidence cutoff; this does not add markers absent from the target.

For example, a copy of the tropical-maize marker data can be imputed without changing the original data set. The marker statistics count missing alleles in column `AM`. Dividing their total by two gives the number of missing diploid genotypes before and after imputation.

```
> m.before <- st.marker.data.statistics(
+     data.set="tropical",
+     ind=FALSE,
+     gen=FALSE
+ )$marker.list
> st.copy.marker.data("tropical.imputed","tropical")
> ph <- list(seed=42)
> st.phase(
+     data.set="tropical.imputed",
+     phasing.options=ph,
+     impute=TRUE
+ )
> m.after <- st.marker.data.statistics(
+     data.set="tropical.imputed",
+     ind=FALSE,
+     gen=FALSE
+ )$marker.list
> imputation <- data.frame(
+     stage=c("Before imputation","After imputation"),
+     missing.genotypes=c(sum(m.before$AM),sum(m.after$AM))/2
+ )
> imputation
```

```
             stage missing.genotypes
1 Before imputation             26927
2  After imputation                 0
```

Observed genotypes are retained. With `impute=FALSE`, missing calls are restored after internal haplotype inference. With `impute=TRUE`, the inferred calls are kept. No posterior-probability output is provided.