PopPsiSeq and Helper Functions

Contents

Introduction

PopPsiSeq is a bioinformatics workflow intended for analyzing the results of evolution & resequencing experiments. It builds on previous software published in Earley and Jones (2011) and has been used to analyze hybrid sterility between Drosophila simulans and mauritiana (Kanippayoor et al. (2024)).

PopPsiSeqR bundles its key functions into a convenient R package. This vignette presents a basic use case.

Hybridize/Select/Backcross/Resequence Experiments

The underlying idea in an experiment of this sort is that the genetic basis of a trait can be mapped by hybridizing genomes that differ in that trait, then selecting for the trait associated with one genome while backcrossing to the other. The procedure is sketched out in Earley and Jones (2011) :

> Populations with a divergent complex trait are hybridized and then selected for a specific phenotype across multiple generations of backcrosses. [...] The trait of interest is selected for each generation, and offspring are mated to the other parental line expressing the unselected phenotype (introgression and backcrossing). Over multiple generations of selection and backcrossing this hybrid population becomes homozygous for the majority of the unselected parent's genome while loci from the selected parent, which contribute to the selected trait, remain.

The experiment design is analogous to older techniques which use discrete, pre-identified molecular markers. Modern high-throughput sequencing allows finer mapping at a greater scale, but requires modern software to extract meaningful results from raw sequence.

Building and Loading Example Data

After an experiment has been performed and reads have been sequenced, a typical workflow ahead of PopPsiSeq will include

- Read filtering and quality control
- Mapping of reads to a reference genome
- Calling genome variants from mapped reads

Example workflows starting from sequenced reads are documented on github: https://github.com/csoeder/PopPsiSeq

In this vignette we will start with an example dataset, in which allele frequencies have been measured for three populations:

- a Drosophila simulans population, constructed from sequenced lab strains downloaded from NCBI; this was treated as a parental population
- a Drosophila sechellia population, constructed from sequenced wild and lab strains downloaded from NCBI; this was treated as a parental population; for some examples these flies will be subdivided by strain source.
- an offspring population from evolve & resequence experiments, in which simulans/sechellia hybrids were repeatedly backcrossed to sechellia while selecting for simulans-like traits (@Earley2011 and unpublished followup experiments)

```
bedtools intersect -wa -wb -a {input.par1_frq} -b  {input.par2_frq} \
  | bedtools intersect -wa -wb -a - -b  {input.off_frq} \
  | cut -f 1,2,4-6,10,12,16,18 | tr ":" "\\t" \
  | awk -v OFS='\\t' '{{print $1,$2,$2+1,"0","0","+",$4,$6,$3,$7,$8,$10,$11,$13}}' \
  > {output.frqShft_out}.pre
```

The *{input.*_frq}* files are the BEDfile-ification () of the VCFtools (Danecek et al. (2011)) allele frequency utility:

```
vcftools --vcf {input.vcf_in} --out {output.report_out} --freq;
cat {output.report_out}.frq | tail -n +2 | awk -v OFS='\\t' '{{print $1,$2,$2+1,$4,$5,$6}}'
```

The full allele frequency measurements have been trimmed to the neighborhood of a feature identified in Earley and Jones (2011) .

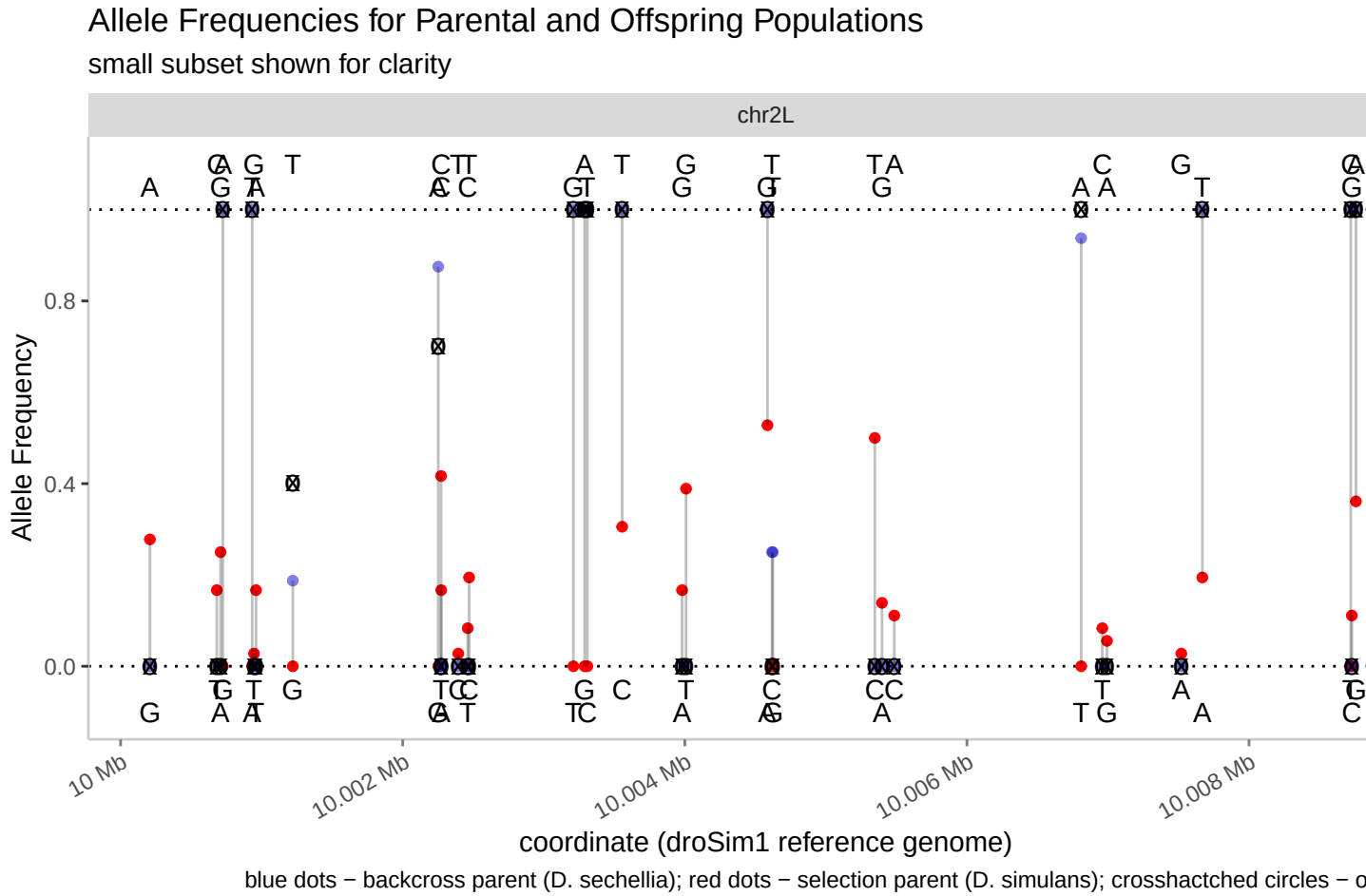
```
shifted_freqs=data/intermediate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_allSech.vs_dr
# this file is generated by the PopPsiSeq workflows documented elsewhere

bedtools intersect -wa -a $shifted_freqs  -b <( echo -e "chr2L\t10000000\t15000000" )  \
  > inst/extdata/merged_frequencies.example_data.tbl
```

This is then loaded with the PopPsiSeqR utility, **`import.freqtbl()`** :

```
merged_frequencies.filename <- system.file("extdata",
  "merged_frequencies.example_data.tbl", package = "PopPsiSeqR")
merged_frequencies.bg <- import.freqtbl(merged_frequencies.filename)
```

The data is loaded as an extended GRanges object; the key fields (beyond those related to the BED6 structure) are the allele frequencies (columns named **_af*). Let's take a look.

```
head(merged_frequencies.bg %>% as.data.frame()
     %>% select(-c(ends_with("_count"), "name", "score"))
     %>% GRanges(), n=5)
#> GRanges object with 5 ranges and 5 metadata columns:
#>       seqnames    ranges strand |         ref         alt
#>          <Rle> <IRanges>  <Rle> | <character> <character>
#>   [1]     chr2L  10000208      * |           G           A
#>   [2]     chr2L  10000682      * |           T           C
#>   [3]     chr2L  10000709      * |           A           G
#>   [4]     chr2L  10000725      * |           G           A
#>   [5]     chr2L  10000933      * |           A           T
#>       selected_parent_alt_af backcrossed_parent_alt_af offspring_alt_af
#>                    <numeric>                 <numeric>        <numeric>
#>   [1]               0.277778                         0                0
#>   [2]               0.166667                         0                0
#>   [3]               0.250000                         0                0
#>   [4]               0.000000                         1                1
#>   [5]               0.000000                         1                1
#>   -------
#>   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

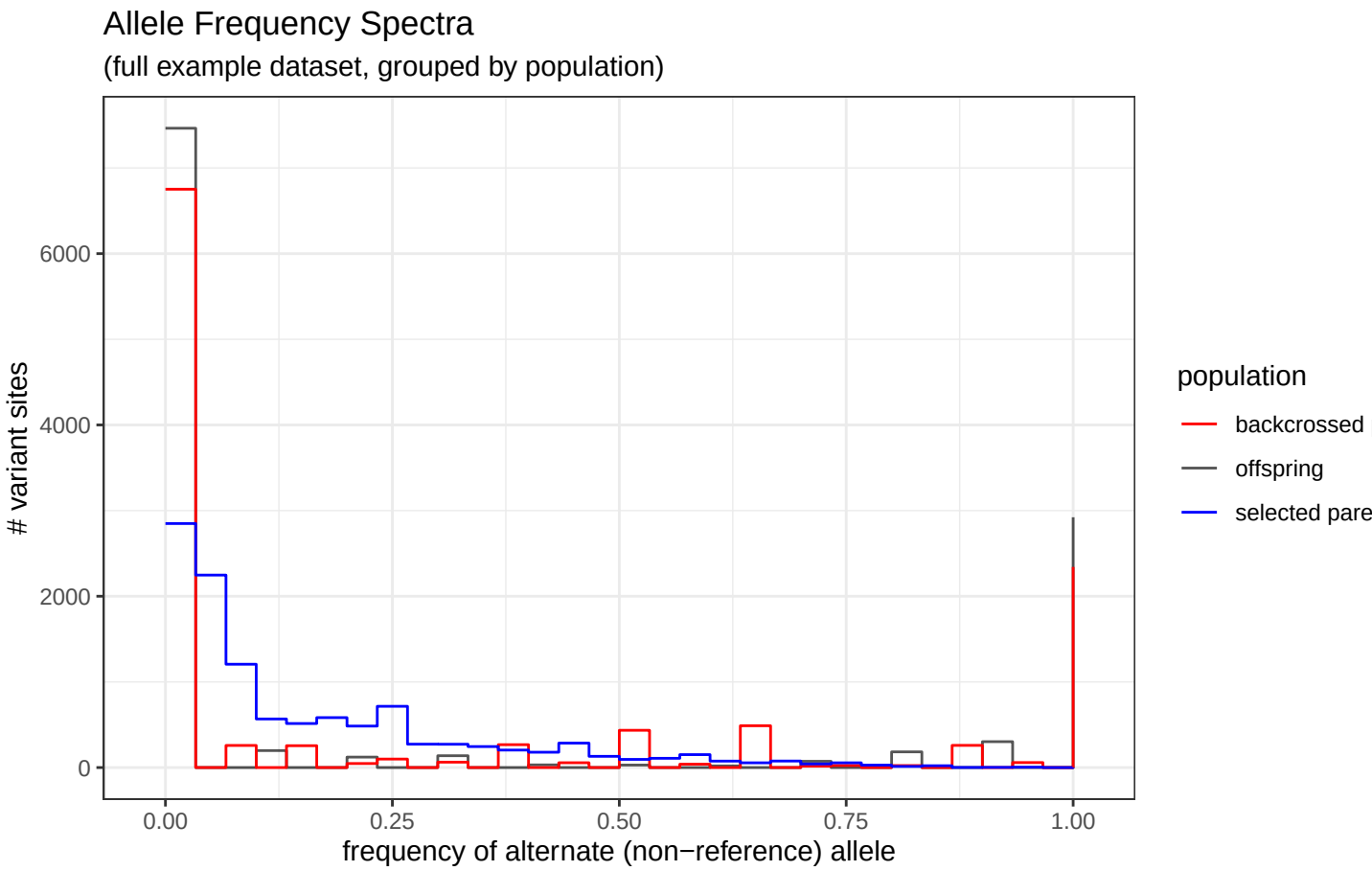


Here the allele frequencies are plotted across the chromosome arm (for visibility, only the first few dozen diagnostic SNPs are shown). The allele frequencies of the population standing in for the selected-for parent species (D. simulans) are displayed in red; the allele frequencies of the population standing in for the backcrossed-to parent species (D. sechellia) are displayed in blue. The allele frequencies of the offspring population are displayed as a black crosshatched circle. The allele appearing in the reference genome (droSim1) is at the bottom at zero; the alternate allele appears at the top at 1. For example, at the very left of the plot we have the SNP described in the first line of the data table above: the backcross population and the offspring population both have an alternate allele frequency of 0 (that is, they are 100% G and 0% A), while the selected-for population carries the A allele at ~25%.

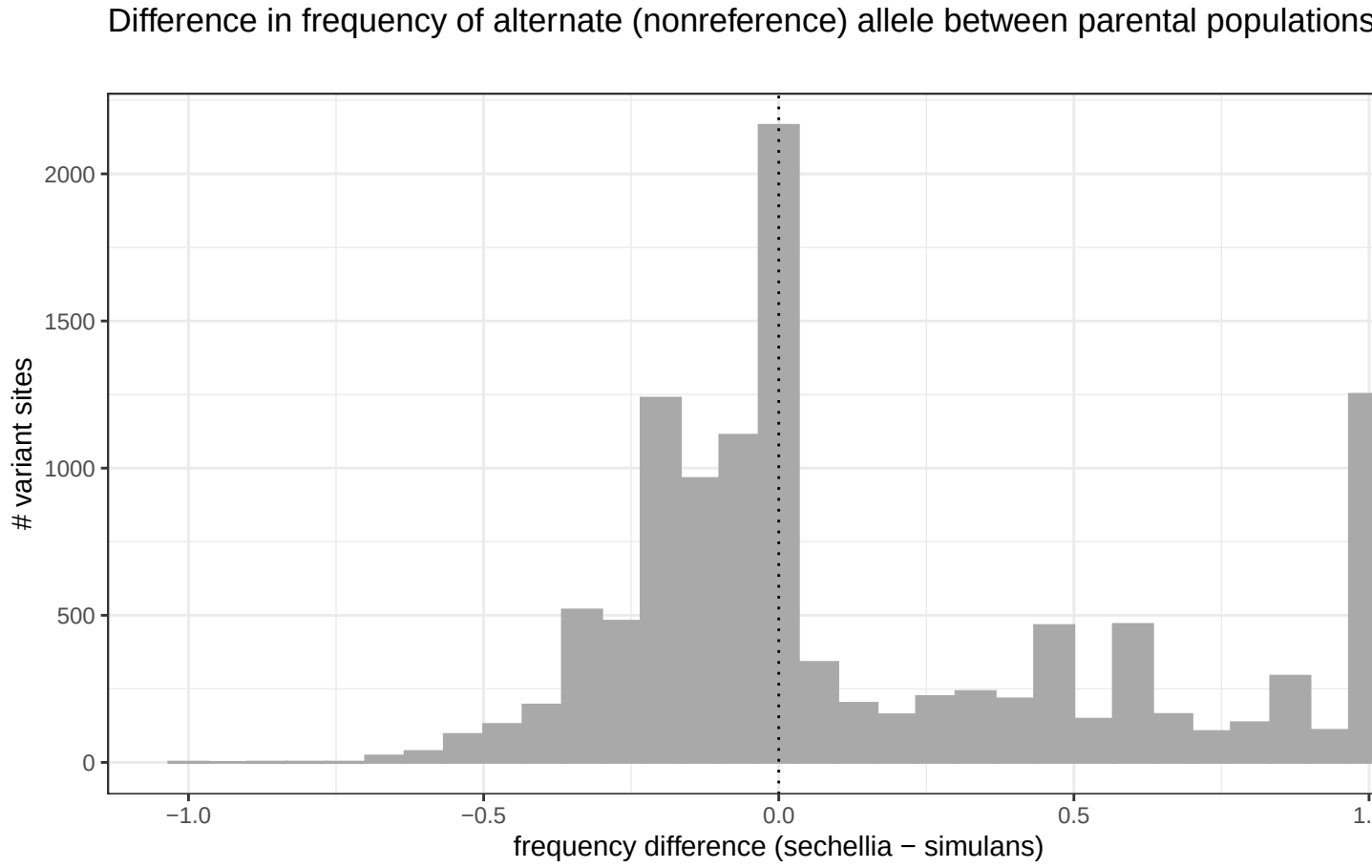
A few observations:

- The population representing the selected-for parental species (D. simulans) tends to have a low alternate allele frequency; indeed, the distribution appears to be centered near zero, which is not surprising, considering that it is the degree of difference from a simulans reference genome which is being measured. However, this population is not uniformly identical to the reference genome. In fact, this population has much more density in the intermediate frequencies, than does the D. sechellia population; this is consistent with the higher within-population diversity of simulans generally (cite this
- Although variants in the the D. sechellia population tend to be more fixed than variants in the D. simulans population, they are not necessarily fixed as the alternate allele; not only is their frequency distribution bimodal, but the peak at 0 is greater than that at 1, indicating that the allele found in the simulans reference genome is nearing fixation in sechellia more often than the alternative allele. (see below)

- The frequency distribution of the offspring is similar to that of the backcrossed-to parent.



- The alternate allele often has a higher frequency in the sechellia population, than it does in the simulans population, as might be expected given its construction. However, this is not always true; in fact, in the majority of cases, the simulans population actually has the non-reference allele at a higher frequency than does the sechellia population. These are typically sites in which the simulans population is polymorphic while the sechellia population is more fixed; this is indicated by the bulk these cases having an allele frequency difference in the [-0.5,0] range. On the other hand, the sites where the simulans population is more like the reference have an allele frequency difference ranging from 0 to 1, with the bulk clustered at 1. This indicates that these are sites where the alternate allele tends to be fixed in the sechellia population, and the reference allele tends to be fixed in simulans. There are very few cases in which the alternate allele is fixed in sechellia and the reference allele is fixed in simulans. (comment on nature of reference genome)



Sitewise Calculations

Now that we have our data loaded, it's time to run PopPsiSeq's **`freqShifter()`**.

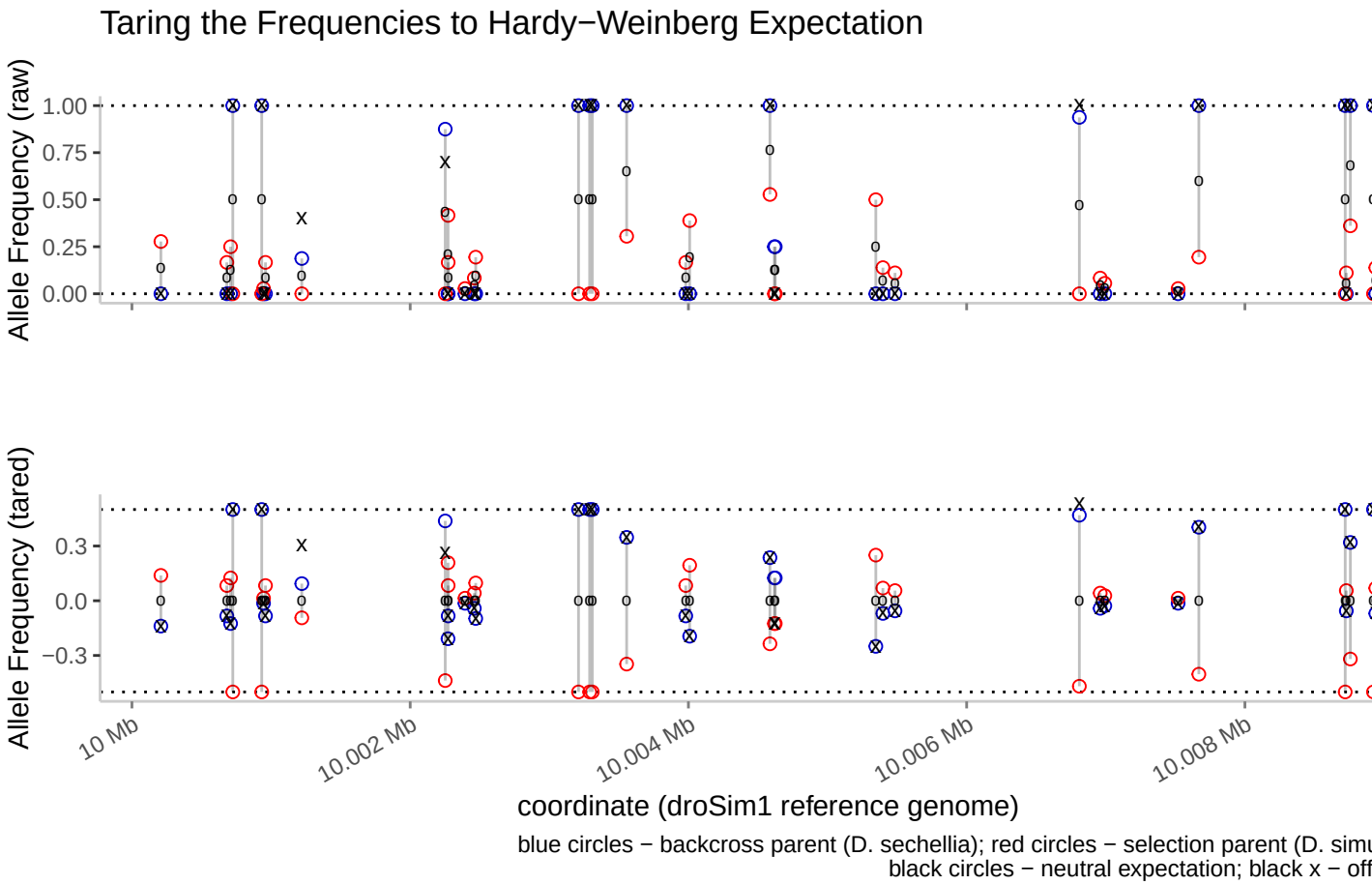
```
frequency_shifts.bg <- freqShifter(merged_frequencies.bg)

head(frequency_shifts.bg %>% as.data.frame()
     %>% select(-c(ends_with("_count"), ends_with("_deltaF"),  "name", "score"))
     %>% GRanges(), n=5)
#> GRanges object with 5 ranges and 10 metadata columns:
#>       seqnames    ranges strand |         ref         alt
#>          <Rle> <IRanges>  <Rle> | <character> <character>
#>   [1]     chr2L  10000208      * |           G           A
#>   [2]     chr2L  10000682      * |           T           C
#>   [3]     chr2L  10000709      * |           A           G
#>   [4]     chr2L  10000725      * |           G           A
#>   [5]     chr2L  10000933      * |           A           T
#>       selected_parent_alt_af backcrossed_parent_alt_af offspring_alt_af
#>                    <numeric>                 <numeric>        <numeric>
#>   [1]               0.277778                         0                0
#>   [2]               0.166667                         0                0
```

```
#>   [3]               0.250000                         0                0
#>   [4]               0.000000                         1                1
#>   [5]               0.000000                         1                1
#>         central mean_oriented_shift max_oriented_shift min_oriented_shift
#>       <numeric>           <numeric>          <numeric>          <numeric>
#>   [1] 0.1388890          -0.1388890           0.861111         -0.1388890
#>   [2] 0.0833335          -0.0833335           0.916667         -0.0833335
#>   [3] 0.1250000          -0.1250000           0.875000         -0.1250000
#>   [4] 0.5000000          -0.5000000           0.500000         -0.5000000
#>   [5] 0.5000000          -0.5000000           0.500000         -0.5000000
#>       AF_difference
#>           <numeric>
#>   [1]      0.277778
#>   [2]      0.166667
#>   [3]      0.250000
#>   [4]      1.000000
#>   [5]      1.000000
#>   -------
#>   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

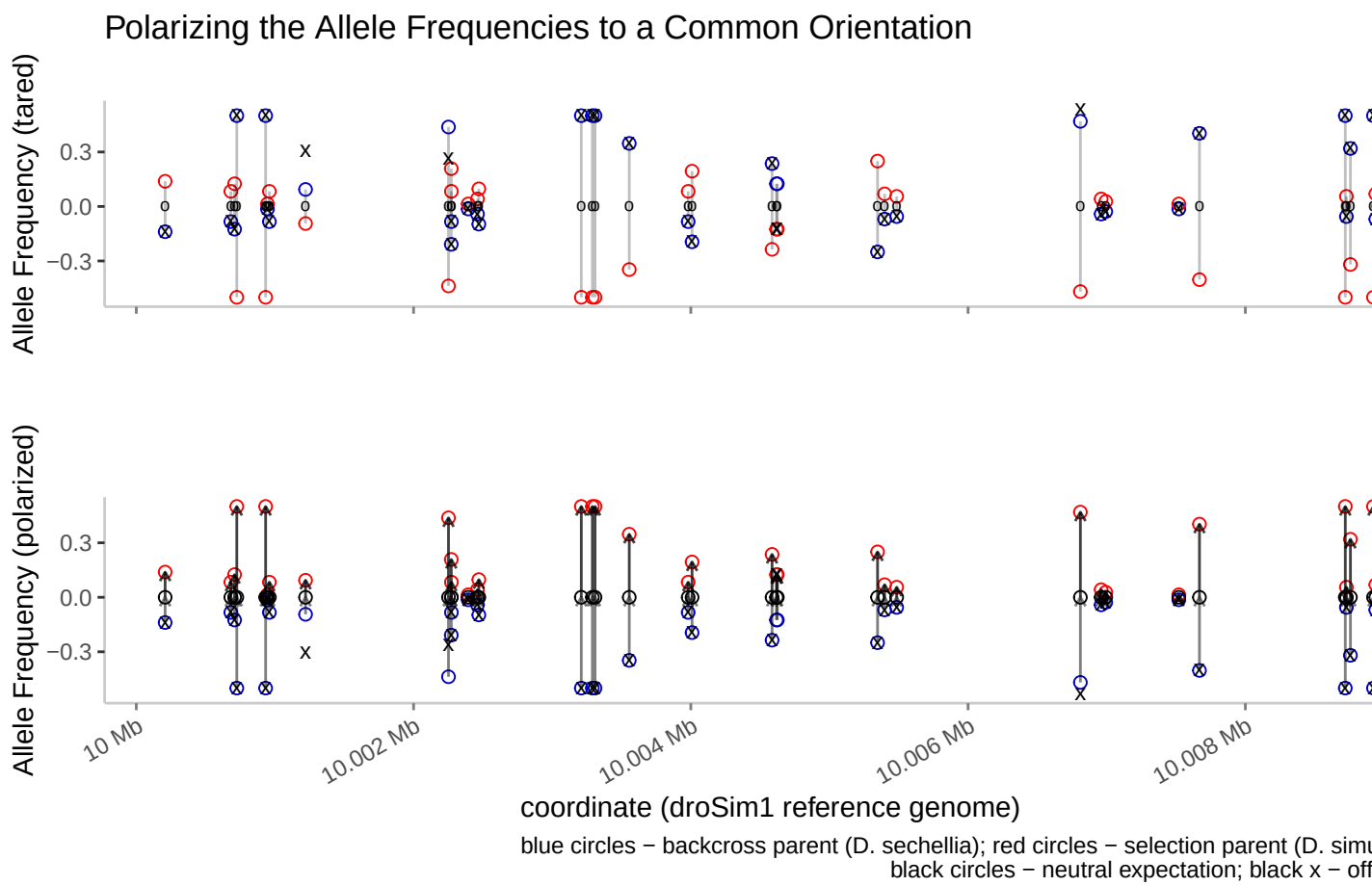
Here's what's happening within the **freqShifter**:

Consider again our evolve and resequence experiment. It began with a hybridization step in which one fly was drawn from a population of D. simulans; at each variant site our expectation for the alternate allele is the alternate allele frequency in our stand-in simulans population. When we pick one fly from the D. sechellia population, we'd also expect the allele frequency at each site to be that of the allele frequency in our stand-in sechellia population.

In the absence of higher-order evolutionary mechanisms (e.g., genome incompatibilities or meiotic drive), we'd expect the F1 generation to have, at each site, an allele frequency exactly the average of its two parents. In the absence of (net) evolutionary forces, all future offspring would also share this intermediate allele frequency, per Hardy-Weinberg. This will therefor be our reference point for understanding the allele frequencies observed experimentally. The first step then is to tare our population frequencies relative to this value.

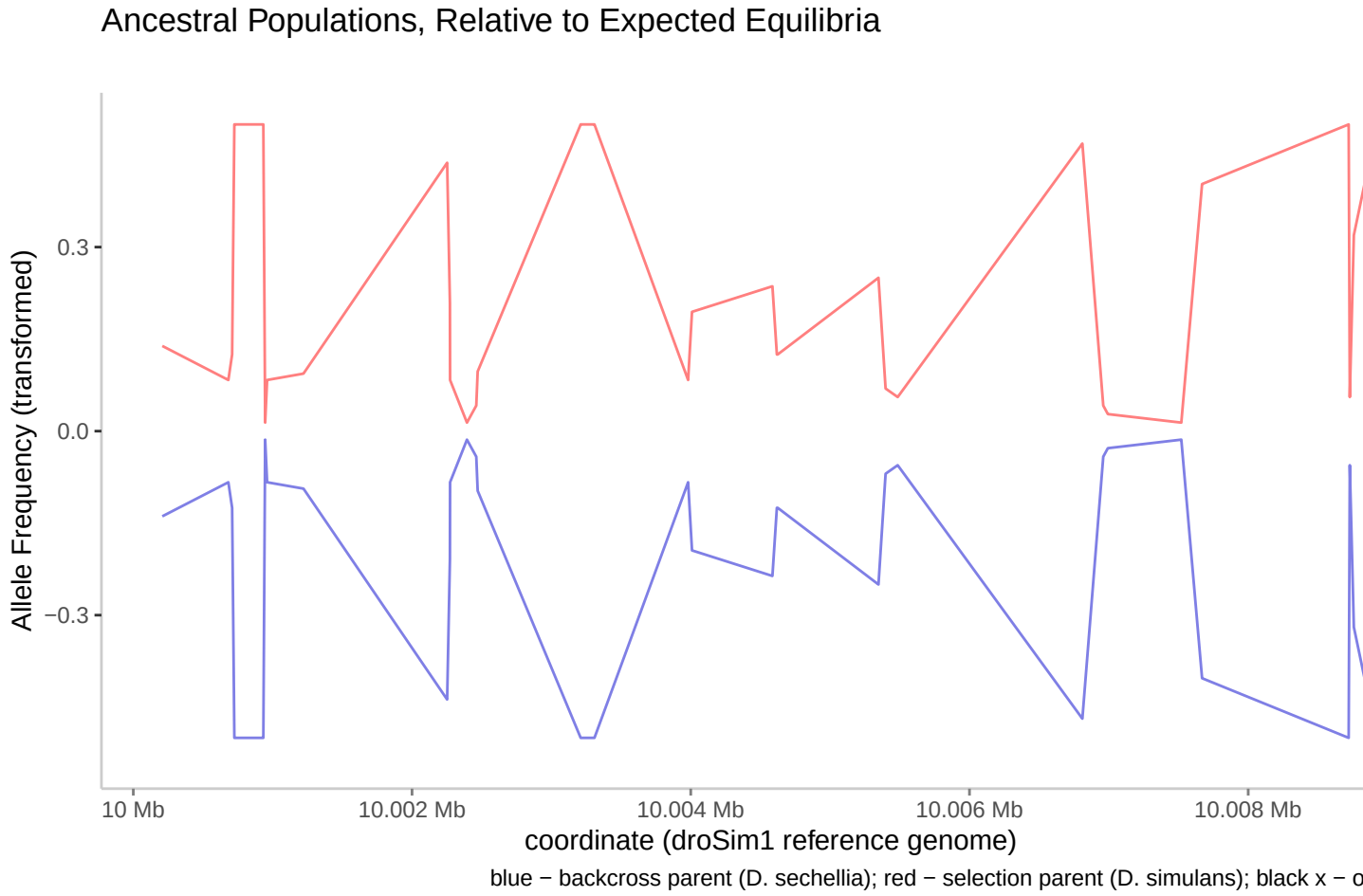


The next step is to polarize the allele frequencies such that they follow a consistent orientation. At each site there is an allele which is more common in the selected-for population and an allele which is more common in the backcrossed-to population. Sites at which the alternate allele is more common in the selected-for population are left alone; at the rest of the sites, the alternate and reference alleles were swapped. Conceptually this is like retroactively editing the reference genome to represent the backcrossed-to population; mathematically this is a simple change of sign to the tared data. Notice that the offspring allele frequencies are also polarized in this manner.



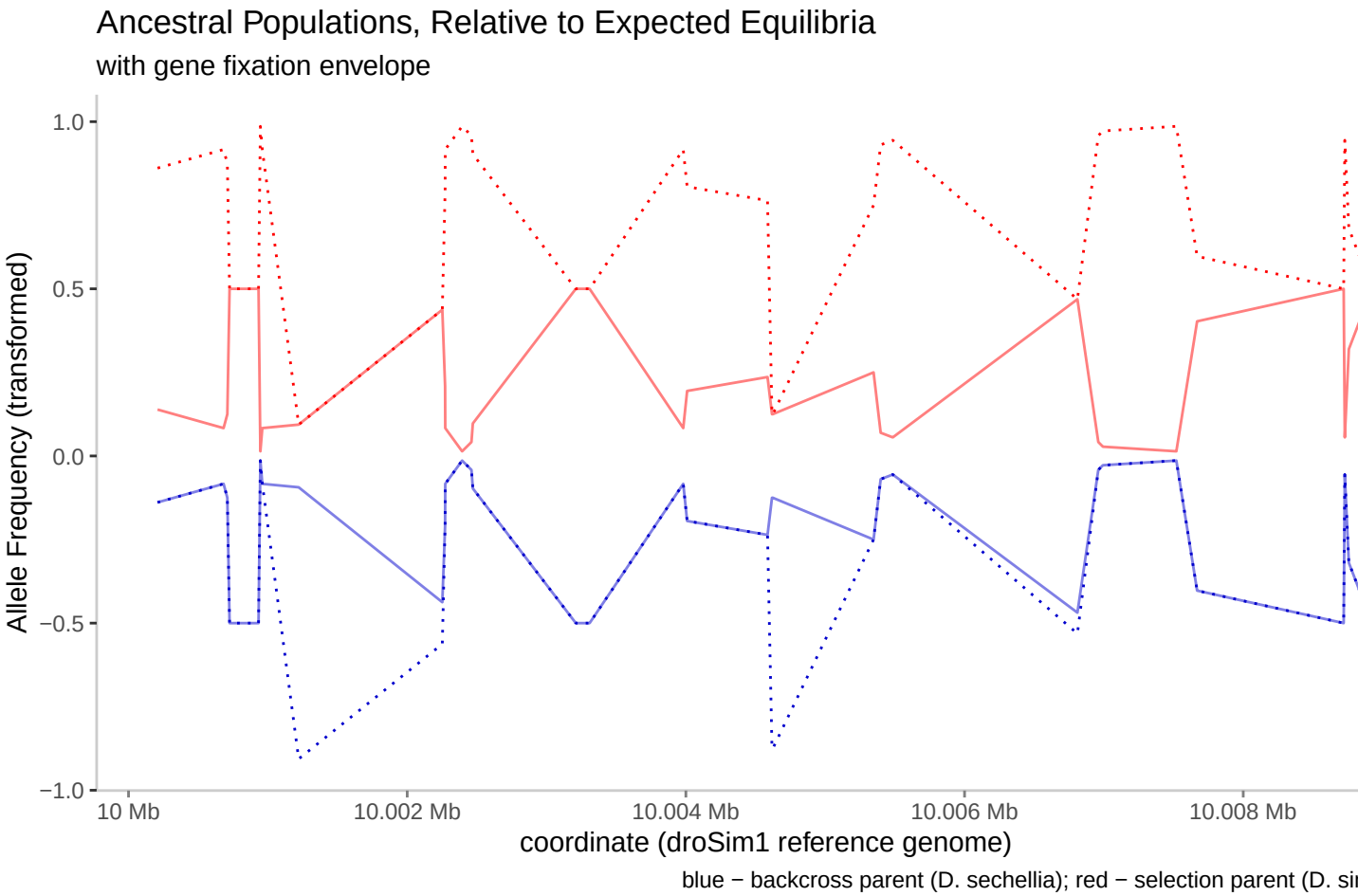
Now that all alternate alleles are characteristic of the selected-for population, allele frequency is a measure of selected-for population character, and a positive tared value indicates more selected-for character than backcrossed-to character, relative to the ideal initial hybridization.

Consider again the backcrossed-to population. Suppose there was not selection applied, just backcrossing. With each successive generation, we'd expect our hypothetical offsprings' allele frequencies to converge on those of the backcrossed-to population. We can thus place a guide on the plot which follows the tranformed allele frequency values from this population, to indicate how much our hypothetical offsprings' alleles would have to change, in order to identically resemble the backcrossed-to population. The change necessary to identically resemble the selected-for population is of equal magnitude but opposite direction, by construction. The guides for the parental populations are thus symmetric around the horizontal axis, but of variable distance from one another. The distance represents the difference in allele frequencies between the two populations at a given site, and thus how distinct the two populations are at that site; it necessarily falls in the range (0,1]. These data are not explicitly recorded in the data frame but correspond to the *center* field +/- ½ the *AF_difference* field.



Now consider a hypothetical population with no backcrossing, only selection. If the allele frequencies of the selected-for population have optimal fitness under the applied selection, then we'd expect our hypothetical offsprings' allele frequencies to converge on those of the selected-for population.

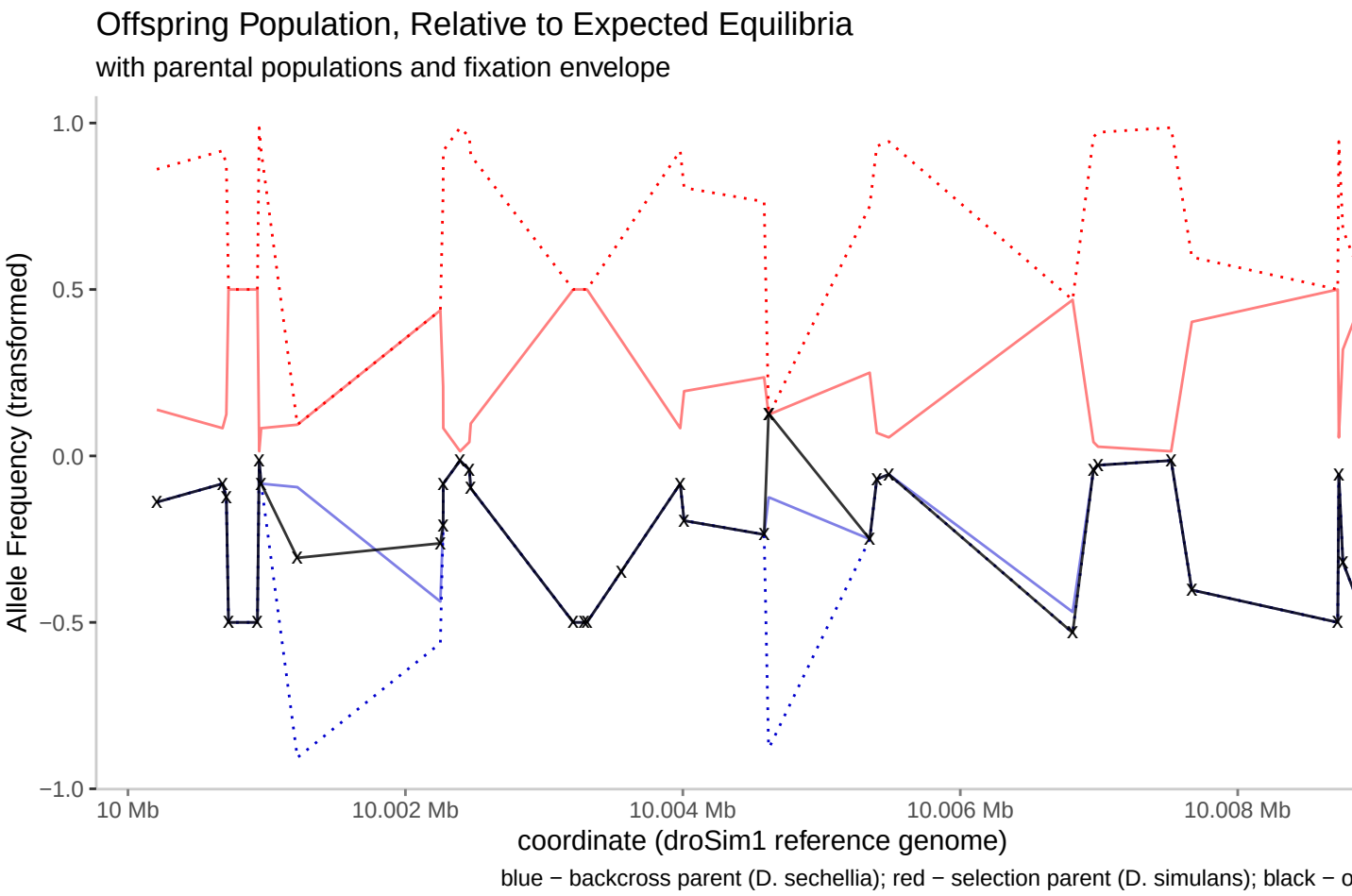
However, the optimal allele configuration is not necessarily total fixation (e.g., balancing or frequency-dependent selection); alternatively, the selected-for population might not be optimal. As a result, some sites can be polymorphic in the selected-for population. At such a site, our hypothetical offspring could conceivably attain a higher allele frequency than the parental population, up to fixation. In the same way, sites which are polymorphic in the backcrossed-to population allow our hypothetical offspring to attain allele frequencies lower than the parental population. These data are recorded in the *max_oriented_shift* and *min_oriented_shift* fields.



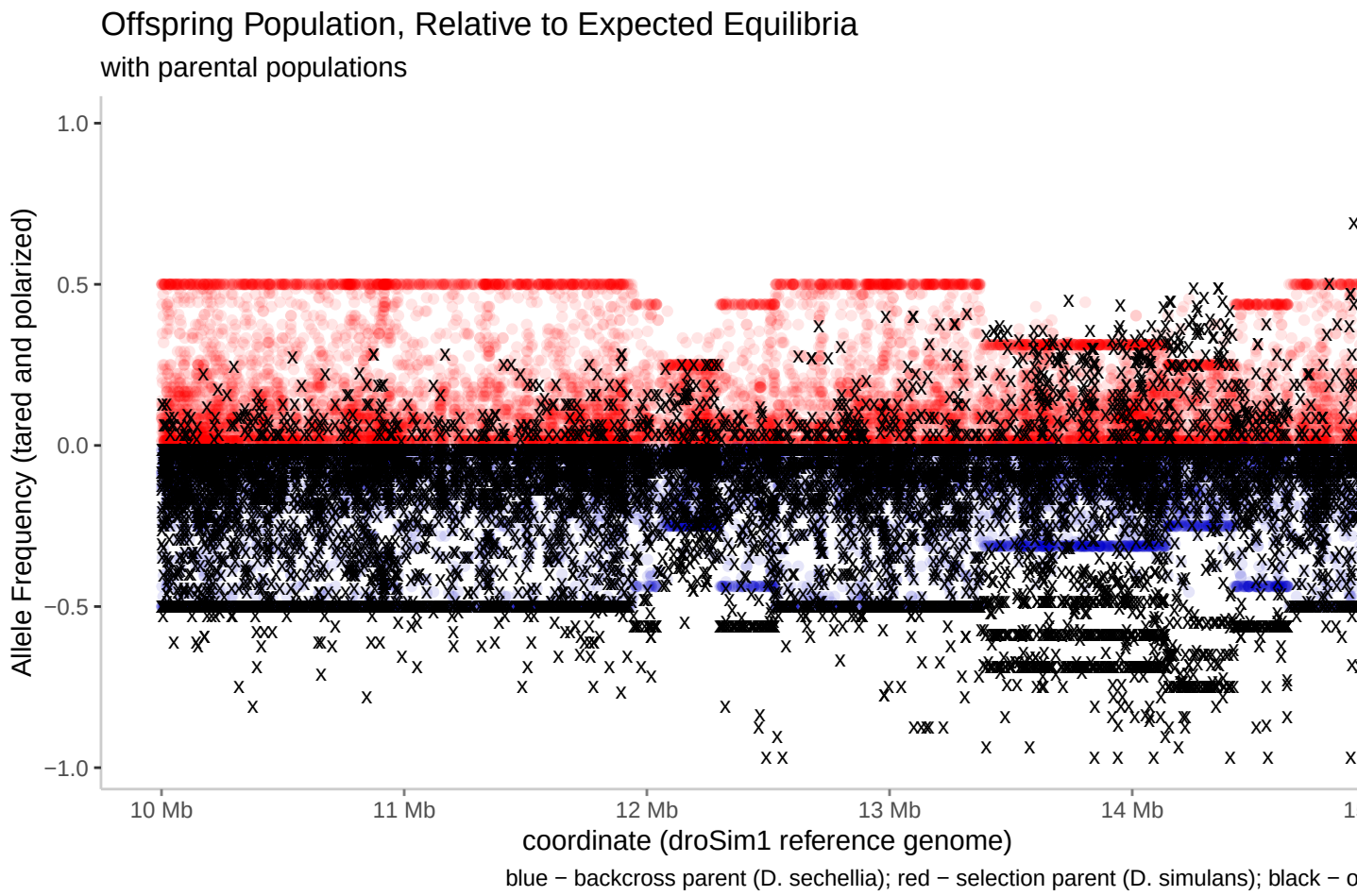
This envelope defines hard bounds on our observations; all transformed allele frequencies must appear somewhere between them. Notice that the upper and lower bounds have a constant separation of 1 unit (the maximum an allele frequency can change), but are not necessarily centered on the horizontal axis. The center of the envelope can be offset by any value in the range [-0.5, 0.5]. This offset represents relative polymorphism in the two parental populations: if one population has an allele frequency of 1 and another population has an allele frequency of 0.5, their offspring will require a larger change to fix in the direction of the second population, than in the direction of the first. Importantly it is not just the presence of polymorphism, but whether there is more in one parent population than the other. The envelope is centered on the axis at an allele with 100% frequency in one population and 0% in another; it is also centered on the axis at an allele with 49% frequency in one population and 51% in the other.

This difference in polymorphism can be interpreted as a difference in the diagnostic value of one allele versus another. Suppose the backcrossed-to population is fixed for A at a given site, while the selected-for population is half As and half Ts. If, at that site in a hybrid offspring, a T is observed, we can be confident that it descended from the selected-for population; an A observation is less informative.

With these guides in place, we can put the measured data from our offspring population in context. Unsurprisingly, the offspring closely resembles the backcrossed-to population, to the point of overplotting:



Here is the whole example dataset:



These site-wise data can then be saved to disk for additional processing or future reference with the **`export.freqshft()`** utility

```
export.freqshft(frequency_shifts.bg , tempfile())
```

Smoothing by Window

The site-wise data we've used so far are downsampled to a 5Mb region on one arm of one chromosome, and while some major features can be discerned, there's still too much data to work with easily, especially when it comes to visualization. One simplifying approach is to divide the data into contiguous chunks and to assign summary to the corresponding genomic intervals. This has two advantages:

- averaging many data points will cancel out random noise while amplifying subtle signals
- the evolutionary forces acting on nearby variant sites are not independent due to physical linkage; collapsing them into a single datum deflates the exaggerated apparent degrees of freedom

This can be done easily using the **`bedtools`**'s **`makewindow`** and **`map`** utilities ((Quinlan and Hall 2010)):

```
bedtools makewindows -w {wildcards.window_size} -s {wildcards.slide_rate} -g {fai_path} -i winnum \
  | bedtools sort -i - > {output.windowed}

bedtools map -c 7,8,9,10,11,12,12 -o sum,sum,sum,sum,sum,sum,count -null NA -a {input.windows_in} \
  -b <( tail -n +2  {input.frqShft_in}   | cut -f  1-3,15-20 | nl -n ln  | \
  awk -v OFS='\t' '{{if( $5!="NA" && $6!="NA")print $2,$3,$4,$1,"0",".",$5,$6,$7,$8,$9,$10}}' \
  | bedtools sort -i - ) > {output.windowed_out}
```

Other windowing schemes are possible. For example, the original Earley and Jones (2011) workflow used intervals containing a fixed number of SNPs, rather than of a fixed length. A python utility for producing such intervals can be found here:

https://github.com/csoeder/PopPsiSeq/blob/master/scripts/bin_by_SNPs.py

```
python3 ../exec/python/bin_by_SNPs.py --help
```

Another possibility is using a gene model annotation to define locii of interest, and measure the shift in/around them. Once a bed file of intervals has been defined, the above map command can be applied.

The full-genome dataset (from which our previous examples were subsampled) was smoothed with bookended windows 100kB wide. These were smoothed data were subsampled to just chromosome 2L for an example file:

```
windowed_freqs=data/ultimate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_allSech.vs_droSi
# this file is generated by the PopPsiSeq workflows documented elsewhere

cat $windowed_freqs | grep -w "chr2L" > inst/extdata/windowed_shifts.example_data.bed
```

This can then be imported using the **import.smvshift()** utility. This loads the file, computes the window averages from the window sums, and names the fields according to the experiment design.

```
windowed_shifts.filename <- system.file("extdata", "windowed_shifts.example_data.bed",
  package = "PopPsiSeqR")

windowed_shifts.bg <- import.smvshift(windowed_shifts.filename, selected_parent = "sim",
  backcrossed_parent = "sech")

windowed_shifts.bg %>% head()
#> GRanges object with 6 ranges and 12 metadata columns:
#>       seqnames        ranges strand |      name sum_sim_deltaF sum_sech_deltaF
#>          <Rle>     <IRanges>  <Rle> | <numeric>      <numeric>       <numeric>
#>   [1]     chr2L      1-100000      * |         1        -0.4375        137.7778
#>   [2]     chr2L 100001-200000      * |         2        -2.3250        210.8556
#>   [3]     chr2L 200001-300000      * |         3        -1.5125        101.8556
#>   [4]     chr2L 300001-400000      * |         4        -1.4000        192.5167
#>   [5]     chr2L 400001-500000      * |         5        -0.0500        156.7833
#>   [6]     chr2L 500001-600000      * |         6        -0.4000         93.2389
#>       sum_simward_AFshift max_simward_AFshift min_simward_AFshift
#>                 <numeric>           <numeric>           <numeric>
#>   [1]           -68.6701            0.713122           -0.286878
#>   [2]          -104.2653            0.726744           -0.273256
#>   [3]           -50.1715            0.791949           -0.208051
```

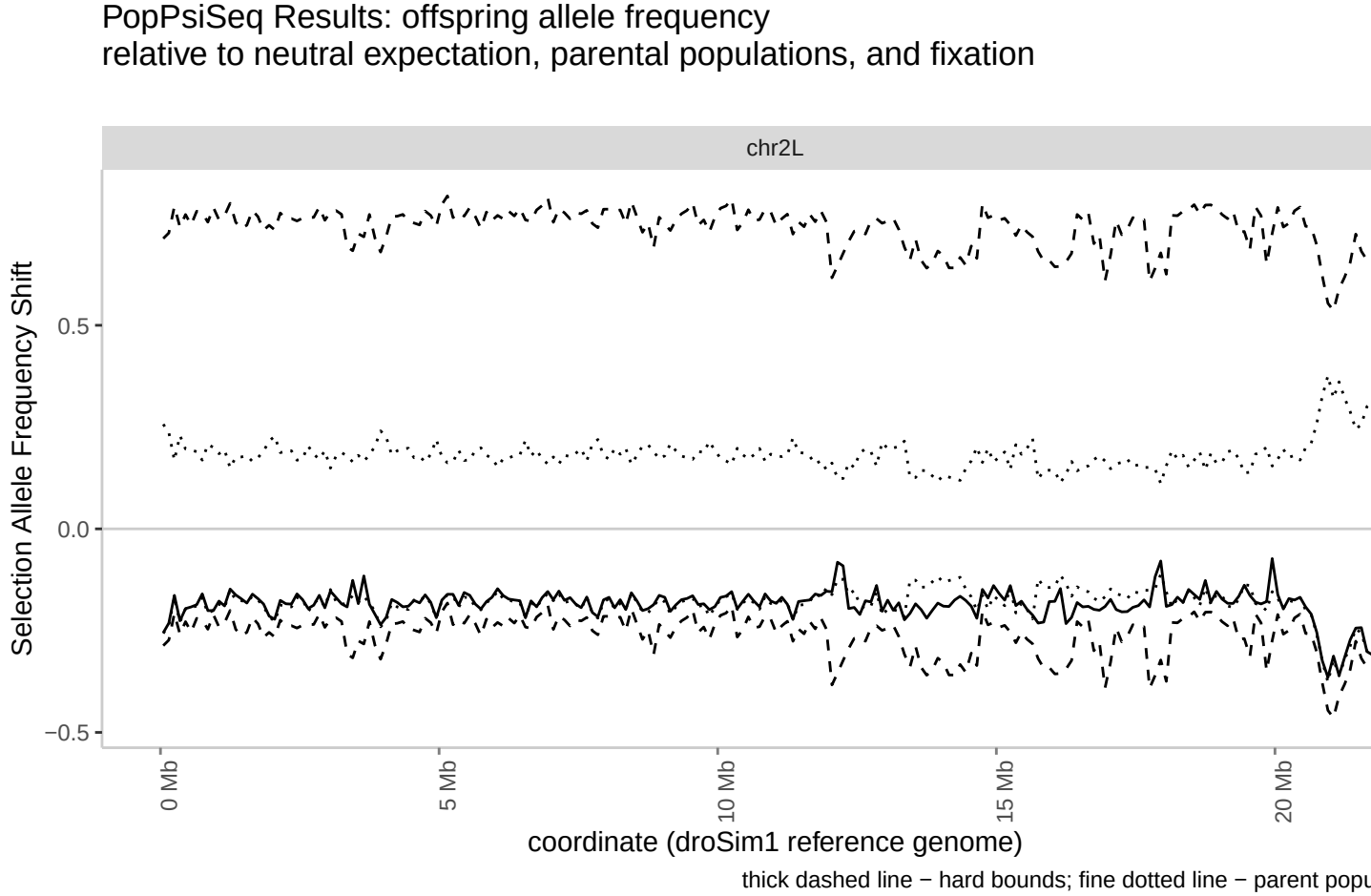
```
#>   [4]             -95.5583            0.741233          -0.258767
#>   [5]             -78.3667            0.772083          -0.227917
#>   [6]             -46.4194            0.746614          -0.253386
#>       sim_sech_difference   num_snp avg_sim_deltaF avg_sech_deltaF
#>                 <numeric> <integer>      <numeric>       <numeric>
#>   [1]            0.513811       269    -0.00162639        0.512185
#>   [2]            0.473735       450    -0.00516667        0.468568
#>   [3]            0.336704       307    -0.00492671        0.331777
#>   [4]            0.458432       423    -0.00330969        0.455122
#>   [5]            0.392083       400    -0.00012500        0.391958
#>   [6]            0.386938       242    -0.00165289        0.385285
#>       avg_simward_AFshift       win
#>                 <numeric> <integer>
#>   [1]           -0.255279         1
#>   [2]           -0.231701         2
#>   [3]           -0.163425         3
#>   [4]           -0.225906         4
#>   [5]           -0.195917         5
#>   [6]           -0.191816         6
#>   -------
#>   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Key fields are the *[min,avg,max]_*ward_AFshift*s, which correspond to the fixation bounds and the offspring average, and **_*_difference*, from which the parental population envelope is calculated in the same manner as in the per-site case.

Data Visualization

The smoothed data can be automatically displayed with the **`windowedFrequencyShift.plotter()`** utility.

```
windowedFrequencyShift.plotter(windowed_shifts.bg,
  selected_parent = "sim", backcrossed_parent = "sech", main_title =
  "PopPsiSeq Results: offspring allele frequency\nrelative to neutral expectation, parental populations
```



By default a single experiment is displayed, with the fixation bounds and the parental populations shown. Simple modifications can be passed as arguments to the **`windowedFrequencyShift.plotter()`** function or by adding ggplot commands (see below for an example including both). The resulting ggbio object can also be modified directly through its ggplot slot.

Comparing and Contrasting Results

It may be useful to compare data tracks from two different conditions. For example, it might be useful to run a control experiment to correct for effects other than the experimental treatment; it might also be useful to test the impact of changing different analysis parameters.

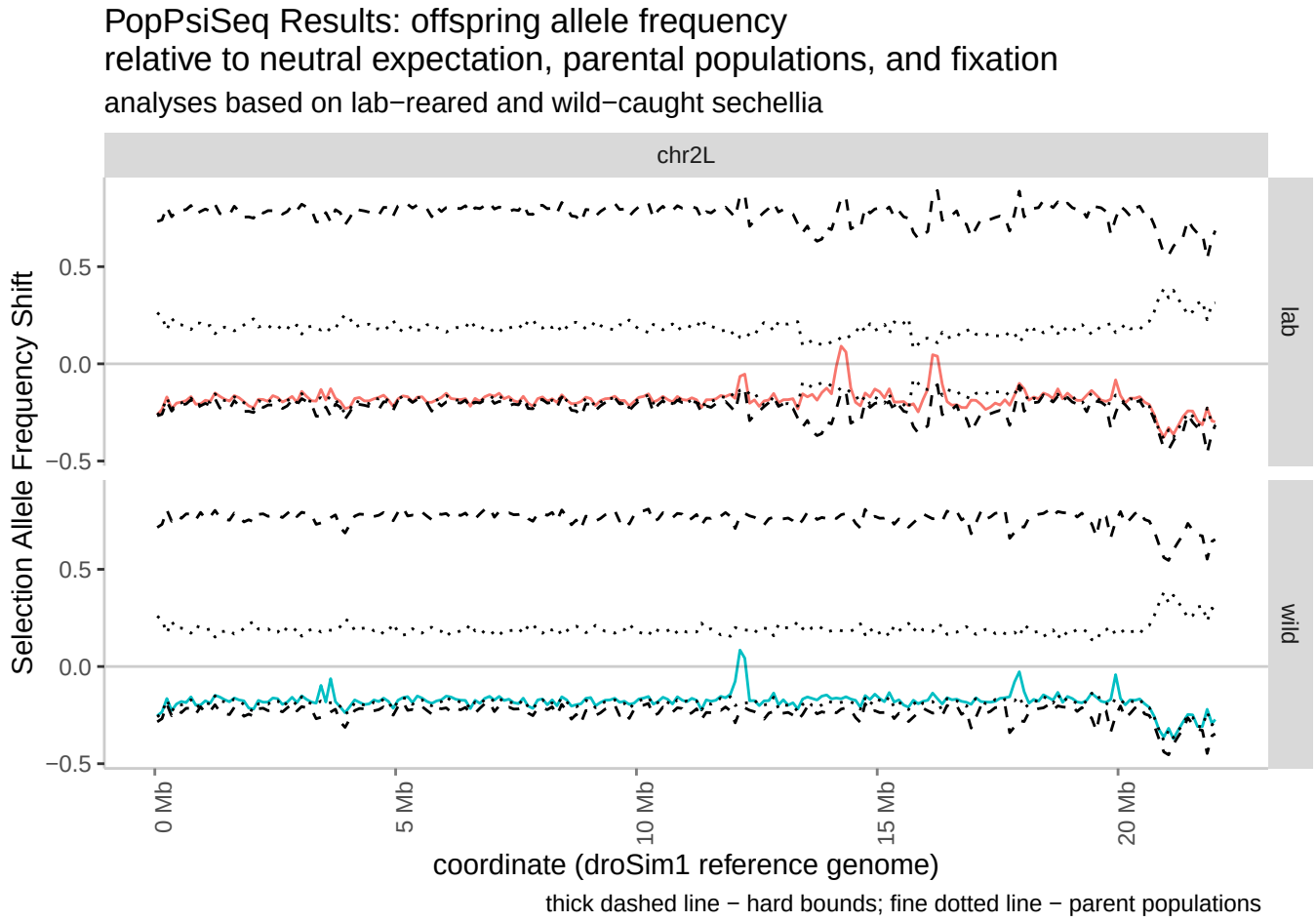
As an example, two more versions of the smoothed data above have been included; one was generated based on variants called from sechellia which were collected from the wild, the other from sechellia which had been maintained as lab culture for many generations.

```
lab_data=data/ultimate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_labSech.vs_droSim1.bwa
# this file is generated by the PopPsiSeq workflows documented elsewhere
wild_data=data/ultimate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_wildSech.vs_droSim1.b
# this file is generated by the PopPsiSeq workflows documented elsewhere

cat $wild_data | grep -w "chr2L" > inst/extdata/wild_sechellia.example_data.bed
```

```
cat $lab_data | grep -w "chr2L" > inst/extdata/lab_sechellia.example_data.bed
```

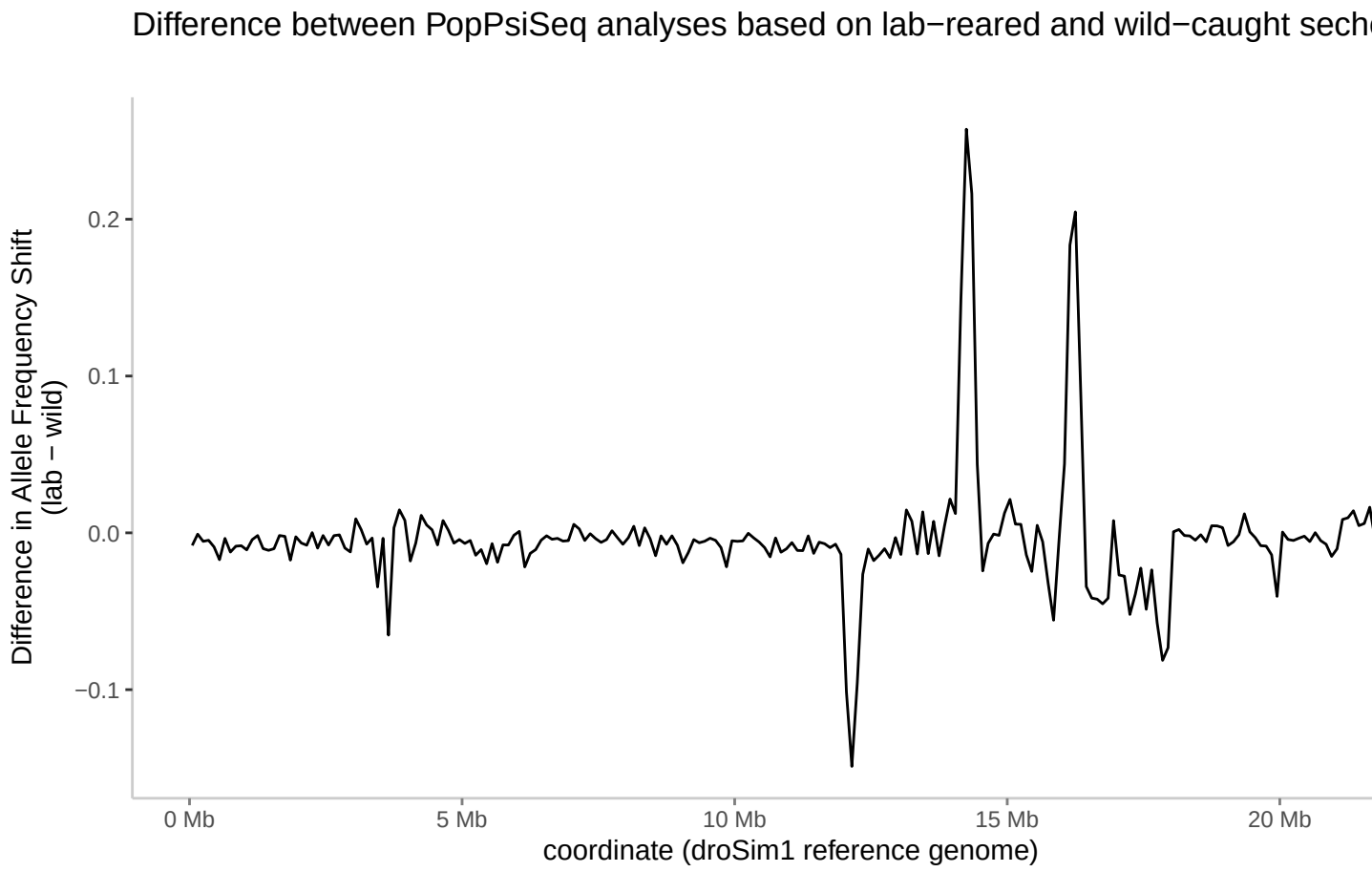
These are loaded and a field added to record the sechellia type; upon visualization, qualitative differences between the two analyses are apparent.

```
lab_sechellia.filename <- system.file("extdata",
  "wild_sechellia.example_data.bed", package = "PopPsiSeqR")
lab.bg <- import.smvshift(lab_sechellia.filename)
lab.bg$sechellia <- "lab"


wild_sechellia.filename <- system.file("extdata",
  "lab_sechellia.example_data.bed", package = "PopPsiSeqR")
wild.bg <- import.smvshift(wild_sechellia.filename)
wild.bg$sechellia <- "wild"


windowedFrequencyShift.plotter(c(lab.bg,wild.bg),
  selected_parent = "sim", backcrossed_parent = "sec",
  primary_aesthetic = aes(color=sechellia) ,
  main_title = "PopPsiSeq Results: offspring allele frequency\nrelative to neutral expectation, parental
  ) + facet_grid(sechellia~seqnames
  ) + labs(subtitle = "analyses based on lab-reared and wild-caught sechellia")
```



To quantify the differences between the two cases, there is the **subTractor()** utility which identifies corresponding intervals between the two cases and subtracts one data track from the other.

```
sub.traction <- subTractor(lab.bg, wild.bg ,treament_name = "sechellia")

autoplot(sub.traction, aes(y=lab_minus_wild), geom="line"
  ) + labs(y="Difference in Allele Frequency Shift\n(lab - wild)",
  title = "Difference between PopPsiSeq analyses based on lab-reared and wild-caught sechellia",
  subtitle = "", caption ="",x= "coordinate (droSim1 reference genome)"
  ) + theme_clear()
```



Since the **subTractor()** requires the same intervals to be present in both data sets, it can't be used for data which have been smoothed with inconsistent windowing schemes. For example, the constant-SNP count windows produced by the **bin_by_SNPs.py** utility will depend on the variant dataset used, which may change between compared cases.

One way to circumvent this problem is to calculate the differences on a per-site basis, write these as a BED file, then using the binning script on these sites. The subtraction step involves an inner join between the two frequency shift tables; the sites binned in this case are therefor those sites common to both tables.

```
wild_shifts=data/intermediate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_wildSech.vs_droS
lab_shifts=data/intermediate/freq_shift/freebayes/all2.cleanEarleyAll_with_allSim2_and_labSech.vs_droSim
# these files are generated by the PopPsiSeq workflows documented elsewhere


bedtools intersect -wa -a $wild_shifts  -b <( echo -e "chr2L\t12400000\t12600000" )  \
  > inst/extdata/wild_frequencies.example_data.tbl

bedtools intersect -wa -a $lab_shifts  -b <( echo -e "chr2L\t12400000\t12600000" )  \
  > inst/extdata/lab_frequencies.example_data.tbl
```

```
lab_frequencies.filename <- system.file("extdata",
  "lab_frequencies.example_data.tbl", package = "PopPsiSeqR")
```

```
lab_frequencies.bg <- import.freqtbl(lab_frequencies.filename)
lab_shifts.bg <- freqShifter(lab_frequencies.bg)
lab_shifts.bg$avg_simward_AFshift <- lab_shifts.bg$mean_oriented_shift
lab_shifts.bg$sechellia <- "lab"

wild_frequencies.filename <- system.file("extdata",
  "wild_frequencies.example_data.tbl", package = "PopPsiSeqR")
wild_frequencies.bg <- import.freqtbl(wild_frequencies.filename)
wild_shifts.bg <- freqShifter(wild_frequencies.bg)
wild_shifts.bg$avg_simward_AFshift <- wild_shifts.bg$mean_oriented_shift
wild_shifts.bg$sechellia <- "wild"


fine_subtraction.bg <- subTractor(lab_shifts.bg, wild_shifts.bg,
  treament_name = "sechellia")

fine_subtraction.df <- fine_subtraction.bg %>% as.data.frame() %>%
  mutate(name = n(), score = 0) %>% select(
    c("seqnames","start","end","name","score","strand", "lab_minus_wild") )

write.table(fine_subtraction.df, file=tempfile(), quote=F,
  sep="\t", row.names=F, col.names = F)
```

```
difference_bedfile="sitewise_differences.bed"
binned_difference="binned_differences.bed"
python3 ../exec/python/bin_by_SNPs.py  -i $difference_bedfile \
  -o $binned_difference -b 25 -s 25 -c 7,7 -m count,mean
```

Bibliography

Danecek, Petr, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert E. Handsaker, et al. 2011. "The variant call format and VCFtools." *Bioinformatics* 27 (15): 2156–58. https://doi.org/10.1093/bioinformatics/btr330.

Earley, Eric J., and Corbin D. Jones. 2011. "Next-generation mapping of complex traits with phenotype-based selection and introgression." *Genetics* 189 (4): 1203–9. https://doi.org/10.1534/genetics.111.129445.

Kanippayoor, Rachelle L., Charles Soeder, Tom Hsiang, Corbin D. Jones, and Amanda J. Moehring. 2024. "Identification and genetic analysis of a pervasive 'needle-eye' sperm phenotype in Drosophila sterile hybrid males." *Proceedings of the Royal Society B: Biological Sciences* 291 (2025): 20240483. https://doi.org/10.1098/rspb.2024.0483.

Quinlan, Aaron R., and Ira M. Hall. 2010. "BEDTools: A flexible suite of utilities for comparing genomic features." *Bioinformatics* 26 (6): 841–42. https://doi.org/10.1093/bioinformatics/btq033.