

Setting up

The data are in three files: “SlimmedData.csv”, “LinkageMap.csv”, “DispersalData.csv”. These are in the electronic supplementary material.

These should be put in the same directory, and the directory name set below, under “Import data”. Then, “Evaluate Initialization” to set up all data and definitions.

Choosing the SNP

Here, we use individuals from the Planoles region; data are in the file xx

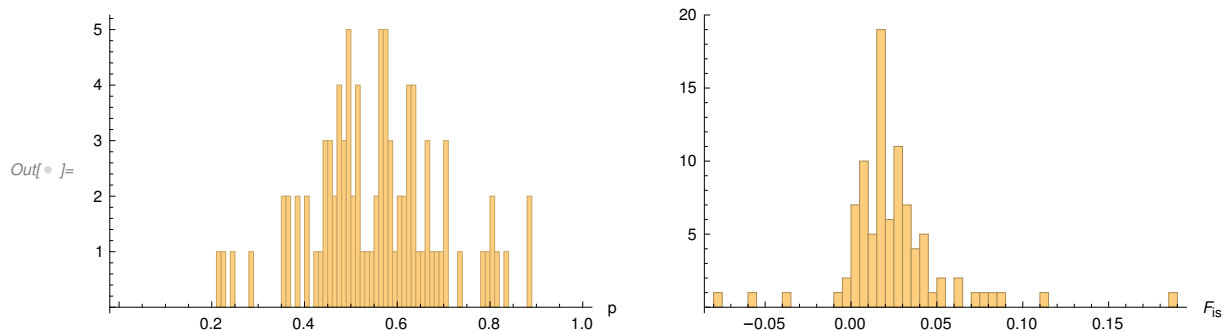
Distribution of p , F_{is}

For the 91 SNP, allele frequencies are between 0.21 and 0.89, and the heterozygote deficit F_{is} is between -0.075 and 0.187

```
In[ ]:= pp = alleleFreq[SNP]; fis = Fis[SNP];  
        {nSNP, MinMax /@ {pp, fis} // N}  
  
Out[ ]:= {91, {{0.213029, 0.889521}, {-0.0750711, 0.1874}}}
```

This is the distribution of p , F_{is} :

```
In[ ]:= GraphicsRow[{Histogram[pp, 50, AxesLabel → {"p", None}, PlotRange → {{0, 1}, {0, 5}}],  
                    Histogram[fis, 50, PlotRange → All, AxesLabel → {"Fis", None}]}]
```



90% of allele frequencies are between 0.353 and 0.803; 88% are between 25% and 75%:

```
In[ ]:= {Quantile[pp, 0.05], Quantile[pp, 0.95],   
        Length[Select[pp, 0.25 < # < 0.75 &]] / nSNP} // N  
  
Out[ ]:= {0.352576, 0.803203, 0.879121}
```

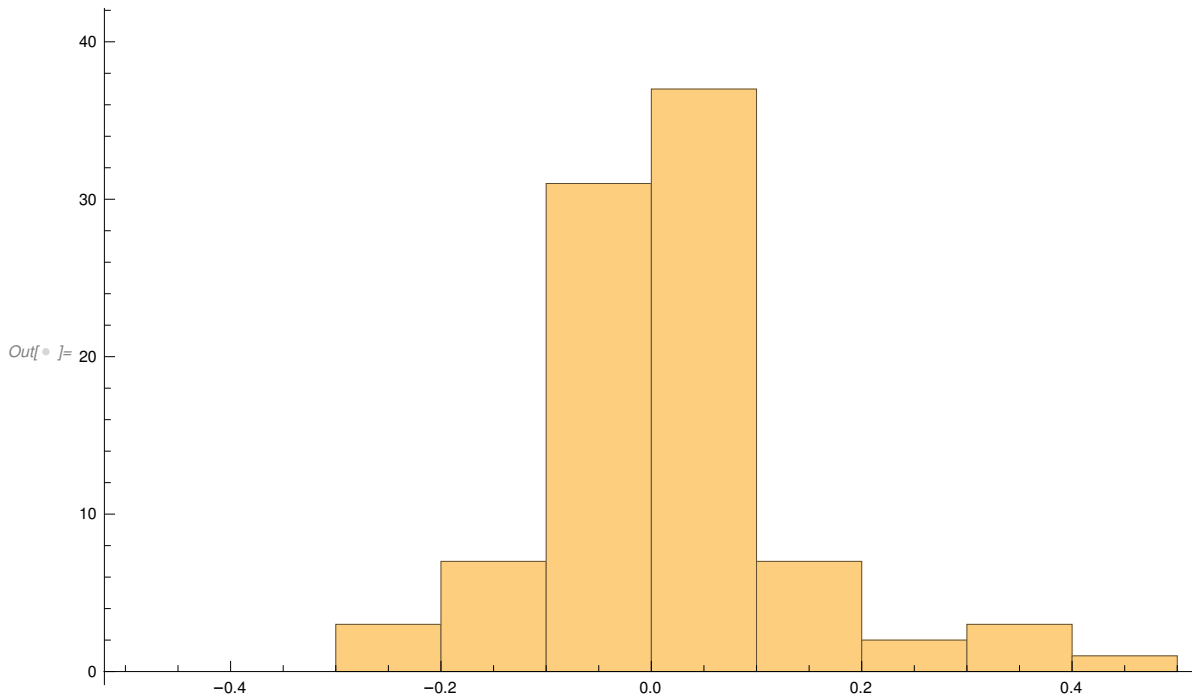
Distribution of clinality

As a rough measure of clinality, we estimate Δp across 5Km, using regX, for all Planoles individuals:

```

In[ ]:= dp = 5000 regX[allInds, #] & /@ allSNP;
Histogram[dp, PlotRange -> {{-0.5, 0.5}, {0, 40}}]

```



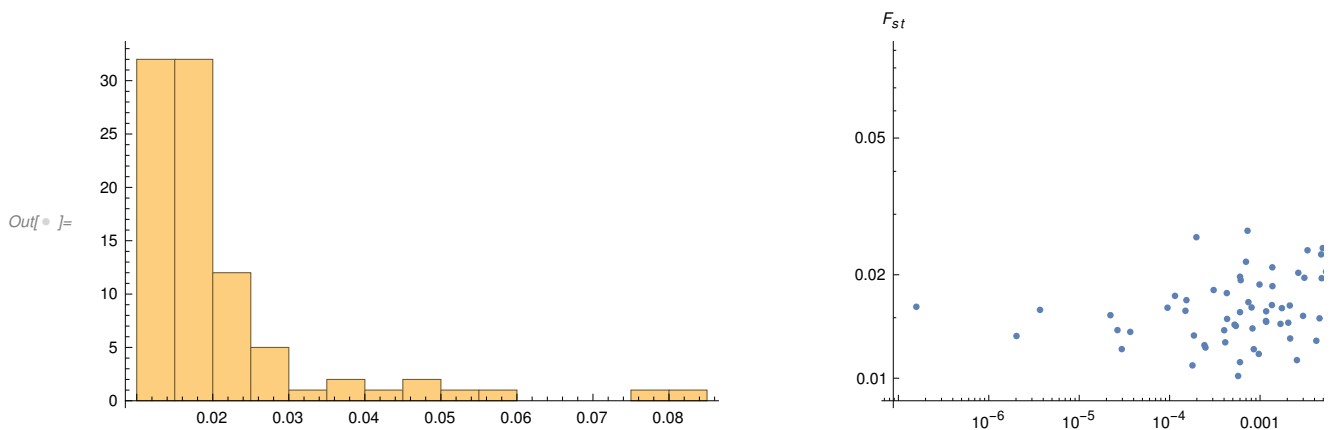
Fst

The left plot shows the distribution of F_{st} across the SNP. The right plot shows the relation between F_{st} and Δp . The high F_{st} SNPs are more clinal (high Δp). Here, F_{st} is calculated by dividing the region into 200m squares; 164 are non-empty.

```

In[ ]:= fst = gridFstTable[PlanolesXY2, 200];
GraphicsRow[{{Histogram[fst, 20, PlotRange -> All],
ListLogLogPlot[Transpose[{dp^2, fst}], AxesLabel -> {"Δp^2", "Fst"}]}]}

```



```
In[ ]:= Mean[fst]
Out[ ]:= 0.0207145
```

Choosing SNP: 170 OK SNP, 110 above 6000, 91 above 13411

We choose SNP according to the following:

- $0.1 < p < 0.9$
- $\Delta p^2 < 0.2$ or slope 0.0894 Km^{-1}
- $F_{st} < 0.1$
- $-0.1 < F_{is} < 0.2$

This shows the {x,y} distance within which 50% are found, the sd of {x,y}, and the min & max of {x,y}:

```
In[ ]:= txy = Transpose[XY];
{nInds, (Quantile[#, 0.75] - Quantile[#, 0.25]) & /@ txy,
StandardDeviation /@ txy, (MinMax /@ txy).{-1, 1}}
Out[ ]:= {22 353, {646.433, 381.553}, {1110.08, 396.326}, {15 788.2, 3251.93}}
```

These are the 5% and 95% quantiles, and the fraction of SNP for which the change in frequency over 1Km is less than 0.029 (corresponding to $\Delta p^2 < 0.2$ for Δp measured over 5Km)

```
In[ ]:= pp = alleleFreq[SNP]; fis = Fis[SNP];
fst = gridFstTable[PlanolesXY2, 200];
dp = 5000 regX[allInds, #] & /@ allSNP;
```

$$\left\{ \text{Quantile}[dp, 0.05], \text{Quantile}[dp, 0.95], \frac{\text{Length}\left[\text{Select}\left[\frac{dp}{5}, -0.029 < \# < 0.029 \&\right]\right]}{n\text{SNP}} \right\} // N$$

```
Out[ ]:= {-0.127669, 0.186246, 0.905882}
```

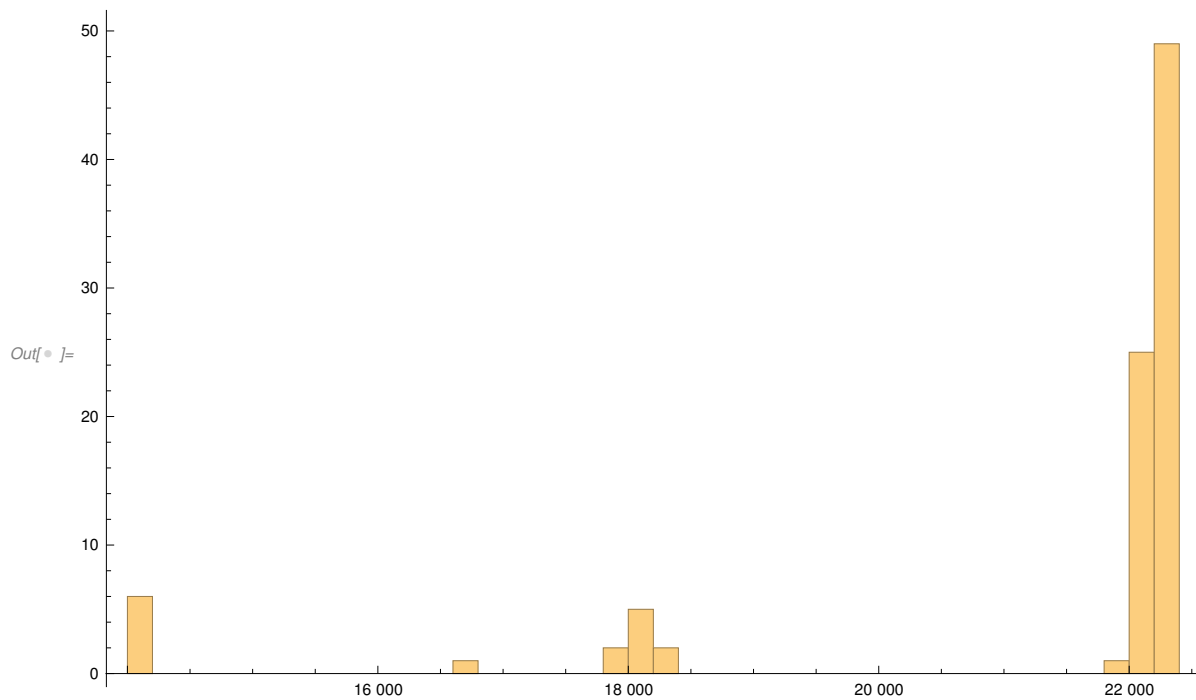
The same for F_{is} :

$$\left\{ \text{Quantile}[fis, 0.05], \text{Quantile}[fis, 0.95], \frac{\text{Length}\left[\text{Select}[fis, -0.04 < \# < 0.1 \&]\right]}{n\text{SNP}} \right\} // N$$

```
Out[ ]:= {-0.00372315, 0.0764982, 0.956044}
```

The threshold 13411 was chosen as 60% of the total sample size. The histogram shows the distribution of # individuals with data for each SNP. Below, the # between 13411 and 20000, and above 20000.

```
In[ ]:= ndat = nIndperSNP[allInds]; Histogram[ndat, 50, PlotRange -> All]
Length[Select[ndat, #]] & /@ {13 411 ≤ # < 20 000 &, 20 000 < # &, True &}
```



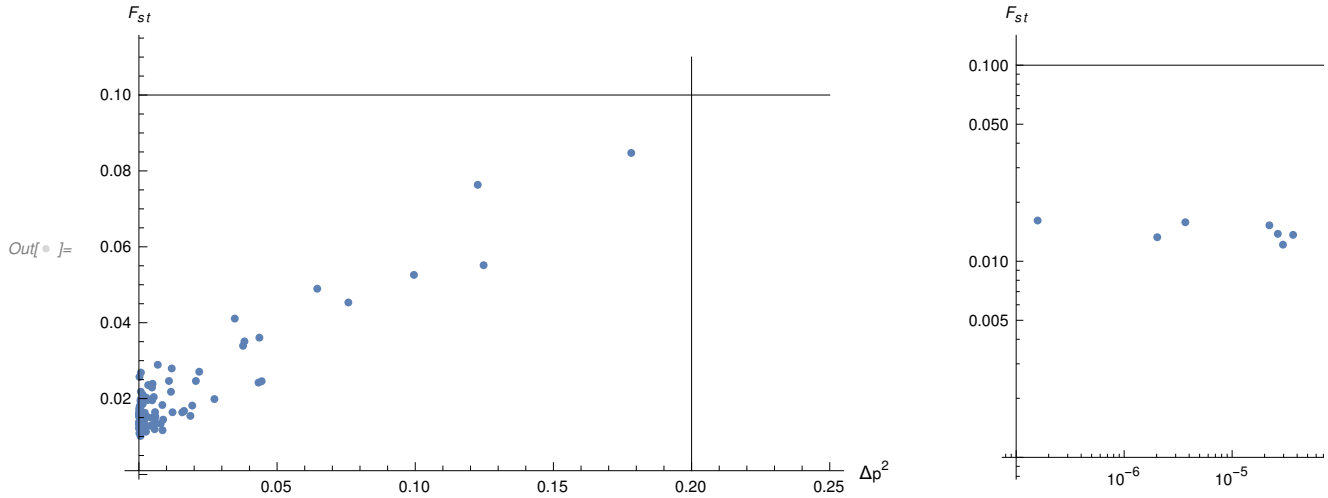
```
Out[ ]:= {16, 75, 91}
```

This shows the thresholds $F_{st} < 0.1$ and $\Delta p^2 < 0.2$ on a linear and a log-log scale:

```

In[ ]:= gr[f_, g_] :=
  Show[f[Transpose[{dp2, fst}], AxesOrigin → {10-7, 0.001}, AxesLabel → {"Δp2", "Fst"},
    Graphics[Line[{10-7, 0.1}, {0.25, 0.1}] // g],
    Graphics[Line[{0.2, 0.001}, {0.2, 0.11}] // g], PlotRange → All];
GraphicsRow[{gr[ListPlot, Identity], gr[ListLogLogPlot, Log]}]

```



Distribution of H

Histograms of H; significant LD (var[H]=0.00336 vs 0.00281 at LE)

This generates 100 shuffled replicates, in LE:

```

In[ ]:= H = MLH[SNP] // N;
In[ ]:= sH = Table[MLH[shuffle[SNP]] // N, {k, 100}];
Out[ ]:= $Aborted

In[ ]:= sH1 = MLH[shuffle[SNP]] // N;

```

The mean of the field data necessarily matches that of the shuffled data (top). The variance of the field data (0.003361) is 22 s.d. higher than the mean of the shuffled data (0.002813):

```

In[ ]:= {{Mean[H], Mean[Mean /@ sH], StandardDeviation[Mean /@ sH]},
  {Variance[H], Mean[Variance /@ sH], StandardDeviation[Variance /@ sH]}} // TableForm

```

Out[]:= //TableForm=

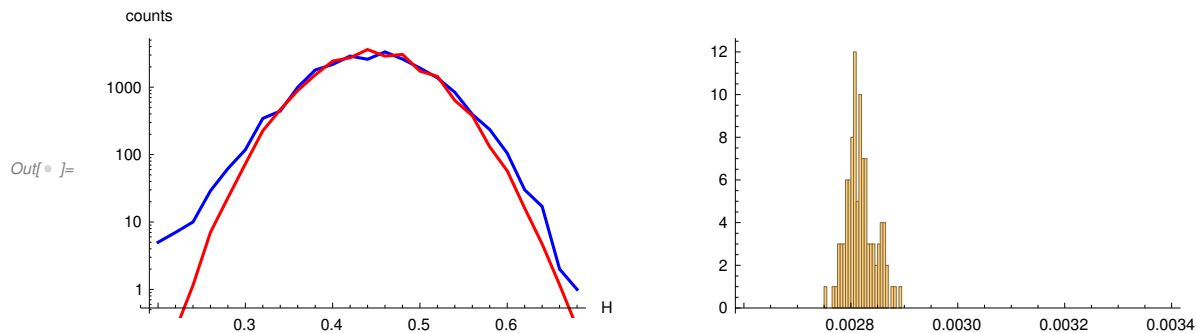
0.446015	0.446036	8.37825×10^{-6}
0.00336137	0.00281821	0.0000269246

Left: the distribution of H (blue) is more variable than the mean of 100 shuffled replicates (red). Right: the observed variance (0.00336) is far above the distribution of variances in shuffled data:

```

In[ ]:=  $\delta = 0.02$ ;
rn = Range[ $0.15 + \delta / 2$ ,  $0.7 - \delta / 2$ ,  $\delta$ ];
GraphicsRow[{Show[ListLogPlot[Transpose[{rn, BinCounts[H, {0.15, 0.7,  $\delta$ ]}],
  Joined → True, PlotStyle → Blue, AxesLabel → {"H", "counts"}],
  ListLogPlot[Transpose[{rn, Mean[BinCounts[#, {0.15, 0.7,  $\delta$ ]} & /@ sH]}],
  Joined → True, PlotStyle → Red]],
Histogram[Variance /@ sH, 20, PlotRange → {{0.0026, 0.0034}, {0, 12}}]}]

```

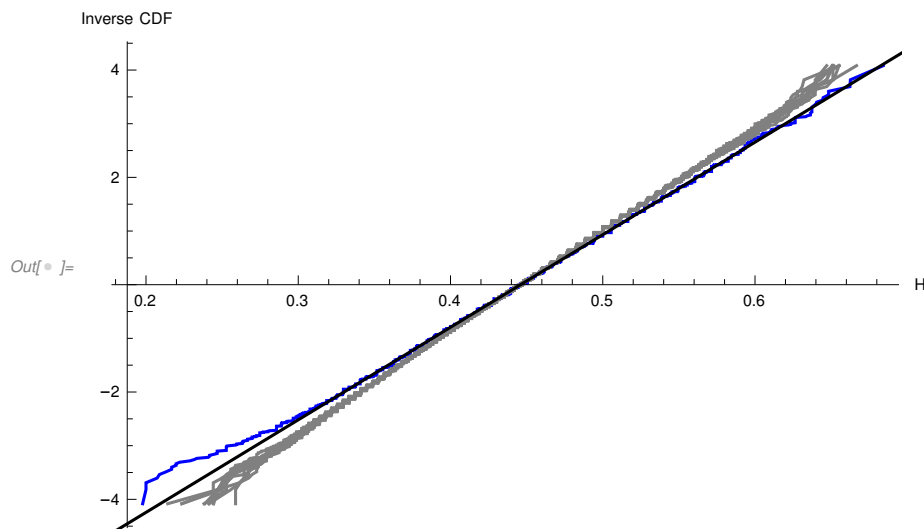


This plots the distribution of H on a probit scale (blue), comparing with a Gaussian (black). There is evidence of the presence of highly inbred individuals (blue, bottom left). The gray plots show 10 shuffled replicates, which have a lower variance, and hence a steeper slope.

```

In[ ]:= mnH = Mean[H]; sdH = StandardDeviation[H];
In[ ]:= mnH = Mean[H]; sdH = StandardDeviation[H];
gr[h_List, c_] := ListLinePlot[Transpose[{Sort[h], sdl[h]}], PlotStyle → c];
Show[gr[H, Blue], gr[#, Gray] & /@ sH[[1 ;; 10]],
Plot[ $\frac{x - mnH}{sdH}$ , {x, 0.1, 0.9}, PlotStyle → Black], AxesLabel → {"H", "Inverse CDF"}]

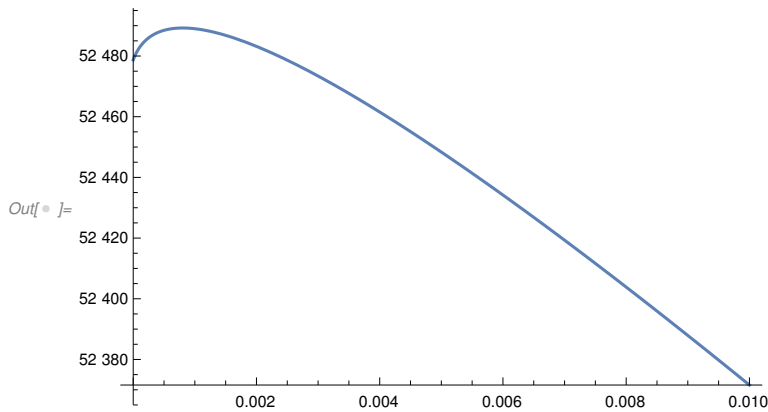
```



Weak indication of offspring from selfing: $\phi=0.00086$, $\Delta L=11.3$

If a fraction ϕ of individuals are produced by selfing, we expect a mix of two Gaussians, with the individuals from selfing having half the mean and variance of H. This plots $\log(L)$ against selfing rate, ϕ , fixing the mean and variance to those observed. Below, the best fits with $\phi=0$, and free; there is significant evidence for selfing $\phi=0.00086$, $\Delta L=11.3$

```
lnf := H = MLH[SNP] // N;
mn = Mean[H]; vv = Variance[H];
LG = Total[LikG[phi, m, v] / @ H];
Plot[LG /. {m -> mn, v -> vv}, {phi, 0, 0.01}]
fmHi0 = FindMaximum[LG /. phi -> 0, {m, mn, 0.35, 0.5}, {v, vv, 0.001, 0.01}]
fmHi = FindMaximum[LG, {phi, 0.001, 0, 0.01}, {m, mn, 0.4, 0.45}, {v, vv, 0.003, 0.005}]
```



```
Out[ ] = {52478.7, {m -> 0.446015, v -> 0.00336122}}
```

```
Out[ ] = {52490., {phi -> 0.000860938, m -> 0.446207, v -> 0.00332422}}
```

This tabulates the best fits for different values of ϕ - the result is the same as the slightly cruder method used above:

```

In[ ]:= mn = Mean[H]; vv = Variance[H];
tt = Table[{ $\phi$ x, FindMaximum[LG /.  $\phi \rightarrow \phi$ x, {m, mn, 0.4, 0.45}, {v, vv, 0.003, 0.005}]}],
{ $\phi$ x, 0, 0.003, 0.0002}];
TableForm[tt, TableDepth  $\rightarrow$  2]

```

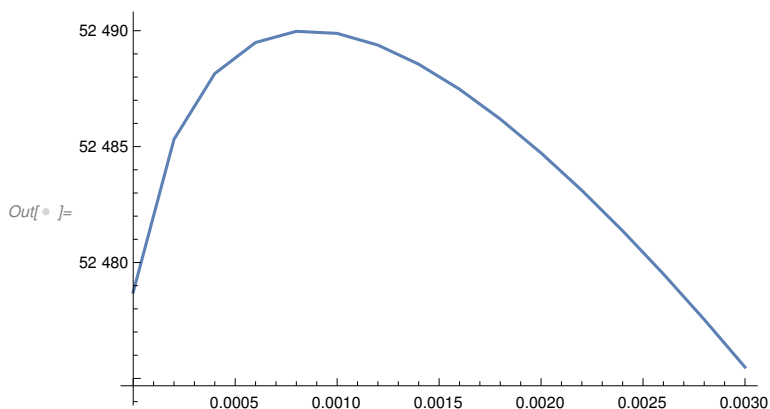
Out[]:= TableForm=

0.	{52 478.7, {m \rightarrow 0.446015, v \rightarrow 0.00336122}}
0.0002	{52 485.3, {m \rightarrow 0.4461, v \rightarrow 0.00334286}}
0.0004	{52 488.2, {m \rightarrow 0.446143, v \rightarrow 0.00333481}}
0.0006	{52 489.5, {m \rightarrow 0.446174, v \rightarrow 0.00332944}}
0.0008	{52 490., {m \rightarrow 0.4462, v \rightarrow 0.00332531}}
0.001	{52 489.9, {m \rightarrow 0.446222, v \rightarrow 0.00332192}}
0.0012	{52 489.4, {m \rightarrow 0.446242, v \rightarrow 0.00331902}}
0.0014	{52 488.6, {m \rightarrow 0.44626, v \rightarrow 0.00331646}}
0.0016	{52 487.5, {m \rightarrow 0.446277, v \rightarrow 0.00331417}}
0.0018	{52 486.2, {m \rightarrow 0.446293, v \rightarrow 0.0033121}}
0.002	{52 484.7, {m \rightarrow 0.446308, v \rightarrow 0.00331019}}
0.0022	{52 483.1, {m \rightarrow 0.446322, v \rightarrow 0.00330843}}
0.0024	{52 481.4, {m \rightarrow 0.446335, v \rightarrow 0.00330679}}
0.0026	{52 479.5, {m \rightarrow 0.446348, v \rightarrow 0.00330525}}
0.0028	{52 477.5, {m \rightarrow 0.44636, v \rightarrow 0.0033038}}
0.003	{52 475.5, {m \rightarrow 0.446371, v \rightarrow 0.00330244}}

```

In[ ]:= tt2 = Transpose[{tt[[All, 1]], tt[[All, 2, 1]]}];
ListLinePlot[tt2]
int = Interpolation[tt2];
fm = FindMaximum[int[x], {x, 0.001, 0, 0.003}];
{x /. fm[[2]], fm[[1]] - int[0]}

```



Out[]:= {0.000862103, 11.2739}

Weak correlations in heterozygosity within LG

SNP are ordered on the map:


```
In[ ] := SNPposn
```

```
Out[ ] := {{1, 4.373}, {1, 8.3265}, {1, 15.42}, {1, 16.487}, {1, 22.6005}, {1, 23.9}, {1, 24.267},
{1, 25.9985}, {1, 34.429}, {1, 40.5502}, {1, 66.75}, {1, 88.0873}, {1, 99.077}, {2, 26.239},
{2, 38.1035}, {2, 41.988}, {2, 42.2825}, {2, 43.18}, {2, 43.186}, {2, 43.188}, {2, 43.189},
{2, 43.191}, {2, 43.193}, {2, 44.312}, {2, 44.907}, {2, 51.1792}, {2, 57.6436}, {2, 69.2646},
{3, 8.997}, {3, 16.2318}, {3, 20.259}, {3, 20.8}, {3, 21.479}, {3, 21.9}, {3, 32.6705},
{3, 32.68}, {3, 37.0261}, {3, 70.1381}, {4, 0.758}, {4, 4.31333}, {4, 6.1336}, {4, 9.364},
{4, 9.372}, {4, 10.475}, {4, 15.43}, {4, 25.5292}, {4, 29.8865}, {4, 33.2662}, {4, 57.7255},
{4, 65.8237}, {5, 32.3185}, {5, 42.9968}, {5, 47.16}, {5, 49.964}, {5, 49.979}, {5, 51.075},
{5, 51.076}, {5, 51.078}, {5, 51.079}, {5, 51.081}, {5, 55.6}, {5, 63.0185}, {6, 2.174},
{6, 2.176}, {6, 4.348}, {6, 5.436}, {6, 20.709}, {6, 34.1798}, {6, 39.564}, {6, 41.16},
{6, 49.0559}, {6, 49.4004}, {6, 49.961}, {6, 50.0082}, {6, 50.3107}, {6, 53.3989},
{6, 63.1877}, {7, 1.064}, {7, 4.835}, {7, 5.367}, {7, 5.37}, {7, 17.8435}, {7, 30.4297},
{8, 2.3}, {8, 2.31}, {8, 3.49}, {8, 3.49}, {8, 7.67329}, {8, 14.947}, {8, 27.648}, {10, 3}}
```

There are 155/22353 individuals with $H < 0.3$ (2.5 sd below the mean of 0.446), 22 with $H < 0.25$ (3.38 sd below the mean of 0.446) and 10 with $H < 0.22$ (3.9 sd below the mean of 0.446)

```
In[ ] := H = MLH[SNP] // N;
```

```
lowH = Select[allInds, H[[#]] < 0.3 &];
```

```
quiteLowH = Select[allInds, H[[#]] < 0.25 &];
```

```
veryLowH = Select[allInds, H[[#]] < 0.22 &];
```

```
{Length[veryLowH], Length[quiteLowH], Length[lowH], nInds}
```

```
Out[ ] := {10, 22, 155, 22353}
```

Looking at the 10 individuals with lowest H, one can imagine seeing blocks of heterozygosity - but this is not easy to test. One could look for junctions between homozygous & heterozygous regions, within LG - though, only around half of homozygous loci are in fact IBD. This shows the ten least heterozygous individuals (heterozygous sites marked as 1)

```
In[ ] := SNP[veryLowH] /. {2 -> 0, -10 -> 0} // TableForm
```

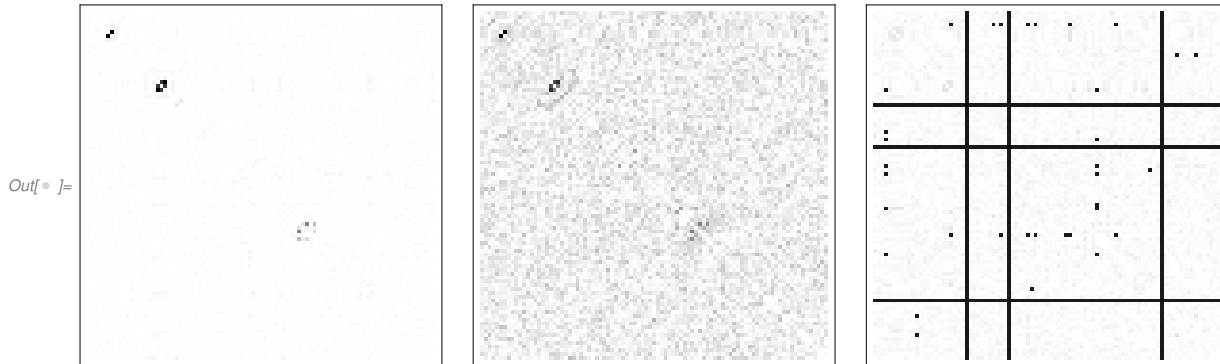
```
Out[ ] // TableForm =
```

1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	1	-9	0	0	0	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0

A more systematic approach is to look for correlations in homozygosity between loci. We can do this for the full dataset, and also for those with very low H, on the grounds that those would be from recent inbreeding.

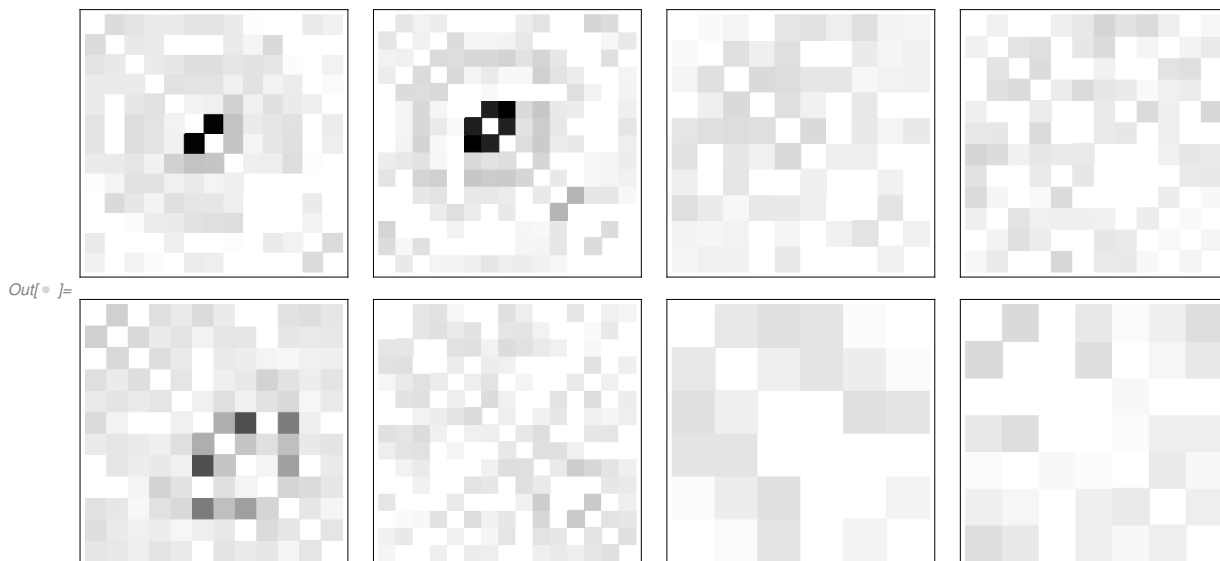
This shows the correlation matrix (diagonal deleted) amongst all 91 SNP, for the full data, for the 155 with $H < 0.3$, and the 22 with $H < 0.25$ (the latter having some SNP with missing data, black). There are two or three clusters of linked loci with strong correlations, but otherwise weak signals. This takes a few minutes:

```
In[ ]:= cm = corrH[allInds, allSNP]; cmLow = corrH[lowH, allSNP];
cmQuiteLow = corrH[quiteLowH, allSNP];
GraphicsRow[ArrayPlot[## - IdentityMatrix[91]] & /@ {cm, cmLow, cmQuiteLow}]
```



This shows the correlations for each of the 8 LG, on a $\sqrt{}$ colour scale. strong correlations are seen for LG 1, 2, 5

```
In[ ]:= cplot[k_] := ArrayPlot[corrH[allInds, SNPLG[k]] - IdentityMatrix[Length[SNPLG[k]]],
ColorFunctionScaling -> False, ColorFunction -> (GrayLevel[1 - Sqrt[Max[0, ##]]] &)];
GraphicsGrid[Partition[cplot /@ Range[8], 4]]
```



There is a weak positive correlation in heterozygosity amongst SNP within LG, averaging 0.011; this is stronger (0.035) for the 155 individuals with low $H < 0.3$. (values for the 22 individuals with $H < 0.25$ are

unreliable). The average correlation amongst all SNP (which are mostly unlinked) is much weaker (last row).

```
In[ ] := mnn[x_List] :=
Module[{n = Length[x]},  $\frac{n}{n-1}$  Mean[DeleteCases[Flatten[x - IdentityMatrix[Length[x]]], -9]]];
tt = Table[ss = SNPLG[k];
{k, Length[ss], mnn[corrH[allInds, ss]],
mnn[corrH[lowH, ss]], mnn[corrH[quiteLowH, ss]]}, {k, 8}];
Prepend[Join[tt, {"Mean", Total[tt[[All, 2]], Mean[tt[[All, 3]], Mean[tt[[All, 4]],
Mean[tt[[All, 5]]}], {"Net", nSNP, mnn[cm], mnn[cmLow], mnn[cmQuiteLow]}],
{"LG", "# SNP", "all", "H<0.3", "H<0.25"}] // TableForm
```

Out[] // TableForm=

LG	# SNP	all	H<0.3	H<0.25
1	13	0.0182819	0.0442279	0.074381
2	15	0.0292915	0.050206	0.00922821
3	10	0.00459604	0.0329792	-0.0517051
4	12	0.00291755	-0.00369953	0.0208617
5	12	0.0240654	0.0785377	0.130077
6	15	0.00402098	0.0312777	0.0169967
7	6	0.00388023	0.0386177	0.118984
8	7	0.00304428	0.0108881	-0.0749898
Mean	90	0.0112622	0.0353793	0.0304792
Net	91	0.00239714	-0.00809899	-0.0137812

This shows the correlations between adjacent SNP, within each LG. These are stronger for the whole dataset, but not much more for the low H individuals - though that may just be noise.

```
In[ ] := mnn2[x_List] := Mean[DeleteCases[Table[x[[j, j + 1]], {j, 1, Length[x] - 1}], -9 | -10]];
tt = Table[ss = SNPLG[k];
{k, Length[ss], mnn2[corrH[allInds, ss]],
mnn2[corrH[lowH, ss]], mnn2[corrH[quiteLowH, ss]]}, {k, 8}];
Prepend[Join[tt, {"Mean", Total[tt[[All, 2]], Mean[tt[[All, 3]], Mean[tt[[All, 4]],
Mean[tt[[All, 5]]}], {"LG", "# SNP", "all", "H<0.3", "H<0.25"}] // TableForm
```

Out[] // TableForm=

LG	# SNP	all	H<0.3	H<0.25
1	13	0.0899783	0.152085	0.309806
2	15	0.121388	0.137233	0.301148
3	10	0.00958531	0.0635943	0.11112
4	12	0.00293526	-0.0052951	0.0350859
5	12	0.0265399	0.103651	0.144844
6	15	0.00981901	0.0674313	0.110537
7	6	-0.00158475	0.0341626	0.197111
8	7	0.0033323	0.00804741	-0.118507
Mean	90	0.0327492	0.0701136	0.136393

Table xx. Correlations in heterozygosity within the 8 linkage groups. The third and fourth columns give the mean correlation in h between loci within each linkage group, for all 22,354 individuals, and

for the 155 individuals with $H < 0.3$. The last two columns give the mean correlations between adjacent SNP. The last row gives the net mean correlations amongst all pairs of SNP, which are mostly unlinked. Note that 1 of the 91 SNP has not been assigned to a linkage group.

LG	# SNP	within LG		adjacent SNP	
		all inds	$H < 0.3$	all inds	$H < 0.3$
1	13	0.01828	0.04423	0.08998	0.15209
2	15	0.02929	0.05021	0.12139	0.13723
3	10	0.00459	0.03298	0.00958	0.06359
4	12	0.00292	-0.00370	0.00294	-0.00530
5	12	0.02407	0.07854	0.02654	0.10365
6	15	0.00402	0.03128	0.00982	0.06743
7	6	0.00388	0.03862	-0.00159	0.03416
8	7	0.00304	0.01089	0.00334	0.00805
Mean	90	0.01126	0.03538	0.03275	0.07011
Net	91	0.00240	-0.00810		

We can ask, how much of this excess variance is due to covariance within vs between linkage groups? The total excess variance is the difference between the observed variance, and that in shuffled replicates (sH, calculated above) which are in LE:

Variance[H] – Mean[Variance /@ sH]

```
Out[ ]:= 0.000546055
```

This calculates the matrices of covariances of H for all individuals, for those with $H < 0.3$, and those with $H < 0.2$. The sum of these covariance gives the total variance in H, allowing us to separate the contributions from ID within & between LG.

```
In[ ]:= cmcv = covH[allInds, allSNP];
cmcvLow = covH[lowH, allSNP];
cmcvQuiteLow = covH[quiteLowH, allSNP];
```

The first two values are the variance of H, and the mean of the whole covariance matrix. The last two values are the means of the diagonal and off-diagonal components of the covariance matrix. The mean of the covariance matrix is slightly smaller than the variance of H - likely due to missing data.

```
In[ ]:= mnn[x_List] := Module[{n = Length[x]},
  
$$\frac{n}{n-1} \text{Mean[DeleteCases[Flatten[x - DiagonalMatrix[Diagonal[x]]], -9]]};$$

  {Variance[H], Mean[Flatten[cmcv]], Mean[Diagonal[cmcv]], mnn[cmcv]}
```

```
Out[ ]:= {0.00336137, 0.00322345, 0.242695, 0.00056266}
```

The mean covariance within LG is 0.002650, and overall, is 0.0005626. Nevertheless, the latter makes a larger contribution.

```

In[ ]:= ttcv = Table[ss = SNPLG[k];
  {k, Length[ss], mnn[covH[allInds, ss]],
    mnn[covH[lowH, ss]], mnn[covH[quiteLowH, ss]]}, {k, 8}];
Prepend[Join[ttcv, {"Mean", Total[ttcv[All, 2]], Mean[ttcv[All, 3]], Mean[ttcv[All, 4]],
  Mean[ttcv[All, 5]]}, {"Net", nSNP, mnn[cmcv], mnn[cmcvLow], mnn[cmcvQuiteLow]}],
  {"LG", "# SNP", "all", "H<0.3", "H<0.25"}] // TableForm

```

Out[]:= `TableForm=`

LG	# SNP	all	H<0.3	H<0.25
1	13	0.00448951	0.00842316	0.0114368
2	15	0.00633856	0.00688672	0.00998969
3	10	0.00113647	0.00641	0.0130351
4	12	0.000728322	-0.00088148	0.004014
5	12	0.00596478	0.0152473	0.0239653
6	15	0.000982156	0.00632464	0.00913581
7	6	0.000880449	0.00685518	0.0227213
8	7	0.000680284	0.00264504	-0.011335
Mean	90	0.00265007	0.00648883	0.0103704
Net	91	0.00056266	-0.00167127	-0.00171442

We can work out the relative contribution by calculating the overall mean off-diagonal, 0.00056262, and comparing it with the mean off-diagonal of each LG (averaging 0.00265). We have that $\bar{c} n_T (n_T - 1) = \sum_{i \neq j} n_i n_j \bar{c}_{i,j} + \sum_i n_i (n_i - 1) \bar{c}_i$. 73% of the covariance is within LG, and 27% between LG. For the highly inbred individuals, the covariance overall is actually negative (by chance) but that within LG is higher.

```

In[ ]:= nn = {13, 15, 10, 12, 12, 15, 6, 7}; nt = Total[nn]; cc = ttcv[All, 3];
{0.00056262 * 90 * 89, (nn (nn - 1)).cc, 0.00056262 * 90 * 89 - (nn (nn - 1)).cc}

```

Out[]:= {4.50659, 3.27847, 1.22812}

This is an algebraic check that these components sum up correctly:

```

In[ ]:= Simplify[(Sum[n_i, {i, 1, 8}])((Sum[n_i, {i, 1, 8}]) - 1) - Sum[n_i, {i, 1, 8}] Sum[n_j, {j, 1, 8}] If[i == j, 0, n_i n_j] - Sum[n_i (n_i - 1), {i, 1, 8}]]

```

Out[]:= 0

Generating random offspring: linkage hardly changes the variance of H

This sets up offspring with given recombination rates

```

In[ ]:= rr = recInt[SNPposn];
ru = ConstantArray[1/2, nSNP - 1];

```

```

In[ ]:= x1 = {0, 1, 2, 5, 10, 20, 50, 100, 200, 400, 600, 800, 1000};
ht = {kidsH[{allInds, allSNP}, #, rr, 2], kidsH[{allInds, allSNP}, #, 2], H} & /@ x1;
Map[Mean, ht, {2}] // N // TableForm
Map[Variance, ht, {2}] // N // TableForm

```

Out[]//TableForm=

0.431697	0.431539	0.446015
0.433003	0.432877	0.446015
0.433573	0.434407	0.446015
0.440694	0.440343	0.446015
0.444326	0.444636	0.446015
0.447486	0.447183	0.446015
0.450674	0.449839	0.446015
0.451699	0.45119	0.446015
0.452289	0.45247	0.446015
0.453791	0.454951	0.446015
0.454252	0.455098	0.446015
0.456467	0.456965	0.446015
0.456691	0.456216	0.446015

Out[]//TableForm=

0.00388362	0.00388872	0.00336137
0.00389496	0.00388591	0.00336137
0.00382801	0.00381898	0.00336137
0.00353546	0.00353587	0.00336137
0.00335693	0.00331282	0.00336137
0.0032458	0.00323266	0.00336137
0.00307684	0.00312778	0.00336137
0.00308426	0.0031215	0.00336137
0.00310857	0.00303359	0.00336137
0.00306126	0.00300578	0.00336137
0.00307396	0.0030222	0.00336137
0.00308558	0.00309129	0.00336137
0.00302077	0.00306266	0.00336137

These are the mean and variance (left, right) as a function of radius. Mean H increases slightly with distance (IBD) and is unaffected by linkage.

radius	linked	unlinked	actual	F_{st} u/l
0	0.431522	0.43177	0.44602	0.0314105
1	0.432633	0.433086		0.0293379
2	0.434104	0.434002		0.0258778
5	0.440956	0.440988		0.0120753
10	0.444453	0.444649		0.0021514
20	0.447483	0.447164		-0.00374261
50	0.450371	0.449992		-0.00914432
100	0.45195	0.452333		-0.011945
200	0.452673	0.45275		-0.0153289
400	0.454318	0.454549		-0.0187562
600	0.455186	0.455702		-0.0209486
800	0.456734	0.457002		
1000	0.456802	0.455772		

radius	linked	unlinked	actual
0	0.00392966	0.00395374	0.003361
1	0.003886	0.00384408	
2	0.00379937	0.00379272	
5	0.00357459	0.00350251	
10	0.00331007	0.0033201	
20	0.00323728	0.00324176	
50	0.00313948	0.00311083	
100	0.0031121	0.00301404	
200	0.00306949	0.00308218	
400	0.00301763	0.00310179	
600	0.00305304	0.00306555	
800	0.00306159	0.00309686	
1000	0.00306338	0.0030457	

Calculation of F_{st} is problematic here, because this is a sample from a limited area: necessarily, F_{st} goes negative if one takes F_t as the overall mean. Comparing with a 1Km baseline gives higher values of 0.053 at 0m and 0.019 at 20m, 0.0066 at 200m

$$In[] := \left\{ 1 - \frac{0.4315}{0.4558}, 1 - \frac{0.4472}{0.4558}, 1 - \frac{0.4528}{0.4558} \right\}$$

Out[] := {0.0533129, 0.0188679, 0.00658183}

Left: mean H, right, var(H); blue: linked, black: unlinked. IBD and the decline in variance occur over very small scales, ~10m

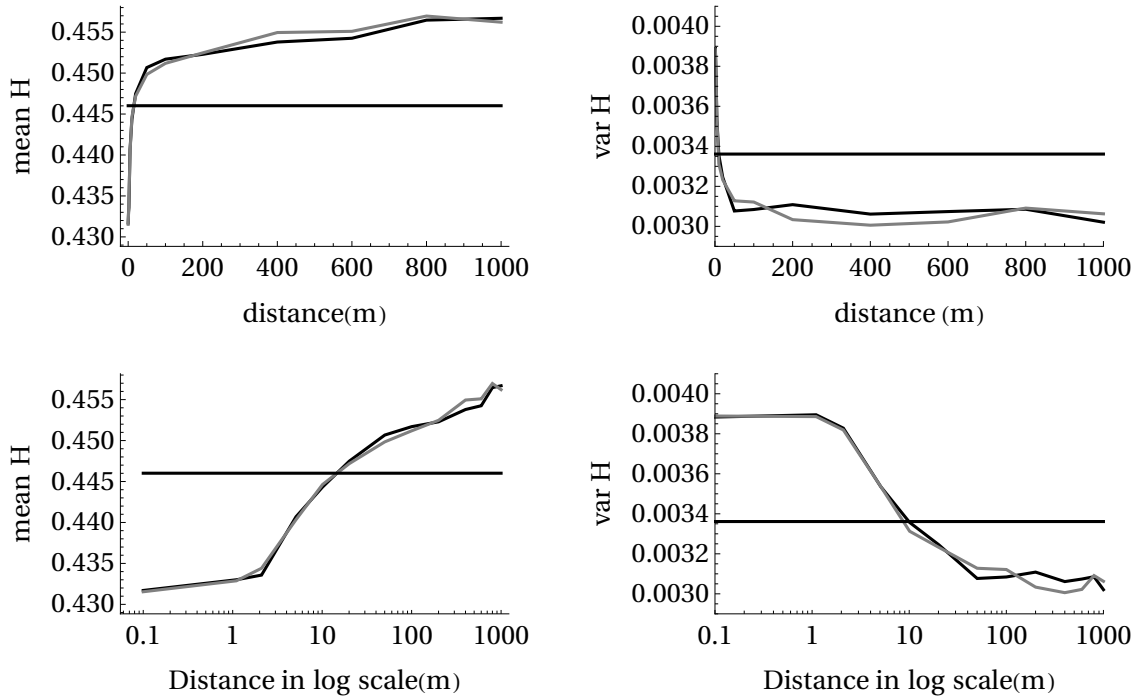
Figure xx. Mean and variance of H (left, right), as a function of the distance between parents. Offspring are generated with no linkage (grey) or with linkage (black); observed values are shown in by

the horizontal line. The bottom row shows distance in a log scale.

```

In[ ]:= bs = {FontFamily → "Times", FontSize → 14}; cols = {Black, Gray};
tkSM = {{0, 500, 1000}, {0.43, 0.44, 0.45}};
tkSV = {{0, 500, 1000}, {0.0030, 0.0035, 0.004}};
tkLM = {{0.1, 10, 1000}, {0.43, 0.44, 0.45}};
tkLV = {{0.1, 10, 1000}, {0.0030, 0.0035, 0.004}};
grm[f_] :=
  Show[MapThread[ListLinePlot[Transpose[{xl, Mean /@ N[f[All, #]]}], PlotStyle → #2] &,
    {{1, 2}, cols}], Plot[Mean[N[f[1, 3]]], {x, 0, 1000}, PlotStyle → Black],
    (*FrameTicks→{{0,500,1000},{0.43,0.44,0.45}},*)Frame → {{True, False}, {True, False}},
    FrameLabel → {"distance(m)", "mean H"}, BaseStyle → bs];
grv[f_] := Show[MapThread[ListLinePlot[Transpose[{xl, Variance /@ N[f[All, #]]}],
  PlotStyle → #2, PlotRange → {{0.1, 1000}, {0.0029, 0.0041}}] &, {{1, 2}, cols}],
  Plot[Variance[N[f[1, 3]]], {x, 0, 1000}, PlotStyle → Black],
  Frame → {{True, False}, {True, False}},
  FrameLabel → {"distance (m)", "var H"}, BaseStyle → bs];
grml[f_] := Show[MapThread[ListLogLinearPlot[Transpose[{xl + 0.1, Mean /@ N[f[All, #]]}],
  PlotStyle → #2, Joined → True] &, {{1, 2}, cols}],
  LogLinearPlot[Mean[N[f[1, 3]]], {x, 0.1, 1000}, PlotStyle → Black],
  (*FrameTicks→tkLM,*)Frame → {{True, False}, {True, False}},
  FrameLabel → {"Distance in log scale(m)", "mean H"}, BaseStyle → bs];
grvl[f_] := Show[MapThread[ListLogLinearPlot[Transpose[{xl + 0.1, Variance /@ N[f[All, #]]}],
  PlotStyle → #2, Joined → True, PlotRange → {{0.1, 1000}, {0.0029, 0.0041}}] &,
  {{1, 2}, cols}], LogLinearPlot[Variance[N[f[1, 3]]], {x, 0.1, 1000},
  PlotStyle → Black], Frame → {{True, False}, {True, False}},
  FrameLabel → {"Distance in log scale(m)", "var H"},
  BaseStyle → bs(*, FrameTicks→tkLV*)];
GraphicsGrid[{{grm[ht], grv[ht]}, {grml[ht], grvl[ht]}}]

```

Drawing random H for different distributions of distance between parents

realhet, gausshet and nearhet calculates the MLH values for offspring generated by matings based on realistic dispersal kernel from the data, a radial gaussian dispersal kernel with $\sigma=300$ and nearest neighbour mating. We check the mean and variance of H and compared to the field data. The mean, variance and shape of the distribution of H are compared by t test, f test and Kolmogorov- Smirnov Test respectively. The probit transform of the four distributions are shown below. The field data is seen to be more consistent with realistic mating simulation .

```
ln[ ]:= snpraw = (# /. {0 -> {0, 0}, 1 -> {0, 1}, 2 -> {1, 1}, -9 -> {-9, -9}, -10 -> {-10, -10}}) & /@ SNP;
pollendisp = disp[All, 2];
distreal = RandomVariate[HistogramDistribution[pollendisp], nInds];

distgauss = RandomVariate[ProbabilityDistribution[r Exp[-r^2 / (2 * 300^2)], {r, 0, 3000}], nInds];
```

This generates offspring for the three dispersal schemes, and finds their heterozygosity. The means and variances of H for the real data & the three simulated schemes are given in the table:

```
In[ ]:= realoff = Offspring[distreal, snpraw]; realhet = MLH[realoff] // N;
      gaussoff = Offspring[distgauss, snpraw]; gausshet = MLH[gaussoff] // N;
      nearoff = Offspring[near[snpraw]; nearhet = MLH[nearoff] // N;
      {Mean[##], Variance[##]} & /@ {H, realhet, gausshet, nearhet} // TableForm
```

Out[]//TableForm=

Mean[H]	Variance[H]
0.445426	0.0033302
0.432298	0.00390552
0.431671	0.00389677

This makes statistical comparisons (t-test, F-test, and Kolmogorov-Smirnov test) between all 6 pairs of data. Each entry gives {test statistic, p-value}:

```
In[ ]:= dataNames = {"field", "realistic", "Gaussian", "Nearest"};
      dataNamePairs = Subsets[dataNames, {2}];
      pairs = Subsets[{H, realhet, gausshet, nearhet}, {2}];
      TableForm[
        Prepend[MapThread[{##1, TTest[##2, 0, "TestData"], VarianceTest[##2, 1, "TestData"],
          KolmogorovSmirnovTest[##2[[1]], ##2[[2]], "TestData"]} &,
          {dataNamePairs, pairs}] // Chop,
        {"datasets", "t-test\non means", "F-test\non variances", "KS-test\non CDF"}],
        TableDepth → 2]
```

Out[]//TableForm=

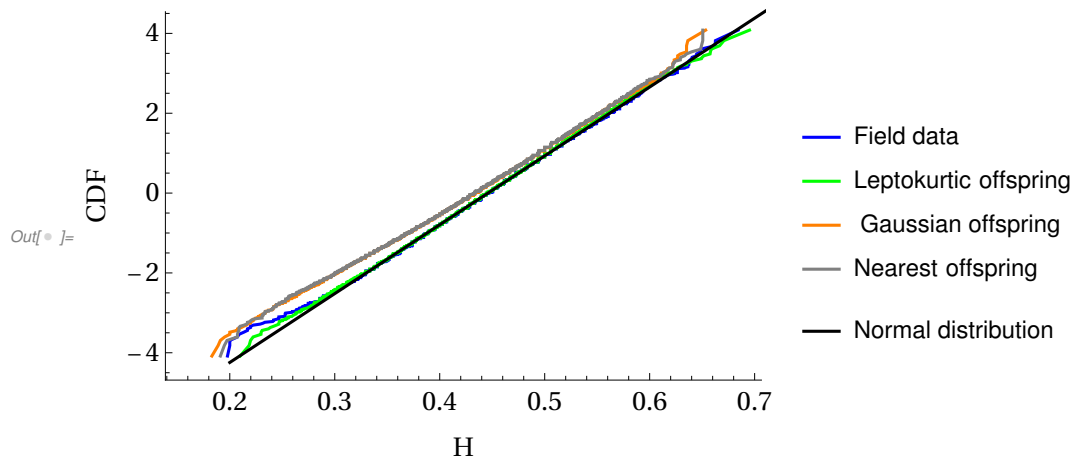
datasets	t-test on means	F-test on variances	KS-test on CDF
{field, realistic}	{-1.23636, 0.21633}	{0.381824, 0.702592}	{0.0195499, 0.0003}
{field, Gaussian}	{25.098, 0}	{-10.9317, 0}	{0.0989129, 0}
{field, Nearest}	{24.6033, 0}	{-10.4267, 0}	{0.0953787, 0}
{realistic, Gaussian}	{26.3286, 0}	{-11.0437, 0}	{0.0964524, 0}
{realistic, Nearest}	{25.8317, 0}	{-10.4986, 0}	{0.0913971, 0}
{Gaussian, Nearest}	{-0.451357, 0.651735}	{0.433252, 0.664832}	{0.00608419, 0.802}

This compares the cumulative distributions (CDF) of the field data (blue) with the three simulated datasets, on a probit scale, such that a normal distribution would appear as a straight line (black). The vertical axis is in standard deviations:

```
In[ ]:= mnH = Mean[H]; sdH = StandardDeviation[H];
```

```
Show[ ListLinePlot[Transpose[{Sort[#, sdl[#]]} & /@ {H, realhet, gausshet, nearhet},
  PlotStyle → {Blue, Green, Orange, Gray}, PlotLegends →
    {"Field data", "Leptokurtic offspring", "Gaussian offspring", "Nearest offspring"},
  AxesOrigin → {0.15, -4.5}, Frame → {{True, False}, {True, False}},
  FrameLabel → {{ "CDF", ""}, {"H", ""}},
  BaseStyle → {FontFamily → "Times", FontSize → 14}],
```

```
Plot[  $\frac{x - mnH}{sdH}$ , {x, 0.2, 0.8}, PlotStyle → Black, PlotLegends → {"Normal distribution"}]]
```



Saving actual and simulated distributions of H

```
In[ ]:= Export["Simulated actual H 7 Aug.csv", H];
Export["Simulated realistic H 7 Aug.csv", realhet];
Export["Simulated Gaussian H 7 Aug.csv", gausshet];
Export["Simulated nearest H 7 Aug.csv", nearhet];
```

Does realistic simulation show an excess of individuals with low H ?

We fit the data to a mixed Gaussian distribution, as described above: a small fraction ϕ of individuals have mean and variance of heterozygosity half that of the bulk distribution, representing a fraction ϕ of individuals produced by selfing. Here, we fit data from “realistic” matings (i.e., with distance between parents drawn from the empirical pollen dispersal distribution) to see whether a mixed Gaussian also fits better than a single Gaussian, even though there is no selfing allowed. Since the tail has only a few individuals, we make 100 replicates to see whether the excess of low heterozygosity individuals seen in the data is consistent with mating between nearby individuals, given the observed pollen dispersal.

For the actual data, allowing a low rate of selfing, $\phi=0.00086$, increases log likelihood by 11.27:

```
In[ ]:= H = MLH[SNP]; likdata = maxlik[H];
```

```
TableForm[Append[likdata, -Differences[likdata[[All, 1]]], TableDepth → 2]
```

```
Out[ ]:= TableForm=
```

```
52 490.      {ϕ → 0.000860938, m → 0.446207, v → 0.00332422}
52 478.7     {m → 0.446015, v → 0.00336122}
11.2722
```

For one round of “realistic” simulated matings, allowing a low rate of selfing, $\phi=0.00060$, increases log likelihood by 0.75:

```
In[ ]:= likreal = maxlik[realhet];
```

```
TableForm[Append[likreal, -Differences[likreal[[All, 1]]], TableDepth → 2]
```

```
Out[ ]:= TableForm=
```

```
52 583.6     {ϕ → 0.000225006, m → 0.445476, v → 0.00332096}
52 582.8     {m → 0.445426, v → 0.00333005}
0.749153
```

This compares the CDF for the field data (blue), one shuffle of the field data (grey), and the best fitting mixed Gaussian, with $\phi=0.0086$ (red):

```
Show[ListLinePlot[Transpose[{Sort[##], sdl[##]} & /@ {H, sH1},
```

```
PlotLegends → {"field data", "shuffled field data"},
```

```
PlotStyle → {Blue, Gray}, AxesOrigin → {0.18, -4.5},
```

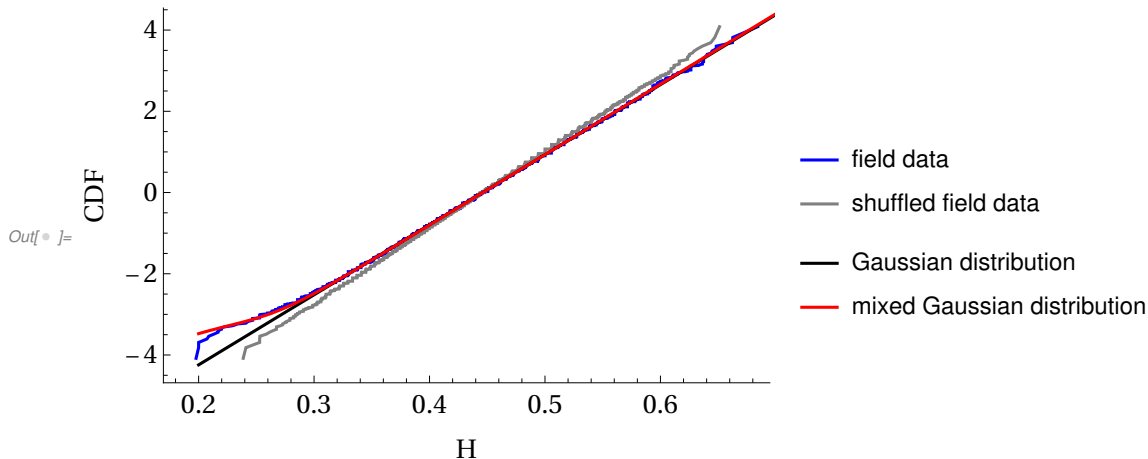
```
Frame → {{True, False}, {True, False}}, FrameLabel → {"H", "CDF"}],
```

```
Plot[{ $\frac{x - mnH}{sdH}$ , invG[ $\frac{\phi}{2} \left( 1 + \text{Erf}\left[\frac{x - m / 2}{\sqrt{v}}\right] \right) + \frac{1 - \phi}{2} \left( 1 + \text{Erf}\left[\frac{x - m}{\sqrt{2 v}}\right] \right)$ ] /. likdata[[1, 2]]},
```

```
{x, 0.2, 0.7}, PlotStyle → {Black, Red},
```

```
PlotLegends → {"Gaussian distribution", "mixed Gaussian distribution"}],
```

```
BaseStyle → {FontFamily → "Times", FontSize → 14}]
```



Compiling this table of 100 replicate sets of offspring, and fitting the mixed Gaussian to them, takes ~ 30 minutes. `Dynamic[i]` tracks progress:

```
In[ ]:= Dynamic[i]
```

```
Out[ ]:= i
```

```
In[ ]:= hetoffreps = Table[distreal = RandomVariate[HistogramDistribution[pollendisp], nInds];
  Offspringrep[distreal, snpraw], {i, 100}];
  liks = maxlik /@ hetoffreps;
  increase = Flatten[-Differences[liks] & /@ liks[All, All, 1]];
   $\phi$ est =  $\phi$  /. liks[All, 1, 2];
   $\phi$ Data =  $\phi$  /. likdata[1, 2]
```

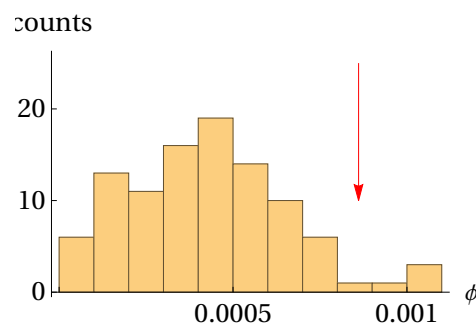
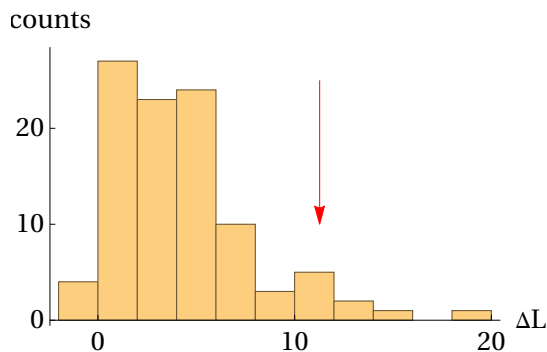
```
Out[ ]:= 0.000860938
```

It is worth saving these replicates:

```
In[ ]:= Save["Replicate offspring hetoffreps", hetoffreps];
  Save["ML from replicate offspring hetoffreps", liks];
```

The figure shows the distribution of the gain in log likelihood by fitting a mixed Gaussian (left), and the estimated fraction of selfed individuals (right). The arrows show the observed values. 5/100 replicates exceed the observed $\Delta L=11.27$, and 4/100 exceed the observed $\phi=0.00086$. Thus, evidence for an excess of highly inbred offspring, compared with offspring from simulated “realistic” matings, is marginally significant.

```
In[ ]:= dlData = likdata[All, 1].{1, -1};
  GraphicsRow[{
    Show[Histogram[increase, 10, AxesLabel → {"ΔL", "counts"},
      Ticks → {{0, 10, 20}, {0, 10, 20}}, PlotRange → All,
      Graphics[{Red, Arrow[{dlData, 25}, {dlData, 10}]}],
      BaseStyle → {FontFamily → "Times", FontSize → 14}],
    Show[Histogram[ $\phi$ est, 10, AxesLabel → {" $\phi$ ", "counts"},
      Ticks → {{0, 0.0005, 0.001}, {0, 10, 20}}, PlotRange → All,
      Graphics[{Red, Arrow[{ $\phi$ Data, 25}, { $\phi$ Data, 10}]}],
      BaseStyle → {FontFamily → "Times", FontSize → 14}]]]
```



```
In[ ]:= {Length[Select[increase, # > dldata &]], Length[Select[phiest, # > phiData &]]}
Out[ ]:= {5, 4}
```

Isolation by distance (IBD) in the data

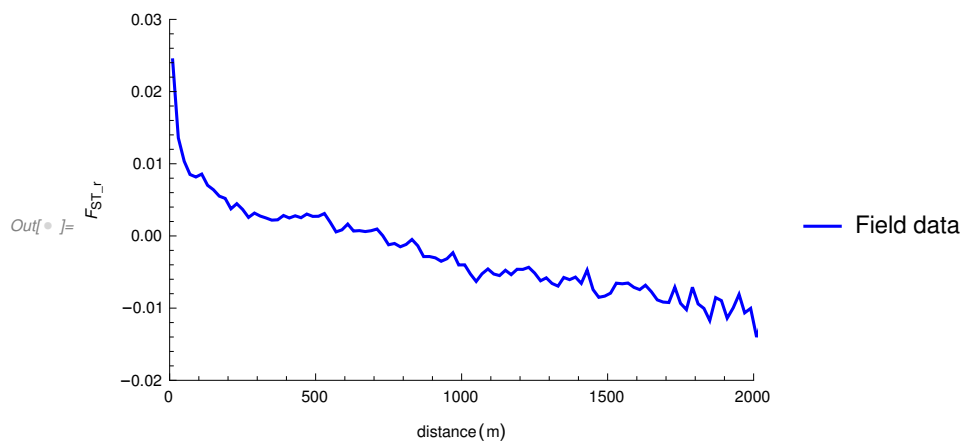
IBD in the complete Planoles dataset

The following code calculates isolation by distance in the SNP data. We calculate IBD for each year separately, from 2009 to 2010; $F_r = \frac{H_t - H_r}{H_t}$, where $H_t = \langle 2 \bar{p} \bar{q} \rangle$, where $\langle \rangle$ is the average over loci (calculated by af). We impute the missing data based on the allele frequency at respective markers. ibdt calculates H_r for different distances r and rel calculates $F_{st,r}$. To get the combined IBD from 11 years, we take the average IBD pattern from 11 years given by relibd. We obtain an F_{st} of 0.024 from these data

```
af = alfrqdata[##] & /@ imputeddata // N;

ibdt = Table[ibddata[locperyear[[i]], snpperyearprocess[[i]], {i, Length[snpperyear]}];
rel = Table[Transpose[{ibdt[[i]][All, 1], ((ibdt[[i]][All, 2] - af[[i]]) / (1 - af[[i]]))}],
  {i, Length[snpperyear]}];
ss = SortBy[N@*First]@Flatten[ibdt, 1];
ibdpooled = Mean[##] & /@ Flatten[DeleteCases[BinLists[ss, 20, 2], {}, 2], 1];
relibd = Transpose[{ibdpooled[All, 1], (ibdpooled[All, 2] - Mean[af]) / (1 - Mean[af])}];
relibd[[1]]

ListLinePlot[relibd, PlotStyle -> Blue, PlotRange -> {{0, 2000}, {0.03, -0.02}},
  AxesOrigin -> {0, -0.02}, PlotLegends -> {"Field data"},
  Frame -> {{True, False}, {True, False}}, FrameLabel -> {"distance(m)", "FST,r"},
  BaseStyle -> {FontFamily -> "Times", FontSize -> 14}]
```



```
In[ ]:= relibd = Import["/home/psurendr/Desktop/Inbreeding Paper/NewDat/ibdpooled.csv"];
```

```
In[ ]:= relibd[[1]]
```

```
Out[ ]:= {10.2325, 0.0243887}
```

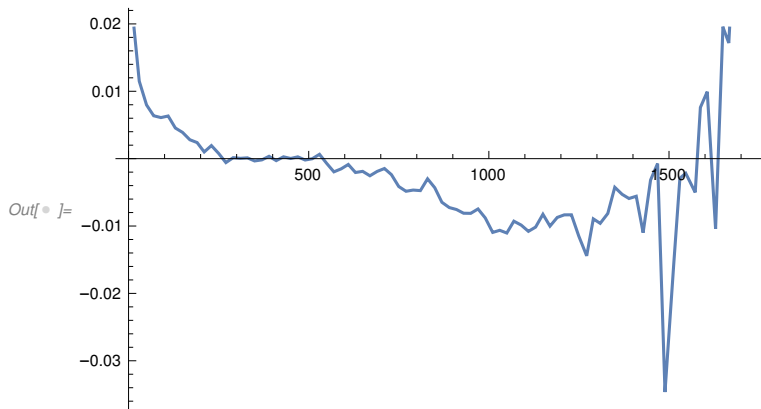
IBD in core region used for simulation

We recalculate IBD using the same procedure as above for a smaller region defined as `coreInds`, the area used for the simulation below. An F_{ST} of 0.022 is obtained. The noise at the right of the plot arises simply because there is now little data at such large distances.

```
afsmall = alfrqdata[##] & /@ imputedsmall;
ibdtsmall = Table[ibddata[locsmall[[i]], snpsmall[[i]], {i, Length[snpsmall]}];
relsm = Table[Transpose[
  {ibdtsmall[[i]][All, 1], ((ibdtsmall[[i]][All, 2] - afsmall[[i]]) / (1 - afsmall[[i]]))}, {i, 11}];
sssm = SortBy[N@*First]@Flatten[ibdtsmall, 1];
ibdpooleddsmall = Mean[##] & /@ Flatten[DeleteCases[BinLists[sssm, 20, 2], {}, 2], 1];
relibdsm = Transpose[
  {ibdpooleddsmall[All, 1], (ibdpooleddsmall[All, 2] - Mean[afsmall]) / (1 - Mean[afsmall])}];
relibdsm[[
  1]]
```

```
Out[ ]:= {10.349, 0.0220151}
```

```
In[ ]:= ListLinePlot[relibdsm]
```



Simulating the spatial pedigree for 1000 generations:

Initializations for simulation

`nsim` is the number of individuals and `loci` is the number of loci for which genotypes will be assigned. `t` specifies the number of generations to simulate. `startloc` chooses the starting locations from the core region in 2009.

```
In[ ]:= nsim = 15500; loci = 91; t = 1000;
```

```
In[ ]:= startloc = LocData[nsim, 1];
```

Generate spatial pedigree

Given the starting locations, `gg` generates a spatial pedigree storing the locations and parents of every individual for all generations. By default the population is simulated for $t=1000$ generations. Set a different value for t under “Initializations for simulation”

```
In[ ]:= gg = makeGenealogyReal[startloc, t]; // Timing
```

Isolation by distance at $t=1000$ from 10 genotypes assigned for a simulated pedigree

`Inigen` assigns two alleles 0 and 1 with equal frequencies for each loci to start with in {0,0}, {0,1} and {1,1} format and `innew` returns the haplotypes. `endloc` returns the locations at the end of t generations. `reps` is the number of replicate genotypes assigned for a given pedigree. As the required `nsim` is large, the analysis is performed using HPC Cluster at IST Austria.

```
inigen = RandomInteger[BernoulliDistribution[0.5], {nsim, loci, 2}];
innew = Transpose[#, 2] & /@ inigen;
reps = 10;
```

```
In[ ]:= endloc = gg[[-1, -1]];
```

`allreps` return the genotypes in {0,1,2} format at the end of t generations. Since finding pairwise genetic distance between all pairs for large N is not feasible, we subsample $l=5000$ individuals to find the F_{ST} and isolation by distance.

`fstv` finds the how F_{ST} changes with geographic distance for all replicates and `fstvcombined` gives the average across ‘rep’ replicates. .

```
In[ ]:= allreps2 = Table[(r = r + 1; Reps[innew, gg[2 ;; -1, 1]]), reps]; // Timing
```

```
l = 5000; list = RandomSample[Range[nsim], l];
```

```
In[ ]:= fstv = FvsR[allreps2, list, endloc];
```

```
In[ ]:= fstvcombined = Transpose[{fstv[1, All, 1], Mean[fstv[All, All, 2]]}];
```

Results:

We simulated 5 pedigrees each from which 10 replicate genotypes are assigned. All simulations are performed in the cluster which returns the spatial pedigree, genotypes and how F_{ST} changes with distance.

`ped1rep` to `ped5rep` contains F_{ST} values across distance for 10 replicate genotypes from each of the 5 simulated pedigrees. `ibdcombined` calculates the average isolation by distance pattern across the 10 replicates. We see that for $N=15,500$ individuals, F_{ST} from the simulation match the data.

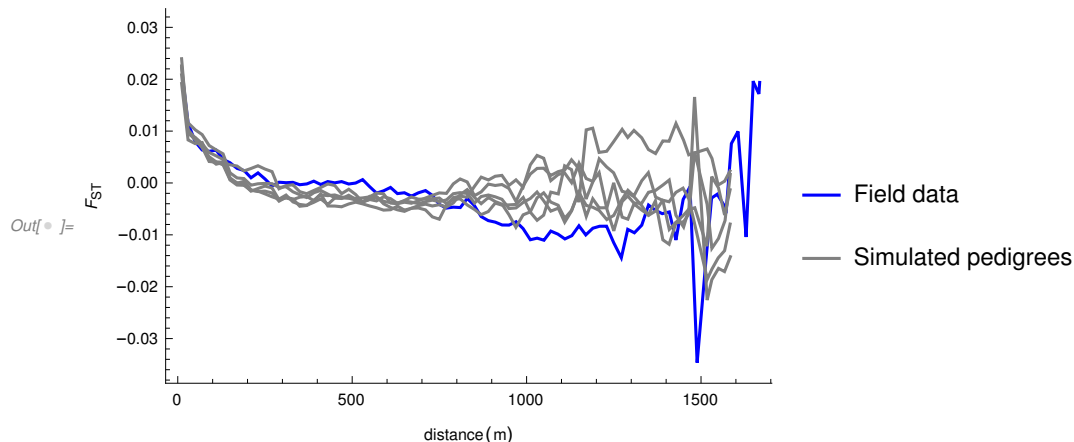
Import saved results:

```
In[ ]:= ped1rep = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree1/ped1rep_",
  ToString[i], ".tsv"]], {i, 10}];
ped2rep = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree2/ped2rep_",
  ToString[i], ".tsv"]], {i, 10}];
ped3rep = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree3/ped3rep_",
  ToString[i], ".tsv"]], {i, 10}];
ped4rep = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree4/ped4rep_",
  ToString[i], ".tsv"]], {i, 10}];
ped5rep = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree5/ped5rep_",
  ToString[i], ".tsv"]], {i, 10}];

In[ ]:= ibdcombined1 = Transpose[{ped1rep[[1, All, 1]], Mean[ped1rep[[All, All, 2]]]}];
ibdcombined2 = Transpose[{ped2rep[[1, All, 1]], Mean[ped2rep[[All, All, 2]]]}];
ibdcombined3 = Transpose[{ped3rep[[1, All, 1]], Mean[ped3rep[[All, All, 2]]]}];
ibdcombined4 = Transpose[{ped4rep[[1, All, 1]], Mean[ped4rep[[All, All, 2]]]}];
ibdcombined5 = Transpose[{ped5rep[[1, All, 1]], Mean[ped5rep[[All, All, 2]]]}];
```

Isolation by distance pattern from the data (blue) matches with that from the simulation. For better visualisation, a log scale is used which shows their concordance.

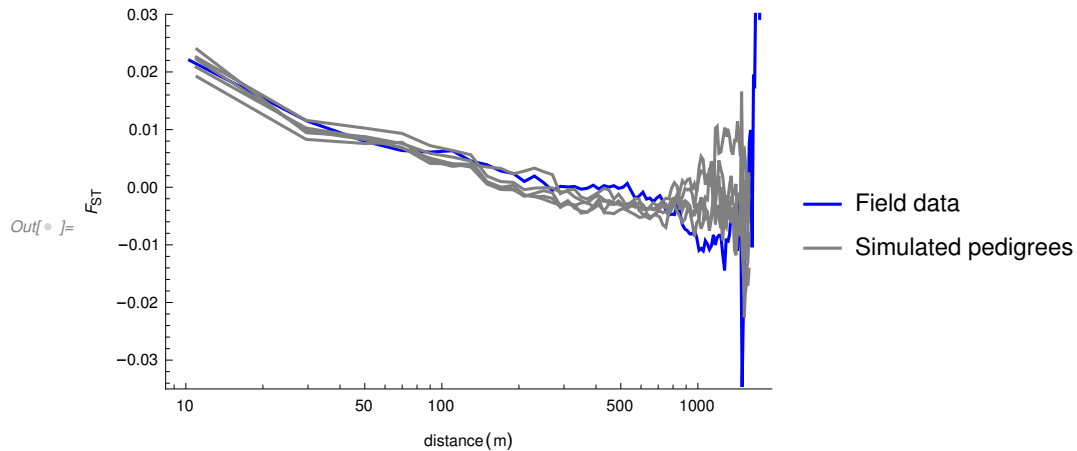
```
In[ ]:= Show[ListLinePlot[relibdsm, PlotStyle → Blue, Frame → {{True, False}, {True, False}},
  FrameLabel → {"distance(m)", "FST"}, PlotLegends → {"Field data"}],
ListLinePlot[{ibdcombined1, ibdcombined2, ibdcombined3, ibdcombined4, ibdcombined5},
  PlotLegends → {"Simulated pedigrees"}, PlotRange → All,
  PlotStyle → {Gray, Gray, Gray, Gray, Gray},
  PlotRange → {All, {0.03, -0.035}}, AxesOrigin → {0, -0.035}]
```



```

In[ ] := ListLogLinearPlot[
  {relibdsm, ibdcombined1, ibdcombined2, ibdcombined3, ibdcombined4, ibdcombined5},
  Frame → {{True, False}, {True, False}}, FrameLabel → {"distance(m)", "FST"},
  PlotLegends → {"Field data", "Simulated pedigrees"},
  Joined → True, PlotRange → {All, {0.03, -0.035}},
  PlotStyle → {Blue, Gray, Gray, Gray, Gray, Gray}, AxesOrigin → {0, -0.035}]

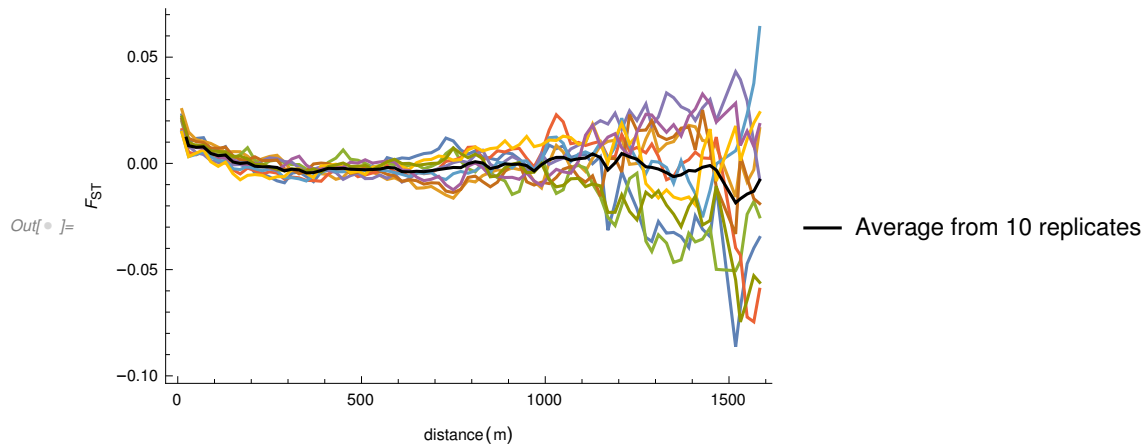
```



```

In[ ] := Show[ListLinePlot[ped1rep, PlotRange → All, AxesOrigin → {0, -0.1},
  Frame → {{True, False}, {True, False}}, FrameLabel → {"distance(m)", "FST"}, ListLinePlot[
  ibdcombined1, PlotStyle → Black, PlotLegends → {"Average from 10 replicates"}]]

```



mwithin and mbw gives the mean Fst within and between replicates. Similarly sdwithin and sdbw gives the standard deviation of Fst within and between replicates. 'Within values' are calculated as the average across the 5 pedigrees (each which contains an average from 10 replicates). Between values are calculated as the average across 5*10 replicate genotypes. Table below summarises the Fst value from the data, from each pedigree and their averages.

```

sd = StandardDeviation[#[[All, 1, 2]] & /@ {ped1rep, ped2rep, ped3rep, ped4rep, ped5rep};
sdwithin = Mean[sd];
mwithin = Mean[{ibdcombined1[[1, 2]], ibdcombined2[[1, 2]],
  ibdcombined3[[1, 2]], ibdcombined4[[1, 2]], ibdcombined5[[1, 2]]};
mbw = Mean[Flatten[#[[All, 1, 2]] & /@ {ped1rep, ped2rep, ped3rep, ped4rep, ped5rep}]]
sdbw = StandardDeviation[
  Flatten[#[[All, 1, 2]] & /@ {ped1rep, ped2rep, ped3rep, ped4rep, ped5rep}]]

```

```

In[ ]:= Transpose[{"", "Data", "Pedigree 1", "Pedigree 2", "Pedigree 3",
  "Pedigree 4", "Pedigree 5", "Within pedigree", "Between pedigree"},
{"Mean", relibdsm[[1, 2]], ibdcombined1[[1, 2]], ibdcombined2[[1, 2]],
  ibdcombined3[[1, 2]], ibdcombined4[[1, 2]], ibdcombined5[[1, 2]], mwithin, mbw},
Join[{"Standard Deviation", ""}, sd, {sdwithin, sdbw}]] // TableForm

```

Out[]:= `TableForm=`

	Mean	Standard Deviation
Data	0.0220151	
Pedigree 1	0.0191636	0.00383073
Pedigree 2	0.0225697	0.00348048
Pedigree 3	0.0222072	0.0022149
Pedigree 4	0.0239484	0.00143087
Pedigree 5	0.0208351	0.0034573
Within pedigree	0.0217448	0.00288286
Between pedigree	0.0217448	0.0033309

Proposed vs Realised dispersal distance:

propseed and proppoll gives th seed and pollen dispersal distributions while realseed and realpoll finds the realised seed and pollen dispersal distributions from the pedigree.

Figures below show that the realised and proposed seed and pollen dispersal distributions agree with each other (values shown in the table below).

```

propseed = HistogramDistribution[seeddisp];
proppoll = HistogramDistribution[polldisp];

gg1 = ToExpression[Import[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree1/gg15500.tsv"]];

```

```

In[ ]:= realdisp = disp[gg1, #] & /@ Range[2, 1001];

```

```

In[ ]:= realseed = HistogramDistribution[Flatten[realdisp[[All, 1]]]];

```

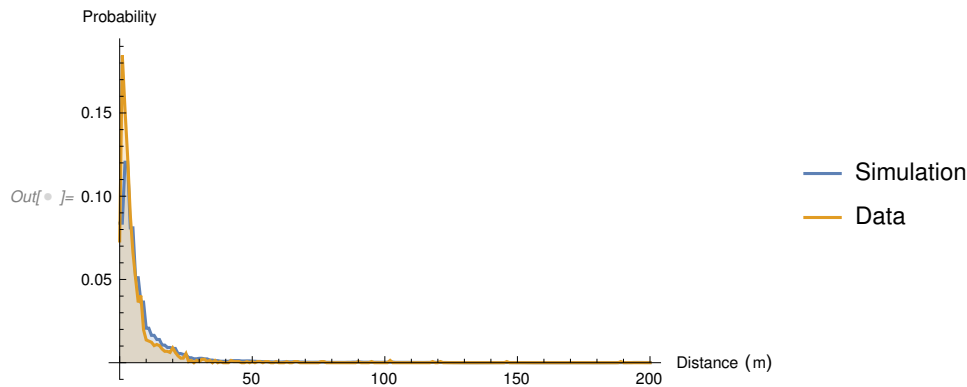
```

In[ ]:= realpoll = HistogramDistribution[Flatten[realdisp[[All, 2]]]];

```

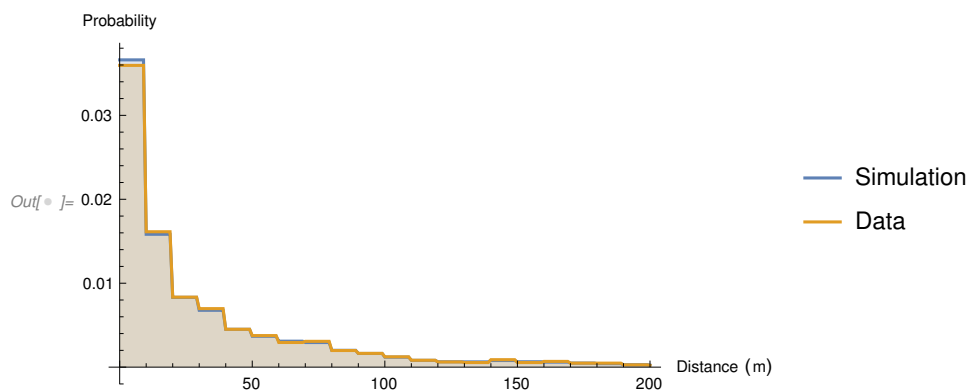
Seed dispersal distribution

```
In[ ] := DiscretePlot[{PDF[realseed, x], PDF[propseed, x]}, {x, 0, 200}, PlotRange → All,
  PlotLegends → {"Simulation", "Data"}, AxesLabel → {"Distance (m)", "Probability"}]
```



Pollen dispersal distribution

```
In[ ] := DiscretePlot[{PDF[realpoll, x], PDF[proppoll, x]}, {x, 0, 200}, PlotRange → All,
  PlotLegends → {"Simulation", "Data"}, AxesLabel → {"Distance (m)", "Probability"}]
```



```
{{"", "Mean", "StandardDeviation"},
 {"Proposed seed", Mean[seeddisp], StandardDeviation[seeddisp]},
 {"Proposed pollen", Mean[polldisp], StandardDeviation[polldisp]},
 {"Realised seed- pedigree 1", Mean[realseed], StandardDeviation[realseed]},
 {"Realised pollen- pedigree 1", Mean[realpoll],
  StandardDeviation[realpoll]}} // TableForm
```

Out[] := TableForm=

	Mean	StandardDeviation		Mean	StandardDeviation
Proposed seed	9.52468	38.2242	Realised seed	12.797	40.1884
Proposed pollen	62.684	119.327	Realised pollen	62.5955	118.765

Simulating population with uniform density

Rx and Ry gives the dimension of the 2D habitat simulated with heterogenous density. startuni gives the initial locations that are uniformly distributed in this 2D space for nsim individuals. We simulate for N=15,500 individuals (for which Fst from data match to the pedigree generated from heteroge-

neously distributed locations) and see that F_{ST} of 0.05 is obtained (higher than heterogeneous case with same N). We also find that one needs 40,000 individuals in this case for F_{ST} to be 0.02.

```
nuni = 40 000;
Rx = Differences[MinMax[Flatten[locsmall, 1][All, 1]];
Ry = Differences[MinMax[Flatten[locsmall, 1][All, 2]];

startuni = makeN[nuni, Rx[[1]], Ry[[1]];

gguni = makeGenealogyReal[startuni, t, Rx[[1]], Ry[[1]]];
```

uniformreps gives F_{ST} vs distance from a single pedigree with 10 replicate genotypes and uniformibd-com finds their average.

```
In[ ]:= uniformreps = Table[Import[
    StringJoin["/home/psurendr/Desktop/Inbreeding Paper/Cluster/Uniform40000/unirep_",
        ToString[i], ".tsv"], {i, 10}];

In[ ]:= uniformibdcomb = Transpose[{uniformreps[[1, All, 1]], Mean[uniformreps[[All, All, 2]]]}];

In[ ]:= uniformibdcomb[[1]]

Out[ ]:= {13.317, 0.0226438}

In[ ]:= StandardDeviation[uniformreps[[All, 1, 2]]

Out[ ]:= 0.000946486
```

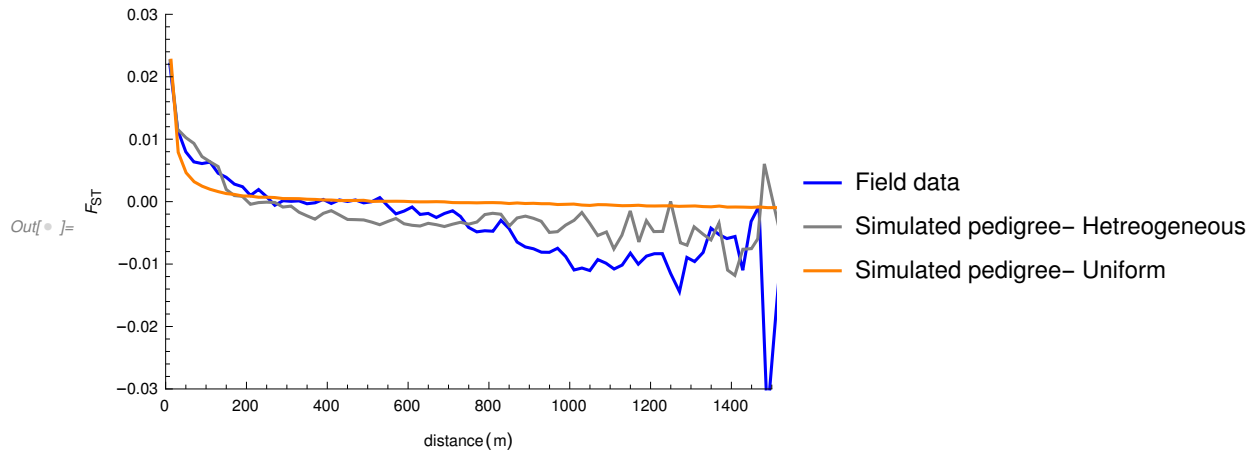
Comparing heterogeneous and homogenous density

Isolation by distance patterns suggest that the data fits the model with heterogeneous density (extrapolated from the known spatial structure) better than a model with uniform density (orange). Comparing the variance within replicates, the heterogeneous case shows much more variance within replicates than the uniform case as expected.

```

In[ ]:= ListLinePlot[{relibdsm, ibdcombined2, uniformibdcomb},
  Frame → {{True, False}, {True, False}}, FrameLabel → {"distance(m)", "FST"},
  PlotStyle → {Blue, Gray, Orange}, PlotLegends →
    {"Field data", "Simulated pedigree- Heterogeneous", "Simulated pedigree- Uniform"},
  PlotRange → {{0, 1500}, {0.03, -0.03}}, AxesOrigin → {0, -0.035}]

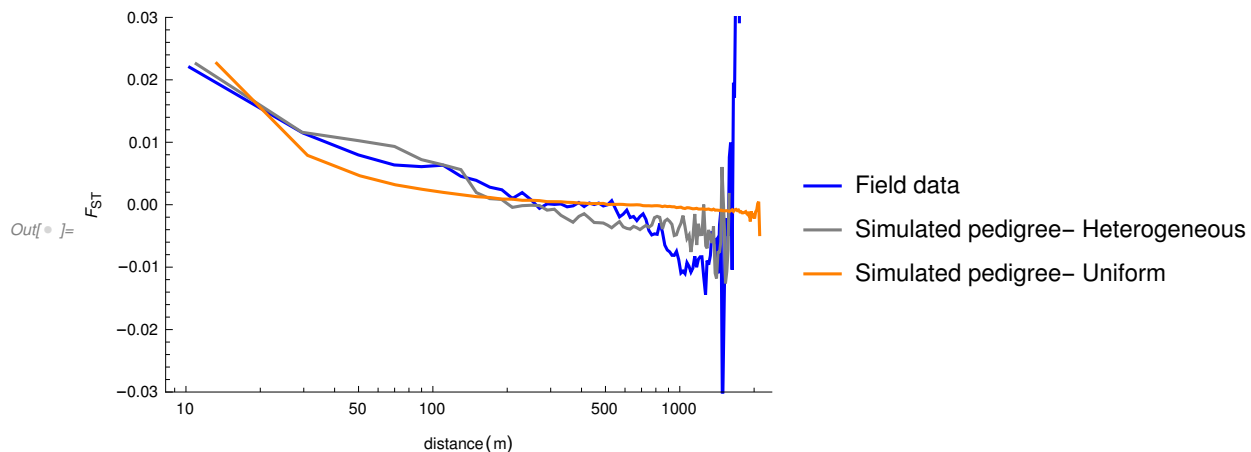
```



```

In[ ]:= ListLogLinearPlot[{relibdsm, ibdcombined2, uniformibdcomb},
  Frame → {{True, False}, {True, False}}, FrameLabel → {"distance(m)", "FST"},
  PlotStyle → {Blue, Gray, Orange}, PlotLegends →
    {"Field data", "Simulated pedigree- Heterogeneous", "Simulated pedigree- Uniform"},
  Joined → True, PlotRange → {All, {0.03, -0.03}}, AxesOrigin → {0, -0.035}]

```



Validation

The simulation is validated by comparing identity coefficients (by finding isolation by distance) directly from the pedigree and from 10 replicate genotypes for a smaller population size of $n_{sim}=1000$ individuals. The steps for finding isolation by distance from the genotypes is similar to that mentioned above under ‘simulating the spatial pedigree for 1000 generations’.

Under ‘initialization for simulation’, use $n_{sim}=1000$. gg generates the spatial pedigree, but addition-

ally follows the index of each individual.

```
gg = makeGenealogyRealindexed[startloc, t];
```

```
pars = #[All, 2] & /@ gg[2 ;; -1];
```

```
par = Mod[#, n, 1] & /@ pars;
```

Calculate isolation by distance following 'Isolation by distance at t=1000 from 10 genotypes assigned for a simulated pedigree', but replace endloc as endloc=gg[-1,All,3] and allreps2 as Table[(r = r + 1; Repts[innew, par]), reps];

Isolation by distance is directly calculated from the pedigree given by fsthet.

```
hetibd = pedibd[gg];
```

```
fsthet =
```

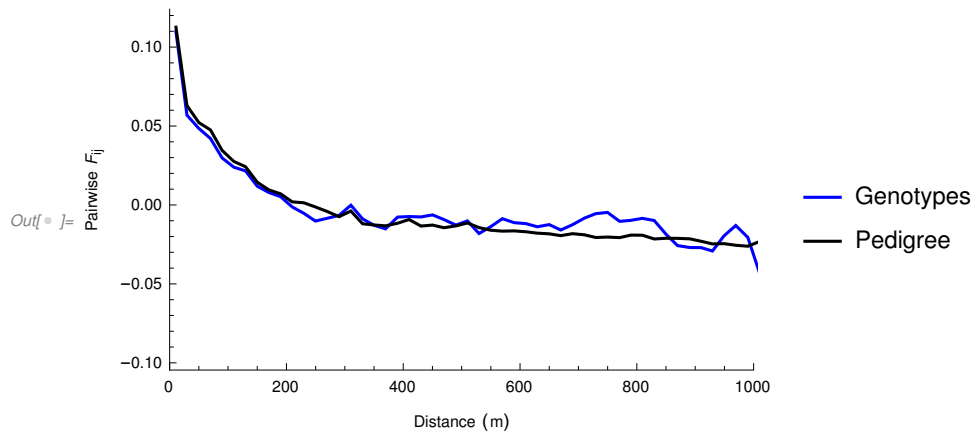
```
Transpose[{hetibd[2][All, 1], (hetibd[2][All, 2] - hetibd[1, -1]) / (1 - hetibd[1, -1])}];
```

```
In[ ]:= ListLinePlot[{fstvrrcombined, fsthet},
```

```
PlotRange -> {{0, 1000}, All}, PlotStyle -> {Blue, Black},
```

```
AxesOrigin -> {0, -0.1}, PlotLegends -> {"Genotypes", "Pedigree"},
```

```
Frame -> {{True, False}, {True, False}}, FrameLabel -> {"Distance (m)", "Pairwise  $F_{ij}$ "}]
```



Fis

ped1gene to ped5gene gives the genotypes from 10 replicates for each of the 5 pedigrees. unigene gives the genotypes from population simulated with uniform density.

```

In[ ]:= ped1gene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree1/ped1gene_",
  ToString[i], ".tsv"]], {i, 10}];
ped2gene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree2/ped2gene_",
  ToString[i], ".tsv"]], {i, 10}];
ped3gene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree3/ped3gene_",
  ToString[i], ".tsv"]], {i, 10}];
ped4gene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree4/ped4gene_",
  ToString[i], ".tsv"]], {i, 10}];
ped5gene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/15500/pedigree5/ped5gene_",
  ToString[i], ".tsv"]], {i, 10}];
unigene = Table[Import[StringJoin[
  "/home/psurendr/Desktop/Inbreeding Paper/Cluster/Uniform40000/unigene_",
  ToString[i], ".tsv"]], {i, 10}];

In[ ]:= Fis[gene_] := Mean[DeleteCases[hetF[##], Indeterminate]] & /@ gene // N;

Fhet = Fis[##] & /@ {ped1gene, ped2gene, ped3gene, ped4gene, ped5gene};
Funi = Fis[unigene];

Fped = Mean[##] & /@ Fhet; Fpedsd = StandardDeviation[##] & /@ Fhet;

Fbetmean = Mean[Flatten[Fhet]];
Fbetstd = StandardDeviation[Flatten[Fhet]];
Fwmean = Mean[Fped];
Fwsd = Mean[Fpedsd];

In[ ]:= Mean[Funi]
Out[ ]:= 0.0203295

In[ ]:= StandardDeviation[Funi]
Out[ ]:= 0.000589165

In[ ]:= Fisdata = Mean[hetF[Flatten[snpssmall, 1]] // N]
Out[ ]:= 0.0208217

```

Fis values from the simulations and data is summarised in the table below. The values from the simulations match with the data. As with F_{ST} , there is much variation between and within pedigrees in the heterogeneous case than in the uniform case


```

In[ ]:= Transpose[{"", "Data", "Pedigree 1", "Pedigree 2", "Pedigree 3", "Pedigree 4",
  "Pedigree 5", "Within pedigree", "Between pedigree", "", "Uniform denisty"},
  Join[{"Mean", FisdData}, Fped, {Fwmean, Fbetmean, "", Mean[Funi]}],
  Join[{"Standard Deviation", ""}, Fpedsd,
    {Fwsd, Fbetsd, "", StandardDeviation[Funi]}]] // TableForm

```

Out[]:= `//TableForm=`

	Mean	Standard Deviation
Data	0.0208217	
Pedigree 1	0.0216405	0.00236404
Pedigree 2	0.0253811	0.00178975
Pedigree 3	0.024722	0.00149331
Pedigree 4	0.0244013	0.00132576
Pedigree 5	0.0259428	0.0016042
Within pedigree	0.0244175	0.00171541
Between pedigree	0.0244175	0.00225323
Uniform denisty	0.0203295	0.000589165

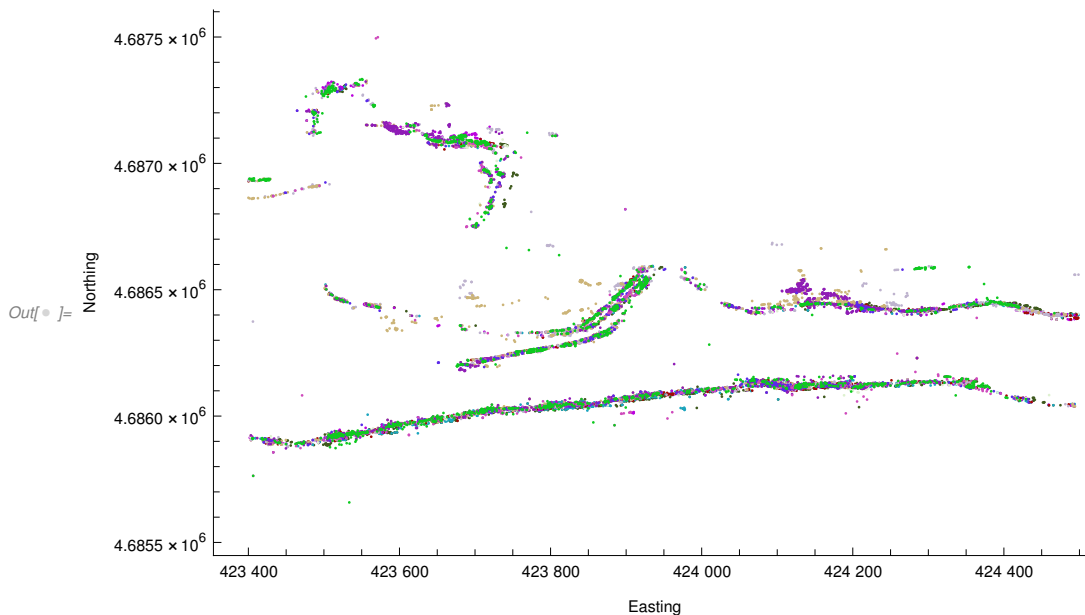
Visualizing locations

This is a map of the whole Planoles area; colours indicate years when plants were first sampled:

```

In[ ]:= ListPlot[listofloc, PlotStyle → RandomColor[11],
  Frame → {{True, False}, {True, False}}, FrameLabel → {"Northing", ""}, {"Easting", ""}]

```

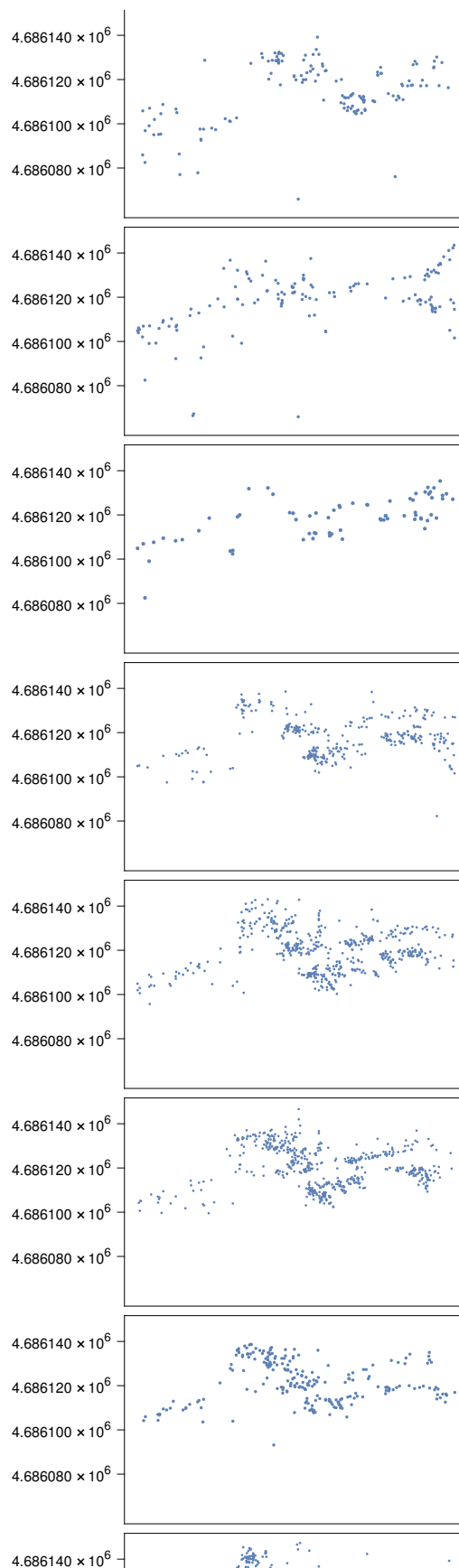


These are the distributions in each year, for the areas indicated by boxes in Fig. S1:

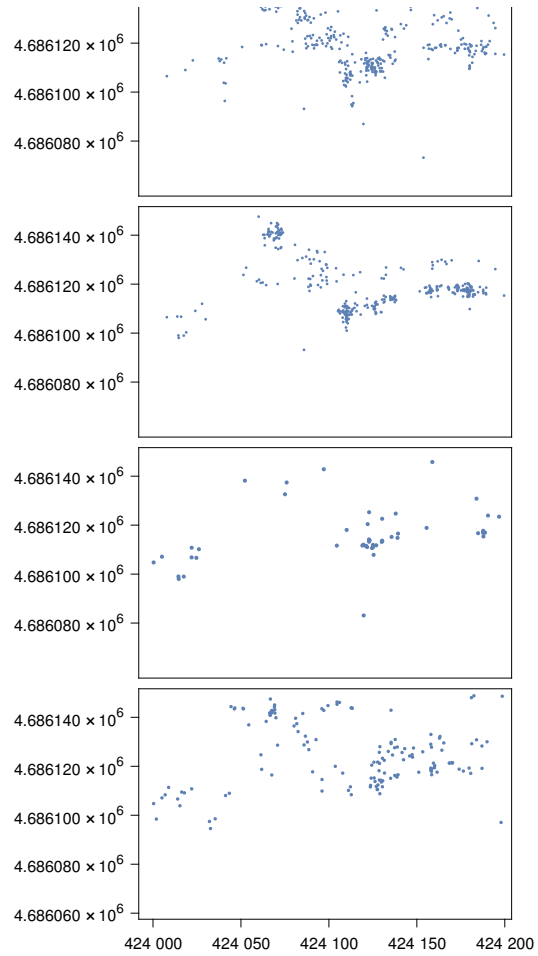
```

Filterloc1[l_] := Select[l, And[424 000 < First[#] < 424 200, Last[#] < 4.68615 * 10^6] &]
floc = Filterloc[#] & /@ listofloc;
ListPlot[floc, PlotLayout → "Column"]

```



Out[]=



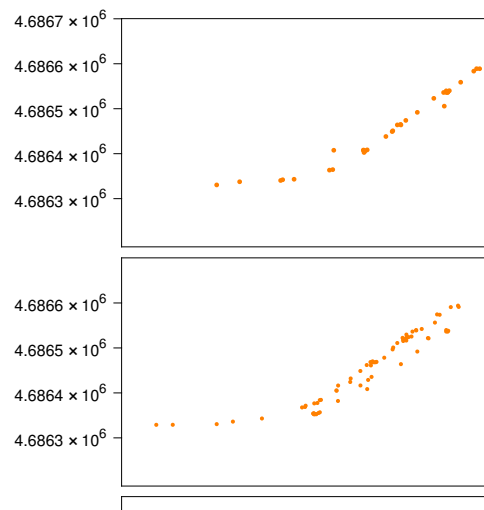
```

In[ ]:= Filterloc2[l_] := Select[l,
  And[First[#] < 423 950, First[#] > 423 750, Last[#] < 4.6867 * 10^6, Last[#] > 4.6862 * 10^6] &]

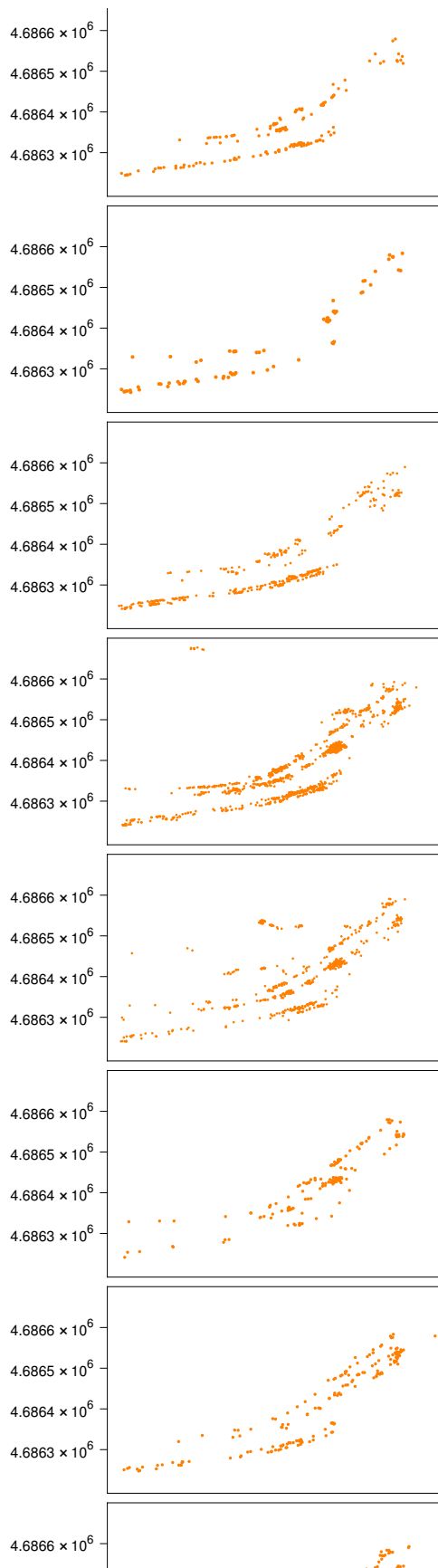
In[ ]:= floc2 = Filterloc2[#] & /@ listofloc;

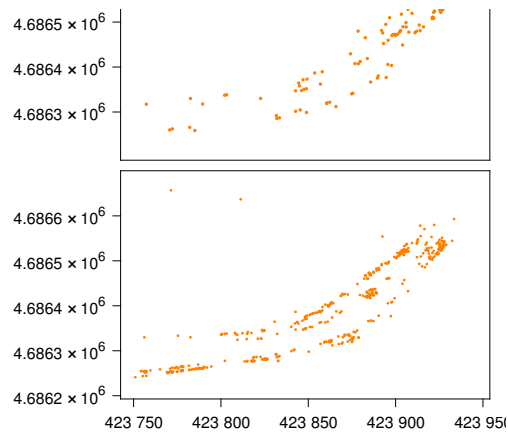
In[ ]:= ListPlot[floc2, PlotLayout -> "Column", PlotStyle -> Orange]

```



$Out[\text{ }] =$





Definitions

Importing data

Import data

Evaluating the cell below generates a button which allows you to search for required files, saving the name in `fn`, and allowing the directory to be set correctly.

`In[]:= FileNameSetter[Dynamic[fn]]`

`Out[]:=`

```
In[ ]:= (*SetDirectory[
  "/Users/NickBarton/Manuscripts/Inbreeding depression PRSB 2020/Data for SI";*)
SetDirectory["/home/psurendr/Desktop/Inbreeding Paper/NewDat/FinalDataset"];
rawData = Import["SlimmedData.csv"];
rawLinkage = Import["LinkageMap.csv"];
dispData = Import["Dispersaldata.xlsx"];
```

nInds, nSNP, ID, XY, SNP, SNPnames, AliveRec

```

In[ ] := nInds::usage = "nInds is the number of individuals in the full dataset";
nSNP::usage = "nSNP is the number of SNP in the full dataset";
ID::usage = "ID lists the individual IDs";
XY::usage = "XY lists the locations";
SNP::usage =
  "SNP lists the SNP genotypes, coded as 0,1,2; missing data are -9 or -10";
SNPnames::usage = "SNPnames are the names of the 91 SNP";
AliveRec::usage =
  "AliveRec[t] lists whether a plant was alive in year t (2009–2019); 0, 1 or -9";

```

```

In[ ] := nInds = Length[rawData] - 1;
ID = rawData[[2 ;; -1, 1]];
XY = rawData[[2 ;; -1, {2, 3}]];
SNP = rawData[[2 ;; -1, 16 ;; 106]];
nSNP = Length[SNP[[1]]];
SNPnames = rawData[[1, 16 ;; 106]];
AliveRec[t_Integer] := AliveRec[t] = rawData[[2 ;; -1, 5 + (t - 2009)]];

```

Linkage map: SNPposn

```

In[ ] := SNPposn::usage =
  "SNPposn lists the linkage group and map position in cM, in the form
  {1,4.1}; derived from {LG,cm_Adj}. LG=9 is
  chloroplast, LG=10 is unknown linkage group.";
SNPposn = rawLinkage[[All, {2, 3}]];

```

nSNPperInd, nIndperSNP

```

In[ ] := nSNPperInd::usage =
  "nSNPperInd[] gives the # of SNP that are scored successfully for each
  individual. nSNPperInd[{s1,...}] restricts to a specified set of SNP";
nSNPperInd[] = nSNPperInd[Range[Length[First[SNP]]]];
nSNPperInd[s_List] := nSNPperInd[s] = Length[DeleteCases[#, [s], -9 | -10]] & /@ SNP;

```

```

In[ ]:= nIndperSNP::usage =
  "nIndperSNP[] gives the # of individuals that are scored successfully for each
  SNP. nIndperSNP[{s1,...}] restricts to a specified set of individuals";
nIndperSNP[] = nIndperSNP[Range[Length[SNP]]];
nIndperSNP[inds_List] :=
  nIndperSNP[inds] = Length[DeleteCases[#, -9 | -10]] & /@ Transpose[SNP];

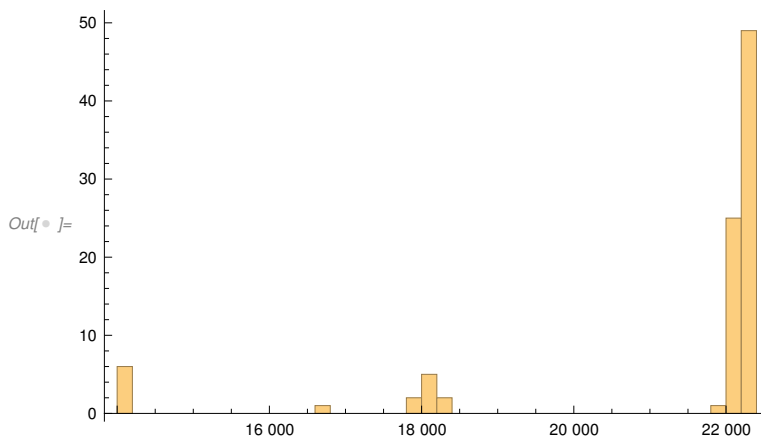
```

For the Planoles individuals, there are only 6 SNP with <15,000 plants scored:

```

In[ ]:= Histogram[ndat = nIndperSNP[], 50, PlotRange -> All]
Length[Select[ndat, #] & /@ {# < 1000 &, 1000 ≤ # < 3000 &, 3000 ≤ # < 15 000 &, 15 000 ≤ # &}

```



```

Out[ ]:= {0, 0, 6, 85}

```

disp

These dispersal data are from trios, where both parents can be assigned to an individual, with high confidence. Assignment was based on SNPPIT (David Field, unpublished data). The distance to the nearest parent was assumed to be seed dispersal, and between the two parents, pollen dispersal.

```

In[ ]:= disp::usage = "disp lists seed & pollen
  dispersal distances from the pedigree, for 1463 trios";
disp = dispData[[1]][2 ;; -1, {1, 2}];

```

Main definitions

alleleFreq

```
In[ ]:= alleleFreq::usage = "alleleFreq[{0,1,2,-9,...}] gives the
      allele frequency. Can also be applied to the full dataset.";
alleleFreq[g : {{{(0 | 1 | 2 | -9 | -10) ...} ...}}] := alleleFreq /@ Transpose[g];
alleleFreq[g : {(0 | 1 | 2 | -9 | -10) ...}] := Module[{gg = DeleteCases[g, -9 | -10]},
      If[gg == {}, Indeterminate, Mean[gg] / 2];
```

```
In[ ]:= alfrqdata::usage=
"alfrqdata calculates the average allele frequency from multiple loci";
alfrqdata[d_] := Module[{p, q},
p = alleleFreq[d];
q = 1 - p;
Mean[p^2 + q^2]]
```

Fis

```
In[ ]:= Fis::usage = "Fis[{0,1,2,-9,...}] gives the
      heterozygote deficit. Can be applied to the full dataset.";
Fis[g : {{{(0 | 1 | 2 | -9 | -10) ...} ...}}] := Fis /@ Transpose[g];
Fis[g : {(0 | 1 | 2 | -9 | -10) ...}] := Module[{gg = DeleteCases[g, -9 | -10], p, bc, tot},
      If[gg == {}, Indeterminate,
      bc = BinCounts[gg, {0, 3}]; tot = Total[bc];
      p = bc.{0, 1 / 2, 1} / tot;
      1 -  $\frac{bc[[2]]}{2 \text{ tot } p (1 - p)}$ ]];

```

hetF

```
In[ ]:= hetF::usage =
      "hetF[{0|1|2}...] estimates the heterozygote deficit from the allele frequency";
```



```

In[ ]:= hetF[g : {{(0 | 1 | 2 | -9 | -10) ...} ...}] := hetF /@ Transpose[g];
hetF[g : {(0 | 1 | 2 | -9 | -10) ...}] := Module[{g1 = DeleteCases[g, -9 | -10], p, n},
  If[g1 == {}, Indeterminate,
    p = Mean[g1] / 2;
    n = Length[g1];
    If[0 < p < 1, 1 -  $\frac{\text{Count}[g1, 1]}{2 n p (1 - p)}$ , Indeterminate]]];

```

regX

```

In[ ]:= regX::usage =
  "regX[{1,3,4,...},k] gives the regression of allele frequency on X, for a
  given set of individuals, and a given SNP.";
regX[inds_List, s_Integer] := D[Fit[DeleteCases[Transpose[{XY[[inds, 1]], SNP[[inds, s]]],
  {_, -9 | -10} | {-9 | -10, _}] /. {x_, p_} :> {x, p / 2}, {1, x}, x], x];

```

planolesIndsT, allInds, allSNP, coreInds, PlanolesXY, PlanolesXY2, coreXY

```

In[ ]:= PlanolesXY::usage =
  "PlanolesXY = {{421000,426000},{4685600,4687700}} defines a rectangle
  containing most of the Planoles samples";
PlanolesXY = {{421 000, 426 000}, {4 685 600, 4 687 700}};

```

```

In[ ]:= MinMax /@ Transpose[XY]

```

```

Out[ ]:= {{418 865., 434 653.}, {4.68473 × 106, 4.68798 × 106}}

```

```

In[ ]:= PlanolesXY2::usage =
  "PlanolesXY2 = {{418800,434700},{4684700,4687800}} defines a rectangle
  containing ALL Planoles samples";
PlanolesXY2 = {{418 800, 434 700}, {4 684 700, 4 687 800}};

```

```

In[ ]:= coreXY::usage = "{{423400,424500},{4685500,4687500}}
  defines a 1.1Km*2Km rectangle that was regularly sampled";
coreXY = {{423 400, 424 500}, {4 685 500, 4 687 500}};

```

```

In[ ]:= allSNP::usage = "allSNP is the full set of SNP";
allSNP = Range[nSNP];

```

```
In[ ]:= allInds::usage = "allInds is the full set of individuals";
allInds = Range[nInds];
```

```
In[ ]:= planolesIndsT::usage =
  "planolesIndsT[t] stores those individuals that were alive at time t,
  according to AliveRec. planolesIndsT[] is the same as allInds";
planolesIndsT[] := Range[nInds];
planolesIndsT[t_] := planolesIndsT[t] = Pick[planolesIndsT[], AliveRec[t], 1];
```

```
In[ ]:= coreInds::usage =
  "coreInds[t] stores those individuals in the core that were alive at time
  t, according to AliveRec. coreInds[] gives all the individuals.
  coreInds[{t1,t2,...}] gives individuals in multiple years";
coreInds[] := coreInds[] = Select[planolesIndsT[],
  (coreXY[[1, 1] < XY[[#, 1] < coreXY[[1, 2]] & (coreXY[[2, 1] < XY[[#, 2] < coreXY[[2, 2]] &];
coreInds[t1 : {___ Integer}] := Flatten[coreInds /@ t1];
coreInds[t_Integer] :=
  coreInds[t] = Select[planolesIndsT[], (coreXY[[1, 1] < XY[[#, 1] < coreXY[[1, 2]] &
  (coreXY[[2, 1] < XY[[#, 2] < coreXY[[2, 2]] & (AliveRec[t][[#, 1] == 1) &];
```

These are the # of individuals in the core and in Planoles for the 11 years:

```
In[ ]:= {#, Length[coreInds[#]], Length[planolesIndsT[#]]} & /@ Range[2009, 2019] // TableForm
Out[ ]//TableForm=
```

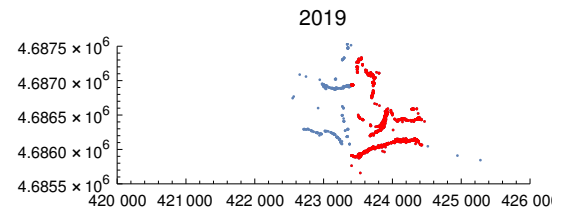
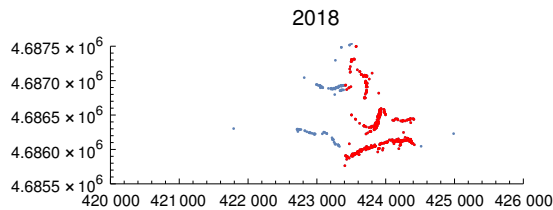
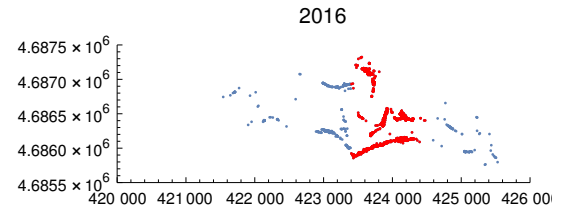
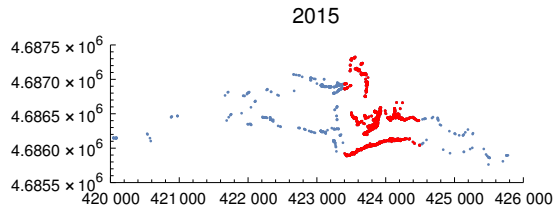
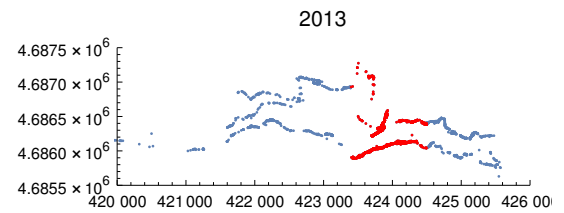
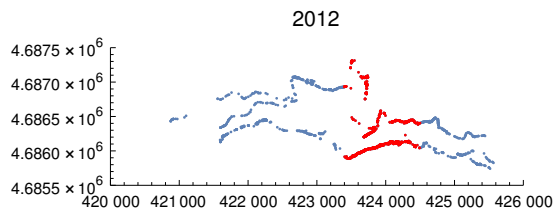
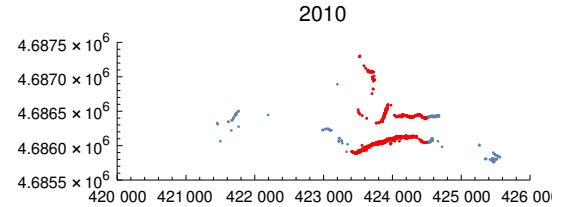
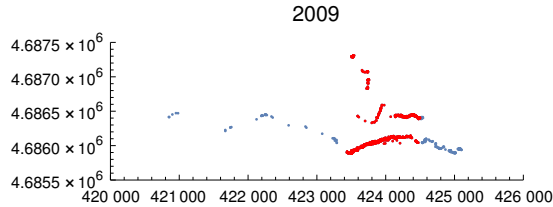
2009	1249	1449
2010	1581	1974
2011	1483	1959
2012	1147	2217
2013	1992	3516
2014	3673	5450
2015	2632	3410
2016	2229	2891
2017	957	1237
2018	624	763
2019	1949	2468

These are the distributions of individuals over the years; the core region that is the basis of simulations is in red:

```

In[ ]:= pr = {{420 000, 426 000}, {4 685 500, 4 687 500}};
gr[t_] := Show[ListPlot[XY[[planolesIndsT[t]]], PlotRange -> pr],
  ListPlot[XY[[coreInds[t]]], PlotStyle -> Red, PlotRange -> pr],
  PlotLabel -> ToString[t], AspectRatio -> 1 / 3];
GraphicsGrid[Map[gr, {{2009, 2010, 2011}, {2012, 2013, 2014},
  {2015, 2016, 2017}, {2018, 2019}}, {2}]]

```



gridTable, gridFreqTable

```

In[ ]:= gridTable::usage =
  "gridTable[{{x0,x1},{y0,y1}},δ] makes a list of {{x,y},{i1,...}}, listing positions
    of squares and the individuals therein. Deletes empty demes.";
gridTable[{{x0_, x1_}, {y0_, y1_}}, δ_] := gridTable[{{x0, x1}, {y0, y1}}, δ] =
  Module[{x, y},
    DeleteCases[Flatten[Table[{{x + δ / 2, y + δ / 2},
      Pick[Range[Length[XY]], ((x ≤ #[[1] < x + δ) && (y ≤ #[[2] < y + δ)) & /@ XY]],
      {x, x0, x1 - δ, δ}, {y, y0, y1 - δ, δ}], 1], {_, {}]]];

```

```

In[ ]:= gridFreqTable::usage =
  "gridFreqTable[{{x0,x1},{y0,y1}},δ] makes a list of {{x,y},{n1,p1},...},
  listing # of individuals in each deme, and
  the frequency of SNP. Deletes empty demes.";
gridFreqTable[{{x0_, x1_}, {y0_, y1_}}, δ_] := gridFreqTable[{{x0, x1}, {y0, y1}}, δ] =
  gridTable[{{x0, x1}, {y0, y1}}, δ] /.
  {{x_, y_}, s_} :> {{x, y}, Transpose[{nIndperSNP[s], alleleFreq[SNP[s]]]}}];

```

Fst

```

In[ ]:= Fst::usage = "Fst[{{n1,p1},...}] estimates Fst for a list of populations";
Fst[np_List] := Module[{np1 = DeleteCases[np, {0, _} | {_, Indeterminate}], n, p, pb},
  n = np1[[All, 1]]; p = np1[[All, 2]]; pb = n.p / Total[n];
  If[0 < pb < 1,  $\frac{(n-1) \cdot (p-pb)^2}{Total[n-1] pb (1-pb)}$ , Indeterminate]];

```

gridFstTable

```

In[ ]:= gridFstTable::usage = "gridFreqTable[{{x0,x1},{y0,y1}},δ]
  makes a list of Fst for each SNP. Deletes empty demes.";
gridFstTable[{{x0_, x1_}, {y0_, y1_}}, δ_] := gridFstTable[{{x0, x1}, {y0, y1}}, δ] =
  Fst /@ Transpose[Last /@ gridFreqTable[{{x0, x1}, {y0, y1}}, δ] // N];

```

MLH

```

In[ ]:= MLH::Usage = "MLH[{0,1,2,-9,...}] or MLH[SNP] gives the fraction of heterozygous loci";
MLH[g : {___ Integer}] := Module[{gg = Select[g, NonNegative]},
  Count[gg, 1] / Length[gg]];
MLH[g : {___ List}] := MLH /@ g;

```

invG

```

In[ ]:= invG::usage =
  "invG[p] gives the # of standard deviations corresponding to probability
  p; it is the inverse of the standard Gaussian.";
invG[p_] :=  $\sqrt{2}$  InverseErf[2 p - 1];

```

tTest

In[]:=

```
tTest::usage =
  "tTest[dm,{v1,v2},{n1,n2}] gives a t statistic, assuming that variances are
    the same; there are n1+n2-2 df. tTest[{f,n},{a,b}] compares the two
    datasets that are generated by applying f to a, b; n[a] gives the #";

tTest[d_, {v1_, v2_}, {n1_, n2_}] := d /  $\sqrt{\frac{(n1-1)v1 + (n2-1)v2}{(n1+n2-2)} \left(\frac{1}{n1} + \frac{1}{n2}\right)}$ ;

tTest[{f_, n_}, {x_, y_}] := Module[{t},
  t = tTest[Mean[f[x]] - Mean[f[y]], Variance[f[#]] & /@ {x, y}, n /@ {x, y}];
  {t, 1 - CDF[StudentTDistribution[n[x] + n[y] - 2], Abs[t]]};
```

fTest

In[]:=

```
fTest::usage =
  "fTest[{f,n},{rh,fh}] compares the variances of two datasets that are generated
    by applying f to a, b; n[a] gives the #";

fTest[{ff_, n_}, {x_, y_}] := Module[{f},
  f =  $\frac{\text{Variance}[ff[x]]}{\text{Variance}[ff[y]']}$ ;
  {f, 1 - CDF[FRatioDistribution[n[x] - 1, n[y] - 1], Max[f,  $\frac{1}{f}$ ]]};
```

likG, rng, sdl, n,maxlik

In[]:=

```
likG::usage =
  "likG[φ,m,v][x] gives the log likelihood for a model which has the bulk
    with mean and variance {m,v}, and a fraction φ with half that mean
    and variance. likG[m,v][x] sets φ=0. The factor  $\sqrt{2\pi}$  is omitted";
```

```
ln[ ]:= likG[m_, v_][x_] := likG[0, m, v][x];
likG[0, m_, v_][x_] := - $\frac{1}{2}$  (Log[v] + (x - m)2 / v);
likG[φ_, m_, v_][x_] := Log[ $\frac{\phi}{\sqrt{v/2}}$  Exp[- $\frac{(x - m/2)^2}{v}$ ] +  $\frac{1 - \phi}{\sqrt{v}}$  Exp[- $\frac{(x - m)^2}{2v}$ ]];
```

```
ln[ ]:= n::usage = "n[H] gives the # of values";
rng::usage = "rng[H] lists n[fh] equally
  spaced values between 0 and 1; used to construct the CDF";
sdl::usage = "sdl[fh] gives the # of sd corresponding to rng[fh], using invG";
n[h_List] := Length[h];
rng[h_List] := rng[h] = Module[{δ =  $\frac{1}{n[h]}$ }, Range[ $\frac{\delta}{2}$ , 1 -  $\frac{\delta}{2}$ , δ]];
sdl[h_List] := sdl[h] = invG[rng[h] // N];
```

```
ln[ ]:= maxlik::usage =
  "Calculates the difference in the
  log likelihood between a single and double Gaussian";
maxlik[d_] := Module[{l1, l2, fm1, fm2},
  l1 = Total[likG[φ, m, v] /@ d];
  l2 = Total[likG[m, v] /@ d];
  fm1 = FindMaximum[l1, {φ, 0.001, 0.00001, 0.01},
    {m, 0.445, 0.44, 0.45}, {v, 0.0033, 0.0001, 0.005}];
  fm2 = FindMaximum[l2, {m, 0.44}, {v, 0.003, 0.0001, 0.005}];
  {fm1, fm2}]
```

corrH

```
ln[ ]:= corrH::usage =
  "corrH[inds,snp] stores the matrix of correlations of homozygosity between
  loci. May take a few minutes for a large dataset. ";
corrH[inds_List, snp_List] := Module[{ss = SNP[inds], fn},
  fn[{a1_, a2_}] :=
    If[Length[Union[a1]] ≤ 1 ∨ Length[Union[a2]] ≤ 1, -9, N[Correlation[a1, a2]]];
  corrH[inds, snp] = Outer[fn[Transpose[DeleteCases[ss[All, {#1, #2}]],
    {_, -9 | -10} | {-9 | -10, _}]] /. {2 → 0} &, snp, snp, 1];
```

covH

```

In[ ]:= covH::usage =
  "covH[inds,snp] stores the matrix of covariance of homozygosity between
    loci. May take a few minutes for a large dataset. ";
covH[inds_List, snp_List] := Module[{ss = SNP[[inds]], fn},
  fn[{a1_, a2_}] := N[Covariance[a1, a2]];
  covH[inds, snp] =
    Outer[fn[Transpose[DeleteCases[ss[[All, {#1, #2}]], {_, -9 | -10} | {-9 | -10, _}]] /.
      {2 -> 0}] &, snp, snp, 1]];

```

SNPLG

```

In[ ]:= SNPLG::usage = "SNPLG[x] stores the set of SNP that are in linkage group x";
SNPLG[x_] := SNPLG[x] = Flatten[Position[SNPposn[[All, 1]], x]];

```

shuffle

```

In[ ]:= shuffle::usage = "shuffle[{{x1,1,...},...}] shuffles columnds";
shuffle[s_List] := Transpose[RandomSample /@ Transpose[s]];

```

nearestF

```

In[ ]:= nearestF::usage =
  "nearestF[inds] stores a function that gives the closest individuals to
    a given point. nearestF[{{x1,y1},...}] gives the same, but indexing
    individuals in numerical order corresponding to the list of positions";

```

```

In[ ]:= nearestF[inds : {__ Integer}] := nearestF[inds] = Nearest[XY[[inds]] -> inds];
nearestF[xy : {{_, _} ...}] := nearestF[xy] = Nearest[xy -> Range[Length[xy]]];

```

randomR

```

In[ ]:= randomR::usage = "randomR[r] generates a random displacement with radius r";

```

```

In[ ]:= randomR[r_] := Module[{θ = 2 π Random[], r {Sin[θ], Cos[θ]}}];

```

neighbour

```

In[ ]:= neighbour::usage =
  "neighbour[inds,seed] stores a set of nearest neighbours. neighbour[inds,r,seed]
  stores a list of the nearest individual to a random point,
  distance r from the focal individual; excludes the focal
  individual. neighbour[inds,r,{j,k},seed] chooses k points on
  a circle, and takes the j individuals nearest the circle. ";

```

```

In[ ]:= neighbour[inds_List, seed_] :=
  neighbour[inds, seed] = Complement[nearestF[inds][XY[[#]], 2], {#}][[1]] & /@ inds;

```

```

In[ ]:= neighbour[inds_List, r_, seed_] := neighbour[inds, r, seed] =
  Complement[nearestF[inds][XY[[#]] + randomR[r], 2], {#}][[1]] & /@ inds;

```

```

In[ ]:= neighbour[inds_List, r_, {j_Integer, k_Integer}, seed_] :=
  neighbour[inds, r, {j, k}, seed] =
    neighbour[#, inds, r, {j, k}] & /@ inds;
neighbour[ind_Integer, inds_List, r_, {j_Integer, k_Integer}] :=
  Module[{θ = 2 π (Random[] + Range[0, 1 - 1/k, 1/k]), y, rnf, dl},
    y = Transpose[r {Cos[θ], Sin[θ]}];
    rnf = DeleteCases[#, ind][[1]] & /@ nearestF[inds][XY[[ind]] + #] & /@ y, 2];
    dl = Abs[Norm[XY[[#]] - XY[[ind]] - r] & /@ rnf;
    Last /@ Take[Sort[Transpose[{dl, rnf}]], j];

```

Checking neighbour

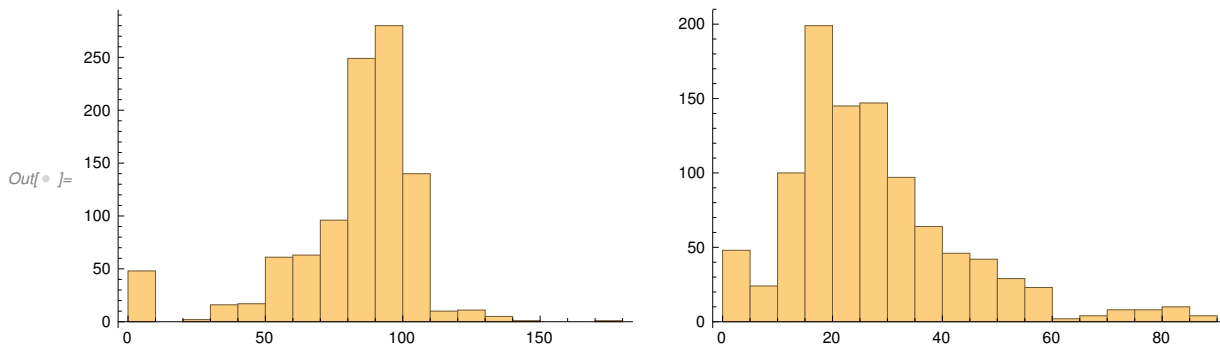
neighbour[allInds,100,{10,12},1] chooses, for every Planoles individual, 10 individuals approx 100m away. There is considerable scatter, because there may not be enough individuals to choose:


```

In[ ]:= xy1 = neighbour[allInds, 100, {10, 12}, 1];
tt = Table[(Norm[# - XY[allInds[[j]]]] & /@ XY[xy1[[j]]]), {j, 1, 1000}];
msd = {Mean[#], StandardDeviation[#]} & /@ tt; 10
GraphicsRow[Histogram[#, 20] & /@ Transpose[msd]]
Mean /@ Transpose[msd]

```

Out[]:= 10



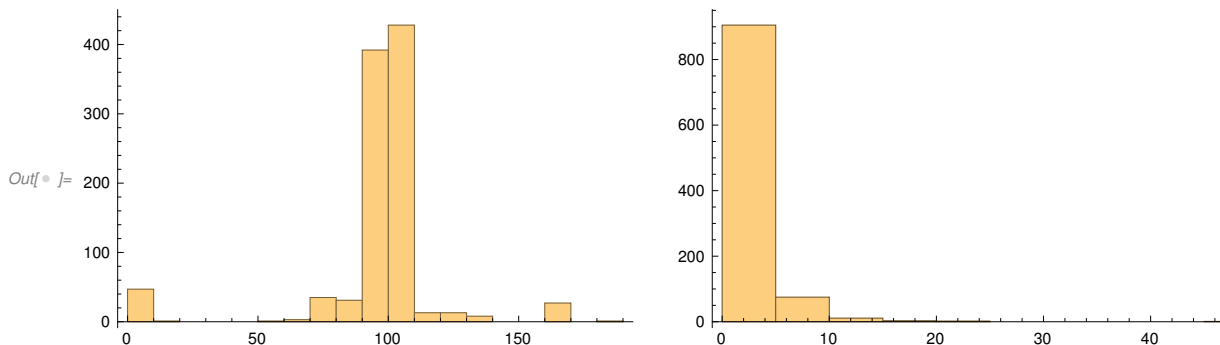
Out[]:= {81.9196, 27.3585}

If we choose just two, the mean clusters around 100 quite closely, with a few exceptions:

```

In[ ]:= xy2 = neighbour[allInds, 100, {2, 12}, 1];
tt2 = Table[(Norm[# - XY[allInds[[j]]]] & /@ XY[xy2[[j]]]), {j, 1, 1000}];
msd = {Mean[#], StandardDeviation[#]} & /@ tt2;
GraphicsRow[Histogram[#, 20] & /@ Transpose[msd]]
Mean /@ Transpose[msd]

```



Out[]:= {96.5687, 2.0761}

```

In[ ]:= Timing[nn = neighbour[allInds, 1]; Dimensions[nn]]

```

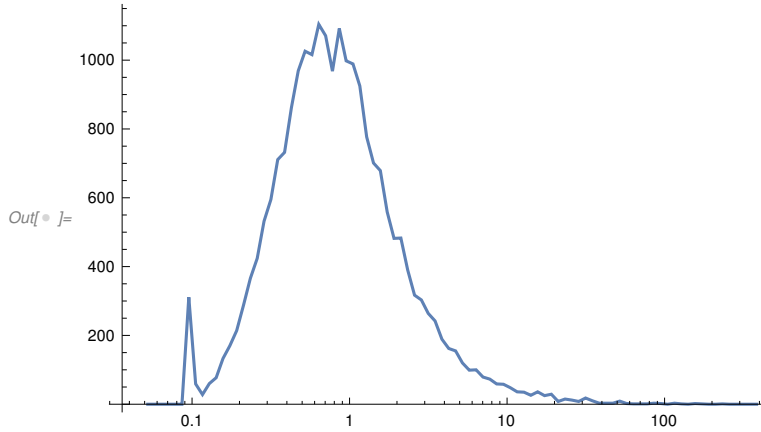
Out[]:= {0.124056, {22 353}}

This is the distribution of distances to the nearest neighbour, on a log scale. Note that 311 individuals have neighbours at exactly the same place. The mean distance is 1.4m.

```

In[ ]:= rr = MapThread[Norm[XY[[#1]] - XY[[#2]]] &, {allInds, nn}];
bc = BinCounts[Log[0.1 + rr], {-3, 6, 0.1}];
ListLogLinearPlot[Transpose[{Exp[Range[-3 + 0.05, 6 - 0.05, 0.1]], bc}], Joined → True]
{Mean[rr], StandardDeviation[rr], MinMax[rr], Length[Cases[rr, 0 | 0.]], Length[rr]}

```



```

Out[ ]:= {1.41828, 4.29069, {0., 235.683}, 311, 22353}

```

makeKids

```

In[ ]:= makeKids::usage =
  "makeKids[{p1,p2},...],snp,rec] makes a set of offspring from the given
  pairs, using the given SNP and recombination intervals. Omitting
  rec assumes no linkage. makeKids[{d1,d2},{m1,m2},{X1,X2}] generates
  an offspring from the two parents {d1,d2} using recombination
  breakpoints listed in {m1,m2} and starting with one or other
  parent chromosome according to Xi=0 or 1; di are expressed as
  {0,1,2,...} and will be randomly phased. alleleFreq->{p1,...} imputes.";
makeKids[{d1 : {__ Integer}, d2 : {__ Integer}}, bp : {{__ Integer}, {__ Integer}},
  xx : {{0 | 1}, {0 | 1}}, opts___Rule] :=
Module[{pp = alleleFreq /. {opts} /. Options[makeKids], p1, p2, n = Length[d1], mm},
  If[pp === False,
    p1 = randomPhase[d1]; p2 = randomPhase[d2],
    p1 = randomPhase[d1, pp]; p2 = randomPhase[d2, pp]];
  mm = MapThread[makeMask[#, #2, n] &, {xx, bp}];
  (p1[[1]] mm[[1]] + p1[[2]] (1 - mm[[1]])) + (p2[[1]] mm[[2]] + p2[[2]] (1 - mm[[2]]));

```

```

In[ ]:= Options[makeKids] = {alleleFreq → False};

```

```

In[ ]:= makeKids[pars : {{_Integer, _Integer} ...}, snp_List, opts___Rule] :=

  makeKids[pars, snp, ConstantArray[ $\frac{1}{2}$ , Length[snp]-1], opts];

makeKids[pars : {{_Integer, _Integer} ...}, snp_List, rec_List, opts___Rule] :=
  Module[{mm, xx, ninds = Length[pars]},
    mm = Partition[makeMask[rec, 2 ninds], 2];
    xx = RandomInteger[{0, 1}, {ninds, 2}];
    MapThread[
      makeKids[{SNP[[#1][1]], snp], SNP[[#1][2], snp]], {#2, #3, opts} &, {pars, mm, xx}]]];

```

randomPhase

```

In[ ]:= randomPhase::usage =
  "randomPhase[{0,1,2,-9,...}] generates two haplotypes, {{0,0,1,-9,...},{0,1,1,-9,...}}
  randomly assigning the heterozygotes.
  randomPhase[{0,1,2,-9,...},{p1,...}] imputes missing values. ";
randomPhase[dip_List] := Transpose[dip /. {0 -> {0, 0}, 1 -> RandomSample[{0, 1]},
  2 -> {1, 1}, (-9 | -10) -> {-9, -9}}];
randomPhase[dip_List, p_List] := Module[{pos = Flatten[Position[dip, -9 | -10]]},
  Transpose[ReplacePart[dip /. {0 -> {0, 0}, 1 -> RandomSample[{0, 1]}, 2 -> {1, 1}},
    ({# -> RandomInteger[BernoulliDistribution[p[[#]]], 2]) & /@ pos}]]];

```

makeMask

```

In[ ]:= makeMask::usage =
  "makeMask[{r1,2,...,rn-1,n},N] makes an Nxn list {{0,0,1,...}...} with probability
  ri,i+1 of a crossover in interval {i,i+1}. makeMask[X,{i1,...},n]
  makes a mask, given the intervals in which effective
  recombinations occur; 1≤i≤n-1, and X=0 or 1";

```

```

In[ ]:= makeMask[r : {___?NumberQ}, n_Integer] :=
  Sort[DeleteCases[#, 0]] & /@ Transpose[
    MapThread[{#2 RandomInteger[BernoulliDistribution[#1], n]} &, {r, Range[Length[r]]}]]];

```

```

In[ ]:= makeMask[X0_, il : {___Integer}, n_Integer] := Module[{kk, skk},
  kk = Transpose[{1 + Prepend[il, 0], Append[il, n]}];
  skk = Transpose[{kk, ((-1)Range[Length[kk]]+X0 + 1) / 2}];
  Join@@(ConstantArray[#, 2] - #1[1, 1] + 1 & /@ skk)];

```

recInt

```
In[ ]:= recInt::usage = "recInt[{{LG,cM},...}] gives the recombination
      rates between each SNP, using Haldane's mapping function";

recInt[xx_List] := If[#[[1]] == 0,  $\frac{1}{2} \left( 1 - \text{Exp} \left[ -2 \frac{\#[[2]]}{100} \right] \right)$ ,  $\frac{1}{2}$ ] & /@ (Drop[xx, 1] - Drop[xx, -1]);
```

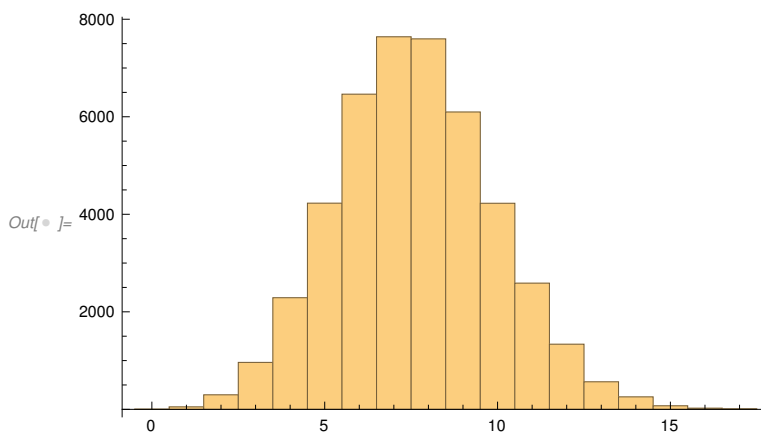
Checking recombination

This sets up recombination rates allowing for linkage (rr), or unlinked (ru). With linkage, there are an average of 7.63 crossovers, including 3.5 between the 8 linkage groups.

```
In[ ]:= rr = recInt[SNPposn];

ru = ConstantArray[ $\frac{1}{2}$ , nSNP - 1];

mmr = makeMask[rr, 2 nInds]; Histogram[Length /@ mmr]
Mean[Length /@ mmr] // N
```



Out[]:= 7.62915

neighbourF

```
In[ ]:= neighbourF::usage =
  "neighbourF[inds,locus,r,seed] lists the genotypes of each individual,
    and of the nearest individual to a random point distance
    r away. neighbourF[inds,locus,r,{j,k},seed] uses
    neighbour[inds,locus,r,{j,k},seed], and is more accurate; for each i
    of the inds, it returns a set of j neighbours in the form {i0, {i1, ..., ij}}";
```

```
ln[ ]:= neighbourF[inds_List, locus_, r_, seed_] :=  
  Transpose[{SNP[[inds, locus]], SNP[neighbour[inds, r, seed], locus]]};
```

```
ln[ ]:= neighbourF[inds_List, locus_, r_, {j_Integer, k_Integer}, seed_] :=  
  Transpose[{SNP[[inds, locus]], SNP[#, locus]] & /@ neighbour[inds, r, {j, k}, seed]]};
```

kids, kidsH

```
ln[ ]:= kids::usage =  
  "kids[{inds,snp},r,rec,seed] generates one child for each individual, using  
  a parent distance r away, with the specified recombination rate. If  
  the list of recombination rates rec is omitted, loci are unlinked."  
kids[{inds_, snp_}, r_, seed_, opts___Rule] := kids[{inds, snp}, r,  
  ConstantArray[ $\frac{1}{2}$ , Length[snp]-1], seed, opts];  
kids[{inds_, snp_}, r_, rec_, seed_, opts___Rule] := Module[{nn},  
  nn = If[r ≤ 0, neighbour[inds, seed], First /@ neighbour[inds, r, {1, 12}, seed]];  
  makeKids[Transpose[{inds, nn}], snp, rec, opts];
```

```
ln[ ]:= kidsH::usage =  
  "kidsH[{inds,snp},r,rec,seed] generates one child for each individual,  
  using a parent distance r away, with the specified recombination  
  rate, and calculates their MLH. If the list of recombination  
  rates rec is omitted, loci are unlinked. Stores results."  
kidsH[{inds_, snp_}, r_, rec_, seed_, opts___Rule] :=  
  kidsH[{inds, snp}, r, rec, seed, opts] = MLH[kids[{inds, snp}, r, rec, seed, opts]];  
kidsH[{inds_, snp_}, r_, seed_, opts___Rule] :=  
  kidsH[{inds, snp}, r, seed, opts] = MLH[kids[{inds, snp}, r, seed, opts]];
```

Offspring, Offspringrep, Offspringnear

```
ln[ ]:= f[a_, b_] := a[[b]]
```

```
ln[ ]:= dd = DistanceMatrix[XY];
```

```

In[ ]:= Offspring:: usage=
"For each focal plant, Offspring[dist, snp] finds nearest fathers at distances given by
Offspring[dist_,snp_] := Module[{ind,pair,pos,gens,off},
ind=Nearest[Delete[dd[[#]],#]→Delete[Range[Length[dd]],#],dist[[#],1]& /@ Range[Length[dd]];
pair=Transpose[{Range[Length[dd]],Flatten[ind]}];
pos=RandomInteger[{1,2},2*nInds*nSNP];
gens=Flatten[Transpose[snp[[#]]&/@pair,2];
off=Partition[Partition[MapThread[f][{gens,pos}],2],nSNP];
#/.{{0,0}→0,{0,1}→1,{1,0}→1,{1,1}→2,{-9,_}→-9,{_,-9}→-9,{-10,_}→-10,{_,-10}→-10} &/@of

```

```

In[ ]:= Offspring:: usage =
"For each focal plant, Offspring[dist, snp] finds its nearest father and
generates offspring for each pair. The genotype is assigned based
on Mendelian inheritance where missing values are retained ";
Offspringnear[snp_] := Module[{gnear, pairsnear, ppnear, posnear, offnear},
gnear = Nearest[Delete[XY, #] → Delete[Range[Length[dd]], #], XY[[#], 1] & /@
Range[Length[dd]];
pairsnear = Transpose[{Range[Length[dd]], Flatten[gnear]}];
ppnear = Flatten[Transpose[snp[[#]] & /@ pairsnear, 2];
posnear = RandomInteger[{1, 2}, 2 * nInds * nSNP];
offnear = Partition[Partition[MapThread[f][{ppnear, posnear}], 2], nSNP];
# /. {{0, 0} → 0, {0, 1} → 1, {1, 0} → 1, {1, 1} → 2,
{-9, _} → -9, {_, -9} → -9, {-10, _} → -10, {_, -10} → -10} & /@ offnear]

```

```

In[ ]:= Offspringrep::usage =
"For each focal plant, Offspring[dist, snp] finds nearest fathers at
distances given by dist and generates offspring for each pair. The
genotype is assigned based on Mendelian inheritance where missing
values are retained and returns the MLH value for each individual";
Offspringrep[dist_, snp_] := Module[{ind, pair, pos, gens, off, offnew},
ind = Nearest[Delete[dd[[#]], #] → Delete[Range[Length[dd]], #], dist[[#], 1] & /@
Range[Length[dd]];
pair = Transpose[{Range[Length[dd]], Flatten[ind]}];
pos = RandomInteger[{1, 2}, 2 * nInds * nSNP];
gens = Flatten[Transpose[snp[[#]] & /@ pair, 2];
off = Partition[Partition[MapThread[f][{gens, pos}], 2], nSNP];
offnew = # /. {{0, 0} → 0, {0, 1} → 1, {1, 0} → 1, {1, 1} → 2,
{-9, _} → -9, {_, -9} → -9, {-10, _} → -10, {_, -10} → -10} & /@ off;
MLH[
offnew]

```

impute

```

In[ ]:= impute::usage =
  "Input the snps for a given marker, returns the snps after imputing for the
    missing values based on the allele frequencies at respective markers";
impute[snp_] := Module[{zero, pos, no, new, final},
  zero =
    ((2 * Count[snp, 0]) + Count[snp, 1]) / (2 (Count[snp, 0] + Count[snp, 1] + Count[snp, 2])) // N;
  no = Count[snp, -9] + Count[snp, -10];
  new = Partition[RandomChoice[{zero, 1 - zero} -> {0, 1}, 2 * no], 2] /.
    {{0, 0} -> 0, {1, 0} -> 1, {0, 1} -> 1, {1, 1} -> 2};
  pos = Join[Flatten[Position[snp, -9]], Flatten[Position[snp, -10]]];
  final = ReplacePart[snp, Normal[AssociationThread[pos -> new]]];
  final]

```

indexperyear, coreindexperyear, locperyear, snpperyear, locsmall, snpsmall

indexperyear and coreindexperyear contain the indices of individuals present in each year for the entire dataset and for the core region used in the simulation

```

In[ ]:= indexperyear = planolesIndsT[##] & /@ Range[2009, 2019];

```

```

In[ ]:= coreindexperyear = coreInds[##] & /@ Range[2009, 2019];

```

locperyear and snpperyear contain the locations and genotypes of the individuals present from 2009 to 2019. imputeddata contains the imputed genotypes.

```

In[ ]:= locperyear = XY[##] & /@ indexperyear;

```

```

In[ ]:= snpperyear = SNP[##] & /@ indexperyear;

```

```

In[ ]:= imputeddata = Table[Transpose[impute[##]& /@ Transpose[snpperyear[[i]]], {i, Length[snpperyear]}]

```

```

In[ ]:= snpperyearprocess = imputeddata /. {0 -> 0, 1 -> 0.5, 2 -> 1};

```

locsmall and snpsmall contain the locations and genotypes of the individuals present from 2009 to 2019 from the area defined as core region. imputedsmall contains imputed genotypes

```

In[ ]:= locsmall = XY[##] & /@ coreindexperyear;

```

```

In[ ]:= snpsmall = SNP[##] & /@ coreindexperyear;

```

```
in[ ]:= imputedsmall= Table[Transpose[impute[##]&/@Transpose[snpssmall[[i]]],{i,Length[snpssmall]}];
```

```
in[ ]:= snpperyearprocesssmall = imputedsmall /. {0 → 0, 1 → 0.5, 2 → 1};
```

hom, ibd, ibddata, fun, FvsR, rvsF

Here IBD refers to isolation by distance

```
hom::usage = "if i and j are multilocus
  genotypes, then hom[i,j] calculates the pairwise homozygosity";
hom[i_, j_] := Mean[i * j + ((1 - i) * (1 - j))] // N
(*hom[i_, j_] := Mean[DeleteCases[i * j + ((1 - i) * (1 - j)), Indeterminate]]//
  N* Use this definition to calculate pairwise homozygosity without imputing.)
```

```
in[ ]:= ibd::usage =
  "Given the geographic and genetic distances,
  ibd[distmat,genmat] calculates genetic distance associated with
  geographic distances and pool them into distance classes of 20m";
ibd[distmat_, genmat_] := Module[{d, gend, list, sortedlist, z, newlist},
  d = Flatten[Table[Diagonal[distmat, k], {k, 0, Length[distmat] - 1}]];
  gend = Flatten[Table[Diagonal[genmat, k], {k, 0, Length[genmat] - 1}]];
  list = Transpose[{d, gend}];
  sortedlist = SortBy[N@*First]@list;
  newlist = Select[sortedlist, ##[[1]] > 0. &];
  z = Flatten[DeleteCases[BinLists[newlist, 20, 2], {}, 2], 1];
  Mean[##] & /@ z]
```

```
in[ ]:= ibddata::usage =
  "ibd[l,gen] calculates the distance matrix from list of locations l and genetic
  distance as pairwise homozygosity from gen and calculates ibd";
ibddata[l_, gen_] := Module[{d, proc, t},
  d = DistanceMatrix[l];
  t = Table[Table[hom[gen[[i]], gen[[j]]], {j, 1, Length[l]}], {i, 1, Length[l]}];
  ibd[d, t]]
```



```

In[ ]:= fun::usage = " find genetic distance, geo
          distance and isolation by distance for each replicate";
fun[p_, loc_] := Module[{tsim, dist},
  tsim = Table[Table[hom[#[[i]], #[[j]]], {j, 1, Length[loc]}], {i, 1, Length[loc]}] & /@ p;
  dist = DistanceMatrix[loc];
  ibd[dist, #[[i]] & /@ tsim]

```

```

In[ ]:= FvsR::usage =
  "FvsR[gen , list, loc] calculates how Fst changes with distance. This is
  calculated for a subsampled list of individual given their
  genotypes (gen) and location at the last generation (loc). ";
FvsR[gen_List, l_List, loc_List] := Module[{processed, simavg, p, q, avsim, fstvsr},
  processed = #[[i]] /. {0 → 0, 1 → 0.5, 2 → 1} & /@ gen;
  simavg = fun[processed[[All, 1]], loc[[1]]];
  p = alleleFreq[#[[i]]] & /@ gen // N;
  q = 1 - # & /@ p // N;
  avsim = Mean[#[[i]]] & /@ (p^2 + q^2);
  Table[Transpose[{simavg[[i, All, 1]], ((simavg[[i, All, 2]] - avsim[[i]]) / (1 - avsim[[i]]))}],
    {i, Length[gen]}]]

```

```

In[ ]:= rvsF::usage = "Calculates IBD from pedigree";
rvsF[g_, f_, t_] :=
  Module[{distmatrix, distance, gendist, list, sortedlist, newlist, x},
    distmatrix = DistanceMatrix[g[[t, All, 3]]];
    (* Calculate pairwise distances
       between every individual to every other individual*)
    distance = Flatten[Table[Diagonal[distmatrix, k], {k, 0, Length[distmatrix] - 1}]];
    (*Extracting upper triangular elements to choose every unique pair*)
    gendist = Flatten[Table[Diagonal[f[[t]], k], {k, 0, Length[f[[t]] - 1}]];
    (*Extract the corresponding identity values*)
    list = Transpose[{distance, gendist}];
    sortedlist = SortBy[N@*First]@list;
    (*Sort the list according to the geographich distances between individuals*)
    newlist = Select[sortedlist, #[[1]] > 0. &];
    x = Flatten[DeleteCases[BinLists[newlist, 20, 2], {}, 2], 1];
    (*Bin the lists according to integer intervals*)
    Mean /@ x]

```

makeN

```
In[ ] := makeN::usage = "makeN[k,R1,R2] makes a list of
      coordinates of individuals, uniformly on {{0,R1},{0,R2}}";
makeN[k_Integer, R1_, R2_] := Transpose[{RandomReal[{0, R1}, k], RandomReal[{0, R2}, k]}];
```

seeddisp, polldisp:

seeddisp and polldisp contains the 1463 seed and pollen dispersal distances.

```
In[ ] := seeddisp = First /@ disp;
```

```
In[ ] := polldisp = Last /@ disp;
```

LocData

listoflocs contains the locations of the individuals for 10 timepoints that would be used in the simulation

```
In[ ] := listofloc = DeleteDuplicates[##] & /@ locsmall;
```

```
In[ ] := listoflocs = {listofloc[[1]], listofloc[[2]], listofloc[[3]],
      listofloc[[4]], listofloc[[5]], listofloc[[6]], listofloc[[7]], listofloc[[8]],
      DeleteDuplicates[Join[listofloc[[9]], listofloc[[10]]], listofloc[[11]]};
```

```
In[ ] := LocData::usage =
      "LocData[t_] takes a random sample of k locations specfied from the data from
      year corresponding to index t. If k > number of individuals, all the
      individuals from the year are included, the remaining k-n individuals
      are randomly sampled again and displaced to a point randomly 3m apart.";
```

```
In[ ] := LocData[k_, t_] := Module[{n = Length[listoflocs[[t]]},
      If[k ≤ n, RandomSample[listoflocs[[t]], k],
      Join[listoflocs[[t]], RandomChoice[listoflocs[[t]], k - n] + Table[randomR[3], k - n]]]
```

findmom, finddad, findparscircle

```
In[ ] := findmom::usage =
      "findmom[kids,pars, {j,k},seed] chooses k points on a circle at a distance
      seed[[i]] for kid i, and takes the j individuals nearest the circle.";
```

```

In[ ]:= findmom[kids:{{_,_}...}, pars_List, {j_Integer, k_Integer}, seed_List] :=
  findmom[kids, pars, {j, k}, seed] =
    findmom[#[[1]], pars, #[[2]], {j, k}] & /@ Transpose[{kids, seed}];
findmom[kidind:{{_,_}}, pars_List, r_, {j_Integer, k_Integer}] :=
  Module[{ $\theta = 2 \pi$  (Random[] + Range[0, 1 - 1/k, 1/k]), y, rnf, dl},
    y = Transpose[r {Cos[ $\theta$ ], Sin[ $\theta$ ]}];
    rnf = Flatten[nearestF[pars][[kidind + #]] & /@ y];
    dl = Abs[Norm[pars[[#]] - kidind] - r] & /@ rnf;
    Last /@ Take[Sort[Transpose[{dl, rnf}]], 1];

```

```

In[ ]:= findddad::usage =
  "findddad[moms,pars, {j,k},poll] chooses k points on a circle at a distance
  poll[[i]] for mom i, and takes the j individuals nearest the circle. ";

```

```

In[ ]:= findddad[moms_List, pars_List, {j_Integer, k_Integer}, poll_List] :=
  findddad[moms, pars, {j, k}, poll] =
    findddad[#[[1]], pars, #[[2]], {j, k}] & /@ Transpose[{moms, poll}];
findddad[mom_Integer, pars_List, r_, {j_Integer, k_Integer}] :=
  Module[{ $\theta = 2 \pi$  (Random[] + Range[0, 1 - 1/k, 1/k]), y, rnf, dl},
    y = Transpose[r {Cos[ $\theta$ ], Sin[ $\theta$ ]}];
    rnf = DeleteCases[#, mom][[1]] & /@ nearestF[pars][[pars[[mom]] + #]] & /@ y, 2];
    dl = Abs[Norm[pars[[#]] - pars[[mom]]] - r] & /@ rnf;
    Last /@ Take[Sort[Transpose[{dl, rnf}]], 1];

```

```

In[ ]:= findparscircle[kidsX_List, pars_List, seed_List, poll_List] := Module[{moms, dad},
  moms = Flatten[findmom[kidsX, pars, {1, 6}, seed]];
  dad = Flatten[findddad[moms, pars, {1, 6}, poll]];
  Transpose[{moms, dad}];

```

iterFReal

```
iterFReal::usage =
  "iterF[{{par1,par2, ...},{x1,...},{y1,...},{s1, s2,...},{p1,p2, ...}}] iterates the
    population back through one generation, with seed dispersal values
    s, pollen dispersal values p. For individuals with locations {y1,
    y2,...}, iterF finds the parents from the previous generation with
    locations given by {x1, x2,...}. Returns a list giving the parents
    (par) and location of all individuals that make up that generation.
  iterF[{{par1,par2, ...},{i1, i2, ...},{x1,...},{y1,...},{s1,
    s2,...},{p1,p2, ...}}] does the same as above, but
    additionally follows the individual indexes i1, i2,... .";
```

```
In[ ]:= iterFReal[g_List, kidsX_List, s_List, p_List] := Module[{pars},
  pars = findparscircle[kidsX, Last@g, s, p];
  {pars, kidsX}];
```

```
iterFReal[g_List, kidsNames_List, kidsX_List, s_List, p_List] := Module[{pars},
  pars = findparscircle[kidsX, Last /@ g, s, p];
  Transpose[{kidsNames, g[[#, 1]] & /@ pars, kidsX}]];
```

makeGenealogyReal, makeGenealogyRealindexed

```
In[ ]:= makeGenealogyReal::usage =
  "makeGenealogyReal[{x1,...},t] iterates a population forwards in time,
    generating a set of individuals through a 2D habitat. Population
    size stays fixed and is distributed conditioned on the data: in
    each generation, n kids are scattered over the space; two parents
    are chosen based on the seed and pollen dispersal distances.
  makeGenealogyReal[{x1,...},t, lengthX, lengthY] iterates a
    pouplation forward in time through lengthX* lengthY
    habitat where individuals have a unifrom density. ";
```

time keeps track of the generation being simulated.

```
In[ ]:= time = 0; Dynamic[time]
```

```
Out[ ]:= time
```

```

In[ ]:= makeGenealogyReal[x0_List, tm_Integer] :=
Module[{n = Length[x0], g0, i, seeds, polls},
  time = 0;
  i = 0;
  seeds = RandomVariate[HistogramDistribution[seeddisp], {tm + 1, n}];
  polls = RandomVariate[HistogramDistribution[polldisp], {tm + 1, n}];
  g0 = {ConstantArray[{}, n], x0};
  NestList[(time += 1;
    i++;
    iterFReal[#, LocData[n, Mod[i, 10] + 1], seeds[[i]], polls[[i]]] &, g0, tm]];

```

```

In[ ]:= makeGenealogyReal[x0_List, tm_Integer, Rx_Real, Ry_Real] :=
Module[{n = Length[x0], g0, i, seeds, polls},
  time = 0;
  i = 0;
  seeds = RandomVariate[HistogramDistribution[seeddisp], {tm + 1, n}];
  polls = RandomVariate[HistogramDistribution[polldisp], {tm + 1, n}];
  g0 = {ConstantArray[{}, n], x0};
  NestList[(time += 1;
    i++;
    iterFReal[#, makeN[n, Rx, Ry], seeds[[i]], polls[[i]]] &, g0, tm]];

```

```

makeGenealogyRealindexed::usage =
  "makeGenealogyRealindexed[{x1,...},t] is same as makeGenealogyReal,
    but additionally index the individuals.";

```

```

makeGenealogyRealindexed[x0_List, tm_Integer] :=
Module[{n = Length[x0], g0, i, seeds, polls},
  time = 0;
  i = 0;
  seeds = RandomVariate[HistogramDistribution[seeddisp], {tm + 1, n}];
  polls = RandomVariate[HistogramDistribution[polldisp], {tm + 1, n}];
  g0 = Transpose[{Range[n], ConstantArray[{}, n], x0}];
  NestList[(time += 1;
    i++;
    iterFRealindexed[#, Max[First /@ #] + Range[n],
      LocData[n, Mod[i, 10] + 1], seeds[[i]], polls[[i]]] &, g0, tm]];

```

getKids

```
In[ ]:= getKids::usage =
  "getKids[{p1,p2},...],snp] makes a set of offspring from the given pairs given
    the snps for all individuals in that generation in the haplotype format.
  getKids[{snp_p1,snp_p2},{X1,X2}] generates an offspring from the two parents p1
    and p2 starting with one or other parent chromosome according to Xi=0
    or 1. snp_p1 is in the haplotype format containing either 0 or 1. ";
```

```
In[ ]:= getKids[{d1 : {{{0 | 1} ...}, {(0 | 1) ...}}, d2 : {{{0 | 1} ...}, {(0 | 1) ...}}}, xx_] :=
  {(d1[[1]] xx[[1]] + d1[[2]] (1 - xx[[1]]), (d2[[1]] xx[[2]] + d2[[2]] (1 - xx[[2]]))};
getKids[pars : {_Integer, _Integer} ...], snp : {{{{0 | 1} ...}, {(0 | 1) ...}} ...}] :=
  Module[{xx, ninds = Length[pars]},
    xx = RandomInteger[{0, 1}, {ninds, 2, 91}];
    MapThread[getKids[{snp[[#1[[1]]], snp[[#1[[2]]]], #2] &, {pars, xx}]]];
```

Reps

```
In[ ]:= Reps::usage =
  "Find the genotype of all individuals for all generations and returns the
    genotypes of at the end of the simulation ";
```

```
In[ ]:= Reps[snp_, par_] := Module[{i, gen},
  i = 0;
  gen = Nest[(i++; getKids[par[[i]], #]) &, snp, 1000];
  #[[1]] + #[[2]] & /@ gen];
```

pedM

In[]:=

```
pedM::usage =
  "pedM[{{mum_j,{grandmum_j,granddad_j},x_i},...},{{i,{mum_i,dad_i},x_i},...}] returns
    the pedigree matrix, as a sparse array, with entries giving
    the probability that a gene in the second population came
    from a parent gene in the first. Each row has two entries of
    1/2 unless there was selfing, in which case there is a single
    entry of 1. The two arguments must be successive generations.
    pedM[{p_0,p_1,...,p_T}] returns a list of pedigree matrices {P_{T,T-1},...,P_{1,0}}";

pedM[p0_List, p1_List] := Module[{i, n0 = Length[p0], n1 = Length[p1], pos},
  pos = Flatten /@ Map[Position[p0[All, 1], #] &, p1[All, 2], {2}];
  SparseArray[Flatten[Table[If[pos[[i, 1]] == pos[[i, 2]], {{i, pos[[i, 1]]} → 1},
    {{i, pos[[i, 1]]} → 1/2, {i, pos[[i, 2]]} → 1/2}], {i, n1}], 1], {n1, n0}]];

pedM[p_List] := MapThread[pedM, {Drop[p, -1], Drop[p, 1]}];
```

newF

```
newF::usage =
  "newF[F_t,P] generates the identity by descent in generation t+1, given a
    sparse pedigree matrix P, which has dimension {n_{t+1},n_{t+1}}";

newF[F : (_SparseArray | _List), P_SparseArray] :=
  Module[{n0 = Last[Dimensions[P]], fnew, nonzero, ind, indsel},
    fnew = P.SparseArray[F +  $\frac{1}{2}$  (IdentityMatrix[Length[P]] - DiagonalMatrix[Diagonal[F]])].
      Transpose[P];
    nonzero = Flatten[P["NonzeroPositions"]];
    (* Find the positions of the parents *)
    ind = nonzero[[# ;; ;; 2]] & /@ {1, 2}
    (* Groups elements at odd and even postions as two lists*)
    indsel = Extract[F, Partition[ind[[2]], 2]];
    (* Partition to get the index to choose from F*)
    fnew - DiagonalMatrix[Diagonal[fnew]] + DiagonalMatrix[indsel]
```

makeF

```
makeF::usage =
  "makeF[{P1,0,P2,1,...}] takes a list of pedigree matrices, and calculates
    Fi,j for each generation. Beware: may use a lot of memory.";
makeF[P_List] := Module[{n0 = Last[Dimensions[First[P]]]},
  FoldList[newF, N[SparseArray[{}, {n0, n0}]], P];
```

pedibd

```
pedibd::usage="pedibd[{{i1,i2,...},{par1,par2,...},{x1,x2,...}},...] returns the pedigree
calculates isolation by distance using rvsF.";
```

```
In[ ]:= pedibd[g_]:=Module[{pml,ft,ftt,ibdped,ibdnorm},
  pml=pedM[g];
  ft=makeF[pml];
  ftt=Mean[Mean[ $\#$ ]]&/@ft;
  ibdped=rvsF[g,ft,-1];
  {ftt,ibdped}]
```

dispcalc

```
In[ ]:= dispcalc::usage =
  "Given the spatial pedigree gg, disp[gg,i] calculates seed and pollen
    dispersal distances in generation i";
```

```
In[ ]:= dispcalc[gg_,i_]:=Module[{parind=Transpose[gg[[i,1]],parloc,seed,poll},
  parloc=gg[[i-1,2]] $\#$ &/@parind;
  seed=Norm[ $\#$ ]/@ (parloc[[1]]-gg[[i,2]]);
  poll=Norm[ $\#$ ]/@ Flatten[Differences[parloc],1];
  {seed,poll}]
```

g2

```
In[ ]:= g2::usage =
  "g2[{m,v}] calculates g2 as Var[f]/(1-Mean[f])2;
    m, v are the mean and variance of F from the pedigree.";
g2[{m_, v_}] := v / (1 - m)2;
```