pbs_tskit

January 10, 2025

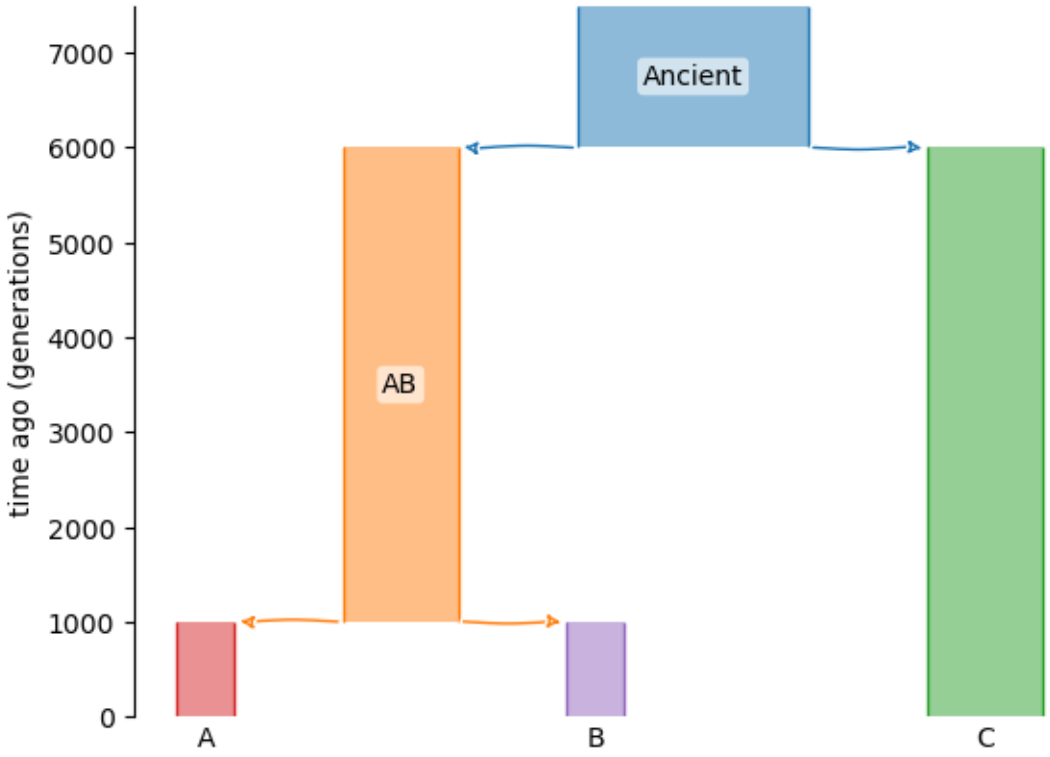
1 PBS in tskit proof of concept

@currocam

[1]:
```
# Import libraries
import msprime, demesdraw
import matplotlib.pyplot as plt
import seaborn as sns
import sgkit as sg
import numpy as np
import xarray as xr
from sgkit.io.vcf import vcf_to_zarr
```

First, we simulate data to estimate the Population Branch Statistic (PBS) from:

[2]:
```
demography = msprime.Demography()
demography.add_population(name="Ancient", initial_size=4_000)
demography.add_population(name="A", initial_size=1_000)
demography.add_population(name="B", initial_size=1_000)
demography.add_population(name="AB", initial_size=2_000)
demography.add_population(name="C", initial_size=2_000)
demography.add_population_split(time=1000, derived=["A", "B"], ancestral="AB")
demography.add_population_split(time=6_000, derived=["AB", "C"],␣
 ↪ancestral="Ancient")
graph = msprime.Demography.to_demes(demography)
fig, ax = plt.subplots()  # use plt.rcParams["figure.figsize"]
demesdraw.tubes(graph, ax=ax, seed=1)
plt.show()
```



We simulate a geneology and overlay mutations:

[3]:
```
ts = msprime.sim_ancestry(
    samples={"A": 25, "B": 25, "C" : 25},
    demography=demography,
    sequence_length=1e8,
    recombination_rate=1e-8,
    random_seed=12
)
ts = msprime.sim_mutations(ts, rate=1e-8, random_seed=12)
ts
```

[3]:
```
<tskit.trees.TreeSequence at 0x7f17c02a6ce0>
```

Then, we compute the PBS with the Fst values calculated from tskit. 1. Compute Fst values per site between populations A, B and C (with indexes 1, 2 and 4 here). 2. Apply Cavalli-Sforza transformation 3. Compute PBS from the formula PBS = 0.5 * (T_AB+T_AC-TBC)

[4]:
```
# Find samples ids
samples_a = [node.id for node in ts.nodes() if node.is_sample() and node.
 ↪population == 1]
samples_b = [node.id for node in ts.nodes() if node.is_sample() and node.
 ↪population == 2]
```

```
samples_c = [node.id for node in ts.nodes() if node.is_sample() and node.
 ↪population == 4]
# Compute matrix of Fst
fsts = ts.Fst(
    [samples_a, samples_b, samples_c],
    indexes=[(0, 1), (0, 2), (1, 2)],
    windows="sites",
    mode='site',
    span_normalise=False
)
fsts.shape == (ts.num_sites, 3)
```

[4]:
```
True
```

[5]:
```
Ts = -np.log(1-fsts)
Ts.shape
```

```
/tmp/ipykernel_36275/2376873481.py:1: RuntimeWarning: divide by zero encountered
in log
  Ts = -np.log(1-fsts)
```

[5]:
```
(94150, 3)
```

[6]:
```
tskit_pbs = 0.5*(Ts[:, 0]+Ts[:, 1] - Ts[:, 2])
tskit_pbs
```

[6]:
```
array([        nan,         nan,         nan, …,         nan,
               nan, -0.04971481])
```

Next, I'll compare these values with the ones we could compute with sgkit (although these are based on the Hudson estimator of Fsts, so I'm not sure this is fair …)

[7]:
```
# Prepare the data
with open("test.vcf", "w") as file:
    ts.write_vcf(file)
```

[8]:
```
%%bash
bcftools view -Oz < test.vcf > test.vcf.gz
bcftools index -t test.vcf.gz
```

[9]:
```
vcf_to_zarr("test.vcf.gz", "test.zarr")
ds = sg.load_dataset("test.zarr")
# Map populations to cohorts
mapping = {1:0, 2:1, 4:2}
cohorts = np.array([mapping[ind.population] for ind in ts.individuals()])
ds["sample_cohort"] = xr.DataArray(cohorts, dims="samples")
cohort_names = [f"co_{i}" for i in range(3)]
```

```
ds = ds.assign_coords({"cohorts_0": cohort_names, "cohorts_1": cohort_names,␣
 ↪"cohorts_2": cohort_names})
sg_pbs = sg.pbs(ds)["stat_pbs"].sel(cohorts_0="co_0", cohorts_1="co_1",␣
 ↪cohorts_2="co_2").values
assert sg_pbs.shape[0] ==  ts.num_sites
sg_pbs
```

[9]:
```
array([        nan,         nan,         nan, …,         nan,
               nan, -0.04641797])
```

Finally, we compare the distribution of PBS values:

[10]:
```
fig, ax = plt.subplots()
g = sns.scatterplot(x=sg_pbs, y=tskit_pbs, ax=ax, alpha=0.8)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, color="black", linestyle="--",␣
 ↪linewidth=1)

ax.set_xlabel("PBS calculated with sgkit")
ax.set_ylabel("PBS calculated with tskit", fontsize=12)
```

[10]:
```
Text(0, 0.5, 'PBS calculated with tskit')
```