***slendr*: a framework for simulation and analysis of spatio-temporal population genomic data**

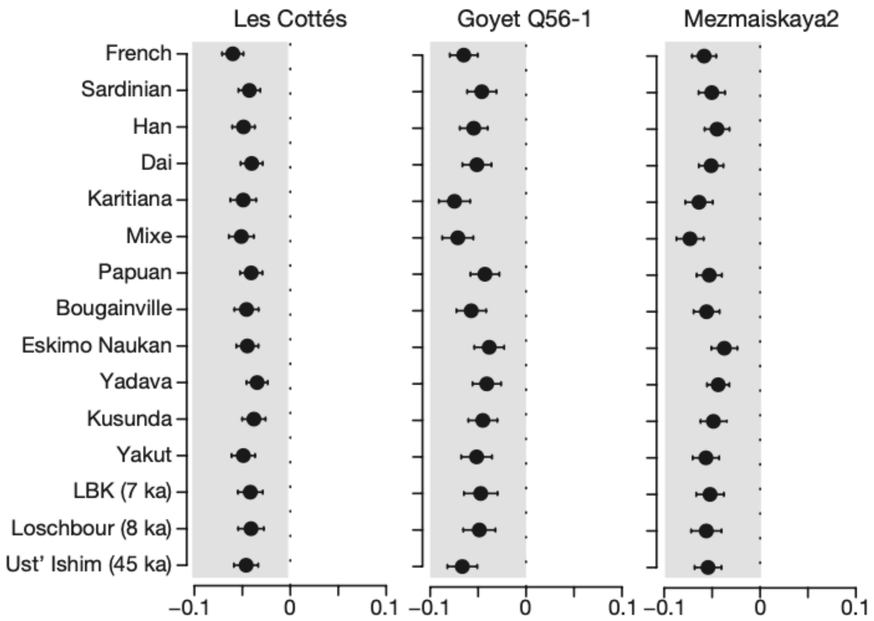
Martin Petr
mp@bodkan.net
@dr_bodkan

Globe Institute
University of Copenhagen

www.slendr.net

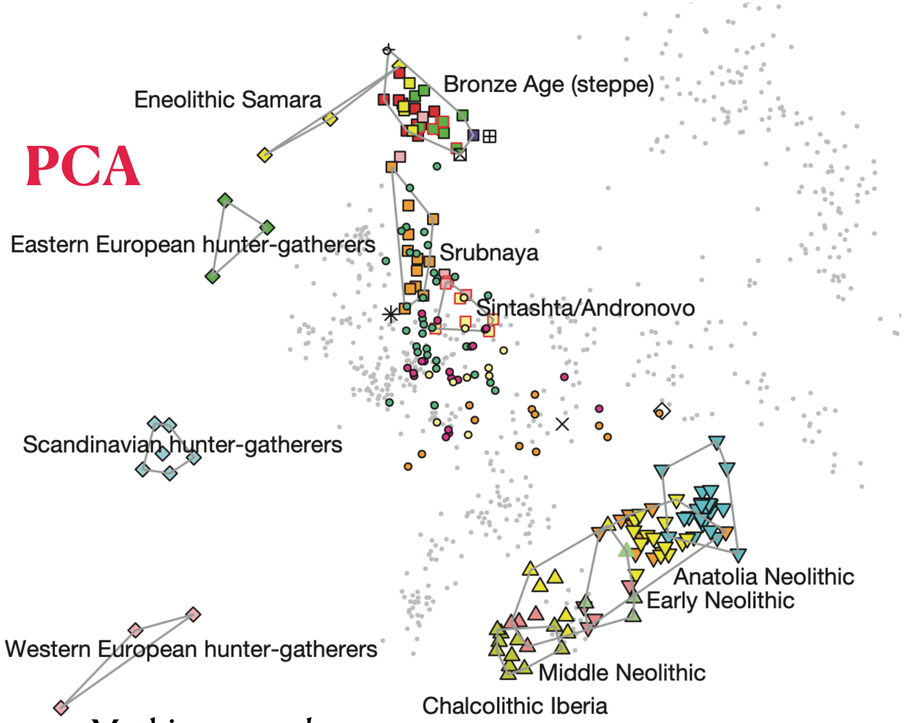
Workhorses of ancient DNA research

D statistics



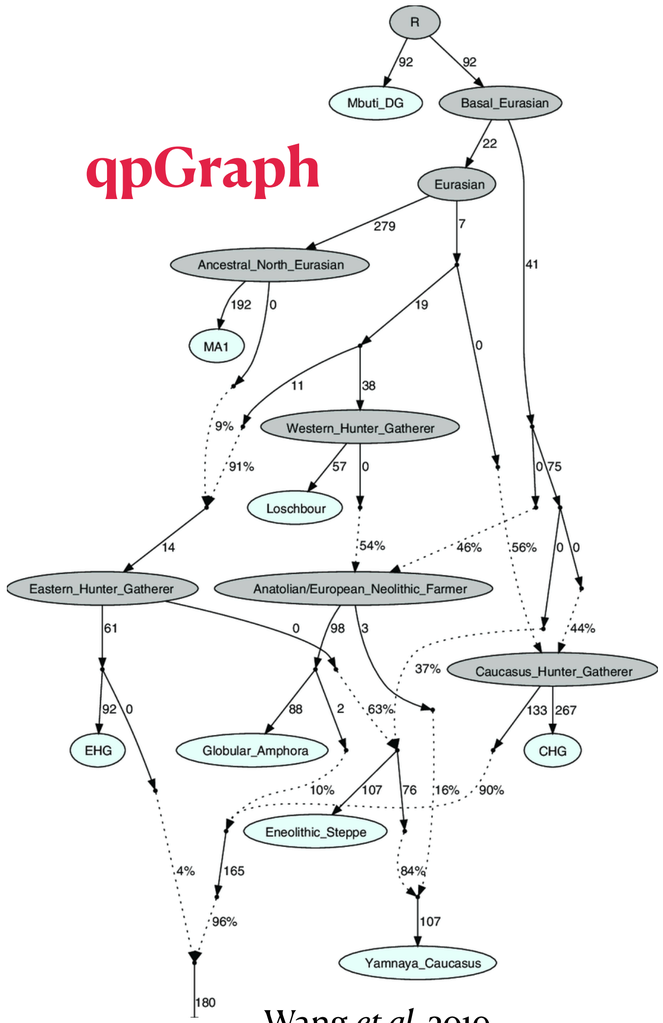
Hajdinjak *et al.* 2018

PCA

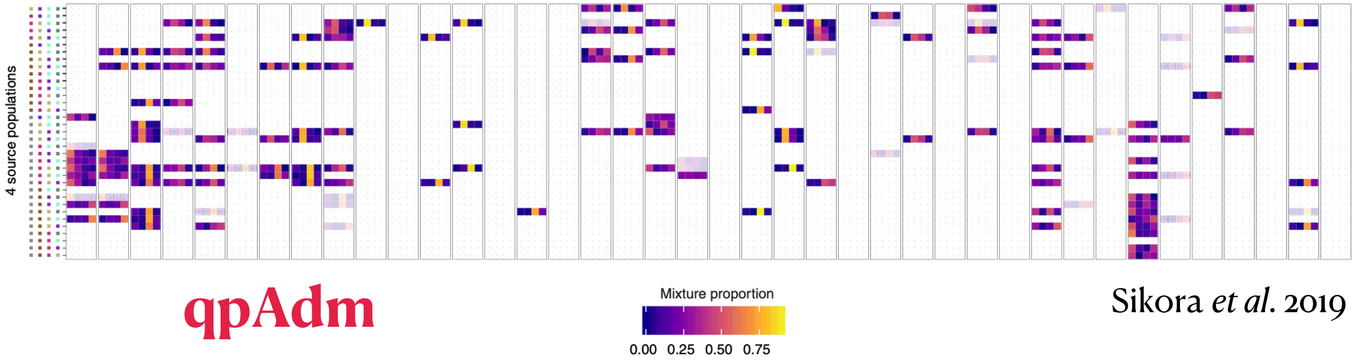


Mathieson *et al.* 2015

qpGraph

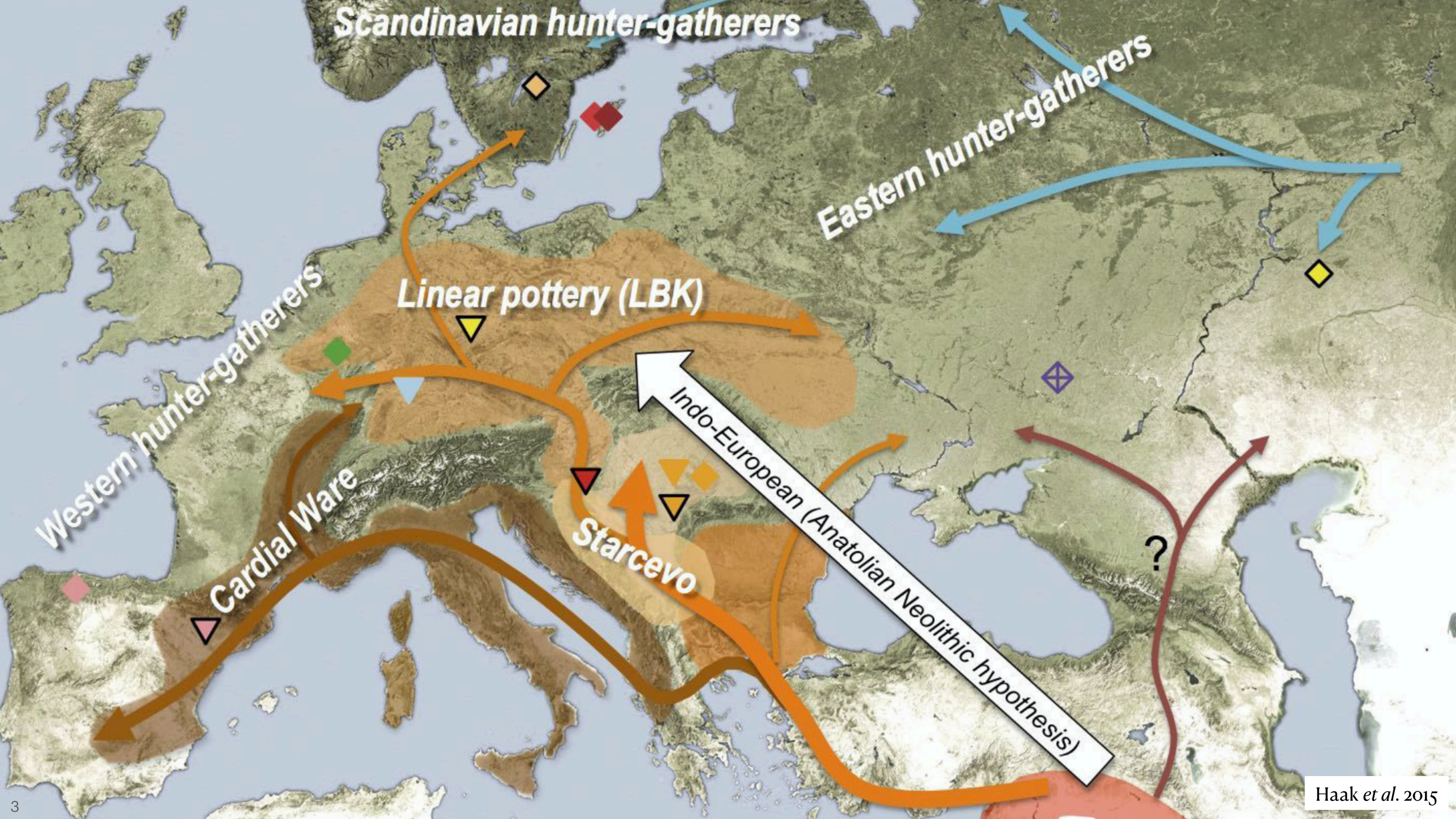


Wang *et al.* 2019



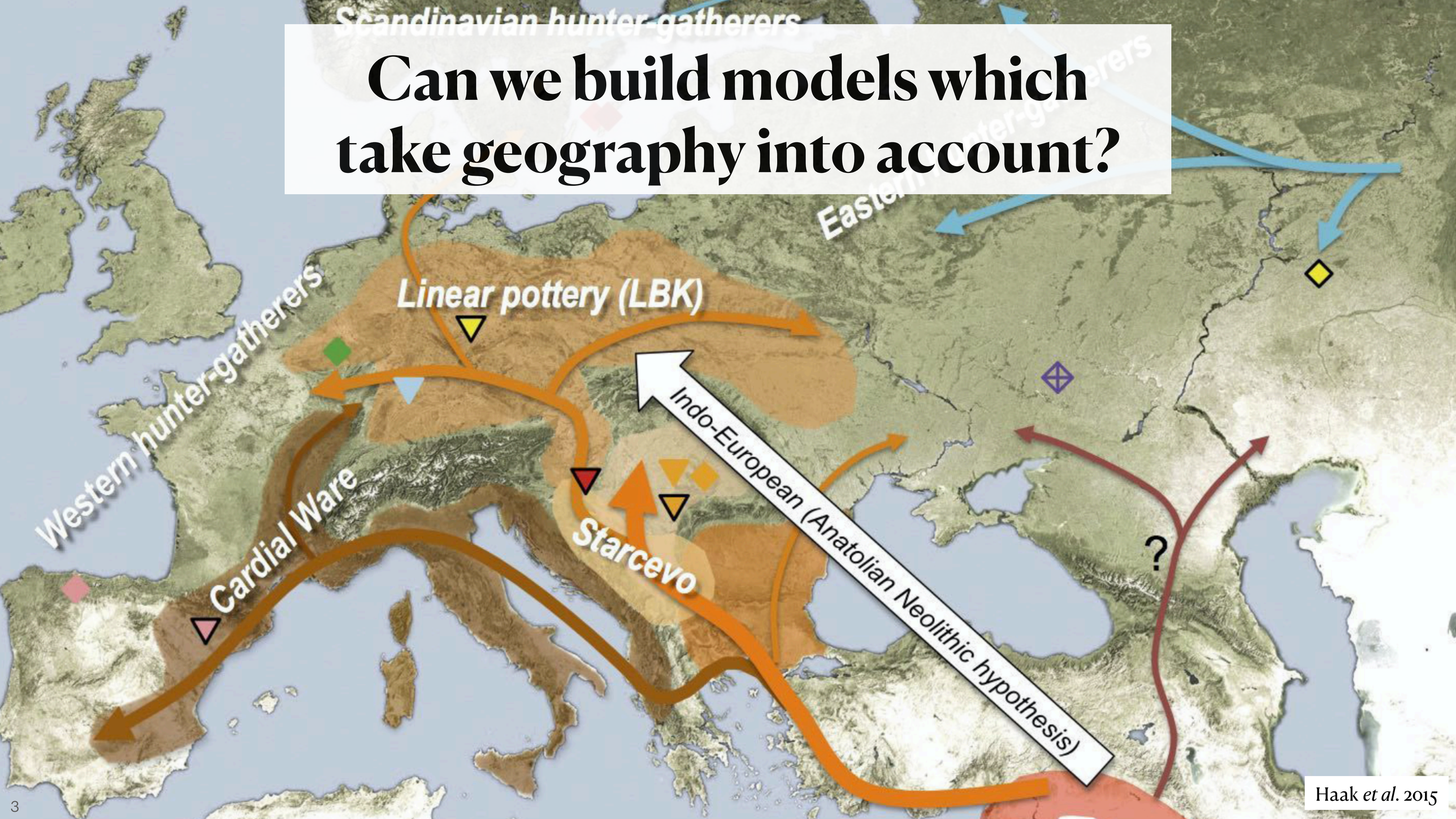
qpAdm

Sikora *et al.* 2019



Haak *et al.* 2015

Can we build models which take geography into account?



Haak *et al.* 2015

ISOLATION BY DISTANCE*

SEWALL WRIGHT

The University of Chicago[1]

Received November 9, 1942

STUDY of statistical differences among local populations is an important line of attack on the evolutionary problem. While such differences can only rarely represent first steps toward speciation in the sense of the splitting of the species, they are important for the evolution of the species as a whole. They provide a possible basis for intergroup selection of genetic systems, a process that provides a more effective mechanism for adaptive advance of the species as a whole than does the mass selection which is all that can occur under panmixia.

THE WAVE OF ADVANCE OF ADVANTAGEOUS GENES

BY R. A. FISHER, Sc.D., F.R.S. 1937

I. THE PROBLEM OF GENE DISPERSION

CONSIDER a population distributed in a linear habitat, such as a shore line, which it occupies with uniform density. If at any point of the habitat a mutation occurs, which happens to be in some degree, however slight, advantageous to survival, in the totality of its effects, we

KIMURA, M., 1953 "Stepping Stone" model of population. Ann. Rept. Nat. Inst. Genetics, Japan **3**: 62–63.

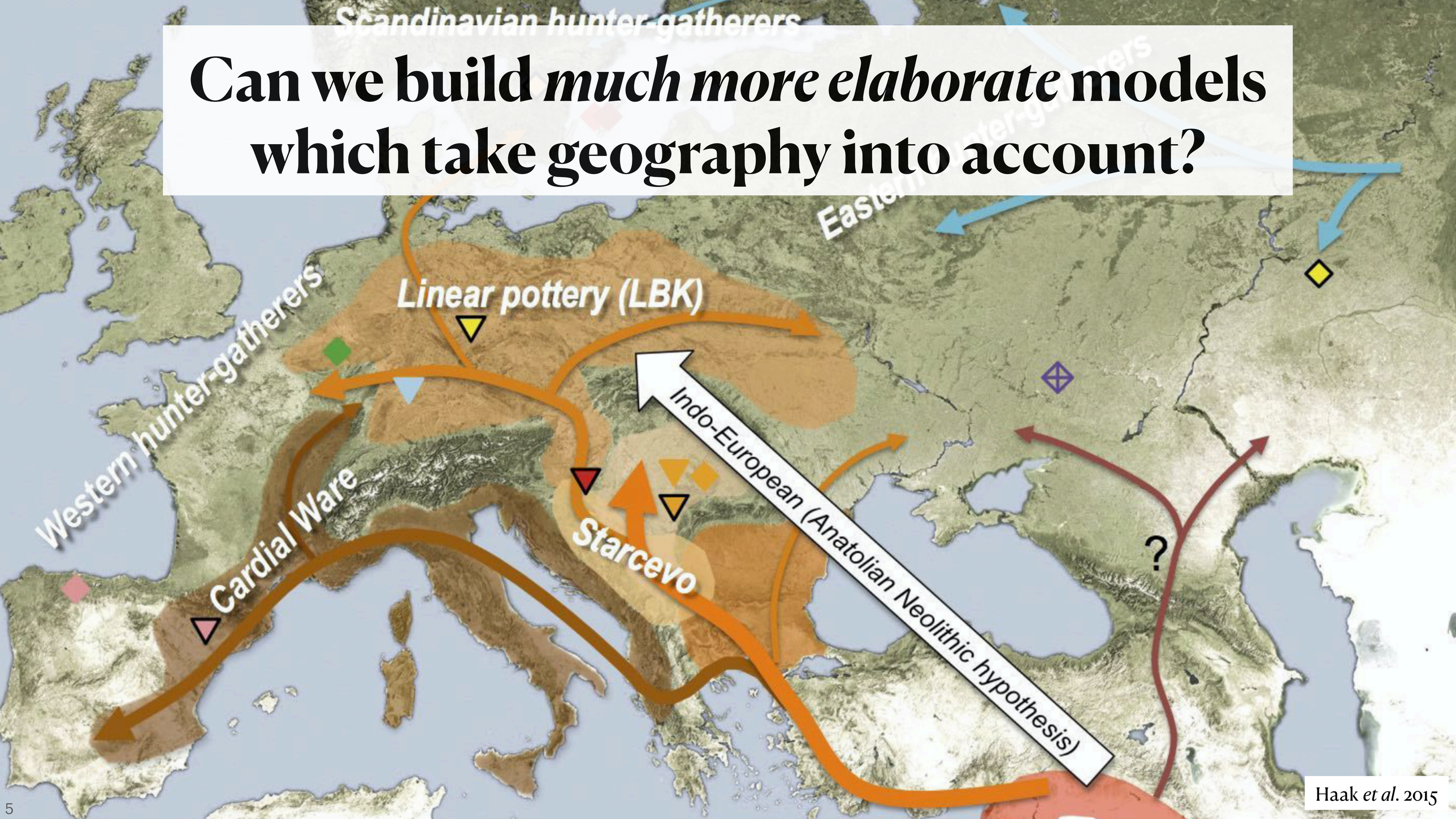
THE STEPPING STONE MODEL OF POPULATION STRUCTURE AND THE DECREASE OF GENETIC CORRELATION WITH DISTANCE[1]

MOTOO KIMURA[2] and GEORGE H. WEISS[3] 1964

c/o Department of Genetics, L.I.G.B. Pavia Section, University of Pavia, Italy, and Institute of Fluid Dynamics and Applied Mathematics, University of Maryland, College Park

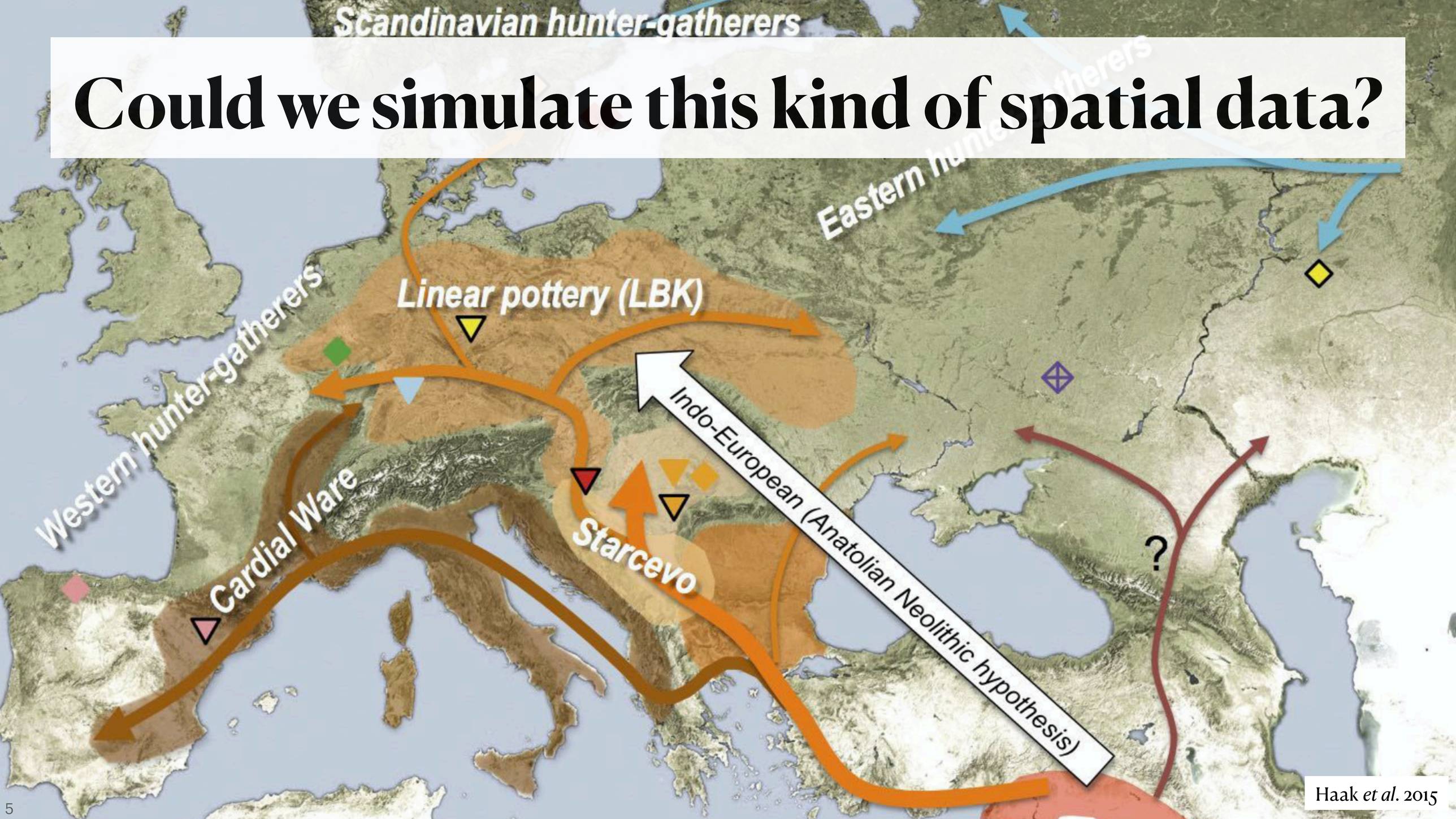
WHEN a species occupies a very large territory, local differentiation is usually noticeable in the form of geographical races. Each race may in turn consist of numerous colonies which are differentiated to a less noticeable extent. The

Can we build *much more elaborate* models which take geography into account?



Haak *et al.* 2015

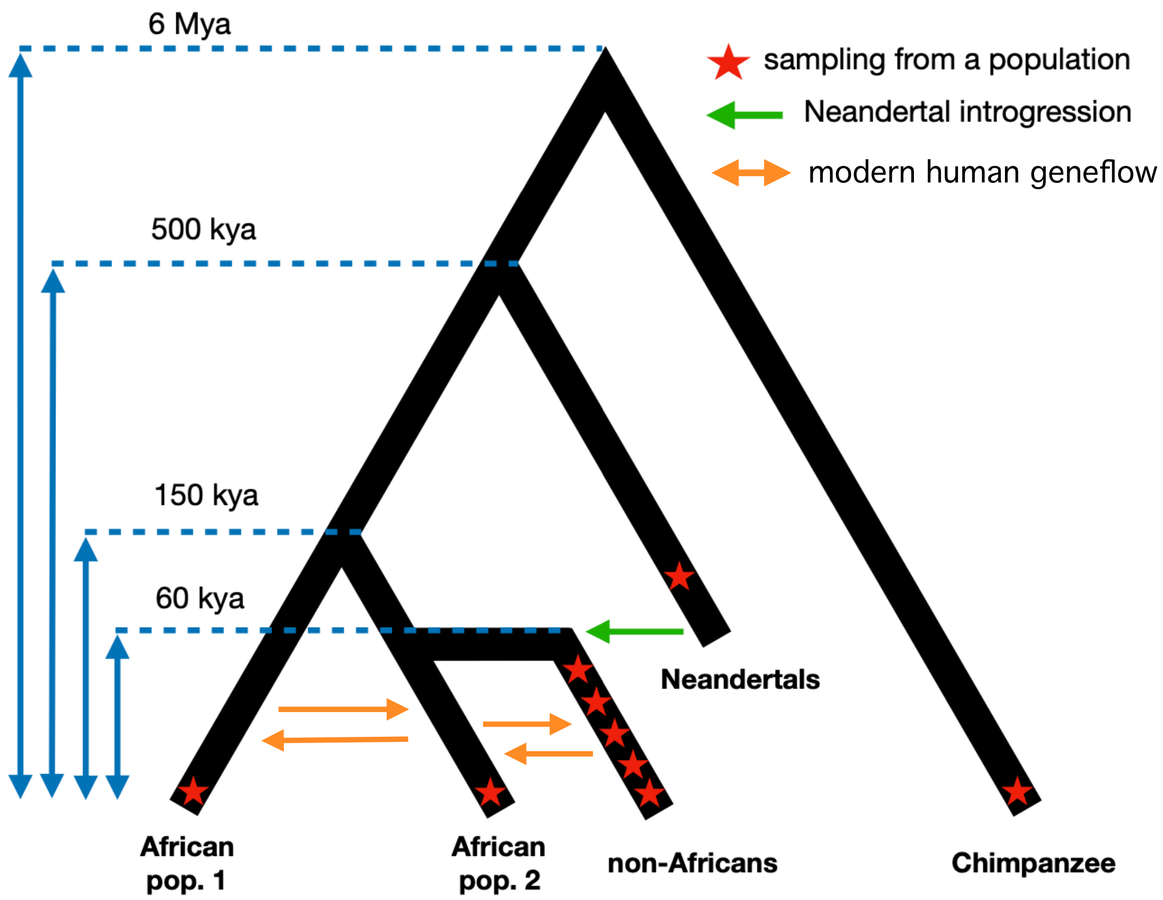
Could we simulate this kind of spatial data?



Haak *et al.* 2015

Traditional population genetic simulations

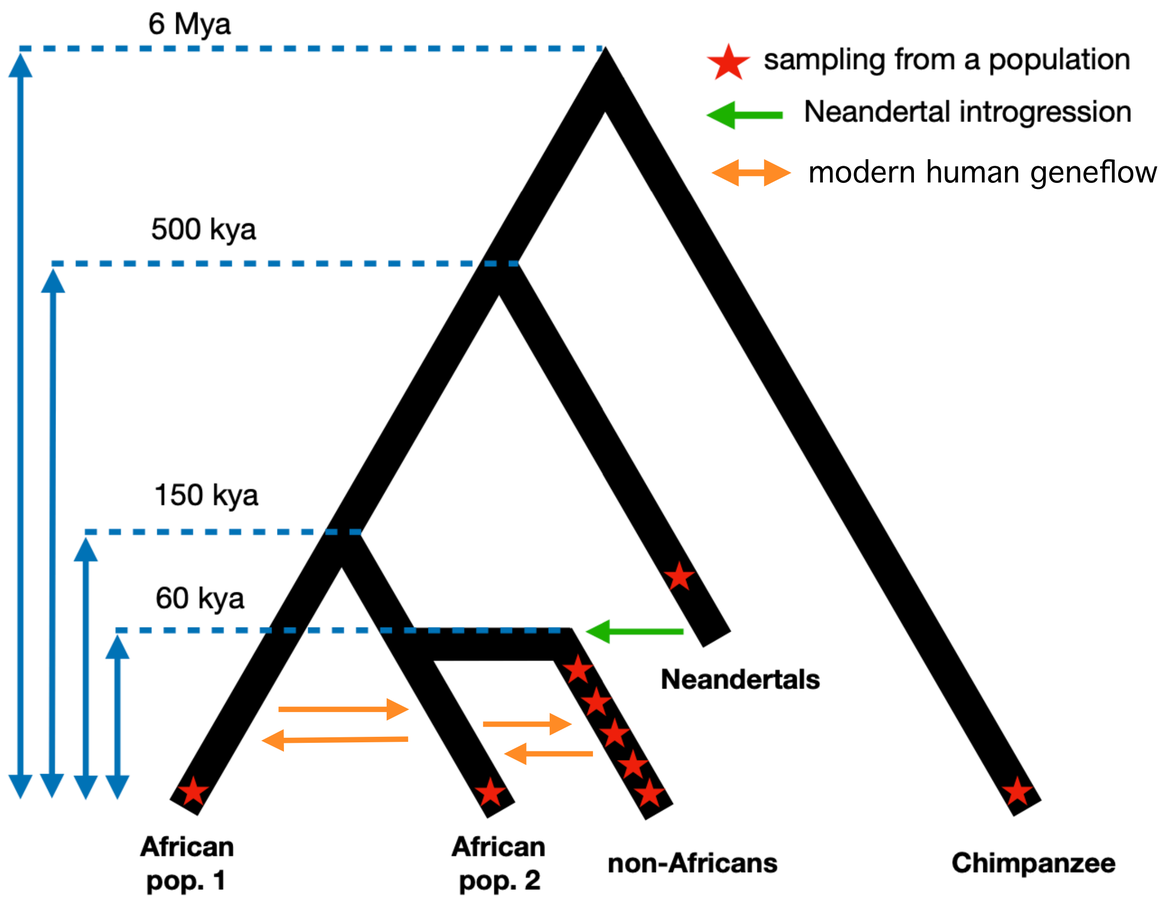
(*msprime*, *SLiM*)



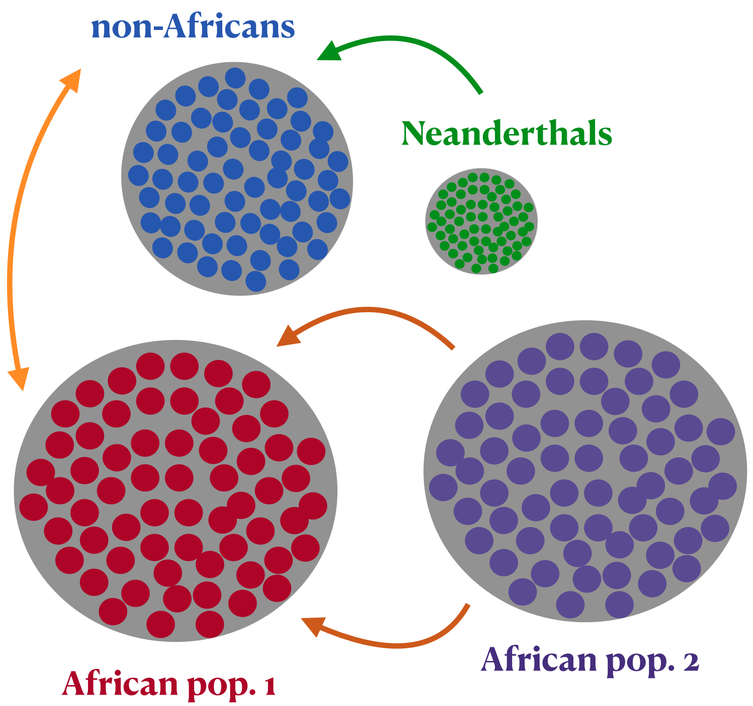
Petr *et al.* 2019

Traditional population genetic simulations

(*msprime*, *SLiM*)

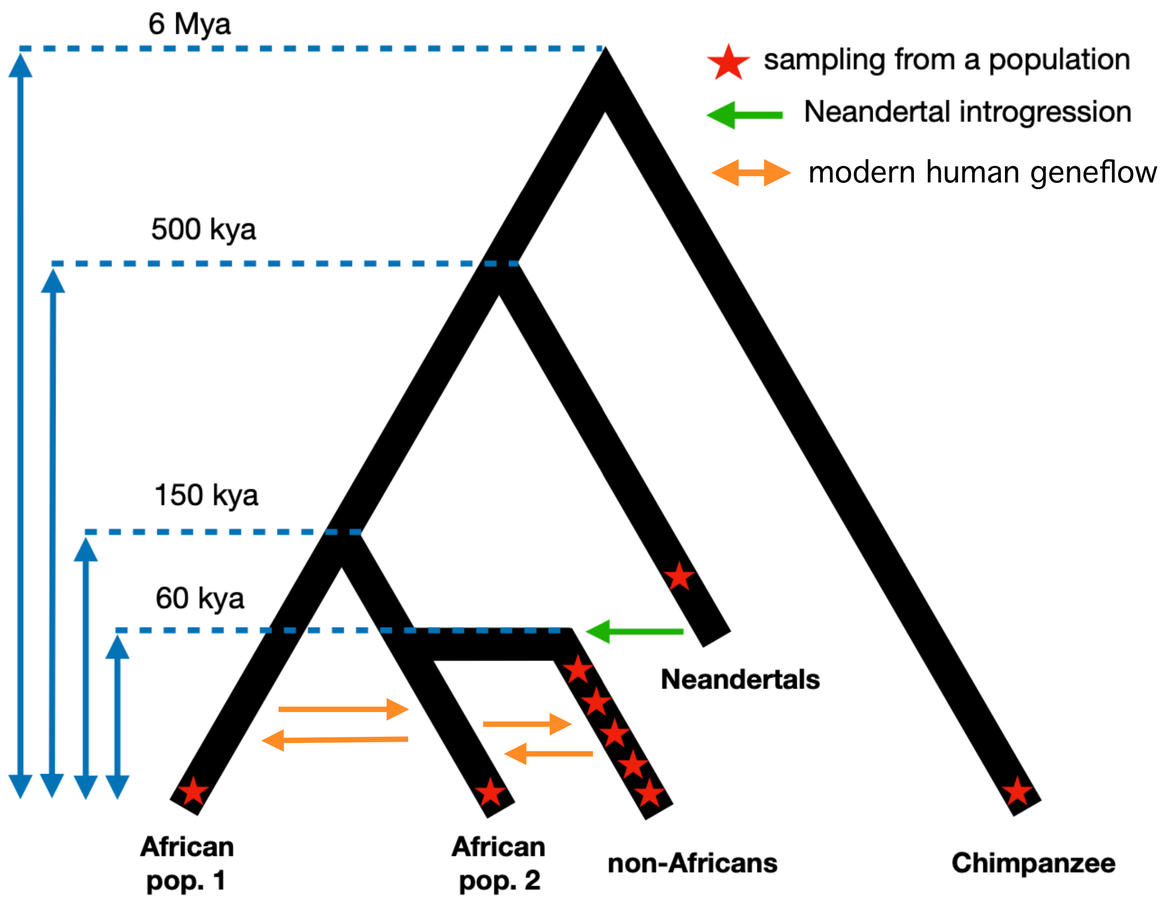


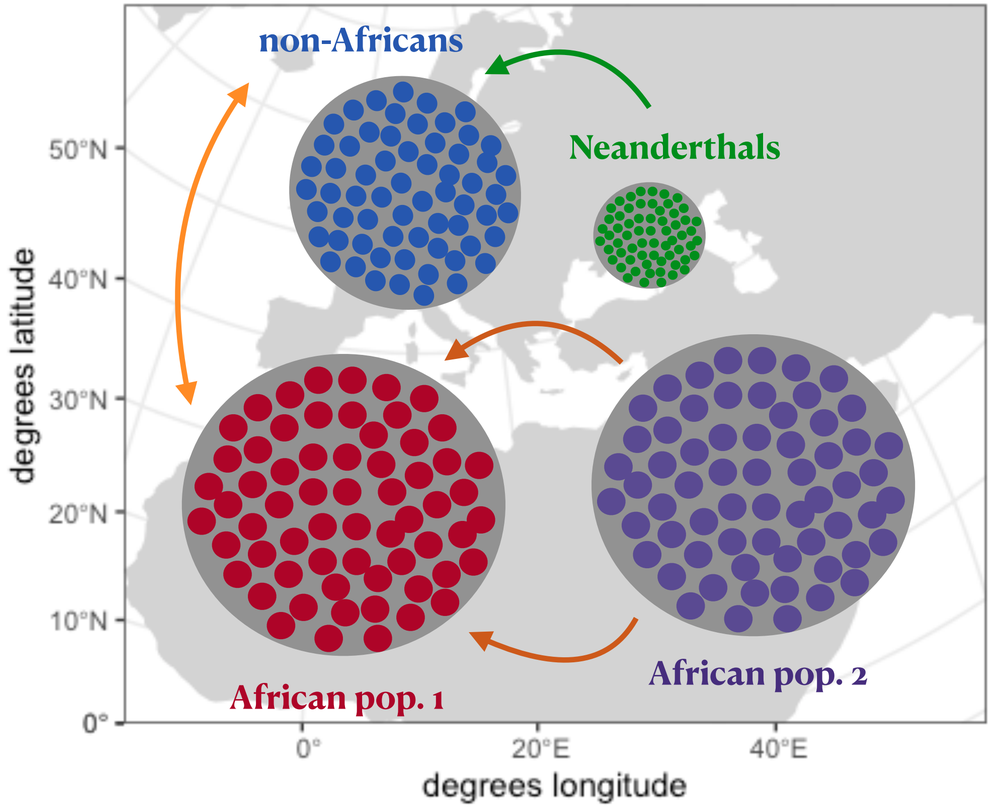
Petr *et al.* 2019



simulations of **massive** datasets
(assuming random mating)

How to add spatiality?

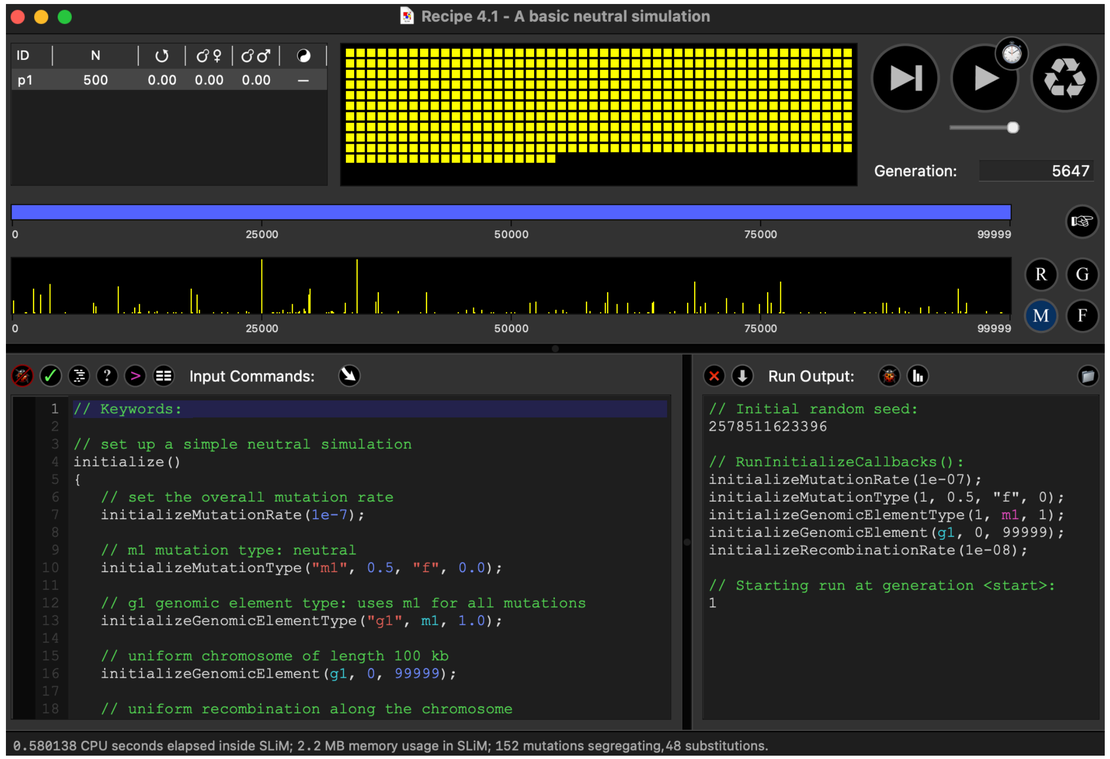




Petr *et al.* 2019

SLiM as the first building block?

- extremely popular and powerful
- custom programming language
- complex demographic models
- custom mating schemes

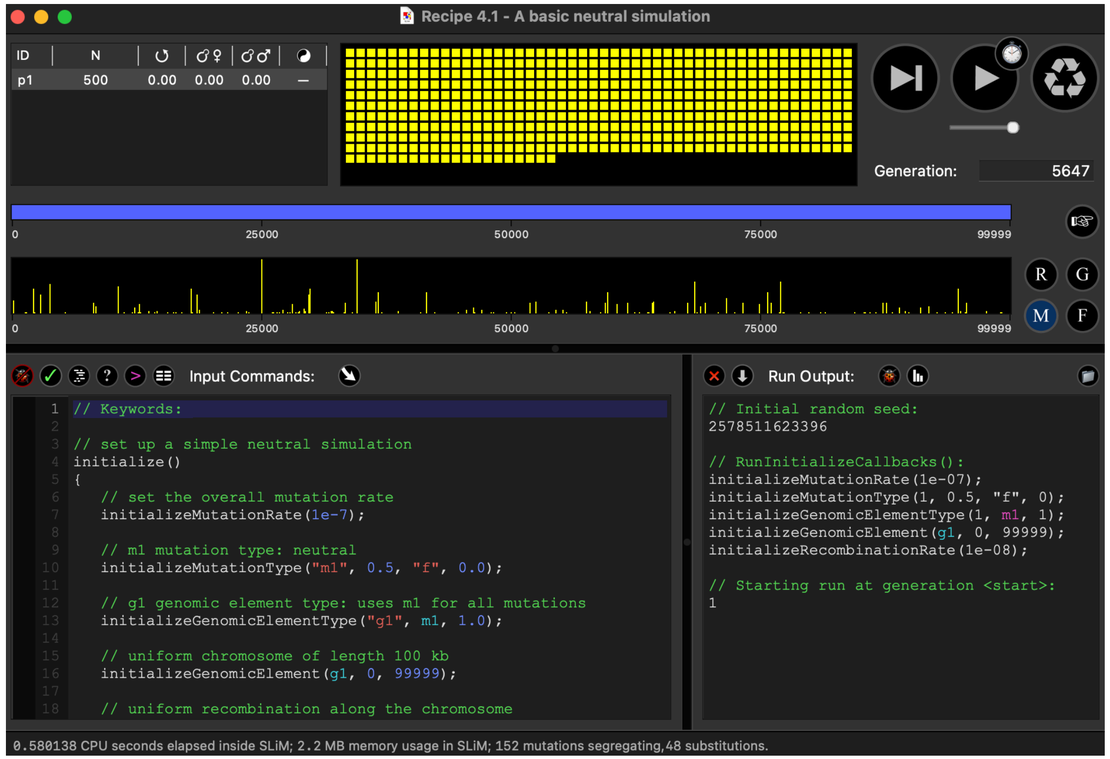


SLiM3: Haller *et al.* 2019

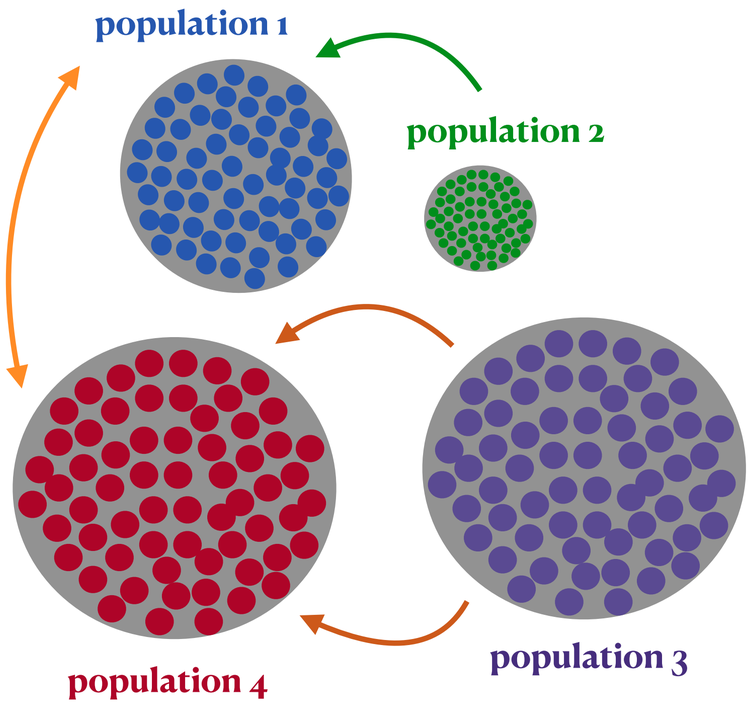
SLiM as the first building block?

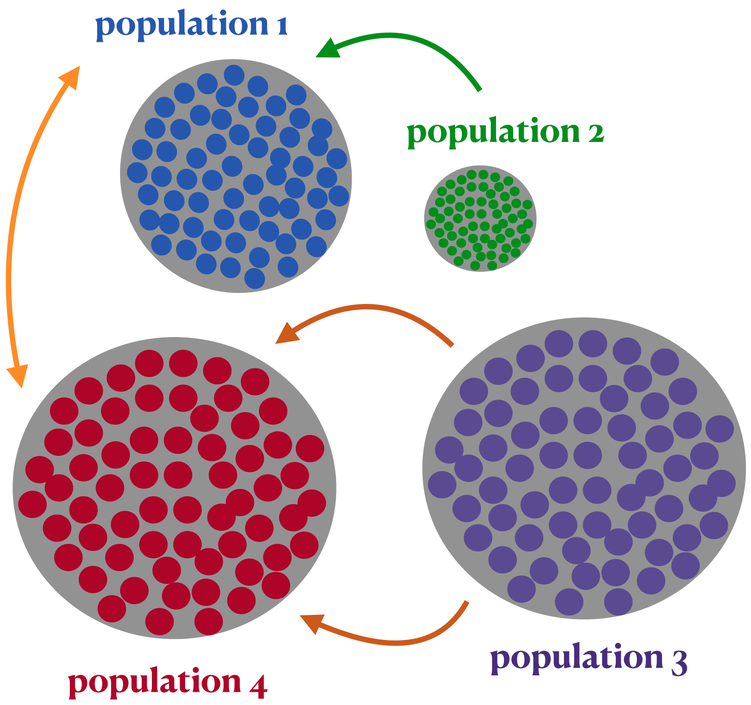
- extremely popular and powerful
- custom programming language
- complex demographic models
- custom mating schemes

... also continuous space!

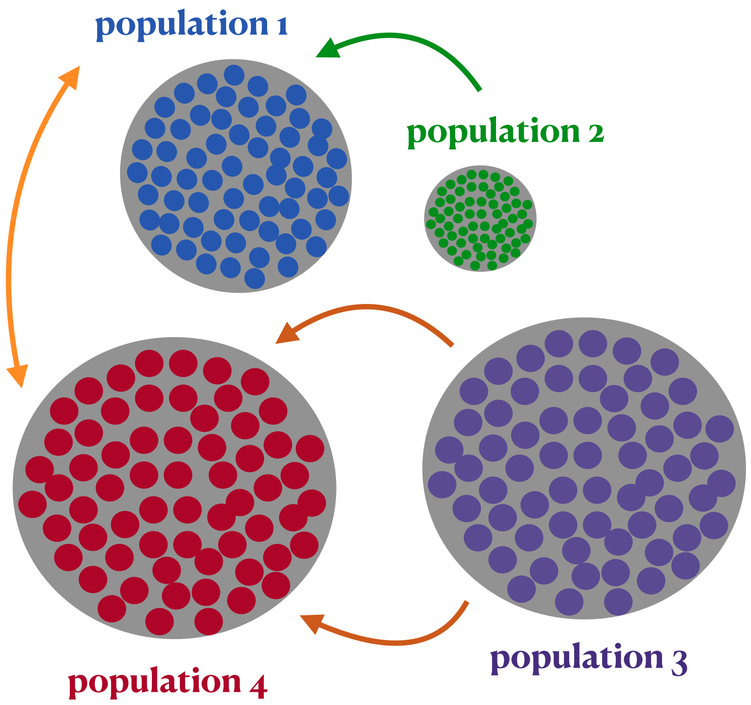


SLiM3: Haller *et al.* 2019

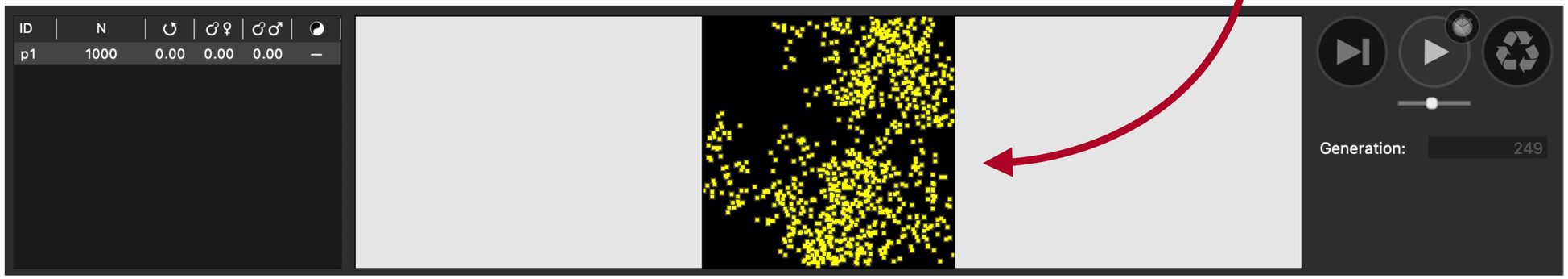

population 1
population 2
population 3
population 4




```
initializeSLiMOptions(dimensionality="xy");
[…]
<pop>.defineSpatialMap("world", "xy", <matrix>, <...>);
```

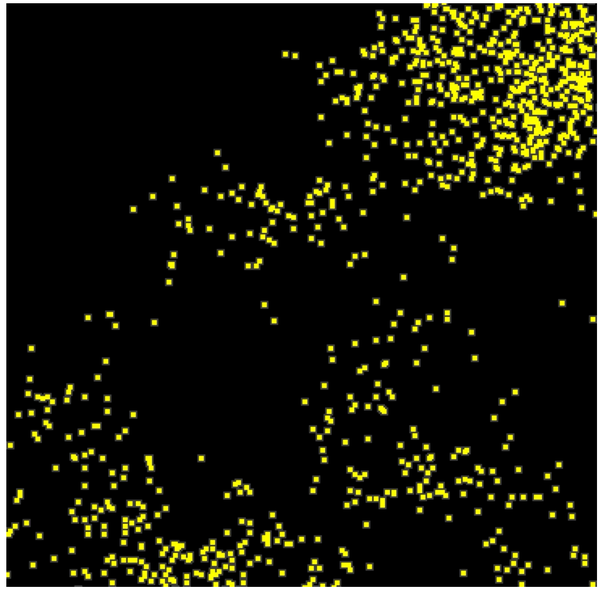



```
initializeSLiMOptions(dimensionality="xy");
[…]
<pop>.defineSpatialMap("world", "xy", <matrix>, <...>);
```

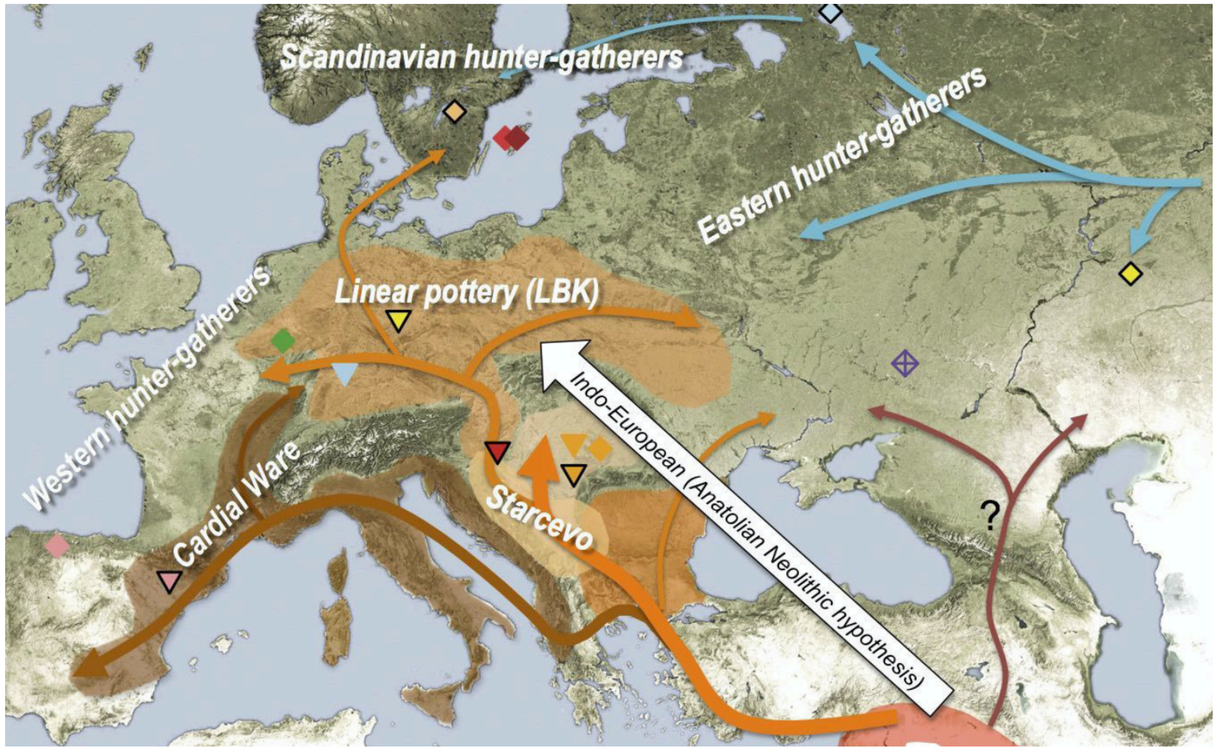


each individual has a position on a two-dimensional plane

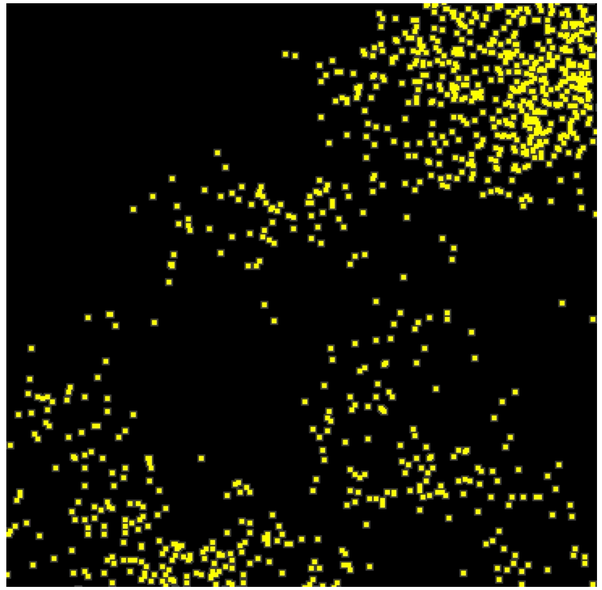
How do we get from here... ... to here?



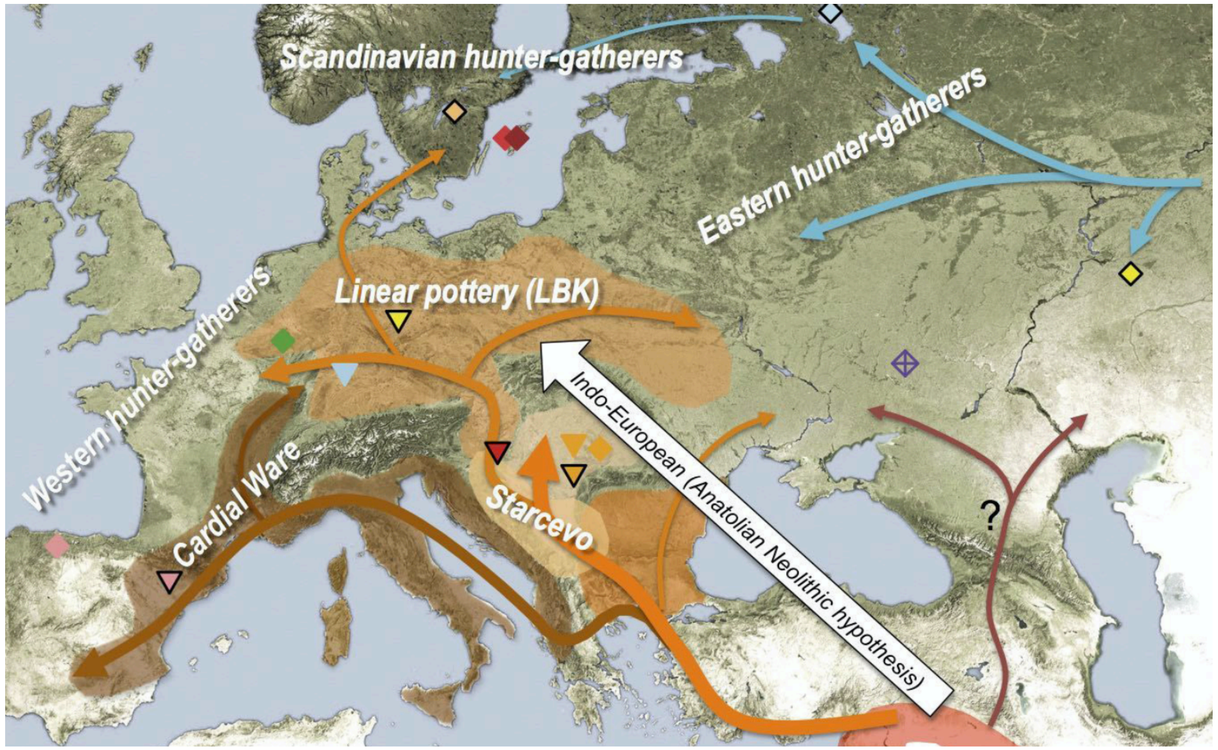
?



How do we get from here... ... to here?



?

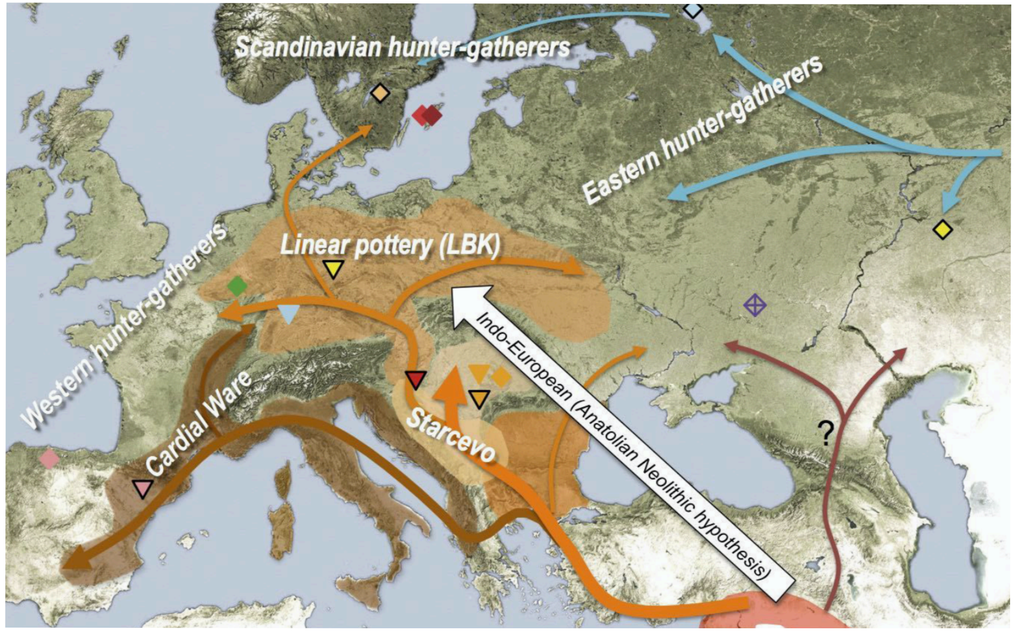


A programmable framework for spatial population genetics

A programmable framework for spatial population genetics

Design goals

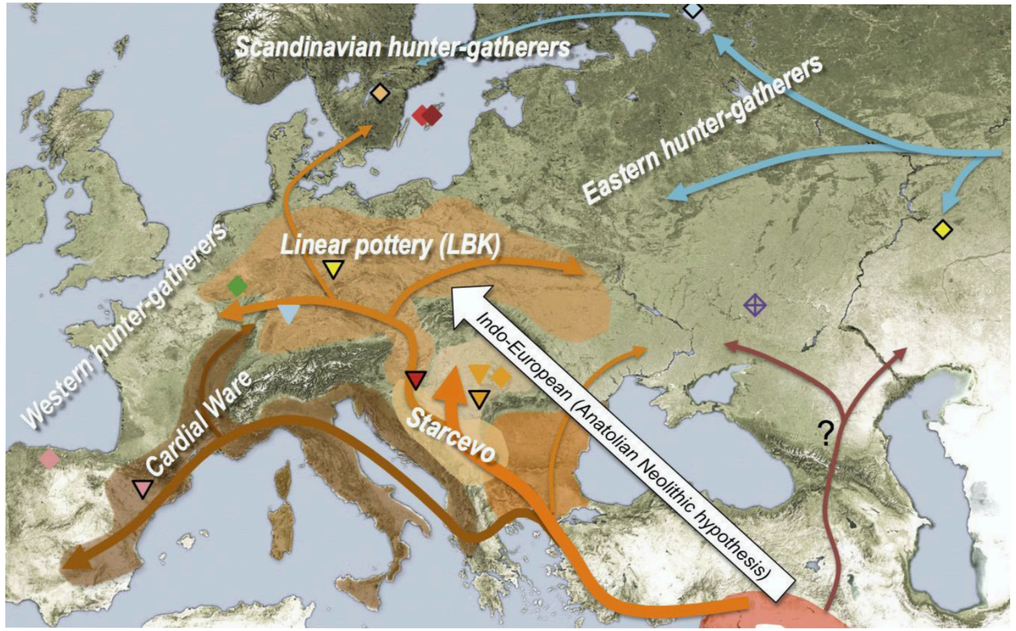
- create a "mini-language" for formal definition of
  - population ranges, movement, expansion, ...



A programmable framework for spatial population genetics

Design goals

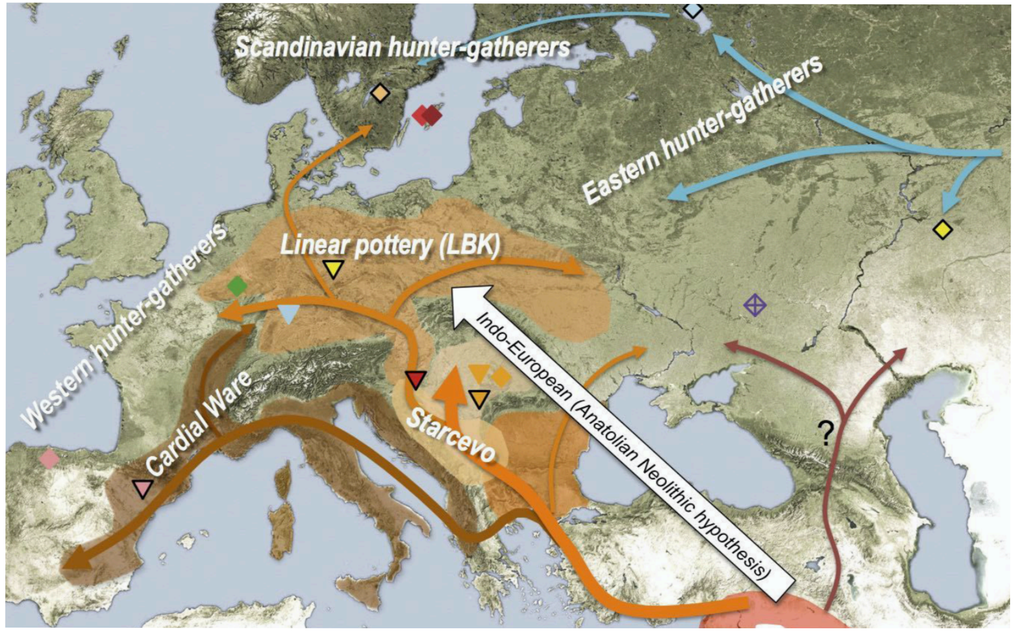
- create a "mini-language" for formal definition of
  - population ranges, movement, expansion, ...
- no geospatial know-how required



A programmable framework for spatial population genetics

Design goals

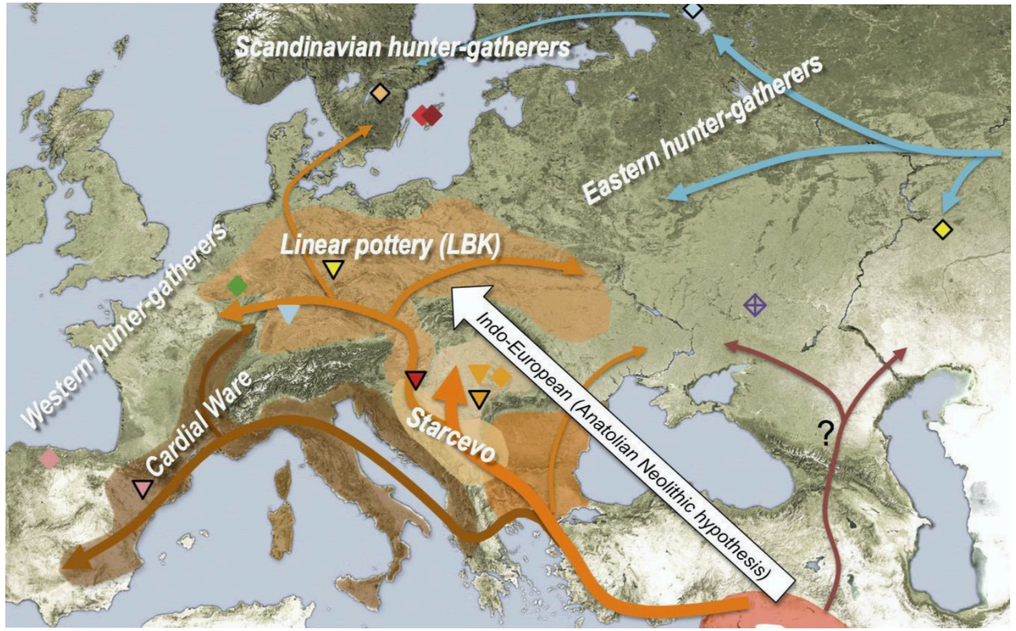
- create a "mini-language" for formal definition of
  - population ranges, movement, expansion, ...
- no geospatial know-how required
- use SLiM as a low level engine



A programmable framework for spatial population genetics

Design goals

- create a "mini-language" for formal definition of
  - population ranges, movement, expansion, ...
- no geospatial know-how required
- use SLiM as a low level engine
- implement in R



R package ***slendr***

www.slendr.net

Model definition & simulation

Model components

`population()`	Define a population and its original spatial range
`world()`	Define a world map for all spatial operations
`geneflow()`	Define a geneflow event
`region()`	Define a geographic region

Population dynamics

`move()`	Move the population to a new location in a given amount of time
`expand()`	Expand the population range
`shrink()`	Shrink the population range
`boundary()`	Update the population range
`resize()`	Resize the population size
`dispersal()`	Change dispersal parameters

Manipulation of spatial objects

`join()`	Merge two spatial `slendr` objects into one
`overlap()`	Generate the overlap of two `slendr` objects
`subtract()`	Generate the difference between two `slendr` objects
`reproject()`	Reproject coordinates between coordinate systems
`distance()`	Calculate the distance between a pair of spatial boundaries
`area()`	Calculate the area covered by the given slendr object
`dimensions()`	Return the dimensions of the world map

Model visualization and diagnostics

`plot(<slendr>)`	Plot `slendr` geographic features on a map
`plot_graph()`	Plot geneflow graph based on given model configuration
`plot_ancestry()`	Plot simulated ancestry proportions
`animate()`	Animate the simulated population dynamics
`explore()`	Open an interactive browser of the spatial model

Compiling and running spatial models

`compile()`	Compile the spatial demographic model
`sampling()`	Define sampling events at specified times for a given set of populations
`slim()`	Run a slendr model as a SLiM script
`read()`	Read a previously serialized model configuration
`script()`	Substitute variables in a template SLiM script

Tree sequence analysis

Tree sequence loading and processing

`ts_load()`	Load a tree sequence file produced by a given model
`ts_recapitate()`	Recapitate the tree sequence
`ts_simplify()`	Simplify the tree sequence down to a given set of individuals
`ts_mutate()`	Add mutations to the given tree sequence
`ts_coalesced()`	Check that all trees in the tree sequence are fully coalesced

Extracting genotypes from tree sequences

`ts_genotypes()`	Extract genotype table from the tree sequence
`ts_eigenstrat()`	Extract genotypes from the tree sequence in the EIGENSTRAT format
`ts_vcf()`	Save genotypes from the tree sequenceas a VCF file

Accessing tree sequence components

`ts_data()`	Extract combined annotated table of individuals and nodes
`ts_ancestors()`	Infer spatio-temporal ancestral history for given nodes/individuals
`ts_individuals()` `ts_edges()` `ts_nodes()`	Get the table of individuals/nodes/edges from the tree sequence
`ts_samples()`	Extract names and times of individuals scheduled for sampling
`ts_tree()`	Get a tree from a given tree sequence
`ts_draw()`	Plot a graphical representation of a single tree

Tree sequence statistics

`ts_f2()` `ts_f3()` `ts_f4()` `ts_f4ratio()`	Calculate the f2, f3, f4, and f4-ratio statistics
`ts_afs()`	Compute the allele frequency spectrum (AFS)
`ts_fst()` `ts_divergence()`	Calculate pairwise statistics between sets of individuals
`ts_diversity()`	Calculate diversity in given sets of individuals
`ts_tajima()`	Calculate Tajima's D for given sets of individuals
`ts_segregating()`	Calculate the density of segregating sites for the given sets of individuals

Model definition & simulation

Model components

`population()`	Define a population and its original spatial range
`world()`	Define a world map for all spatial operations
`geneflow()`	Define a geneflow event
`region()`	Define a geographic region

Population dynamics

`move()`	Move the population to a new location in a given amount of time
`expand()`	Expand the population range
`shrink()`	Shrink the population range
`boundary()`	Update the population range
`resize()`	Resize the population size
`dispersal()`	Change dispersal parameters

Manipulation of spatial objects

`join()`	Merge two spatial `slendr` objects into one
`overlap()`	Generate the overlap of two `slendr` objects
`subtract()`	Generate the difference between two `slendr` objects
`reproject()`	Reproject coordinates between coordinate systems
`distance()`	Calculate the distance between a pair of spatial boundaries
`area()`	Calculate the area covered by the given slendr object
`dimensions()`	Return the dimensions of the world map

Model visualization and diagnostics

`plot(<slendr>)`	Plot `slendr` geographic features on a map
`plot_graph()`	Plot geneflow graph based on given model configuration
`plot_ancestry()`	Plot simulated ancestry proportions
`animate()`	Animate the simulated population dynamics
`explore()`	Open an interactive browser of the spatial model

Compiling and running spatial models

`compile()`	Compile the spatial demographic model
`sampling()`	Define sampling events at specified times for a given set of populations
`slim()`	Run a slendr model as a SLiM script
`read()`	Read a previously serialized model configuration
`script()`	Substitute variables in a template SLiM script

Tree sequence analysis

Tree sequence loading and processing

`ts_load()`	Load a tree sequence file produced by a given model
`ts_recapitate()`	Recapitate the tree sequence
`ts_simplify()`	Simplify the tree sequence down to a given set of individuals
`ts_mutate()`	Add mutations to the given tree sequence
`ts_coalesced()`	Check that all trees in the tree sequence are fully coalesced

Extracting genotypes from tree sequences

`ts_genotypes()`	Extract genotype table from the tree sequence
`ts_eigenstrat()`	Extract genotypes from the tree sequence in the EIGENSTRAT format
`ts_vcf()`	Save genotypes from the tree sequenceas a VCF file

Accessing tree sequence components

`ts_data()`	Extract combined annotated table of individuals and nodes
`ts_ancestors()`	Infer spatio-temporal ancestral history for given nodes/individuals
`ts_individuals()` `ts_edges()` `ts_nodes()`	Get the table of individuals/nodes/edges from the tree sequence
`ts_samples()`	Extract names and times of individuals scheduled for sampling
`ts_tree()`	Get a tree from a given tree sequence
`ts_draw()`	Plot a graphical representation of a single tree

Tree sequence statistics

`ts_f2()` `ts_f3()` `ts_f4()` `ts_f4ratio()`	Calculate the f2, f3, f4, and f4-ratio statistics
`ts_afs()`	Compute the allele frequency spectrum (AFS)
`ts_fst()` `ts_divergence()`	Calculate pairwise statistics between sets of individuals
`ts_diversity()`	Calculate diversity in given sets of individuals
`ts_tajima()`	Calculate Tajima's D for given sets of individuals
`ts_segregating()`	Calculate the density of segregating sites for the given sets of individuals

slendr workflow demo

interactive RStudio session in the browser:

github.com/bodkan/slendr

click on launch binder

(no installation needed!)

1. Define a spatial context

(section of the world which will be occupied by populations)

1. Define a spatial context

(section of the world which will be occupied by populations)

```
map ← world(
  xrange = c(-15, 60),
  yrange = c(20, 65),
  landscape = "naturalearth",
  crs = "EPSG:3035"
)
```

(longitude, latitude)

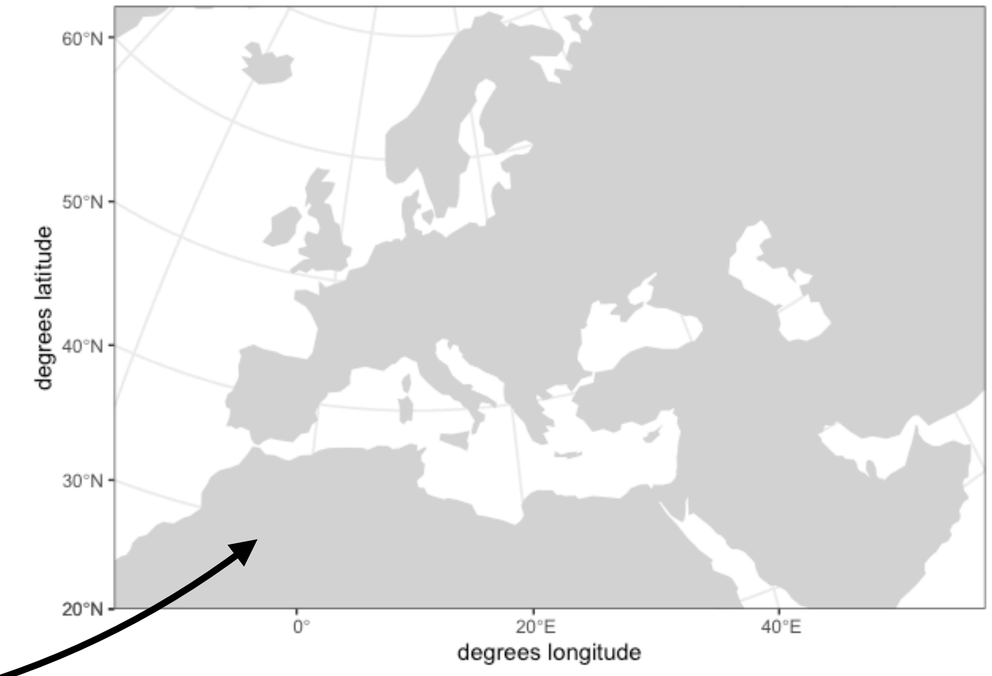
1. Define a spatial context

(section of the world which will be occupied by populations)

```
map ← world(
  xrange = c(-15, 60),
  yrange = c(20, 65),
  landscape = "naturalearth",
  crs = "EPSG:3035"
)
```

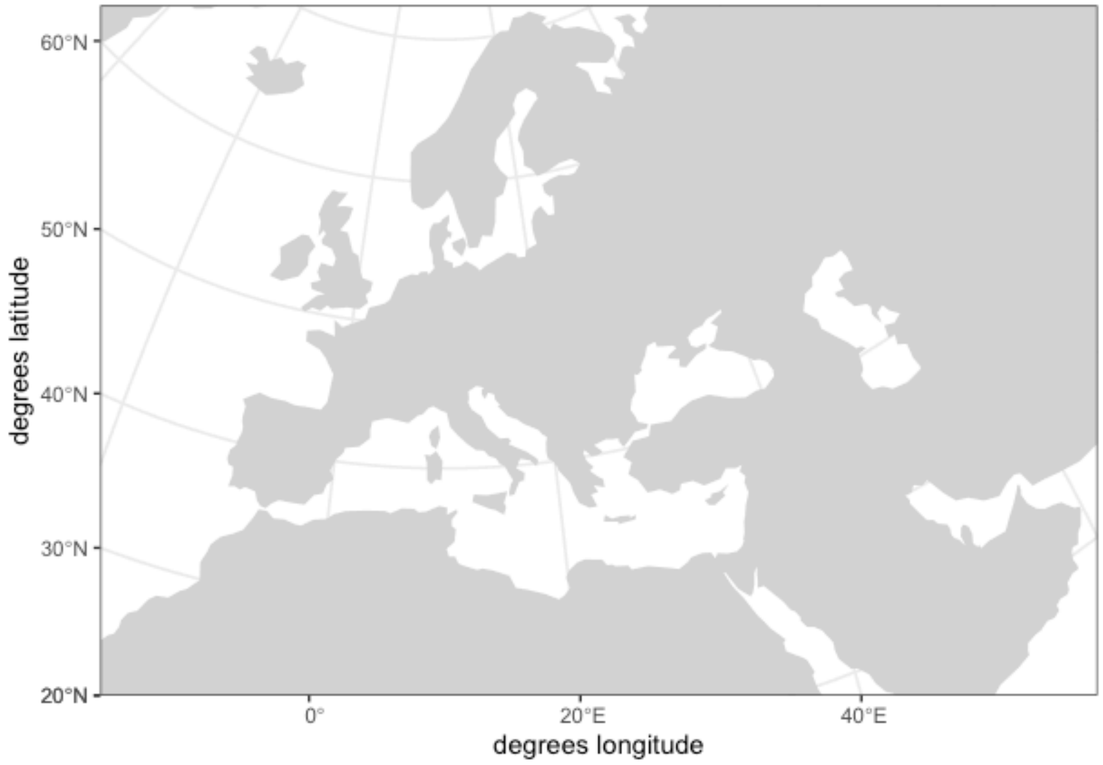
(longitude, latitude)

```
plot(map)
```



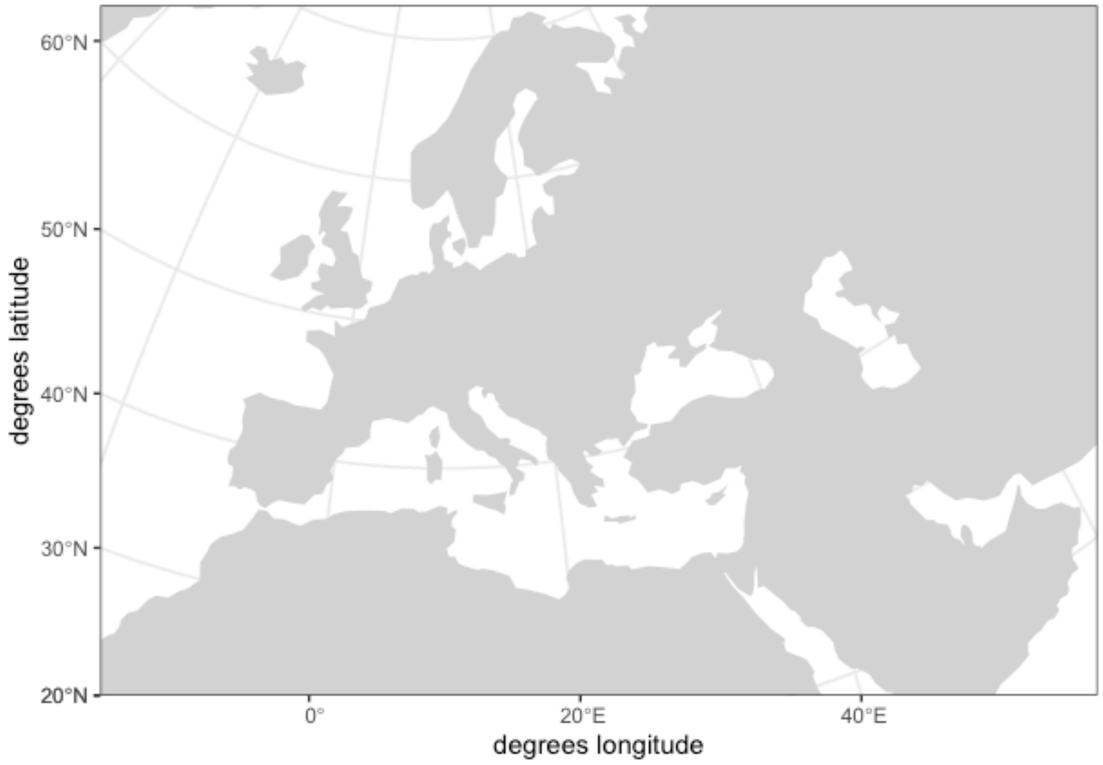
(using freely available *Natural Earth* cartographic data)

2. Population boundaries



2. Population boundaries

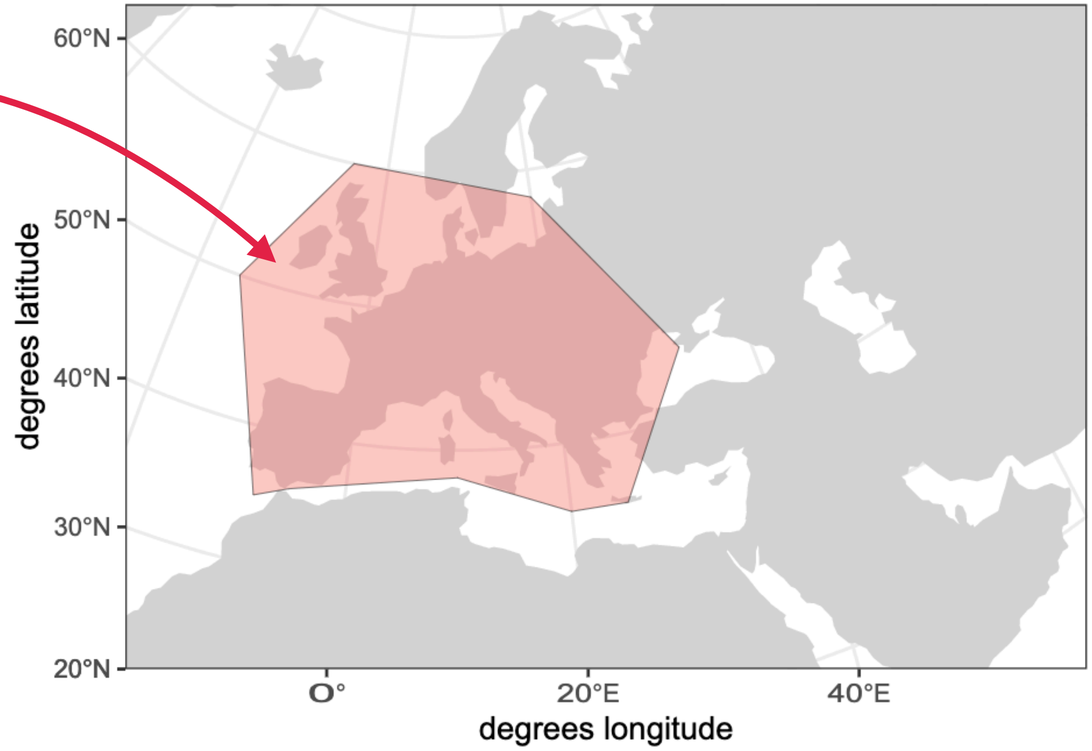
```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```



2. Population boundaries

```r
europe <- region("Europe", map = map, polygon = list(
    c(-8, 35), c(10, 38), c(20, 35), c(25, 35),
    c(32, 45), c(20, 58), c(-5, 60), c(-15, 50))
)
```

```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```

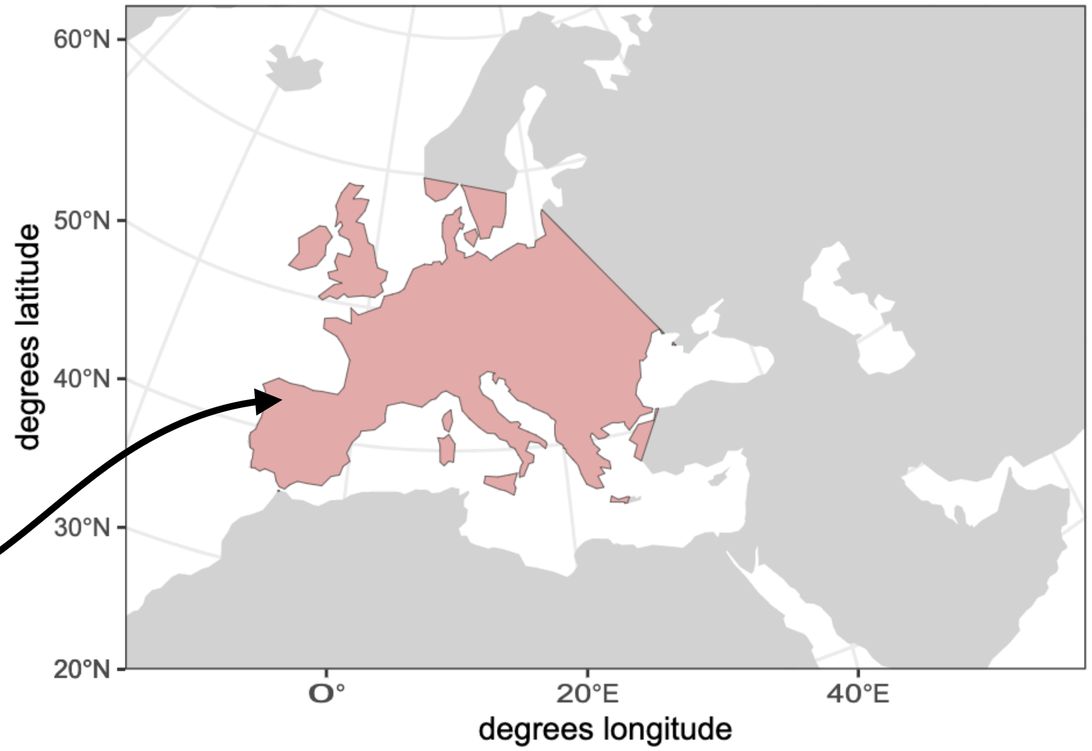


2. Population boundaries

```r
europe <- region("Europe", map = map, polygon = list(
    c(-8, 35), c(10, 38), c(20, 35), c(25, 35),
    c(32, 45), c(20, 58), c(-5, 60), c(-15, 50))
)
```

```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```

```r
plot(eur)
```



2. Population boundaries

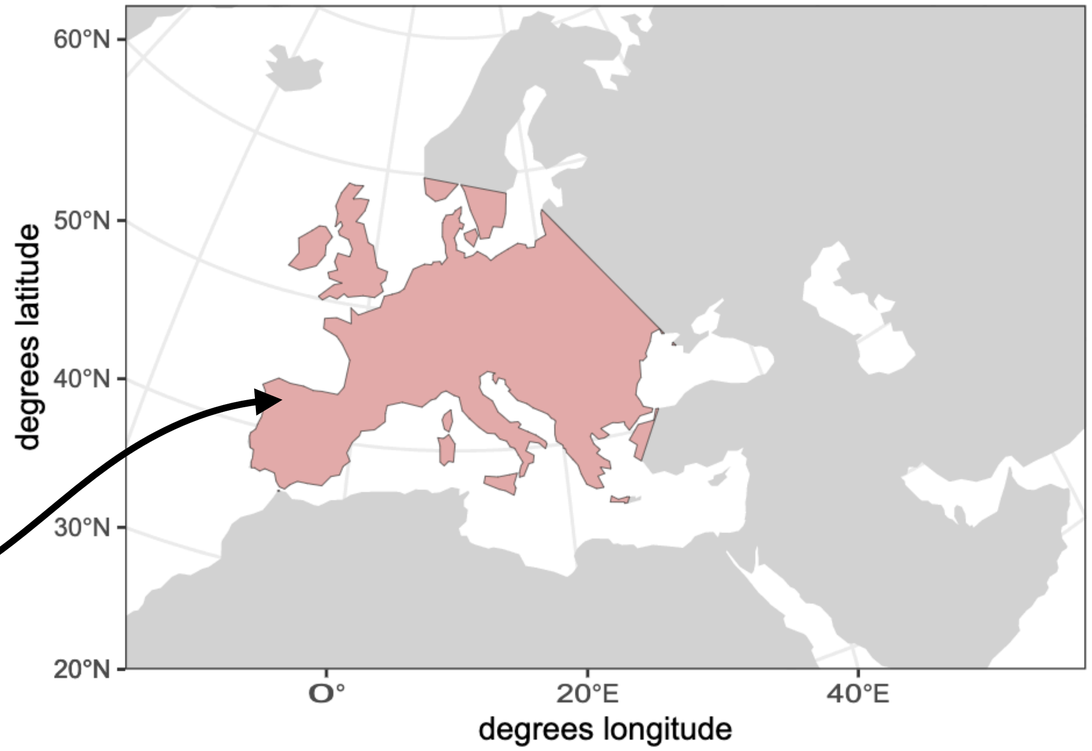
Everything is stored in a vectorised form!

```
> eur$geometry[[1]]
POLYGON ((2676851 1520500, 4321000 1657812, 5240847 1387400, 5695390 14617
22, 6029973 2686840, 4910944 3919000, 3489173 4188384, 2568327 3290814, 26
76851 1520500))
```

```r
europe <- region("Europe", map = map, polygon = list(
    c(-8, 35), c(10, 38), c(20, 35), c(25, 35),
    c(32, 45), c(20, 58), c(-5, 60), c(-15, 50))
)
```

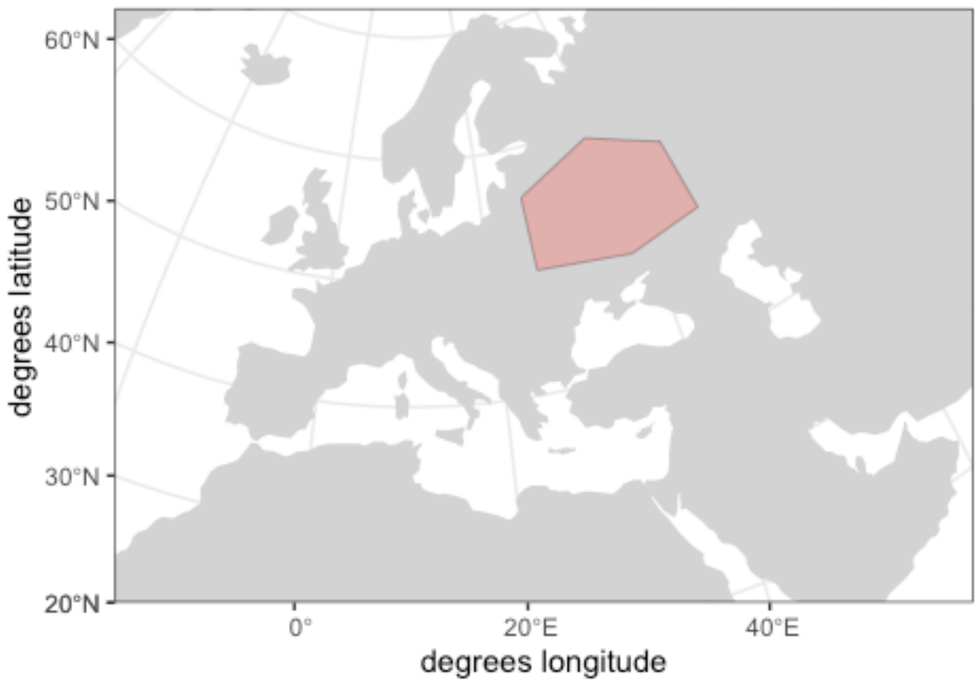
```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```

```r
plot(eur)
```



3. Population displacement

```r
yam <- population( # Yamnaya steppe population
  name = "YAM", time = 7000, N = 1000,
  parent = ehg, polygon = list(
    c(26, 50), c(38, 49), c(48, 50),
    c(48, 56), c(38, 59), c(26, 56)
  )
)
```



3. Population displacement

```r
yam <- population( # Yamnaya steppe population
  name = "YAM", time = 7000, N = 1000,
  parent = ehg, polygon = list(
    c(26, 50), c(38, 49), c(48, 50),
    c(48, 56), c(38, 59), c(26, 56)
  )
)
```

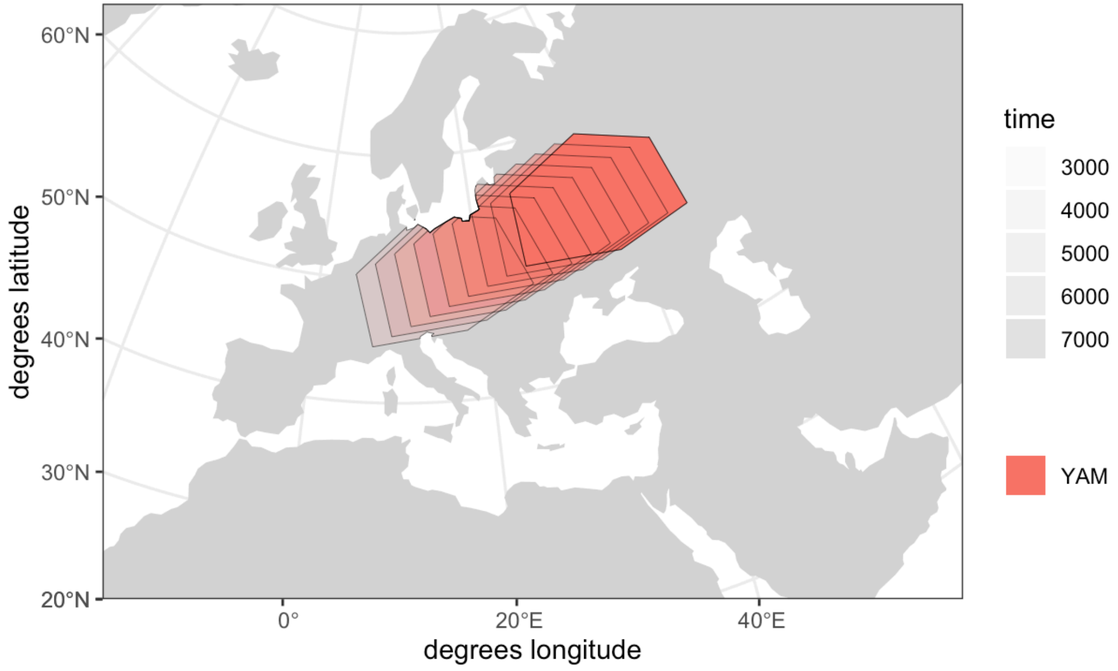
3. Population displacement

```r
yam <- population( # Yamnaya steppe population
  name = "YAM", time = 7000, N = 1000,
  parent = ehg, polygon = list(
    c(26, 50), c(38, 49), c(48, 50),
    c(48, 56), c(38, 59), c(26, 56)
  )
) %>%
  move(trajectory = c(15, 50), start = 5000, end = 3000)
```

3. Population displacement

```r
yam <- population( # Yamnaya steppe population
  name = "YAM", time = 7000, N = 1000,
  parent = ehg, polygon = list(
    c(26, 50), c(38, 49), c(48, 50),
    c(48, 56), c(38, 59), c(26, 56)
  )
) %>%
  move(trajectory = c(15, 50), start = 5000, end = 3000)
```

```r
plot(yam)
```

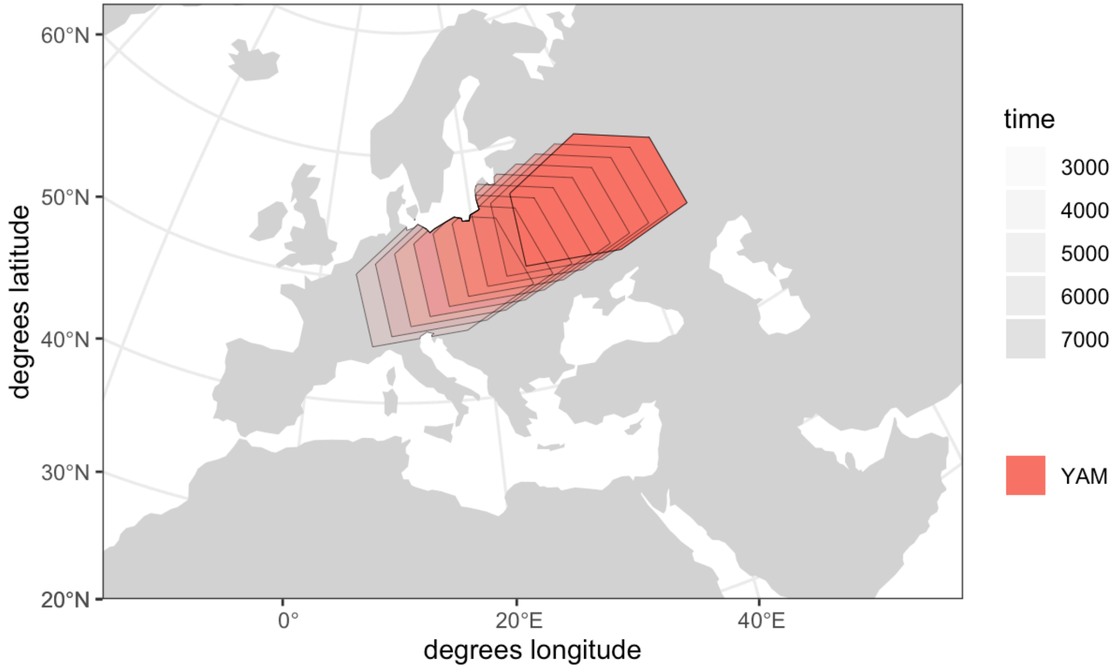


3. Population displacement

```r
yam <- population( # Yamnaya steppe population
  name = "YAM", time = 7000, N = 1000,
  parent = ehg, polygon = list(
    c(26, 50), c(38, 49), c(48, 50),
    c(48, 56), c(38, 59), c(26, 56)
  )
) %>%
  move(trajectory = c(15, 50), start = 5000, end = 3000)
```

```r
plot(yam)
```

```
> yam
slendr 'population' object
--------------------------
name: YAM
stays until the end of the simulation
habitat: terrestrial

population history overview:
  - time 7000: split from EHG
  - time 5000-3000: movement across a landscape

number of spatial maps: 10
map: internal coordinate reference system: EPSG 3035
```

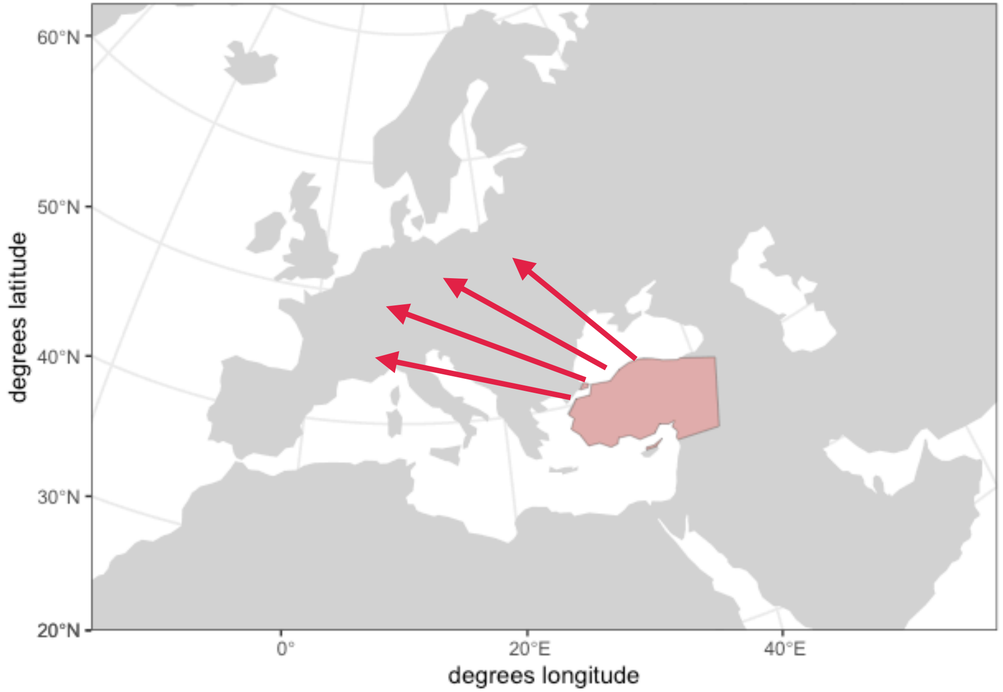


4. Population range expansion

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
)
```

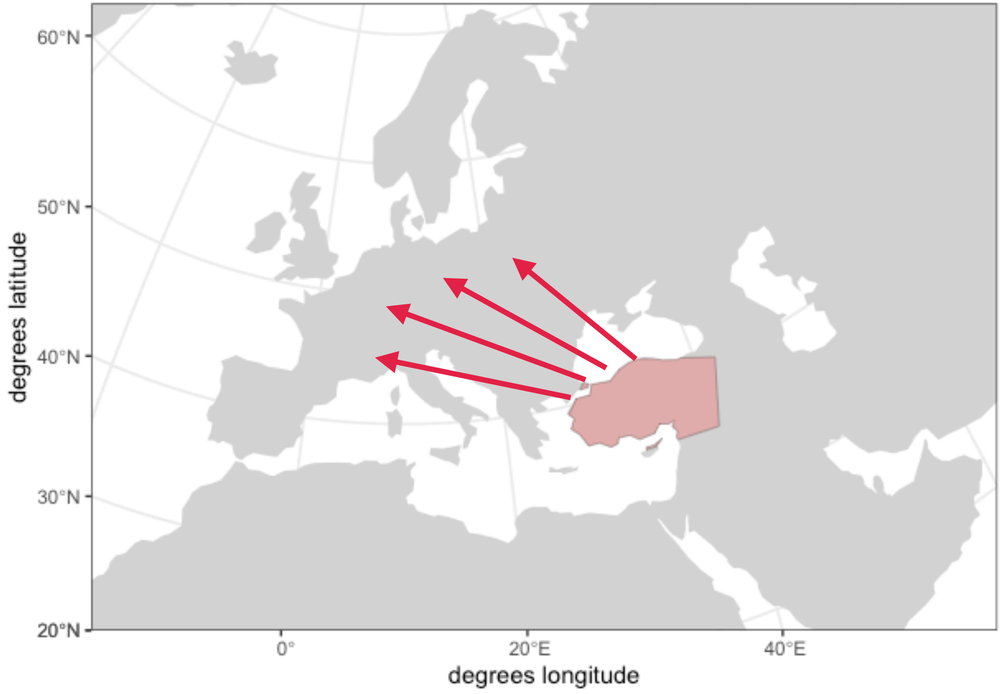
4. Population range expansion

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
)
```



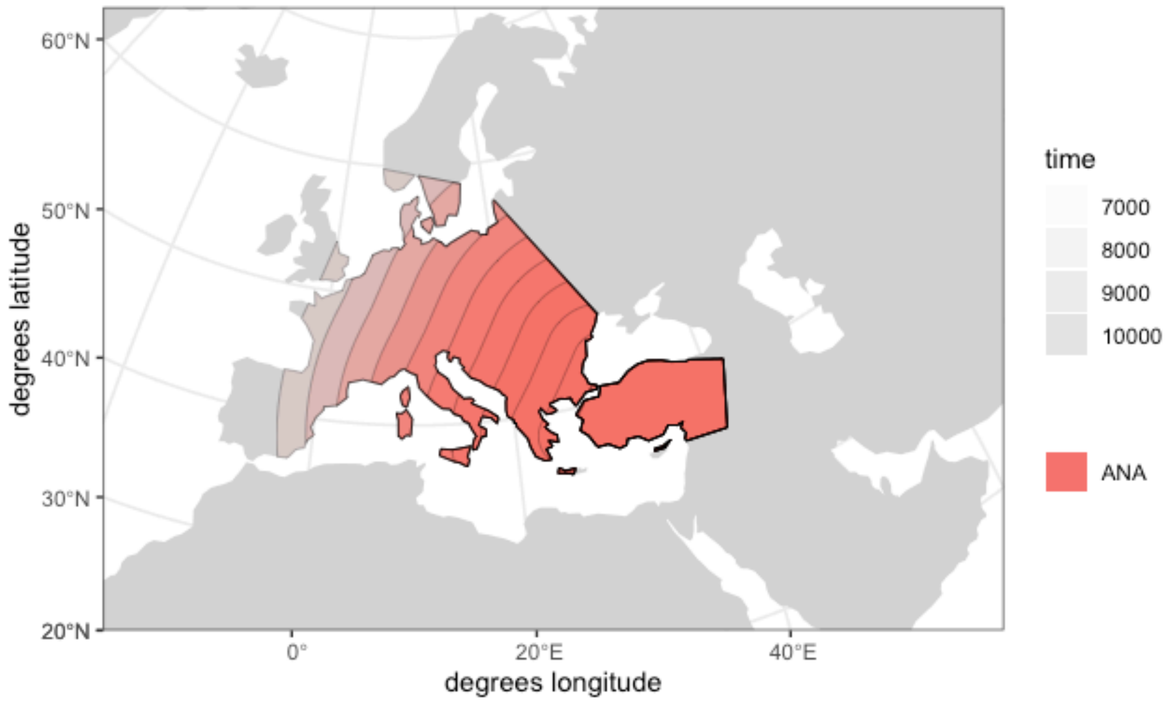
4. Population range expansion

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
) %>%
  expand(by = 2500e3, start = 10000, end = 7000,polygon = join(europe, anatolia))
```



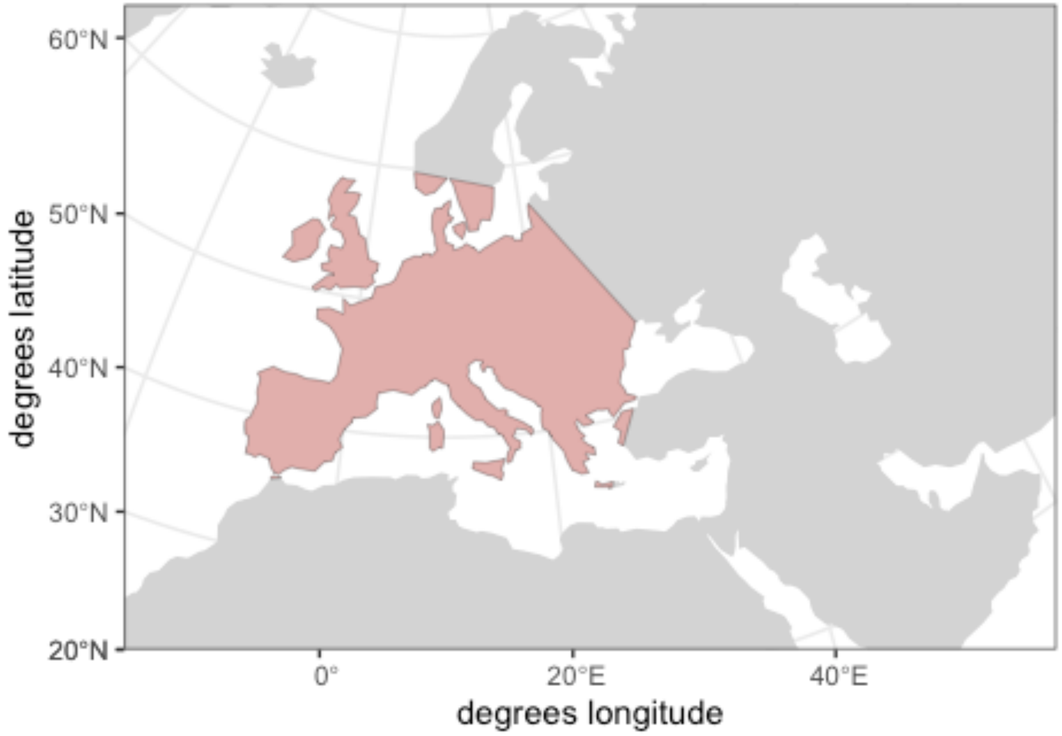
4. Population range expansion

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
) %>%
  expand(by = 2500e3, start = 10000, end = 7000,polygon = join(europe, anatolia))
```



5. Gene flow

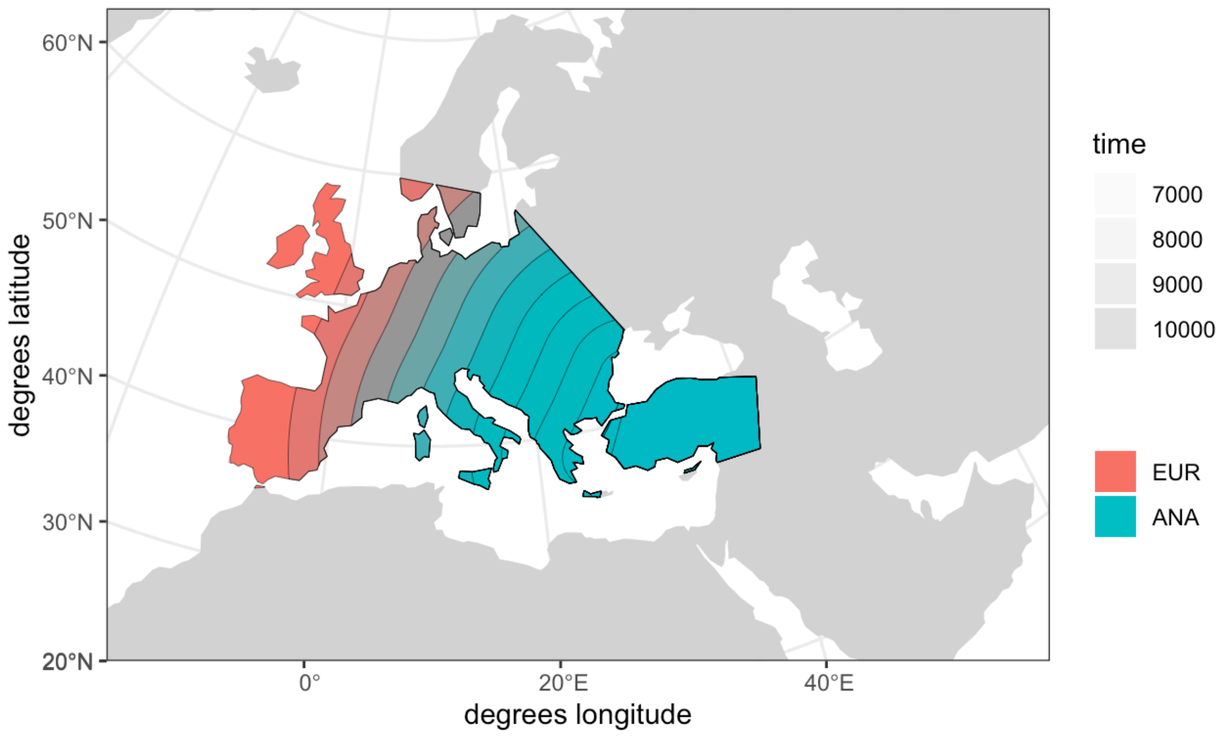
```
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```



5. Gene flow

```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
) %>%
  expand(by = 2500e3, start = 10000, end = 7000, polygon = join(europe, anatolia))
```

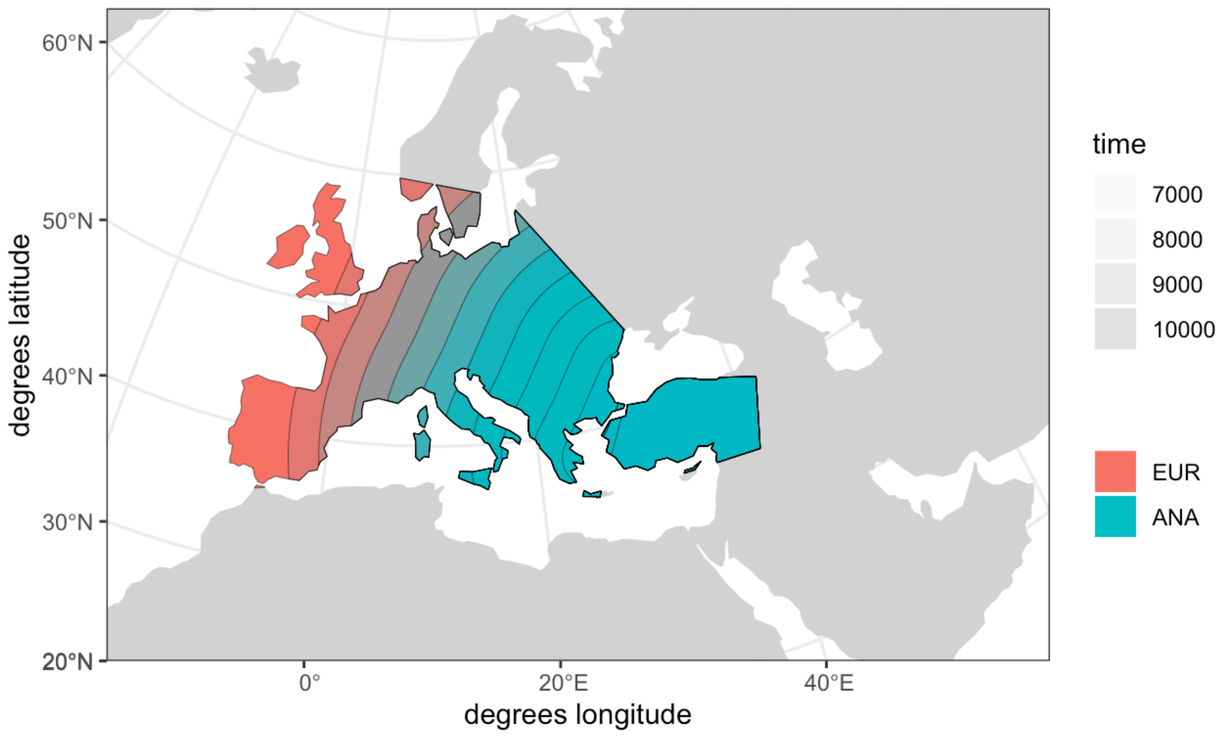


5. Gene flow

```r
eur <- population( # European population
  name = "EUR", time = 10000, N = 1000,
  map = map, polygon = europe
)
```

```r
ana <- population( # Anatolian farmers
  name = "ANA", time = 10000, N = 1000,
  map = map, polygon = anatolia
) %>%
  expand(by = 2500e3, start = 10000, end = 7000, polygon = join(europe, anatolia))
```

```r
gf <- geneflow(from = ana, to = eur, rate = 0.5, start = 8000, end = 6000)
```



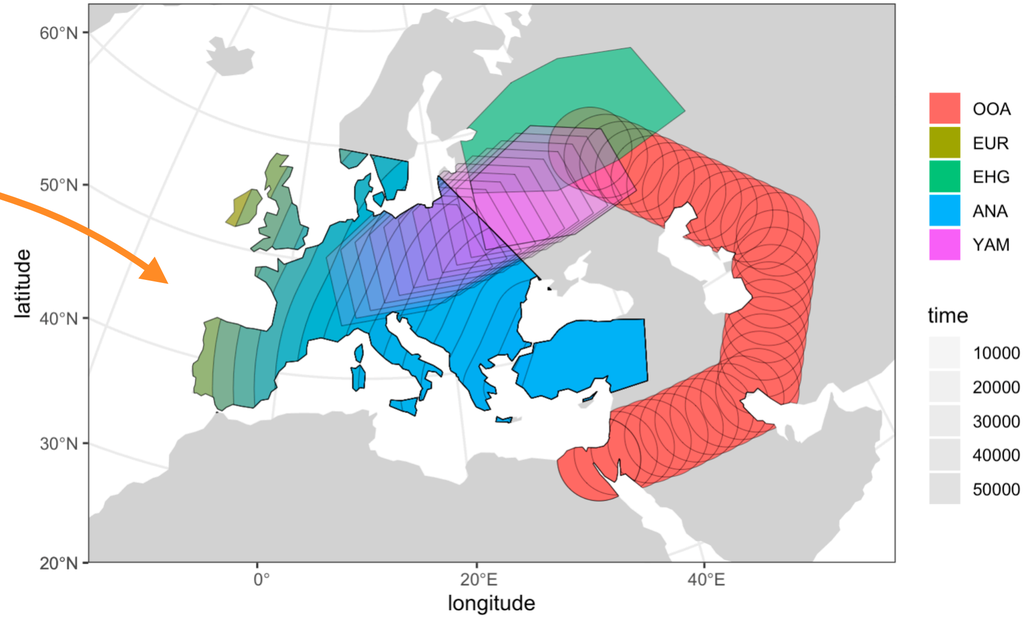
6. Model compilation

```
model <- compile_model(
  populations = list(ooa, ehg, eur, ana, yam), gene_flow = gf,
  generation_time = 30, resolution = 10e3,
  competition = 130e3, mating = 100e3, dispersal = 70e3,
)
```

6. Model compilation

we created these objects earlier

```
model <- compile_model(
  populations = list(ooa, ehg, eur, ana, yam), gene_flow = gf,
  generation_time = 30, resolution = 10e3,
  competition = 130e3, mating = 100e3, dispersal = 70e3,
)
```



7. Simulation of the model

```
slim(
  model, # model object compiled in the previous step
  seq_length = 100e6, recomb_rate = 1e-8, # simulate 100 Mb segment
)
```

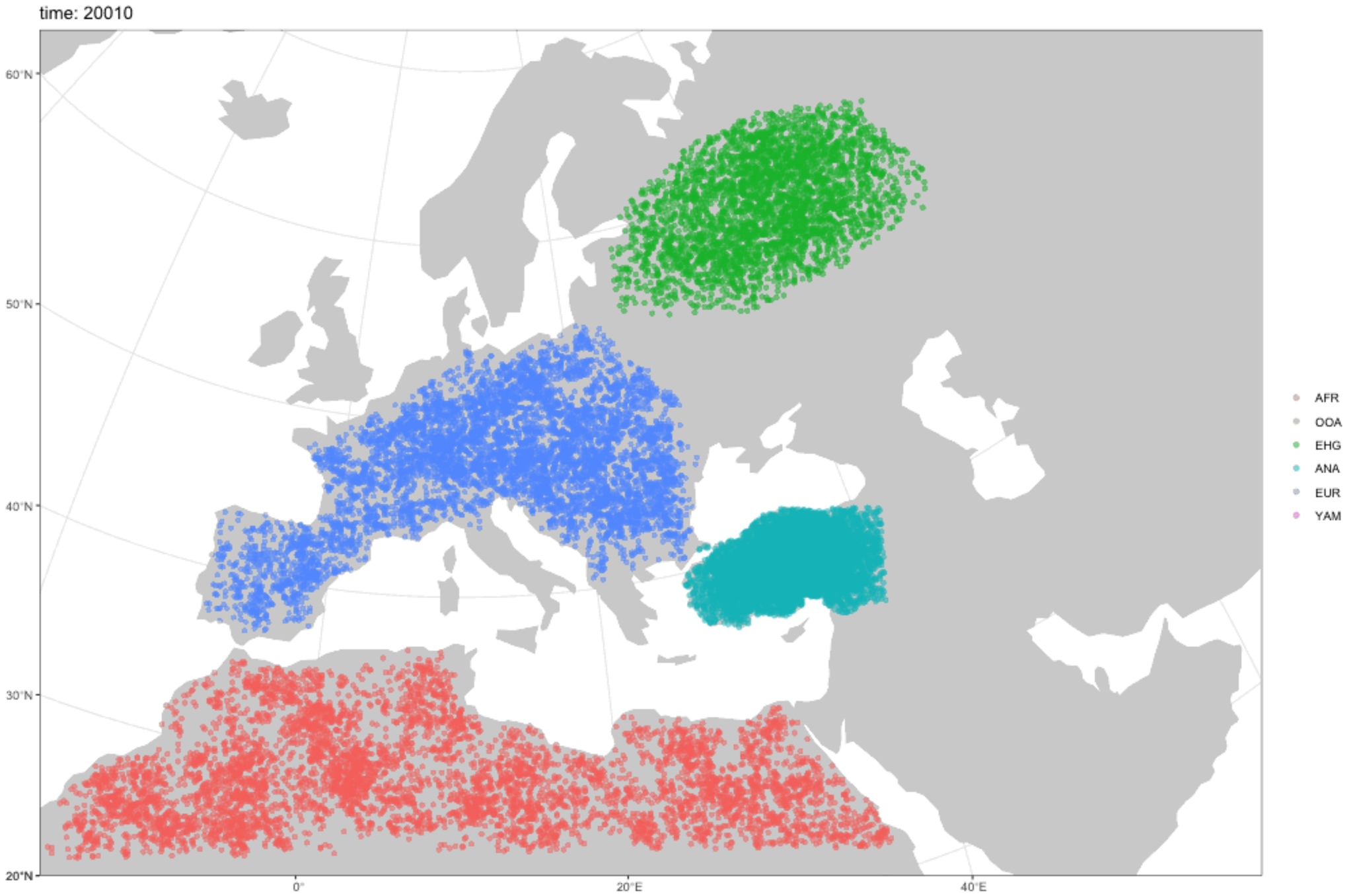
`slim()` executes a built-in SLiM script specifically designed for *slendr* demographic models

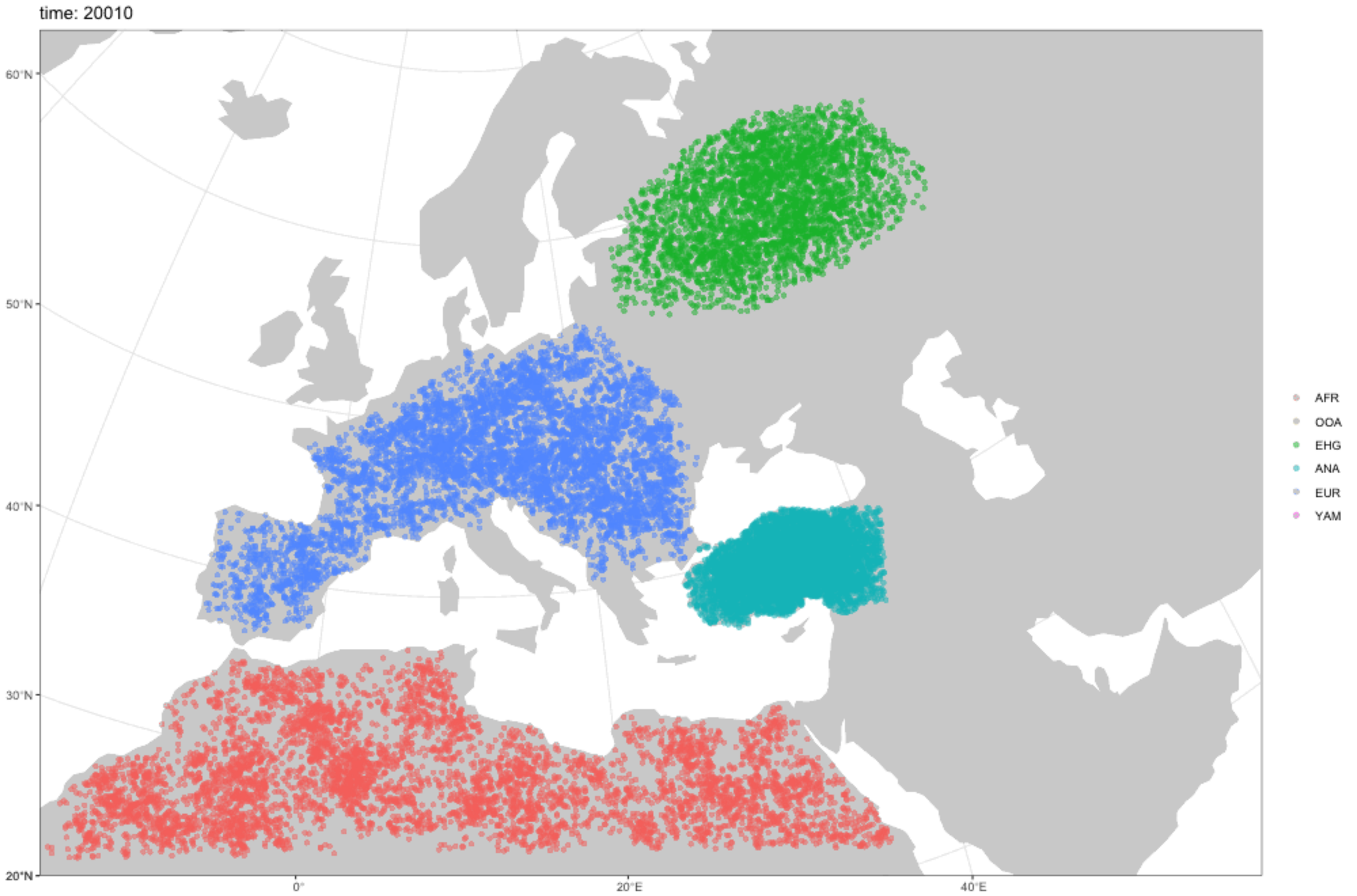
7. Simulation of the model

```
slim(
  model, # model object compiled in the previous step
  seq_length = 100e6, recomb_rate = 1e-8, # simulate 100 Mb segment
)
```

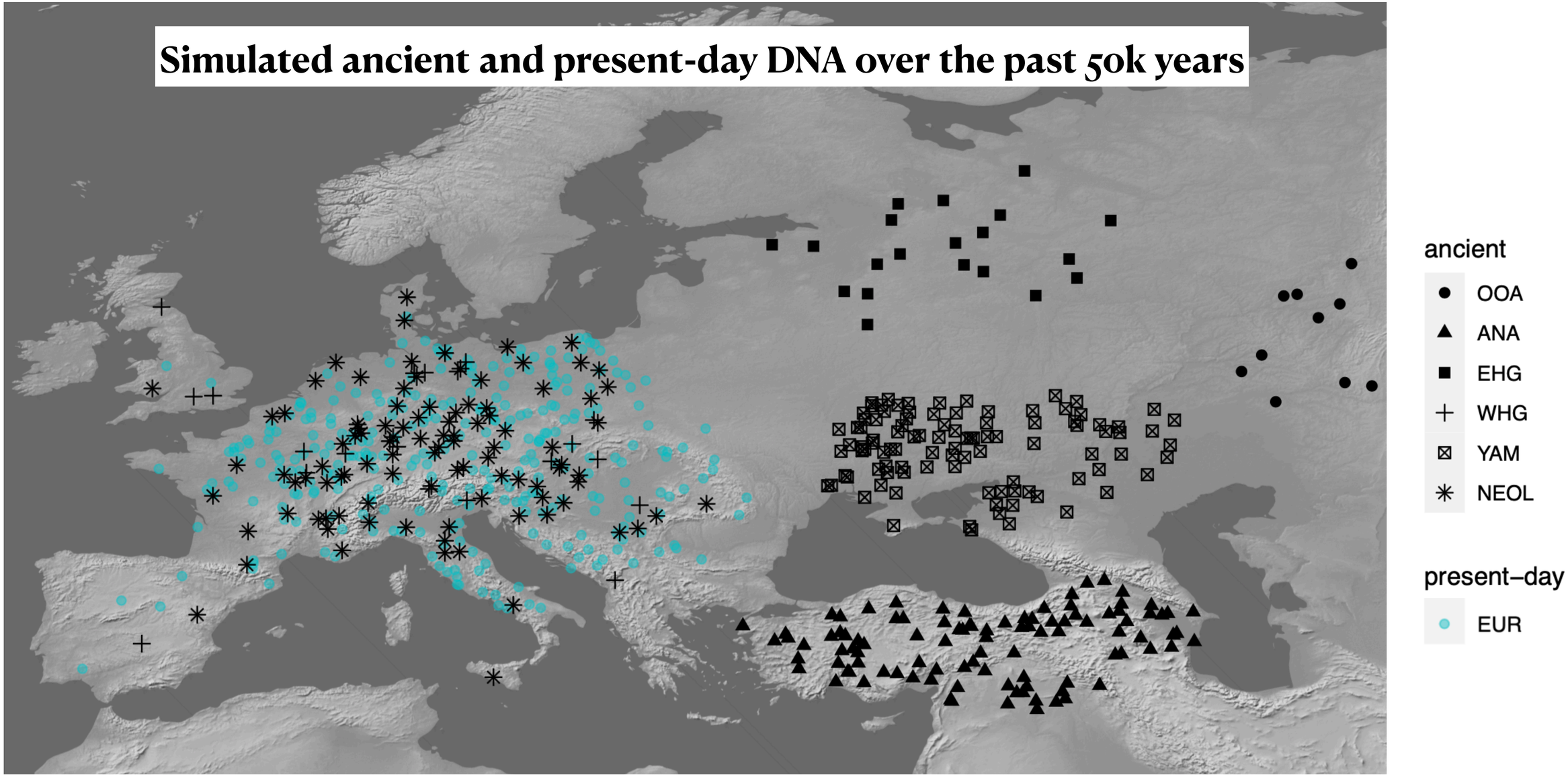
`slim()` executes a built-in SLiM script specifically designed for *slendr* demographic models

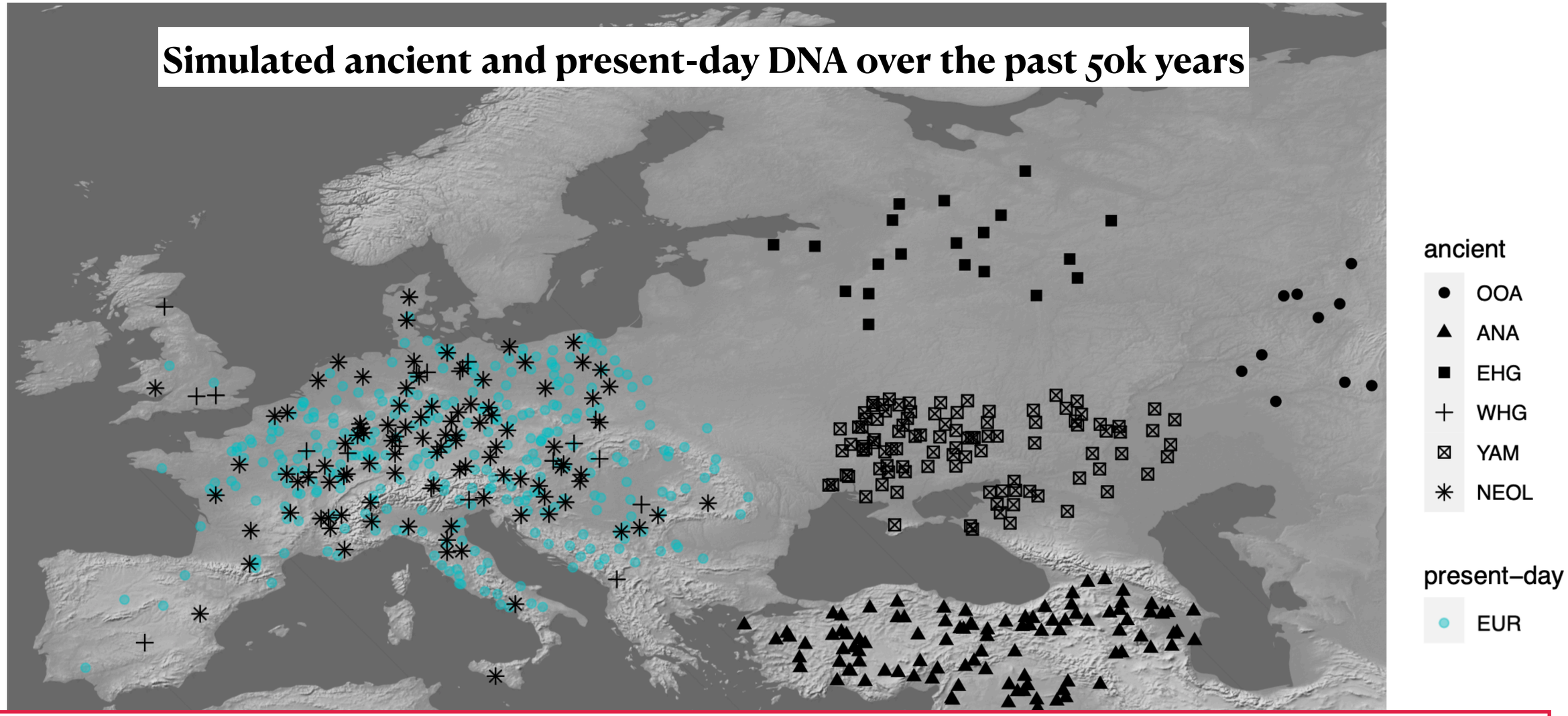
The output is a **tree sequence** (more on that soon...)


time: 20010
60°N
50°N
40°N
30°N
20°N
0°
20°E
40°E
AFR
OOA
EHG
ANA
EUR
YAM


time: 20010
60°N
50°N
40°N
30°N
20°N
0°
20°E
40°E
AFR
OOA
EHG
ANA
EUR
YAM

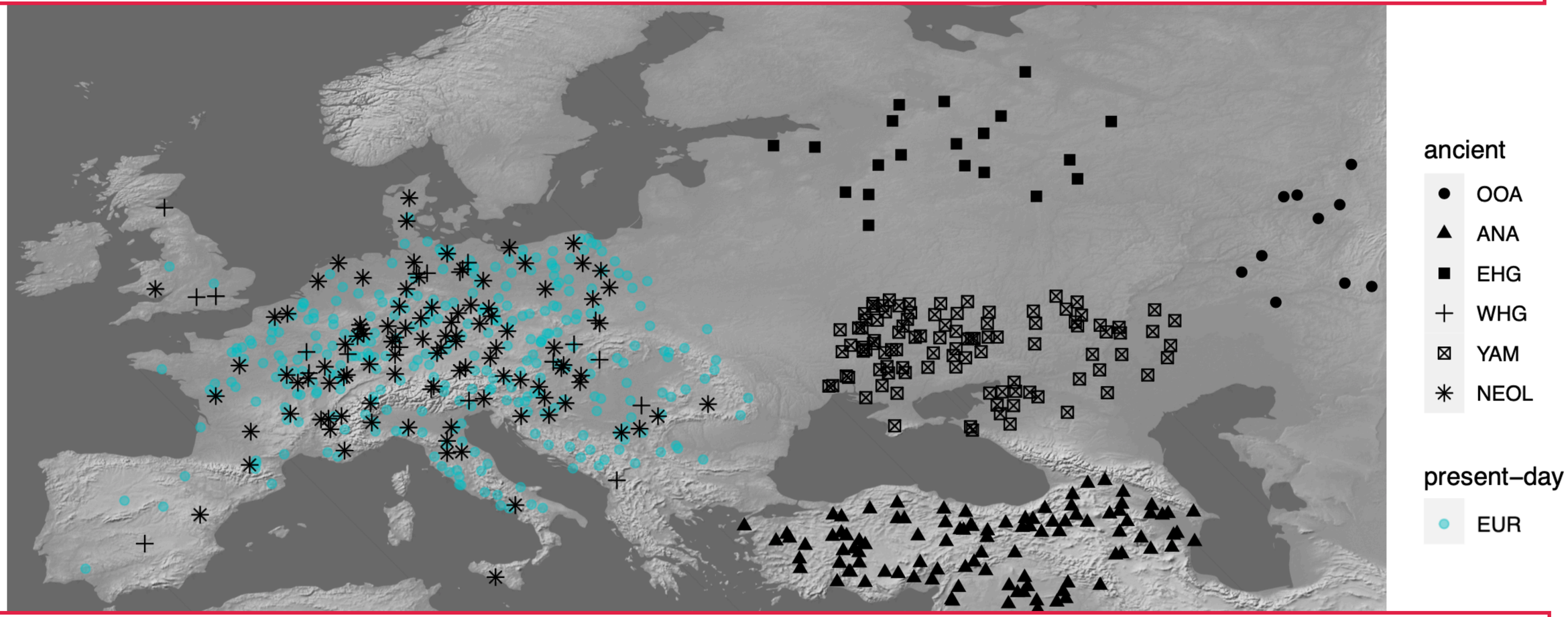
Simulated ancient and present-day DNA over the past 50k years





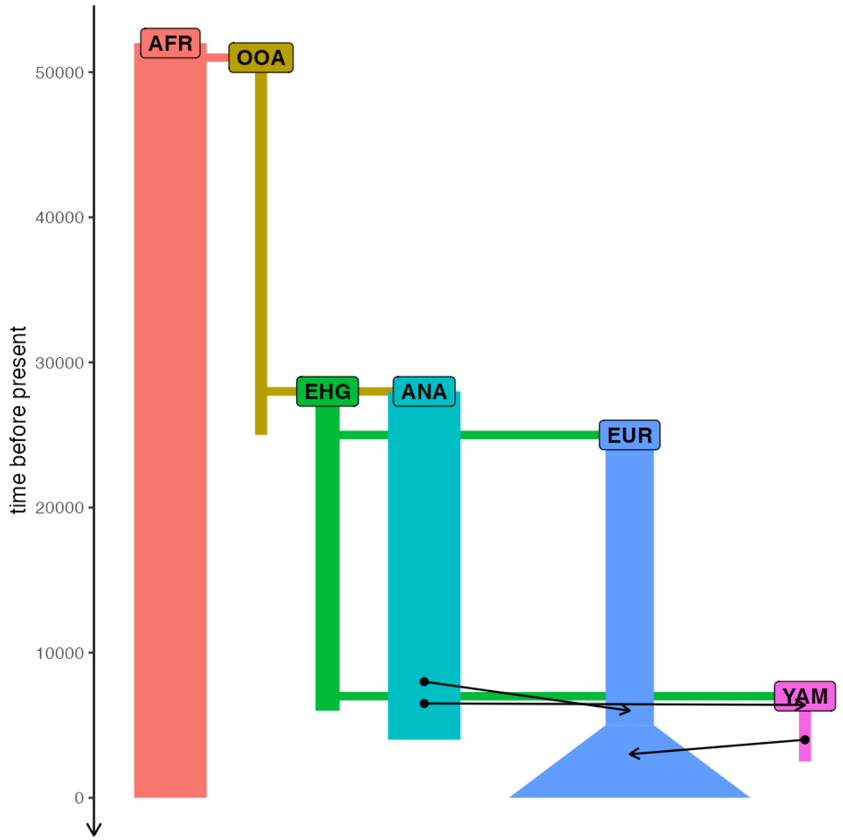
We know the genome and exact location & time of everyone

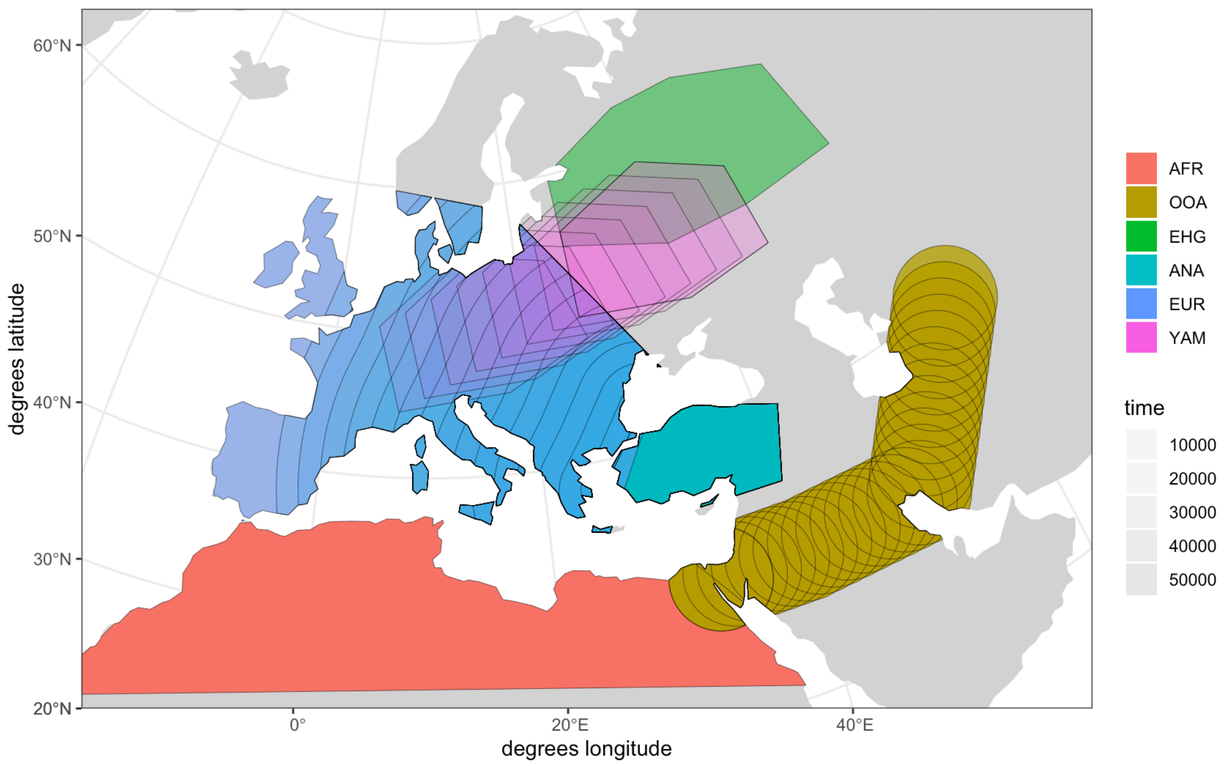
Generated by a **formally defined spatio-temporal model**



We know the genome and exact location & time of everyone

Generated by a **formally defined spatio-temporal model**





We know the genome and exact location & time of everyone

The output is a

tree sequence

https://tskit.dev

Model definition & simulation

Model components

`population()`	Define a population and its original spatial range
`world()`	Define a world map for all spatial operations
`geneflow()`	Define a geneflow event
`region()`	Define a geographic region

Population dynamics

`move()`	Move the population to a new location in a given amount of time
`expand()`	Expand the population range
`shrink()`	Shrink the population range
`boundary()`	Update the population range
`resize()`	Resize the population size
`dispersal()`	Change dispersal parameters

Manipulation of spatial objects

`join()`	Merge two spatial `slendr` objects into one
`overlap()`	Generate the overlap of two `slendr` objects
`subtract()`	Generate the difference between two `slendr` objects
`reproject()`	Reproject coordinates between coordinate systems
`distance()`	Calculate the distance between a pair of spatial boundaries
`area()`	Calculate the area covered by the given slendr object
`dimensions()`	Return the dimensions of the world map

Model visualization and diagnostics

`plot(<slendr>)`	Plot `slendr` geographic features on a map
`plot_graph()`	Plot geneflow graph based on given model configuration
`plot_ancestry()`	Plot simulated ancestry proportions
`animate()`	Animate the simulated population dynamics
`explore()`	Open an interactive browser of the spatial model

Compiling and running spatial models

`compile()`	Compile the spatial demographic model
`sampling()`	Define sampling events at specified times for a given set of populations
`slim()`	Run a slendr model as a SLiM script
`read()`	Read a previously serialized model configuration
`script()`	Substitute variables in a template SLiM script

Tree sequence analysis

Tree sequence loading and processing

`ts_load()`	Load a tree sequence file produced by a given model
`ts_recapitate()`	Recapitate the tree sequence
`ts_simplify()`	Simplify the tree sequence down to a given set of individuals
`ts_mutate()`	Add mutations to the given tree sequence
`ts_coalesced()`	Check that all trees in the tree sequence are fully coalesced

Extracting genotypes from tree sequences

`ts_genotypes()`	Extract genotype table from the tree sequence
`ts_eigenstrat()`	Extract genotypes from the tree sequence in the EIGENSTRAT format
`ts_vcf()`	Save genotypes from the tree sequenceas a VCF file

Accessing tree sequence components

`ts_data()`	Extract combined annotated table of individuals and nodes
`ts_ancestors()`	Infer spatio-temporal ancestral history for given nodes/individuals
`ts_individuals()` `ts_edges()` `ts_nodes()`	Get the table of individuals/nodes/edges from the tree sequence
`ts_samples()`	Extract names and times of individuals scheduled for sampling
`ts_tree()`	Get a tree from a given tree sequence
`ts_draw()`	Plot a graphical representation of a single tree

Tree sequence statistics

`ts_f2()` `ts_f3()` `ts_f4()` `ts_f4ratio()`	Calculate the f2, f3, f4, and f4-ratio statistics
`ts_afs()`	Compute the allele frequency spectrum (AFS)
`ts_fst()` `ts_divergence()`	Calculate pairwise statistics between sets of individuals
`ts_diversity()`	Calculate diversity in given sets of individuals
`ts_tajima()`	Calculate Tajima's D for given sets of individuals
`ts_segregating()`	Calculate the density of segregating sites for the given sets of individuals

What we usually have in population genetics

```
##fileformat=VCFv4.0
##source=BCM:SNPTools:hapfuse
##reference=1000Genomes-NCBI37
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=AP,Number=2,Type=Float,Description="Allelic Probability, P(Allele=1|Haplotype)">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  HG00096 HG00097 HG00099 HG00100 HG00101 HG00102
12      60076   .       A       C       100     PASS    .       GT:AP   1|0:1.000,0.000 0|0:0.000,0.005 0|0:0.000,0.005
12      60252   .       A       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.300 0|1:0.010,0.590
12      60317   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60344   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.020,0.025 0|0:0.005,0.000
12      60383   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60405   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60474   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.015,0.705 0|1:0.020,0.775
12      60614   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.005 0|0:0.000,0.015
12      60628   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.070 0|0:0.000,0.000
12      60654   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61021   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61107   .       G       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.015 0|0:0.000,0.000
12      61172   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61220   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.005,0.265 0|1:0.020,0.665
12      61258   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.005,0.465 0|1:0.020,0.895
12      61272   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61329   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61341   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.010,0.560 0|1:0.020,0.855
12      61368   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.020,0.630 0|1:0.020,0.955
12      61392   .       T       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61405   .       G       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61411   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.015
12      61416   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.025 0|0:0.000,0.010 0|0:0.015,0.075
12      61422   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.105 0|0:0.005,0.010
12      61476   .       C       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.005,0.000
12      61510   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61516   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.005 0|0:0.000,0.000
12      61552   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61604   .       T       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61687   .       G       A       100     PASS    .       GT:AP   1|0:1.000,0.000 0|1:0.015,0.625 0|1:0.020,0.960
12      61700   .       C       T       100     PASS    .       GT:AP   0|0:0.005,0.000 0|0:0.000,0.035 0|0:0.025,0.060
```

What we usually want to know?

What we usually want to know?

- which individuals are more closely related than others

What we usually want to know?

- which individuals are more closely related than others
- when did populations diverge from one another

What we usually want to know?

- which individuals are more closely related than others
- when did populations diverge from one another
- which populations admixed, at which time and rate

What we usually want to know?

- which individuals are more closely related than others
- when did populations diverge from one another
- which populations admixed, at which time and rate
- what was the N_e trajectory of a population across time

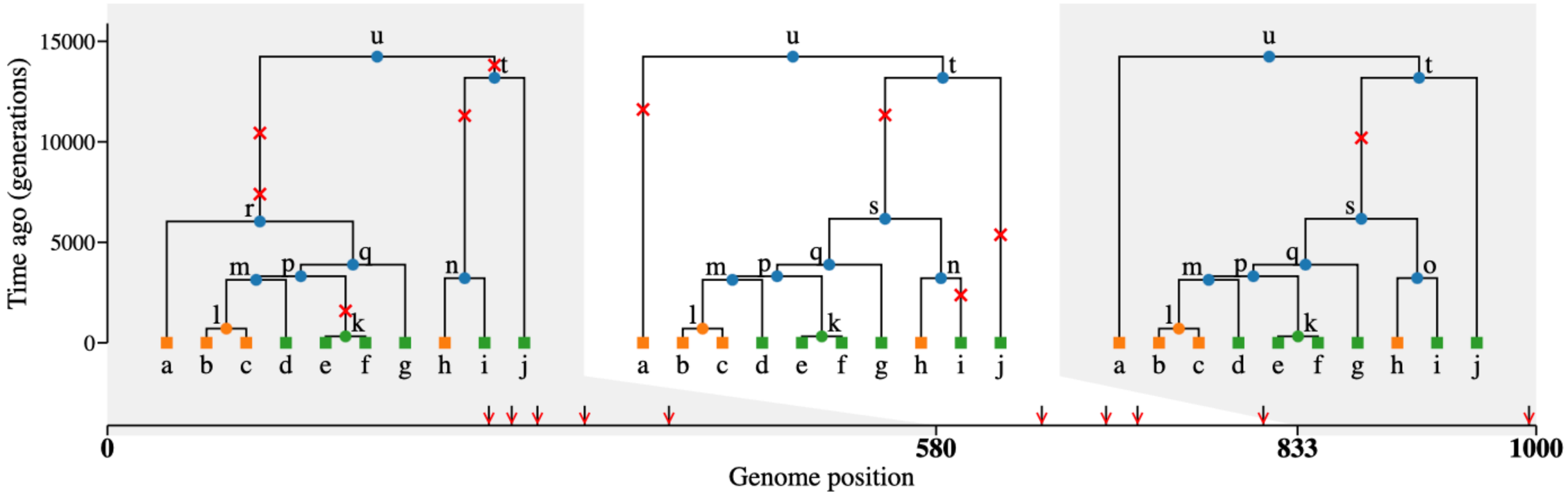
What we usually want to know?

```
##fileformat=VCFv4.0
##source=BCM:SNPTools:hapfuse
##reference=1000Genomes-NCBI37
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=AP,Number=2,Type=Float,Description="Allelic Probability, P(Allele=1|Haplotype)">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  HG00096 HG00097 HG00099 HG00100 HG00101 HG00102
12      60076   .       A       C       100     PASS    .       GT:AP   1|0:1.000,0.000 0|0:0.000,0.005 0|0:0.000,0.005
12      60252   .       A       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.300 0|1:0.010,0.590
12      60317   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60344   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.020,0.025 0|0:0.005,0.000
12      60383   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60405   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      60474   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.015,0.705 0|1:0.020,0.775
12      60614   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.005 0|0:0.000,0.015
12      60628   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.070 0|0:0.000,0.000
12      60654   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61021   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61107   .       G       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.015 0|0:0.000,0.000
12      61172   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61220   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.005,0.265 0|1:0.020,0.665
12      61258   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.005,0.465 0|1:0.020,0.895
12      61272   .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61329   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61341   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.010,0.560 0|1:0.020,0.855
12      61368   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|1:0.020,0.630 0|1:0.020,0.955
12      61392   .       T       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61405   .       G       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61411   .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.015
12      61416   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.025 0|0:0.000,0.010 0|0:0.015,0.075
12      61422   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.105 0|0:0.005,0.010
12      61476   .       C       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.005,0.000
12      61510   .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61516   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.005 0|0:0.000,0.000
12      61552   .       C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61604   .       T       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.000,0.000 0|0:0.000,0.000
12      61687   .       G       A       100     PASS    .       GT:AP   1|0:1.000,0.000 0|1:0.015,0.625 0|1:0.020,0.960
12      61700   .       C       T       100     PASS    .       GT:AP   0|0:0.005,0.000 0|0:0.000,0.035 0|0:0.025,0.060
```

What we usually want to know?

- which individuals are more closely related than others
- when did populations diverge from one another
- which populations admixed, at which time and rate
- what was the N_e trajectory of a population across time

This is what a tree sequence gives us!

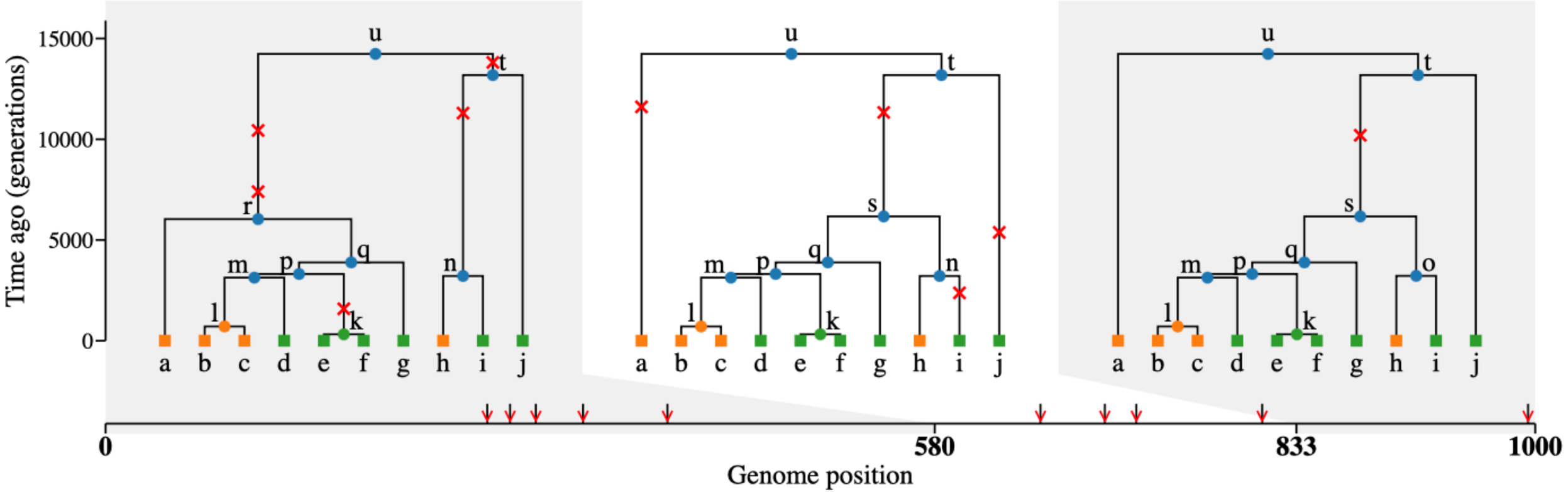


We get this for free with simulated data (SLiM, *msprime*).

https://tskit.dev

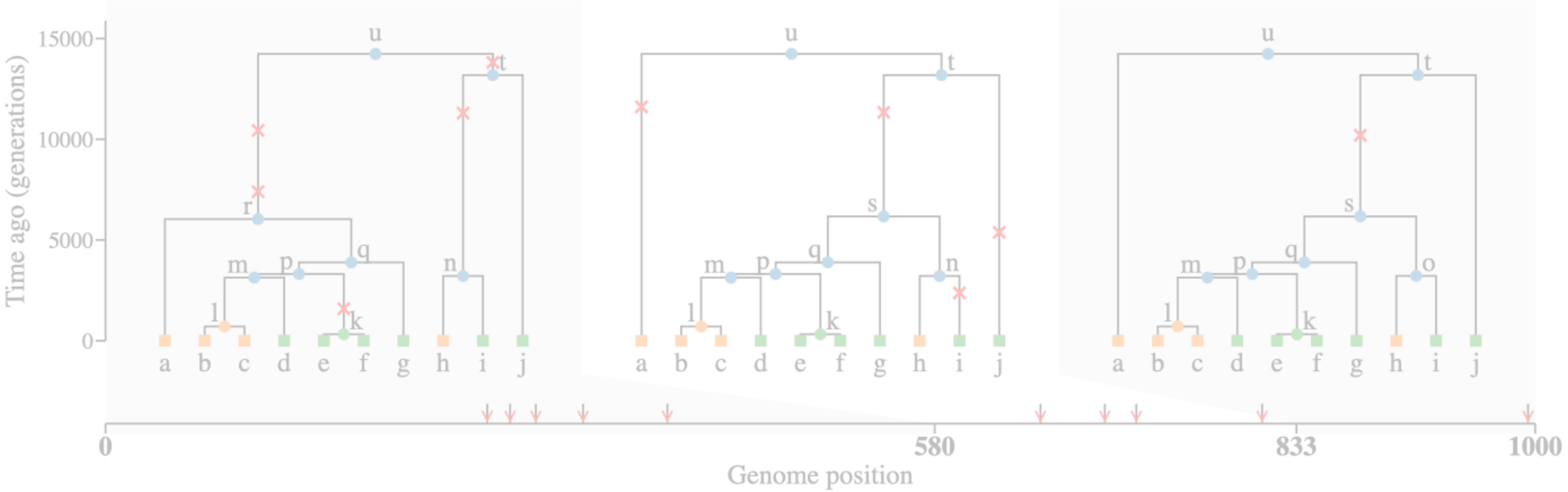
Example: allele frequency spectrum

Example: allele frequency spectrum



https://tskit.dev

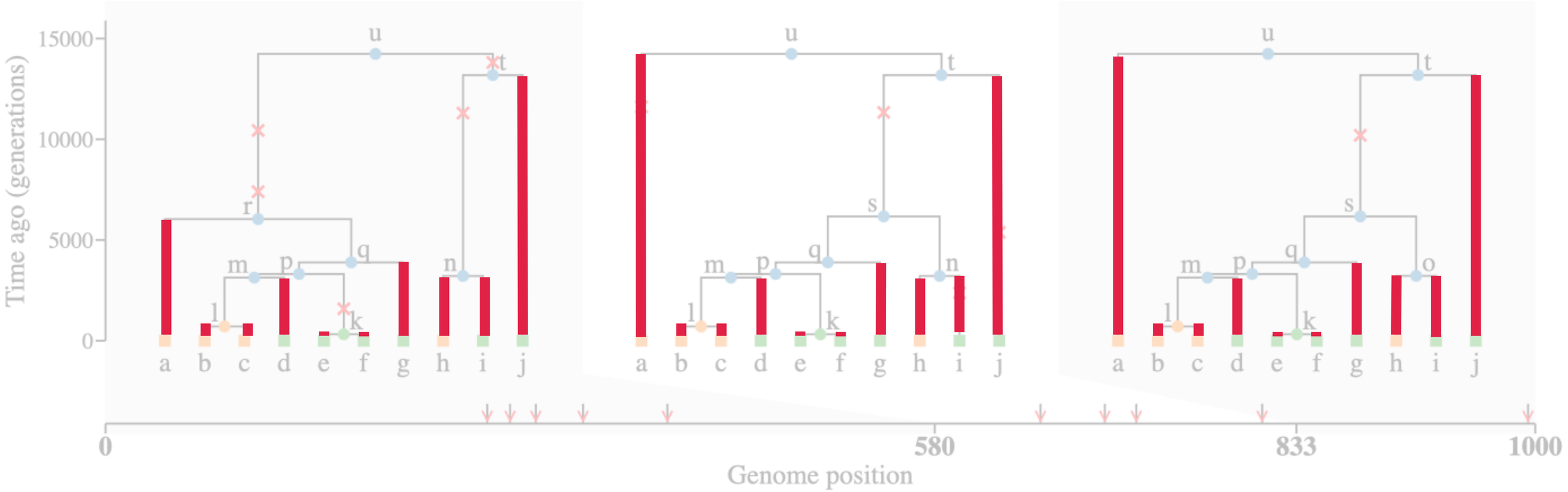
Example: allele frequency spectrum (singletons)




```
ts_afs()
```


https://tskit.dev

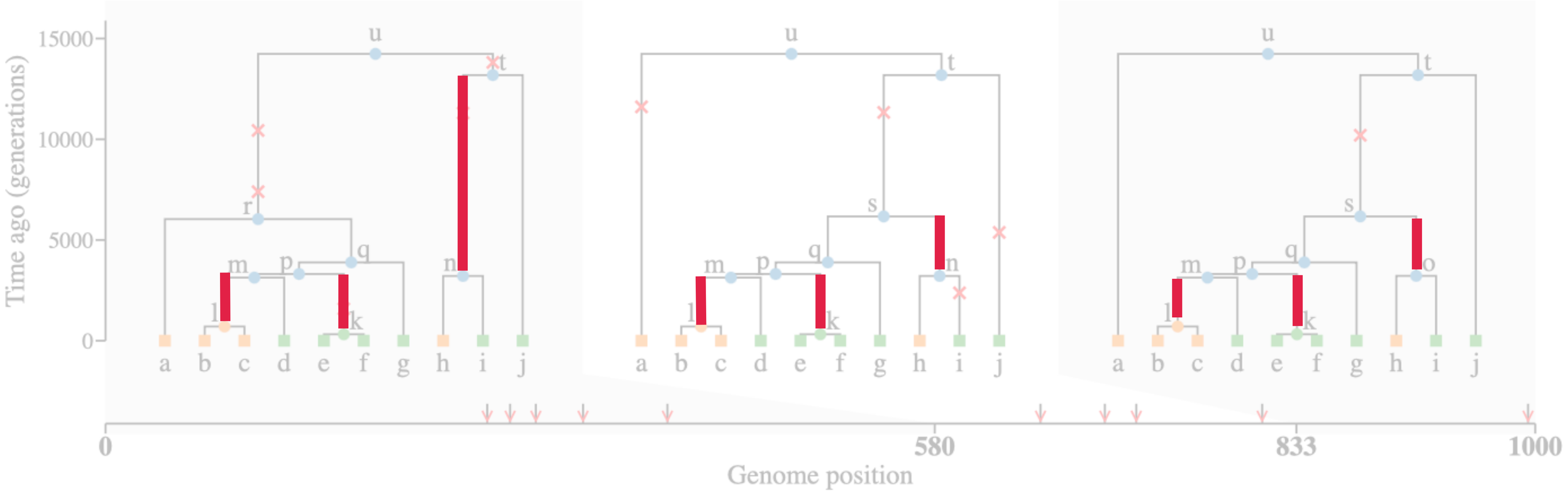
Example: allele frequency spectrum (singletons)




```
ts_afs()
```


https://tskit.dev

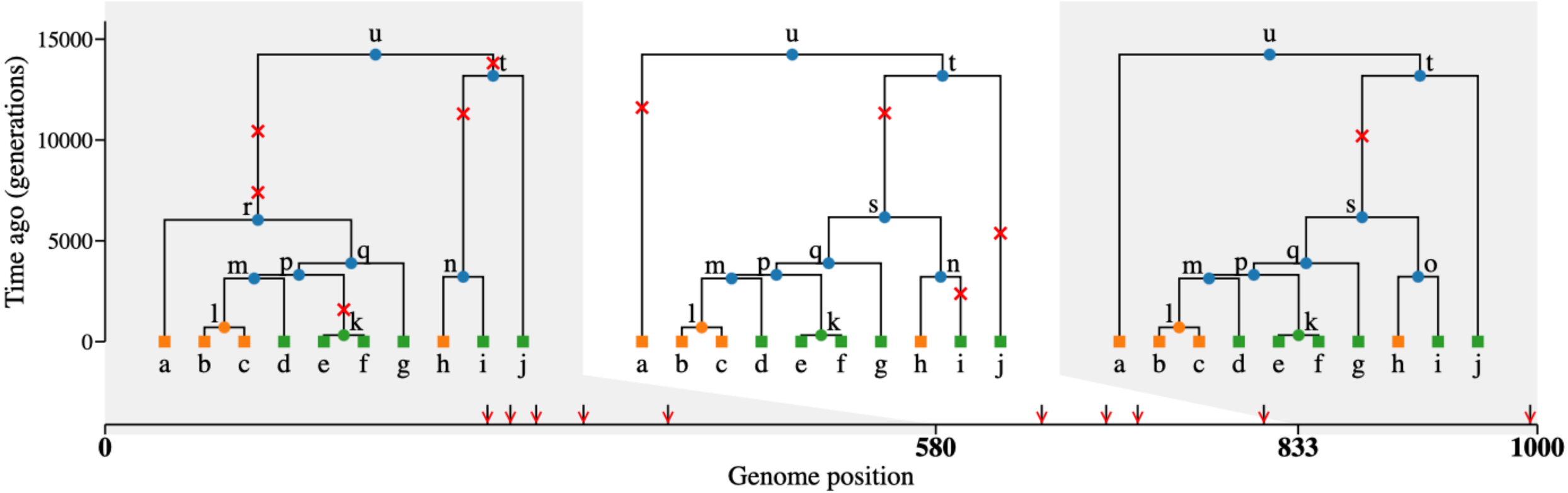
Example: allele frequency spectrum (doubletons)




```
ts_afs()
```


https://tskit.dev

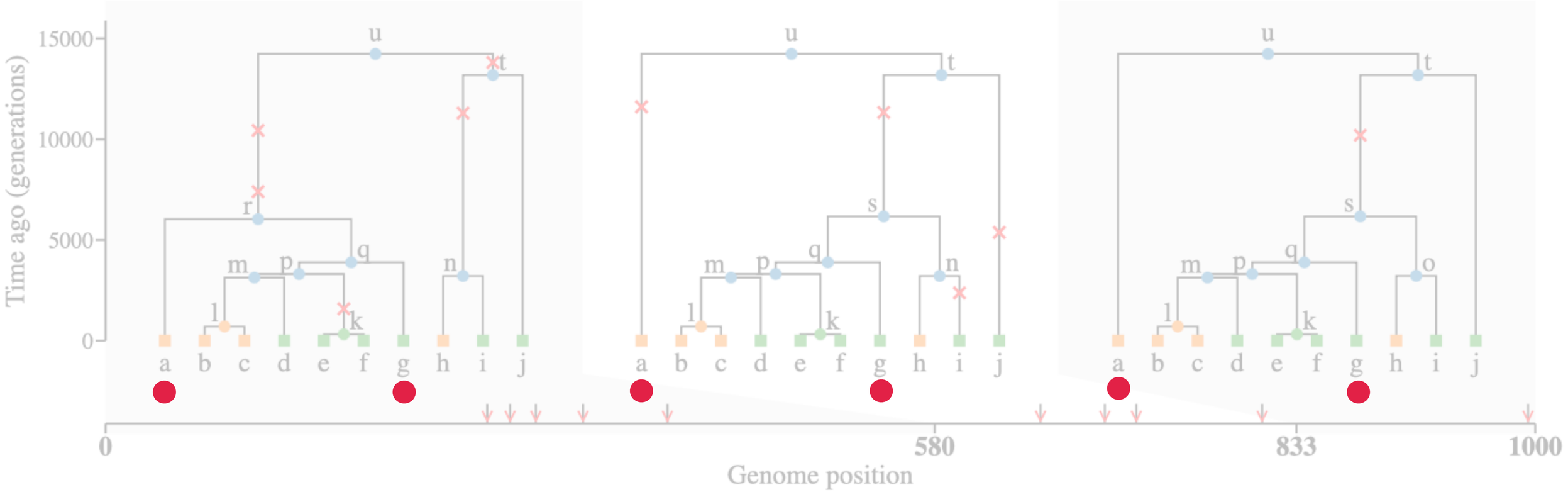
Example: divergence between samples




```
ts_divergence()
```

https://tskit.dev

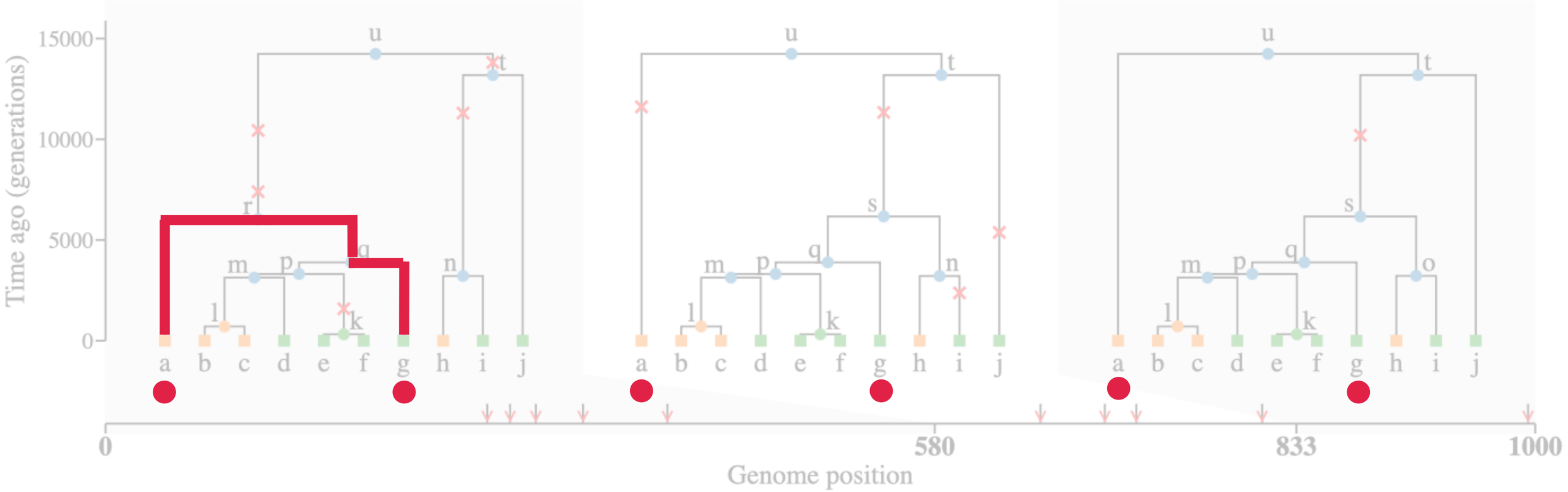
Example: divergence between samples




```
ts_divergence()
```

https://tskit.dev

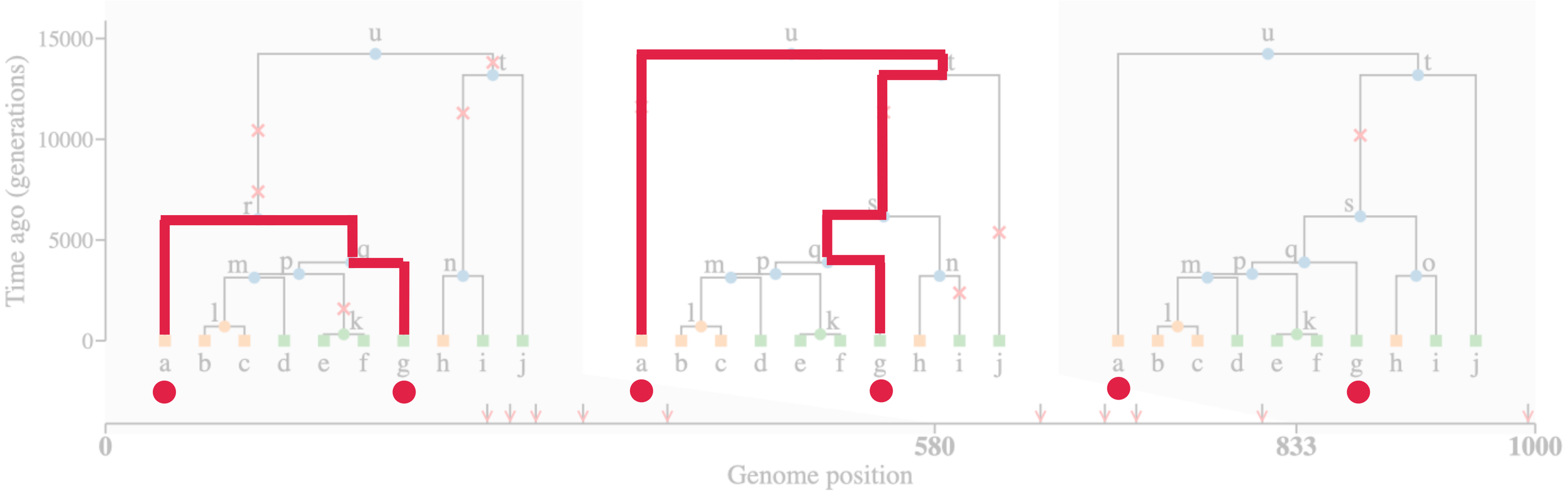
Example: divergence between samples




```
ts_divergence()
```

https://tskit.dev

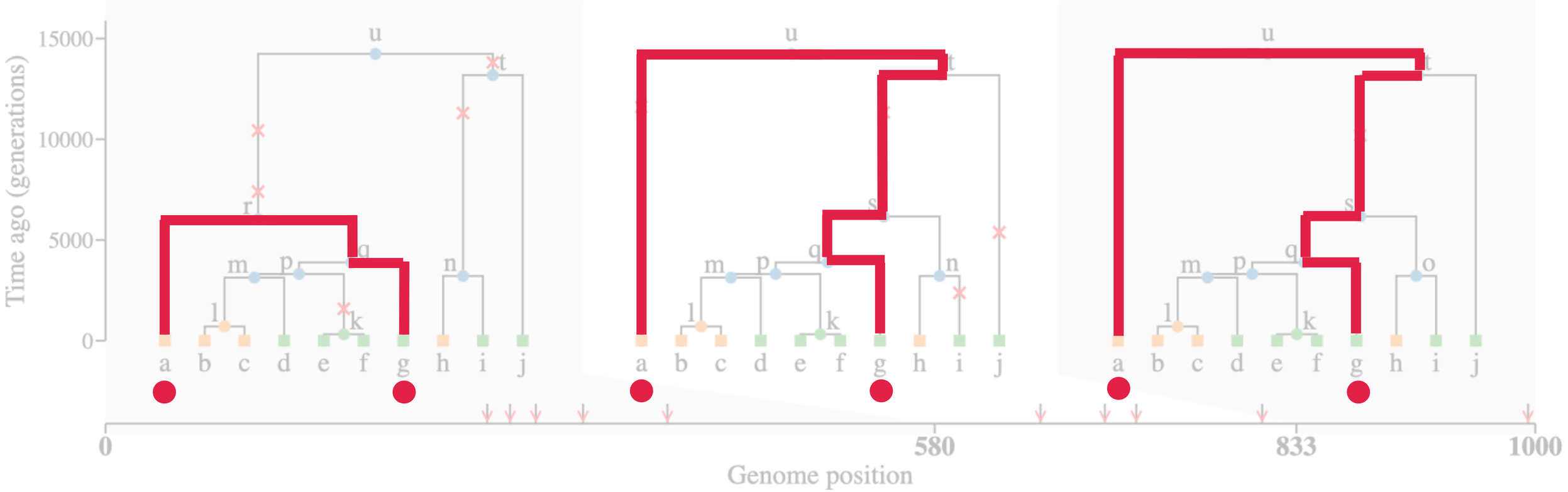
Example: divergence between samples




```
ts_divergence()
```

https://tskit.dev

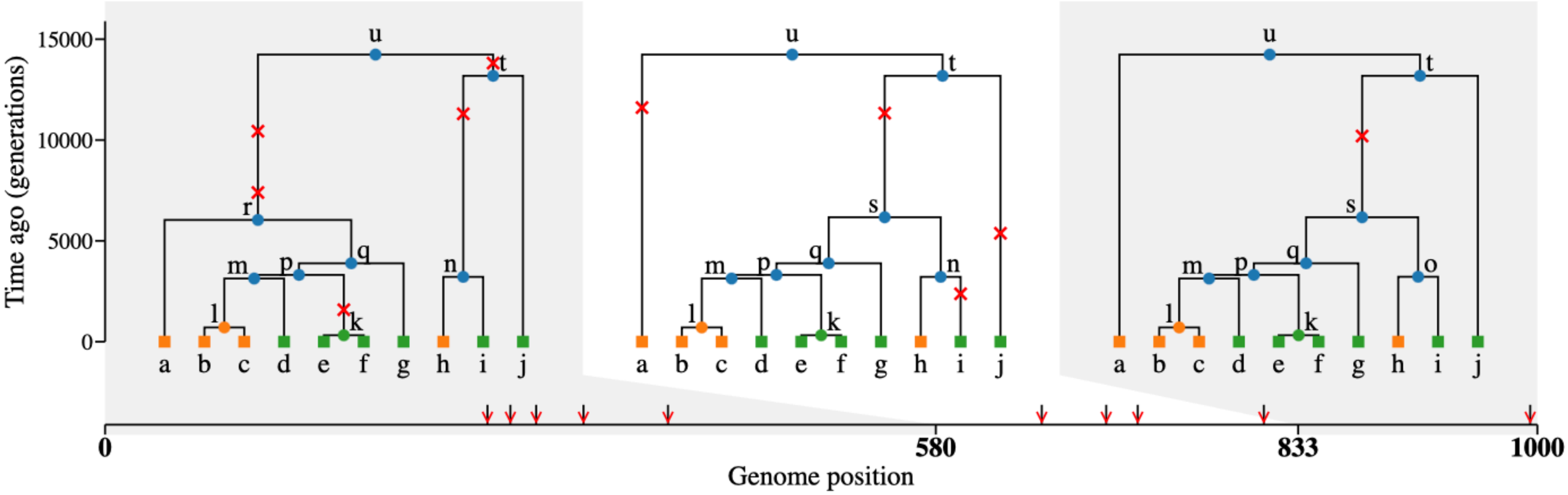
Example: divergence between samples




```
ts_divergence()
```

https://tskit.dev

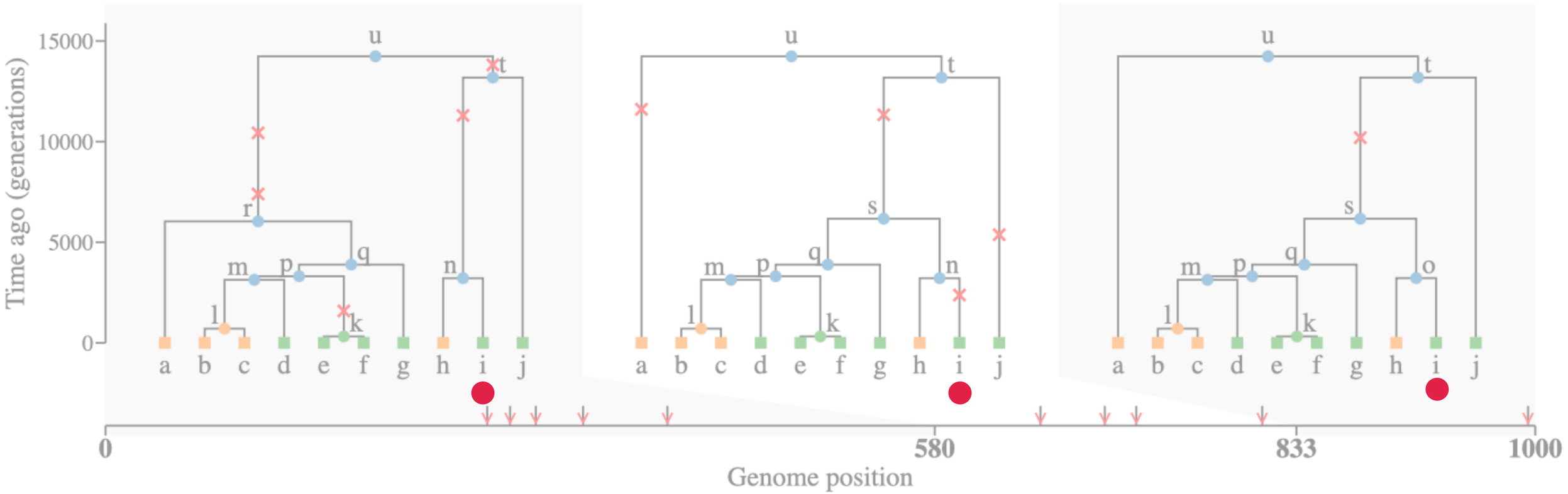
Example: conversion to standard genotype table




```
ts_vcf()
```

https://tskit.dev

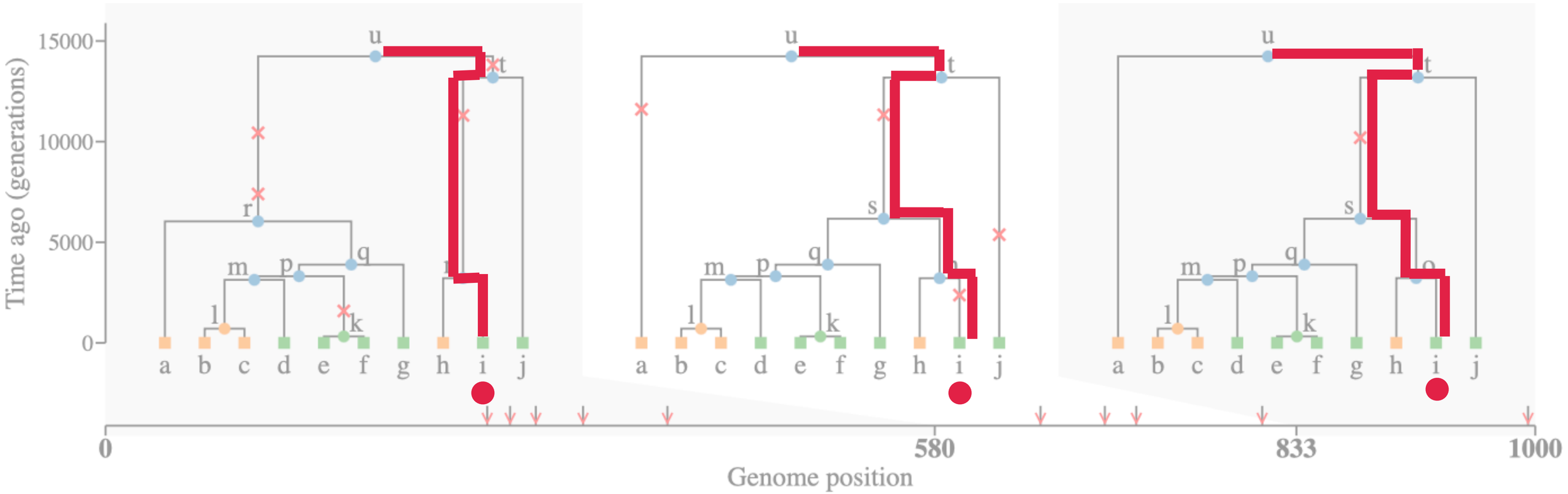
Example: conversion to standard genotype table




```
ts_vcf()
```

https://tskit.dev

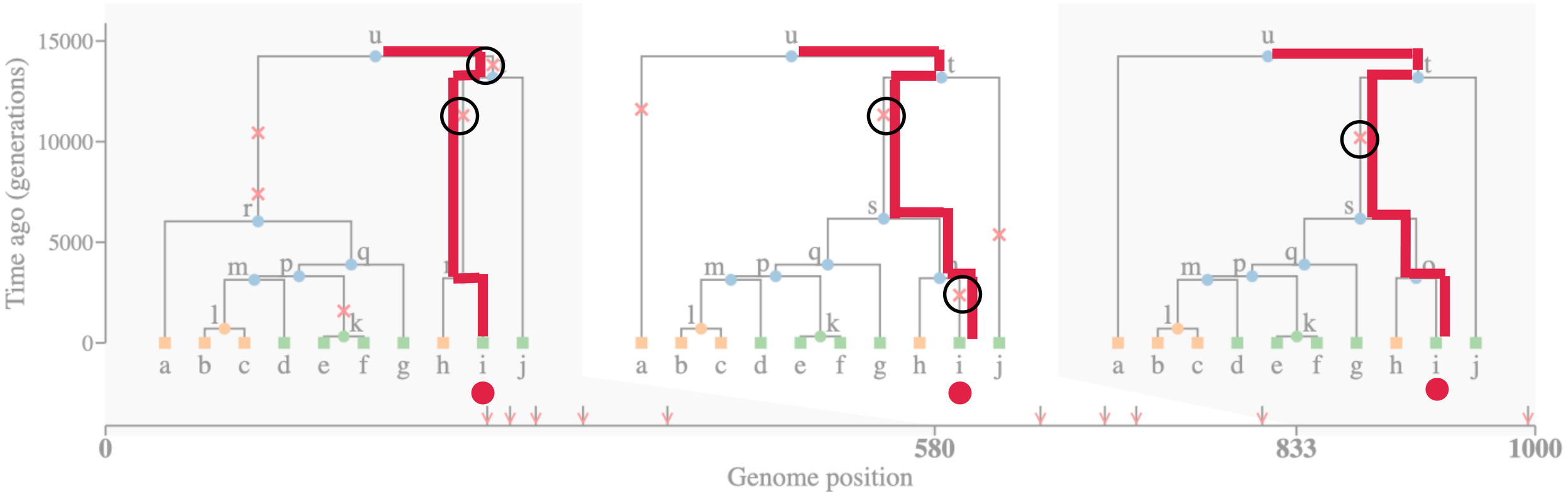
Example: conversion to standard genotype table




```
ts_vcf()
```

https://tskit.dev

Example: conversion to standard genotype table




```
ts_vcf()
```

https://tskit.dev

Model definition & simulation

Model components

`population()`	Define a population and its original spatial range
`world()`	Define a world map for all spatial operations
`geneflow()`	Define a geneflow event
`region()`	Define a geographic region

Population dynamics

`move()`	Move the population to a new location in a given amount of time
`expand()`	Expand the population range
`shrink()`	Shrink the population range
`boundary()`	Update the population range
`resize()`	Resize the population size
`dispersal()`	Change dispersal parameters

Manipulation of spatial objects

`join()`	Merge two spatial `slendr` objects into one
`overlap()`	Generate the overlap of two `slendr` objects
`subtract()`	Generate the difference between two `slendr` objects
`reproject()`	Reproject coordinates between coordinate systems
`distance()`	Calculate the distance between a pair of spatial boundaries
`area()`	Calculate the area covered by the given slendr object
`dimensions()`	Return the dimensions of the world map

Model visualization and diagnostics

`plot(<slendr>)`	Plot `slendr` geographic features on a map
`plot_graph()`	Plot geneflow graph based on given model configuration
`plot_ancestry()`	Plot simulated ancestry proportions
`animate()`	Animate the simulated population dynamics
`explore()`	Open an interactive browser of the spatial model

Compiling and running spatial models

`compile()`	Compile the spatial demographic model
`sampling()`	Define sampling events at specified times for a given set of populations
`slim()`	Run a slendr model as a SLiM script
`read()`	Read a previously serialized model configuration
`script()`	Substitute variables in a template SLiM script

Tree sequence analysis

Tree sequence loading and processing

`ts_load()`	Load a tree sequence file produced by a given model
`ts_recapitate()`	Recapitate the tree sequence
`ts_simplify()`	Simplify the tree sequence down to a given set of individuals
`ts_mutate()`	Add mutations to the given tree sequence
`ts_coalesced()`	Check that all trees in the tree sequence are fully coalesced

Extracting genotypes from tree sequences

`ts_genotypes()`	Extract genotype table from the tree sequence
`ts_eigenstrat()`	Extract genotypes from the tree sequence in the EIGENSTRAT format
`ts_vcf()`	Save genotypes from the tree sequenceas a VCF file

Accessing tree sequence components

`ts_data()`	Extract combined annotated table of individuals and nodes
`ts_ancestors()`	Infer spatio-temporal ancestral history for given nodes/individuals
`ts_individuals()` `ts_edges()` `ts_nodes()`	Get the table of individuals/nodes/edges from the tree sequence
`ts_samples()`	Extract names and times of individuals scheduled for sampling
`ts_tree()`	Get a tree from a given tree sequence
`ts_draw()`	Plot a graphical representation of a single tree

Tree sequence statistics

`ts_f2()` `ts_f3()` `ts_f4()` `ts_f4ratio()`	Calculate the f2, f3, f4, and f4-ratio statistics
`ts_afs()`	Compute the allele frequency spectrum (AFS)
`ts_fst()` `ts_divergence()`	Calculate pairwise statistics between sets of individuals
`ts_diversity()`	Calculate diversity in given sets of individuals
`ts_tajima()`	Calculate Tajima's D for given sets of individuals
`ts_segregating()`	Calculate the density of segregating sites for the given sets of individuals

No information loss.

Very fast computation.

No information loss.

Very fast computation.

https://tskit.dev

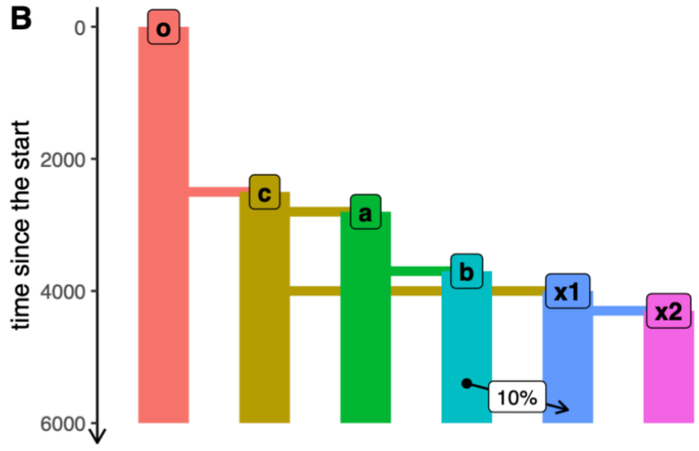
How does everything work in practice?

slendr as a "popgen calculator" (for non-spatial simulations)

slendr as a "popgen calculator" (for non-spatial simulations)

```
o <- population("o", time = 1, N = 100)
c <- population("c", time = 2500, N = 100, parent = o)
a <- population("a", time = 2800, N = 100, parent = c)
b <- population("b", time = 3700, N = 100, parent = a)
x1 <- population("x1", time = 4000, N = 15000, parent = c)
x2 <- population("x2", time = 4300, N = 15000, parent = x1)

gf <- gene_flow(from = b, to = x1, start = 5400, end = 5800, 0.1)

model <- compile_model(
  populations = list(o, a, b, c, x1, x2), gene_flow = gf,
  generation_time = 1, sim_length = 6000
)
```

```
plot_model(model, sizes = FALSE, proportions = TRUE) # panel B
```

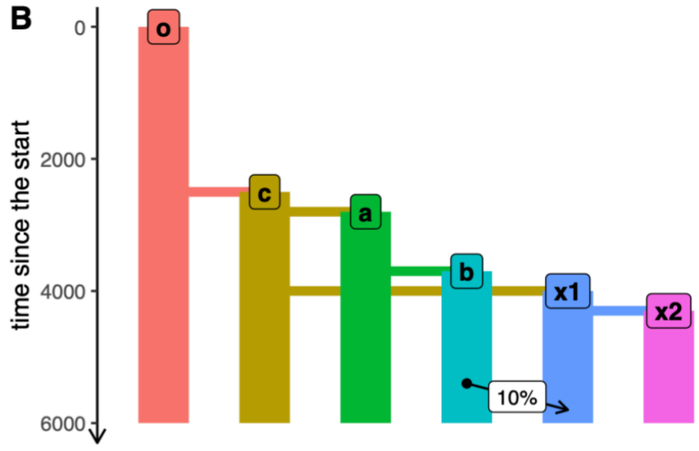


slendr as a "popgen calculator" (for non-spatial simulations)

```
o <- population("o", time = 1, N = 100)
c <- population("c", time = 2500, N = 100, parent = o)
a <- population("a", time = 2800, N = 100, parent = c)
b <- population("b", time = 3700, N = 100, parent = a)
x1 <- population("x1", time = 4000, N = 15000, parent = c)
x2 <- population("x2", time = 4300, N = 15000, parent = x1)

gf <- gene_flow(from = b, to = x1, start = 5400, end = 5800, 0.1)

model <- compile_model(
  populations = list(o, a, b, c, x1, x2), gene_flow = gf,
  generation_time = 1, sim_length = 6000
)
```

```
plot_model(model, sizes = FALSE, proportions = TRUE) # panel B
```

```
ts <-
  msprime(model, sequence_length = 100e6,
          recombination_rate = 1e-8) %>%
  ts_mutate(mutation_rate = 1e-8)
```



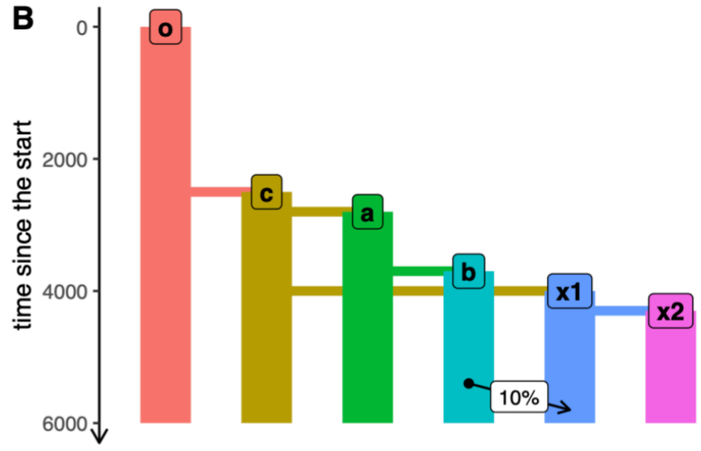
slendr as a "popgen calculator" (for non-spatial simulations)

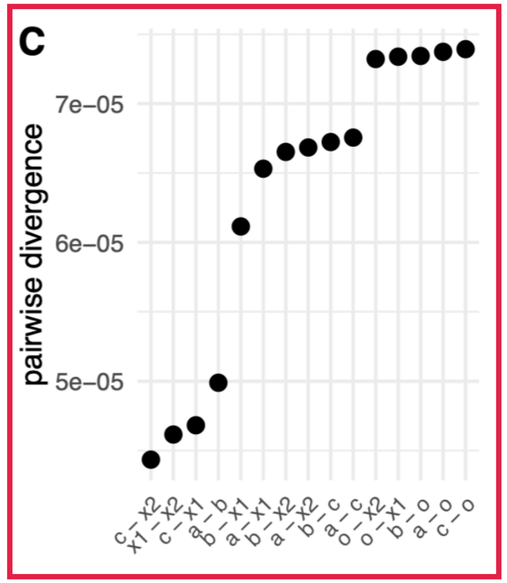
```
o <- population("o", time = 1, N = 100)
c <- population("c", time = 2500, N = 100, parent = o)
a <- population("a", time = 2800, N = 100, parent = c)
b <- population("b", time = 3700, N = 100, parent = a)
x1 <- population("x1", time = 4000, N = 15000, parent = c)
x2 <- population("x2", time = 4300, N = 15000, parent = x1)

gf <- gene_flow(from = b, to = x1, start = 5400, end = 5800, 0.1)

model <- compile_model(
  populations = list(o, a, b, c, x1, x2), gene_flow = gf,
  generation_time = 1, sim_length = 6000
)
```

```
plot_model(model, sizes = FALSE, proportions = TRUE) # panel B
```

```
ts <-
  msprime(model, sequence_length = 100e6,
          recombination_rate = 1e-8) %>%
  ts_mutate(mutation_rate = 1e-8)
```

```
# panel C
divergence <- ts_divergence(ts, split(samples$name, samples$pop))
```





slendr as a "popgen calculator" (for non-spatial simulations)

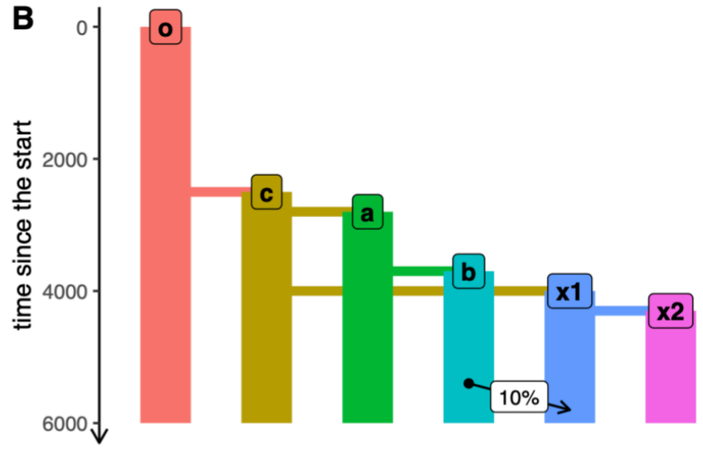
```
o <- population("o", time = 1, N = 100)
c <- population("c", time = 2500, N = 100, parent = o)
a <- population("a", time = 2800, N = 100, parent = c)
b <- population("b", time = 3700, N = 100, parent = a)
x1 <- population("x1", time = 4000, N = 15000, parent = c)
x2 <- population("x2", time = 4300, N = 15000, parent = x1)

gf <- gene_flow(from = b, to = x1, start = 5400, end = 5800, 0.1)

model <- compile_model(
  populations = list(o, a, b, c, x1, x2), gene_flow = gf,
  generation_time = 1, sim_length = 6000
)
```

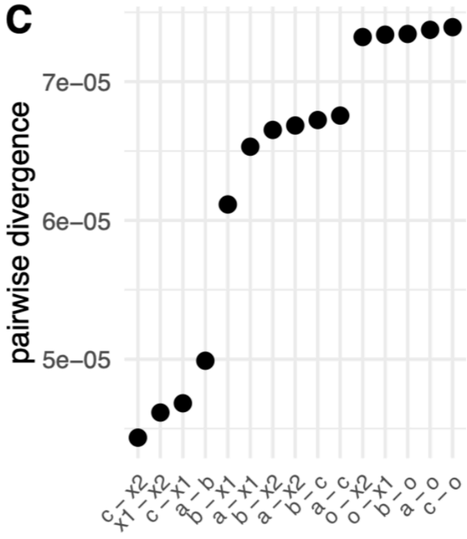
```
plot_model(model, sizes = FALSE, proportions = TRUE) # panel B
```

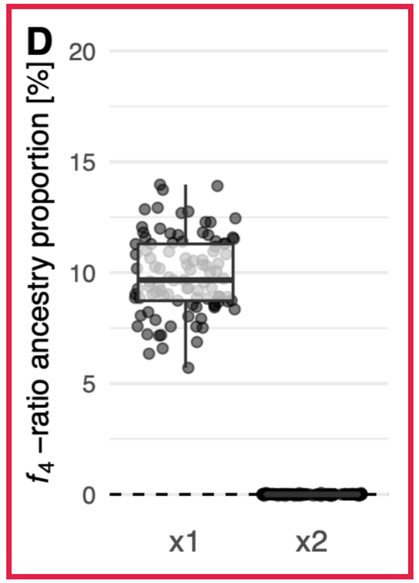
```
ts <-
  msprime(model, sequence_length = 100e6,
          recombination_rate = 1e-8) %>%
  ts_mutate(mutation_rate = 1e-8)
```

```
# panel C
divergence <- ts_divergence(ts, split(samples$name, samples$pop))
```

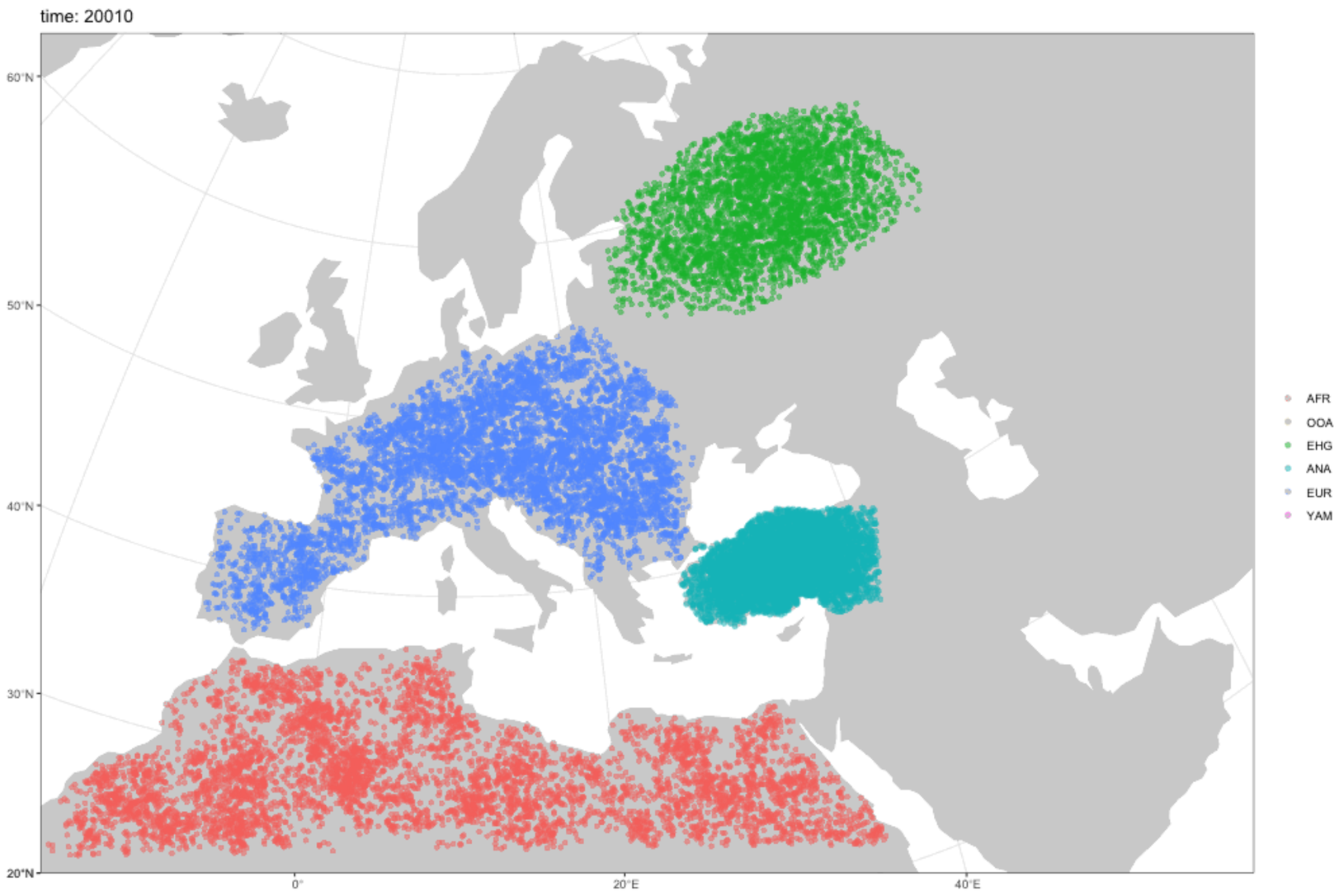
```
# panel D
f4ratio <- ts_f4ratio(
  ts, X = filter(samples, pop %in% c("x1", "x2"))$name,
  A = "a_1", B = "b_1", C = "c_1", O = "o_1"
)
```



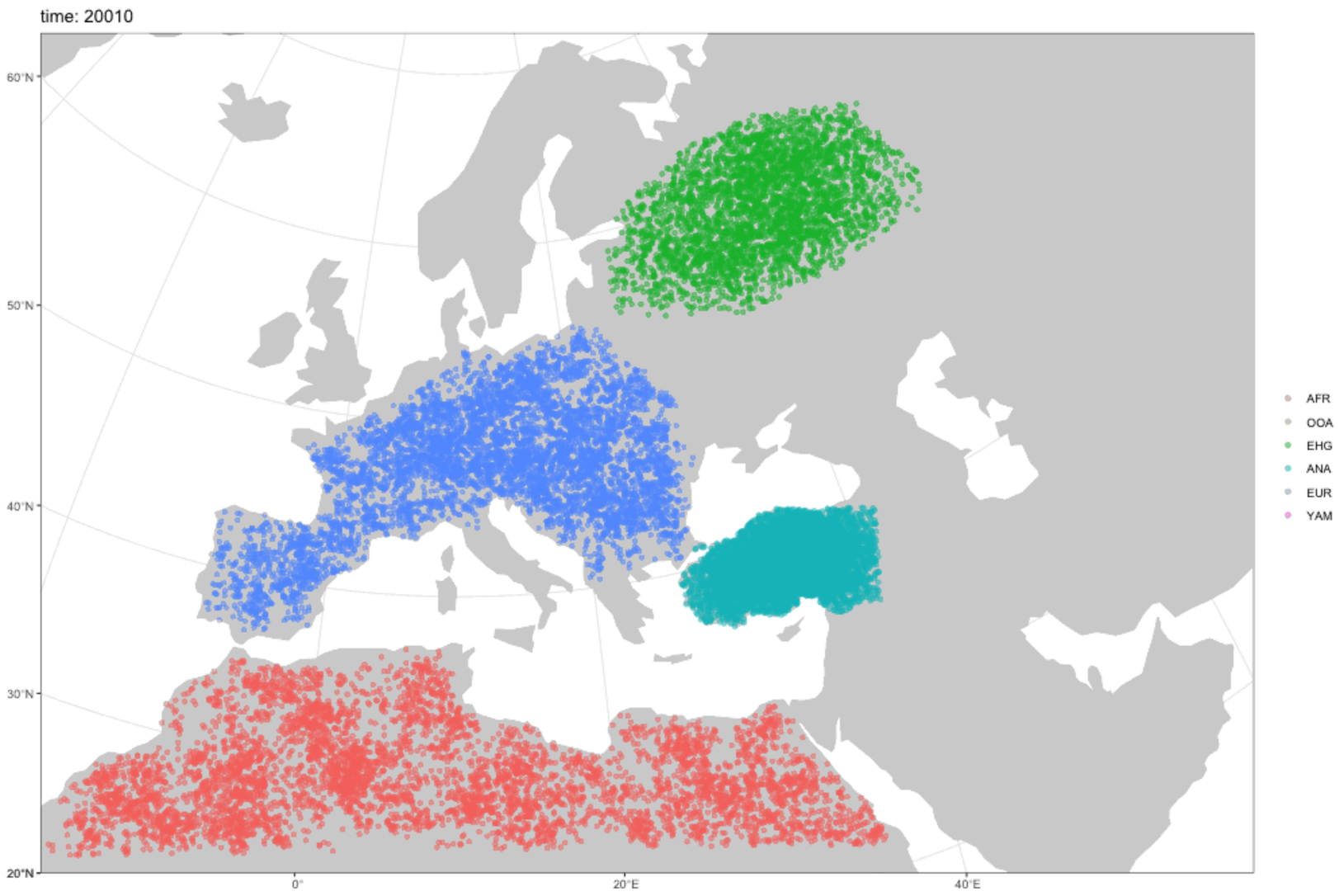




Fine-grained within-population dynamics

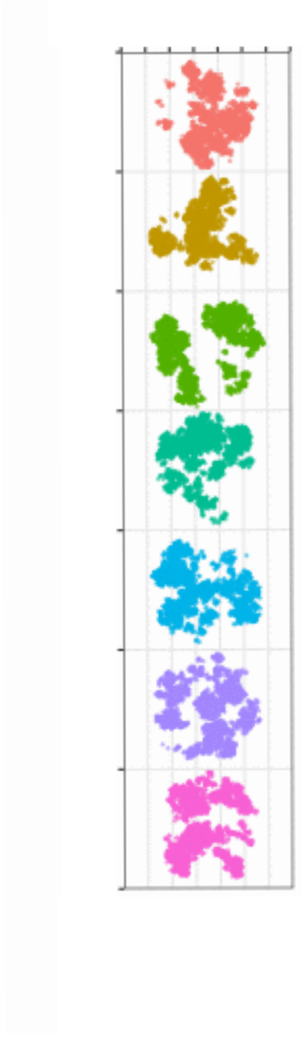


Fine-grained within-population dynamics



Fine-grained within-population dynamics

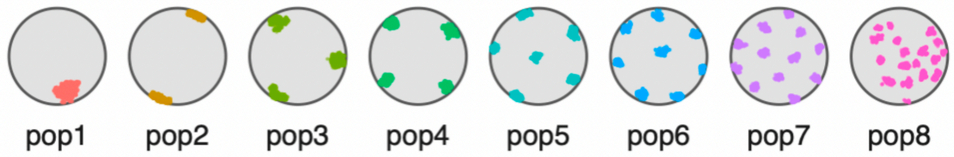
- **"competition distance"**
  - to what degree individuals stay close to others
- **"offspring dispersal distance"**
  - how far offspring ends up from a parent
- **"mating distance"**
  - how far will individuals look for mates



Fine-grained within-population dynamics

Fine-grained within-population dynamics

```r
map <- world(xrange = c(0, 10), yrange = c(0, 10),
             landscape = region(center = c(5, 5), radius = 5))

p1 <- population("pop1", time = 1, N = 2000, map = map, competition = 0)
p2 <- population("pop2", time = 1, N = 2000, map = map, competition = 9)
p3 <- population("pop3", time = 1, N = 2000, map = map, competition = 6)
p4 <- population("pop4", time = 1, N = 2000, map = map, competition = 5)
p5 <- population("pop5", time = 1, N = 2000, map = map, competition = 4)
p6 <- population("pop6", time = 1, N = 2000, map = map, competition = 3)
p7 <- population("pop7", time = 1, N = 2000, map = map, competition = 2)
p8 <- population("pop8", time = 1, N = 2000, map = map, competition = 1)

model <- compile_model(
  populations = list(p1, p2, p3, p4, p5, p6, p7, p8),
  generation_time = 1, sim_length = 5000, resolution = 0.1,
  mating = 0.1, dispersal = 0.05
)
```

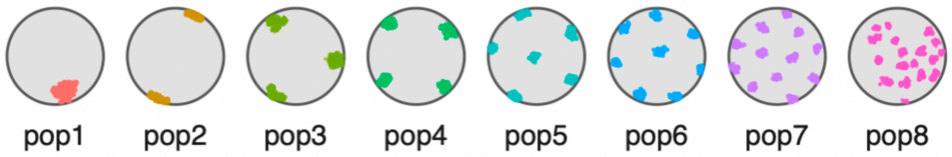


spatial distributions emerged in the simulation

Fine-grained within-population dynamics

```
map <- world(xrange = c(0, 10), yrange = c(0, 10),
             landscape = region(center = c(5, 5), radius = 5))

p1 <- population("pop1", time = 1, N = 2000, map = map, competition = 0)
p2 <- population("pop2", time = 1, N = 2000, map = map, competition = 9)
p3 <- population("pop3", time = 1, N = 2000, map = map, competition = 6)
p4 <- population("pop4", time = 1, N = 2000, map = map, competition = 5)
p5 <- population("pop5", time = 1, N = 2000, map = map, competition = 4)
p6 <- population("pop6", time = 1, N = 2000, map = map, competition = 3)
p7 <- population("pop7", time = 1, N = 2000, map = map, competition = 2)
p8 <- population("pop8", time = 1, N = 2000, map = map, competition = 1)

model <- compile_model(
  populations = list(p1, p2, p3, p4, p5, p6, p7, p8),
  generation_time = 1, sim_length = 5000, resolution = 0.1,
  mating = 0.1, dispersal = 0.05
)
```

```
ts <- slim(model, sequence_length = 10e6, recombination_rate = 1e-8) %>%
  ts_simplify() %>%
  ts_mutate(mutation_rate = 1e-7)
```



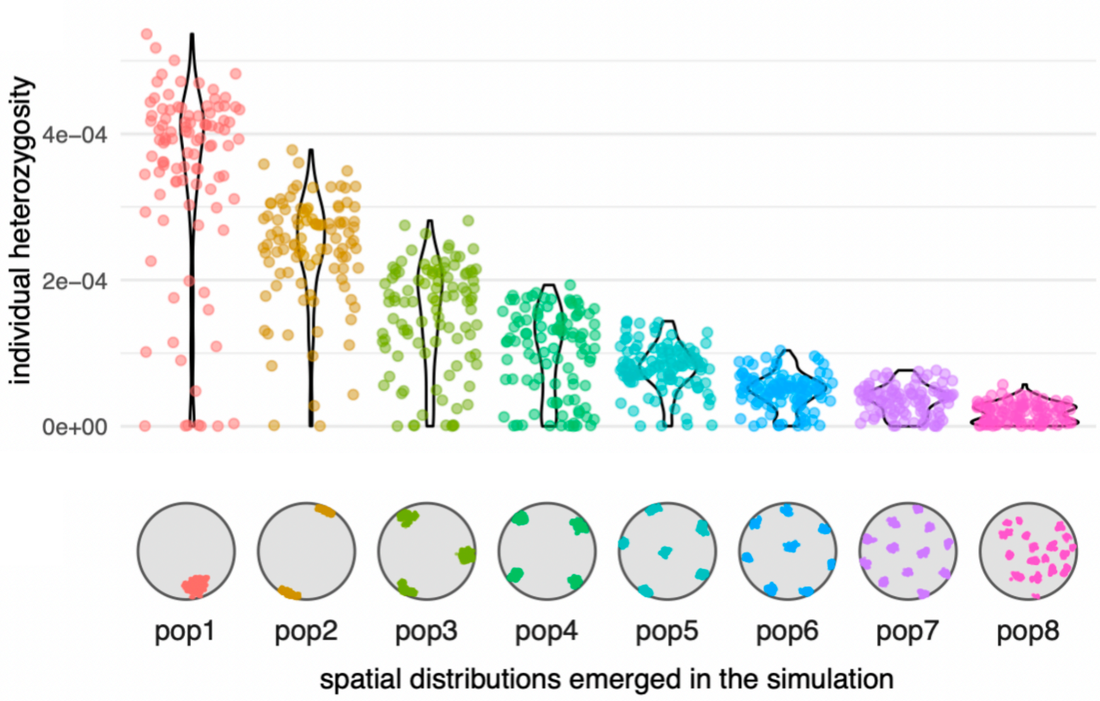
spatial distributions emerged in the simulation

Fine-grained within-population dynamics

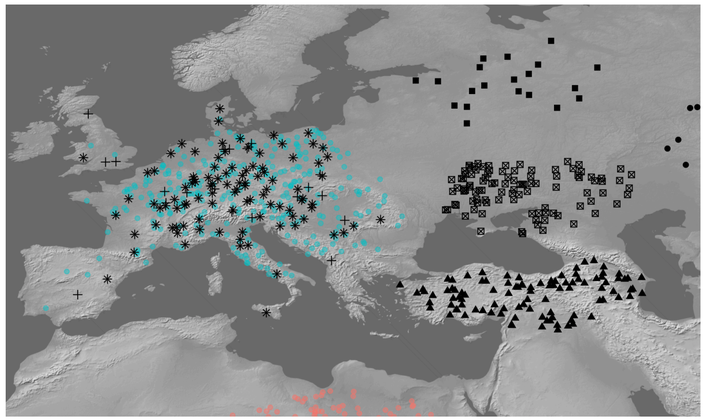
```
map <- world(xrange = c(0, 10), yrange = c(0, 10),
             landscape = region(center = c(5, 5), radius = 5))

p1 <- population("pop1", time = 1, N = 2000, map = map, competition = 0)
p2 <- population("pop2", time = 1, N = 2000, map = map, competition = 9)
p3 <- population("pop3", time = 1, N = 2000, map = map, competition = 6)
p4 <- population("pop4", time = 1, N = 2000, map = map, competition = 5)
p5 <- population("pop5", time = 1, N = 2000, map = map, competition = 4)
p6 <- population("pop6", time = 1, N = 2000, map = map, competition = 3)
p7 <- population("pop7", time = 1, N = 2000, map = map, competition = 2)
p8 <- population("pop8", time = 1, N = 2000, map = map, competition = 1)

model <- compile_model(
  populations = list(p1, p2, p3, p4, p5, p6, p7, p8),
  generation_time = 1, sim_length = 5000, resolution = 0.1,
  mating = 0.1, dispersal = 0.05
)
```

```
ts <- slim(model, sequence_length = 10e6, recombination_rate = 1e-8) %>%
  ts_simplify() %>%
  ts_mutate(mutation_rate = 1e-7)
```

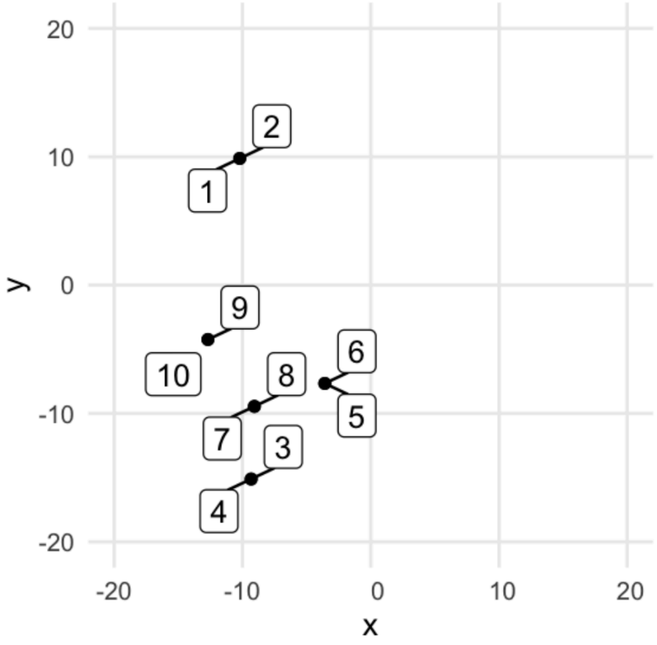
```
heterozygosity <- ts_samples(ts) %>%
  group_by(pop) %>%
  sample_n(100) %>%
  mutate(pi = ts_diversity(ts, name)$diversity)
```



Simulated spatio-temporal genomic data

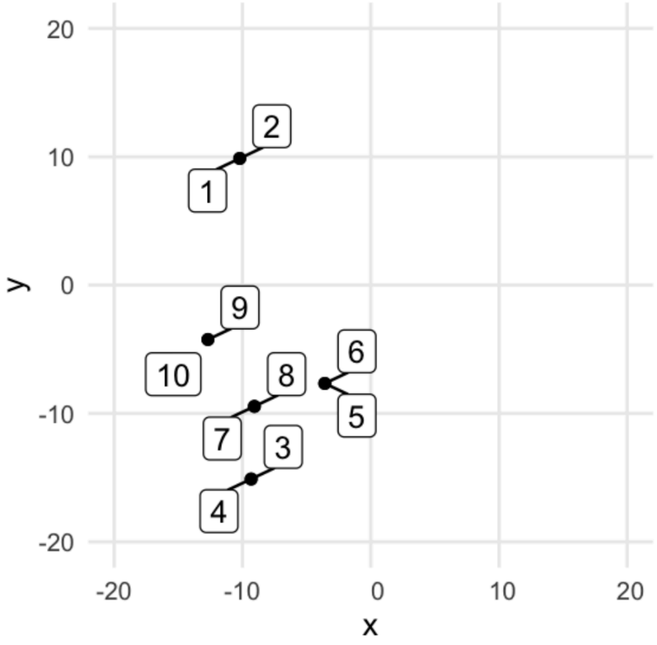


Simulated spatio-temporal genomic data



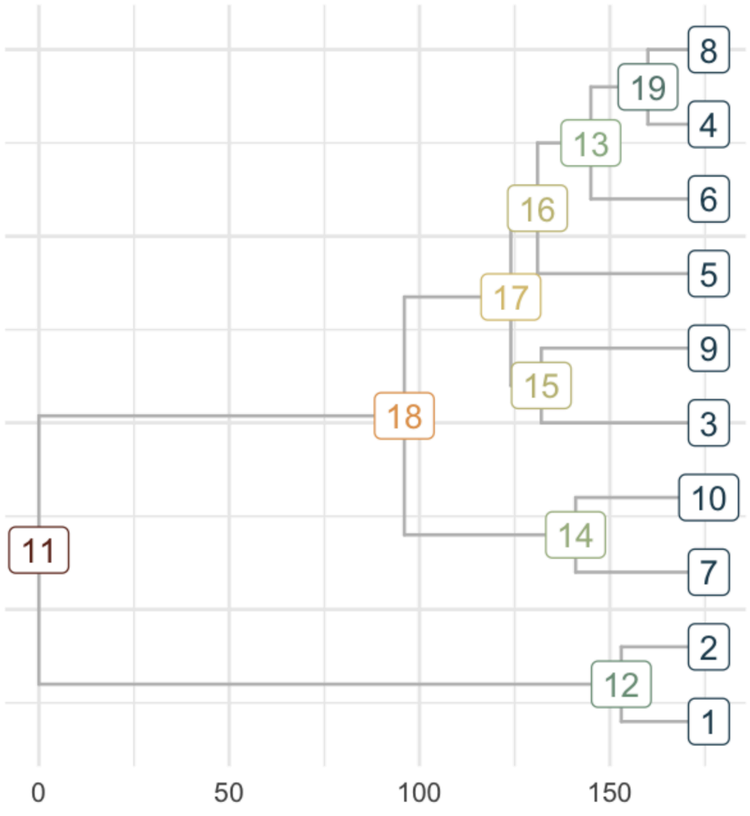
a sample of 5 individuals
(two chromosomes from each)

Simulated spatio-temporal genomic data



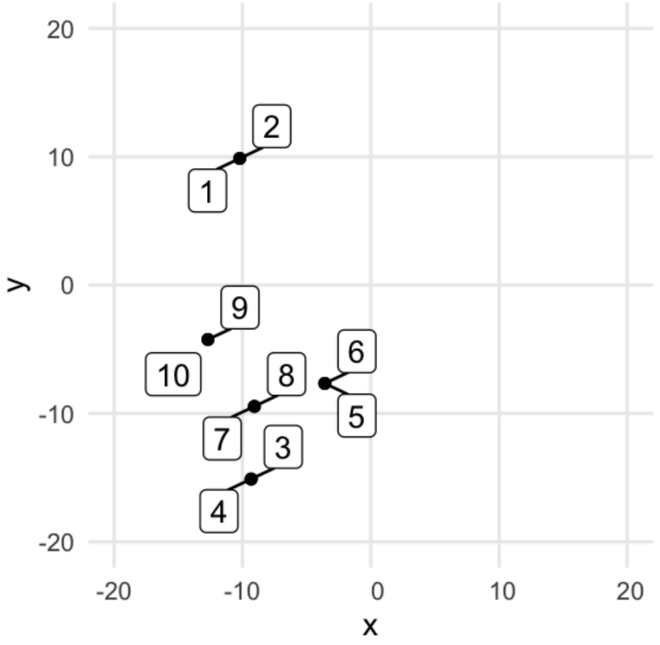
a sample of 5 individuals
(two chromosomes from each)

A single phylogenetic tree out of many more along the genome



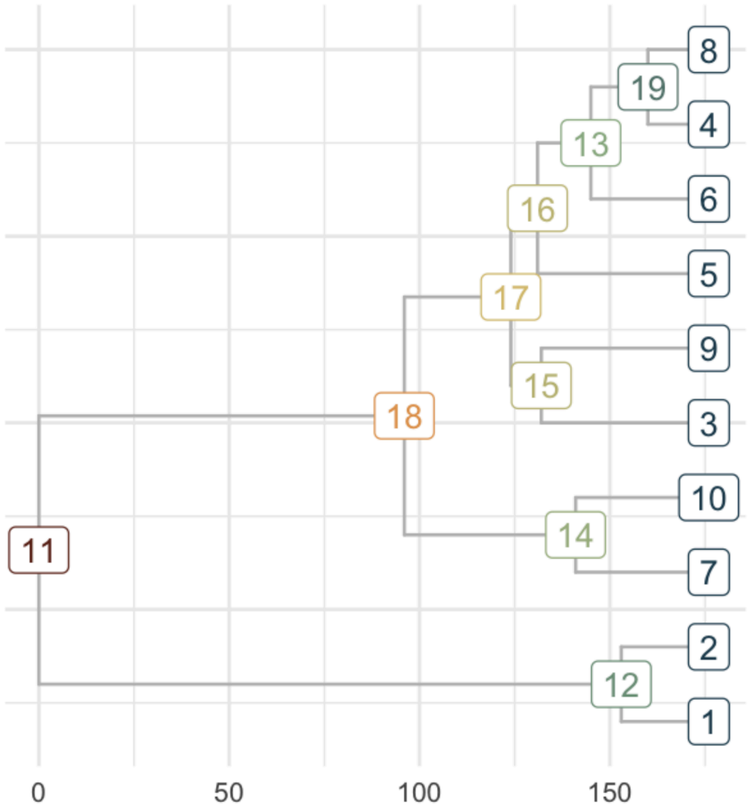
`ts_phylo()`

Simulated spatio-temporal genomic data



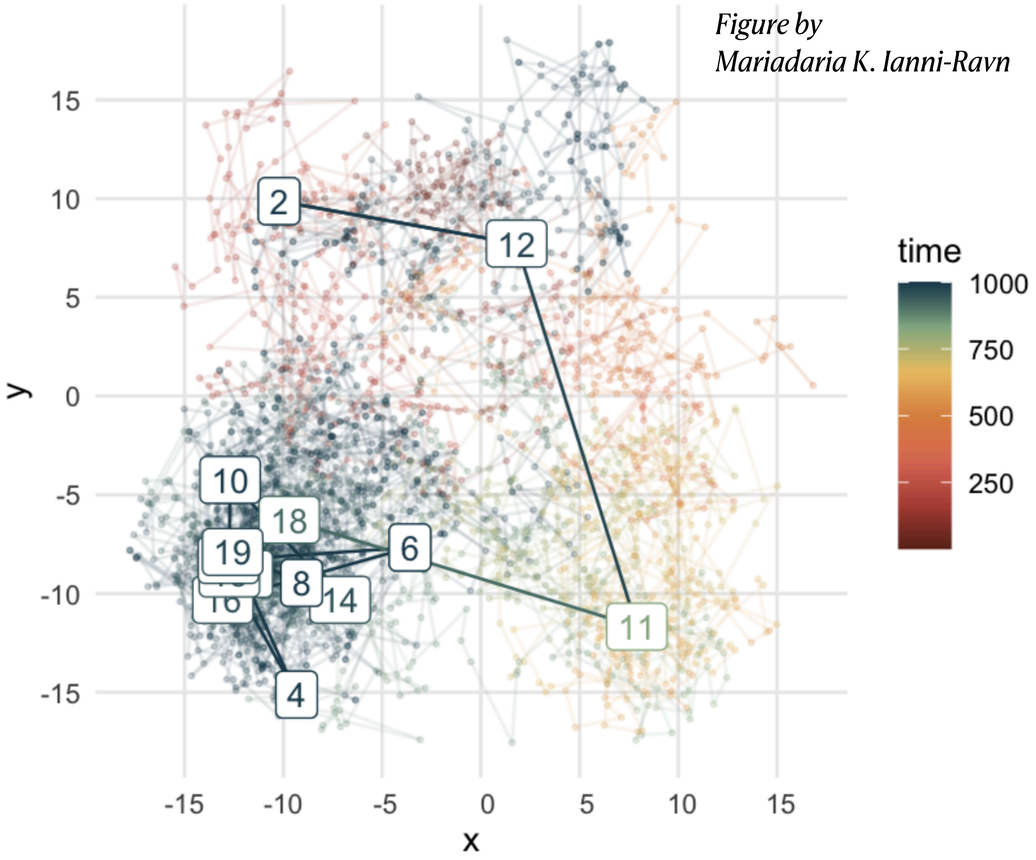
a sample of 5 individuals
(two chromosomes from each)

A single phylogenetic tree out of many more along the genome



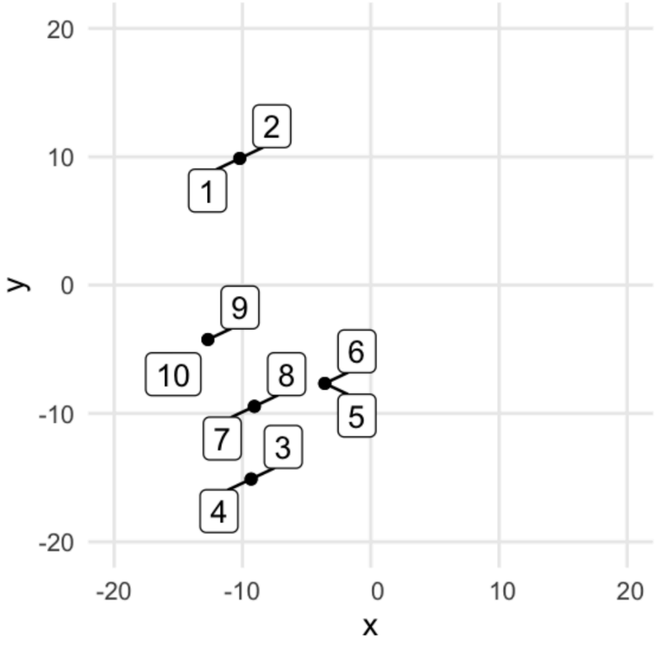
`ts_phylo()`

We have the full genealogical spatio-temporal ancestry!



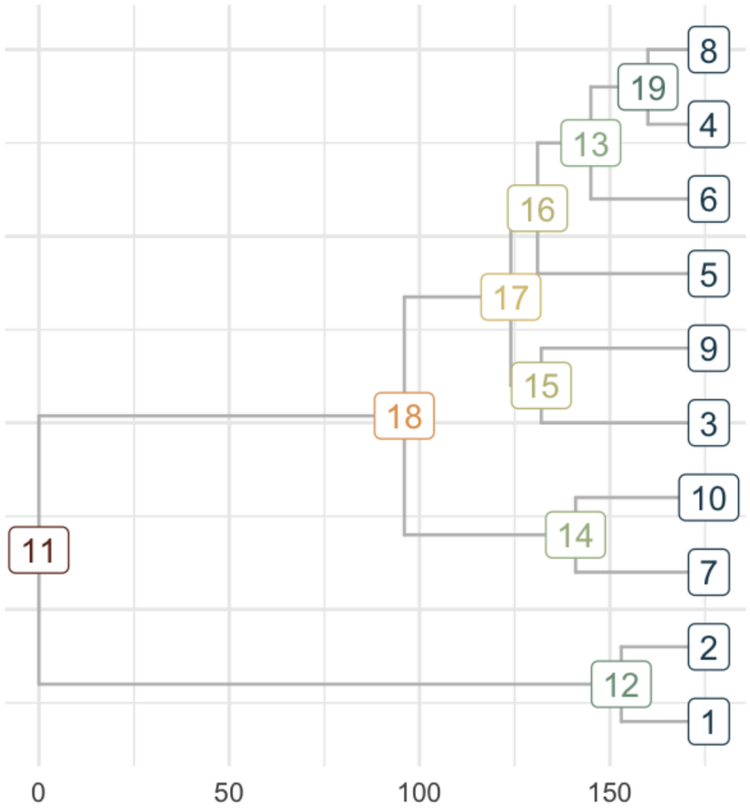
`ts_nodes(),ts_edges()`

Simulated spatio-temporal genomic data



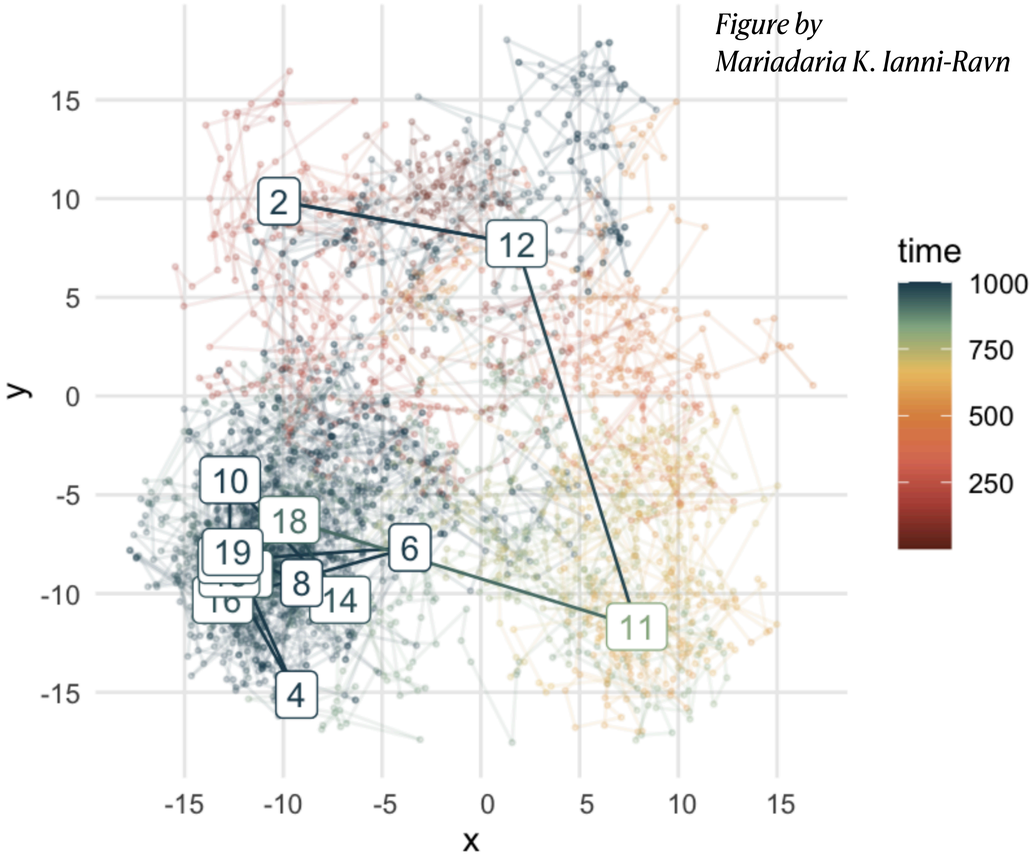
a sample of 5 individuals
(two chromosomes from each)

A single phylogenetic tree out of many more along the genome



`ts_phylo()`

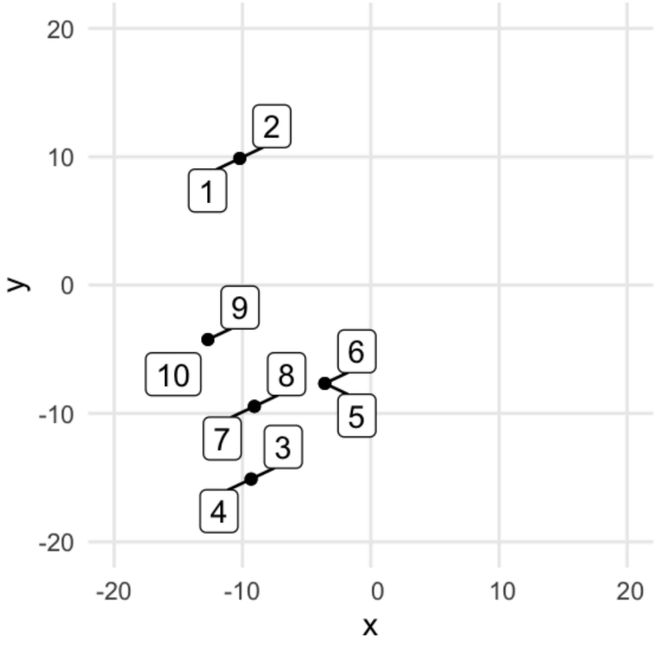
We have the full genealogical spatio-temporal ancestry!



`ts_nodes(),ts_edges()`

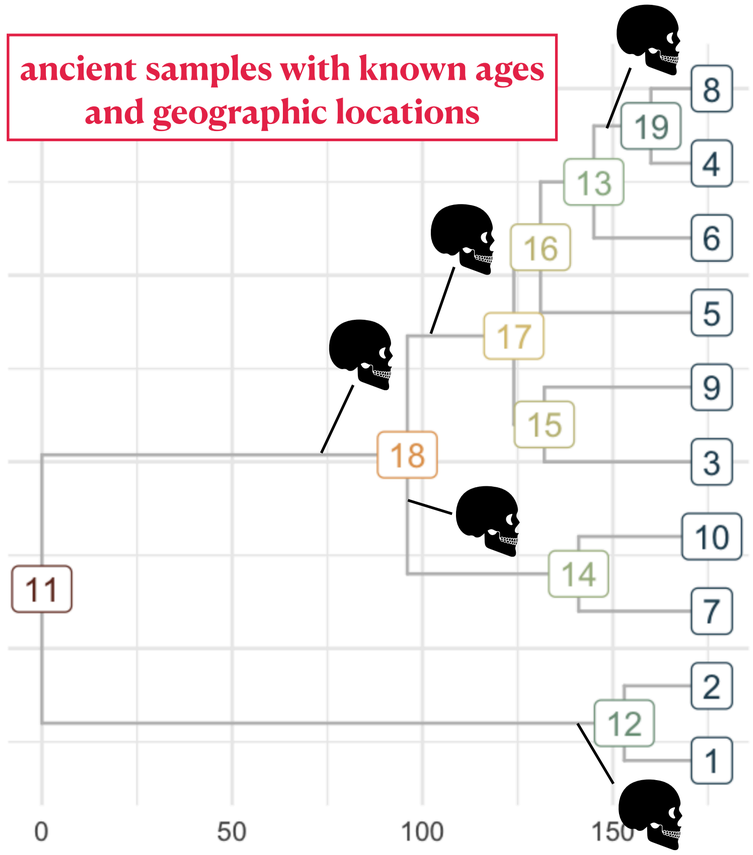
In progress: estimating ancestral locations from empirical data

Simulated spatio-temporal genomic data



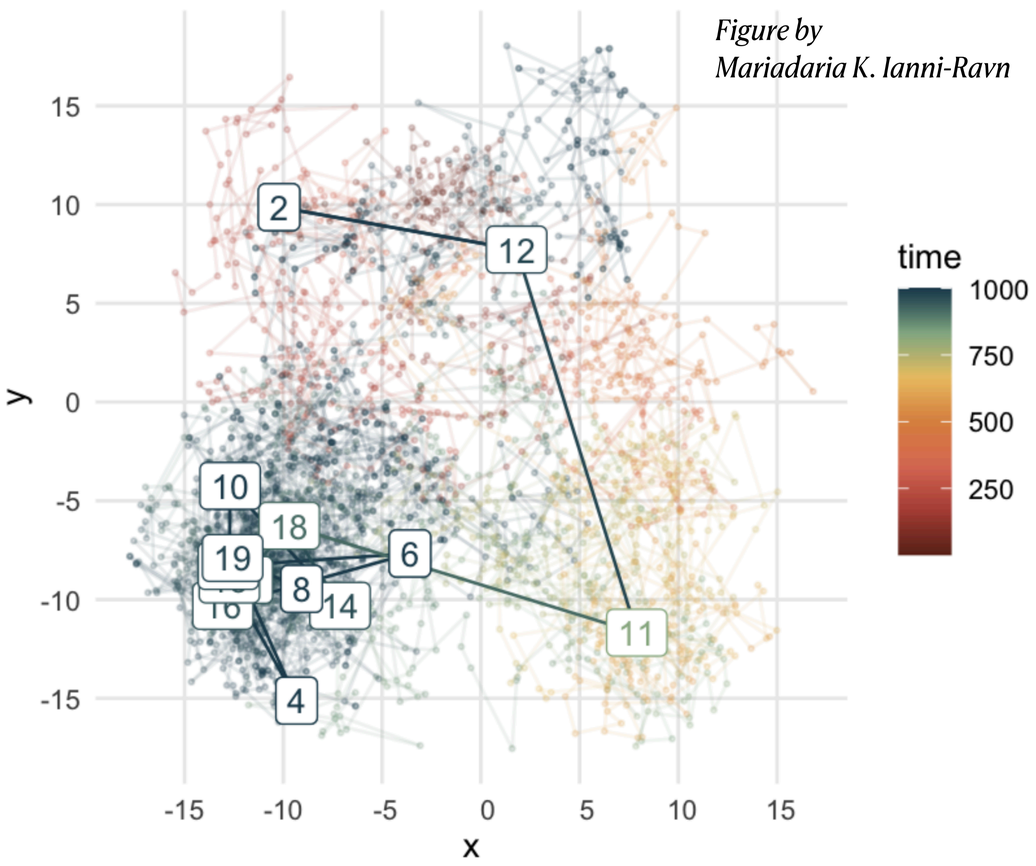
a sample of 5 individuals
(two chromosomes from each)

A single phylogenetic tree out of many more along the genome



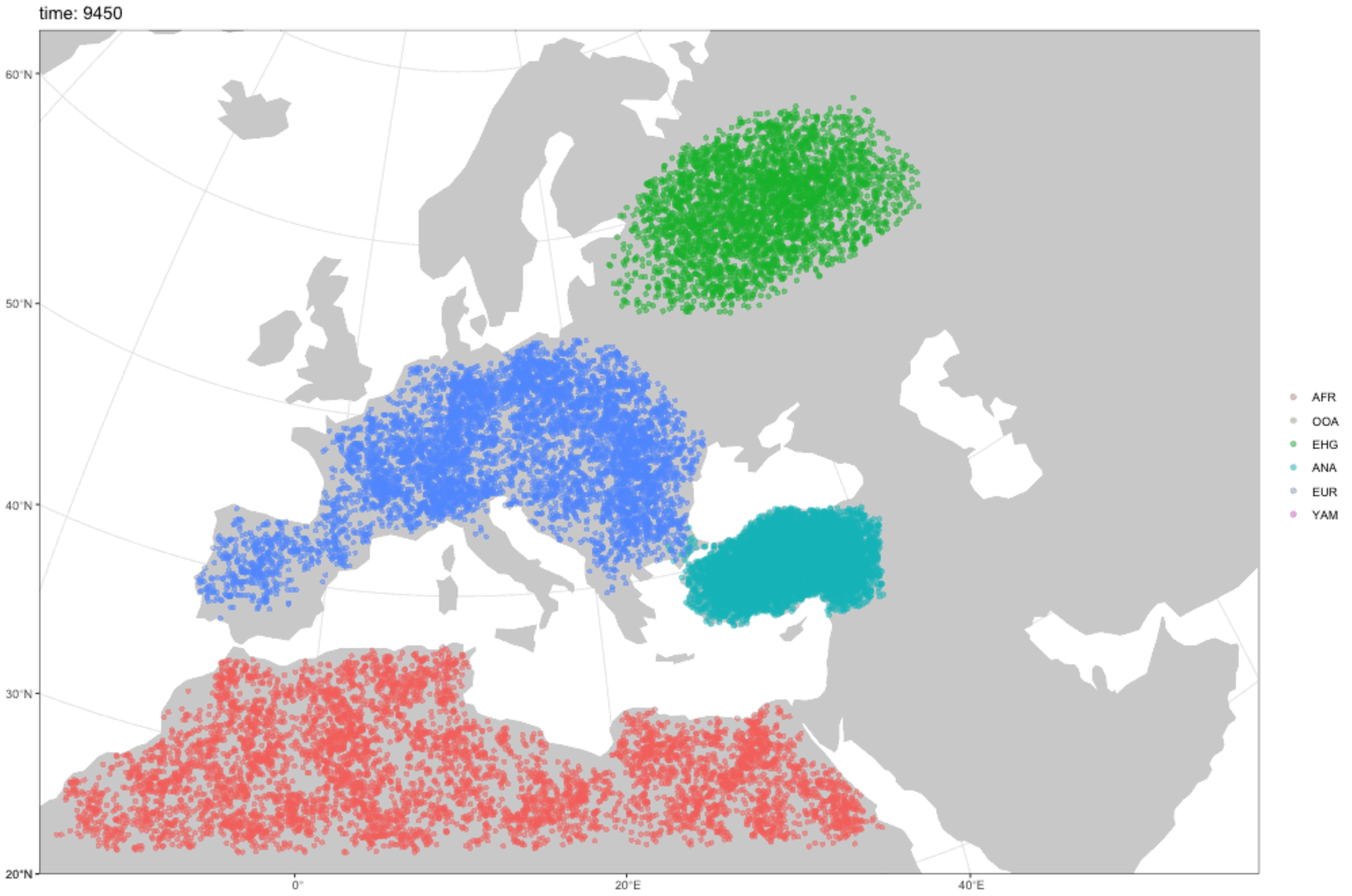
`ts_phylo()`

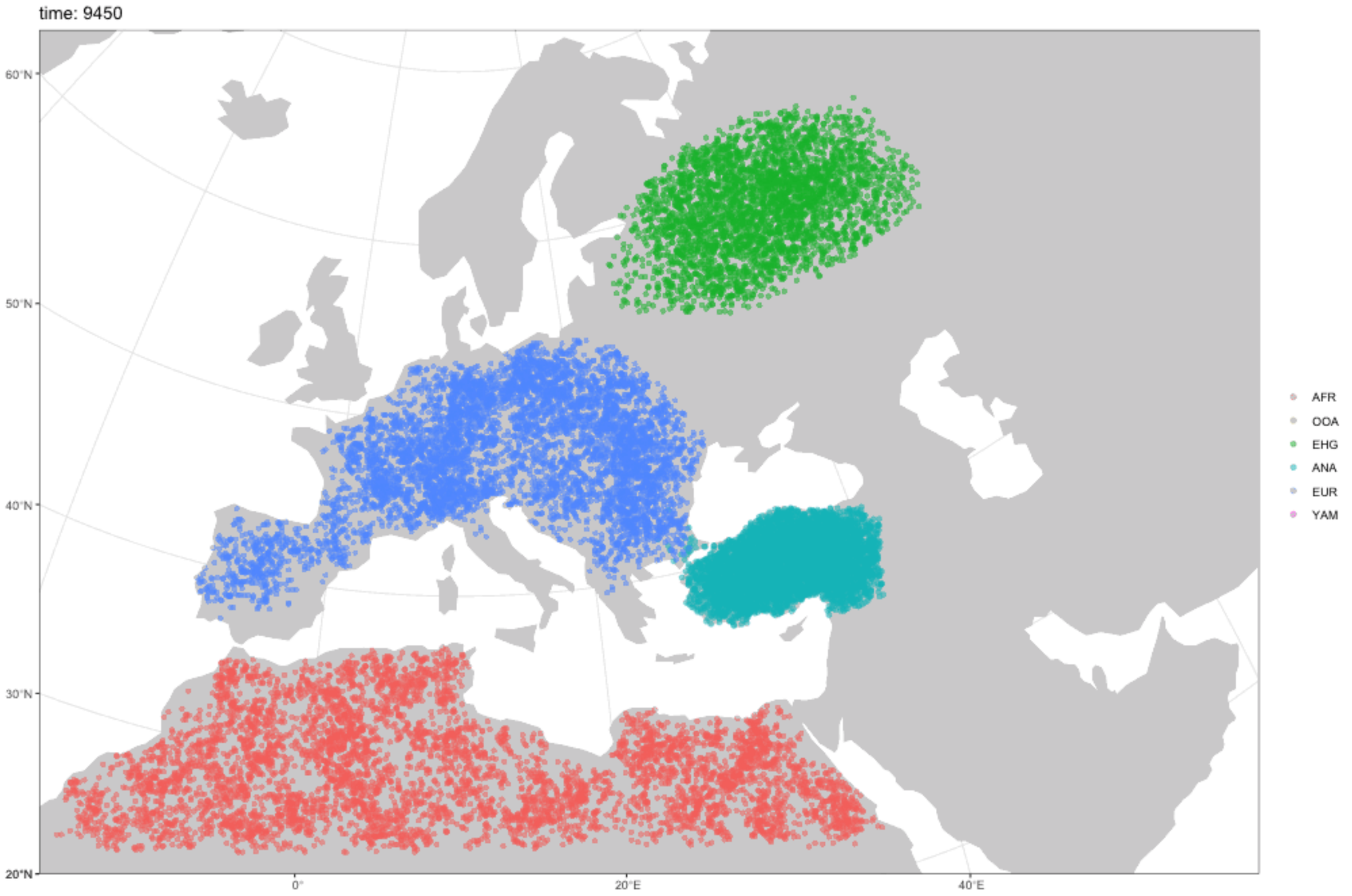
We have the full genealogical spatio-temporal ancestry!

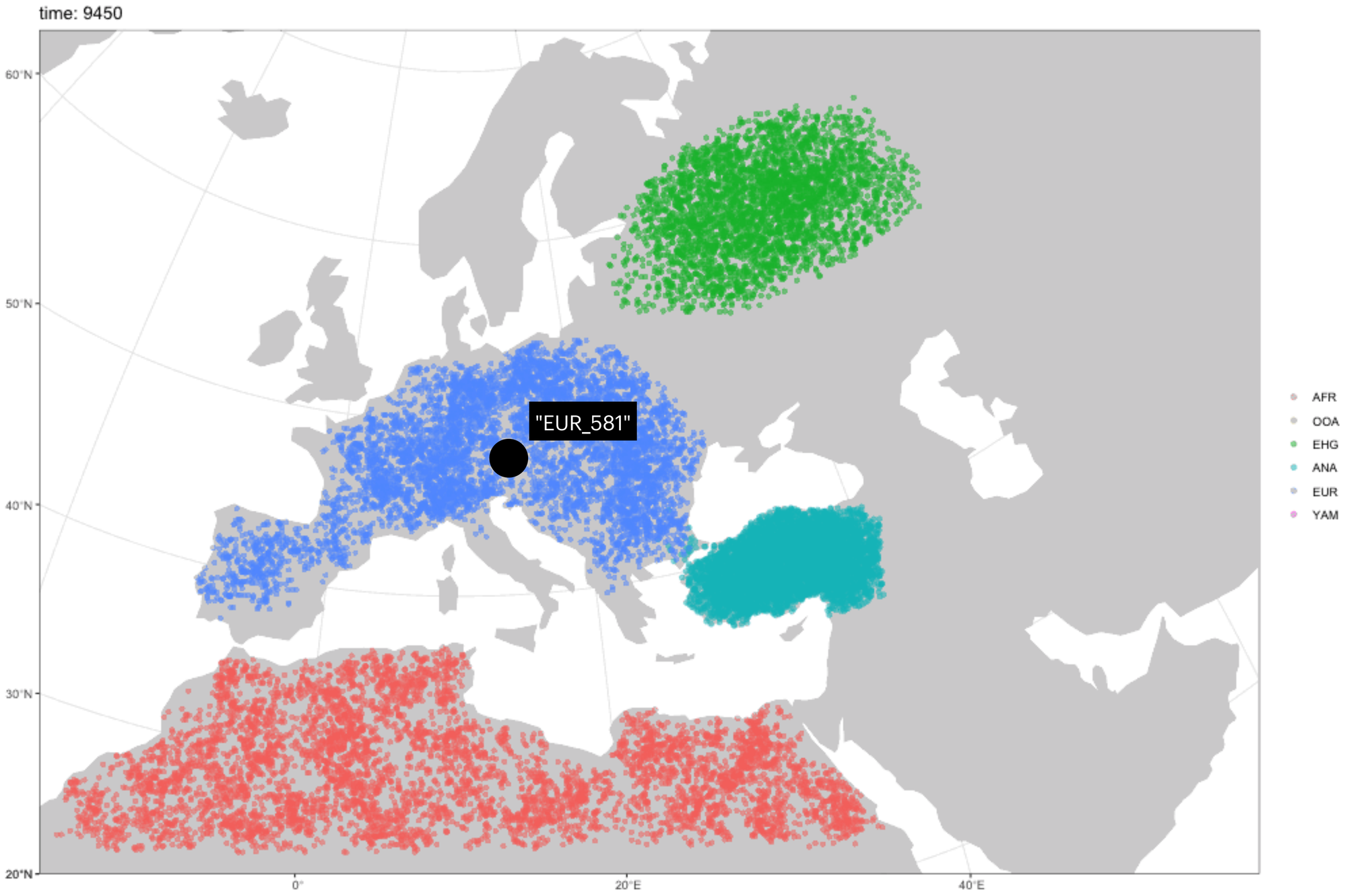


`ts_nodes(),ts_edges()`

In progress: estimating ancestral locations from empirical data

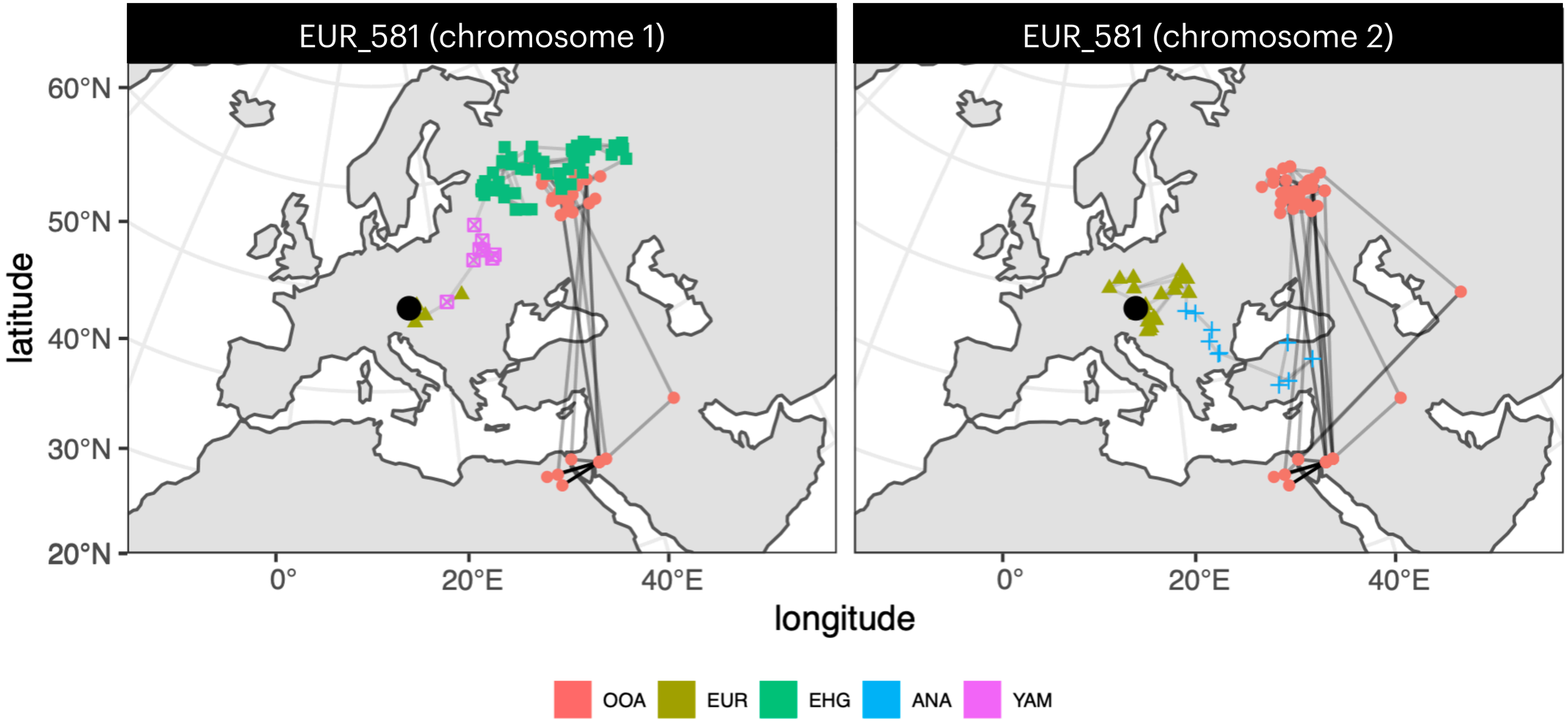

time: 9450
60°N
50°N
40°N
30°N
20°N
0°
20°E
40°E
AFR
OOA
EHG
ANA
EUR
YAM


time: 9450
60°N
50°N
40°N
30°N
20°N
0°
20°E
40°E
AFR
OOA
EHG
ANA
EUR
YAM


time: 9450
60°N
50°N
40°N
30°N
20°N
0°
20°E
40°E
"EUR_581"
AFR
OOA
EHG
ANA
EUR
YAM

Full spatio-temporal ancestry of a single "European"

(across just 200 kb tree sequence)



Thanks!

UNIVERSITY OF COPENHAGEN

Fernando Racimo & the group
Benjamin C. Haller, **Peter L. Ralph**

GEOGENETICS CENTRE
LUNDBECK FOUNDATION

Emma Prantoni
Mariadaria K. Ianni-Ravn
Jule Anne Drewalowski

novo nordisk fonden

Lundbeck Ancient Genomics Consortium
Danish popgen community

LUNDBECKFONDEN

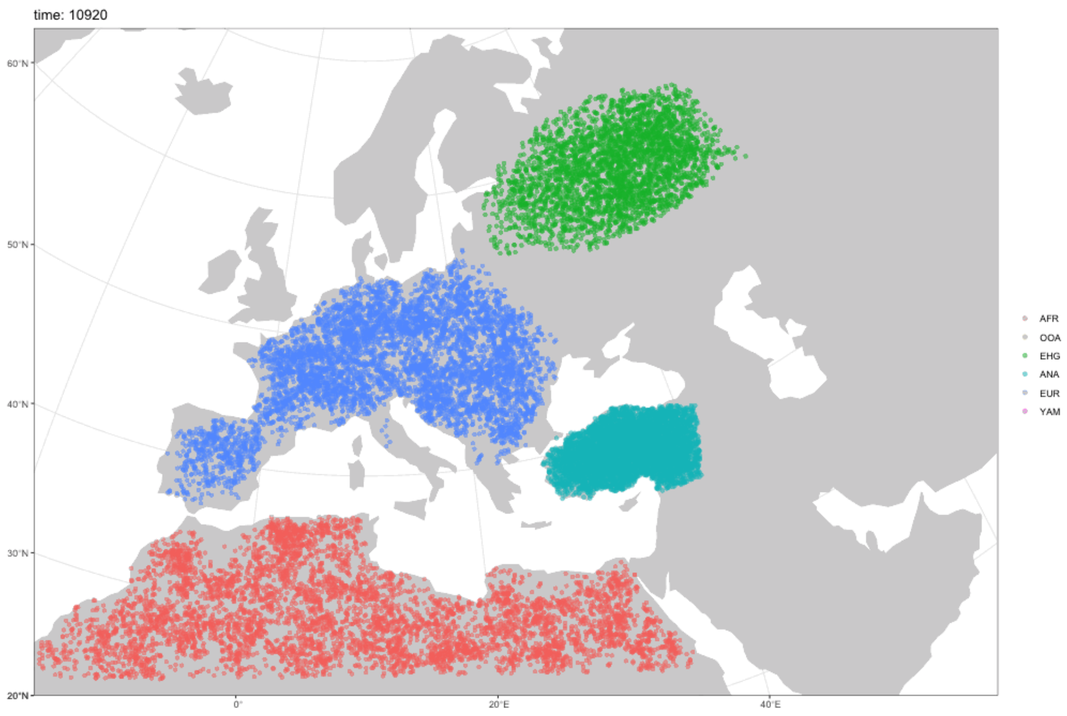
Documentation, tutorials: www.slendr.net

bioRxiv pre-print: tinyurl.com/mp-slendr

Contact: mp@bodkan.net

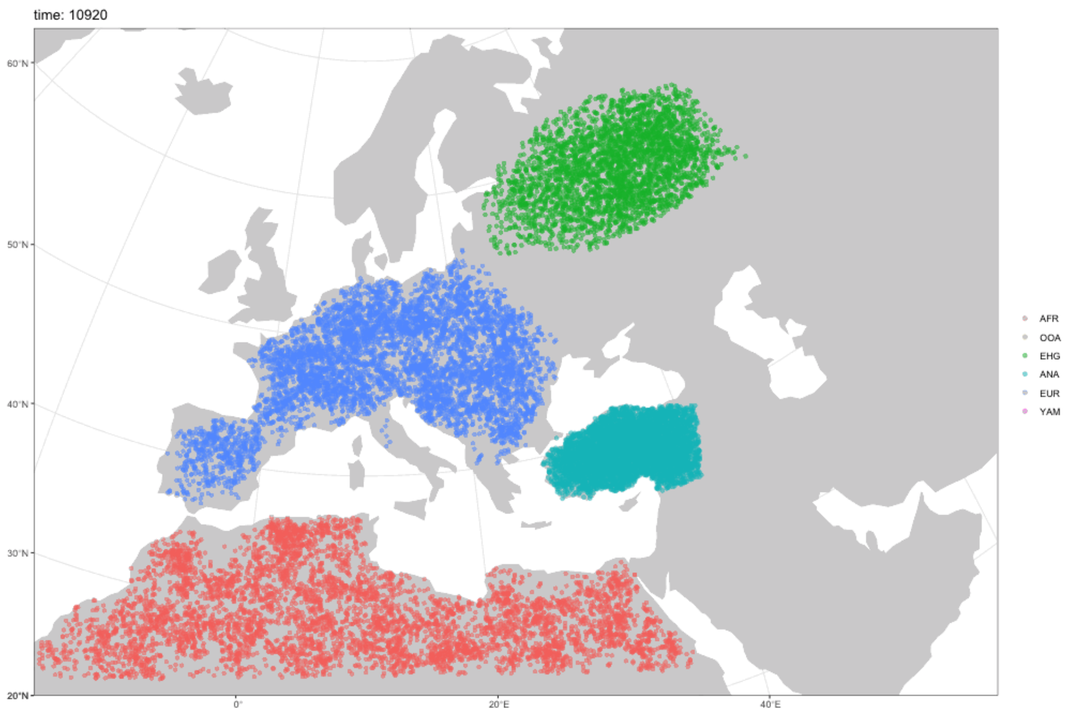
Next steps: **realistic raster-based ecological landscapes**

At the moment, each point on a map is either habitable or non-habitable



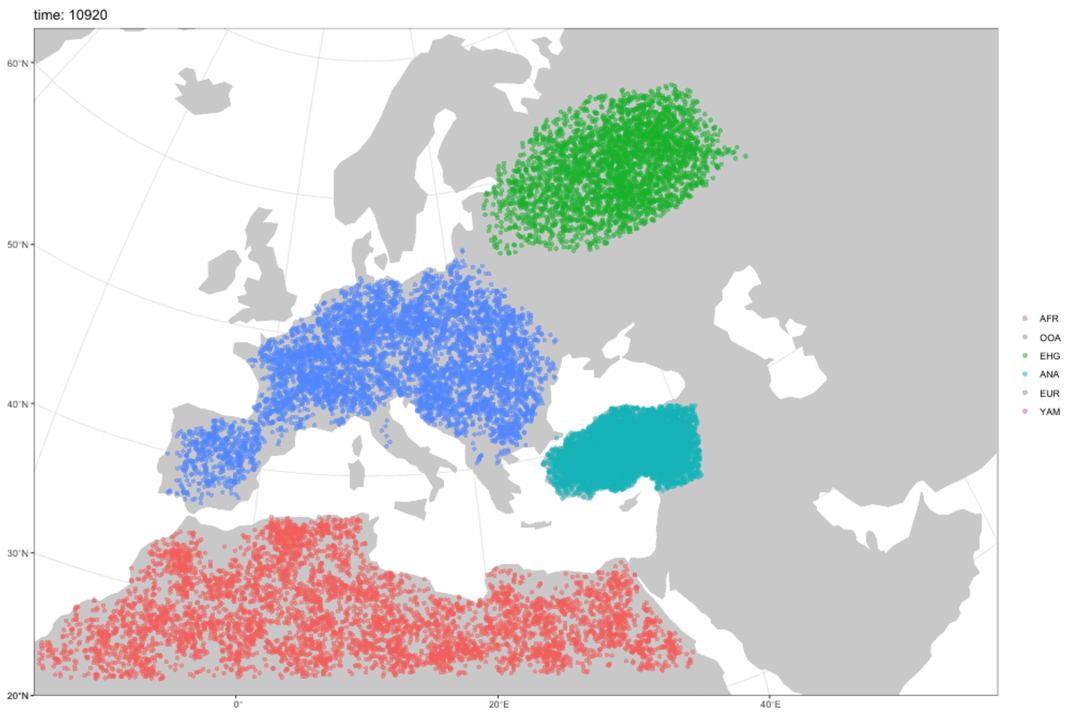
Next steps: **realistic raster-based ecological landscapes**

At the moment, each point on a map is either habitable or non-habitable



Next steps: **realistic raster-based ecological landscapes**

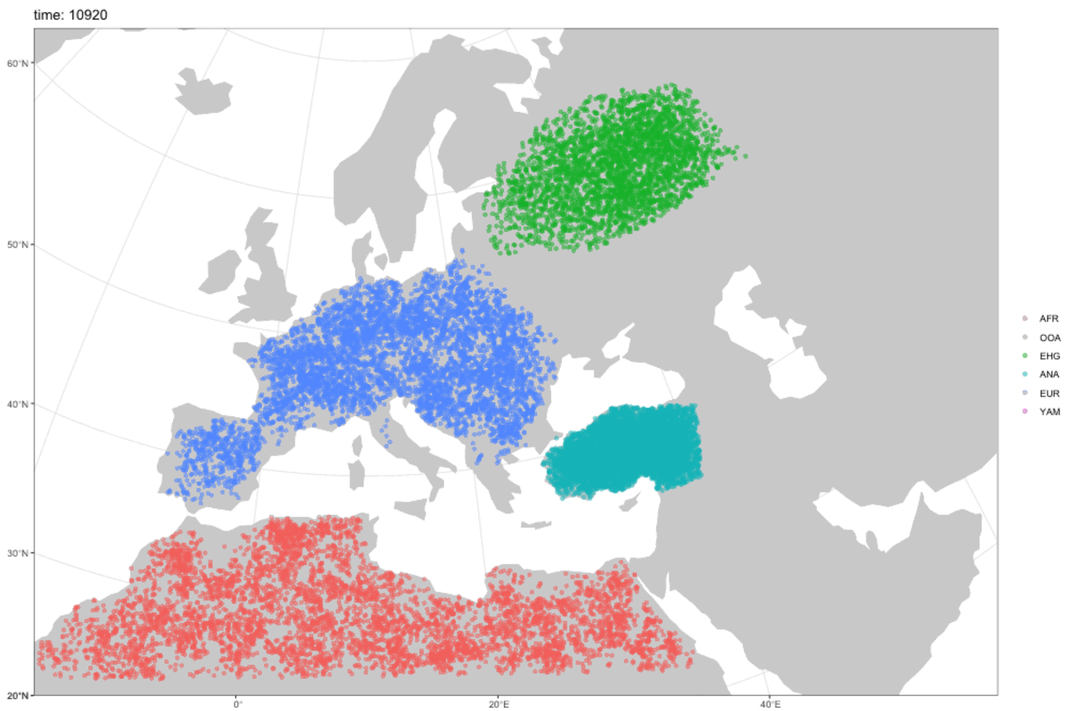
At the moment, each point on a map is either habitable or non-habitable



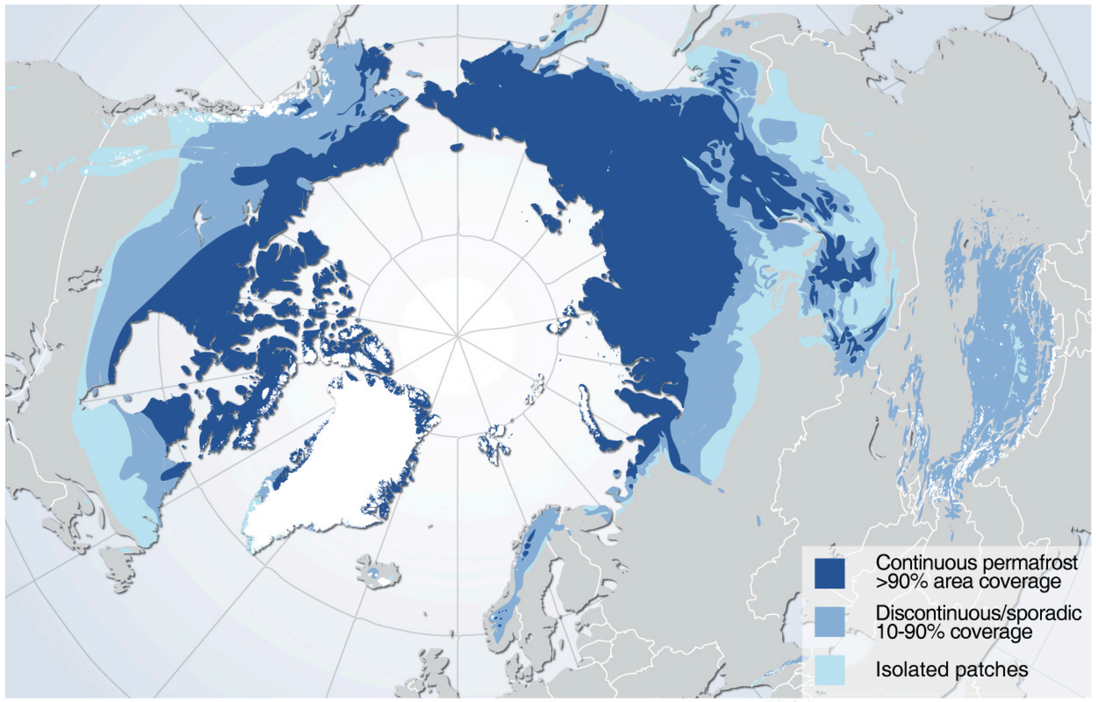
We can do much better than that!

Next steps: **realistic raster-based ecological landscapes**

At the moment, each point on a map is either habitable or non-habitable



We can do much better than that!

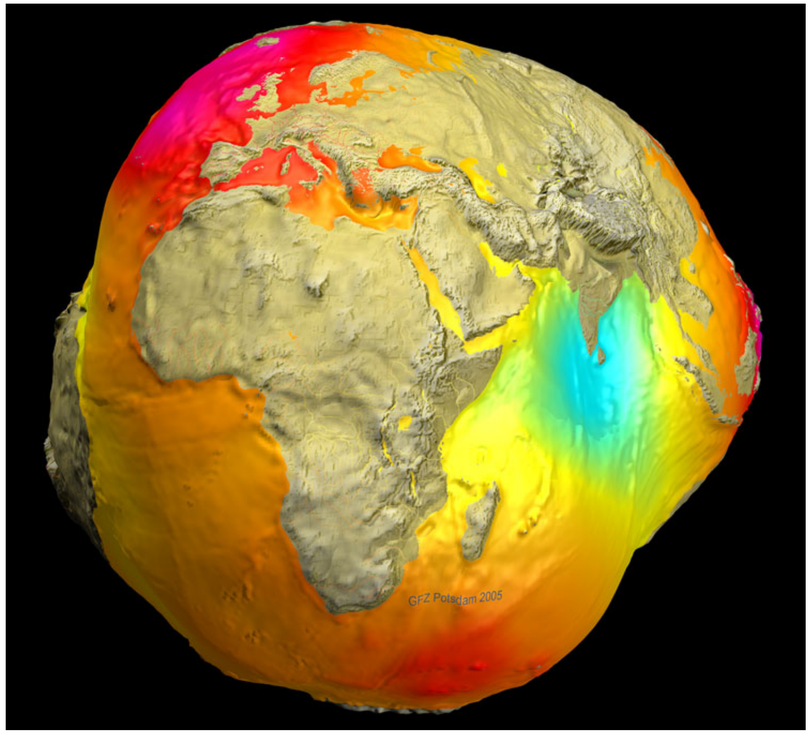


Challenges: Need to model "carrying capacity", non-Wright-Fisher setting, many more model parameters that need to be specified (and can be mis-specified...)

Why an explicit *Coordinate Reference System* (CRS)?

Why an explicit *Coordinate Reference System* (CRS)?

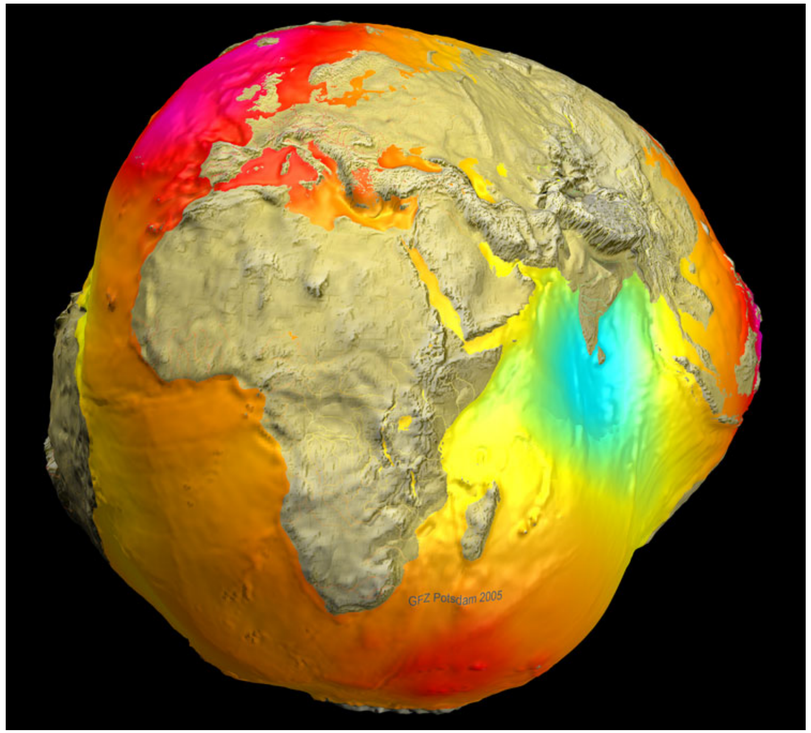
Earth is a ~~potato~~ ~~sphere~~ ellipsoid...



Astronomy Picture of the Day: 2014 December 15

Why an explicit *Coordinate Reference System* (CRS)?

Earth is a ~~potato~~ ~~sphere~~ ellipsoid...



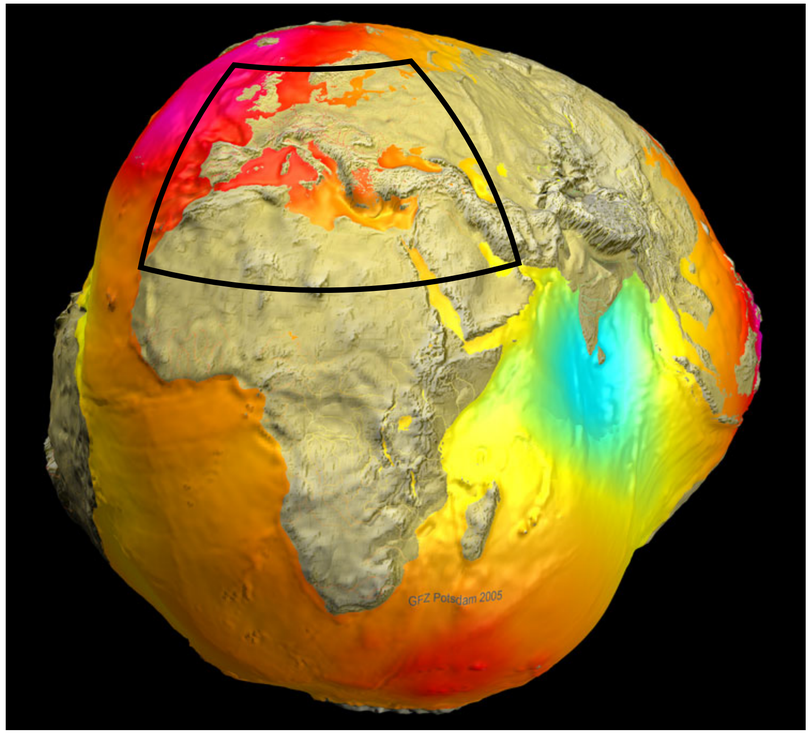
Astronomy Picture of the Day: 2014 December 15

... but we need to simulate on a two-dimensional surface

(and maintain distances, shapes, proportions)

Why an explicit *Coordinate Reference System* (CRS)?

Earth is a ~~potato~~ ~~sphere~~ ellipsoid...



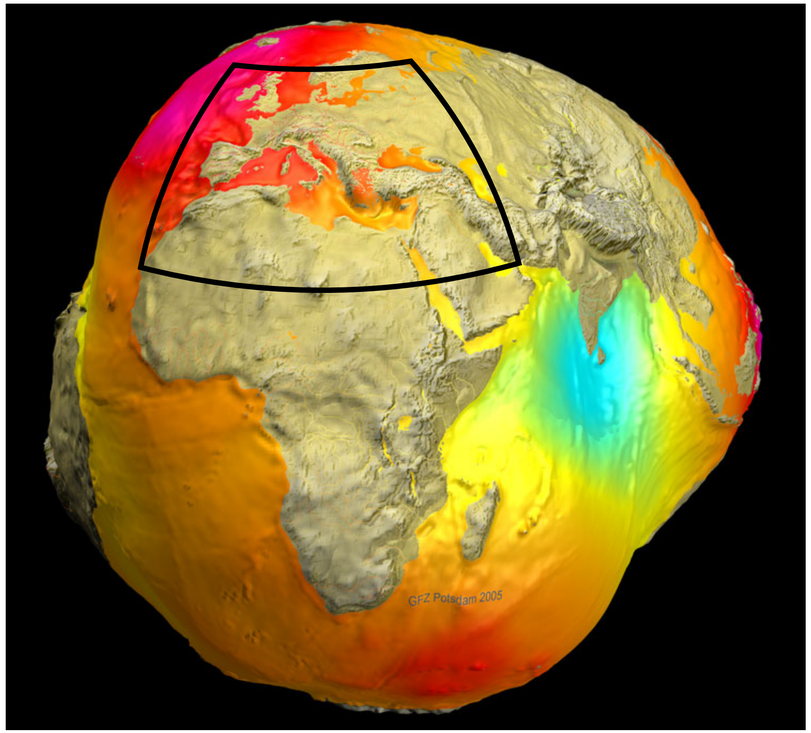
Astronomy Picture of the Day: 2014 December 15

... but we need to simulate on a two-dimensional surface

(and maintain distances, shapes, proportions)

Why an explicit *Coordinate Reference System* (CRS)?

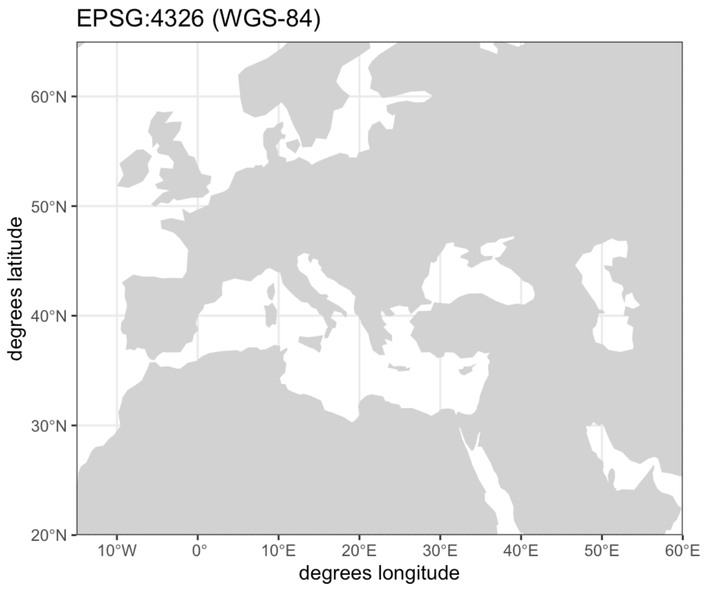
Earth is a ~~potato~~ ~~sphere~~ ellipsoid...



Astronomy Picture of the Day: 2014 December 15

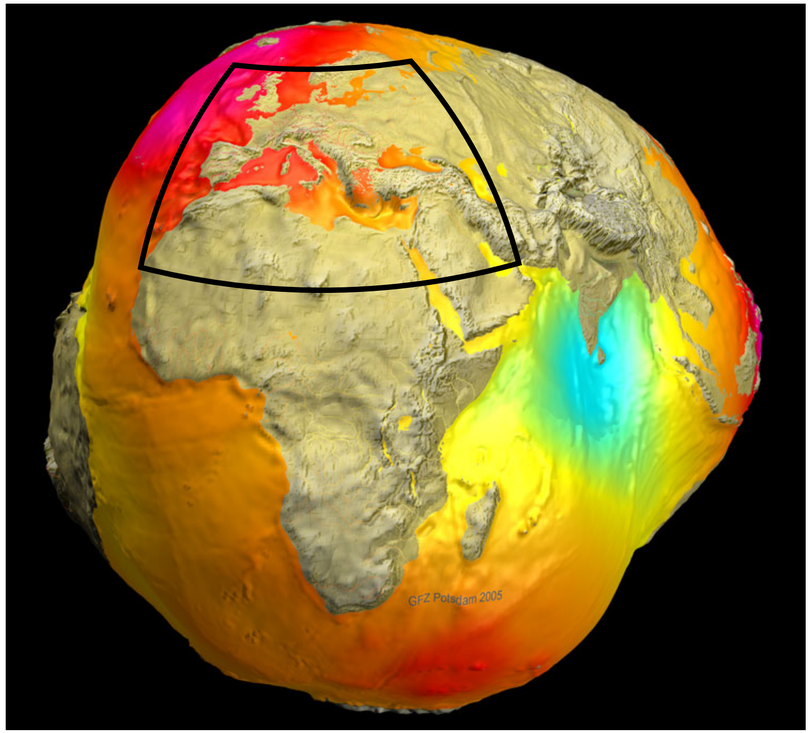
... but we need to simulate on a two-dimensional surface

(and maintain distances, shapes, proportions)



Why an explicit *Coordinate Reference System* (CRS)?

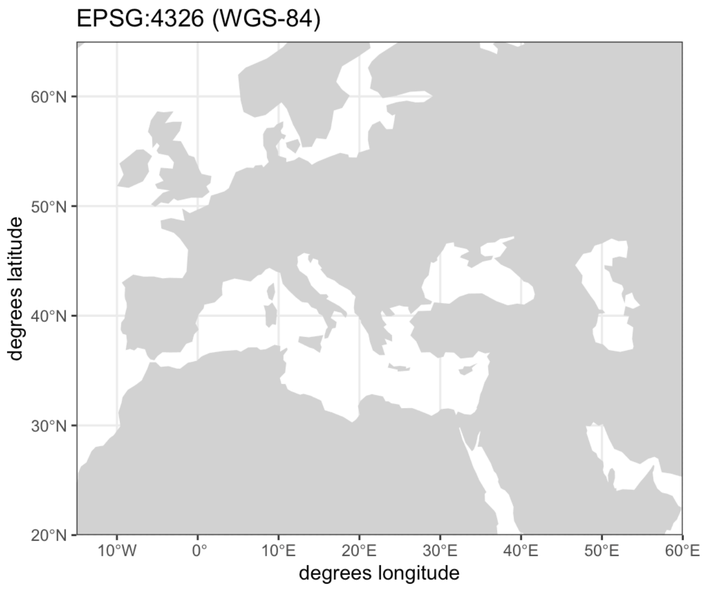
Earth is a ~~potato~~ ~~sphere~~ ellipsoid...

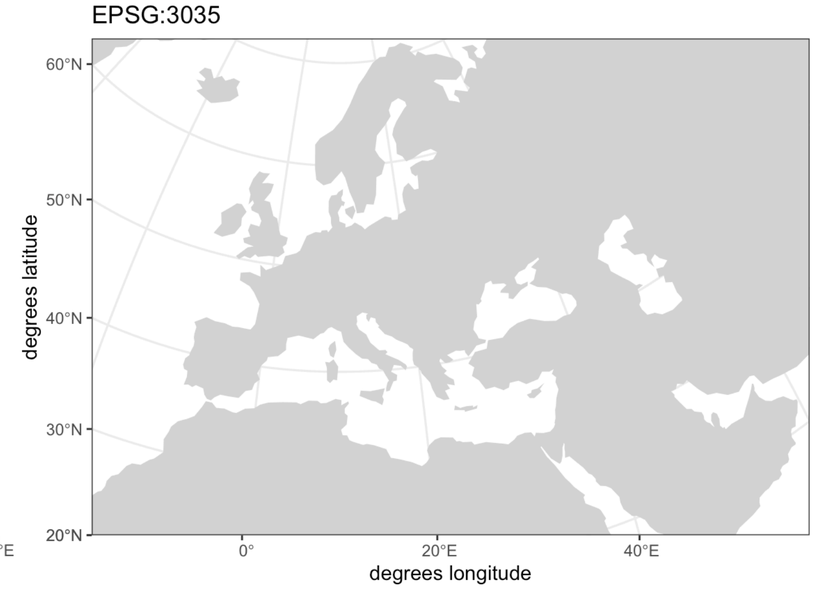


Astronomy Picture of the Day: 2014 December 15

... but we need to simulate on a two-dimensional surface

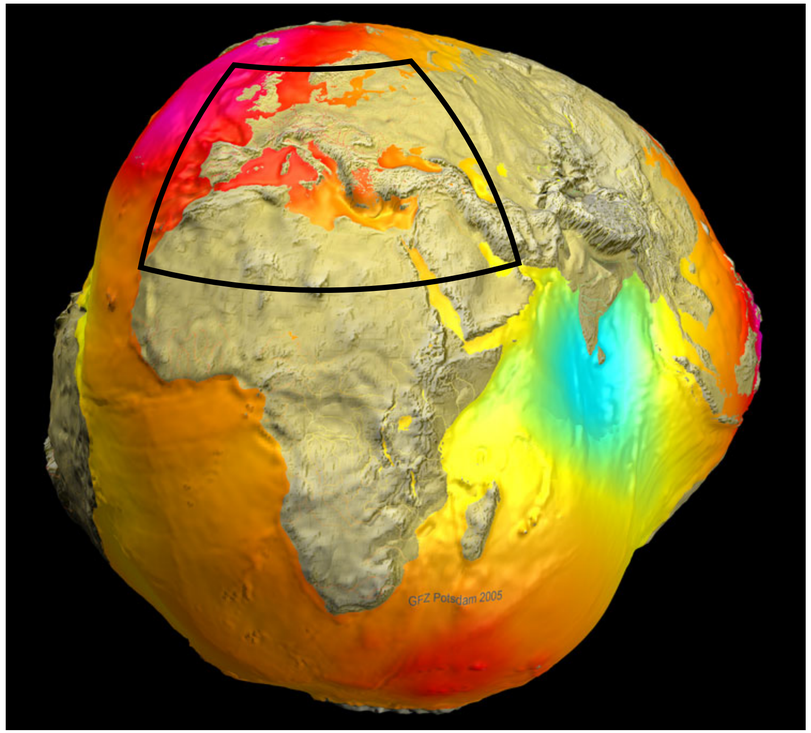
(and maintain distances, shapes, proportions)





Why an explicit *Coordinate Reference System* (CRS)?

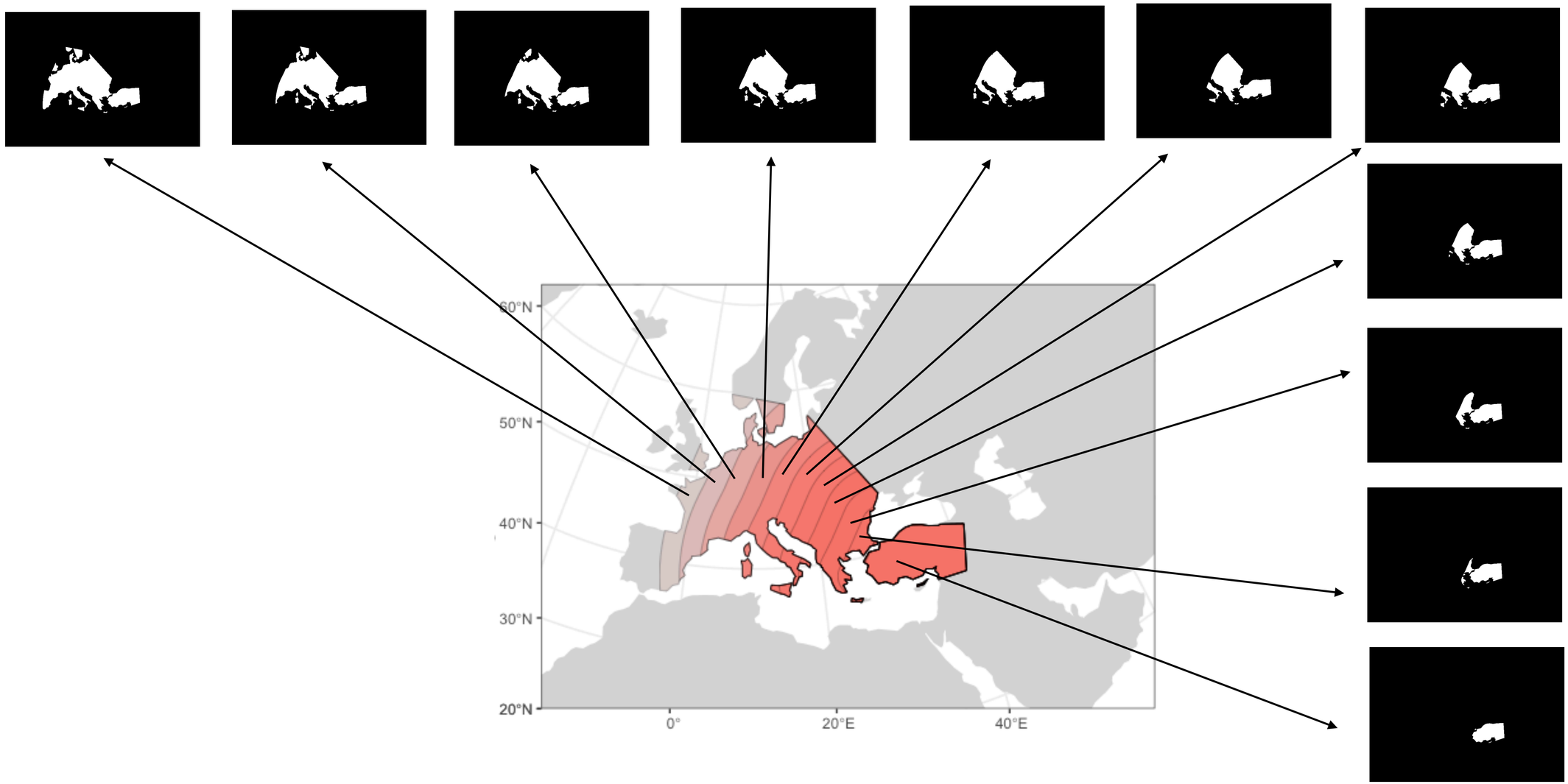
Earth is a ~~potato~~ ~~sphere~~ ellipsoid...



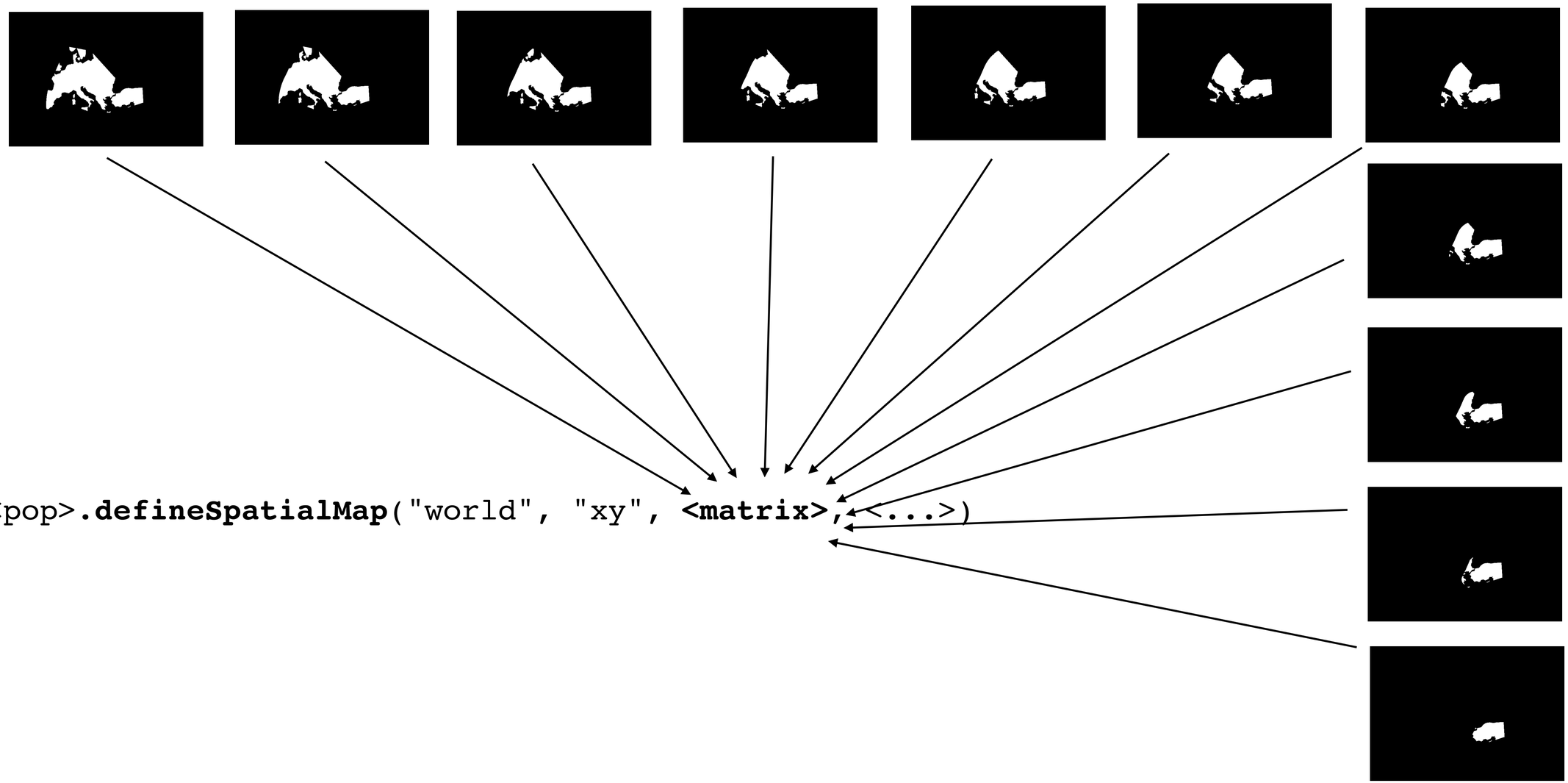
Astronomy Picture of the Day: 2014 December 15

Vector → raster conversion during compilation

Vector → raster conversion during compilation



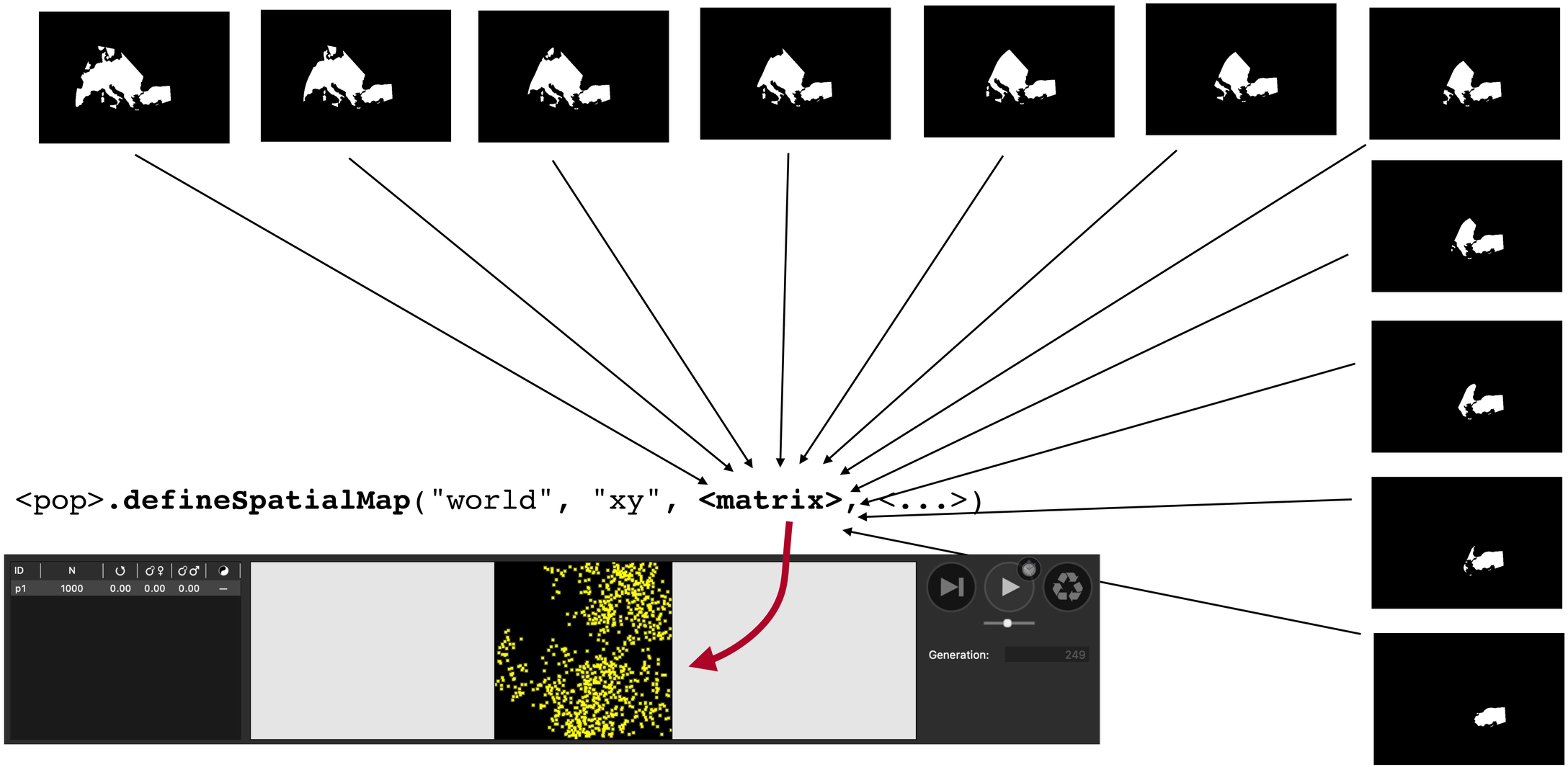
In the background...

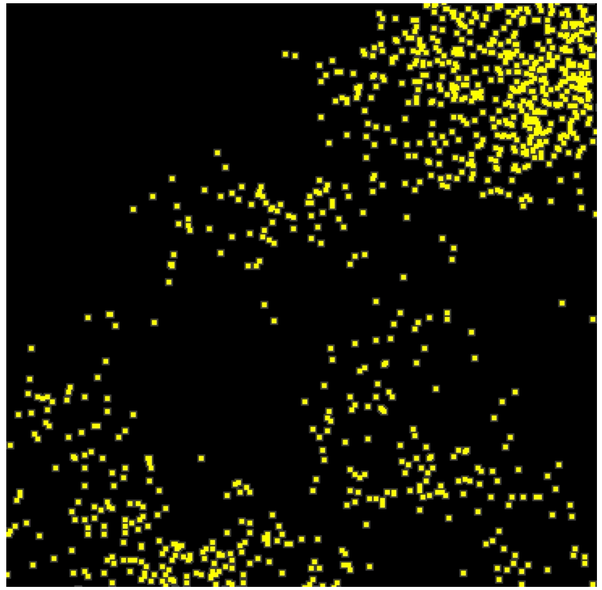


```
<pop>.defineSpatialMap("world", "xy", <matrix>,<...>)
```

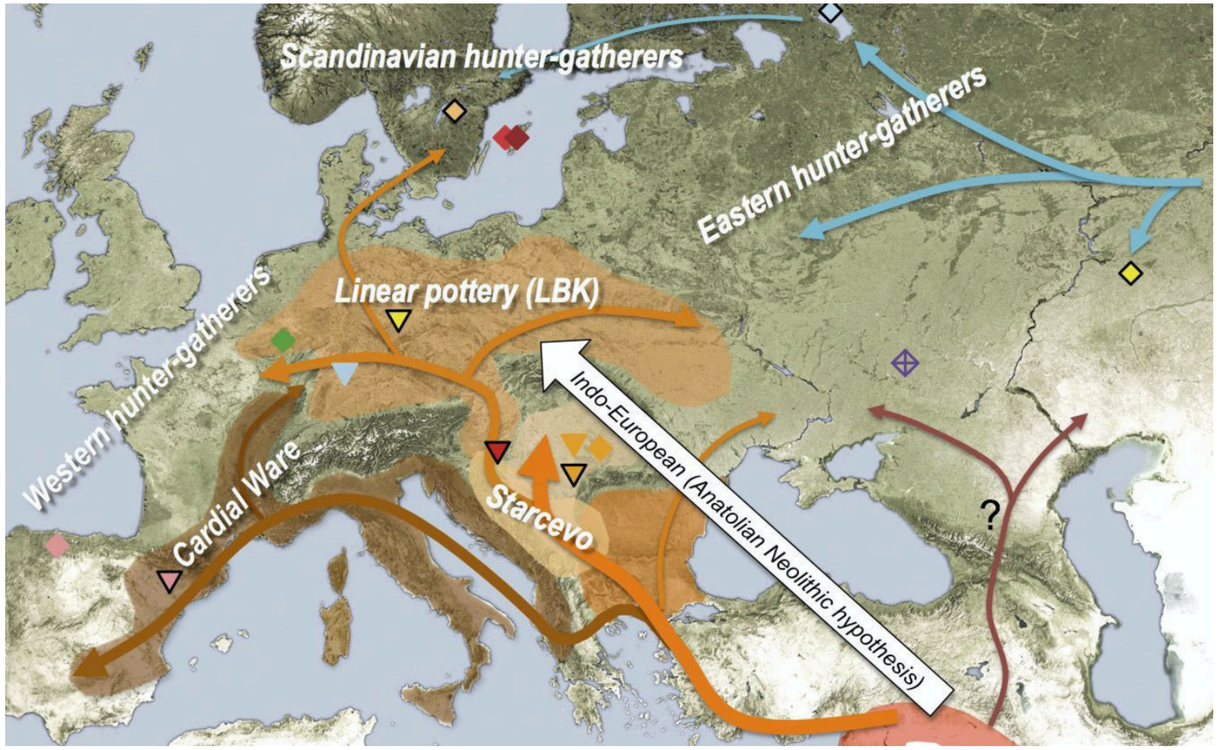
In the background...

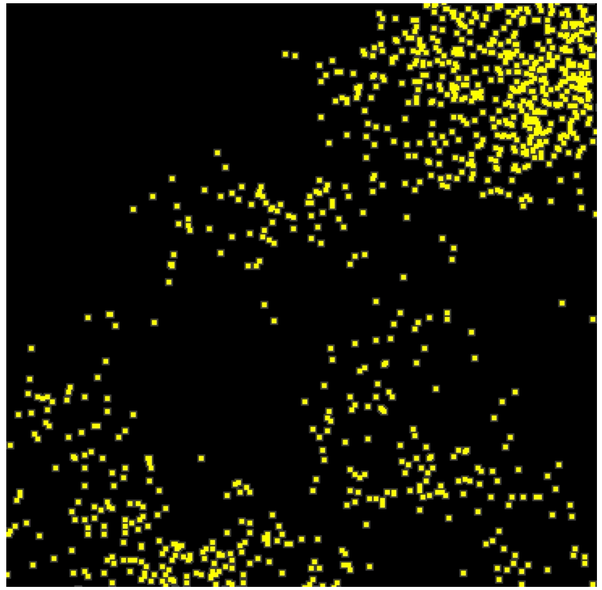
■ pixel→ 🛑 □ pixel→ ✓



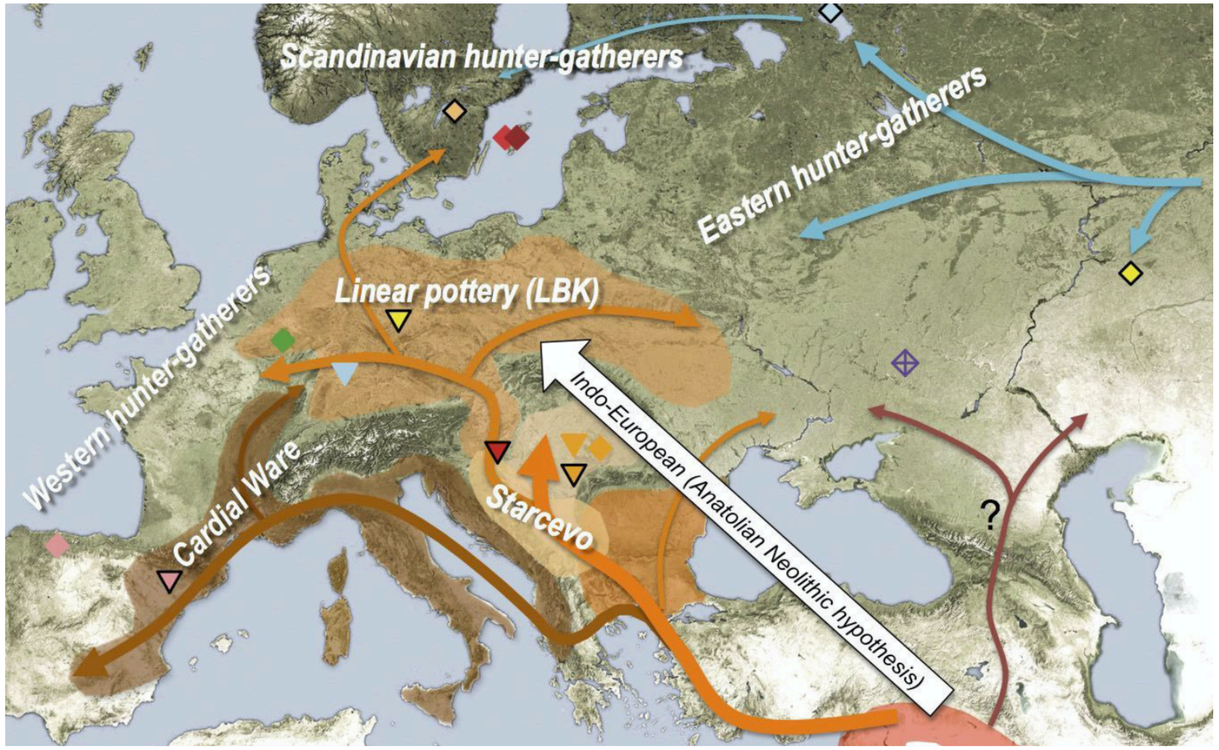


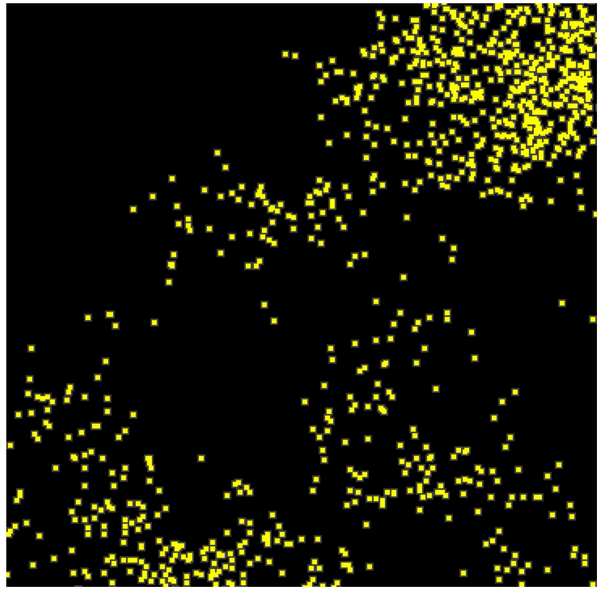
?


Scandinavian hunter-gatherers
Eastern hunter-gatherers
Linear pottery (LBK)
Western hunter-gatherers
Cardial Ware
Starcevo
Indo-European (Anatolian Neolithic hypothesis)
?

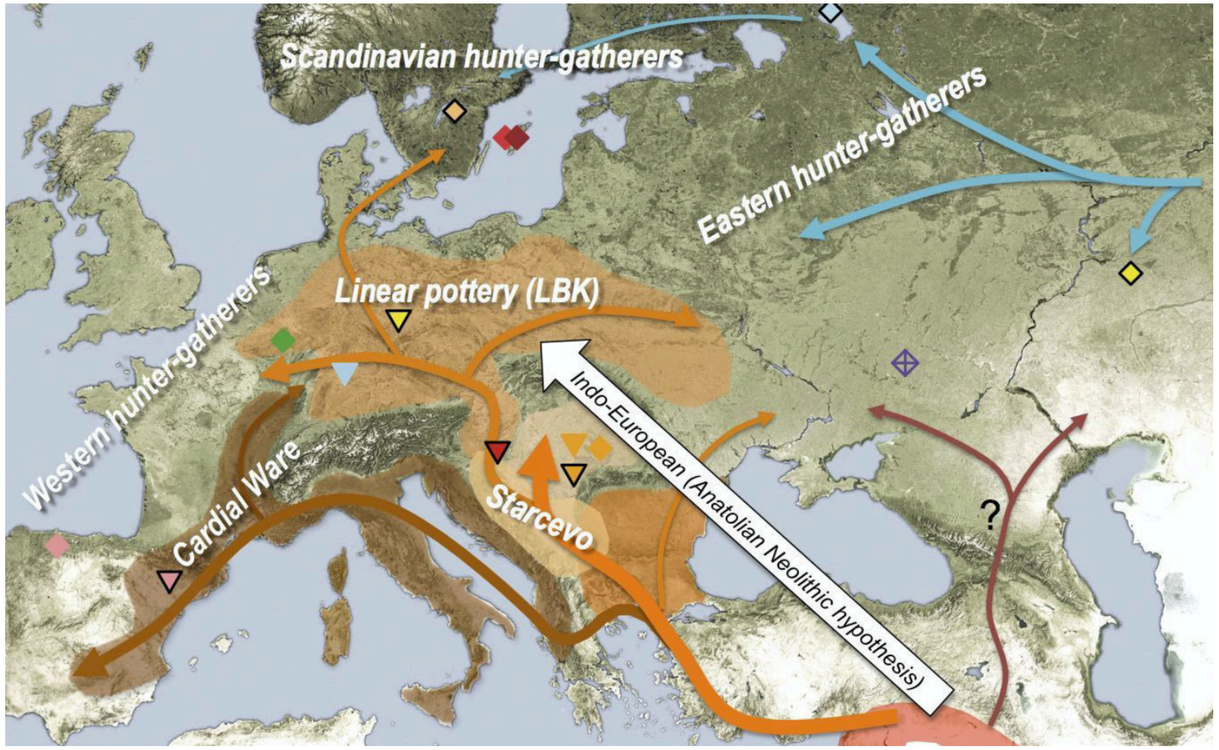


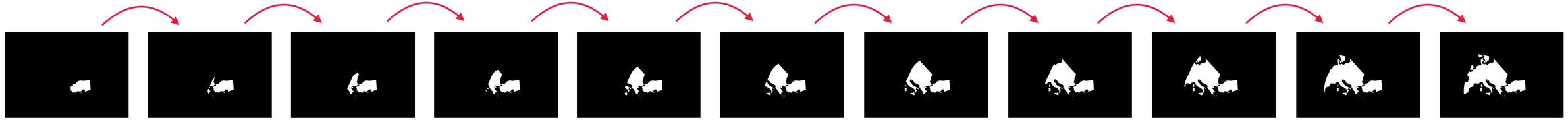
?


Scandinavian hunter-gatherers
Eastern hunter-gatherers
Linear pottery (LBK)
Western hunter-gatherers
Cardial Ware
Starcevo
Indo-European (Anatolian Neolithic hypothesis)
?



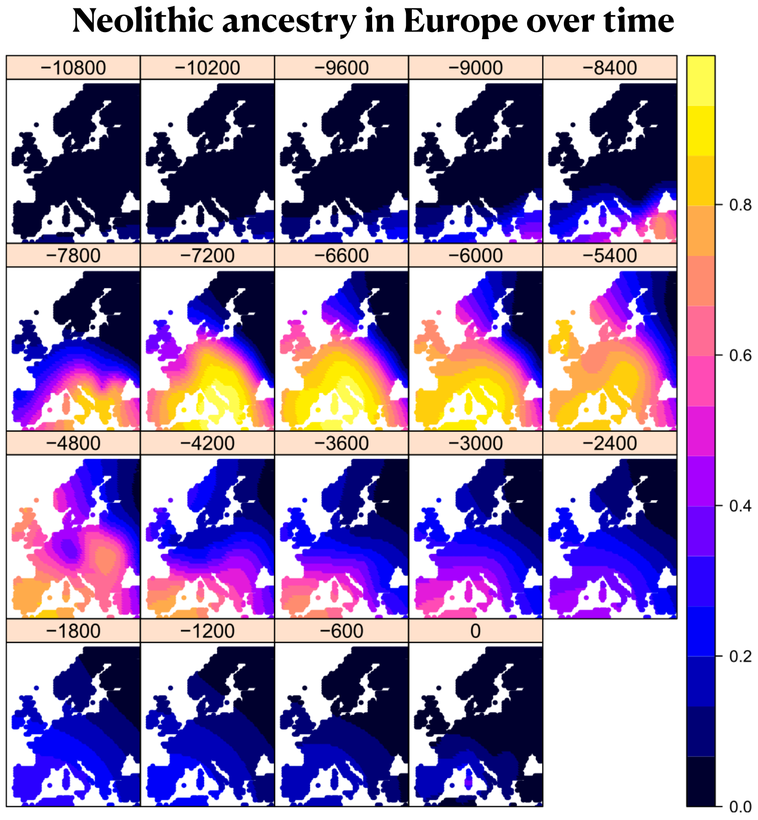
?


Scandinavian hunter-gatherers
Eastern hunter-gatherers
Linear pottery (LBK)
Western hunter-gatherers
Cardial Ware
Starcevo
Indo-European (Anatolian Neolithic hypothesis)
?



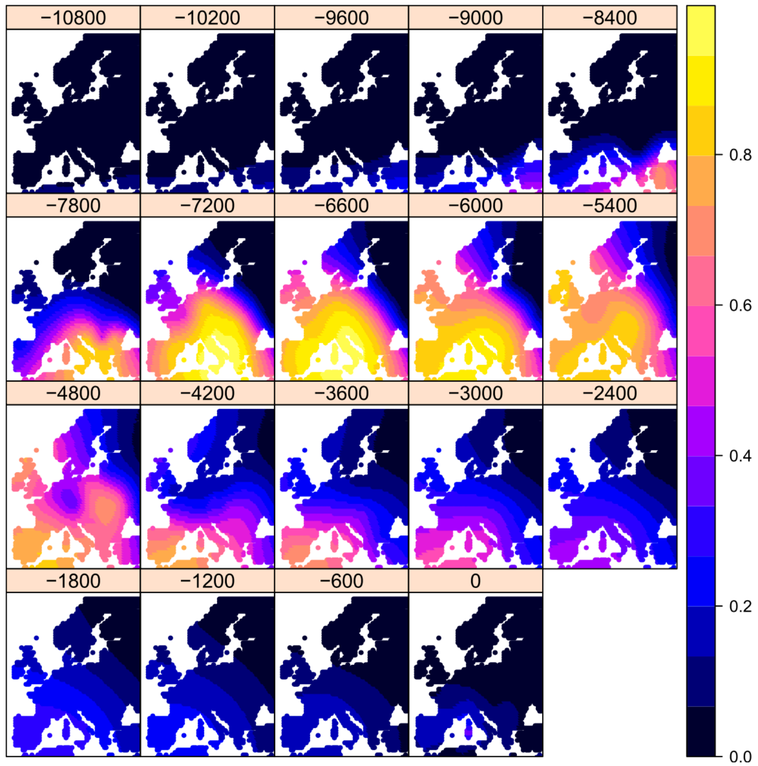
Recent steps towards spatial population genetics

Recent steps towards spatial population genetics

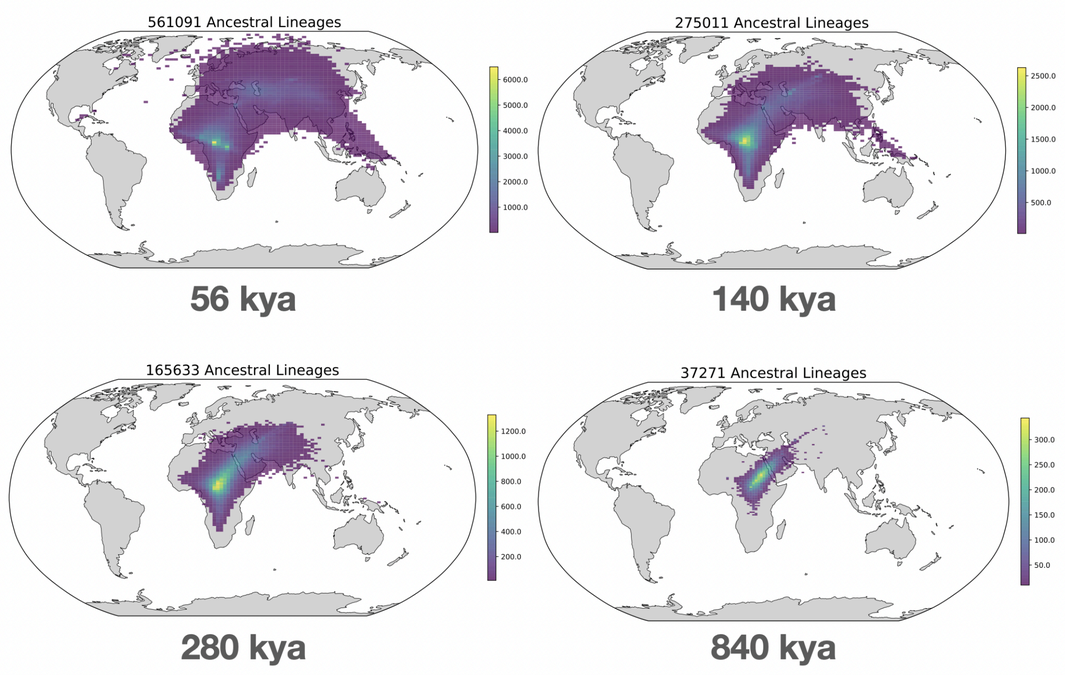


Racimo *et al.* 2020

Recent steps towards spatial population genetics

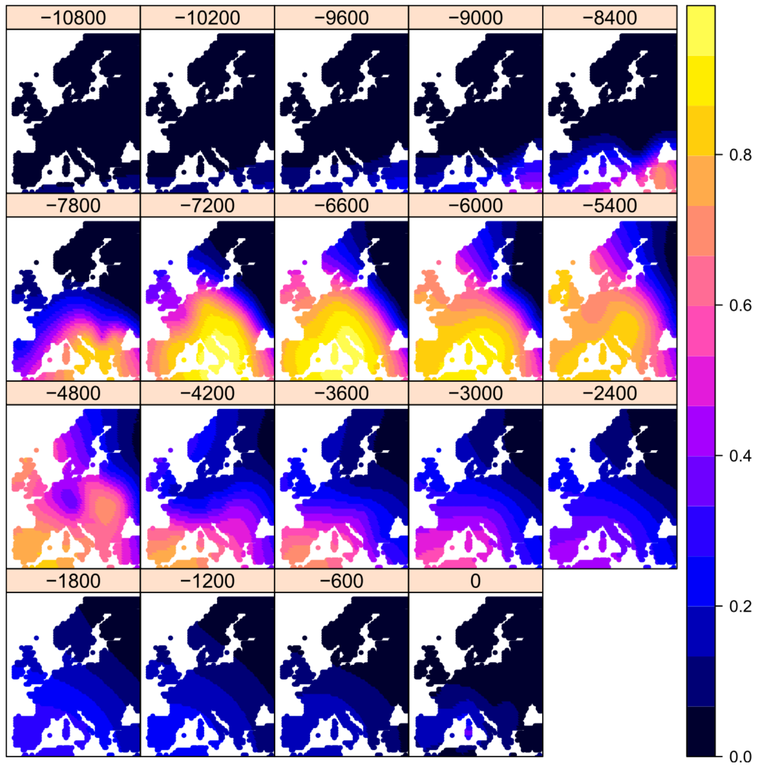


Racimo *et al.* 2020

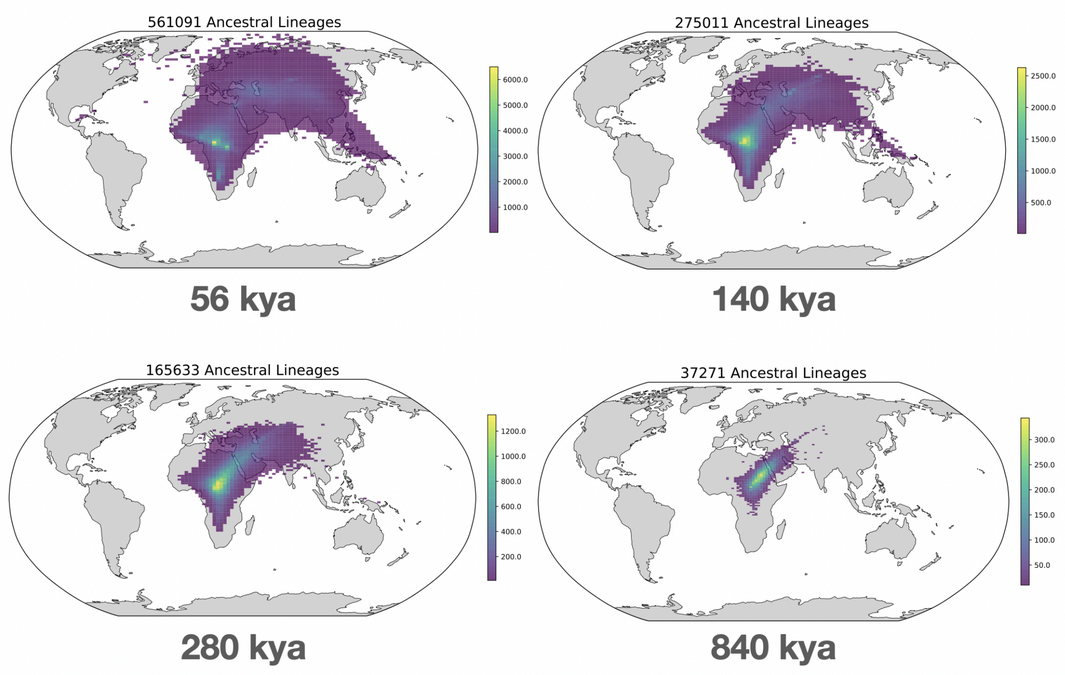


Wohns *et al.* 2022

Recent steps towards spatial population genetics

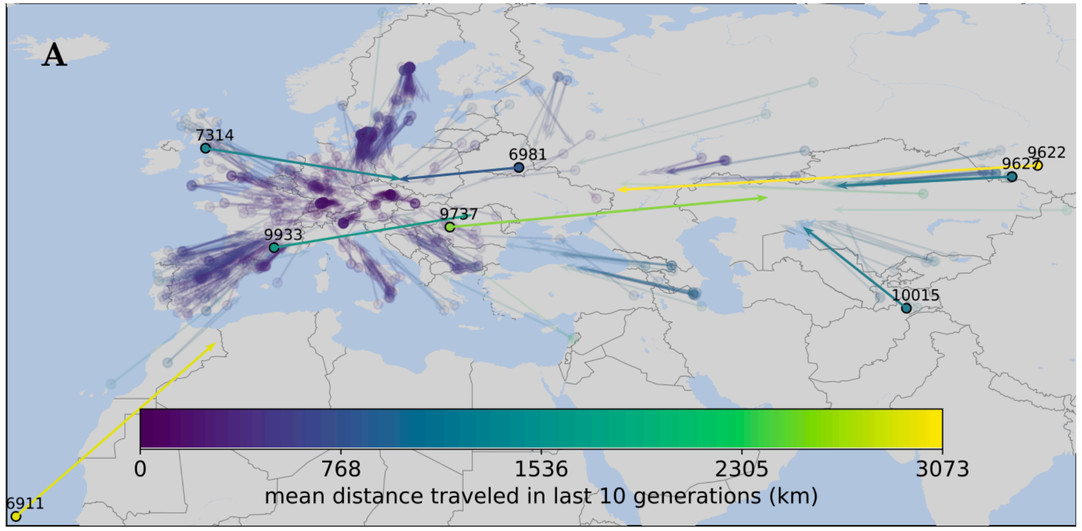


Racimo *et al.* 2020



Wohns *et al.* 2022

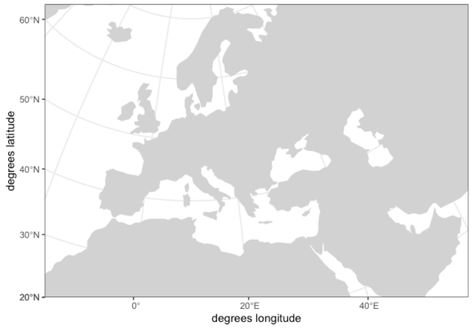
Osmond and Coop 2021



Summary of the slendr simulation workflow

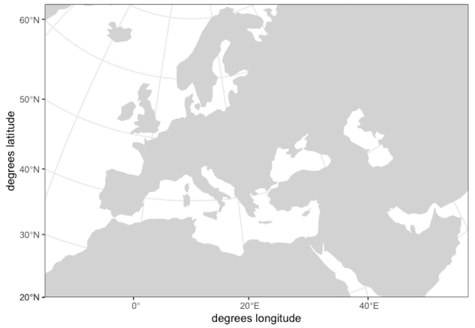
Summary of the slendr simulation workflow

Define the spatial context

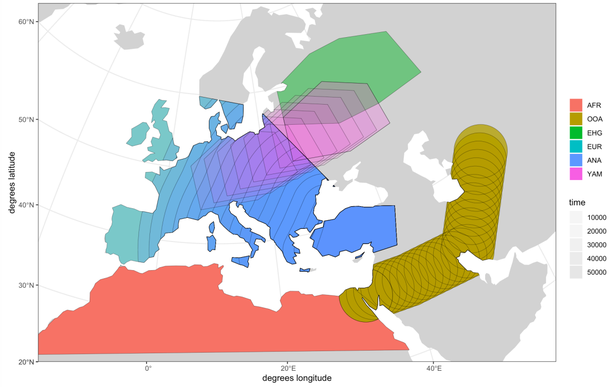


Summary of the slendr simulation workflow

Define the spatial context



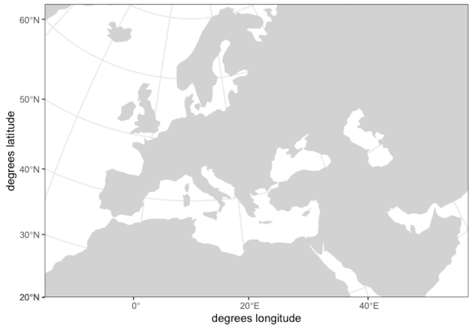
Define population dynamics



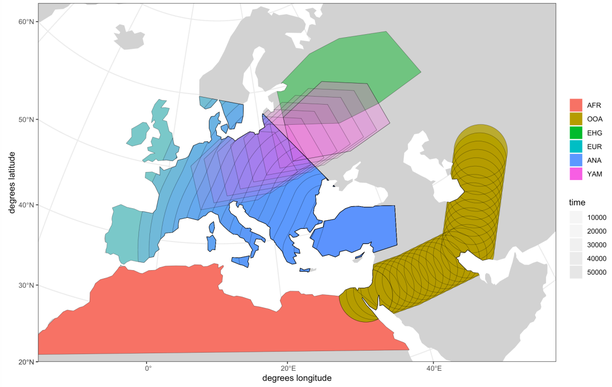
- simulation built interactively from smaller components
- immediate **visual** feedback
- only a couple of lines of simple R code

Summary of the slendr simulation workflow

Define the spatial context



Define population dynamics

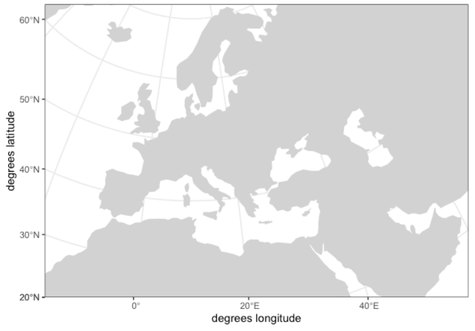


Execute the simulation

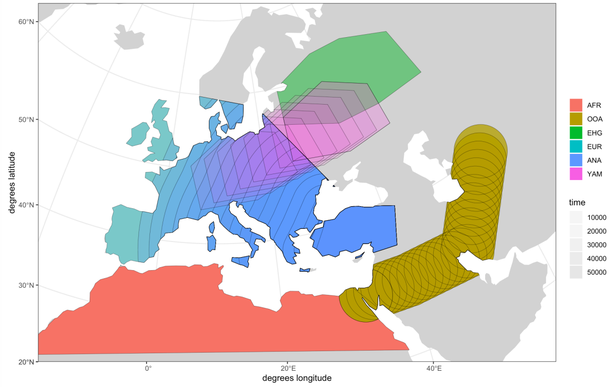
- simulation built interactively from smaller components
- immediate **visual** feedback
- only a couple of lines of simple R code

Summary of the slendr simulation workflow

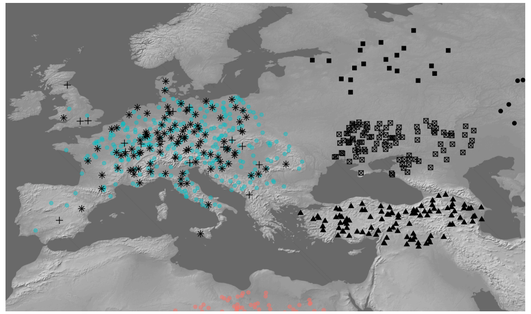
Define the spatial context



Define population dynamics



Execute the simulation



Simulated spatio-temporal data

- simulation built interactively from smaller components
- immediate **visual** feedback
- only a couple of lines of simple R code