# Coding Answers

StatGen Genealogical analysis workshop 1

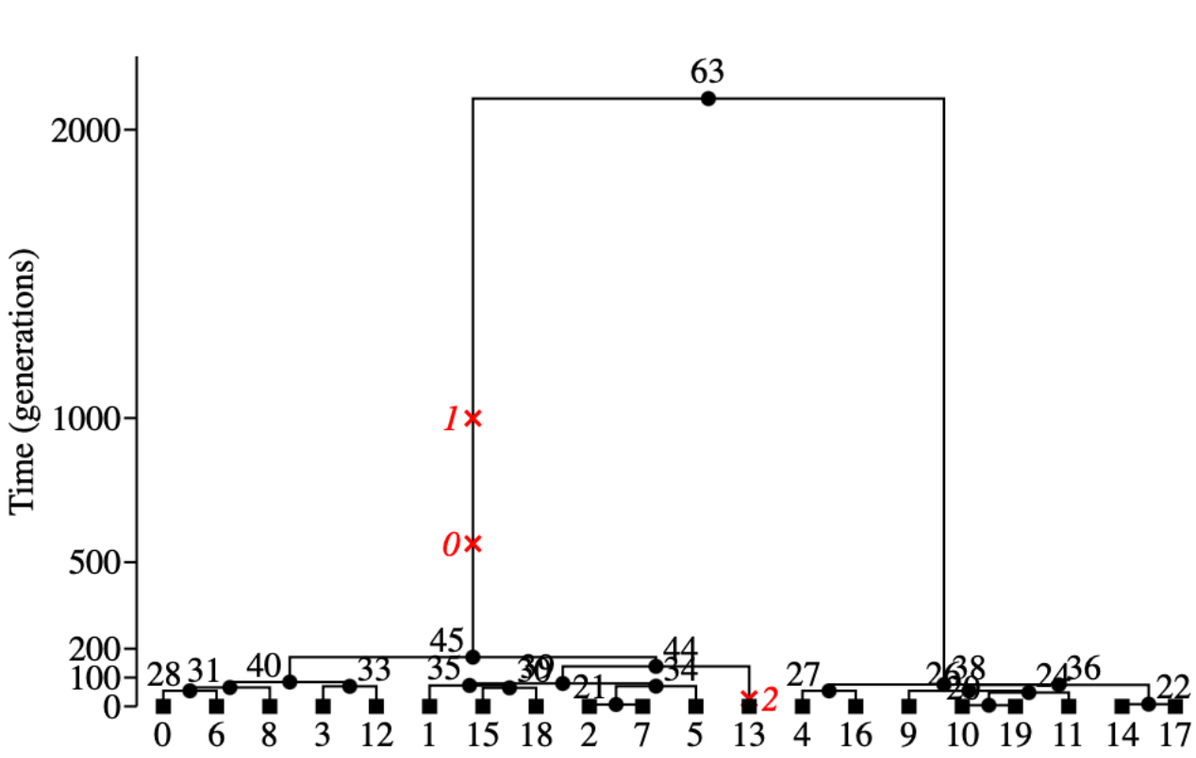
```python
# Exercise 1: modify me
print("Tree sequence has", ts.num_nodes, "nodes, of which",
      ts.num_samples, 'are "sample nodes"')
print("Tree sequence has", ts.num_edges, "edges,",
      ts.num_sites, 'sites, and', ts.num_mutations,
      "mutations")
```

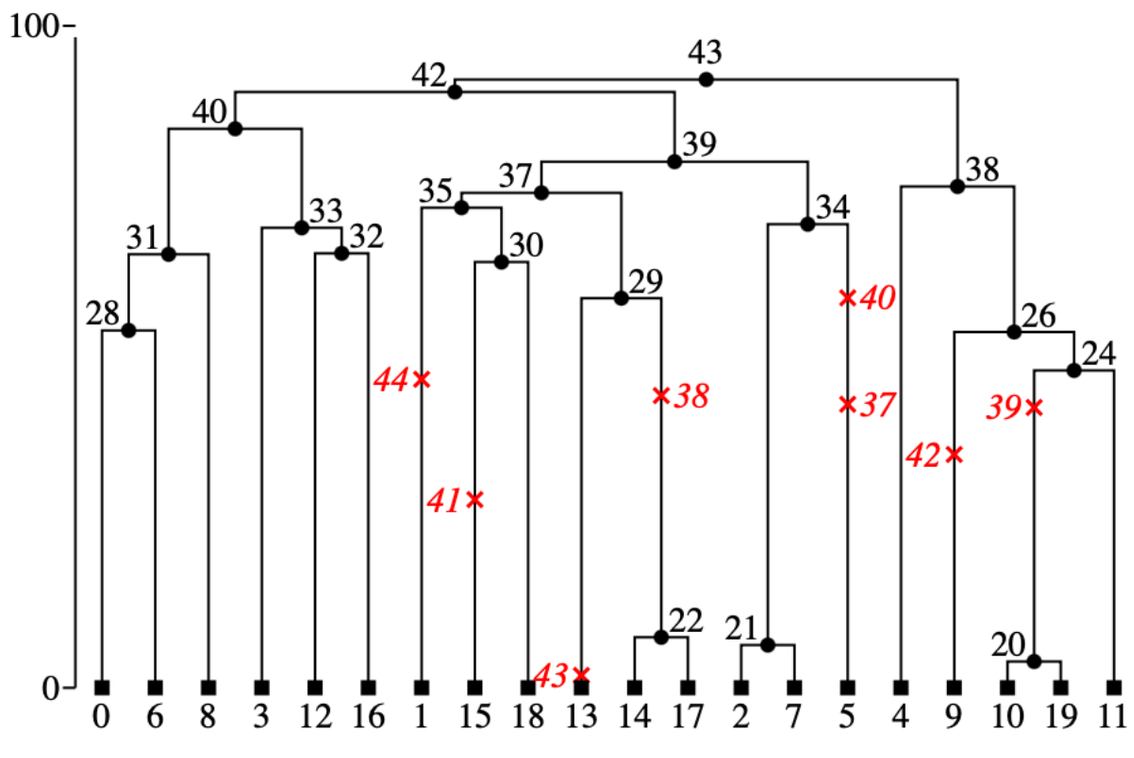
```
Tree sequence has 73 nodes, of which 20 are "sample nodes"
Tree sequence has 150 edges, 99 sites, and 100 mutations
```

```python
# Exercise 2: print out the site object for site 11 and again for site 12
print(ts.site(11))
print(ts.site(12))
```

```
Site(id=11, position=94774.0, ancestral_state='C', mutations=[Mutation(id=11,
site=11, node=38, derived_state='T', parent=-1, metadata=b'', time=3830.17173
6387555)], metadata=b'')
Site(id=12, position=96396.0, ancestral_state='T', mutations=[Mutation(id=12,
site=12, node=51, derived_state='G', parent=-1, metadata=b'', time=4291.70181
3964168), Mutation(id=13, site=12, node=51, derived_state='T', parent=12, met
adata=b'', time=3973.698606992847)], metadata=b'')
```

```
# Exercise 3: use draw_svg() to create two SVG trees (the first in the tree sequence and t
# position 400Kb), and plot them by calling display(SVG(...)) twice. They should look quit
svg_text = ts.first().draw_svg(
    size=(500, 300),
    y_axis=True,
    y_ticks=[0, 100, 200, 500, 1000, 2000],
)
display(SVG(svg_text))
svg_text = ts.at(400_000).draw_svg(
    size=(500, 300),
    y_axis=True,
    y_ticks=[0, 100, 200, 500, 1000, 2000],
)
display(SVG(svg_text))
```



```
# Exercise 4: print out the age (in generations ago) of the oldest node in the first tree
ts.node(63)
```

```
Node(id=63, flags=0, time=2108.0062486338006, population=0, individual=-1, metadata=b'')
```

```
# Exercise 5a: Display the individuals table
ts.tables.individuals
```

| id | flags | location | parents | metadata |
|---|---|---|---|---|
| 0 | 0 | | | {'name': 'Ada'} |
| 1 | 0 | | | {'name': 'Bob'} |
| 2 | 0 | | | {'name': 'Cat'} |
| 3 | 0 | | | {'name': 'Dee'} |
| 4 | 0 | | | {'name': 'Eli'} |
| 5 | 0 | | | {'name': 'Fi'} |
| 6 | 0 | | | {'name': 'Guy'} |
| 7 | 0 | | | {'name': 'Hal'} |
| 8 | 0 | | | {'name': 'Ida'} |
| 9 | 0 | | | {'name': 'Jo'} |

```
# Exercise 5b: Display the populations table
ts.tables.populations
```

| id | metadata |
|---|---|
| 0 | {'description': '', 'name': 'pop_0'} |

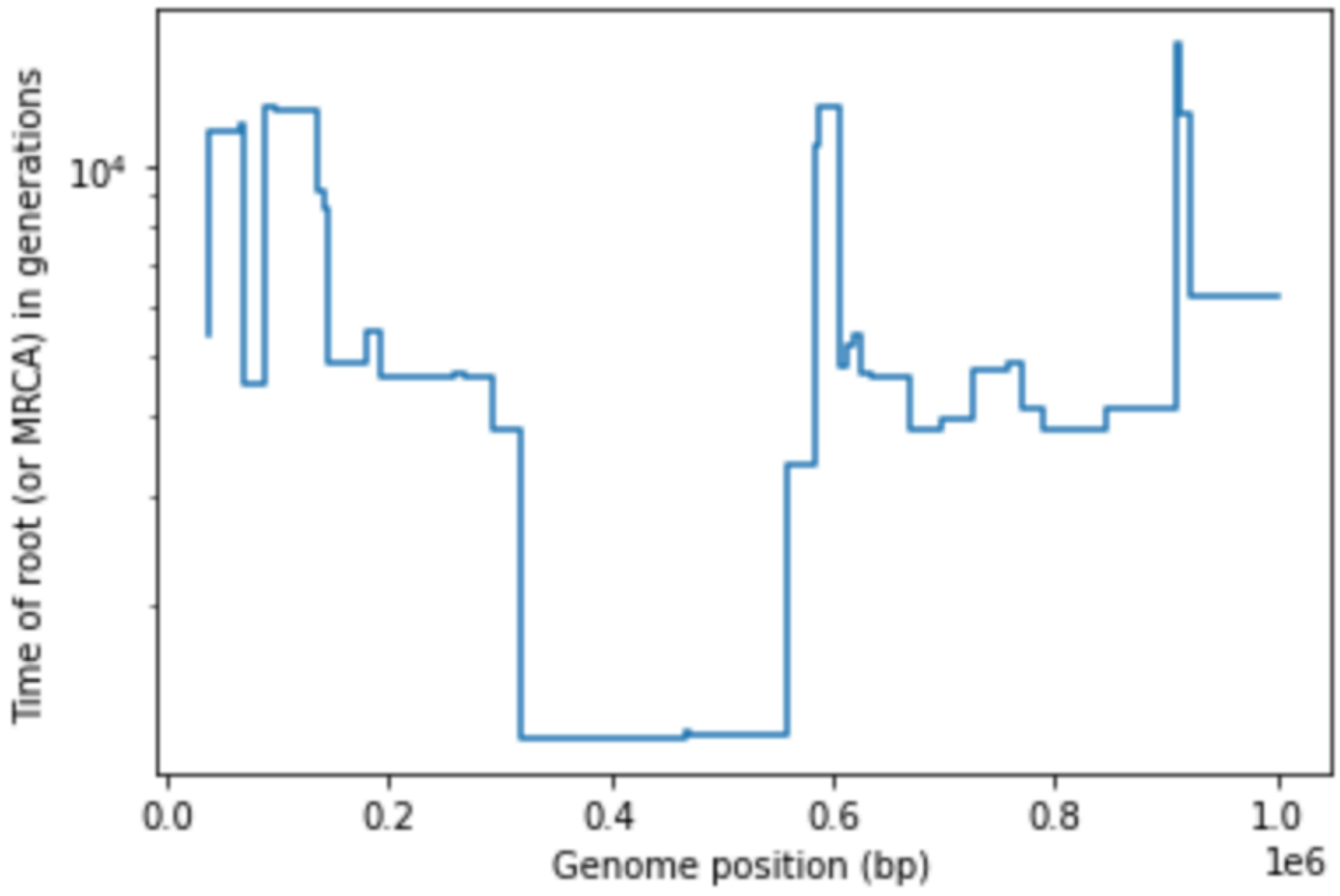
```python
# Exercise 6: iterate over all 99 variants in `ts`, printing out the genotypes at each site
# For brevity, illustrate using the simplified_ts which has only 5 samples
for v in ts.variants():
    print(
        v.genotypes, # the decoded variation data
    )
```

```
[1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0]
[1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0]
[0 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1]
[1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0]
```

■ ■ ■

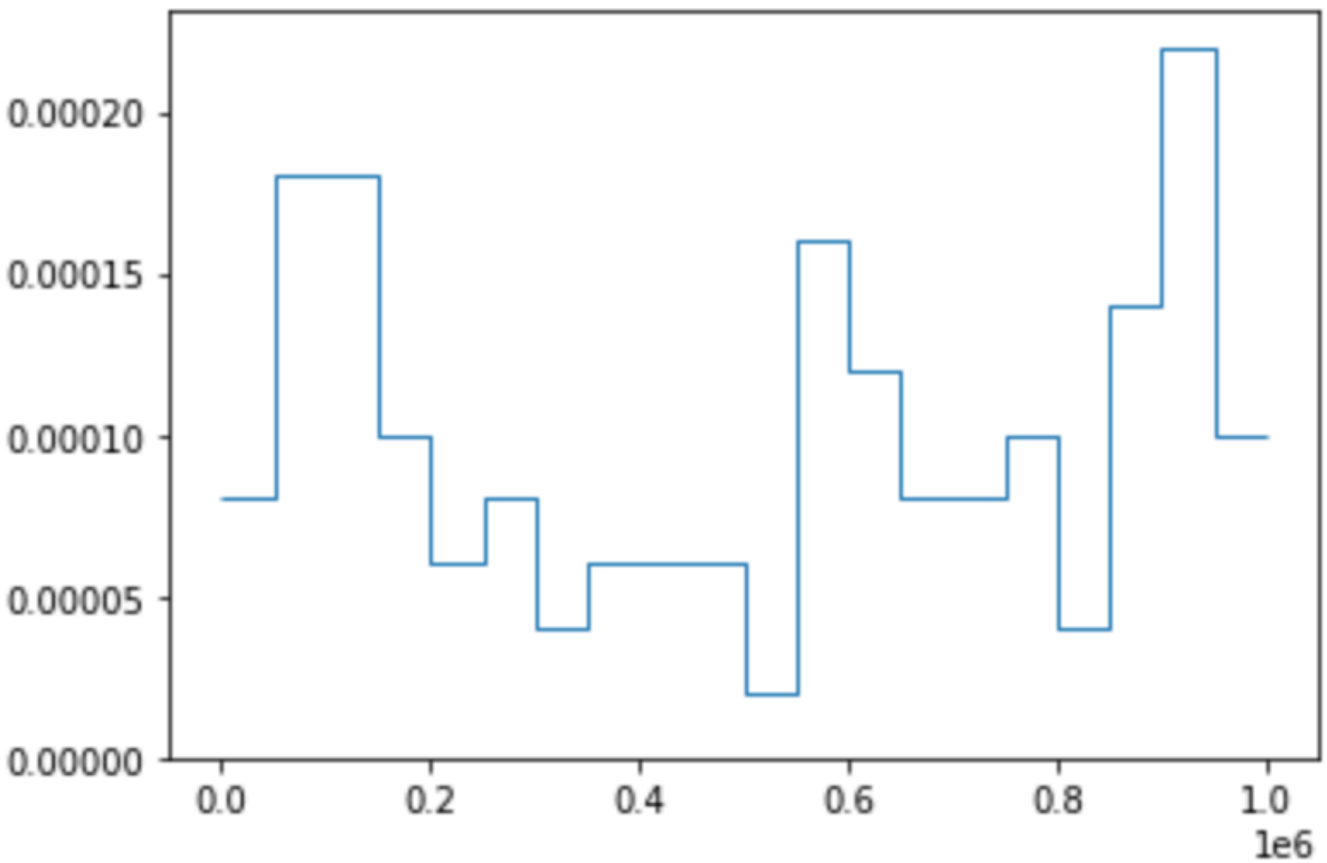
```
[1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1]
[0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1]
[1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1]
[0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
```

```python
# Exercise 7: plot the total branch length of the trees along the genome
import matplotlib.pyplot as plt

x = []
y = []
for tree in ts.trees():
    x.append(tree.interval.right)
    y.append(tree.total_branch_length)
plt.step(x, y)
plt.xlabel("Genome position (bp)")
plt.ylabel(f"Time of root (or MRCA) in {ts.time_units}")
plt.yscale("log")
```



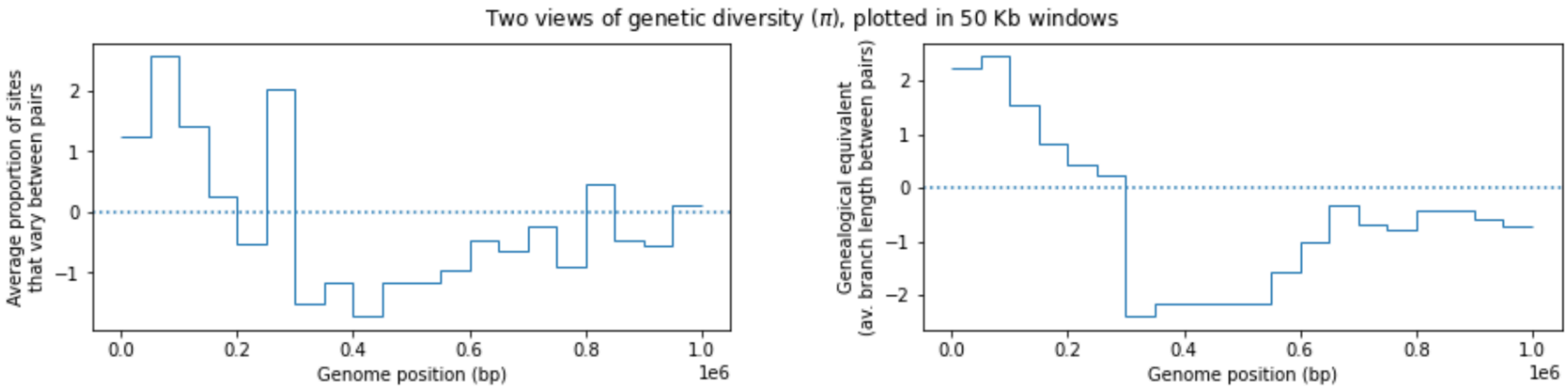
```
# Exercise 8: plot the density of variable sites in fixed-sized windows along the genome
plt.stairs(site_density, window_locations, baseline=None)
```

```
<matplotlib.patches.StepPatch at 0x7f2a1d97d0c0>
```



```
import msprime
mutation_rate = 2e-7  # Higher than the human average of ~1e-8
high_mutation_ts = msprime.sim_mutations(ts, rate=mutation_rate, random_seed=123)

# Exercise 9: illustrate that with high mutation rate, the density of variable sites tends
# to the average branch length statistic. Do this by using the same plotting code as in the
# previous code cell, but substituting `high_mutation_ts` for `ts`.
# Use the branch length as the statistic
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(15, 3))
plt.subplots_adjust(wspace=0.3)
site_density = high_mutation_ts.segregating_sites(windows=window_locations)
branch_length = high_mutation_ts.segregating_sites(windows=window_locations, mode="branch")

# Same as the previous plot
ax1.stairs(site_density, window_locations, baseline=None)
ax1.set_xlabel("Genome position (bp)")
ax1.set_ylabel(f"Proportion of variable sites")

# Genealogical equivalent (mode="branch")
ax2.stairs(branch_length, window_locations, baseline=None)
ax2.set_xlabel("Genome position (bp)")
ax2.set_ylabel(f"Genealogical equivalent\n(average branch length)")

plt.suptitle(
    "Two views of the segregating_sites statistic (low mutation rate)"
    f", plotted in {step/1000:.0f} Kb windows")

plt.show()
```

```python
# Exercise 10: plot Tajima's D in windows along the genome
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(15, 3))
plt.subplots_adjust(wspace=0.3)

window_locations, step = np.linspace(0, ts.sequence_length, num=21, retstep=True)
diversity_site = ts.Tajimas_D(windows=window_locations)
diversity_branch = ts.Tajimas_D(windows=window_locations, mode="branch")

ax1.stairs(diversity_site, window_locations, baseline=None)
ax1.set_xlabel("Genome position (bp)")
ax1.set_ylabel("Average proportion of sites\nthat vary between pairs")
ax1.axhline(0, linestyle=":")

# Genealogical equivalent (mode="branch")
ax2.stairs(diversity_branch, window_locations, baseline=None)
ax2.set_xlabel("Genome position (bp)")
ax2.set_ylabel("Genealogical equivalent\n(av. branch length between pairs)")
ax2.axhline(0, linestyle=":")

plt.suptitle(r"Two views of genetic diversity ($\pi$), " + f"plotted in {step/1000:.0f} Kb windows")

plt.show()
```



# StatGen Genealogical analysis workshop 2

```python
# Exercise 1: Set `ts` to a new large tree sequence, generated using msprime.sim_a
# specific parameters (random_seed=2022, etc.), then output the tree sequence summ
ts = msprime.sim_ancestry(
    samples=20_000,
    sequence_length=1_000_000,
    recombination_rate=1e-8, # Small recombination rate
    population_size=20_000, # Rough "effective population size" suitable for human
    random_seed=2022
)
ts
```

tskit **Tree Sequence**

| | |
|---|---|
| Trees | 8615 |
| Sequence Length | 1000000.0 |
| Time Units | generations |
| Sample Nodes | 40000 |
| Total Size | 7.2 MiB |
| Metadata | No Metadata |

| Table | Rows | Size | Has Metadata |
|---|---|---|---|
| Edges | 112914 | 3.4 MiB | |
| Individuals | 20000 | 546.9 KiB | |
| Migrations | 0 | 8 Bytes | |
| Mutations | 0 | 16 Bytes | |
| Nodes | 86673 | 2.3 MiB | |
| Populations | 1 | 224 Bytes | ✅ |
| Provenances | 1 | 954 Bytes | |
| Sites | 0 | 16 Bytes | |

```
# Exercise 2: Print out the mutations table. Notice that one site has
mts_small.tables.mutations
```

| id | site | node | time | derived_state | parent | metadata |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 6280.57457692 | T | -1 | b'' |
| 1 | 1 | 1 | 300.03900794 | C | -1 | b'' |
| 2 | 2 | 5 | 17601.10716936 | A | -1 | b'' |
| 3 | 3 | 5 | 31041.47149974 | G | -1 | b'' |
| 4 | 3 | 1 | 25271.85405329 | C | -1 | b'' |
| 5 | 4 | 4 | 9037.19890652 | A | -1 | b'' |
| 6 | 5 | 1 | 19218.67456043 | T | -1 | b'' |
| 7 | 6 | 1 | 10675.94608952 | A | -1 | b'' |
| 8 | 7 | 5 | 20843.39959853 | C | -1 | b'' |
| 9 | 8 | 1 | 13428.65262360 | C | -1 | b'' |
| 10 | 9 | 5 | 36455.12186042 | C | -1 | b'' |

```
# Exercise 3: Add lots of mutations to the huge genealogy you simulated earlier.
msprime.sim_mutations(ts, rate=1e-8, random_seed=2022)
```

tskit **Tree Sequence**

| | |
|---|---|
| Trees | 8615 |
| Sequence Length | 1000000.0 |
| Time Units | generations |
| Sample Nodes | 40000 |
| Total Size | 7.7 MiB |
| Metadata | No Metadata |

| Table | Rows | Size | Has Metadata |
|---|---|---|---|
| Edges | 112914 | 3.4 MiB | |
| Individuals | 20000 | 546.9 KiB | |
| Migrations | 0 | 8 Bytes | |
| Mutations | 9126 | 329.8 KiB | |
| Nodes | 86673 | 2.3 MiB | |
| Populations | 1 | 224 Bytes | ✅ |
| Provenances | 2 | 1.6 KiB | |
| Sites | 9087 | 221.9 KiB | |

```python
# Exercise 4: Loop over the populations, printing out Fst between each and pop_0.
for i in range(1, 8):
    print(mts.Fst([pop_0_sample_ids, mts.samples(population=i)]))
```

```
-0.0004229567769440 39
0.0042090861564245685
0.00530775429597 8628
0.00513671546336 7136
0.0038666037155368205
0.0052154743130 21495
0.0018151588832213683
```

```
# Exercise 5: plot a histogram of Fst values, and also print out the mean.
plt.hist(Fst_vals)
```

```
(array([ 1.,  8.,  9., 20., 17., 13., 12.,  8.,  6.,  6.]),
 array([-0.00431861, -0.00303536, -0.00175211, -0.00046887,  0.00081438,
         0.00209763,  0.00338088,  0.00466413,  0.00594737,  0.00723062,
         0.00851387]),
 <BarContainer object of 10 artists>)
```