Haplotype-based approaches to cluster individuals and infer ancestry proportions

Garrett Hellenthal
g.hellenthal@ucl.ac.uk

University College London

Population genetics summer course, Denmark
August 22, 2024

Introduction

- this lecture/practical will cover chromosome painting using `CHROMOPAINTER`, using its output in three additional programs:

  1. `fineSTRUCTURE` – cluster individuals based on similar haplotype-sharing patterns

  2. `GLOBETROTTER, SOURCEFIND` – infer individuals' haplotype make-up as a mixture of that of other "ancestry surrogate" groups

Outline

Clustering individuals: CHROMOPAINTER + fineSTRUCTURE

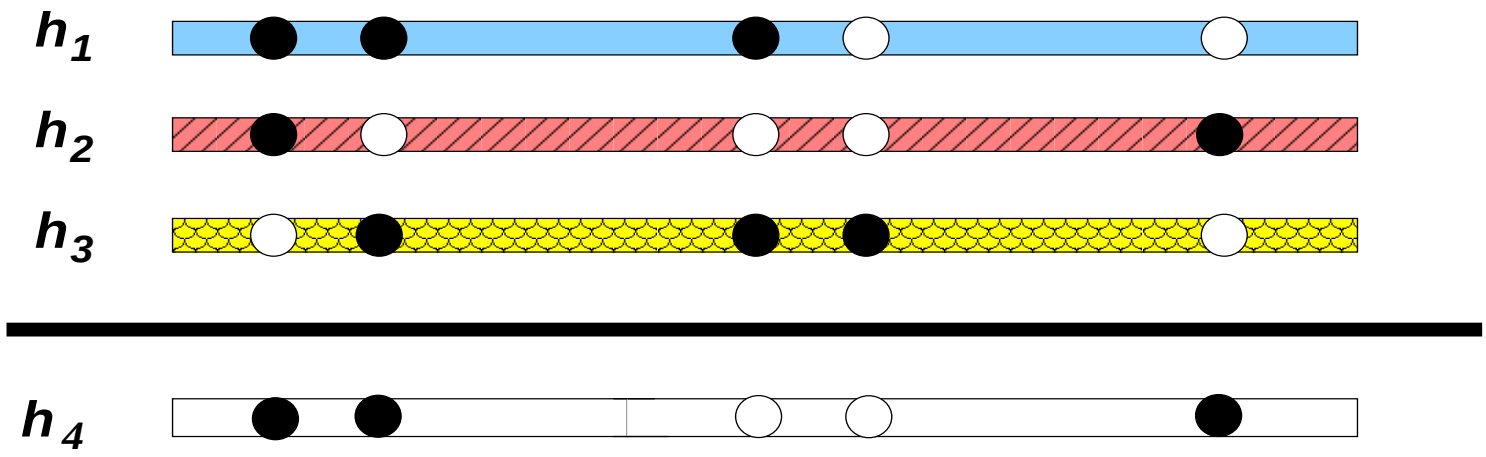
Inferring ancestry proportions: CHROMOPAINTER + GLOBETROTTER/SOURCEFIND

Outline

Clustering individuals: CHROMOPAINTER + fineSTRUCTURE

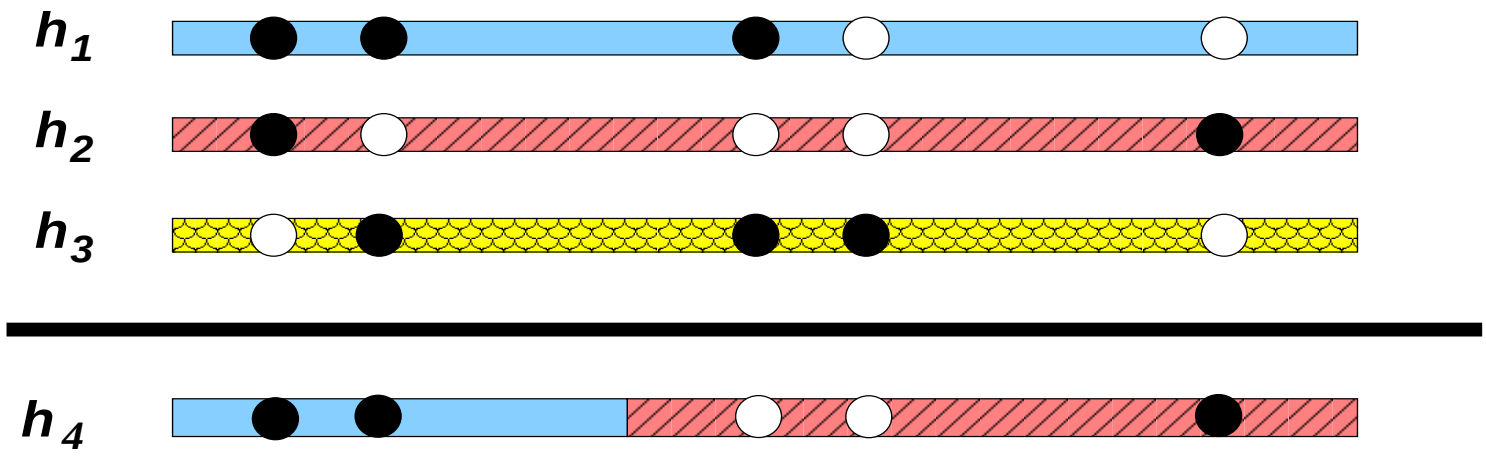
Inferring ancestry proportions: CHROMOPAINTER + GLOBETROTTER/SOURCEFIND

Incorporating haplotype information: chromosome painting (Lawson et al 2012, *PLoS Genet* **8**:e1002453)



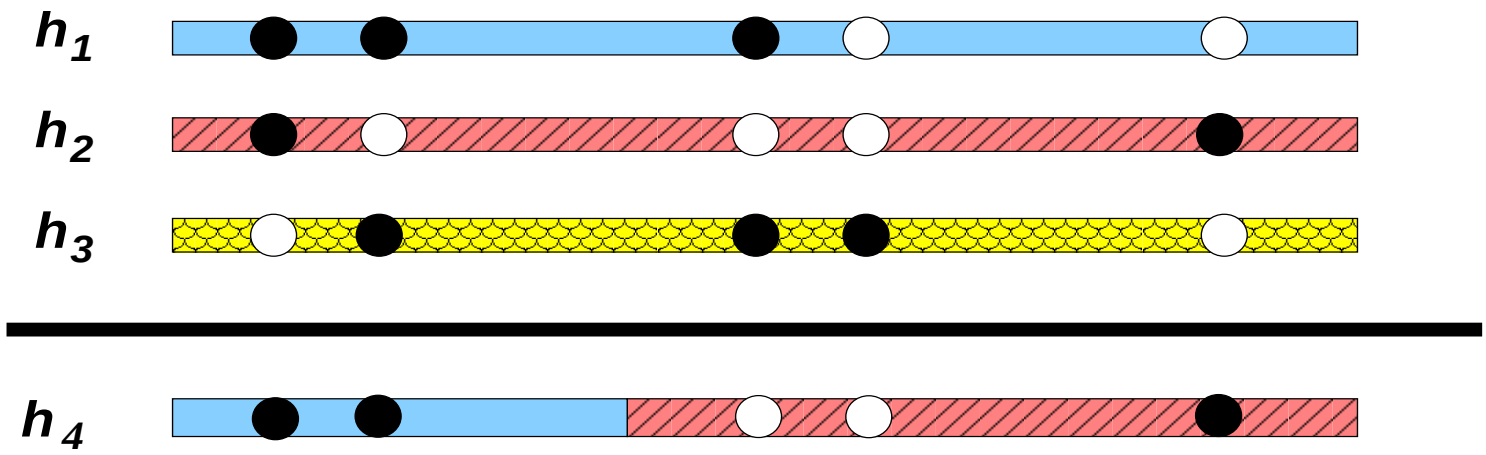
- use some sampled chromosomes (e.g. h_1, h_2, h_3) as "donors"
- match (or "paint") other chromosomes (e.g. h_4) to donors' DNA
- $\longrightarrow$ cluster based on who shares many **blocks of SNPs** rather than who shares similar **SNP frequencies**

Incorporating haplotype information: chromosome painting (Lawson et al 2012, *PLoS Genet* **8**:e1002453)



- use some sampled chromosomes (e.g. h_1, h_2, h_3) as "donors"
- match (or "paint") other chromosomes (e.g. h_4) to donors' DNA
- $\longrightarrow$ cluster based on who shares many **blocks of SNPs** rather than who shares similar **SNP frequencies**

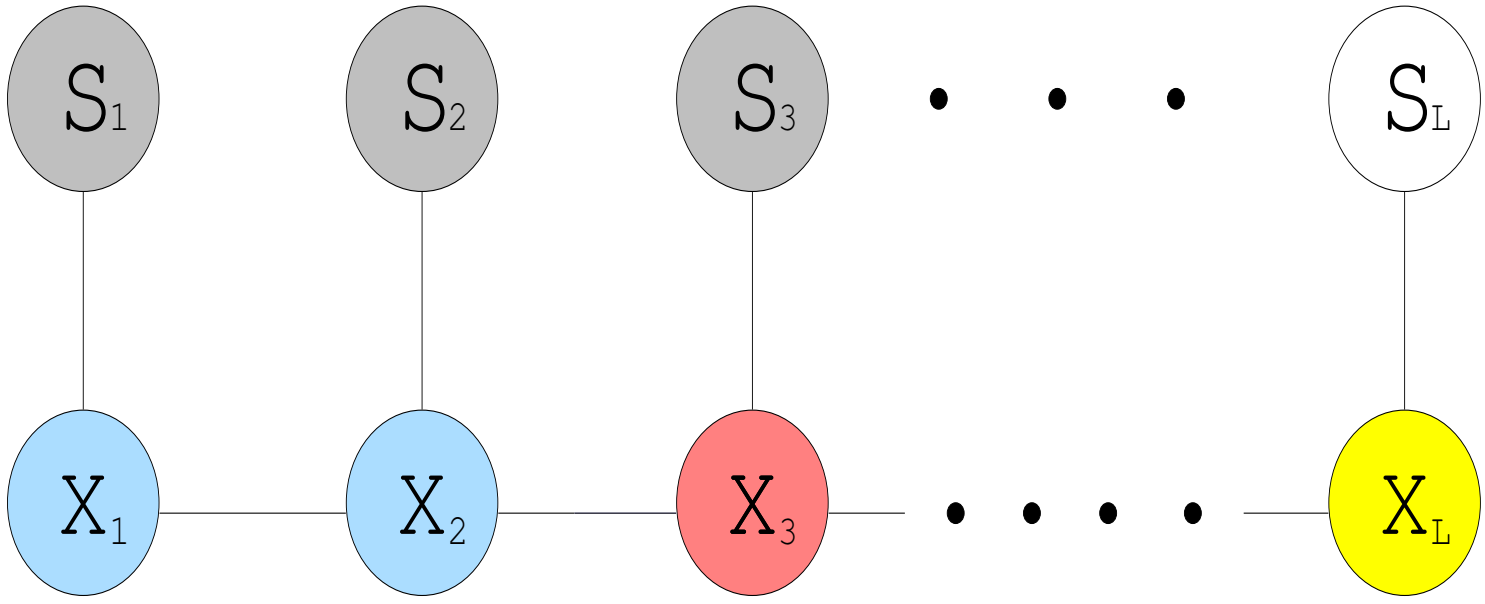
Incorporating haplotype information: chromosome painting (Lawson et al 2012, *PLoS Genet* **8**:e1002453)



- use some sampled chromosomes (e.g. h_1, h_2, h_3) as "donors"
- match (or "paint") other chromosomes (e.g. h_4) to donors' DNA
- $\longrightarrow$ cluster based on who shares many **blocks of SNPs** rather than who shares similar **SNP frequencies**
- can do "painting" using many approaches, e.g:
  1. *CHROMOPAINTER* (Lawson et al 2012, *PLoS Genet* **8**:e1002453)
  2. *HAPMIX* (Price et al 2009, *PLoS Genet* **5**:e1000519)
  3. *RFMIX* (Maples et al 2013, *AJHG* **93**:278)
  4. *MULTIMIX* (Churchhouse & Marchini 2013, *Genet Epidemiol* **37**:1)

CHROMOPAINTER – Hidden Markov Model (HMM)

- S_l = (observed state) SNP data at locus l of haploid i
- X_l = (hidden state) donor haploid 1,...,d copied at SNP l



- each donor haploid is depicted with a unique color here
- assume “switches” in donor copied occur as a Poisson process of rate $\propto g_l$, the cM distance between SNPs l and $l+1$

Copying Model (HMM) (Lawson et al 2012, *PLoS Genet* **8**:e1002453)

(based on Li & Stephens 2003, *Genetics***165**:2213)

X_l = (unknown) "donor" haplotype copied at SNP l

S_l = observed data at SNP l

$$\Pr(X_{l+1} = d \mid X_l = d') = \begin{cases} \exp(-g_l N_e) + \Big(1 - \exp(-g_l N_e)\Big) q_d & \text{if } d = d' \\ \Big(1 - \exp(-g_l N_e)\Big) q_d & \text{otherwise} \end{cases}$$

$$\Pr(S_{l+1} = s \mid S_{X_{l+1}} = s_d) = \begin{cases} 1 - \theta & s = s_d \\ \theta & s \neq s_d \end{cases}$$

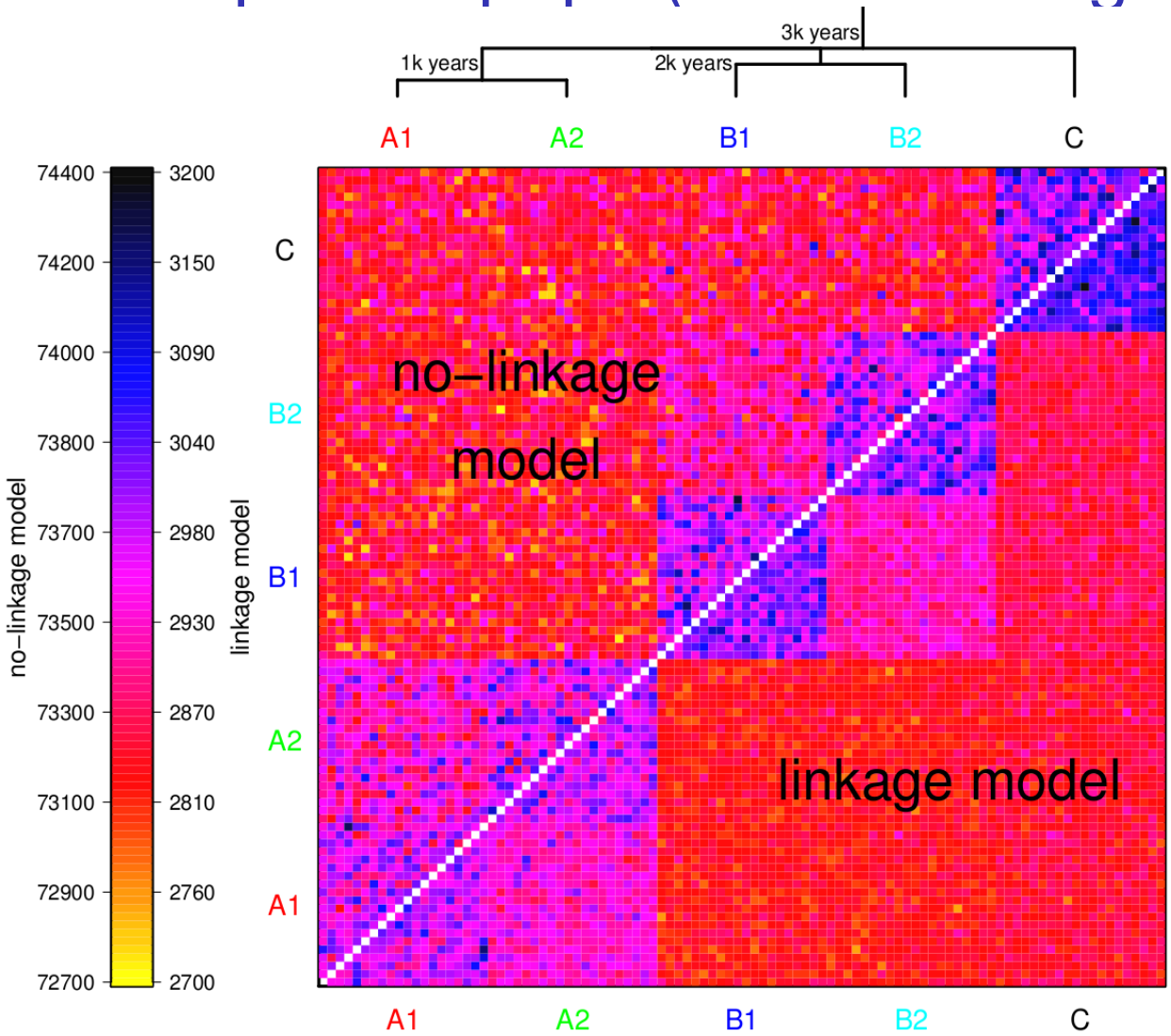
- where g_l = genet distance btwn SNPs l and $l + 1$, N_e = "switch param", θ = mutation (emission) rate, $q_d = \Pr(\text{copy } d)$
- estimates proportion copied from donor d conditional on data
- can sample X_l along genome

Summarizing *CHROMOPAINTER* painting: Heatmaps

Input for running *fineSTRUCTURE*:

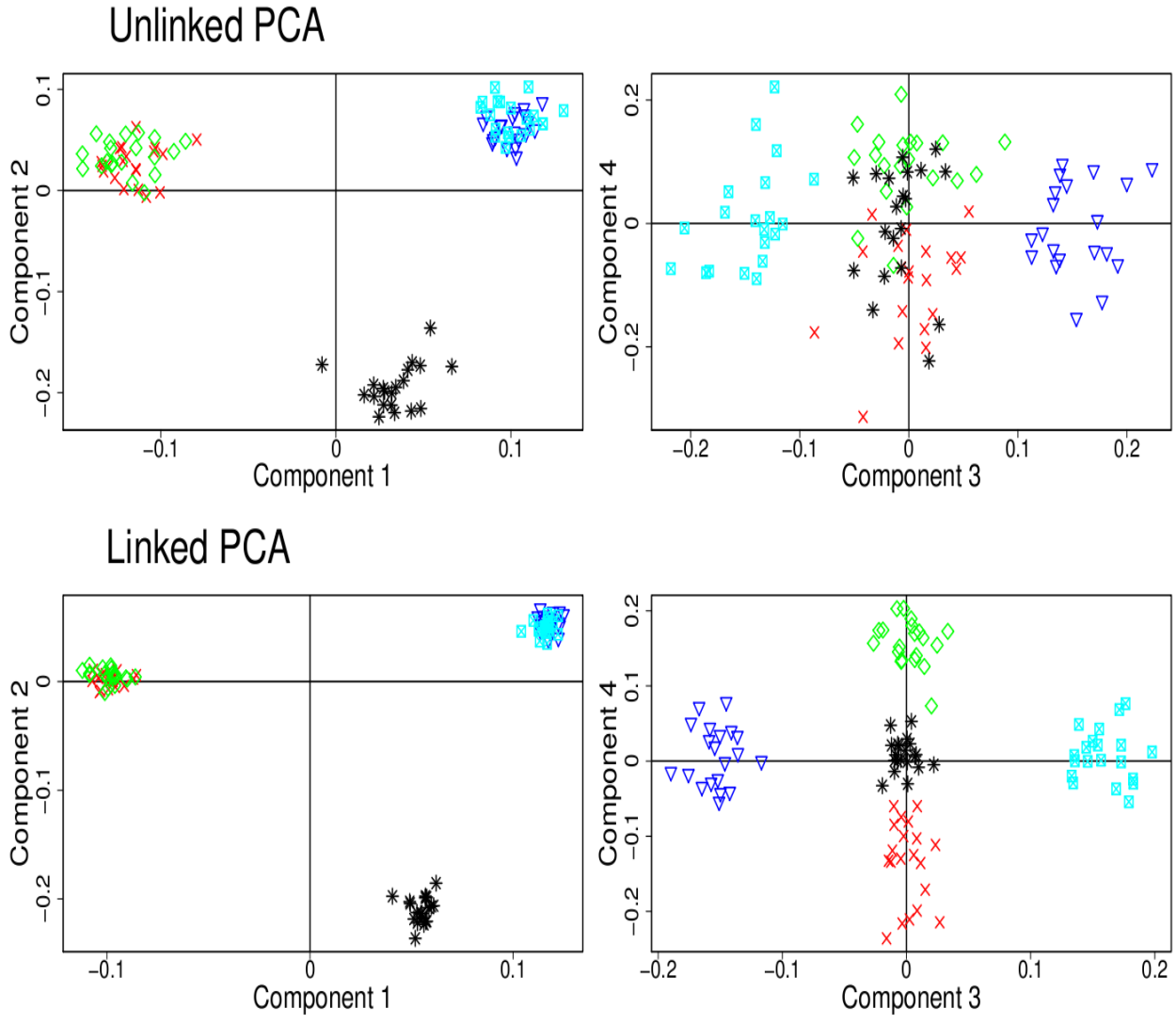
- run *CHROMOPAINTER* allowing each individual i to copy from every other individual $j \neq i$
- calculate y_{ij} – expected number of "chunks" ind i copies from ind j
  - if you assume each SNP is a "chunk" (i.e. "no-linkage" model) you capture information equivalent to PCA of genetic data
  - considerably more power if you use haplotype-based model

Simulated Example – 5 pops (150 5-Mb regions)



- **Heatmap:** each square is the number of DNA segments that each row (recipient) copies from (i.e. is painted by) each column (donor)
- **upper left triangle:** ignoring haplotypes
  lower right triangle: using haplotypes

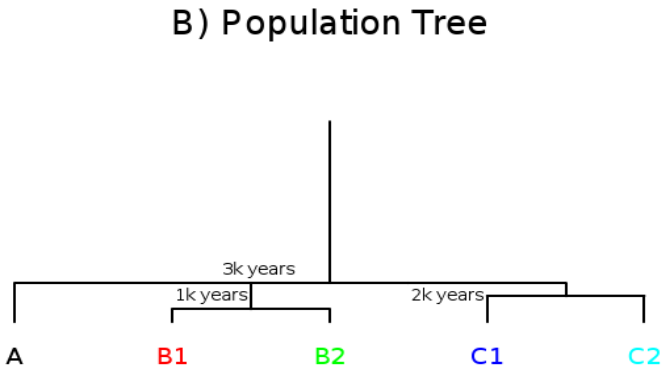
Simulated Example – 5 pops (PCA of heatmap)



top row: ignoring haplotypes (equivalent to regular PCA)

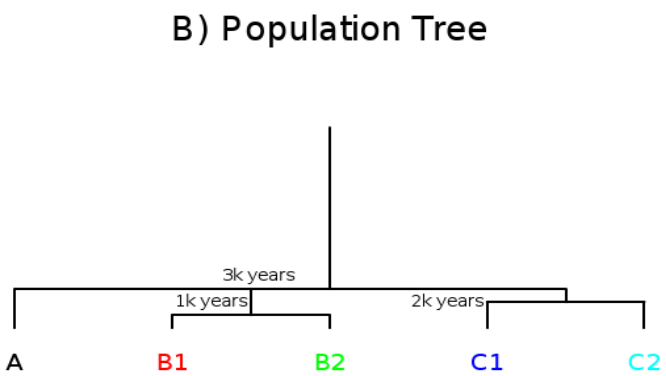
bottom row: using haplotypes

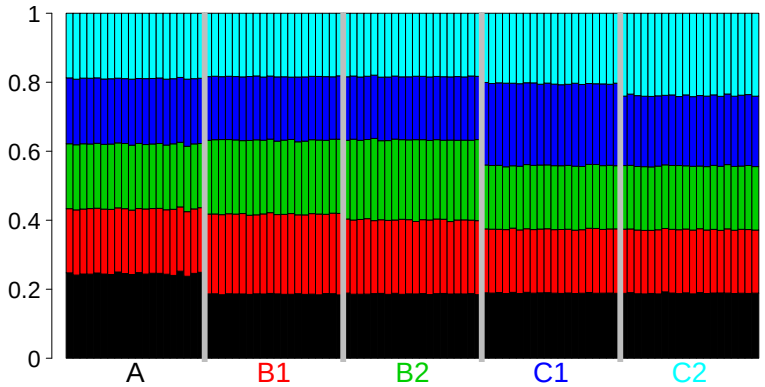
fineSTRUCTURE: cluster using *CHROMOPAINTER* paintings



(Lawson et al 2012, *PLoS Genet* **8**:e1002453)

fineSTRUCTURE: cluster using *CHROMOPAINTER* paintings

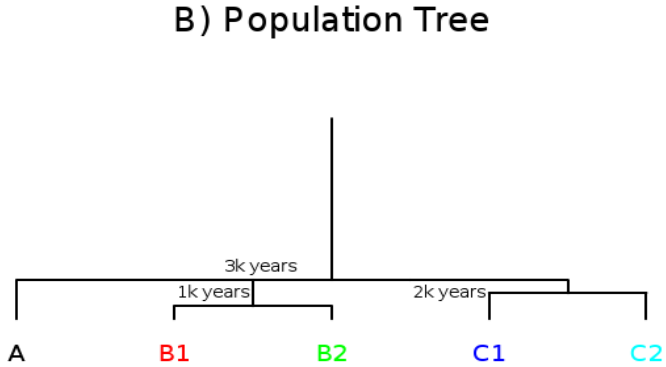


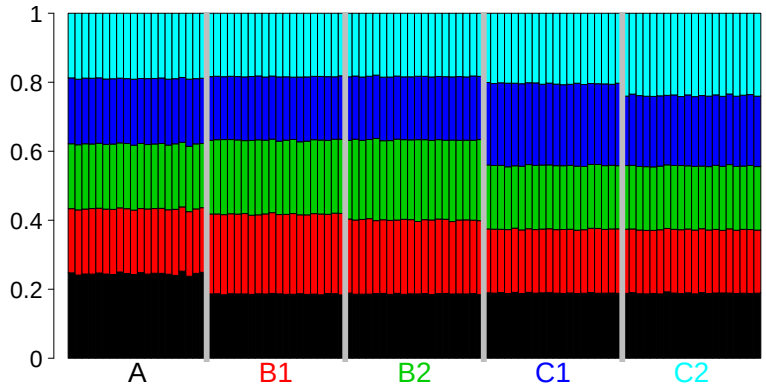


- paint each individual *i* using all other inds as “donors”
- calculate y_{ij} – number of DNA segments of ind *i* painted by ind *j*

(Lawson et al 2012, *PLoS Genet* **8**:e1002453)

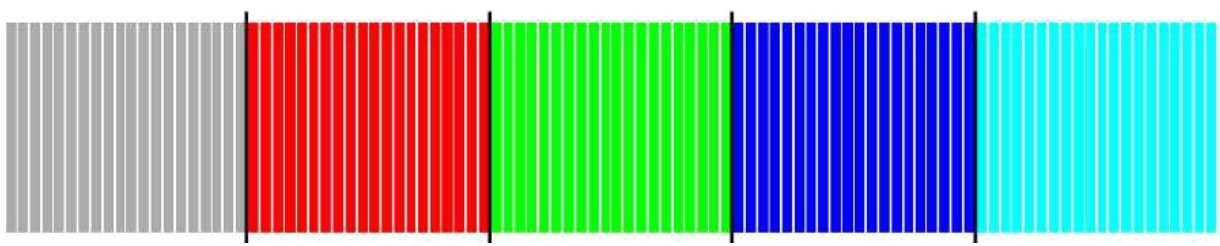
fineSTRUCTURE: cluster using *CHROMOPAINTER* paintings





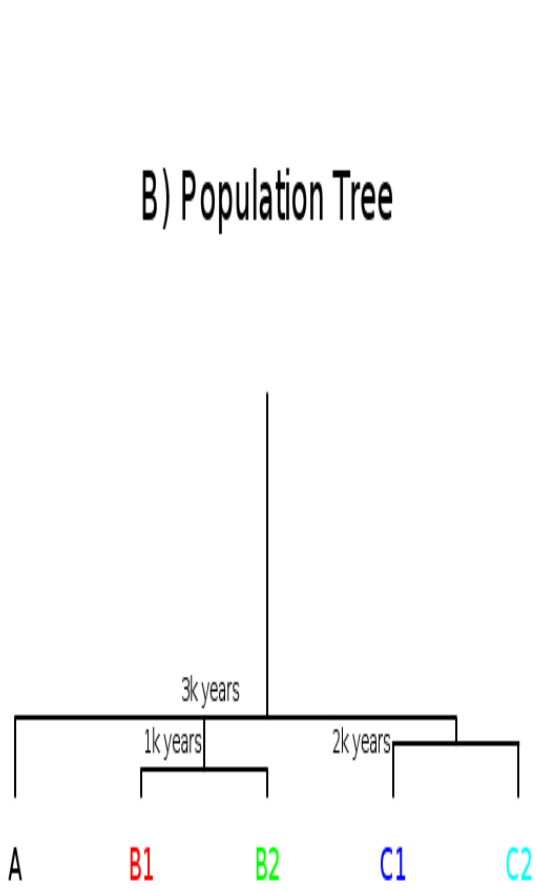
- paint each individual *i* using all other inds as "donors"
- calculate y_{ij} – number of DNA segments of ind *i* painted by ind *j*
- cluster individuals who have similar painting patterns (MCMC):
  1. start with random assignment of inds to clusters 1,...,*K*
  2. $(y_{i1}, ..., y_{iK}) \sim \text{Mult}(P_{A1}, ..., P_{AK})$ for ind *i* assigned to cluster *A*
  3. infer P_{Ak} by using number of segments by which inds in cluster *A* are painted by inds in cluster *k*
  4. if $\Pr(y_{i1}, ..., y_{iK})$ low $\longrightarrow$ move ind *i* to different cluster

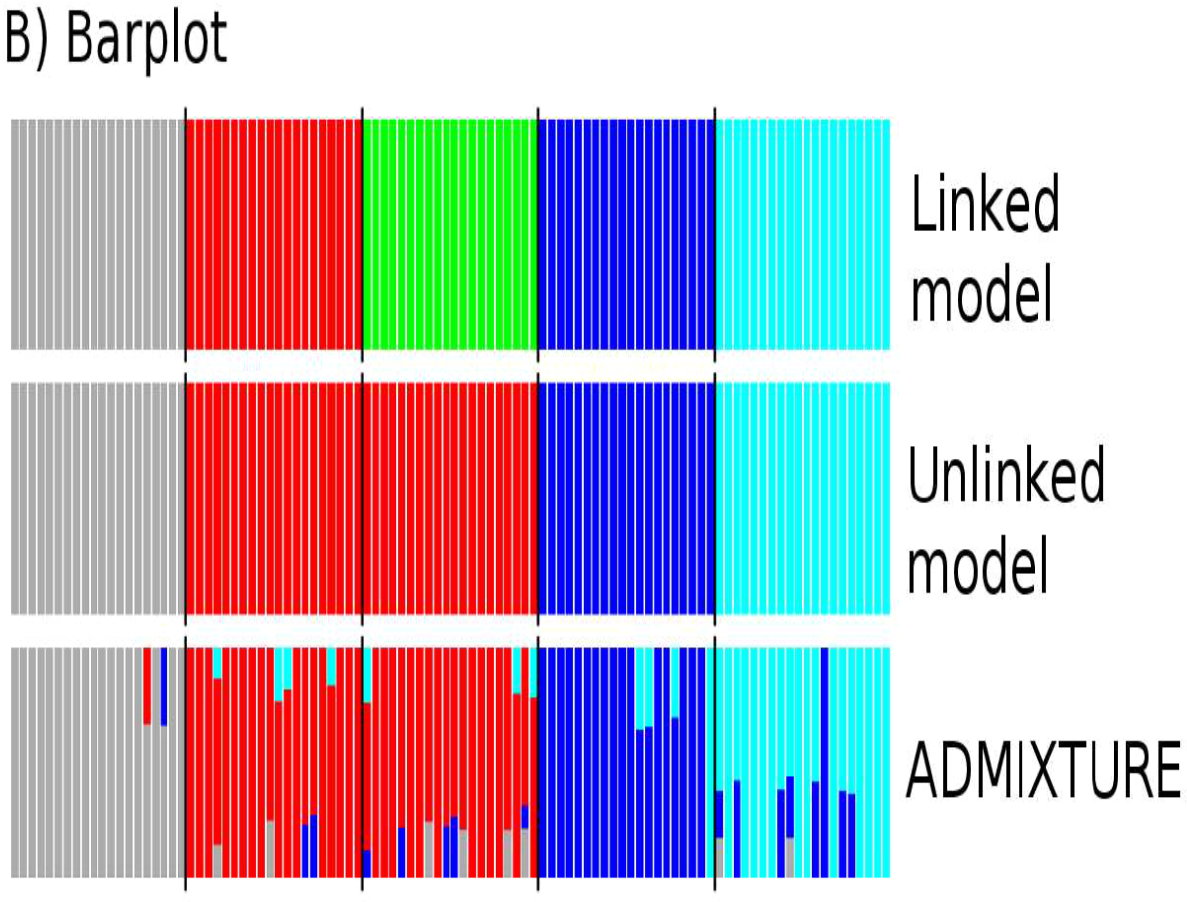
  (likelihood shrunk by factor *c* to account for non-independence of chunks)



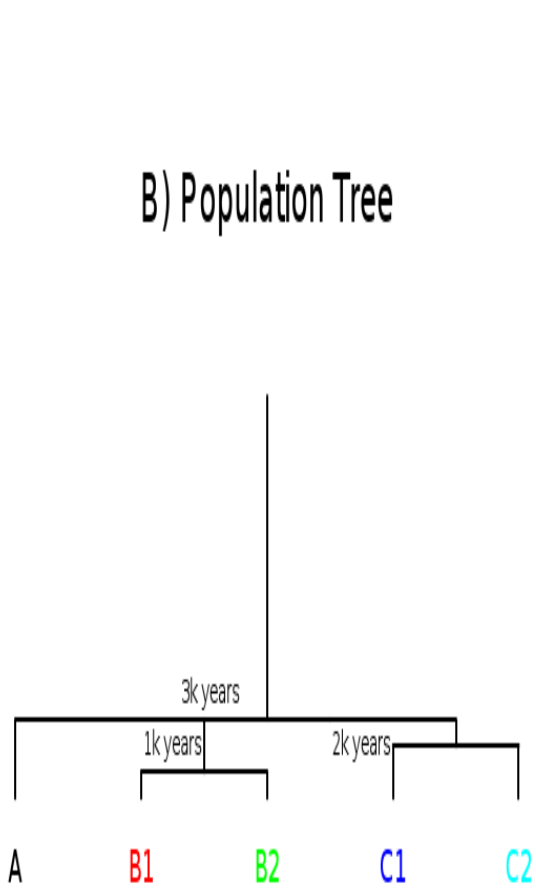
(Lawson et al 2012, *PLoS Genet* **8**:e1002453)

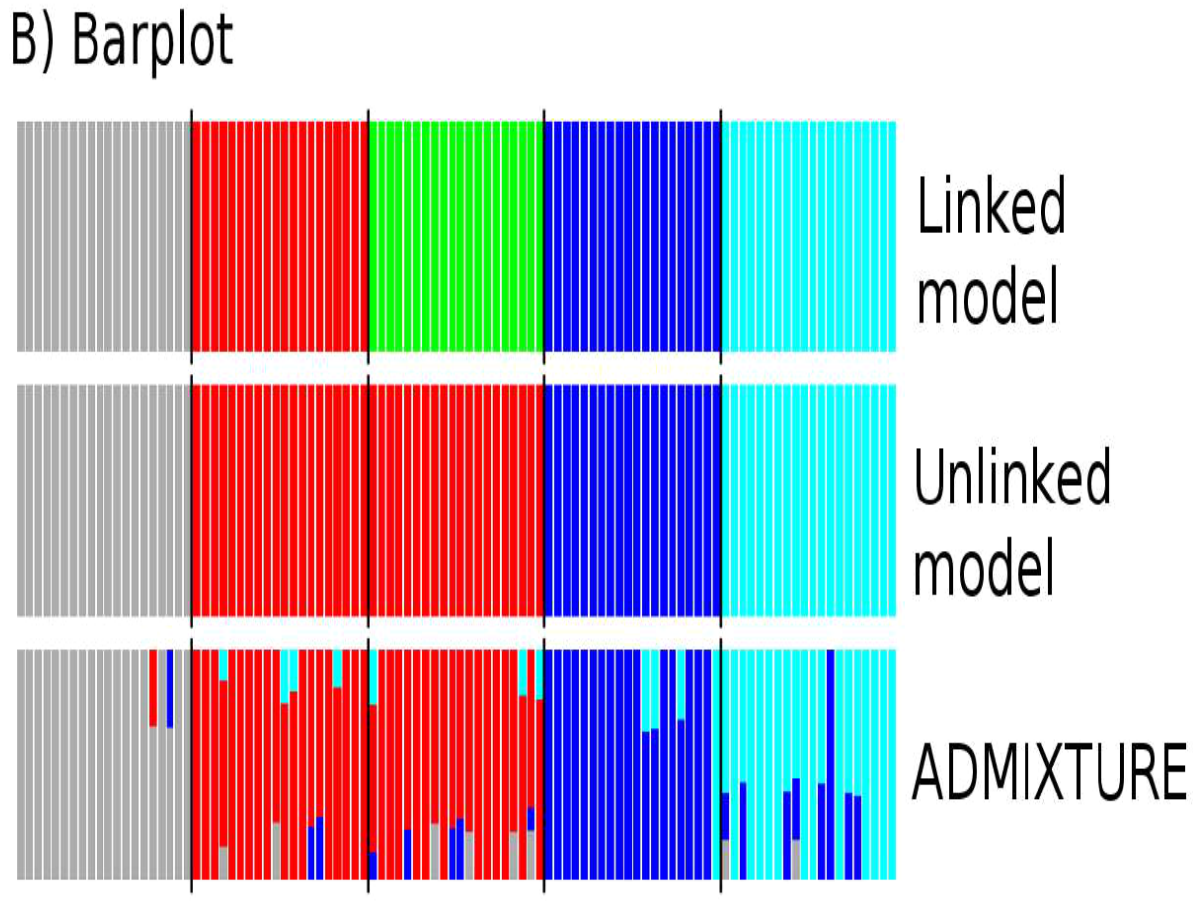
Simulated Example – 5 pops (classification)





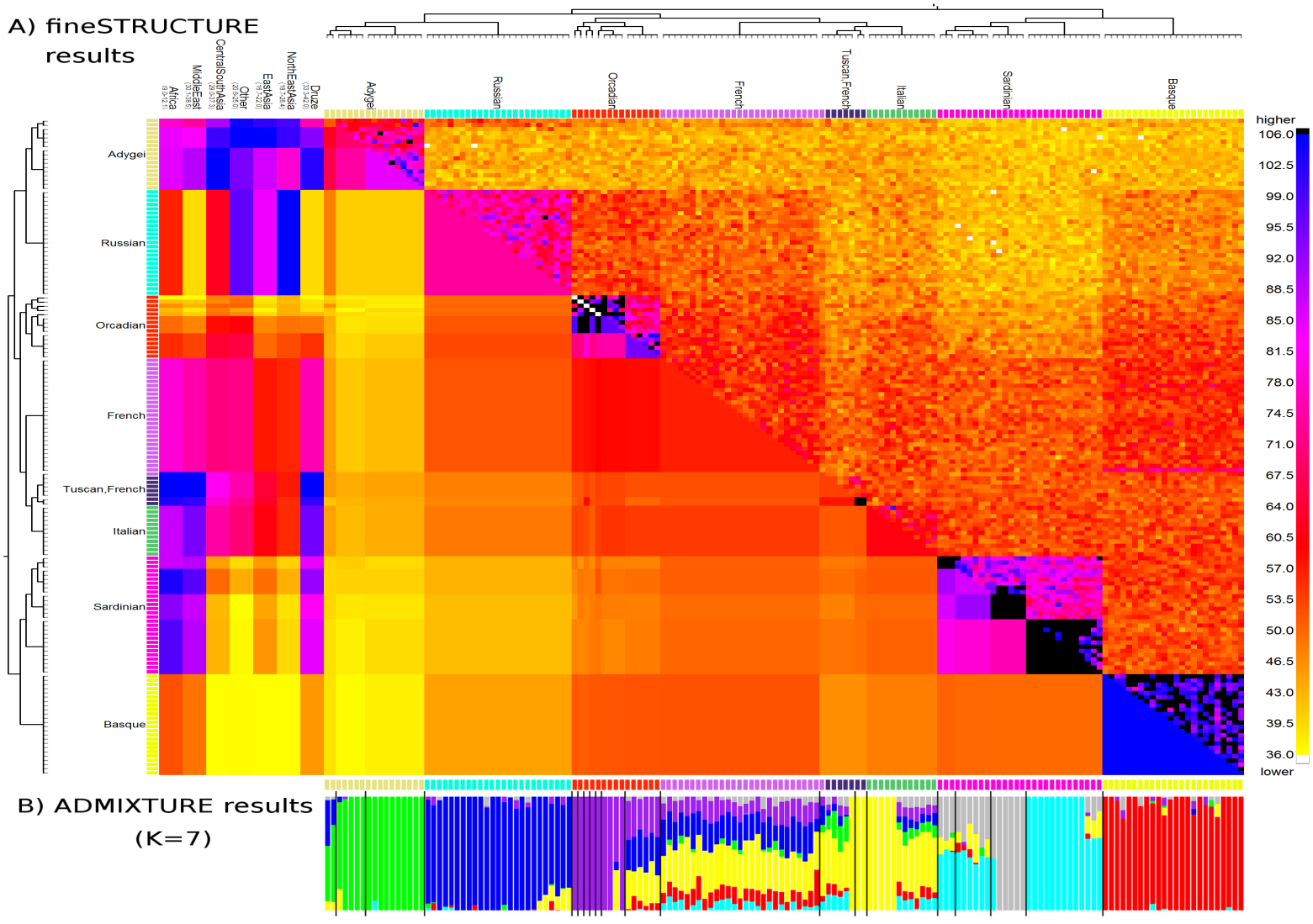
Simulated Example – 5 pops (classification)





- can build tree by greedily merging pairs of clusters, two-at-a-time, until all are merged

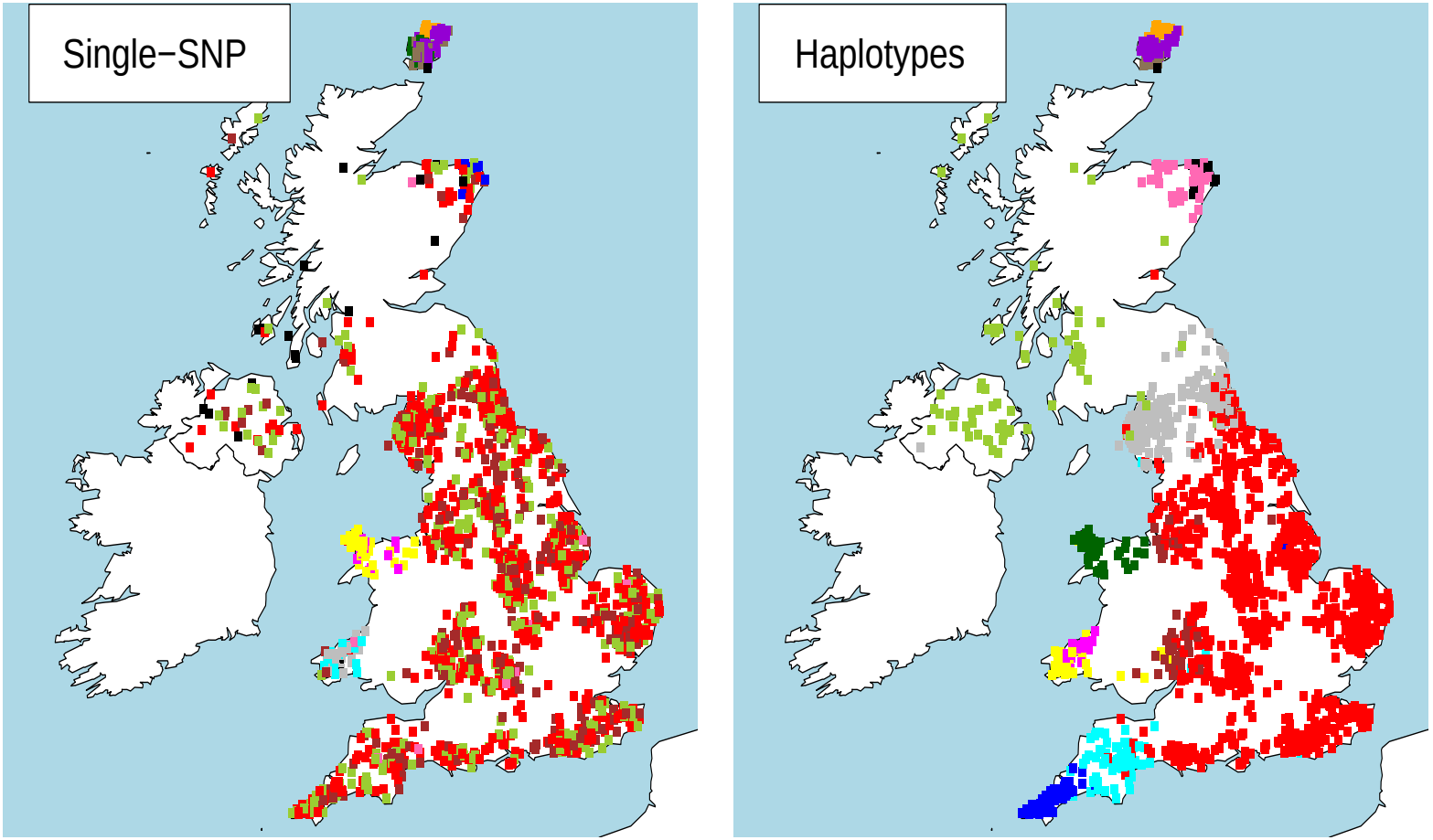
European populations (Lawson et al 2012, *PLoS Genet* **8**:e1002453)



upper right triangle: # of segments each row (recipient) copies from each column (donor)

lower left triangle: averages across all inds within same cluster

United Kingdom – clustering using haplotype info



(Leslie et al 2015, *Nature* **519**:309)

- dots = individuals / colors = clusters
- **left:** ignore haplotypes (as in PCA, STRUCTURE/ADMIXTURE)
- **right:** using haplotypes shows more localised (though subtle!) correspondence of genetics and geography

running *CHROMOPAINTER* and *fineSTRUCTURE*

Running fineSTRUCTURE requires three steps:

0. phase the data, e.g. using *SHAPEIT*, *GLIMPSE*, *EAGLE*, or *BEAGLE*,...

1. use *CHROMOPAINTER* to paint each phased individual against all other individuals

2. calculate the “c” parameter used by *fineSTRUCTURE*

3. run *fineSTRUCTURE* using (1)-(2)

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (1) – *CHROMOPAINTER* input files:

1. "`BrahuiYorubaSimulationChrom22.haplotypes.inp`" – phased haplotypes for 1,466 individuals, per chromo

2. "`BrahuiYorubaSimulationChrom22.recomrates`" – genetic map per chromo (row = position (bp), recombination rate (M/bp))

3. "`BrahuiYorubaSimulation.idfile.txt`" – 1,466 individuals (row = ID,population,1=include)

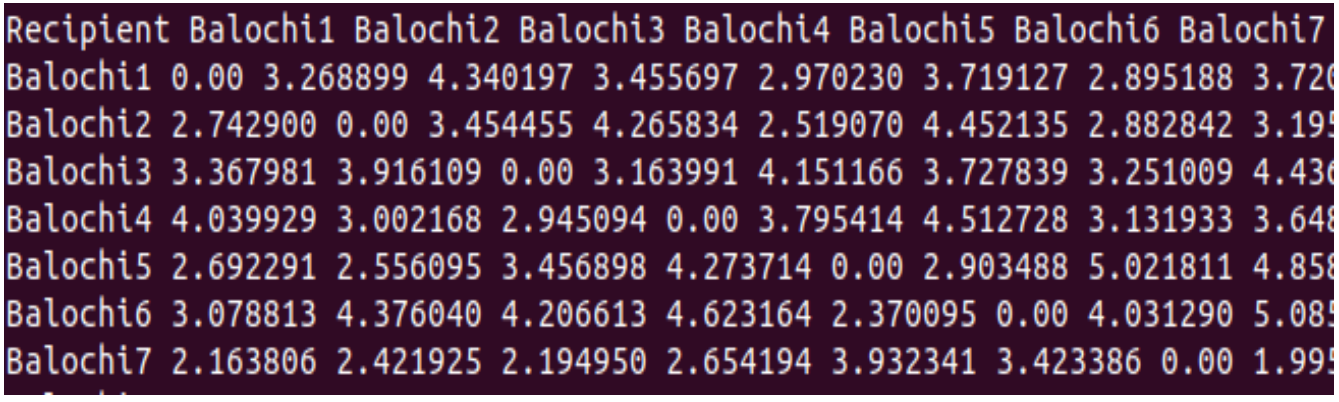
4. "`BrahuiYorubaSimulationSurrogatesOnly.poplist.txt`" – lists which populations you want to include (if not all):

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (1) – run *CHROMOPAINTER* on individuals to cluster:

```
./ChromoPainterv2 -g [].haplotypes
-r [].recomrates -t [].idfile.txt -f [].poplist.txt
0 0 -o [outputname] -a 0 0 -s 0
```

- "`-a 0 0`" specifies to paint each individual using every other individual
- "`-s 0`" specifies not to print painting samples files, which are large (though we will use these in next practical)
- output file of interest: `[outputname].chunkcounts.out`



Recipient	Balochi1	Balochi2	Balochi3	Balochi4	Balochi5	Balochi6	Balochi7
Balochi1	0.00	3.268899	4.340197	3.455697	2.970230	3.719127	2.895188
Balochi2	2.742900	0.00	3.454455	4.265834	2.519070	4.452135	2.882842
Balochi3	3.367981	3.916109	0.00	3.163991	4.151166	3.727839	3.251009
Balochi4	4.039929	3.002168	2.945094	0.00	3.795414	4.512728	3.131933
Balochi5	2.692291	2.556095	3.456898	4.273714	0.00	2.903488	5.021811
Balochi6	3.078813	4.376040	4.206613	4.623164	2.370095	0.00	4.031290
Balochi7	2.163806	2.421925	2.194950	2.654194	3.932341	3.423386	0.00

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (1a)* – run *CHROMOPAINTER* on individuals to cluster:

```
./ChromoPainterv2 -g [].haplotypes
-r [].recomrates -t [].idfile.txt -f [].poplist.txt
0 0 -o [outputnameEM] -a 0 0 -s 0 -i 10 -in -iM
```

- **Best practice:** first run Expectation-Maximisation (E-M) algorithm (on subset of chromosomes & painted individuals, keeping **all** donors) to estimate:
  - `-in`: estimate HMM "switch" parameter
  - `-iM`: estimate HMM "emission" parameter
  - `-i 10`: estimate above over 10 E-M iterations

- Relevant output will be in `[outputnameEM].EMprobs.out` file:

```
Balochi1
0 -1197.4467921360 -1326.2703819599 784.3137254902 0.0001438258
1 -1191.9740201581 -1321.5223601777 715.6688207298 0.0003446008
2 -1189.0247455945 -1318.9673783224 664.8573349699 0.0005137862
3 -1187.4164041169 -1317.6371003881 628.0776391760 0.0006408585
4 -1186.5517692789 -1316.9685639856 601.8825216129 0.0007343492
5 -1186.0950921372 -1316.6435744776 583.4344440167 0.0008029370
6 -1185.8571810738 -1316.4903093718 570.5387679785 0.0008531195
7 -1185.7342481549 -1316.4201677725 561.5671442892 0.0008896319
8 -1185.6708889143 -1316.3891660162 555.3437971466 0.0009160015
9 -1185.6381408653 -1316.3761152422 551.0343932022 0.0009348968
10 -1185.6210766539 -1316.3710574231 548.0533130105 0.0009483379
Balochi2
0 -1232.8221474220 -1226.4397362855 784.3137254902 0.0001438258
1 -1224.5166352975 -1216.9883066602 681.7766483588 0.0000822284
```

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (1b)* – run *CHROMOPAINTER* on individuals to cluster:

```
./ChromoPainterv2 -g [].haplotypes
-r [].recomrates -t [].idfile.txt -f [].poplist.txt
0 0 -o [outputnameEM] -a 0 0 -s 0 -n [nval] -M [mval]
```

- **Best practice:** then average E-M estimated values across (subset of) painted individuals and chromosomes

- re-run *CHROMOPAINTER* on all individuals/chromosomes:
  - `-n [nval]`: average inferred “switch” parameter (column 4)
  - `-M [mval]`: average inferred “emission” parameter (column 5)

- When averaging, use the **last** E-M iteration values (e.g. iter 10):

```
Balochi1
0 -1197.4467921360 -1326.2703819599 784.3137254902 0.0001438258
1 -1191.9740201581 -1321.5223601777 715.6688207298 0.0003446008
2 -1189.0247455945 -1318.9673783224 664.8573349699 0.0005137862
3 -1187.4164041169 -1317.6371003881 628.0776391760 0.0006408585
4 -1186.5517692789 -1316.9685639856 601.8825216129 0.0007343492
5 -1186.0950921372 -1316.6435744776 583.4344440167 0.0008029370
6 -1185.8571810738 -1316.4903093718 570.5387679785 0.0008531195
7 -1185.7342481549 -1316.4201677725 561.5671442892 0.0008896319
8 -1185.6708889143 -1316.3891660162 555.3437971466 0.0009160015
9 -1185.6381408653 -1316.3761152422 551.0343932022 0.0009348968
10 -1185.6210766539 -1316.3710574231 548.0533130105 0.0009483379
Balochi2
0 -1232.8221474220 -1226.4397362855 784.3137254902 0.0001438258
1 -1224.5166352975 -1216.9883066602 681.7766483588 0.0000822284
```

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (2) – calculate the "c" parameter used by *fineSTRUCTURE*:

```
Rscript calcC_Continents.R [outputname]
```

- uses `[].regionchunkcounts.out` and `[].regionsquaredchunkcounts.out` to find variance in "chunk" counts matched to each donor
- does this for regions of size *X* chunks, with *X* specified by "-k" (default: `'-k 100'`)
- compares observed variance to that expected under multinomial model; ratio gives `c`, which is printed to screen
- **IN PRACTICE:** should check `[].regionchunkcounts.out` to ensure `num.regions` $>$ 0; otherwise decrease `'-k'`

Recipient	num.regions	Balochi1	Balochi2	Balochi3	Balochi4	Balochi5	Baloch
Balochi1	6	0.00	2.461094	3.313439	2.676441	2.434774	3.249375	2.384335	2.9
Balochi2	6	2.342869	0.00	3.138498	3.834000	1.747270	4.102572	2.352419	2.1
Balochi3	6	2.460739	3.522314	0.00	2.494246	3.779198	3.300247	2.778274	3.7
Balochi4	6	3.799145	2.801584	2.525160	0.00	3.622373	4.354997	2.934322	3.4
Balochi5	6	2.387207	2.138053	3.181526	4.020499	0.00	2.626142	3.603930	4.0
Balochi6	6	3.012488	4.303778	4.135052	4.517962	2.201075	0.00	3.965993	5.0
Balochi7	4	1.386228	1.536604	1.637940	1.665844	2.514825	2.776182	0.00	1.4

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (3) – run *fineSTRUCTURE* clustering:

```
fs_4.0.0/fs_linux_glibc2.3 finestructure -I 1 -c
[c-value] -x 1000000 -y 2000000 -z 10000
[].chunkcounts.out [outputname]
```

- `-c [c-value]`: value of `c` calculated in step (2)
- `-x, -y, -z`: run *y* MCMC iterations, sampling every *z*th iteration after *x* burn-in iterations
- `-I 1`: individuals are all clustered together at first; this seems to help with efficiency

running *CHROMOPAINTER* and *fineSTRUCTURE*

Step (3) – run *fineSTRUCTURE* tree-building:

```
fs_4.0.0/fs_linux_glibc2.3 finestructure -c
[c-value] -x 100000 -k 2 -m T -t 1000000
[].chunkcounts.out [outputname] [outputnameTREE]
```

- `-m T`: perform tree-building (rather than clustering)
- `-x 100000`: perform 100,000 additional hill-climbing iterations before building tree
- `-t 1000000`: at each level of the tree, compare 1,000,000 pairs of clusters, merging two that give lowest decrease in posterior prob (recommend setting very high, so `fineSTRUCTURE` does *all* pairwise comparisons)

fineSTRUCTURE (Lawson et al 2012, *PLoS Genet* **8**:e1002453)

- **Advantages:**
  - increased power – finds more subtle structure in data missed by *ADMIXTURE* (e.g. Europe, United Kingdom)
  - heatmaps can show clear structure/admixture
  - current implementation infers number of clusters K automatically, and builds "tree" relating these clusters

- **Disadvantages:**
  - how to interpret results? Drift/admixture/other can result in similar signals
  - requires phased data (e.g. using *SHAPEIT*)
  - currently only has "no admixture" model only
  - computationally slow (relative to *ADMIXTURE*), more complicated to use

Outline

Clustering individuals: CHROMOPAINTER + fineSTRUCTURE

Inferring ancestry proportions: CHROMOPAINTER + GLOBETROTTER/SOURCEFIND

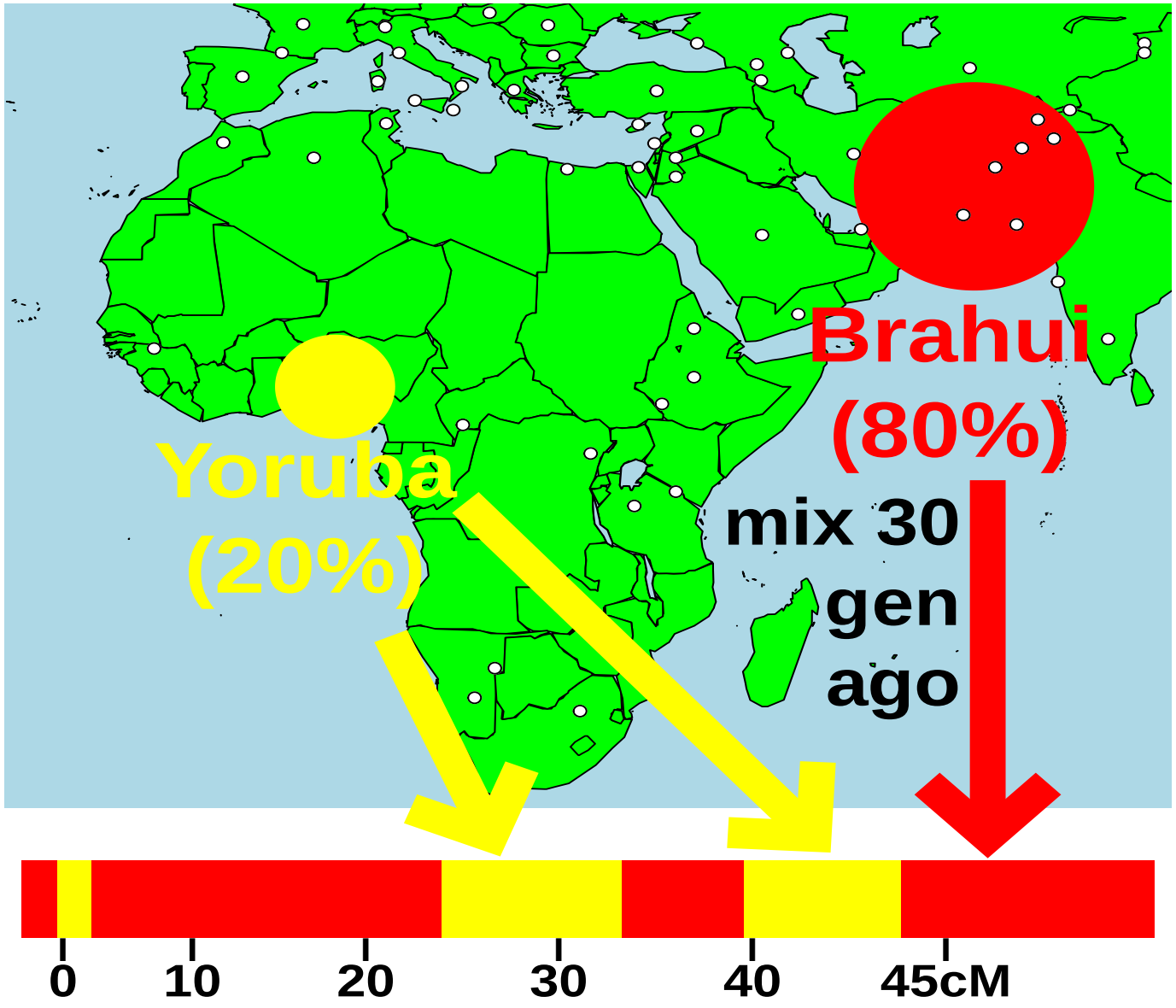
Simulated Example

We will use a simulation for illustration.

- 20 admixed “individuals”
- descend from admixture occurring 30 generations ago

  (Price et al 2009, Hellenthal et al 2014)
- **80%** of DNA from **Brahui** from Pakistan
- **20%** from **Yoruba** from Nigeria
- will use proxy (surrogate) populations to represent each admixing source:
  - **Brahui** $\longrightarrow$ Balochi
  - **Yoruba** $\longrightarrow$ BantuKenya, BantuSouthAfrica, Mandenka

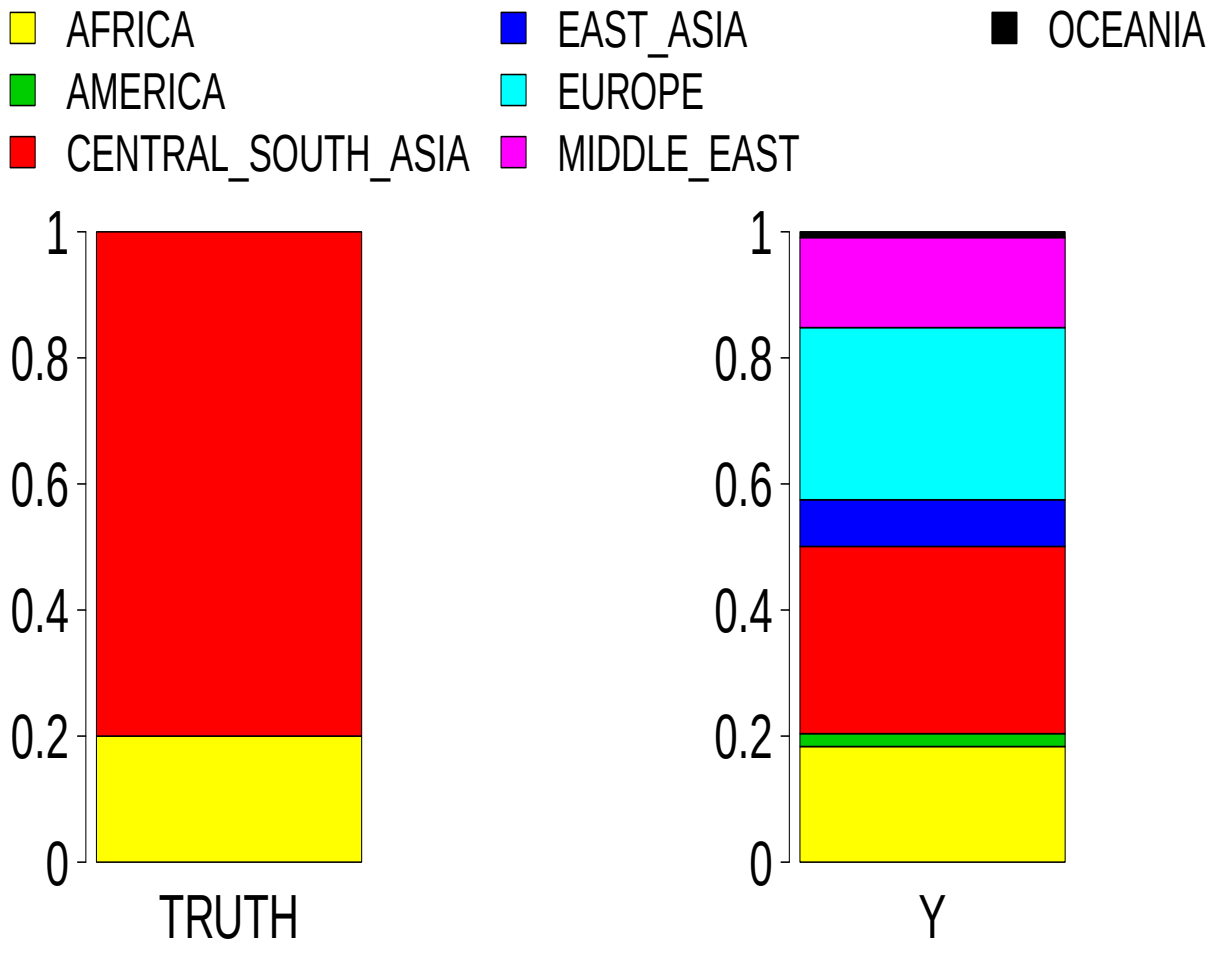
Simulation: 80% **Brahui** + 20% **Yoruba**, 30gen



(Hellenthal et al 2014, *Science* **343**:747)

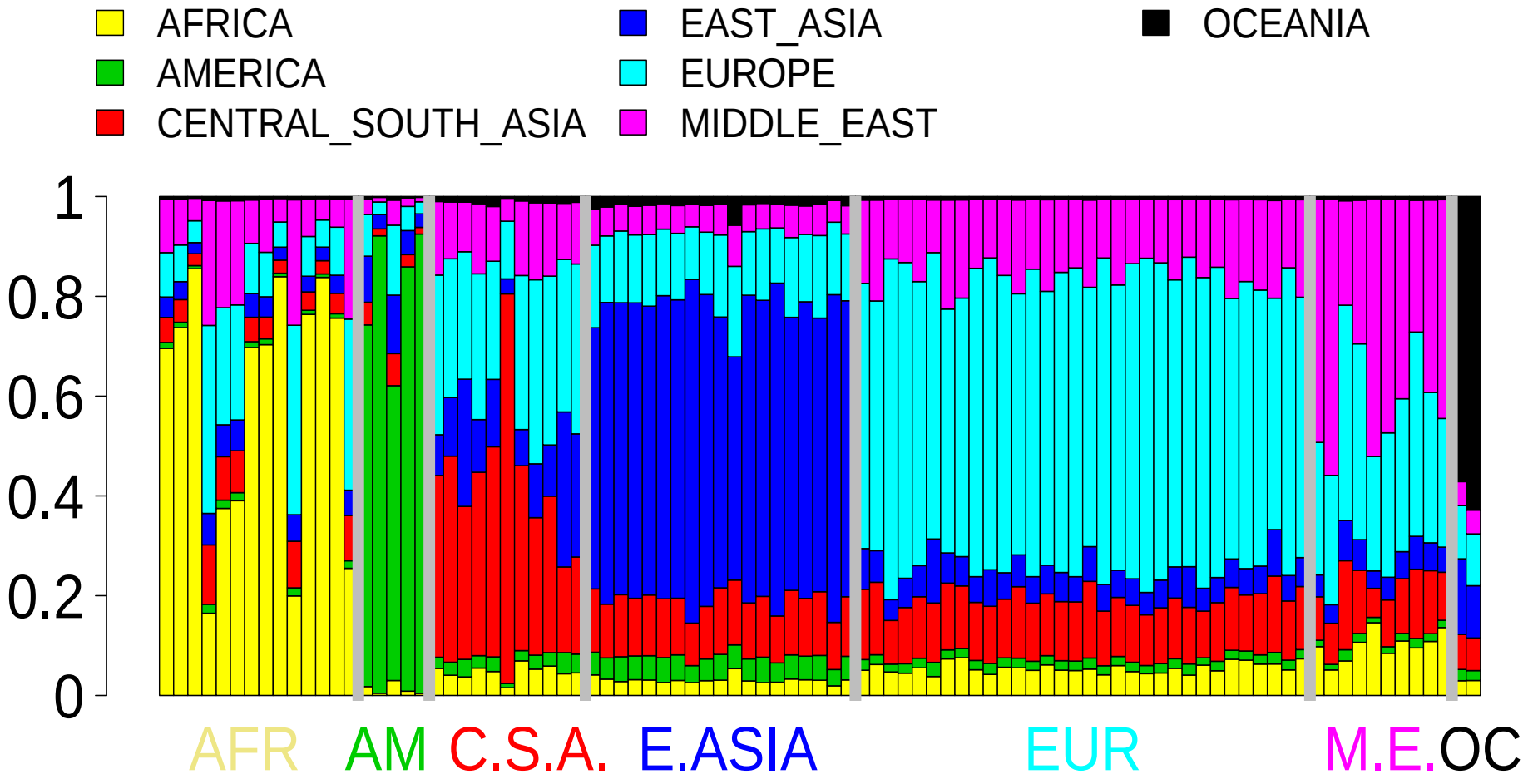
Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)

(1) paint admixed inds using 93 world groups (*CHROMOPAINTER*)



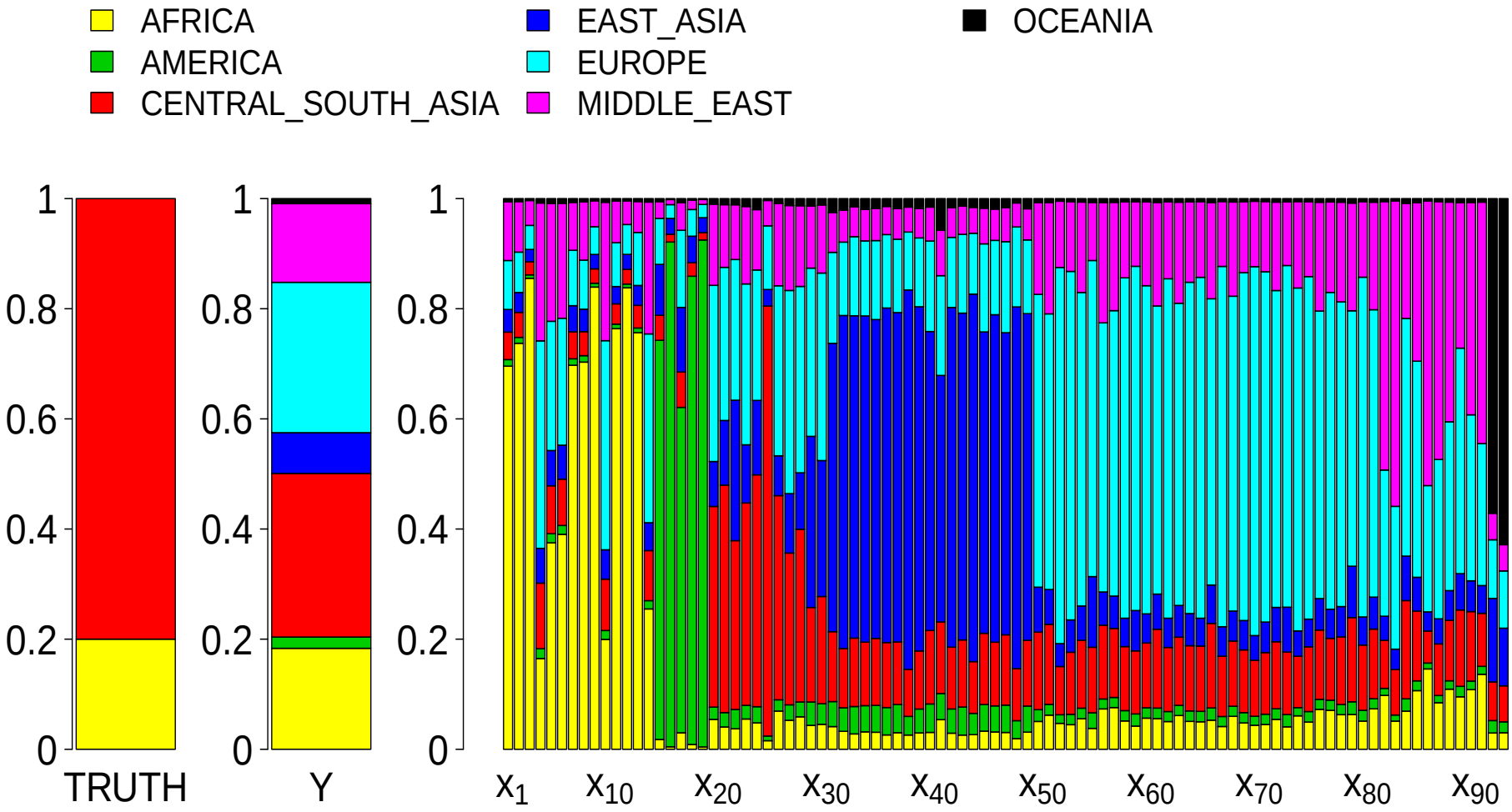
Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)

(1) paint admixed inds using 93 world groups (*CHROMOPAINTER*)



Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)

(1) paint admixed inds using 93 world groups (*CHROMOPAINTER*)

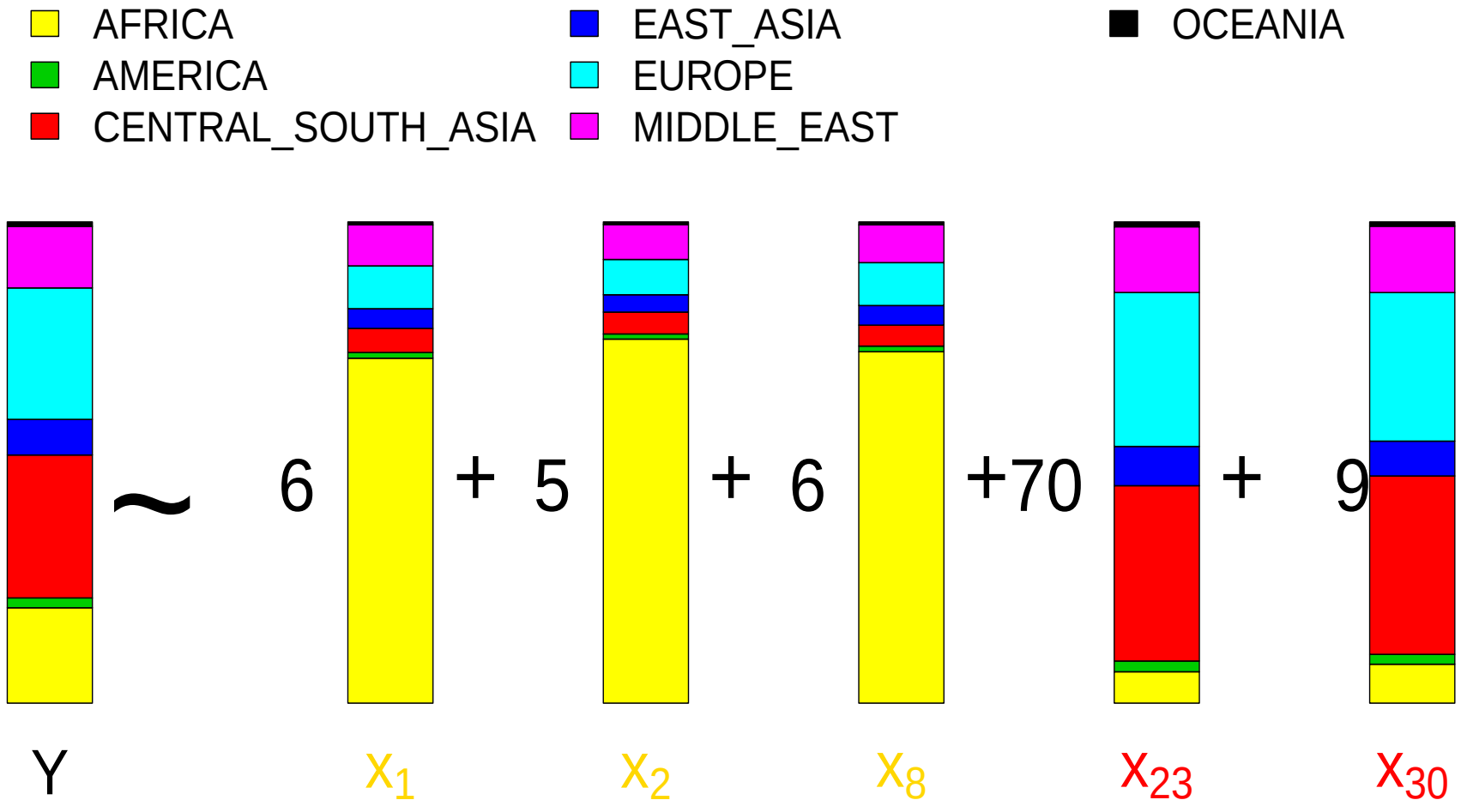


(2) form admixed ind's painting as mixture of that for all 93 groups:

$$E[Y] = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{93} X_{93} \qquad \text{(with } \sum_i \beta_i = 1.0\text{)}$$

Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)

(1) paint admixed inds using 93 world groups (*CHROMOPAINTER*)

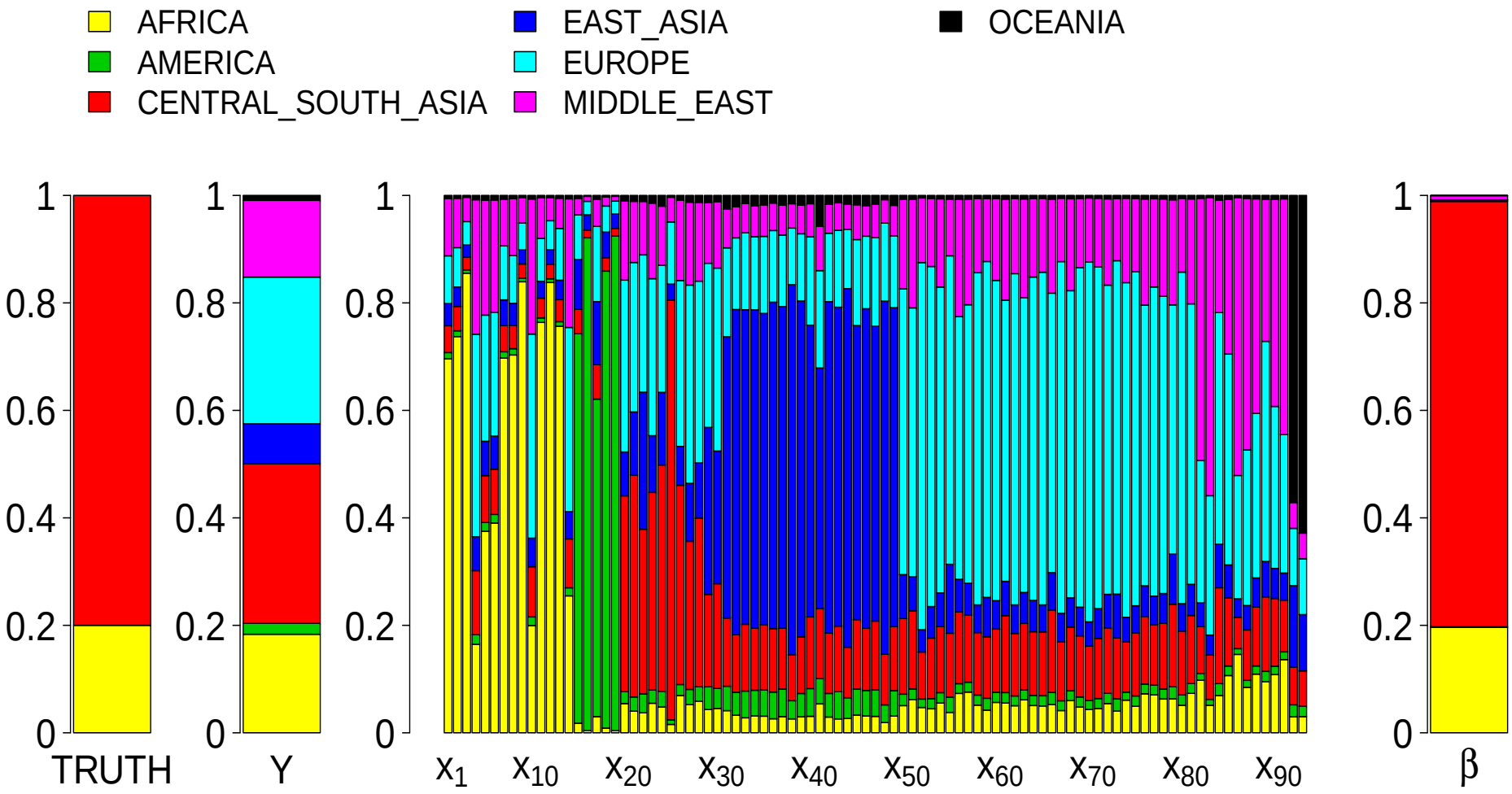


(2) form admixed ind's painting as mixture of that for all 93 groups:

$$E[Y] = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{93} X_{93} \qquad \text{(with } \textstyle\sum_i \beta_i = 1.0\text{)}$$

Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)

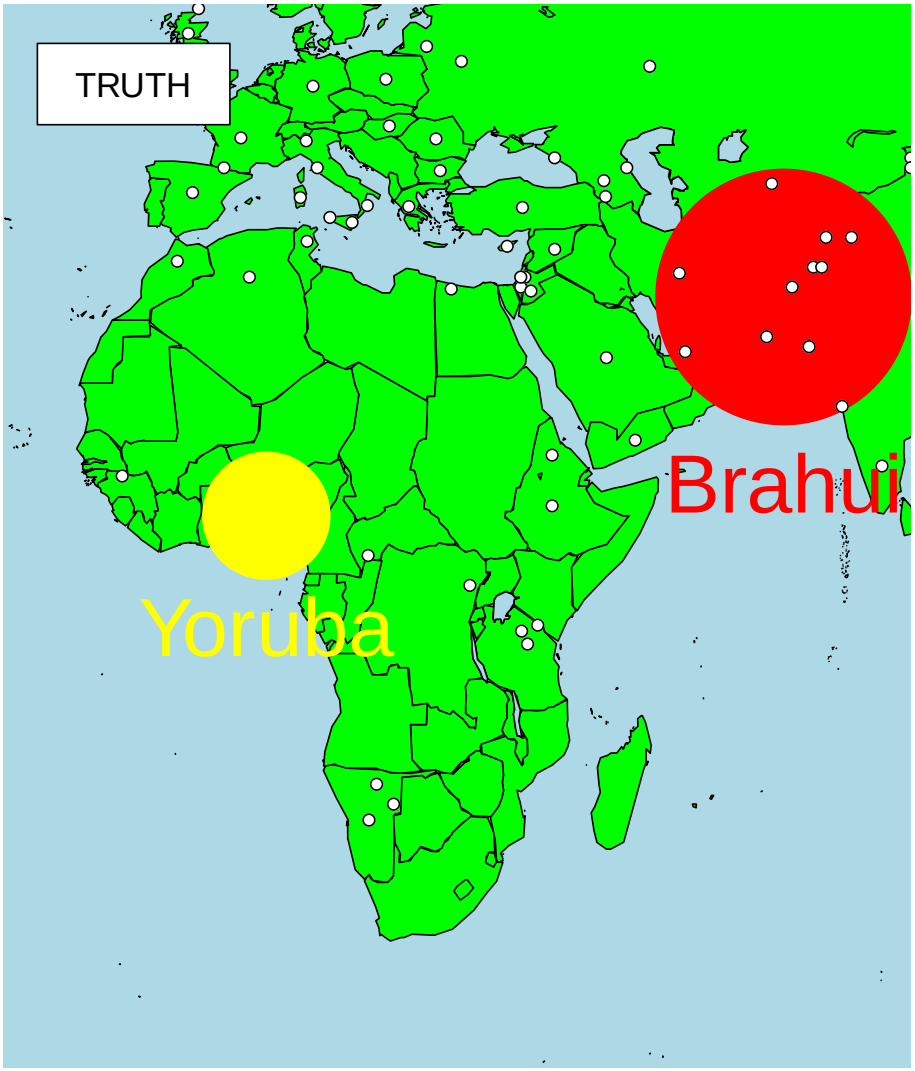
(1) paint admixed inds using 93 world groups (*CHROMOPAINTER*)



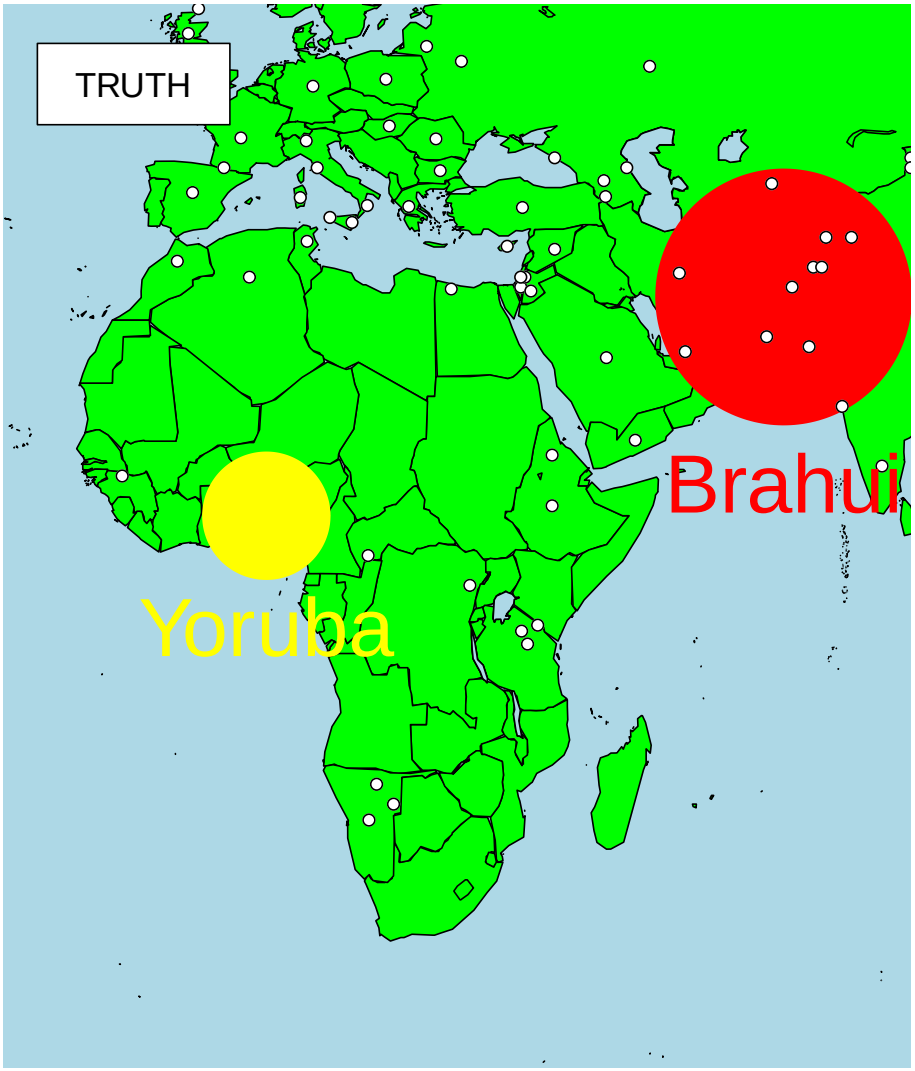
(2) form admixed ind's painting as mixture of that for all 93 groups:

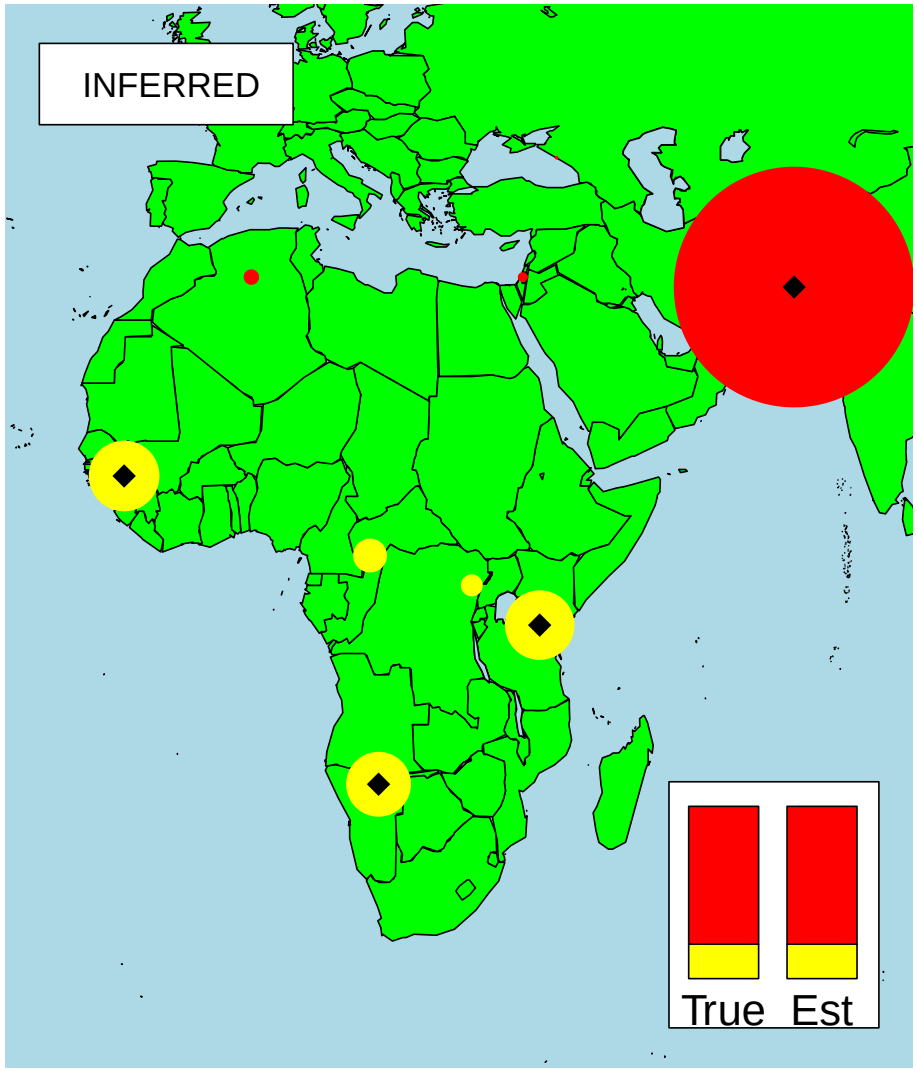
$$E[Y] = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{93} X_{93} \qquad \text{(with } \sum_i \beta_i = 1.0\text{)}$$

Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)



Infer mixing groups (Sim: 80% **Brahui** + 20% **Yoruba**, 30gen)





GLOBETROTTER (NNLS) vs SOURCEFIND

(1) *GLOBETROTTER (NNLS):* $Y = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_K X_K + \epsilon$

(2) *SOURCEFIND:* $Y \sim \text{Multinom}\left(\vec{p} \equiv \sum_{i=1}^{K} \left[\beta_i X_i\right]\right)$

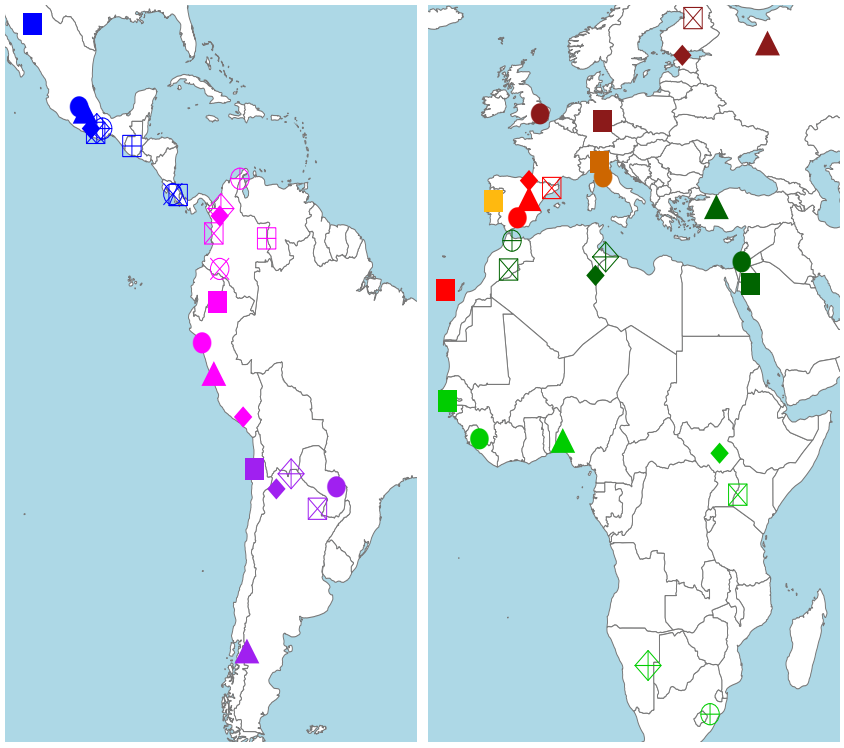
$$(\text{number of } \beta_i > 0) \sim \text{Poisson}(\lambda)$$

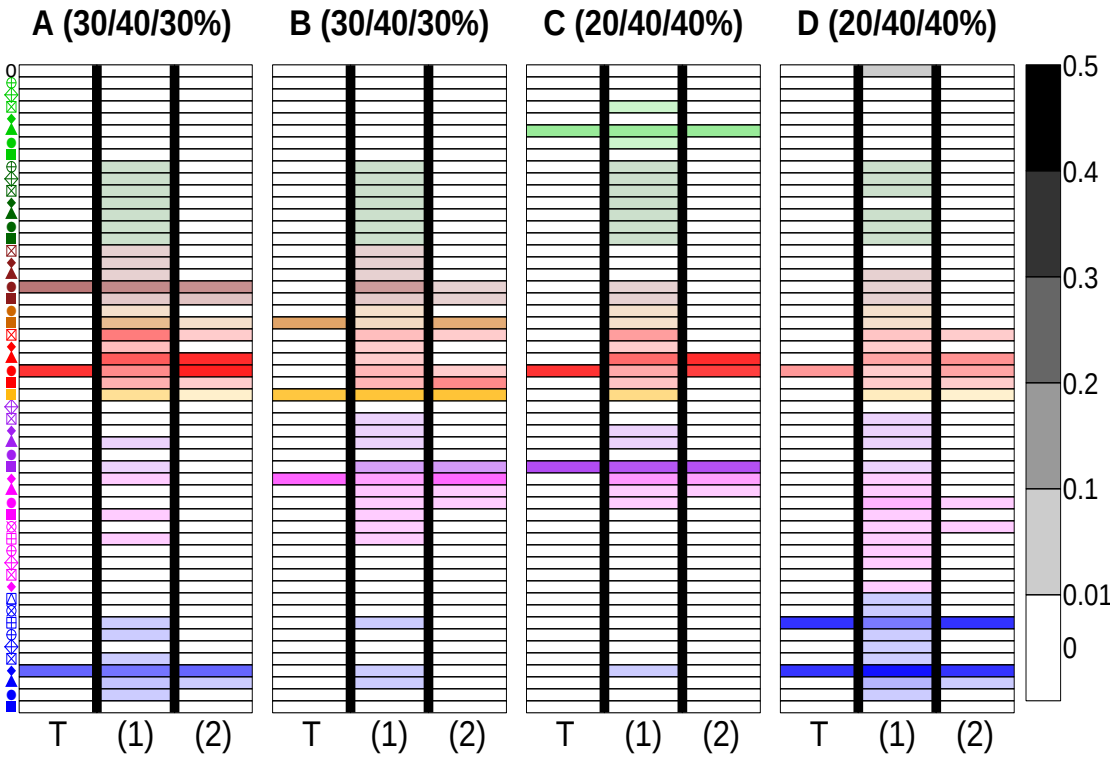
GLOBETROTTER (NNLS) vs SOURCEFIND

(1) *GLOBETROTTER (NNLS):* $Y = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_K X_K + \epsilon$

(2) *SOURCEFIND:* $Y \sim \text{Multinom}\left(\vec{p} \equiv \sum_{i=1}^{K} \left[\beta_i X_i\right]\right)$

$$(\text{number of } \beta_i > 0) \sim \text{Poisson}(\lambda)$$





Chacon-Duque et al 2018, *Nat Comm* **9**:5388

Inferring ancestry proportions

First use *CHROMOPAINTER*:

0. phase the data, e.g. using *SHAPEIT*, *GLIMPSE*, *EAGLE*, or *BEAGLE*,...

1. use *CHROMOPAINTER* to paint each **ancestry surrogate** individual against all other surrogate individuals

2. use *CHROMOPAINTER* to paint each **target (e.g. admixed)** individual against all other surrogate individuals

3. combine data from (1)-(2)

Inferring ancestry proportions

First use *CHROMOPAINTER*:

- step (1) can be done following what was done in previous section

- step (2): paint each **target (e.g. admixed)** individual against all other surrogate individuals:

```
./ChromoPainterv2 -g [].haplotypes
-r [].recomrates -t [].idfile.txt
-f [].poplist.txt 0 0 -o [outputname]
```

```
Balochi D
BantuKenya D
BantuSouthAfrica D
Burusho D
English D
HanNchina D
Kalash D
Makrani D
Mandenka D
MbutiPygmy D
Mongola D
NorthItalian D
Orcadian D
Pathan D
Sardinian D
Tuscan D
BrahuiYorubaSimulation R
```

- **NOTE:** do **not** use target population as donors, since this will mask any ancestry (and admixture) signals

Inferring ancestry proportions

First use *CHROMOPAINTER*:

- step (1) can be done following what was done in previous section

- step (2): paint each **target (e.g. admixed)** individual against all other surrogate individuals:

  output file of interest: `[outputname].chunklengths.out`

Recipient	Balochi	BantuKenya	BantuSouthAfrica	Burusho	English	HanNchina
BrahuiYorubaSimulation1	15.267393	5.594554	4.748974	14.843482	2.499876	
BrahuiYorubaSimulation2	18.262486	5.701112	3.388840	16.227230	3.227271	
BrahuiYorubaSimulation3	19.230889	10.064674	5.059272	11.737793	3.327937	
BrahuiYorubaSimulation4	19.675592	10.509179	6.004827	13.751833	2.569535	
BrahuiYorubaSimulation5	23.308532	5.334717	4.534991	14.432567	2.890095	
BrahuiYorubaSimulation6	28.160628	3.728591	2.808427	14.318823	2.919141	
BrahuiYorubaSimulation7	18.713144	5.441725	2.475358	15.021447	2.836016	
BrahuiYorubaSimulation8	22.204492	3.918717	4.454338	13.339341	3.089362	
BrahuiYorubaSimulation9	27.629866	4.315782	4.486023	11.948615	2.543154	
BrahuiYorubaSimulation10	17.808761	4.353916	3.174916	16.644821	3.667181	
BrahuiYorubaSimulation11	20.284687	5.607935	3.226419	14.489446	2.517388	
BrahuiYorubaSimulation12	21.898868	4.358842	2.588553	15.256689	2.892703	

- **NOTE:** do **not** use target population as donors, since this will mask any ancestry (and admixture) signals

Inferring ancestry proportions (*GLOBETROTTER*)

Run NNLS using *GLOBETROTTER*:

```
R < GLOBETROTTER.R [].paramfile -no-save >
[screenoutput]
```

- one input parameter file:

```
prop.ind: 1
bootstrap.date.ind: 1
null.ind: 1
input.file.ids: example/BrahuiYorubaSimulation.idfile.txt
input.file.copyvectors: example/BrahuiYorubaSimulationSurrogatesAndTargetsChrom22.chunklengths.out
save.file.main: example/BrahuiYorubaSimulationAdmixed.GTnnls.main
save.file.bootstraps: example/BrahuiYorubaSimulationAdmixed.GT.boot
copyvector.popnames: Balochi BantuKenya BantuSouthAfrica Burusho English HanNchina Kalash Makrani Ma
surrogate.popnames: Balochi BantuKenya BantuSouthAfrica Burusho English HanNchina Kalash Makrani Man
target.popname: BrahuiYorubaSimulation
num.mixing.iterations: 0
props.cutoff: 0.001
bootstrap.num: 20
num.admixdates.bootstrap: 1
num.surrogatepops.perplot: 3
curve.range: 1 30
bin.width: 0.1
xlim.plot: 0 30
prop.continue.ind: 0
haploid.ind: 0
```

- `num.mixing.iterations:0` specifies to use NNLS only
- output in `save.file.main` location

Inferring ancestry proportions (*SOURCEFIND*)

Run *SOURCEFIND*:

```
R < sourcefindv2.R [].SFparamfile -no-save >
[screenoutput]
```

- one input parameter file:

```
self.copy.ind: 0
num.surrogates: 8
exp.num.surrogates: 4
input.file.ids: example/BrahuiYorubaSimulation.idfile.txt
input.file.copyvectors: example/BrahuiYorubaSimulationSurrogatesAndTargetsChrom22.chunklengths.out
save.file: BrahuiYorubaSimulation.sourcefind.txt
copyvector.popnames: Balochi BantuKenya BantuSouthAfrica Burusho English HanNchina Kalash Makrani M
surrogate.popnames: Balochi BantuKenya BantuSouthAfrica Burusho English HanNchina Kalash Makrani Ma
target.popnames: BrahuiYorubaSimulation
num.slots: 100
num.iterations: 200000
num.burnin: 50000
num.thin: 5000
```

- Poisson mean of `exp.num.surrogates` surrogates
- `num.surrogates` maximum surrogates per MCMC iteration

Summary

- clustering algorithms highlight genetic differences/similarities among groups (e.g. correlations between genetics and geography)

- *CHROMOPAINTER* + *fineSTRUCTURE* – uses correlations among SNPs (haplotype information) to increase power
  - computationally slower than (e.g.) *ADMIXTURE*
  - individuals cannot be mixtures of multiple clusters

Summary

- clustering algorithms highlight genetic differences/similarities among groups (e.g. correlations between genetics and geography)
- *CHROMOPAINTER* + *fineSTRUCTURE* – uses correlations among SNPs (haplotype information) to increase power
  - computationally slower than (e.g.) *ADMIXTURE*
  - individuals cannot be mixtures of multiple clusters
- Despite “sloppy” painting, *GLOBETROTTER (NNLS)* and *SOURCEFIND* can accurately infer ancestry proportions, using *CHOMOPAINTER* output
  - **NOTE:** this does not imply admixture!!