Statistical phasing with SelectionTools

Matthias Frisch

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()`.

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(4)
```

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" )
```

With the two homozygous parents as reference, `st.phase()` recognizes the F2 design automatically and phases the F2 population.

```
> st.phase( reference.data.set="parents",
+           data.set="f2")
```

Because the true simulated phase is available, the phasing result can be evaluated directly with `st.switch.error()`. The returned data frame can be printed directly.

```
> st.switch.error( true.data.set="f2.true",
+                  data.set="f2")
```

```
                         result
errors                2.0000e+03
links                 3.0140e+04
switch.error          6.6357e-02
switch.error.percent  6.6357e+00
heterozygotes         3.2088e+04
scored.heterozygotes  3.2088e+04
coverage              1.0000e+00
missing.heterozygotes 0.0000e+00
genotype.mismatches   0.0000e+00
```

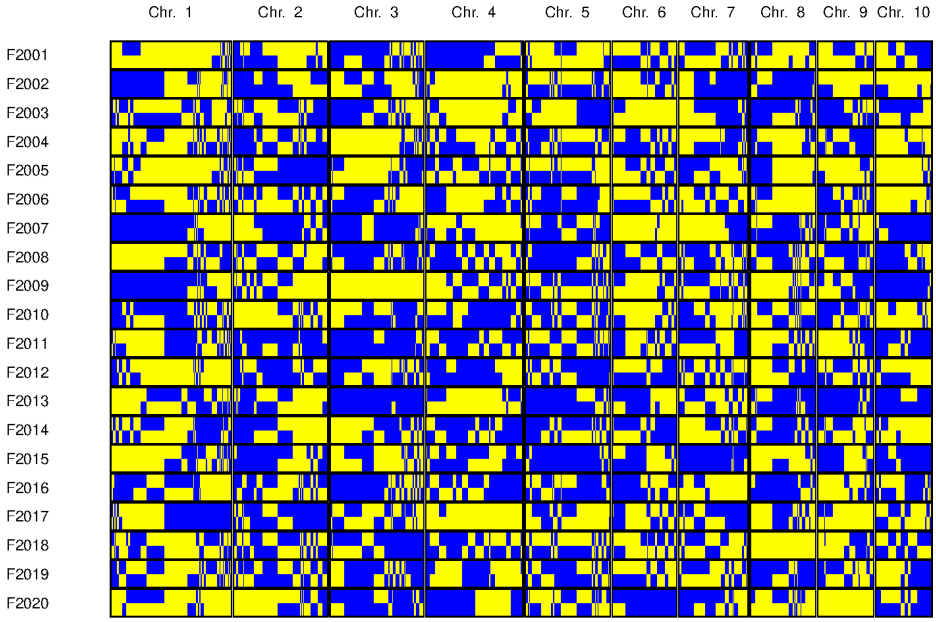
2.1 Graphical genotypes

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.

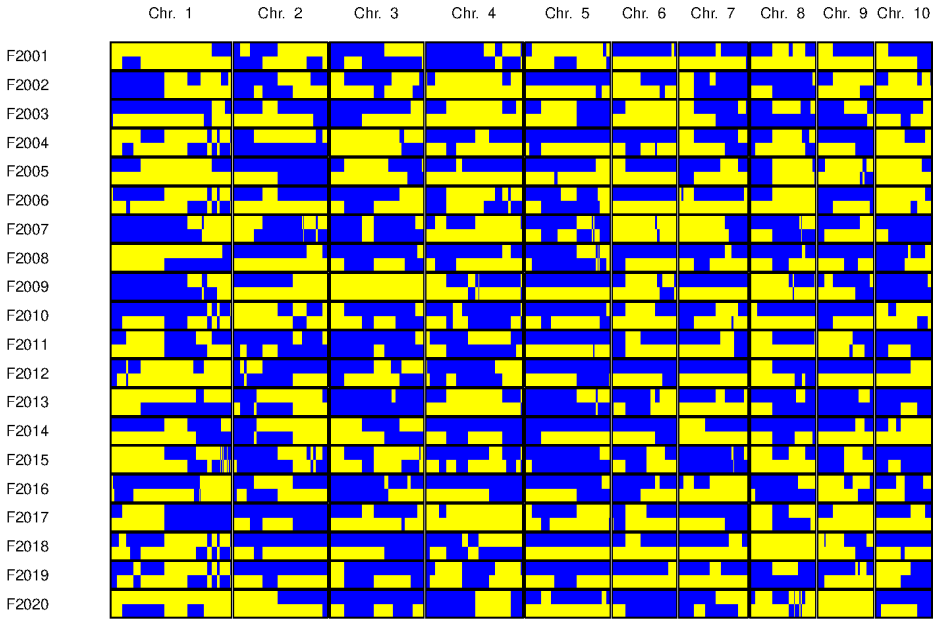
```
> st.copy.marker.data( "informative", "parents")
> st.restrict.marker.data( ExHet.MIN=0.5,
+                          data.set="informative")
> informative.markers <- st.get.map(data.set="informative")$Name
> st.copy.marker.data( "f2.plot.before", "f2.unphased")
> st.restrict.marker.data( mar.list=informative.markers,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

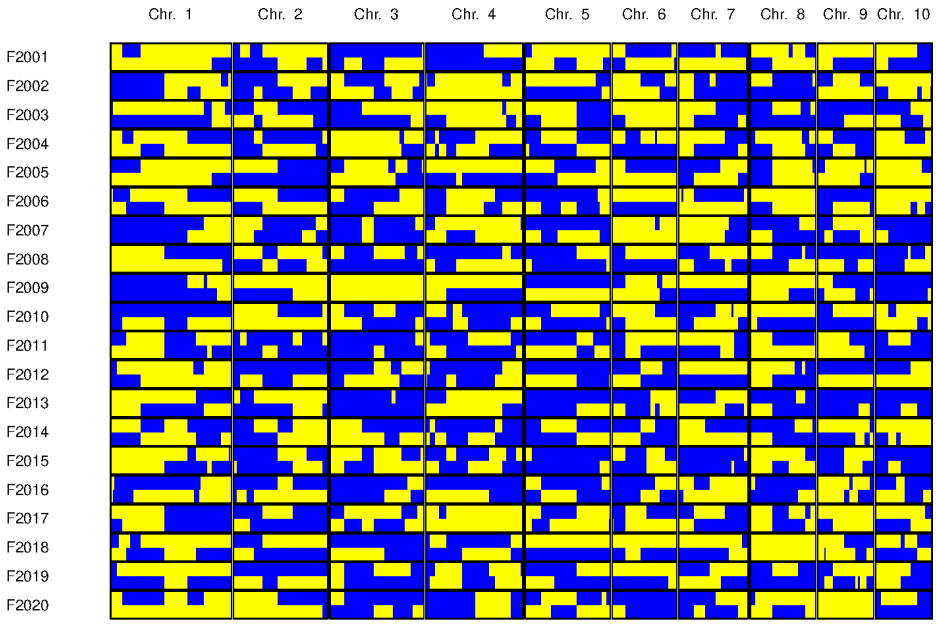
```
> st.plot.ggt(data.set="f2.plot.before",f.ind=1,l.ind=20,
+             color=c("yellow","blue"),z.min=1,z.max=2)
> st.plot.ggt(data.set="f2.plot.after",f.ind=1,l.ind=20,
+             color=c("yellow","blue"),z.min=1,z.max=2)
> st.plot.ggt(data.set="f2.plot.true",f.ind=1,l.ind=20,
+             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. Here the F2 population type is specified explicitly.

```
> st.phase( reference.data.set="f1.reference",
+           population.type="F2",
+           data.set="f2.hetero")
```

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

```
> st.switch.error( true.data.set="f2.true",
+                  data.set="f2.hetero")
```

```
                          result
errors                2.0000e+03
links                 3.0140e+04
switch.error          6.6357e-02
switch.error.percent  6.6357e+00
heterozygotes         3.2088e+04
scored.heterozygotes  3.2088e+04
coverage              1.0000e+00
missing.heterozygotes 0.0000e+00
genotype.mismatches   0.0000e+00
```

2.3 Population types and expected breaks per Morgan

For known breeding designs, `population.type` selects a fixed expected ancestry-junction density. The current defaults are:

Population type	F2	F3	F4	S2	S3	S4
Expected breaks/Morgan	1.000	1.500	2.000	1.500	1.750	1.875

The expert override `expected.breaks.per.morgan` can be supplied when a more appropriate value is known.

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. With no population type supplied, the general `Finfty` model is used and its copying-break intensity is estimated from the cohort. Imputation is disabled here so that the example focuses on phasing.

```
> st.phase( impute   = FALSE,
+           data.set = "popSYN1")
```

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

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

```
                          result
errors                6.370000e+03
links                 7.349900e+04
switch.error          8.666785e-02
switch.error.percent  8.666785e+00
heterozygotes         7.613800e+04
scored.heterozygotes  7.613800e+04
coverage              1.000000e+00
missing.heterozygotes 0.000000e+00
genotype.mismatches   0.000000e+00
```

4 References

Li N, Stephens M (2003) Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. Genetics 165:2213–2233. doi:10.1093/genetics/165.4.2213.

Durbin R (2014) Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT). Bioinformatics 30:1266–1272. doi:10.1093/bioinformatics/btu014.

Browning SR, Browning BL (2007) Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. American Journal of Human Genetics 81:1084–1097. doi:10.1086/521987.

Browning BL, Zhou Y, Browning SR (2018) A one-penny imputed genome from next-generation reference panels. American Journal of Human Genetics 103:338–348. doi:10.1016/j.ajhg.2018.07.015.

Browning BL, Tian X, Zhou Y, Browning SR (2021) Fast two-stage phasing of large-scale sequence data. American Journal of Human Genetics 108:1880–1890. doi:10.1016/j.ajhg.2021.08.005.

Browning BL, Browning SR (2022) Genotype error biases trio-based estimates of haplotype phase accuracy. American Journal of Human Genetics 109:1016–1025. doi:10.1016/j.ajhg.2022.04.019.

A Beagle comparison

```
jar <- list.files(".",pattern="^beagle.*[.]jar$",full.names=TRUE)[1]
st.copy.marker.data("f2True","f2.true")

comparison <- data.frame(
    Method=character(),
    `Switch error (%)`=numeric(),
    `Time (s)`=numeric(),
    check.names=FALSE
)
```

The Beagle runs use an explicit PLINK-format genetic map. SelectionTools map positions are in centiMorgans; the final column reproduces the integer VCF positions written by `write.vcf()`.

```
beagleMap <- st.get.map(data.set="tropical")
beagleMap$BP <- round(beagleMap$Pos * 1000000) + 1
for (chrom in unique(beagleMap$Chrom)) {
    chrom.index <- which(beagleMap$Chrom == chrom)
    if (length(chrom.index) > 1) {
        for (j in 2:length(chrom.index)) {
            if (beagleMap$BP[chrom.index[j]] <=
                beagleMap$BP[chrom.index[j-1]])
                beagleMap$BP[chrom.index[j]] <-
                    beagleMap$BP[chrom.index[j-1]] + 1
        }
    }
}
write.table(
    beagleMap[,c("Chrom","Name","Pos","BP")],
    file="tropical-beagle.map",
    quote=FALSE,
    row.names=FALSE,
    col.names=FALSE
)
```

Beagle requires reference genotypes to use the phased VCF separator. The two F2 founders are homozygous at every retained marker, so replacing the VCF separator changes only the file representation, not the biological phase.

```
write.vcf("parentsBgRef.vcf",data.set="parents")
parentsBgRef <- readLines("parentsBgRef.vcf")
parentsBgRef <- gsub(
```

```
    "([0-9]+)/([0-9]+)",
    "\\1|\\2",
    parentsBgRef
)
parentsBgRefCon <- gzfile("parentsBgRef.vcf.gz","wt")
writeLines(parentsBgRef,parentsBgRefCon)
close(parentsBgRefCon)
unlink("parentsBgRef.vcf")
```

At this point the simulation backend still contains `popSYN1` from the cohort example. Repeated random mating within each successive population gives five generations of intermating in total before the simulation backend is reinitialized for the other populations.

```
cross("popSYN2","popSYN1","popSYN1",NoPg=n.base)
cross("popSYN3","popSYN2","popSYN2",NoPg=n.base)
cross("popSYN4","popSYN3","popSYN3",NoPg=n.base)
cross("popSYN5","popSYN4","popSYN4",NoPg=n.base)
st.get.simpop("popSYN5",data.set="popSYN5True")
st.copy.marker.data("popSYN1True","popSYN1.true")
```

For the five-line series, the founders are selected reproducibly with R’s random number generator. Only markers at which all five founders are observed and homozygous are retained. Their reference phase is therefore unambiguous. The cohort and Beagle analyses use the same marker set because the progeny are simulated from this restricted founder set.

```
set.seed(57721)
sel5.ids <- sample(
    as.character(st.marker.data.statistics(
        data.set="tropical",
        mar=FALSE,
        gen=FALSE
    )$individual.list$Name),
    5
)
st.copy.marker.data("sel5Ref","tropical")
st.restrict.marker.data(
    ind.list=sel5.ids,
    data.set="sel5Ref"
)
sel5.markers <- st.get.map(data.set="sel5Ref")$Name
for (sel5.id in sel5.ids) {
    st.copy.marker.data("sel5Tmp","tropical")
    st.restrict.marker.data(
        ind.list=sel5.id,
        data.set="sel5Tmp"
    )
    st.restrict.marker.data(
        NoAll.MAX=1,
        MaMis.MAX=0,
        data.set="sel5Tmp"
    )
    sel5.markers <- intersect(
        sel5.markers,
        st.get.map(data.set="sel5Tmp")$Name
    )
}
st.restrict.marker.data(
    mar.list=sel5.markers,
    data.set="sel5Ref"
)
```

The five selected founders are likewise complete and homozygous on this marker set. A Beagle reference VCF is therefore made by writing the same founder genotypes and changing only the homozygous GT separator.

```
write.vcf("sel5BgRef.vcf",data.set="sel5Ref")
sel5BgRef <- readLines("sel5BgRef.vcf")
sel5BgRef <- gsub(
    "([0-9]+)/([0-9]+)",
    "\\1|\\2",
    sel5BgRef
)
sel5BgRefCon <- gzfile("sel5BgRef.vcf.gz","wt")
writeLines(sel5BgRef,sel5BgRefCon)
close(sel5BgRefCon)
unlink("sel5BgRef.vcf")


st.set.simpop("sel5",data.set="sel5Ref")
cross("sel5SYN1","sel5","sel5",NoPg=n.base)
st.get.simpop("sel5SYN1",data.set="sel5SYN1True")
cross("sel5SYN2","sel5SYN1","sel5SYN1",NoPg=n.base)
cross("sel5SYN3","sel5SYN2","sel5SYN2",NoPg=n.base)
cross("sel5SYN4","sel5SYN3","sel5SYN3",NoPg=n.base)
cross("sel5SYN5","sel5SYN4","sel5SYN4",NoPg=n.base)
st.get.simpop("sel5SYN5",data.set="sel5SYN5True")
```

Finally, the phased F1 is restored to the simulation backend and selfed for five generations to produce S5.

```
st.set.simpop("F1",data.set="f1.reference")
ssd.mating("S5","F1",NoPg=200,maxcycles=5)
st.get.simpop("S5",data.set="s5True")
```

For every benchmark population a single phase-destroyed data set is prepared first. SelectionTools cohort mode, SelectionTools reference mode where available, and Beagle therefore start from identical unordered genotypes.

```
st.copy.marker.data("f2Unphased","f2True")
set.seed(314159)
st.destroy.phase(data.set="f2Unphased")
st.copy.marker.data("f2STcoh","f2Unphased")
st.copy.marker.data("f2STref","f2Unphased")


time <- system.time(
    st.phase(
        population.type="F2",
        impute=FALSE,
        data.set="f2STcoh"
    )
)
score <- st.switch.error(
    true.data.set="f2True",
    data.set="f2STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "F2 SelectionTools-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

```
time <- system.time(
    st.phase(
        reference.data.set="parents",
        population.type="F2",
        impute=FALSE,
        data.set="f2STref"
    )
)
score <- st.switch.error(
    true.data.set="f2True",
    data.set="f2STref"
)
comparison[nrow(comparison)+1,] <- list(
    "F2 SelectionTools-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

Beagle is first run without a reference panel. The second run receives the same two founders used by SelectionTools reference mode.

```
write.vcf("f2Unphased.vcf.gz",data.set="f2Unphased")
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=f2Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=f2BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("f2BgCoh.vcf.gz",data.set="f2BgCoh")
score <- st.switch.error(
    true.data.set="f2True",
    data.set="f2BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "F2 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=f2Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "ref=parentsBgRef.vcf.gz",
            "out=f2BgRef",
            "impute=false",
            "nthreads=4"
        )
```

```
    )
)


read.vcf("f2BgRef.vcf.gz",data.set="f2BgRef")
score <- st.switch.error(
    true.data.set="f2True",
    data.set="f2BgRef"
)
comparison[nrow(comparison)+1,] <- list(
    "F2 Beagle-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

The S5 cohort and reference runs use the same theoretical 1.9375 breaks per Morgan. The reference run uses the two phased founders; the cohort run uses no reference haplotypes.

```
st.copy.marker.data("s5Unphased","s5True")
set.seed(141421)
st.destroy.phase(data.set="s5Unphased")
st.copy.marker.data("s5STcoh","s5Unphased")
st.copy.marker.data("s5STref","s5Unphased")


time <- system.time(
    st.phase(
        expected.breaks.per.morgan=1.9375,
        impute=FALSE,
        data.set="s5STcoh"
    )
)
score <- st.switch.error(
    true.data.set="s5True",
    data.set="s5STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "S5 SelectionTools-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


time <- system.time(
    st.phase(
        reference.data.set="parents",
        expected.breaks.per.morgan=1.9375,
        impute=FALSE,
        data.set="s5STref"
    )
)
score <- st.switch.error(
    true.data.set="s5True",
    data.set="s5STref"
)
comparison[nrow(comparison)+1,] <- list(
    "S5 SelectionTools-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


write.vcf("s5Unphased.vcf.gz",data.set="s5Unphased")
time <- system.time(
```

```
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=s5Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=s5BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("s5BgCoh.vcf.gz",data.set="s5BgCoh")
score <- st.switch.error(
    true.data.set="s5True",
    data.set="s5BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "S5 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=s5Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "ref=parentsBgRef.vcf.gz",
            "out=s5BgRef",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("s5BgRef.vcf.gz",data.set="s5BgRef")
score <- st.switch.error(
    true.data.set="s5True",
    data.set="s5BgRef"
)
comparison[nrow(comparison)+1,] <- list(
    "S5 Beagle-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

The complete tropical-maize founder panel contains missing genotypes, whereas reference mode requires complete reference genotypes at every target marker. Consequently `popSYN1` and `popSYN5` are compared in cohort mode only; no artificial founder filtering is introduced for these two populations.

```
st.copy.marker.data("popSYN1Unphased","popSYN1True")
set.seed(271828)
st.destroy.phase(data.set="popSYN1Unphased")
st.copy.marker.data("popSYN1STcoh","popSYN1Unphased")
```

```
time <- system.time(
    st.phase(
        impute=FALSE,
        data.set="popSYN1STcoh"
    )
)
score <- st.switch.error(
    true.data.set="popSYN1True",
    data.set="popSYN1STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "popSYN1 SelectionTools-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


write.vcf("popSYN1Unphased.vcf.gz",data.set="popSYN1Unphased")
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=popSYN1Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=popSYN1BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)
read.vcf("popSYN1BgCoh.vcf.gz",data.set="popSYN1BgCoh")
score <- st.switch.error(
    true.data.set="popSYN1True",
    data.set="popSYN1BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "popSYN1 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


st.copy.marker.data("popSYN5Unphased","popSYN5True")
set.seed(223607)
st.destroy.phase(data.set="popSYN5Unphased")
st.copy.marker.data("popSYN5STcoh","popSYN5Unphased")


time <- system.time(
    st.phase(
        impute=FALSE,
        data.set="popSYN5STcoh"
    )
)
score <- st.switch.error(
    true.data.set="popSYN5True",
    data.set="popSYN5STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "popSYN5 SelectionTools-coh",
    score["switch.error.percent","result"],
```

```
    unname(time["elapsed"])
)


write.vcf("popSYN5Unphased.vcf.gz",data.set="popSYN5Unphased")
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=popSYN5Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=popSYN5BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)
read.vcf("popSYN5BgCoh.vcf.gz",data.set="popSYN5BgCoh")
score <- st.switch.error(
    true.data.set="popSYN5True",
    data.set="popSYN5BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "popSYN5 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

For the five-line synthetic populations, both modes are meaningful. Cohort mode uses only the progeny cohort; reference mode additionally receives the five complete phased founders in `sel5Ref`.

```
st.copy.marker.data("sel5SYN1Unphased","sel5SYN1True")
set.seed(244949)
st.destroy.phase(data.set="sel5SYN1Unphased")
st.copy.marker.data("sel5SYN1STcoh","sel5SYN1Unphased")
st.copy.marker.data("sel5SYN1STref","sel5SYN1Unphased")


time <- system.time(
    st.phase(
        impute=FALSE,
        data.set="sel5SYN1STcoh"
    )
)
score <- st.switch.error(
    true.data.set="sel5SYN1True",
    data.set="sel5SYN1STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN1 SelectionTools-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


time <- system.time(
    st.phase(
        reference.data.set="sel5Ref",
        population.type="Finfty",
        impute=FALSE,
```

```
        data.set="sel5SYN1STref"
    )
)
score <- st.switch.error(
    true.data.set="sel5SYN1True",
    data.set="sel5SYN1STref"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN1 SelectionTools-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


write.vcf("sel5SYN1Unphased.vcf.gz",data.set="sel5SYN1Unphased")
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=sel5SYN1Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=sel5SYN1BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("sel5SYN1BgCoh.vcf.gz",data.set="sel5SYN1BgCoh")
score <- st.switch.error(
    true.data.set="sel5SYN1True",
    data.set="sel5SYN1BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN1 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=sel5SYN1Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "ref=sel5BgRef.vcf.gz",
            "out=sel5SYN1BgRef",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("sel5SYN1BgRef.vcf.gz",data.set="sel5SYN1BgRef")
score <- st.switch.error(
    true.data.set="sel5SYN1True",
    data.set="sel5SYN1BgRef"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN1 Beagle-ref",
```

```
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


st.copy.marker.data("sel5SYN5Unphased","sel5SYN5True")
set.seed(264575)
st.destroy.phase(data.set="sel5SYN5Unphased")
st.copy.marker.data("sel5SYN5STcoh","sel5SYN5Unphased")
st.copy.marker.data("sel5SYN5STref","sel5SYN5Unphased")


time <- system.time(
    st.phase(
        impute=FALSE,
        data.set="sel5SYN5STcoh"
    )
)
score <- st.switch.error(
    true.data.set="sel5SYN5True",
    data.set="sel5SYN5STcoh"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN5 SelectionTools-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


time <- system.time(
    st.phase(
        reference.data.set="sel5Ref",
        population.type="Finfty",
        impute=FALSE,
        data.set="sel5SYN5STref"
    )
)
score <- st.switch.error(
    true.data.set="sel5SYN5True",
    data.set="sel5SYN5STref"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN5 SelectionTools-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)


write.vcf("sel5SYN5Unphased.vcf.gz",data.set="sel5SYN5Unphased")
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=sel5SYN5Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "out=sel5SYN5BgCoh",
            "impute=false",
            "nthreads=4"
        )
    )
)
```

```
read.vcf("sel5SYN5BgCoh.vcf.gz",data.set="sel5SYN5BgCoh")
score <- st.switch.error(
    true.data.set="sel5SYN5True",
    data.set="sel5SYN5BgCoh"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN5 Beagle-coh",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
time <- system.time(
    system2(
        command="java",
        args=c(
            "-jar",
            shQuote(jar),
            "gt=sel5SYN5Unphased.vcf.gz",
            "map=tropical-beagle.map",
            "ref=sel5BgRef.vcf.gz",
            "out=sel5SYN5BgRef",
            "impute=false",
            "nthreads=4"
        )
    )
)


read.vcf("sel5SYN5BgRef.vcf.gz",data.set="sel5SYN5BgRef")
score <- st.switch.error(
    true.data.set="sel5SYN5True",
    data.set="sel5SYN5BgRef"
)
comparison[nrow(comparison)+1,] <- list(
    "sel5SYN5 Beagle-ref",
    score["switch.error.percent","result"],
    unname(time["elapsed"])
)
```

The switch-error percentage and elapsed time are added to `comparison` immediately after each run. Only the row names are prepared before printing; the final data frame is rounded by ordinary R code.

```
rownames(comparison) <- comparison$Method
comparison$Method <- NULL
round(comparison,2)
```

```
                          Switch error (%) Time (s)
F2 SelectionTools-coh                 4.58     2.39
F2 SelectionTools-ref                 3.65     2.35
F2 Beagle-coh                         6.58     1.84
F2 Beagle-ref                         7.36     1.87
S5 SelectionTools-coh                 4.47     1.63
S5 SelectionTools-ref                 5.18     1.77
S5 Beagle-coh                         6.81     1.66
S5 Beagle-ref                         7.17     1.87
popSYN1 SelectionTools-coh            8.01    15.27
popSYN1 Beagle-coh                   25.48     3.12
popSYN5 SelectionTools-coh           19.70    15.47
popSYN5 Beagle-coh                   26.00     2.91
sel5SYN1 SelectionTools-coh           0.00     5.14
sel5SYN1 SelectionTools-ref           0.00     4.72
```

```
sel5SYN1 Beagle-coh                    4.23     2.48
sel5SYN1 Beagle-ref                    4.79     2.60
sel5SYN5 SelectionTools-coh            9.44     7.65
sel5SYN5 SelectionTools-ref            7.32     4.87
sel5SYN5 Beagle-coh                   11.62     2.44
sel5SYN5 Beagle-ref                   12.22     2.83
```