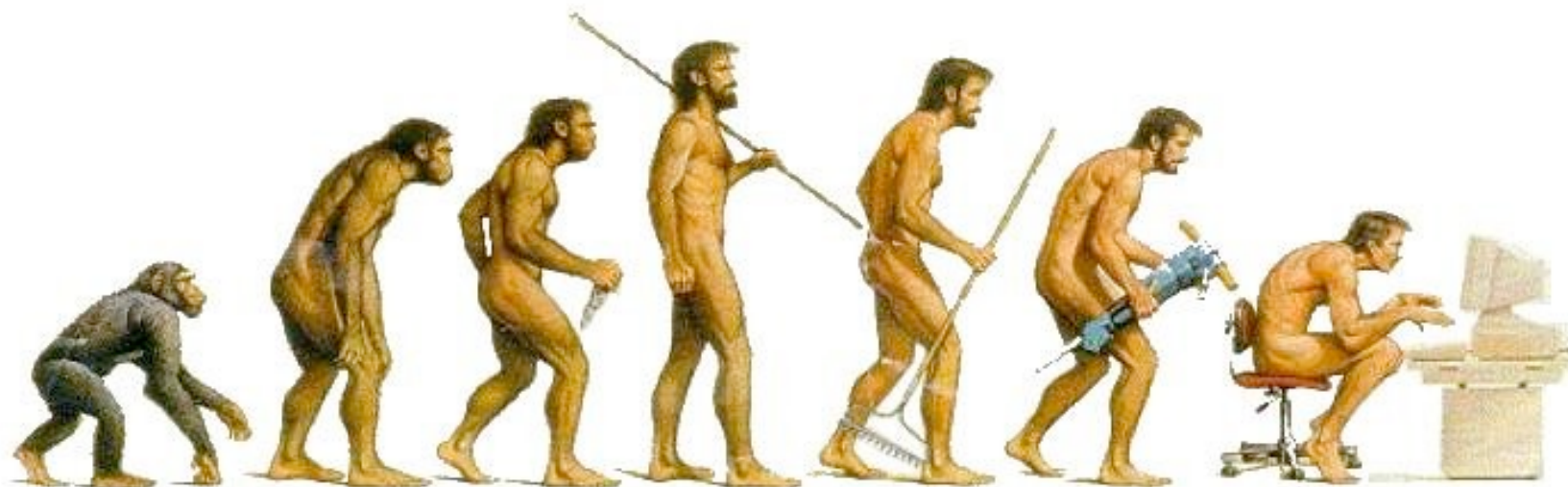


# Summer Institute in Statistical Genetics

## Module 07: Applications in Population Genetics

**Instructors:** Ryan Hernandez & Timothy O'Connor

**TAs:** Nobu Masaki and Ruoyi Cai



# Let's Simulate!!

---

- Simulations are one of the best ways to learn!
- There are many kinds of simulators, and many implementations of simulators.
- SLiM is one of many, and one of the most popular in active population genetics research.

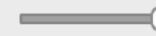
# Installing SLiMgui

---

- Download and install SLiM from <https://messerlab.org/slim/>. Note that you may need to consult Chapter 2 of the **manual** if you are not using a Mac. This installation will include SLiMgui.
- To run, open the SLiMgui application!
  - On Mac: /Applications/SLiMgui.app

ID

N



Tick:

initialize()

Cycle:

initialize()



Input Commands:



Run Output:



```
1 // set up a simple neutral simulation
2 initialize() {
3   initializeMutationRate(1e-7);
4
5   // m1 mutation type: neutral
6   initializeMutationType("m1", 0.5, "f", 0.0);
7
8   // g1 genomic element type: uses m1 for all mutations
9   initializeGenomicElementType("g1", m1, 1.0);
10
11  // uniform chromosome of length 100 kb with uniform recombination
12  initializeGenomicElement(g1, 0, 99999);
13  initializeRecombinationRate(1e-8);
14 }
15
16 // create a population of 500 individuals
17 1 early() {
18   sim.addSubpop("p1", 500);
19 }
20
21 // output samples of 10 genomes periodically, all fixed mutations at end
22 1000 late() { p1.outputSample(10); }
```

```
// Initial random seed:
1771802506970
```

# Drift

---


# Drift

---

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation


# Drift

---

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation
- <click the  button!>

# Drift



---

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation
- <click the  button!>
- Now plots!



# Drift

---

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation
- <click the  button!>
- Now plots!
  - <click the  button!>

# Drift

---

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation

• <click the  button!>

- Now plots!

• <click the  button!>



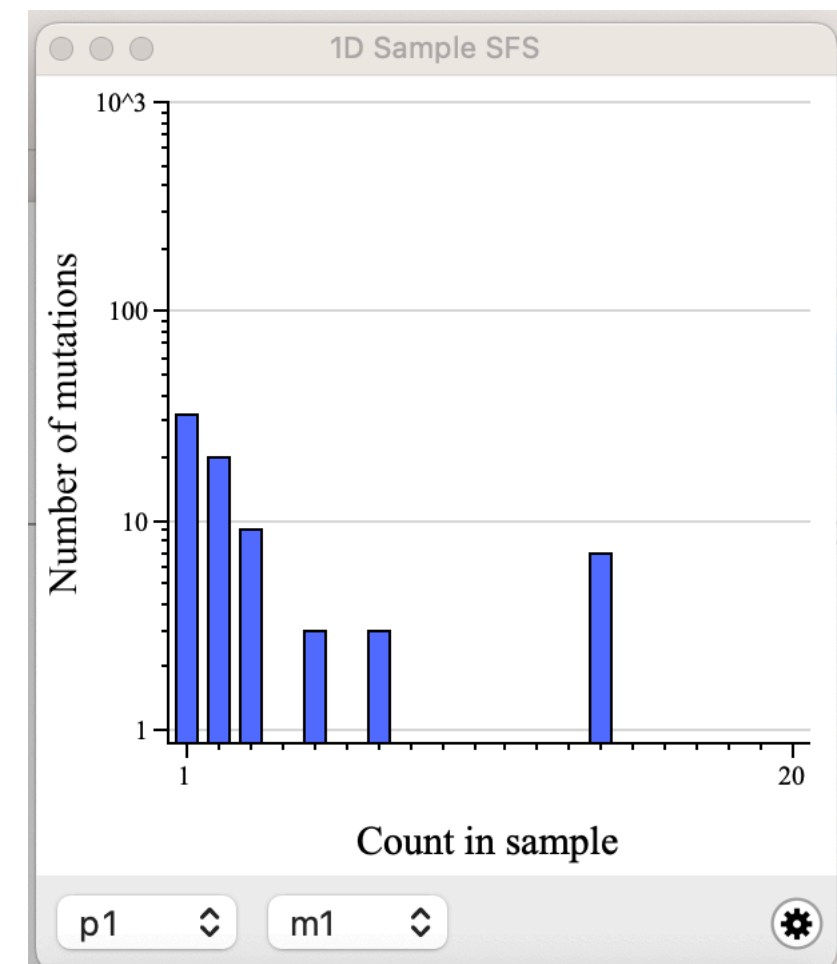
# Drift

- Navigate to:
  - > File
  - > Open Recipe
  - > 4 – Getting Started: Neutral evolution in a panmictic population
  - > 4.1 – A basic neutral simulation

• <click the  button!>

• Now plots!

• <click the  button!>



// Keywords:

// set up a simple neutral simulation

initialize()

{

    // set the overall mutation rate

    initializeMutationRate(1e-7);

    // m1 mutation type: neutral

    initializeMutationType("m1", 0.5, "f", 0.0);

    // g1 genomic element type: uses m1 for all mutations

    initializeGenomicElementType("g1", m1, 1.0);

    // uniform chromosome of length 100 kb

    initializeGenomicElement(g1, 0, 99999);

    // uniform recombination along the chromosome

    initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

    sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

    sim.simulationFinished();

}

// Keywords:

// set up a simple neutral simulation  
initialize()

{

// set the overall mutation rate  
initializeMutationRate(1e-7);

// m1 mutation type: neutral  
initializeMutationType("m1", 0.5, "f", 0.0);

// g1 genomic element type: uses m1 for all mutations  
initializeGenomicElementType("g1", m1, 1.0);

// uniform chromosome of length 100 kb  
initializeGenomicElement(g1, 0, 99999);

// uniform recombination along the chromosome  
initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

sim.simulationFinished();

}

Set mutation rate!

// Keywords:

// set up a simple neutral simulation  
initialize()

{

// set the overall mutation rate  
initializeMutationRate(1e-7);

// m1 mutation type: neutral  
initializeMutationType("m1", 0.5, "f", 0.0);

// g1 genomic element type: uses m1 for all mutations  
initializeGenomicElementType("g1", m1, 1.0);

// uniform chromosome of length 100 kb  
initializeGenomicElement(g1, 0, 99999);

// uniform recombination along the chromosome  
initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

sim.simulationFinished();

}

Set mutation rate!

Set recombination rate!

// Keywords:

// set up a simple neutral simulation  
initialize()

{

// set the overall mutation rate  
initializeMutationRate(1e-7);

// m1 mutation type: neutral  
initializeMutationType("m1", 0.5, "f", 0.0);

// g1 genomic element type: uses m1 for all mutations  
initializeGenomicElementType("g1", m1, 1.0);

// uniform chromosome of length 100 kb  
initializeGenomicElement(g1, 0, 99999);

// uniform recombination along the chromosome  
initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

sim.simulationFinished();

}

Set mutation rate!

Set genomic elements

Set recombination rate!

// Keywords:

// set up a simple neutral simulation  
initialize()

{

// set the overall mutation rate  
initializeMutationRate(1e-7);

// m1 mutation type: neutral  
initializeMutationType("m1", 0.5, "f", 0.0);

// g1 genomic element type: uses m1 for all mutations  
initializeGenomicElementType("g1", m1, 1.0);

// uniform chromosome of length 100 kb  
initializeGenomicElement(g1, 0, 99999);

// uniform recombination along the chromosome  
initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

sim.simulationFinished();

}

Set mutation rate!

Set mutation types

Set genomic elements

Set recombination rate!



// Keywords:

```
// set up a simple neutral simulation
initialize()
```

```
{
```

```
    // set the overall mutation rate
    initializeMutationRate(1e-7);
```

```
    // m1 mutation type: neutral
    initializeMutationType("m1", 0.5, "f", 0.0);
```

```
    // g1 genomic element type: uses m1 for all mutations
    initializeGenomicElementType("g1", m1, 1.0);
```

```
    // uniform chromosome of length 100 kb
    initializeGenomicElement(g1, 0, 99999);
```

```
    // uniform recombination along the chromosome
    initializeRecombinationRate(1e-8);
```

```
}
```

```
// create a population of 500 individuals
```

```
1 early()
```

```
{
```

```
    sim.addSubpop("p1", 500);
```

```
}
```

```
// run to tick 10000
```

```
10000 early()
```

```
{
```

```
    sim.simulationFinished();
```

```
}
```

Set mutation rate!

Set mutation types

Set genomic elements

Set sequence length!

Set recombination rate!

// Keywords:

// set up a simple neutral simulation  
initialize()

{

// set the overall mutation rate  
initializeMutationRate(1e-7);

// m1 mutation type: neutral  
initializeMutationType("m1", 0.5, "f", 0.0);

// g1 genomic element type: uses m1 for all mutations  
initializeGenomicElementType("g1", m1, 1.0);

// uniform chromosome of length 100 kb  
initializeGenomicElement(g1, 0, 99999);

// uniform recombination along the chromosome  
initializeRecombinationRate(1e-8);

}

// create a population of 500 individuals

1 early()

{

sim.addSubpop("p1", 500);

}

// run to tick 10000

10000 early()

{

sim.simulationFinished();

}

Set mutation rate!

Set mutation types

Set genomic elements

Set sequence length!

Set recombination rate!

Set population size

// Keywords:

```
// set up a simple neutral simulation  
initialize()  
{
```

```
    // set the overall mutation rate  
    initializeMutationRate(1e-7);
```

```
    // m1 mutation type: neutral  
    initializeMutationType("m1", 0.5, "f", 0.0);
```

```
    // g1 genomic element type: uses m1 for all mutations  
    initializeGenomicElementType("g1", m1, 1.0);
```

```
    // uniform chromosome of length 100 kb  
    initializeGenomicElement(g1, 0, 99999);
```

```
    // uniform recombination along the chromosome  
    initializeRecombinationRate(1e-8);
```

```
}
```

```
// create a population of 500 individuals
```

```
1 early()  
{
```

```
    sim.addSubpop("p1", 500);
```

```
}
```

```
// run to tick 10000
```

```
10000 early()  
{
```

```
    sim.simulationFinished();
```

```
}
```

Set mutation rate!

Set mutation types

Set genomic elements

Set sequence length!

Set recombination rate!



Set population size

Set simulated generations

# Play!


---

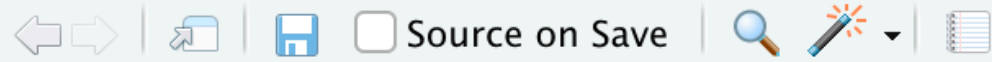


- To run a new simulation, click the  button then the  button.
- The SFS we generated had a lot of missing data... What parameters should we change to fill out the plot?
- Change some parameters to change, rerun the simulation, and see if you can get a smooth plot!

# Plotting in Studio

---

- From the plotting window, click 
  - You can change some plot settings
  - Click “Copy Data”
- Open Rstudio!
  - Click File > New File > R Script
  - Paste clipboard into the black script.



```
1 # Graph data: 1D Sample SFS
2 # 7/12/23, 8:28:59 AM
3
4 4590, 1881, 1153, 903, 716, 695, 700, 397, 452, 437, 285, 420, 298, 202, 195, 189, 218, 261, 210, 0
5
```

5:1 (Top Level) ⌵

R Script ⌵

---

- # Graph data: 1D Sample SFS

- # 7/12/23, 8:28:59 AM

- 4590, 1881, 1153, 903, 716, 695, 700, 397, 452, 437, 285,  
420, 298, 202, 195, 189, 218, 261, 210, 0






---

- # Graph data: 1D Sample SFS

- # 7/12/23, 8:28:59 AM

- **sf = c**(4590, 1881, 1153, 903, 716, 695, 700, 397, 452,  
437, 285, 420, 298, 202, 195, 189, 218, 261, 210, 0)**)**

- 
- # Graph data: 1D Sample SFS
  - # 7/12/23, 8:28:59 AM
  - **sf = c(**4590, 1881, 1153, 903, 716, 695, 700, 397, 452,  
437, 285, 420, 298, 202, 195, 189, 218, 261, 210, 0**)**
  - **plot(sf, type='l')**

- 
- # Graph data: 1D Sample SFS
  - # 7/12/23, 8:28:59 AM
  - `sf = c(4590, 1881, 1153, 903, 716, 695, 700, 397, 452, 437, 285, 420, 298, 202, 195, 189, 218, 261, 210, 0)`
  - `plot(sf, type='l')`
  - Now click  Source

- 
- # Graph data: 1D Sample SFS
  - # 7/12/23, 8:28:59 AM
  - **sf = c**(4590, 1881, 1153, 903, 716, 695, 700, 397, 452, 437, 285, 420, 298, 202, 195, 189, 218, 261, 210, 0)
  - **plot(sf, type='l')**

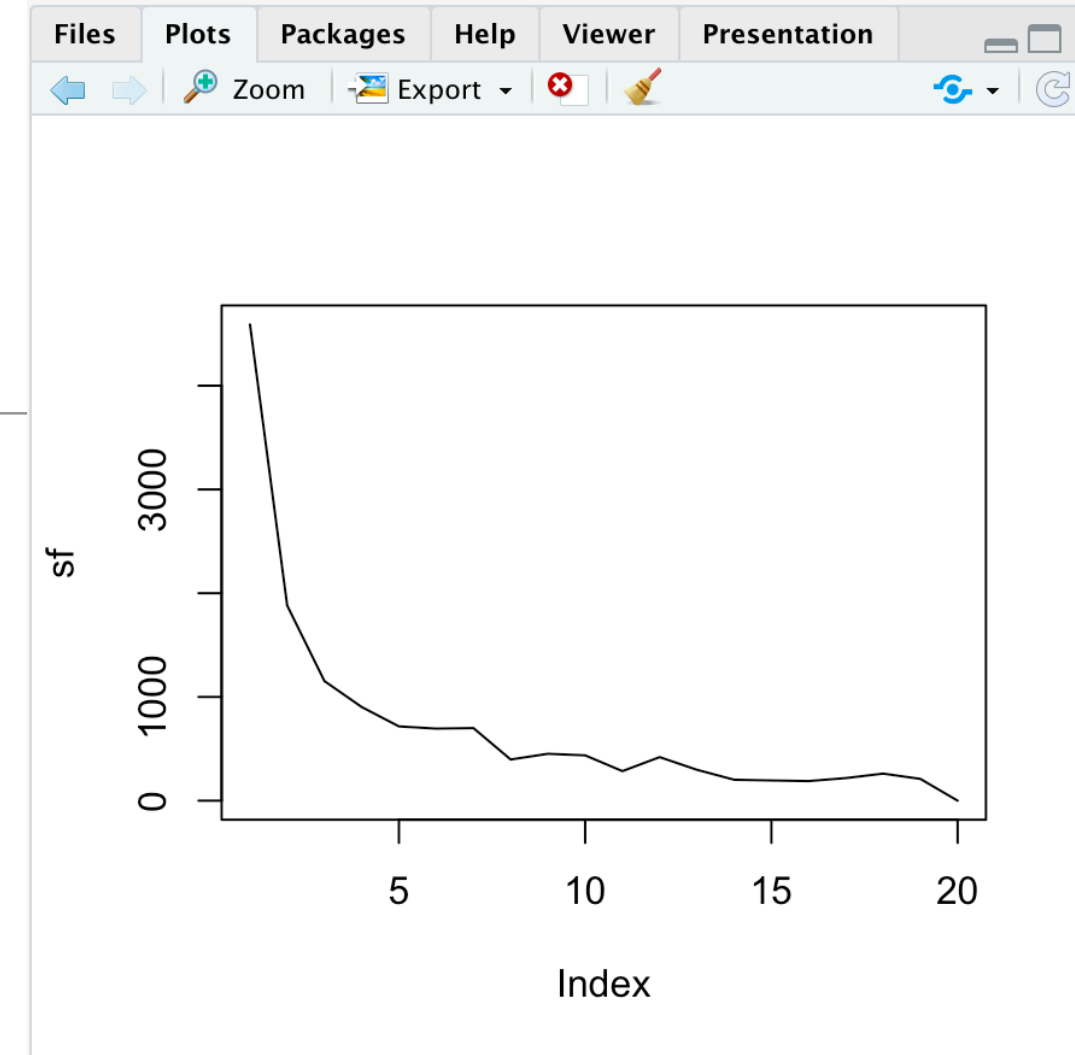
- Now click  Source

- You should see a plot!

---

- # Graph data: 1D Sample SFS

- # 7/12/23, 8:28:59 AM



- **sf = c(**4590, 1881, 1153, 903, 716, 695, 700, 397, 452,  
437, 285, 420, 298, 202, 195, 189, 218, 261, 210, **)**

- **plot(sf, type='l')**

- Now click  Source

- You should see a plot!

- 
- Let's see what happens with Recipe 5.1.1.

- 
- Let's see what happens with Recipe 5.1.1.

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
10000 late() { sim.outputFixedMutations(); }
```

- 
- Let's see what happens with Recipe 5.1.1.

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=1000$



- 
- Let's see what happens with Recipe 5.1.1.

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=1000$

At generation 1000,  
 $N_e$  shrinks to 100

- 
- Let's see what happens with Recipe 5.1.1.

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=1000$

At generation 1000,  
 $N_e$  shrinks to 100

At generation 2000,  
 $N_e$  grows to 1000

- Let's see what happens with Recipe 5.1.1.

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=1000$

At generation 1000,  
 $N_e$  shrinks to 100

At generation 2000,  
 $N_e$  grows to 1000

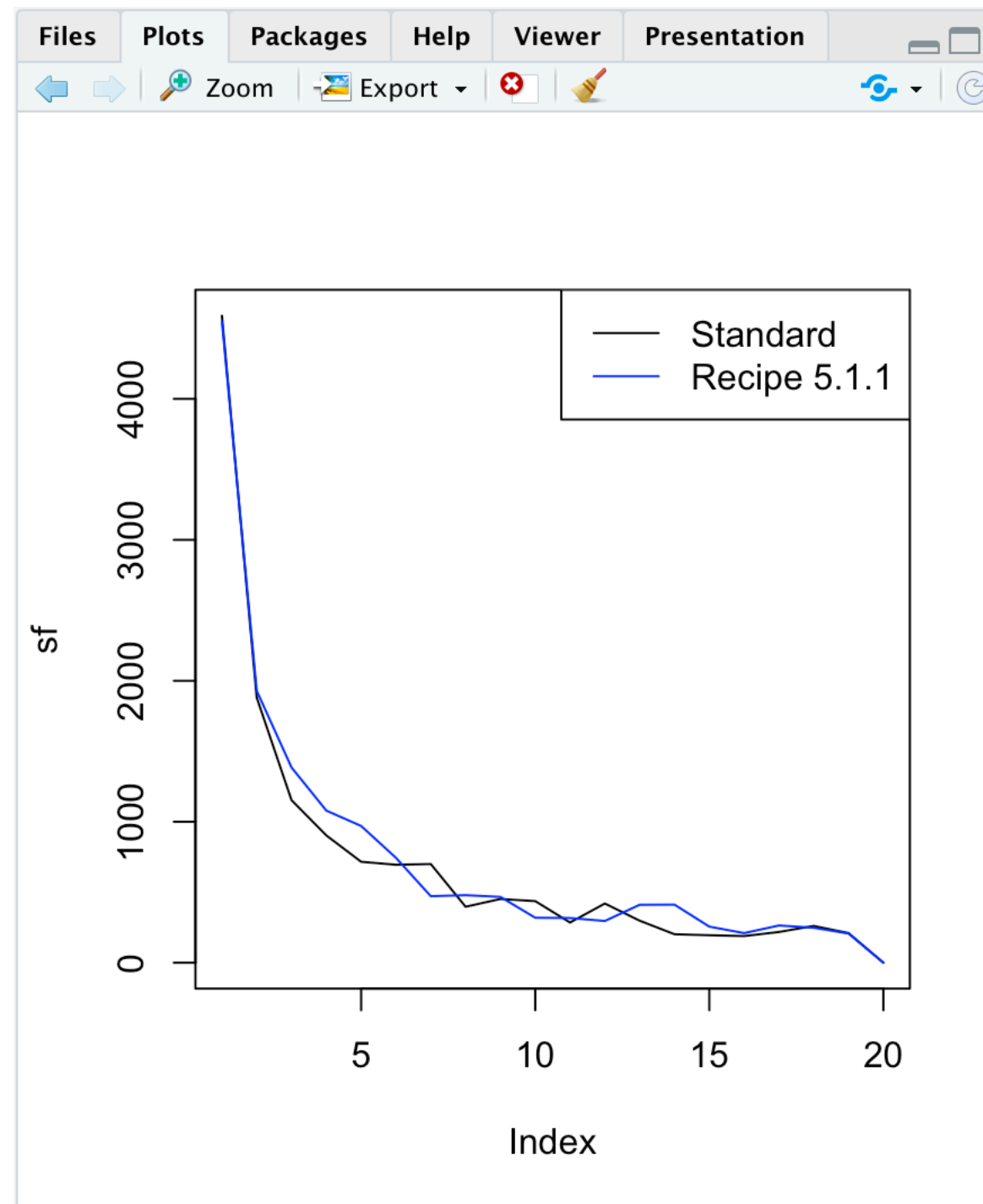
At generation 10000,  
simulation ends

# Graph 1D Sample SFS

---

# Graph 1D Sample SFS

---



# Population Bottlenecks: Recipe 5.1.1

---

# Population Bottlenecks: Recipe 5.1.1

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
3000 late() { sim.outputFixedMutations(); }
```

# Population Bottlenecks: Recipe 5.1.1

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
3000 late() { sim.outputFixedMutations(); }
```

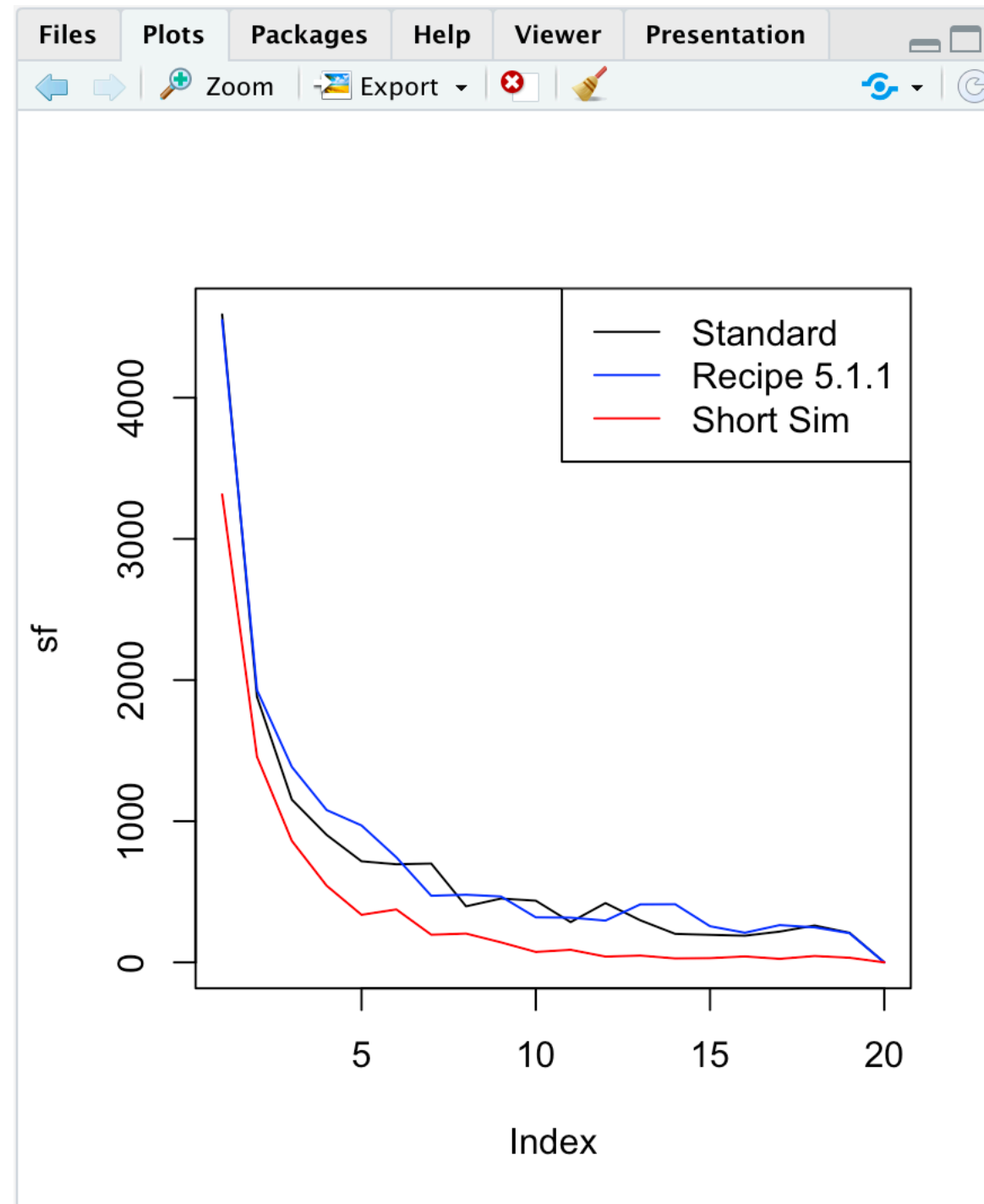


At generation 3000,  
simulation ends



# Graph 1D Sample SFS

---



# Exponential Growth: Recipe 5.1.2

---

# Exponential Growth: Recipe 5.1.2

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 9999999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 100); }  
1000:1099 early() {  
    newSize = asInteger(p1.individualCount * 1.03);  
    p1.setSubpopulationSize(newSize);  
}  
10000 late() { sim.outputFixedMutations(); }
```

# Exponential Growth: Recipe 5.1.2

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 9999999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 100); }  
1000:1099 early() {  
    newSize = asInteger(p1.individualCount * 1.03);  
    p1.setSubpopulationSize(newSize);  
}  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=100$

# Exponential Growth: Recipe 5.1.2

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 9999999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 100); }  
1000:1099 early() {  
    newSize = asInteger(p1.individualCount * 1.03);  
    p1.setSubpopulationSize(newSize);  
}  
10000 late() { sim.outputFixedMutations(); }
```

Population starts  
with  $N_e=100$

From gen  
1000-1099,  
population grows  
by 3% every  
generation

# Exponential Growth: Recipe 5.1.2

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 9999999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 100); }  
1000:1099 early() {  
    newSize = asInteger(p1.individualCount * 1.03);  
    p1.setSubpopulationSize(newSize);  
}  
10000 late() { sim.outputFixedMutations(); }
```

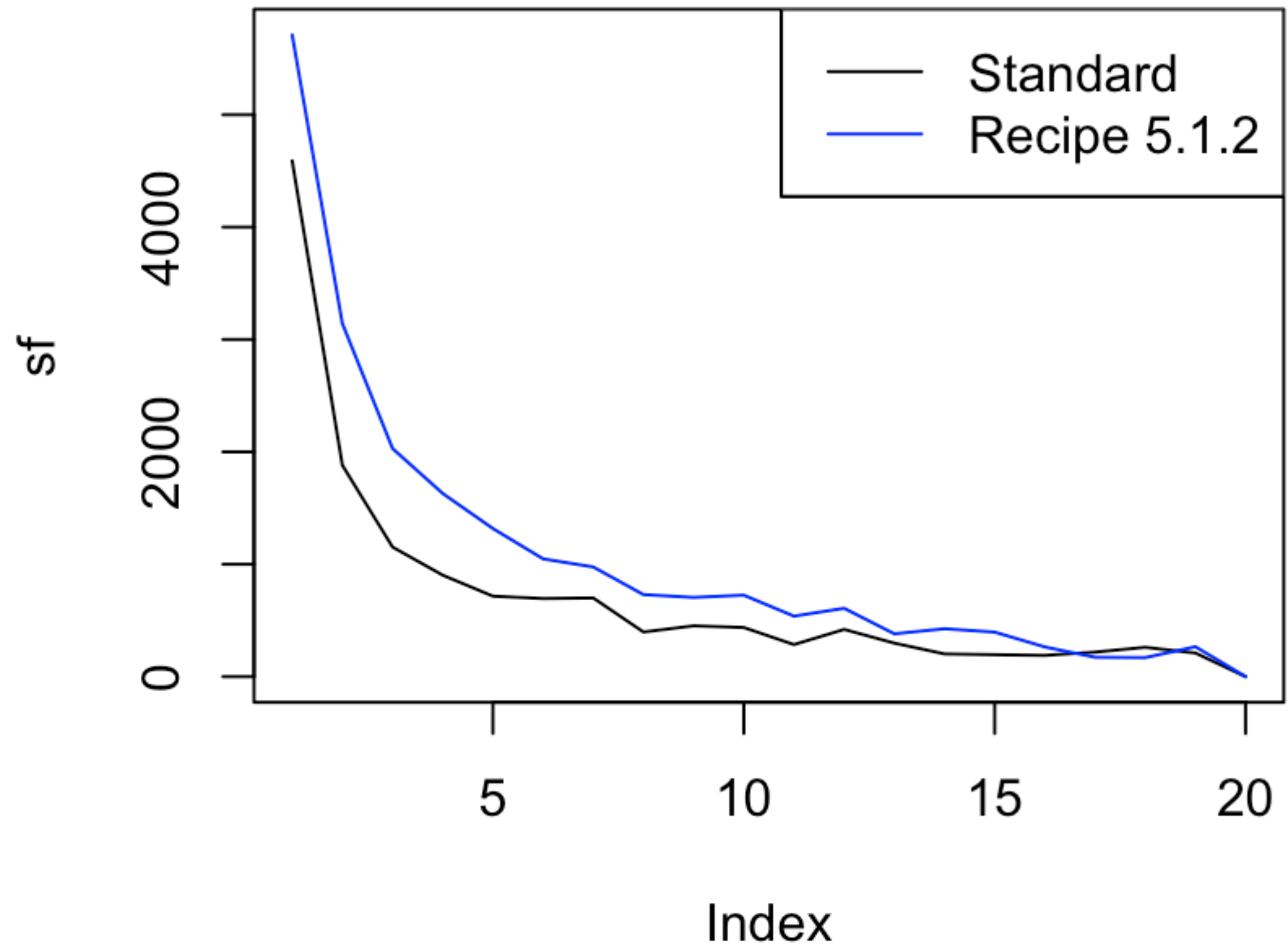
Population starts  
with  $N_e=100$

From gen  
1000-1099,  
population grows  
by 3% every  
generation

At generation 10000,  
simulation ends

# 1D Sample SFS

---



# Population Bottlenecks: Recipe 5.1.1

---



# Population Bottlenecks: Recipe 5.1.1

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
3000 late() { sim.outputFixedMutations(); }
```

# Population Bottlenecks: Recipe 5.1.1

---

// Keywords:

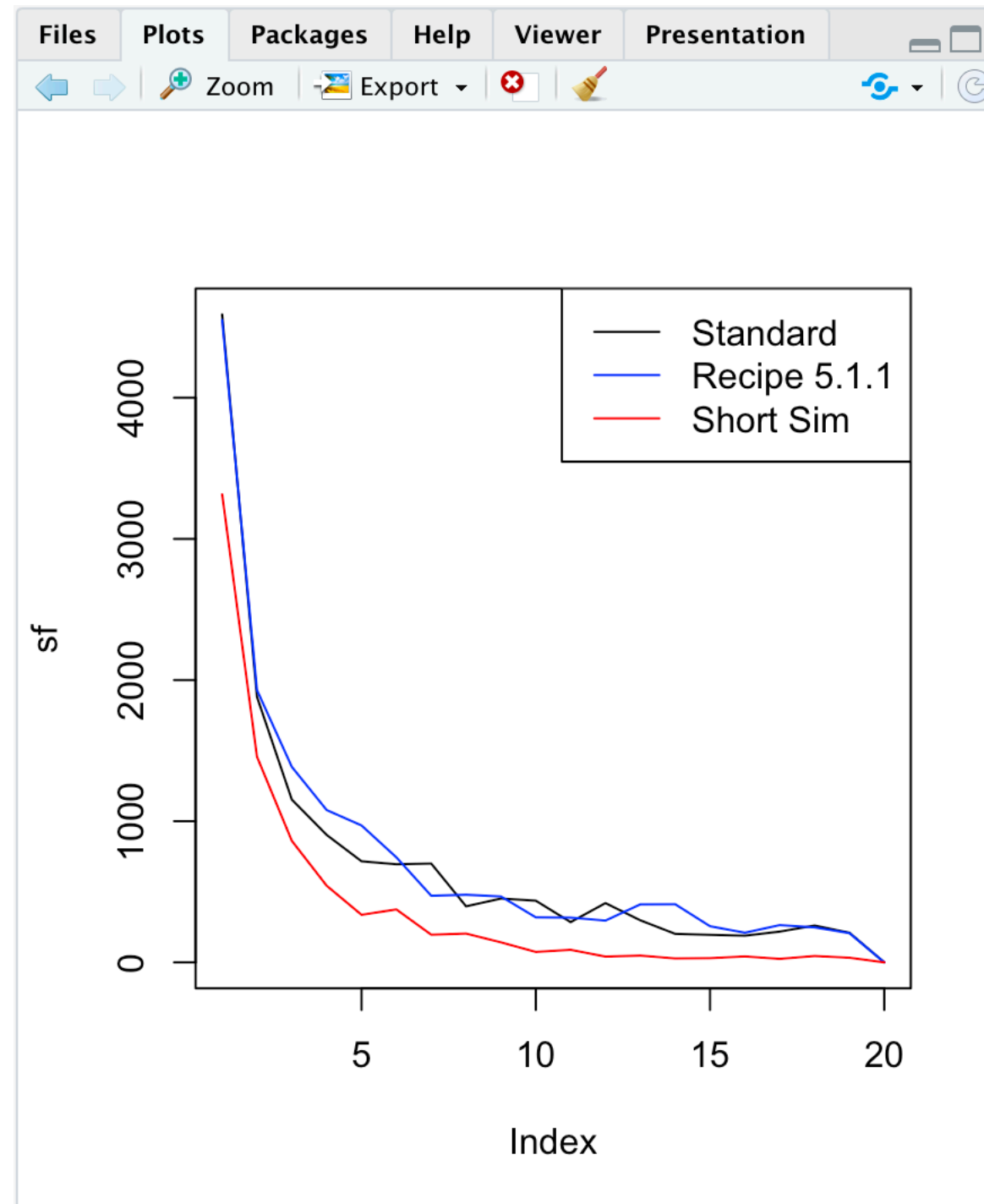
```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 1000); }  
1000 early() { p1.setSubpopulationSize(100); }  
2000 early() { p1.setSubpopulationSize(1000); }  
3000 late() { sim.outputFixedMutations(); }
```



At generation 3000,  
simulation ends

# Graph 1D Sample SFS

---



# Population Structure: Recipe 5.2.1

---

# Population Structure: Recipe 5.2.1

---

```
// Keywords: migration, dispersal

initialize() {
  initializeMutationRate(1e-7);
  initializeMutationType("m1", 0.5, "f", 0.0);
  initializeGenomicElementType("g1", m1, 1.0);
  initializeGenomicElement(g1, 0, 1999999);
  initializeRecombinationRate(1e-8);
}

1 early() {
  sim.addSubpop("p1", 500);
  sim.addSubpop("p2", 100);
  sim.addSubpop("p3", 1000);
  p1.setMigrationRates(c(p2,p3), c(0.2,0.1));
  p2.setMigrationRates(c(p1,p3), c(0.8,0.01));
}

10000 late() { sim.outputFixedMutations(); }
```

# Population Structure: Recipe 5.2.1

---

```
// Keywords: migration, dispersal
```

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 1999999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() {  
    sim.addSubpop("p1", 500);  
    sim.addSubpop("p2", 100);  
    sim.addSubpop("p3", 1000);  
    p1.setMigrationRates(c(p2,p3), c(0.2,0.1));  
    p2.setMigrationRates(c(p1,p3), c(0.8,0.01));  
}  
10000 late() { sim.outputFixedMutations(); }
```

Population 1 starts  
with  $N_e=500$

# Population Structure: Recipe 5.2.1

---

```
// Keywords: migration, dispersal
```

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 1999999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() {  
  sim.addSubpop("p1", 500);  
  sim.addSubpop("p2", 100);  
  sim.addSubpop("p3", 1000);  
  p1.setMigrationRates(c(p2,p3), c(0.2,0.1));  
  p2.setMigrationRates(c(p1,p3), c(0.8,0.01));  
}  
10000 late() { sim.outputFixedMutations(); }
```

Population 1 starts  
with  $N_e=500$

Population 2 starts  
with  $N_e=100$

# Population Structure: Recipe 5.2.1

---

```
// Keywords: migration, dispersal
```

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 1999999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() {  
  sim.addSubpop("p1", 500);  
  sim.addSubpop("p2", 100);  
  sim.addSubpop("p3", 1000);  
  p1.setMigrationRates(c(p2,p3), c(0.2,0.1));  
  p2.setMigrationRates(c(p1,p3), c(0.8,0.01));  
}  
10000 late() { sim.outputFixedMutations(); }
```

Population 1 starts  
with  $N_e=500$

Population 2 starts  
with  $N_e=100$

Population 3 starts  
with  $N_e=1000$



# Population Structure: Recipe 5.2.1

---

```
// Keywords: migration, dispersal
```

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 1999999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() {  
  sim.addSubpop("p1", 500);  
  sim.addSubpop("p2", 100);  
  sim.addSubpop("p3", 1000);  
  p1.setMigrationRates(c(p2,p3), c(0.2,0.1));  
  p2.setMigrationRates(c(p1,p3), c(0.8,0.01));  
}  
10000 late() { sim.outputFixedMutations(); }
```

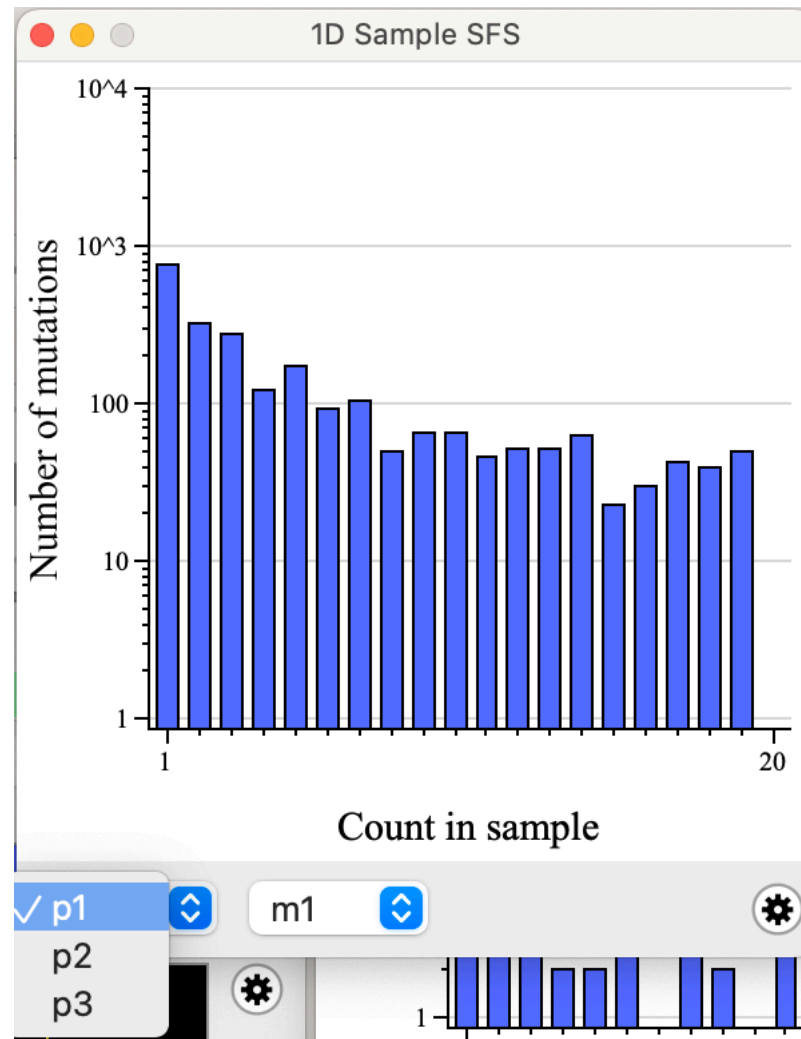
Population 1 starts  
with  $N_e=500$

Population 2 starts  
with  $N_e=100$

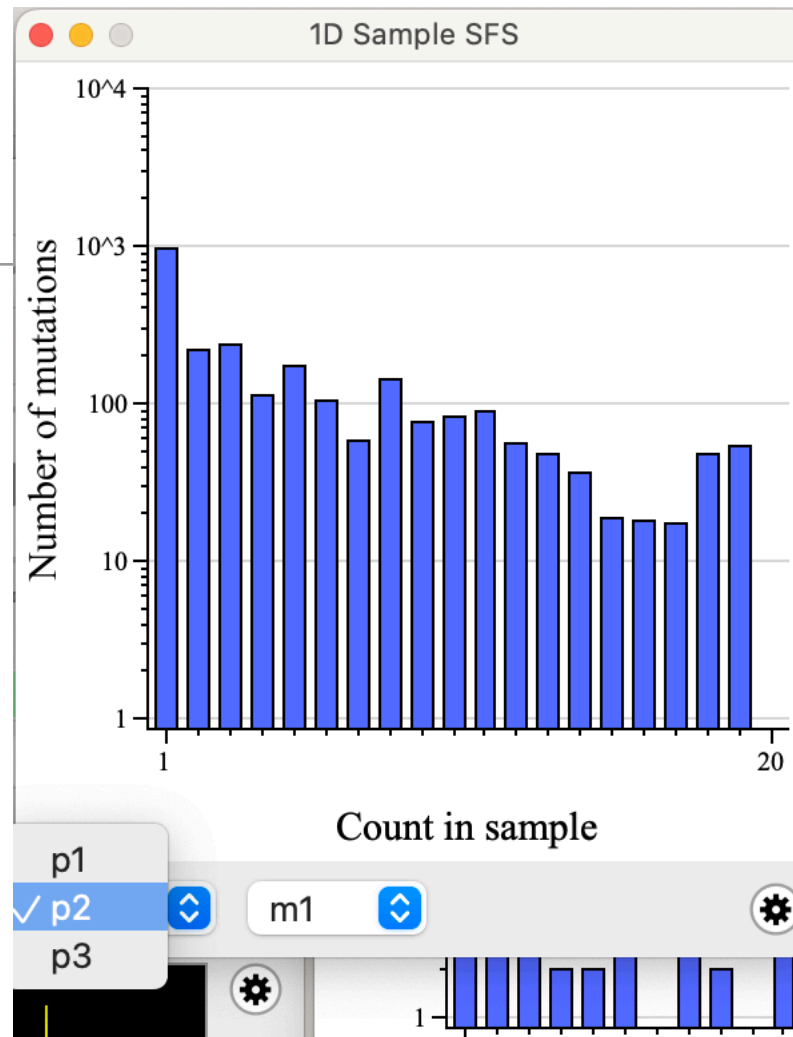
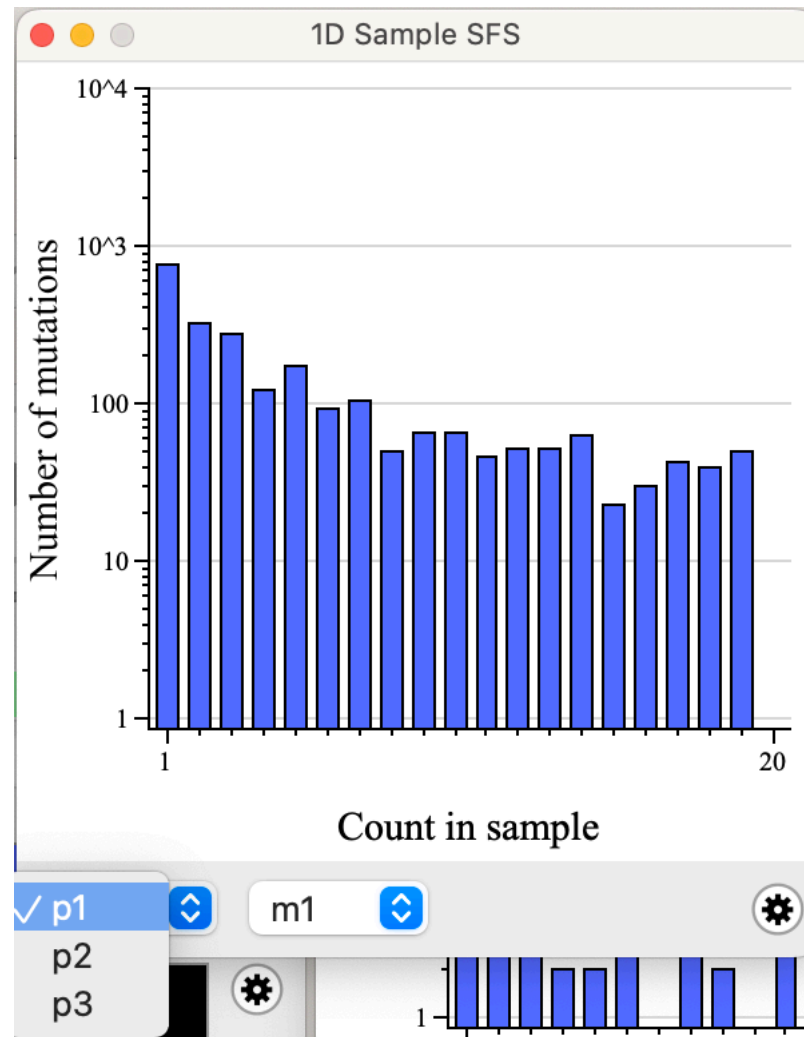
Population 3 starts  
with  $N_e=1000$

p1 migrates to p2 and p3;  
p2 migrates to p1 and p3

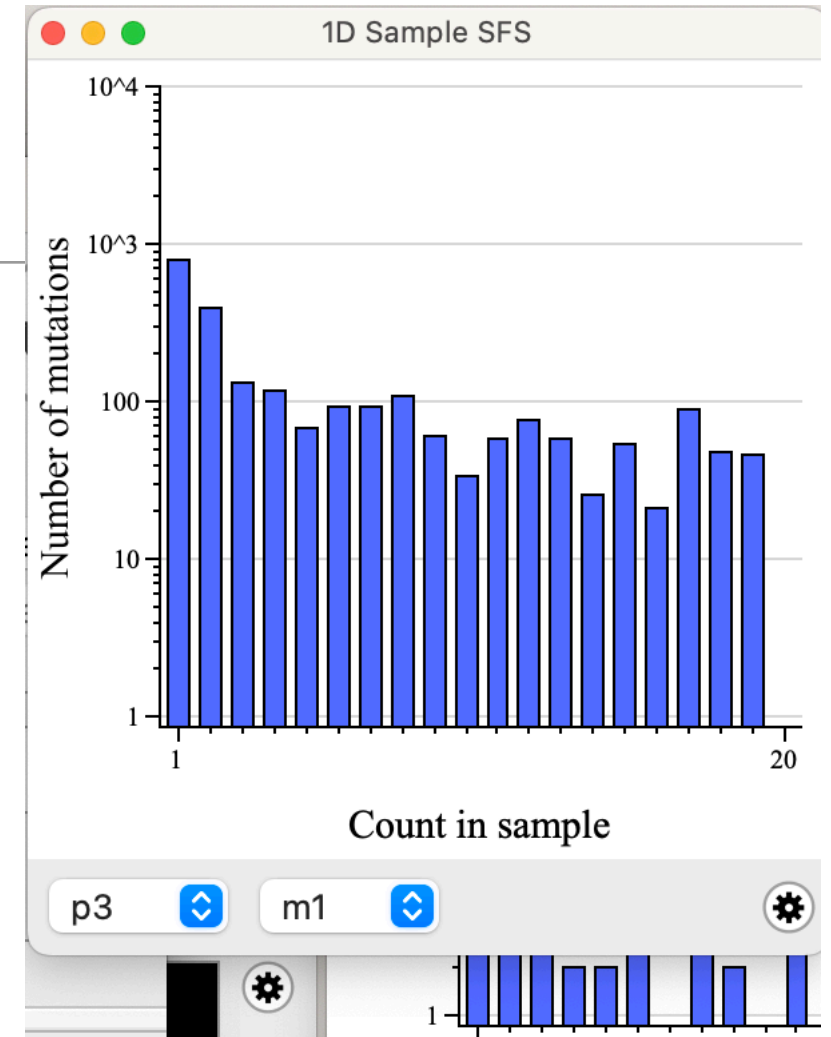
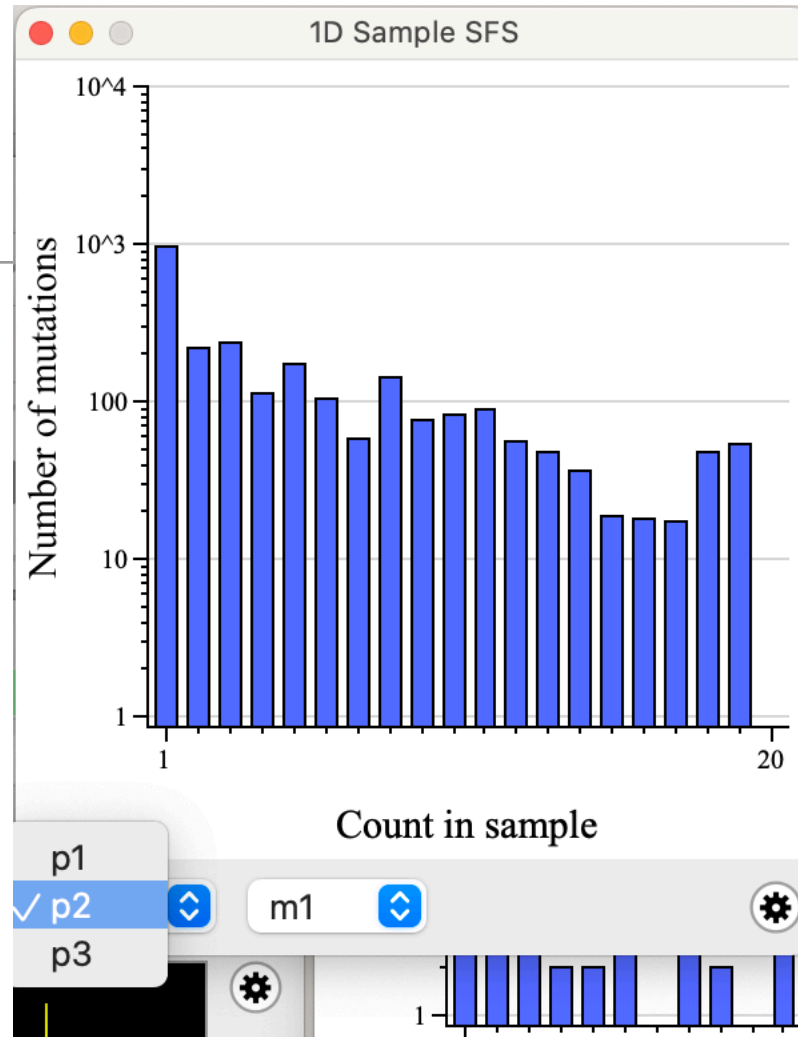
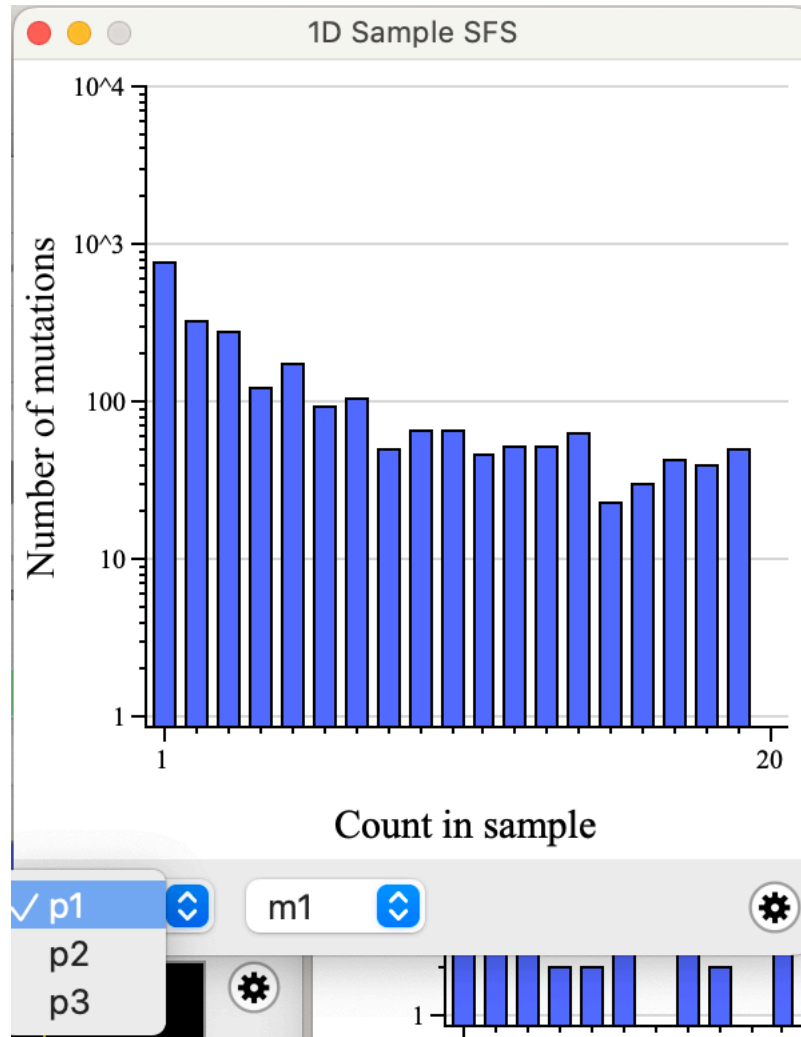
# 1D Sample SFS



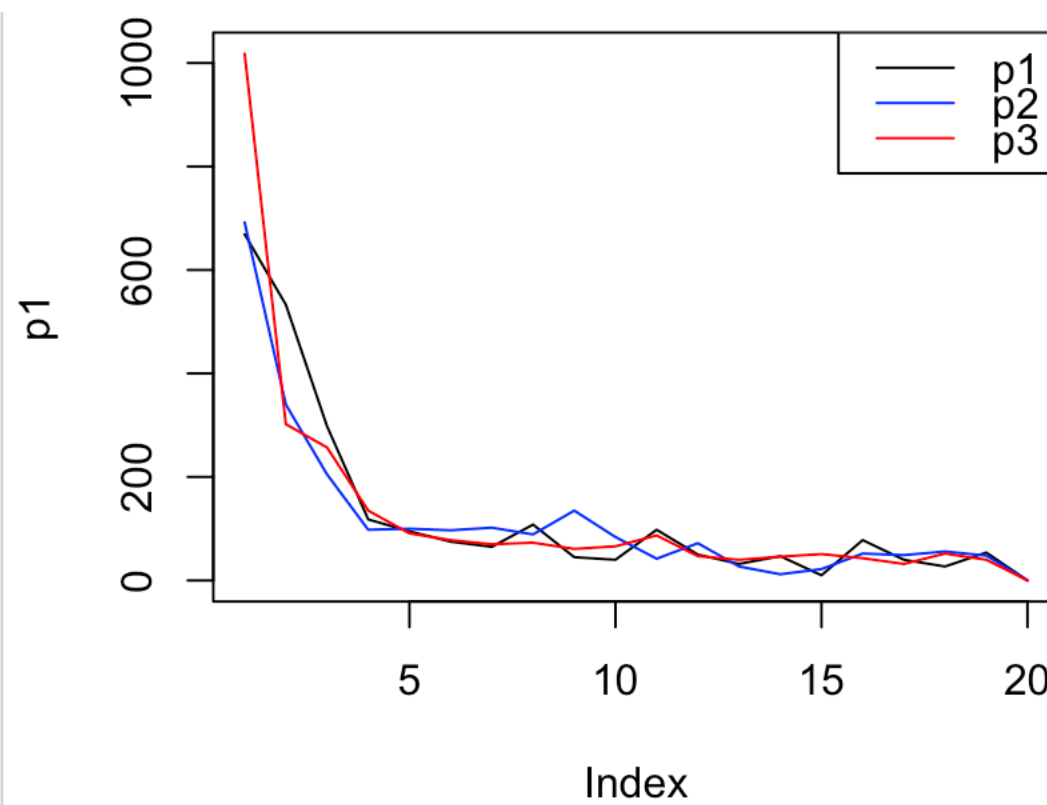
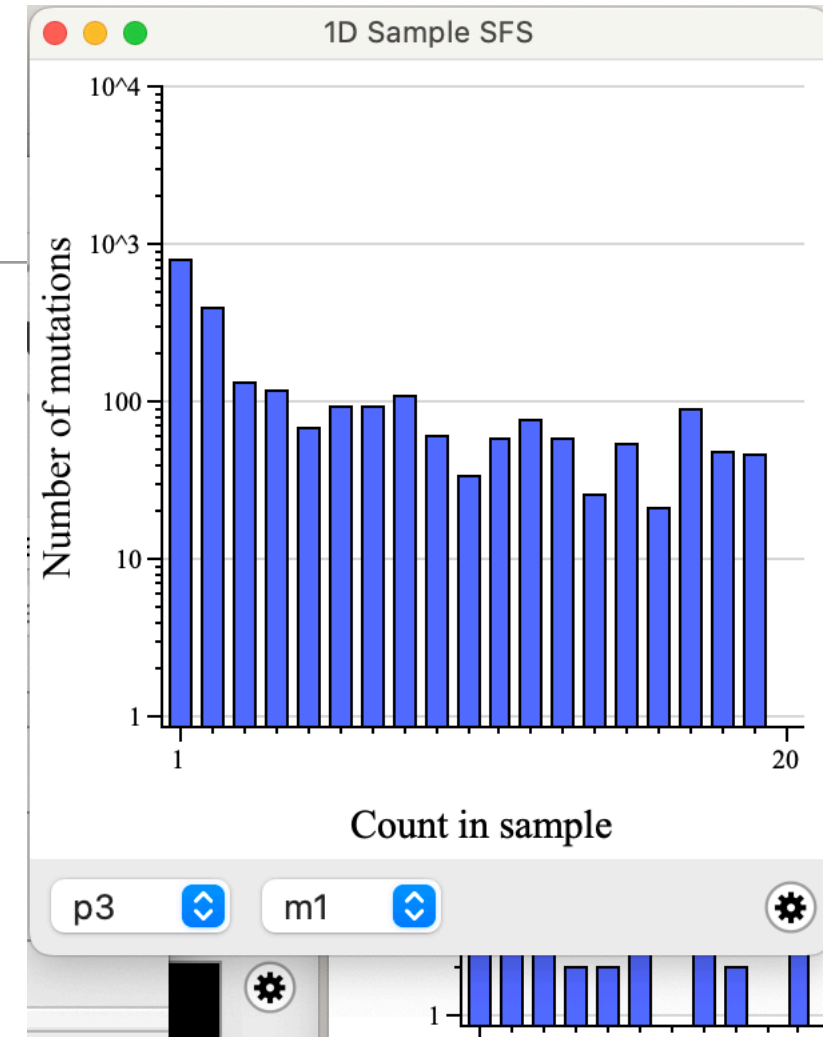
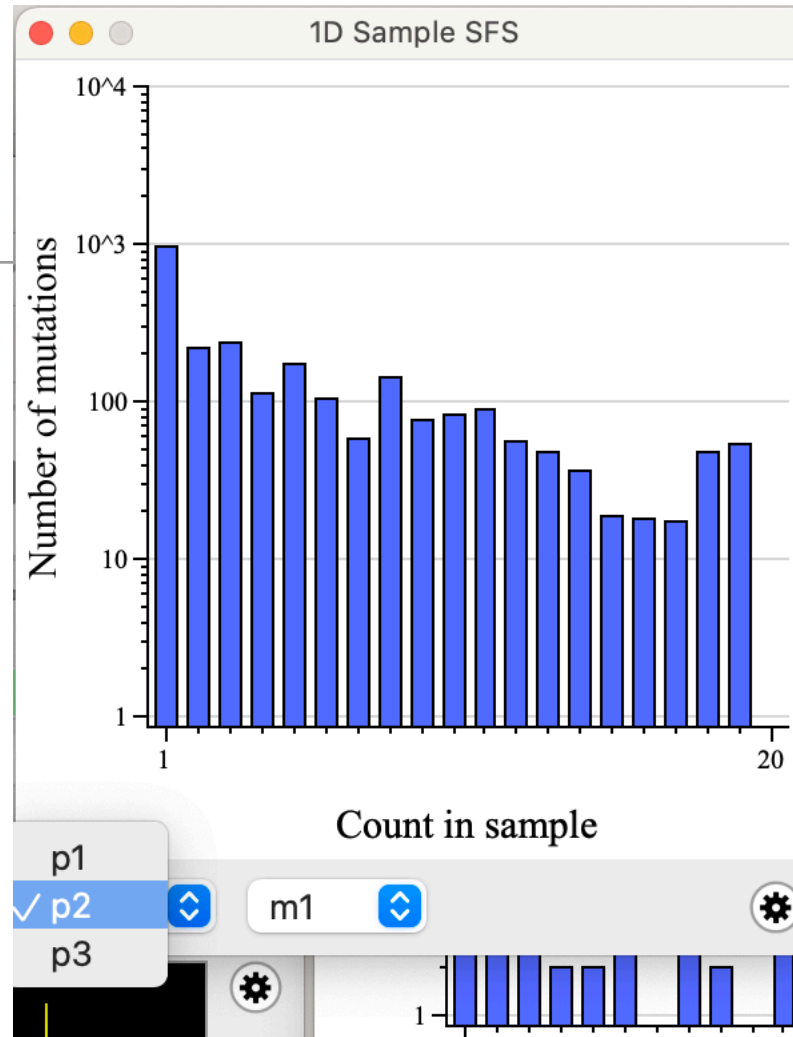
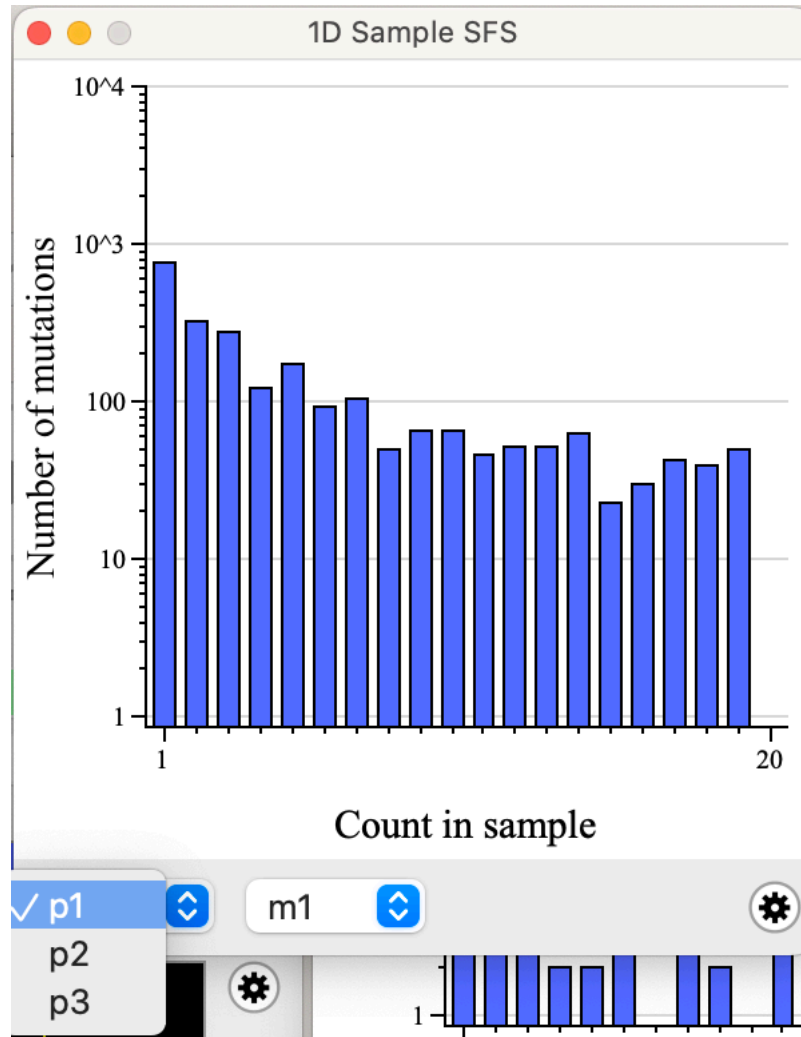
# 1D Sample SFS



# 1D Sample SFS



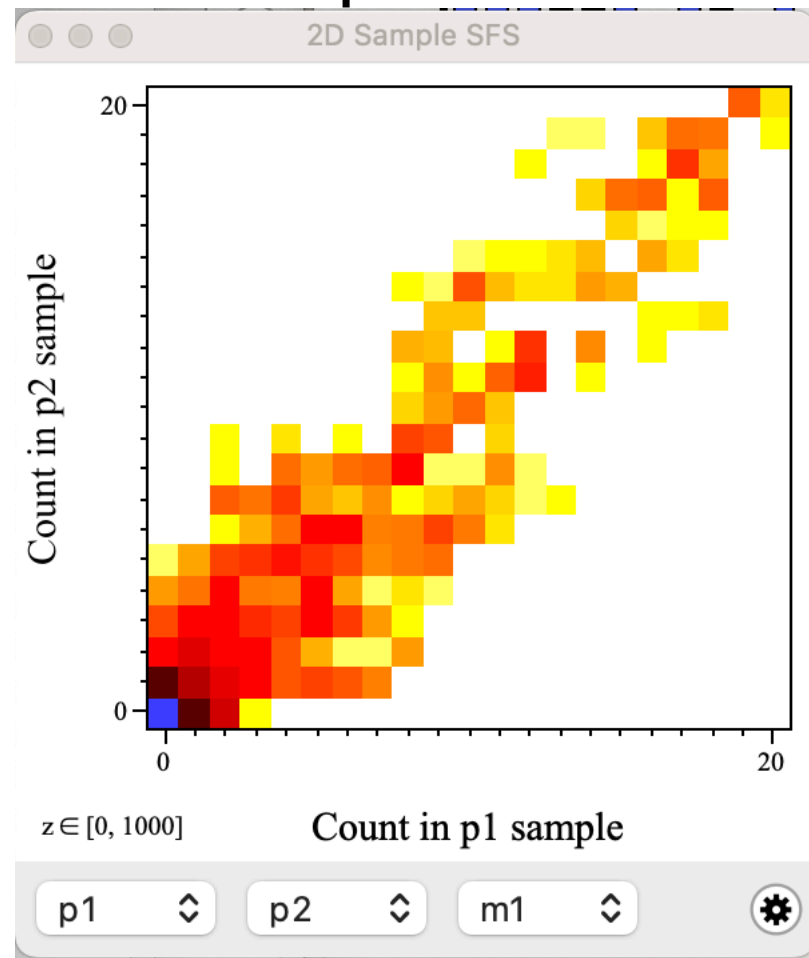
# 1D Sample SFS



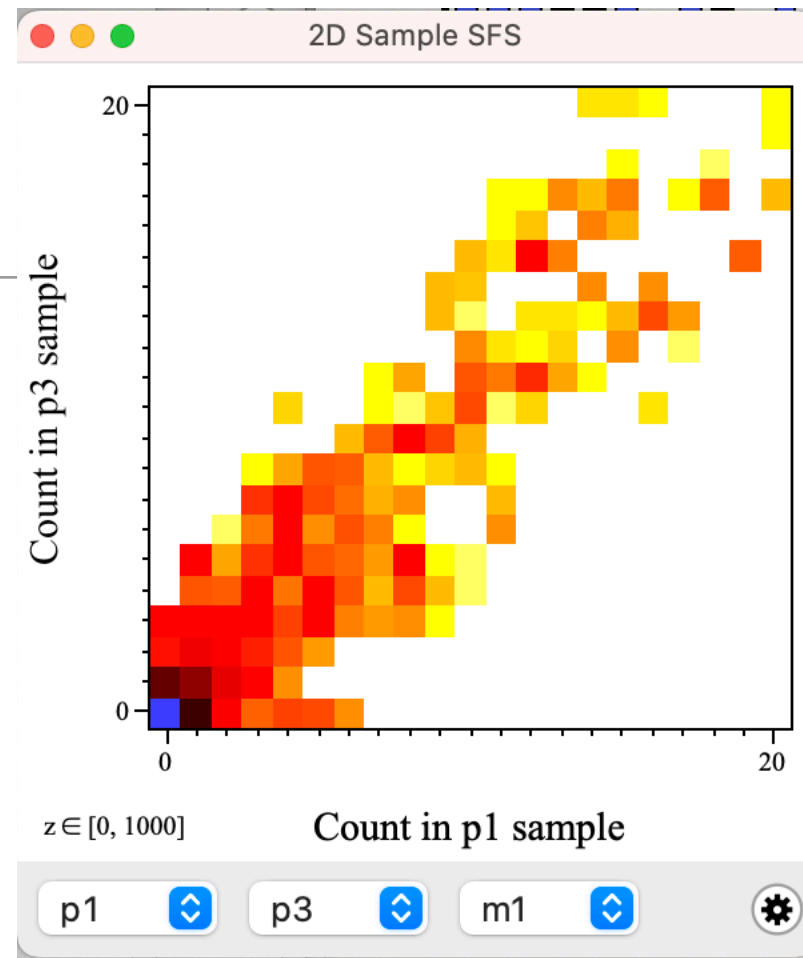
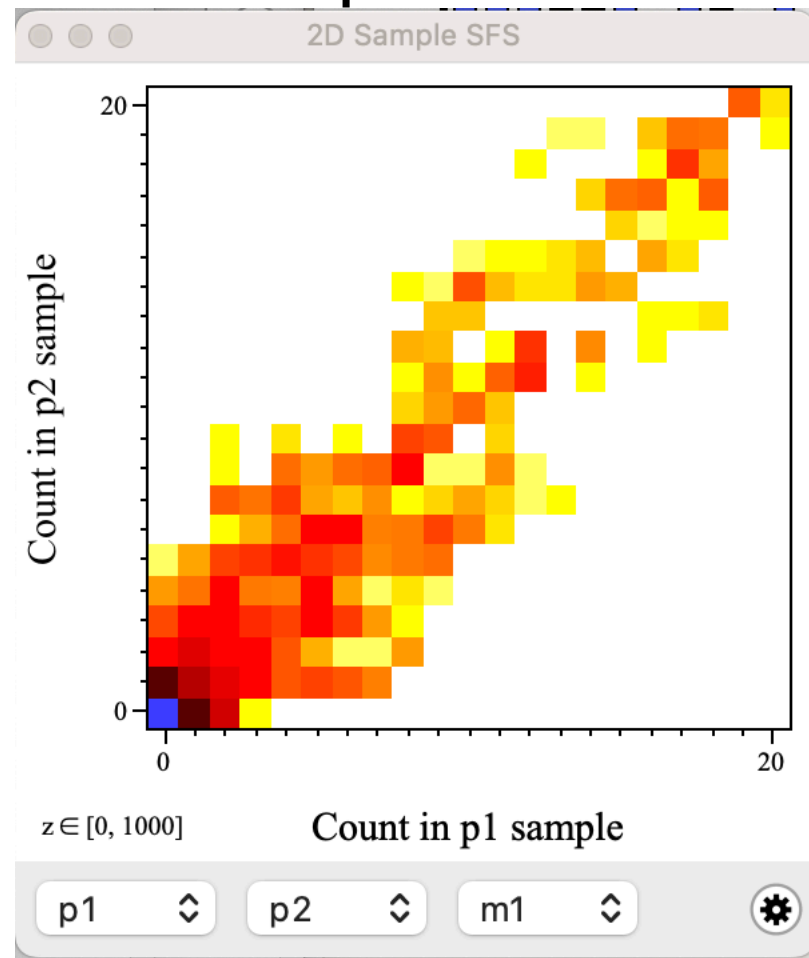
# 2D Sample SFS

---

# 2D Sample SFS

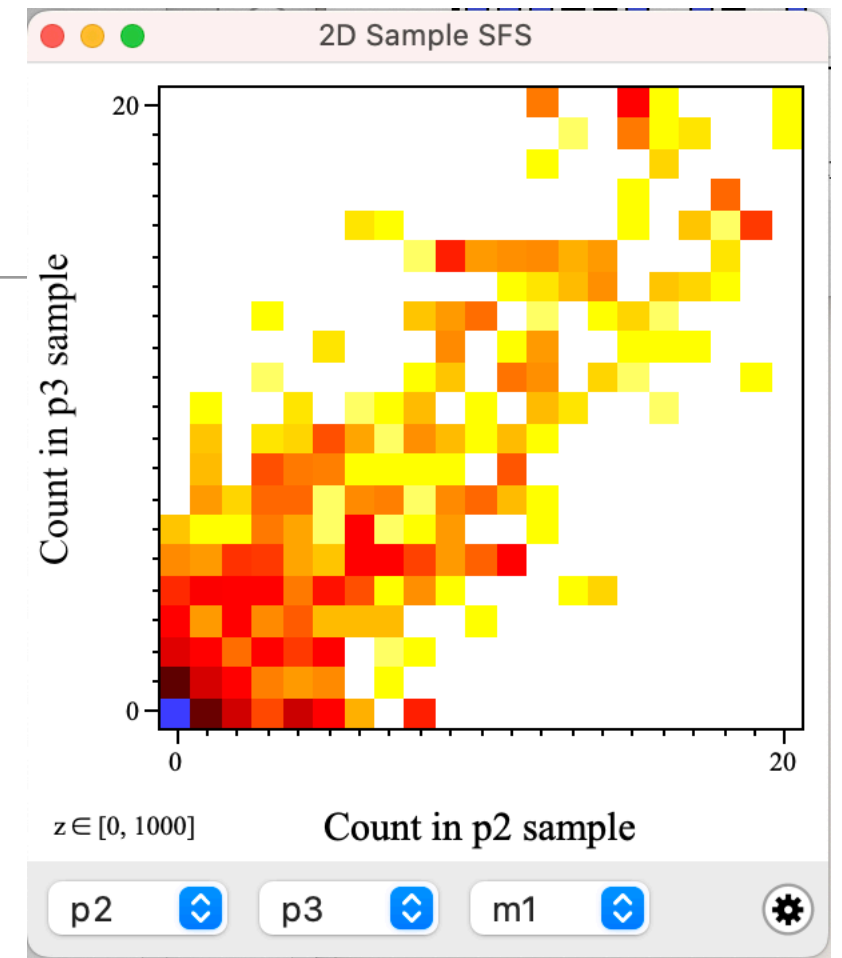
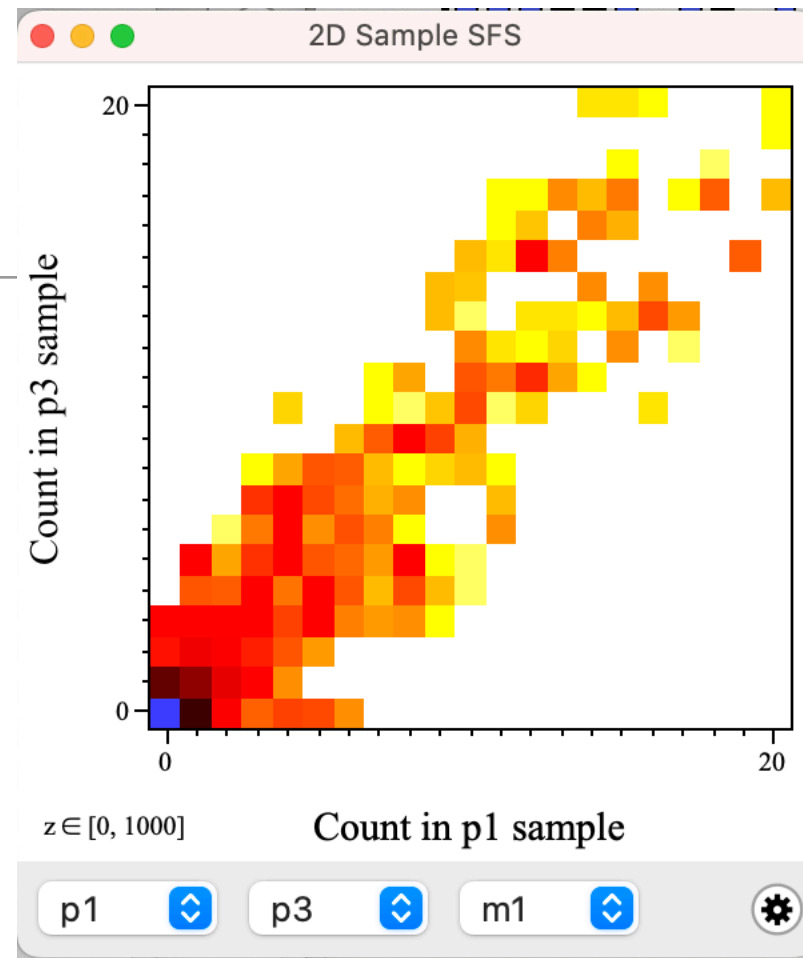
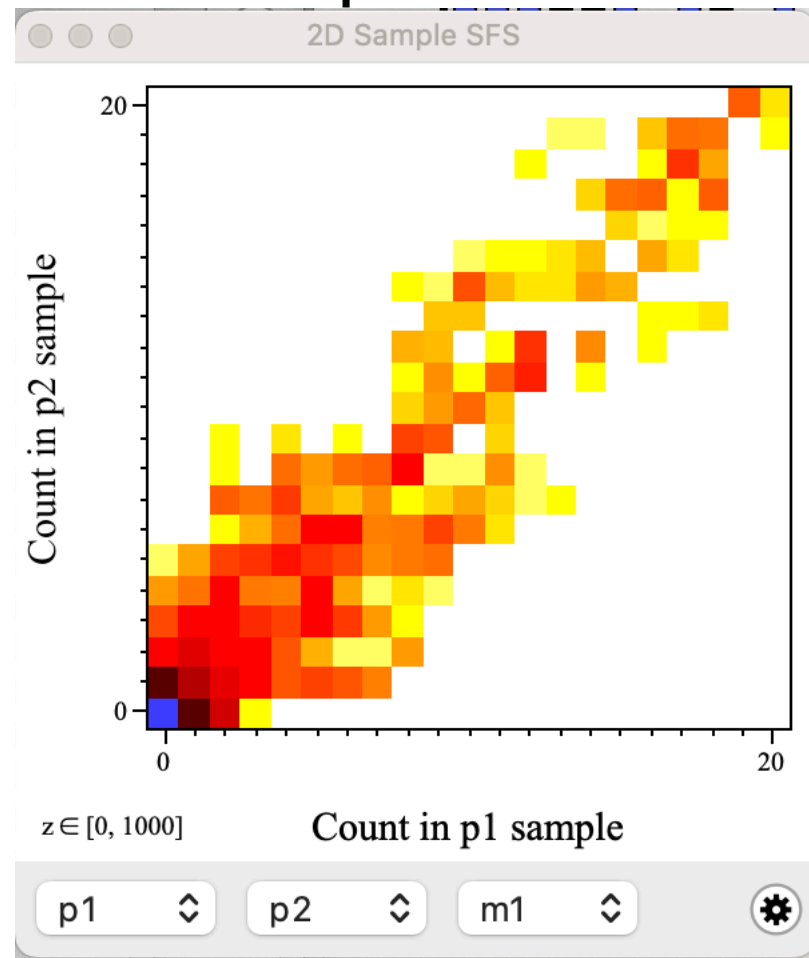


# 2D Sample SFS

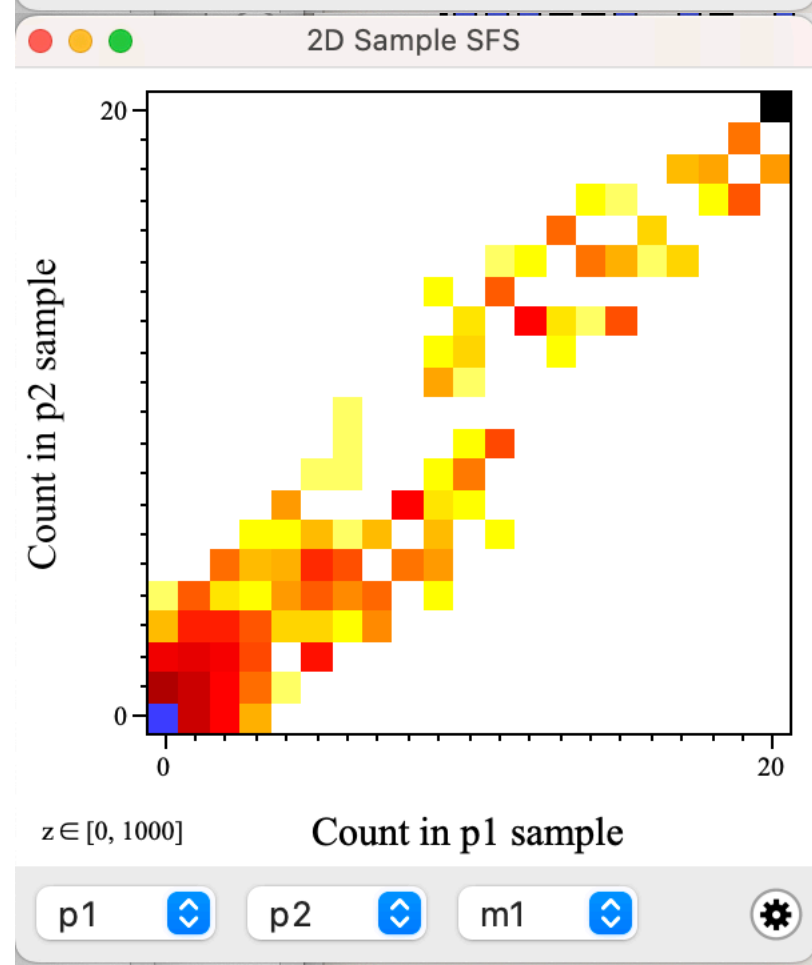
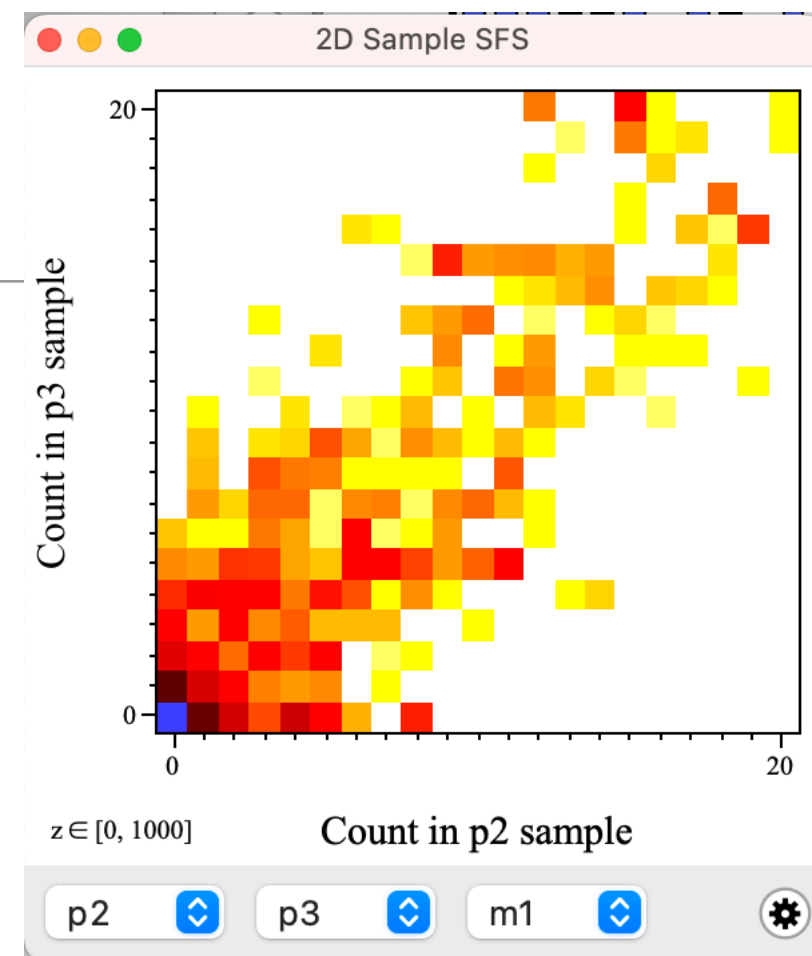
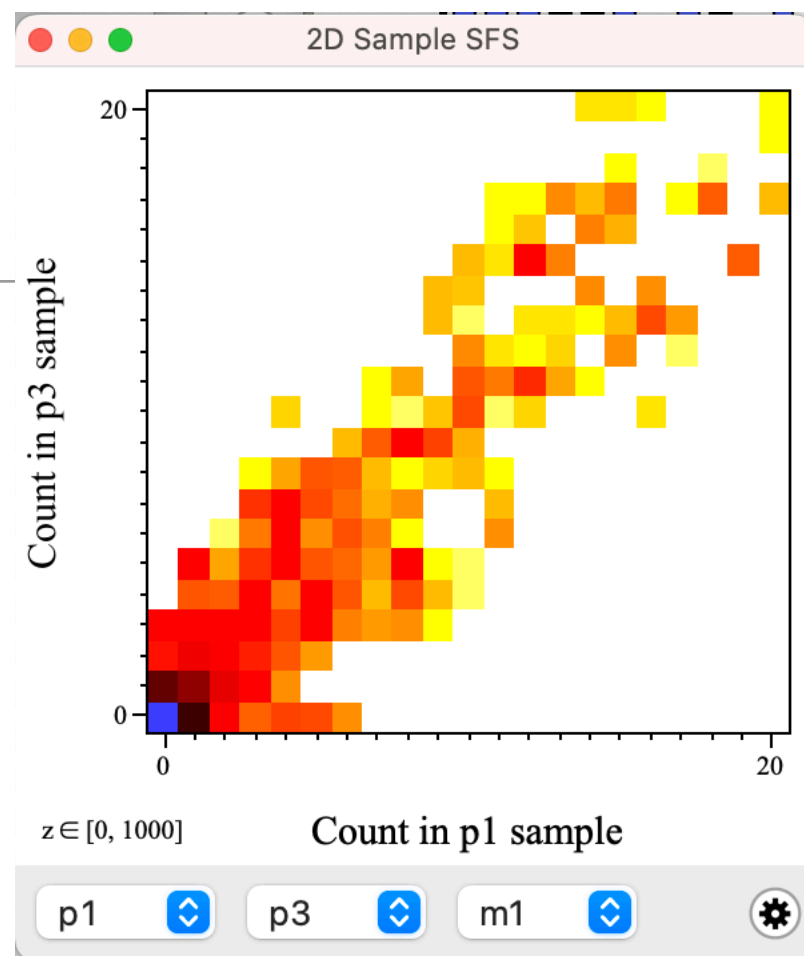
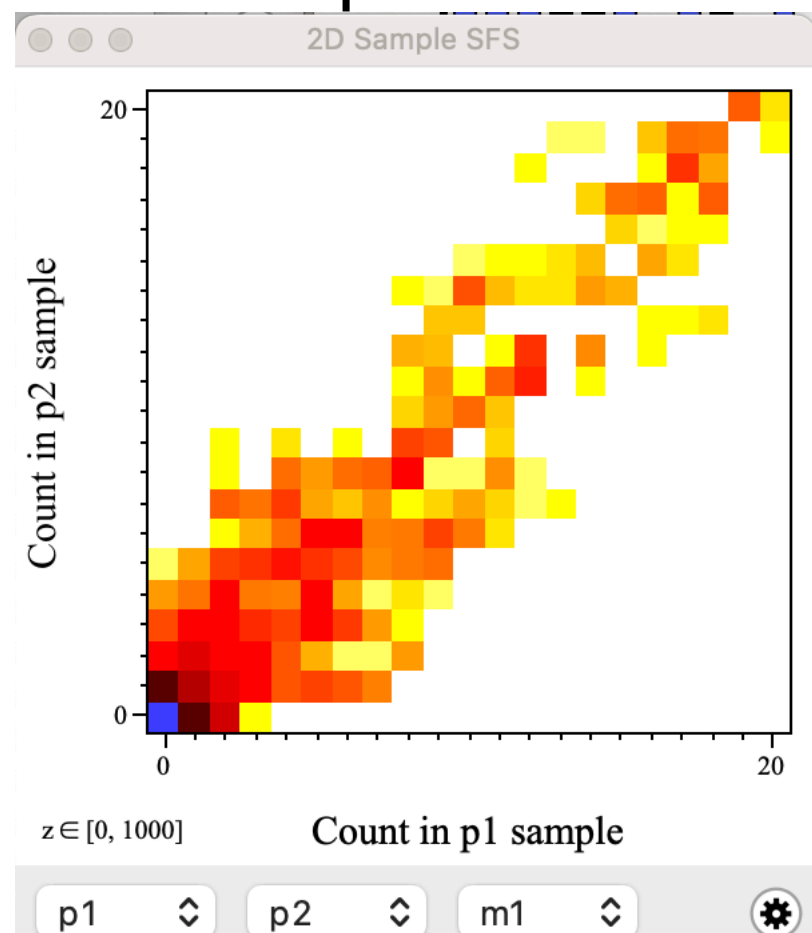




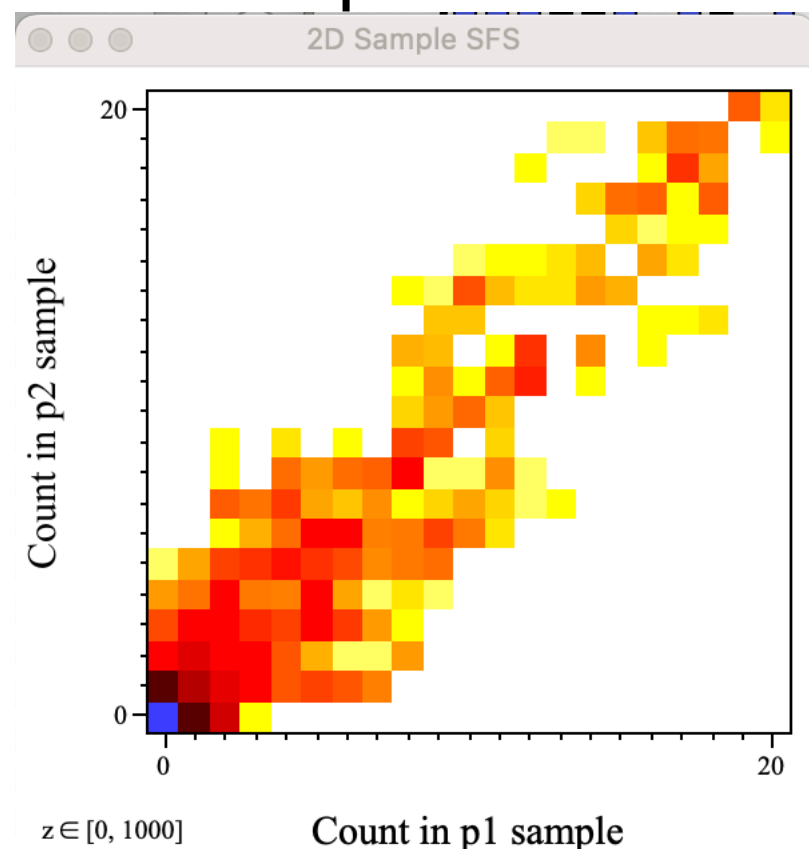
# 2D Sample SFS



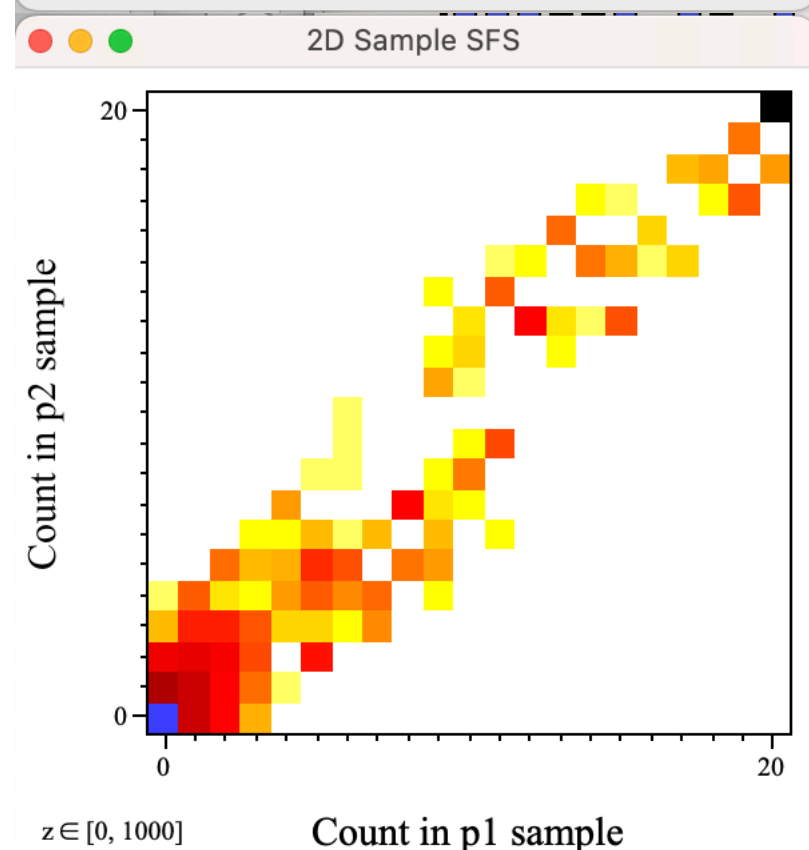
# 2D Sample SFS



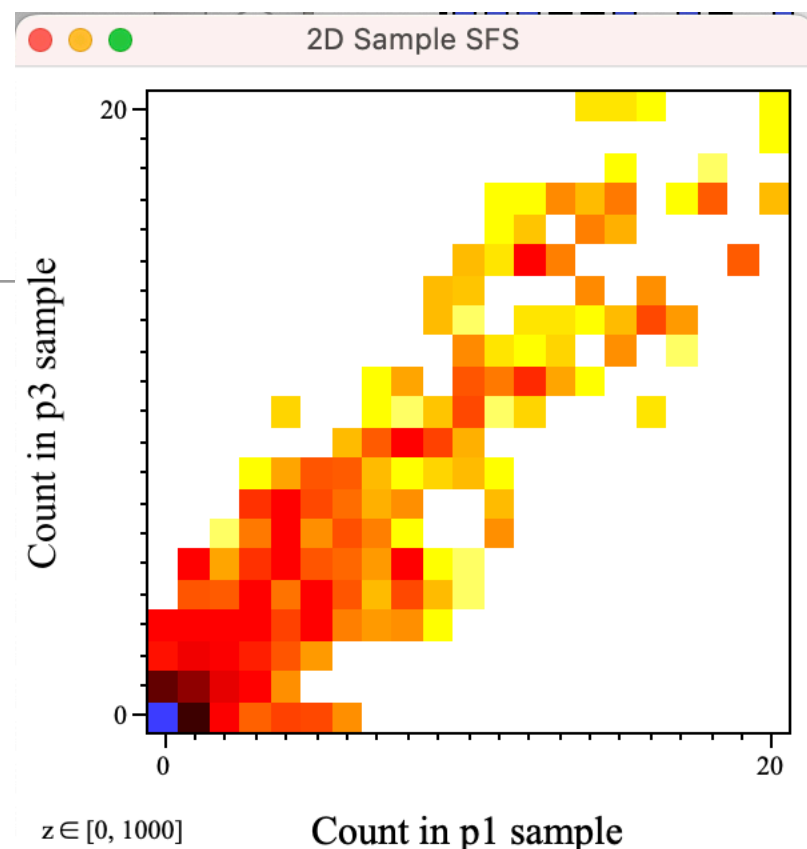
# 2D Sample SFS



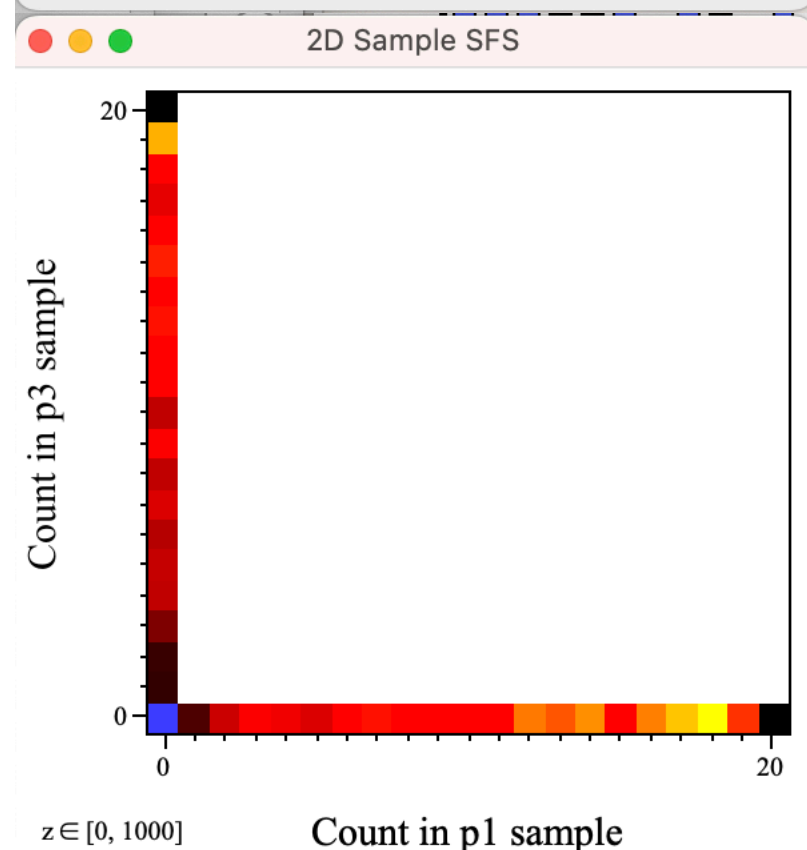
p1 p2 m1



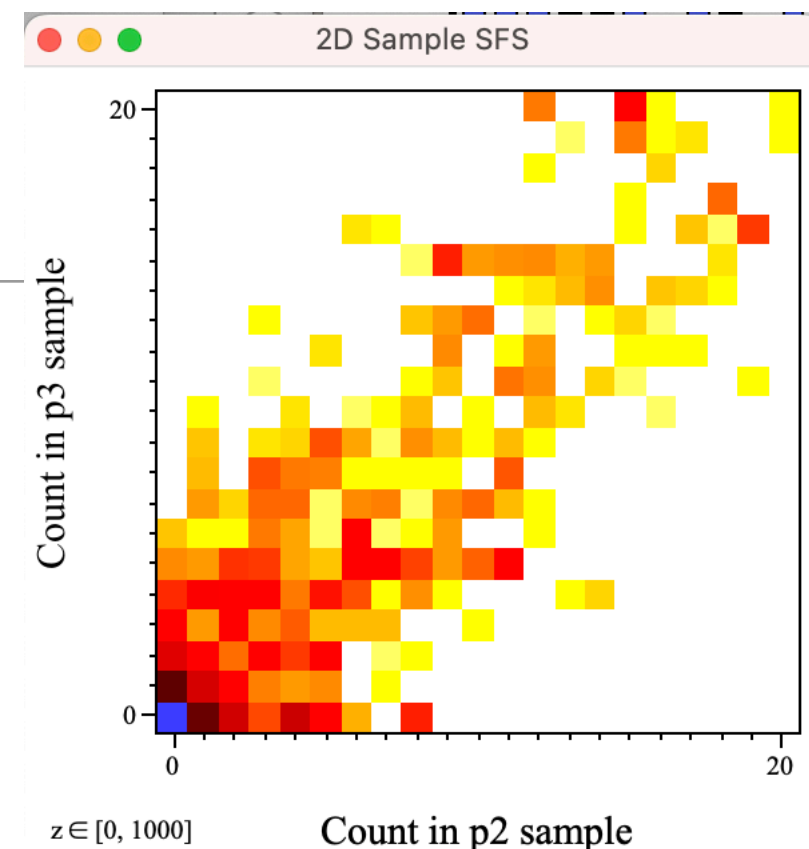
p1 p2 m1



p1 p3 m1

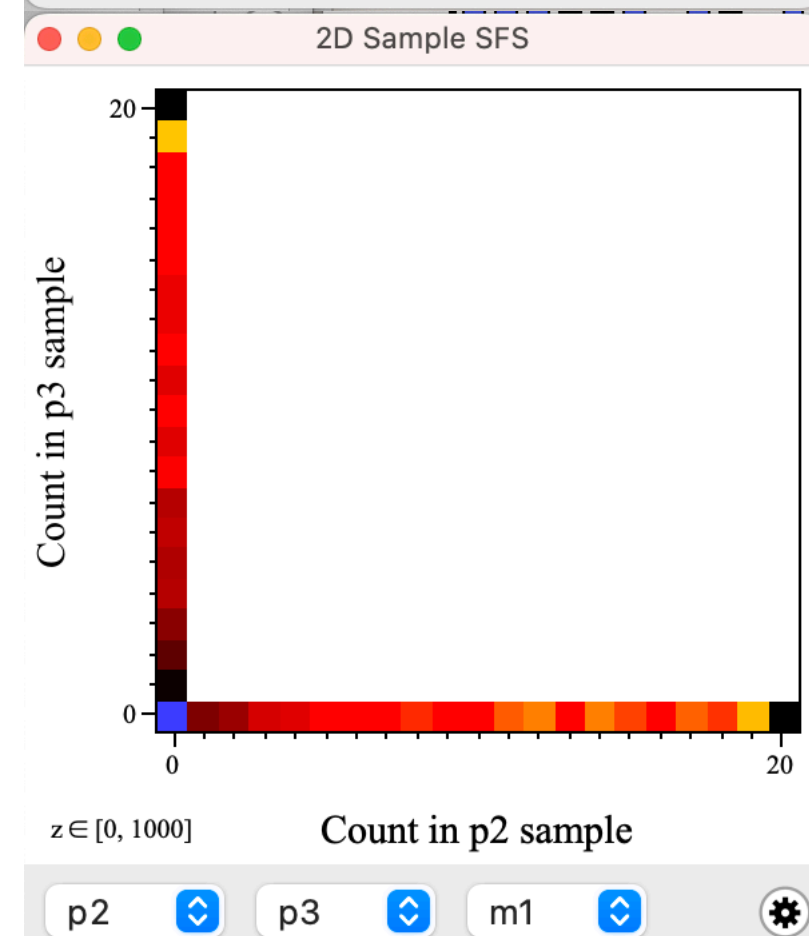
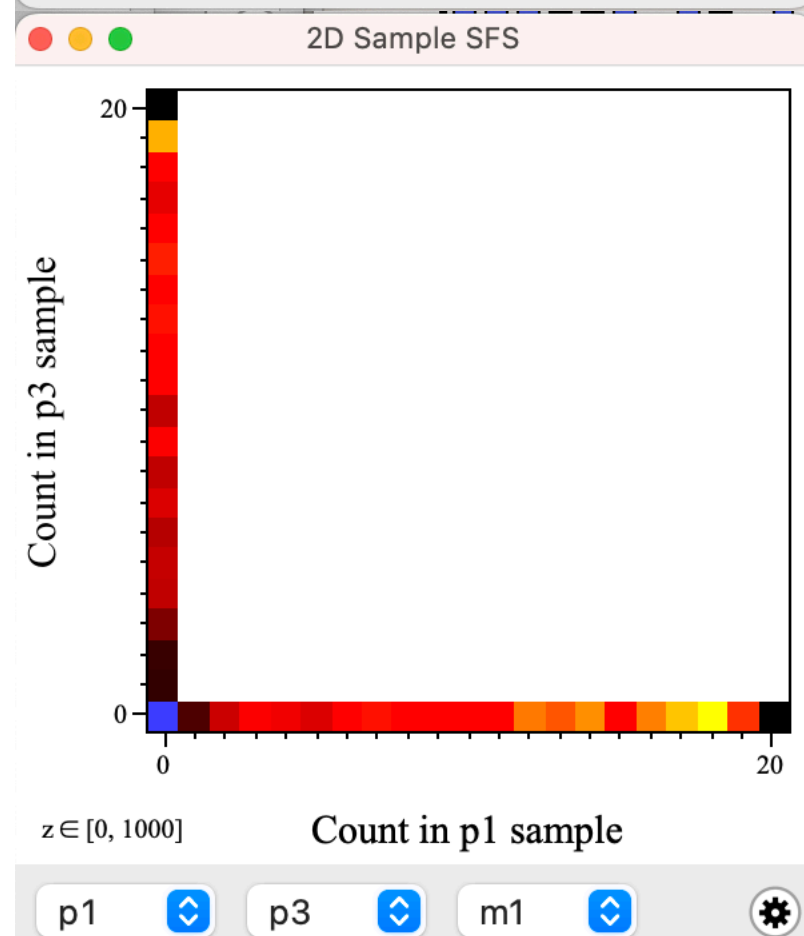
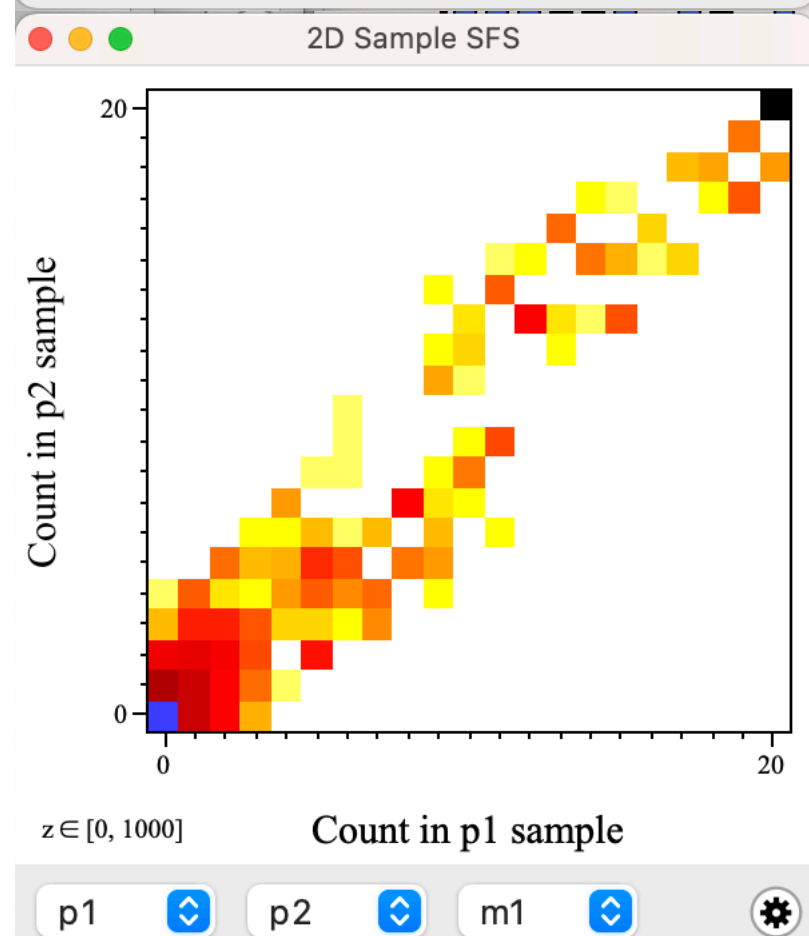
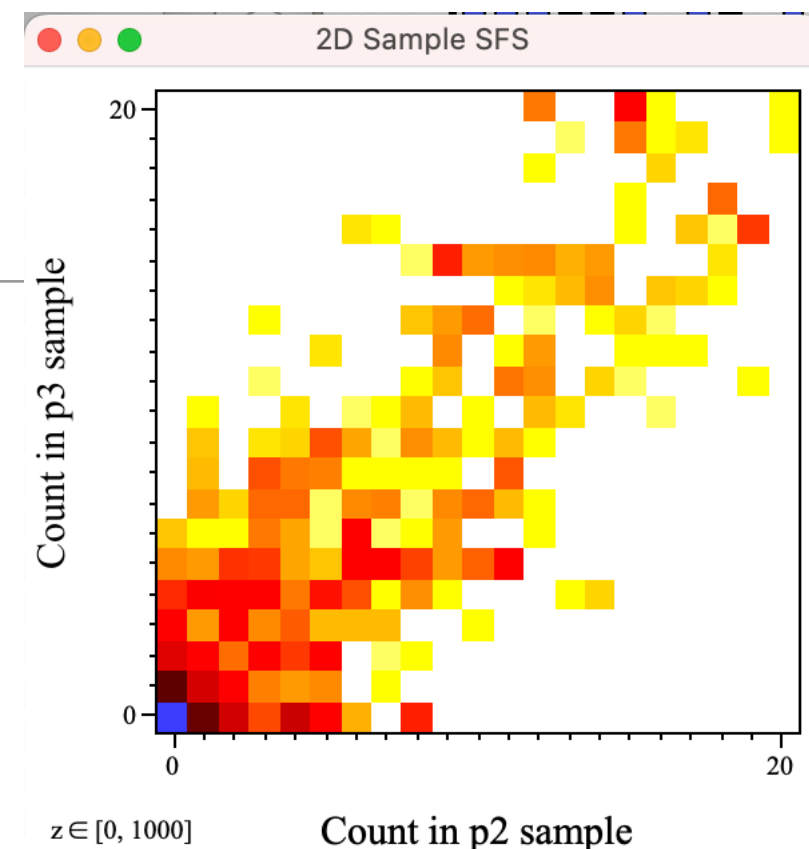
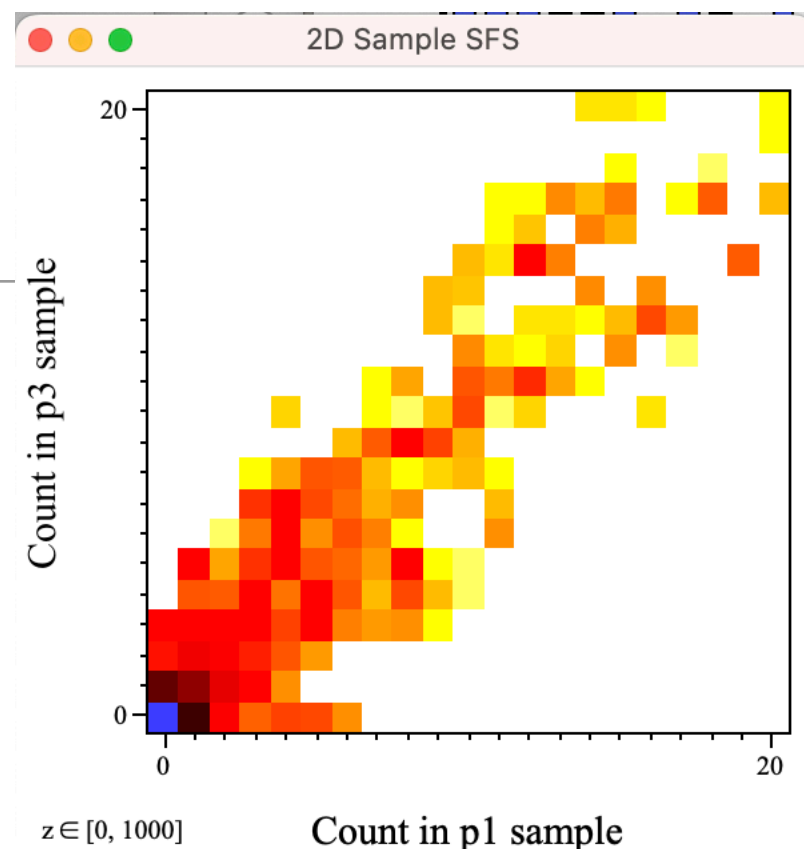
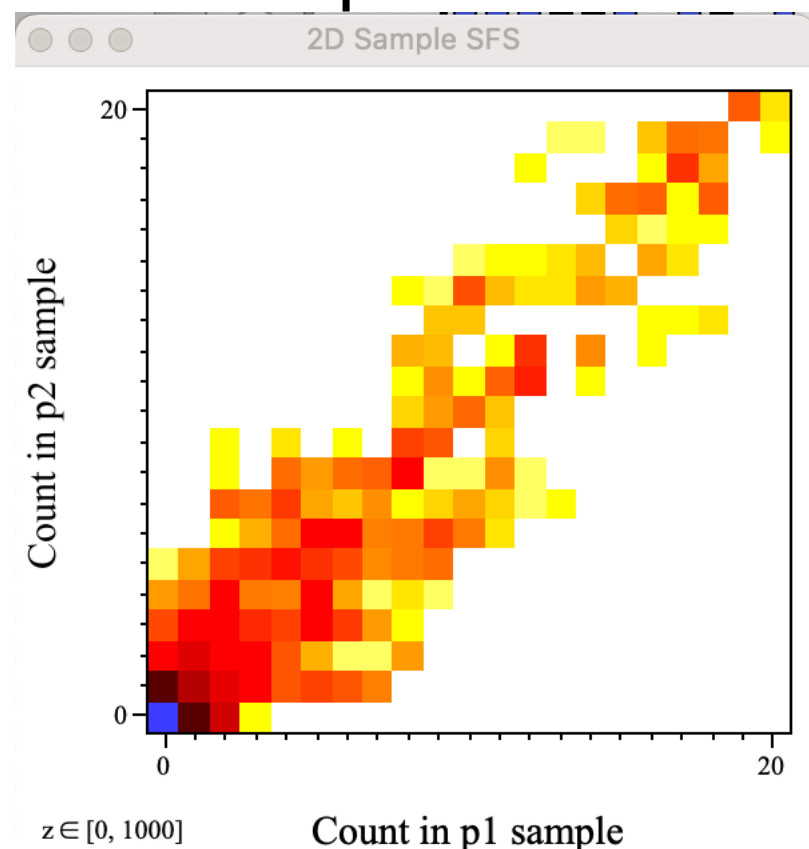


p1 p3 m1



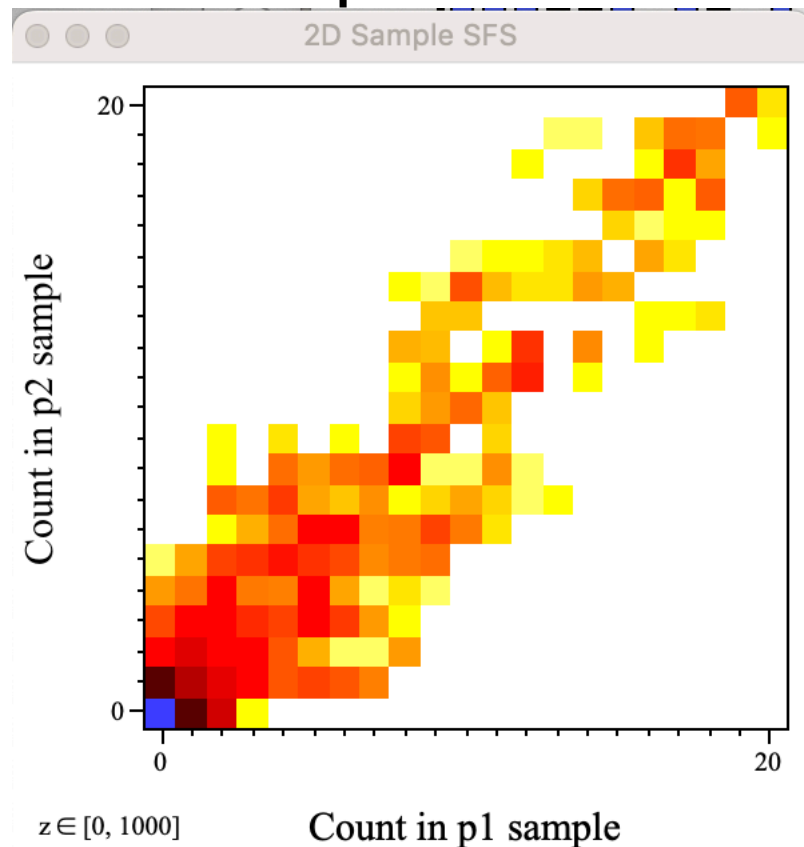
p2 p3 m1

# 2D Sample SFS

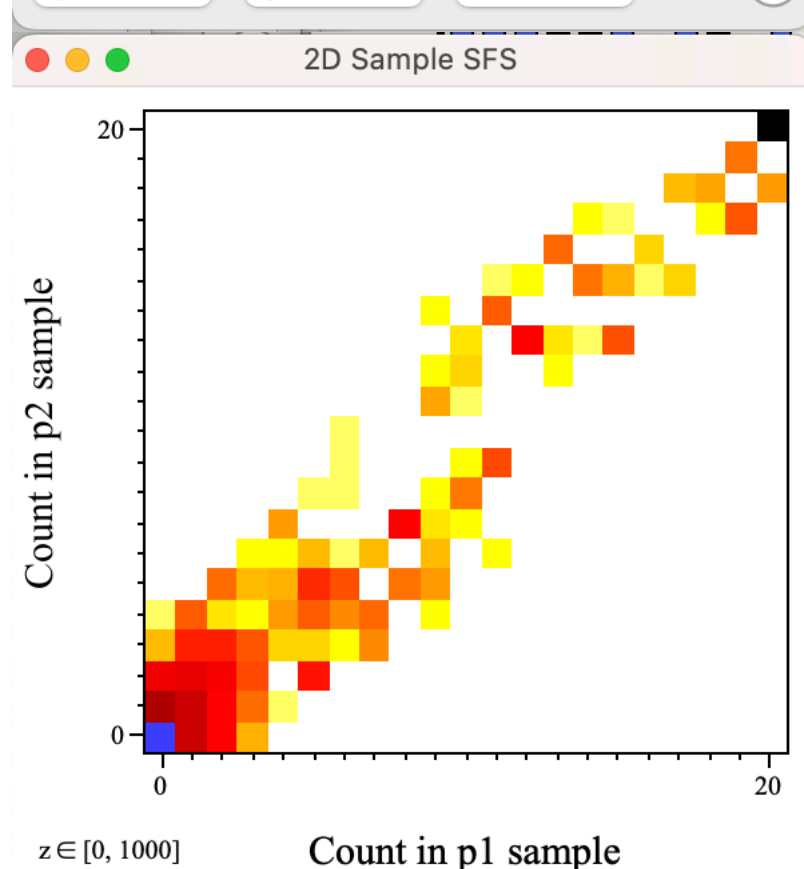


# 2D Sample SFS

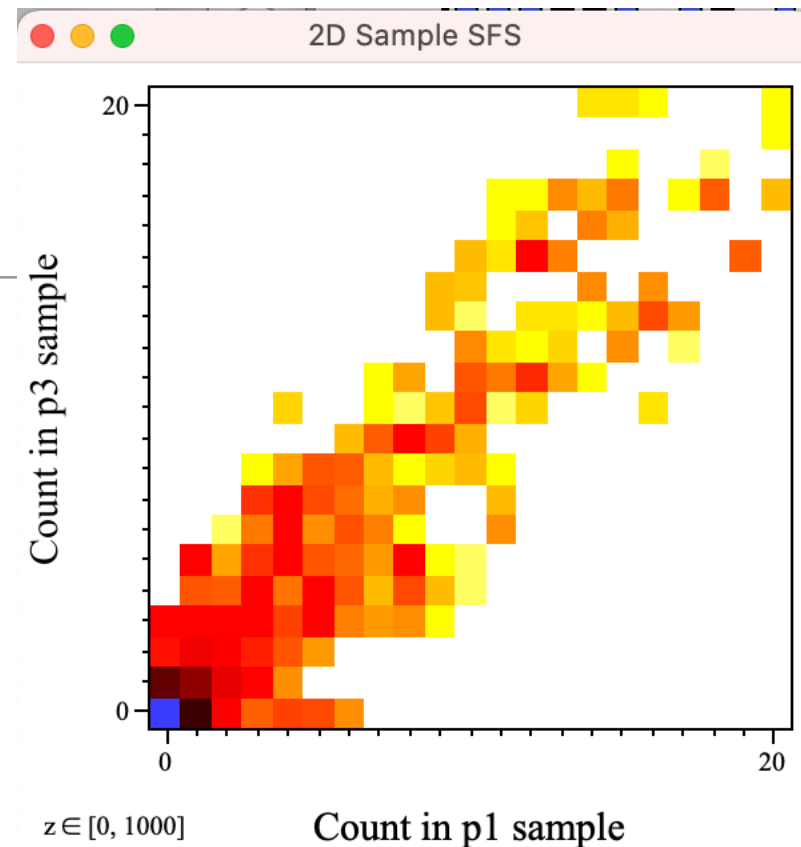
*What change did I make??*



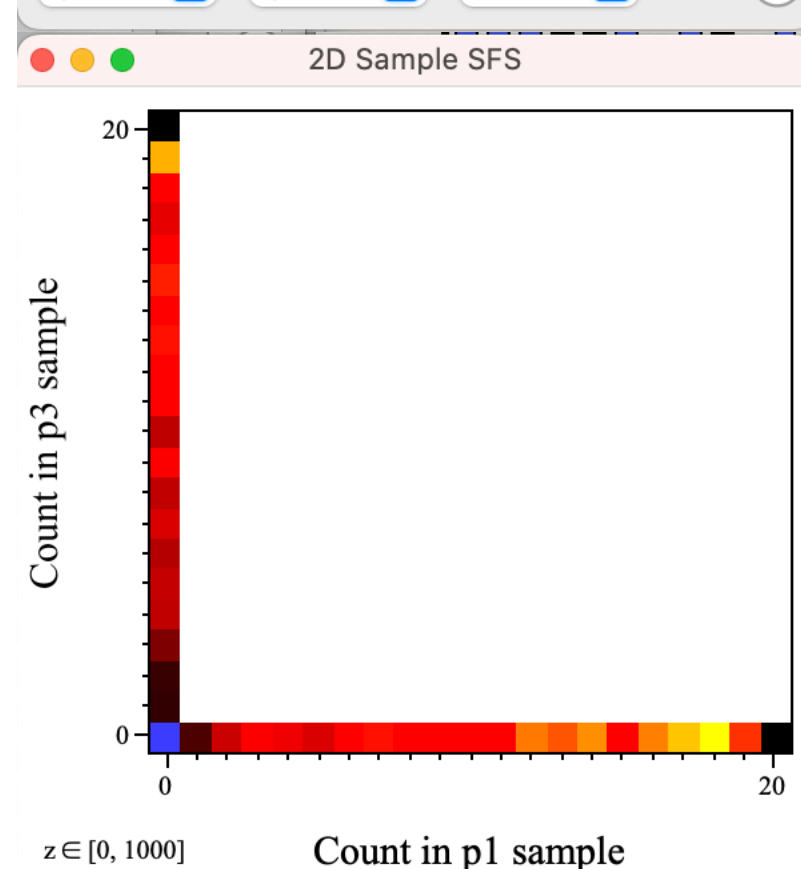
p1 p2 m1



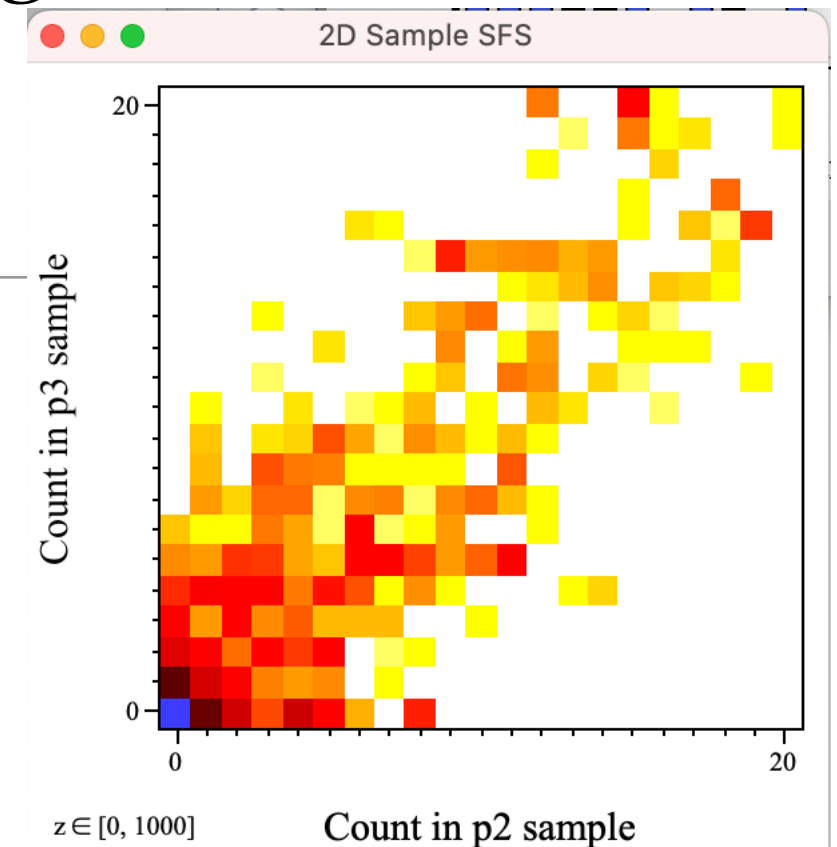
p1 p2 m1



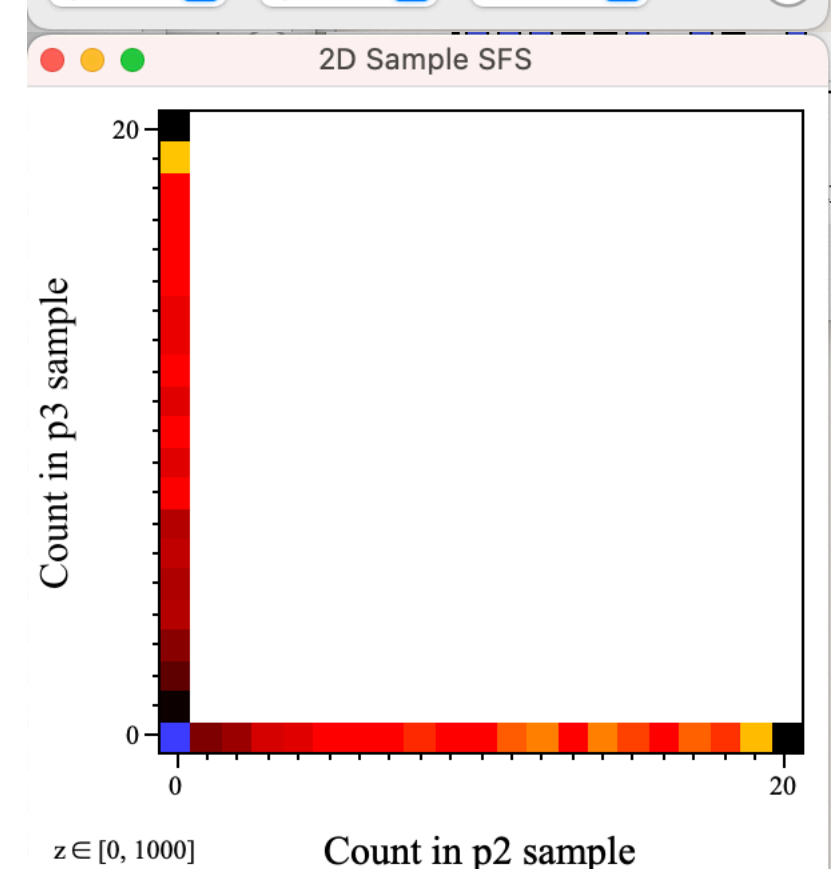
p1 p3 m1



p1 p3 m1



p2 p3 m1

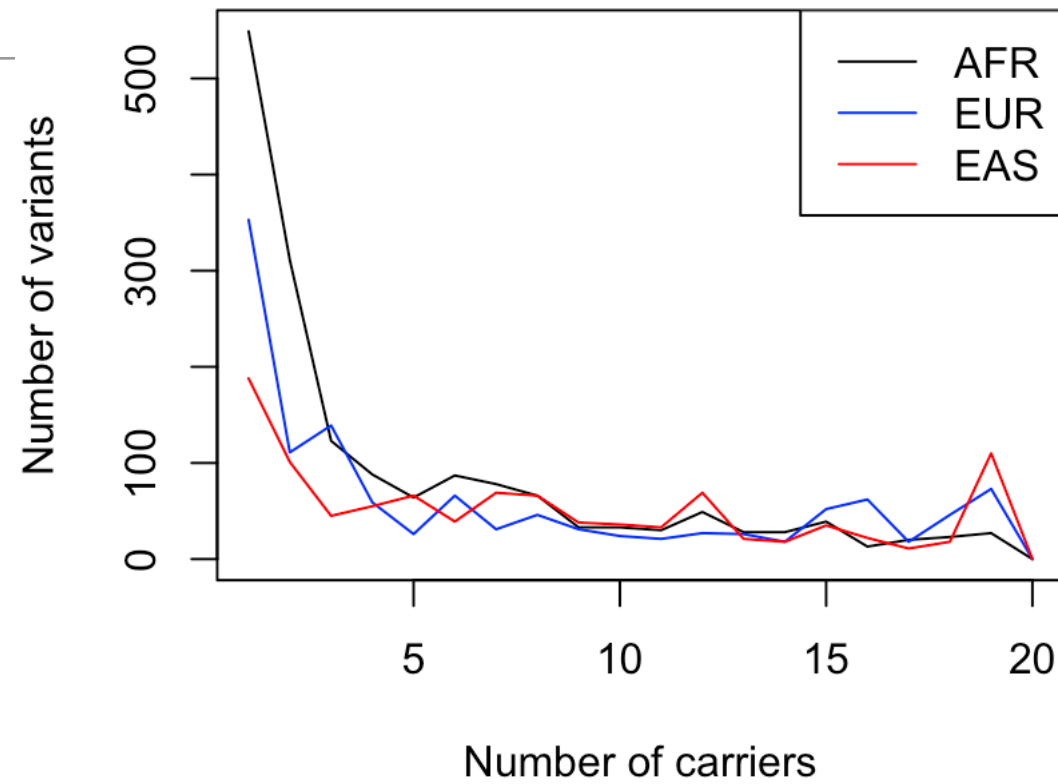


p2 p3 m1

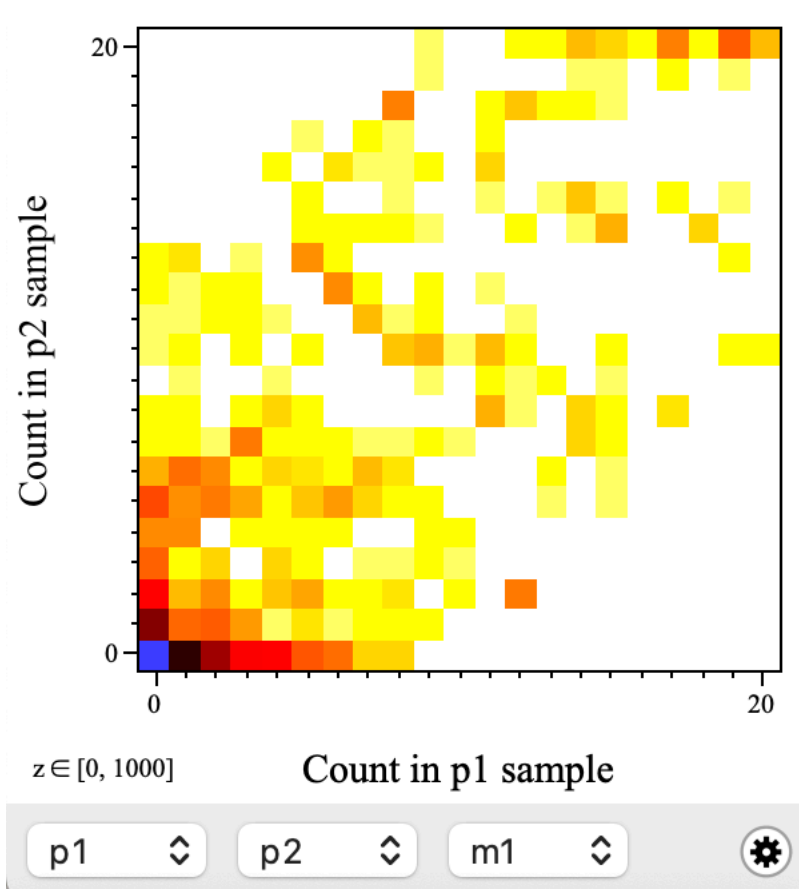
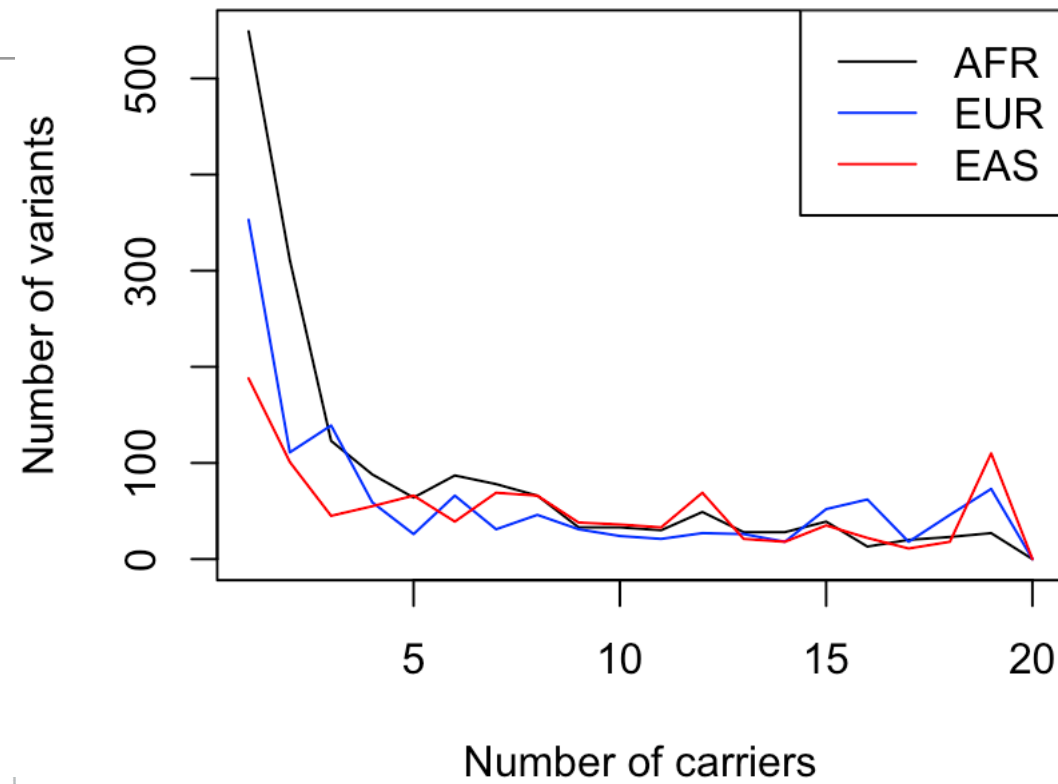
# Complex Human demography: Recipe 5.4 (I)

---

# Complex Human demography: Recipe 5.4 (I)

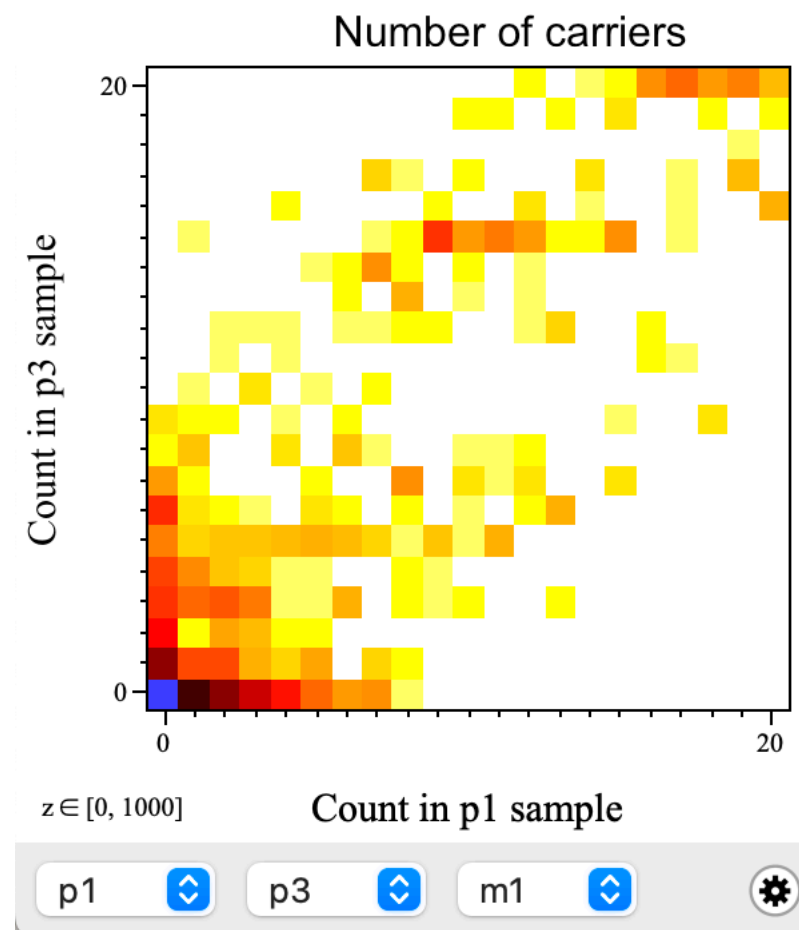
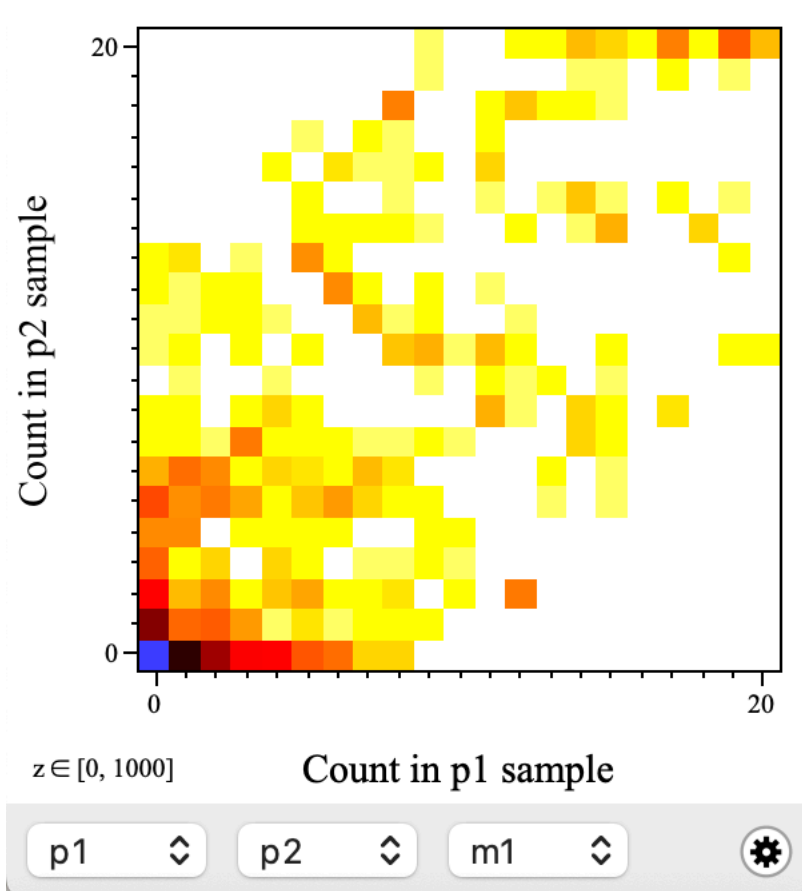
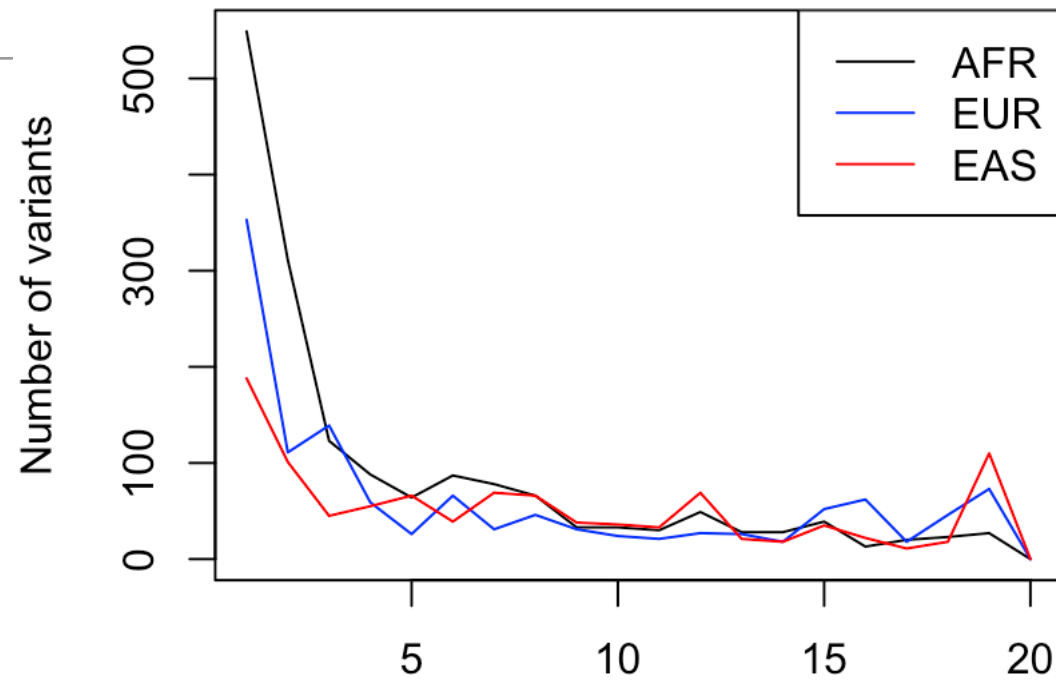


# Complex Human demography: Recipe 5.4 (I)

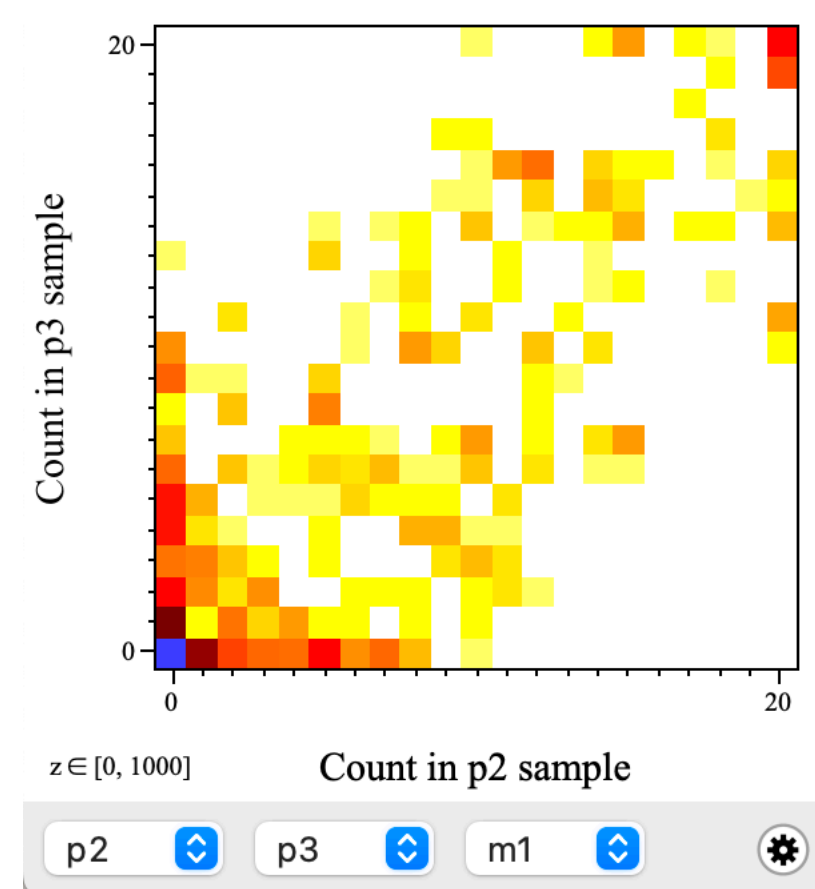
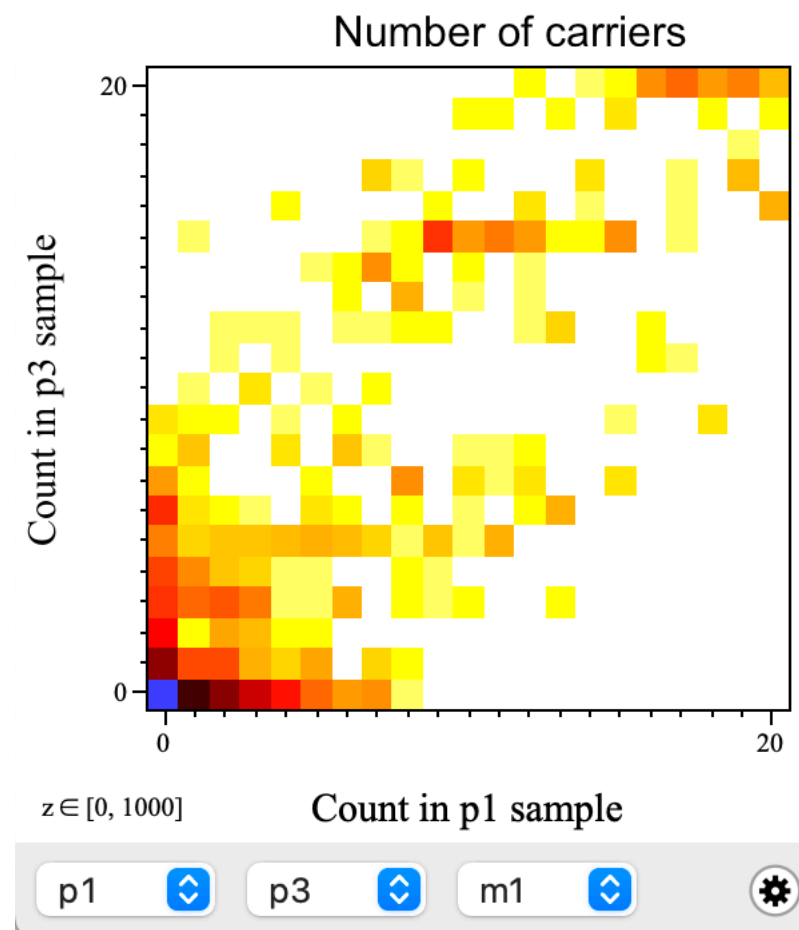
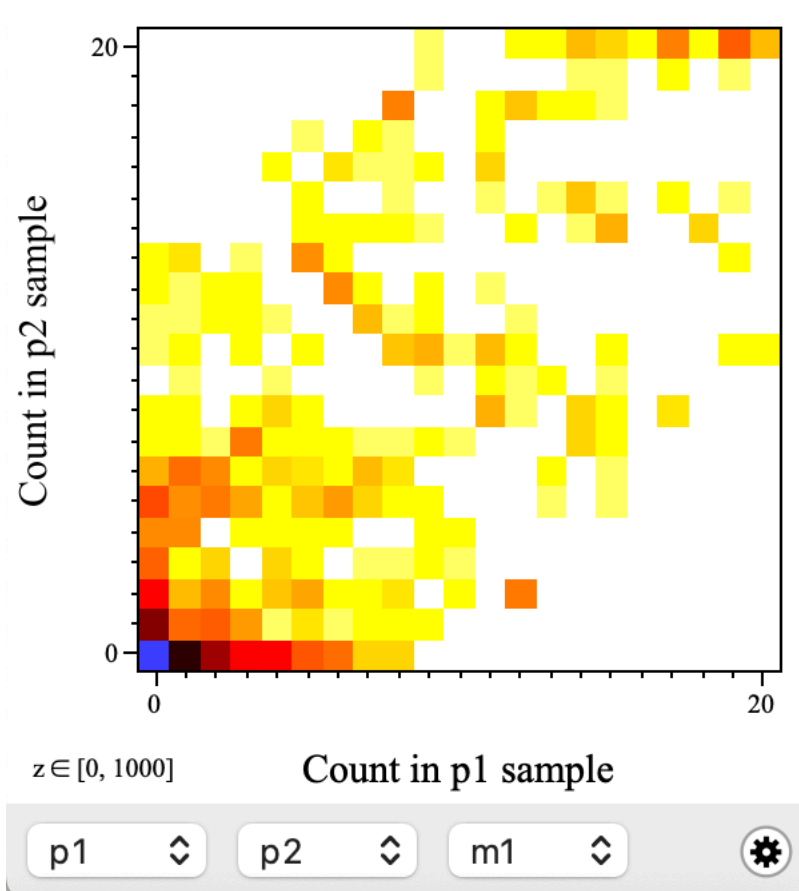
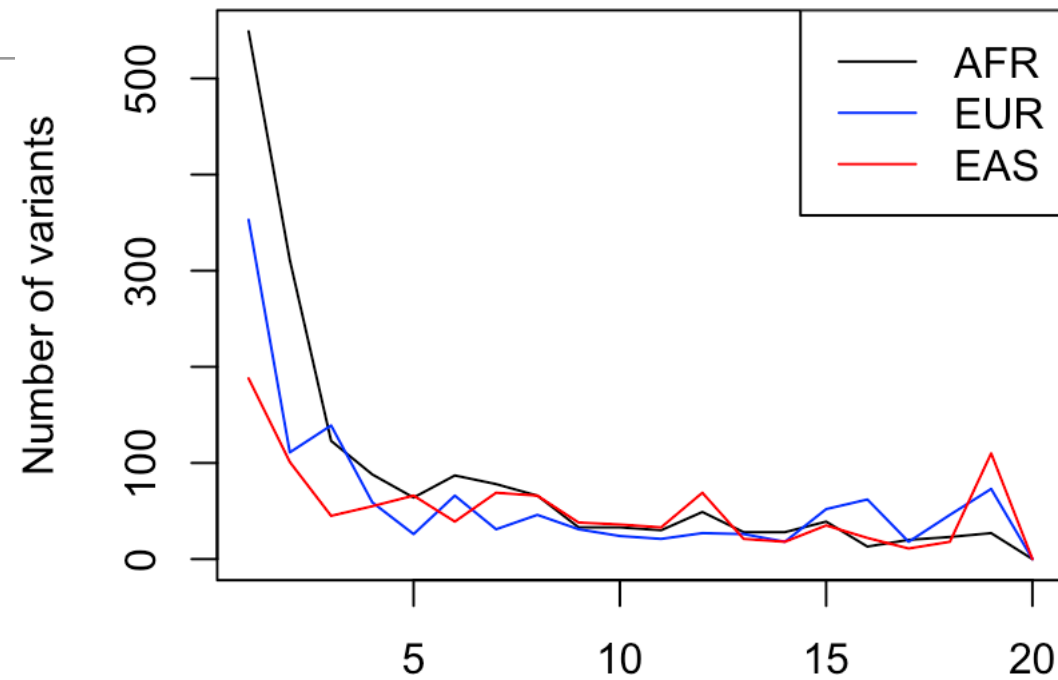




# Complex Human demography: Recipe 5.4 (I)



# Complex Human demography: Recipe 5.4 (I)



# Natural selection: Recipe 7.1

---

# Natural selection: Recipe 7.1

---

// Keywords:

```
initialize() {  
    initializeMutationRate(1e-7);  
    initializeMutationType("m1", 0.5, "f", 0.0);           // non-coding  
    initializeMutationType("m2", 0.5, "f", 0.0);           // synonymous  
    initializeMutationType("m3", 0.1, "g", -0.03, 0.2);     // deleterious  
    initializeMutationType("m4", 0.8, "e", 0.1);           // beneficial  
    initializeGenomicElementType("g1", m1, 1.0);  
    initializeGenomicElement(g1, 0, 99999);  
    initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

# Natural selection: Recipe 7.1

Mutation class 1;  
dominance coefficient = 0.5;  
“f”ixed selection coefficient (s)=0

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);           // non-coding  
  initializeMutationType("m2", 0.5, "f", 0.0);           // synonymous  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2);     // deleterious  
  initializeMutationType("m4", 0.8, "e", 0.1);           // beneficial  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

# Natural selection: Recipe 7.1

Mutation class 1;  
dominance coefficient = 0.5;  
“f”ixed selection coefficient ( $s$ )=0

Mutation class 3;  
dominance coefficient = 0.1;  
“g”amma distribution for  $s$ .

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeMutationType("m2", 0.5, "f", 0.0);           // synonymous  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2);    // deleterious  
  initializeMutationType("m4", 0.8, "e", 0.1);           // beneficial  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

# Natural selection: Recipe 7.1

Mutation class 1;  
dominance coefficient = 0.5;  
“f”ixed selection coefficient ( $s$ )=0

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);  
  initializeMutationType("m2", 0.5, "f", 0.0);  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2);  
  initializeMutationType("m4", 0.8, "e", 0.1);  
  initializeGenomicElementType("g1", m1, 1.0);  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

Mutation class 3;  
dominance coefficient = 0.1;  
“g”amma distribution for  $s$ .

Mutation class 4;  
dominance coefficient = 0.8;  
“e”xponential distribution  
for  $s$ .

# Natural selection: Recipe 7.1

---



# Natural selection: Recipe 7.1

---

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0);           // non-coding  
  initializeMutationType("m2", 0.5, "f", 0.0);           // synonymous  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2);     // deleterious  
  initializeMutationType("m4", 0.8, "e", 0.1);           // beneficial  
  initializeGenomicElementType("g1", c(m1,m2,m3,m4), c(0.05,0.2,0.74,0.01));  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

# Natural selection: Recipe 7.1

Genomic element 1 (g1) has all 4 mutation classes

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0); // non-coding  
  initializeMutationType("m2", 0.5, "f", 0.0); // synonymous  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2); // deleterious  
  initializeMutationType("m4", 0.8, "e", 0.1); // beneficial  
  initializeGenomicElementType("g1", c(m1,m2,m3,m4), c(0.05,0.2,0.74,0.01));  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

# Natural selection: Recipe 7.1

Genomic element 1 (g1) has all 4 mutation classes

// Keywords:

```
initialize() {  
  initializeMutationRate(1e-7);  
  initializeMutationType("m1", 0.5, "f", 0.0); // non-coding  
  initializeMutationType("m2", 0.5, "f", 0.0); // synonymous  
  initializeMutationType("m3", 0.1, "g", -0.03, 0.2); // deleterious  
  initializeMutationType("m4", 0.8, "e", 0.1); // beneficial  
  initializeGenomicElementType("g1", c(m1,m2,m3,m4), c(0.05,0.2,0.74,0.01));  
  initializeGenomicElement(g1, 0, 99999);  
  initializeRecombinationRate(1e-8);  
}  
1 early() { sim.addSubpop("p1", 5000); }  
10000 early() { sim.simulationFinished(); }
```

Relative proportion of mutations from each mutation class