Visualizing population genomic data across a landscape

Neha Savant

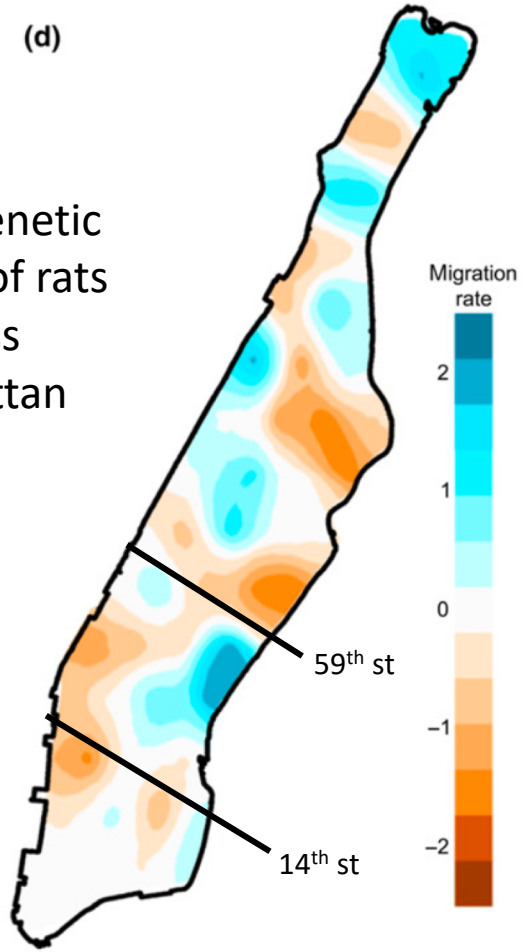
Monday, April 30, 2018

the “problem”

- More and more researchers are using genomics to answer conservation questions
- Need an easy way to communicate genetic patterns across a landscape to conservation practitioners and managers

the tools

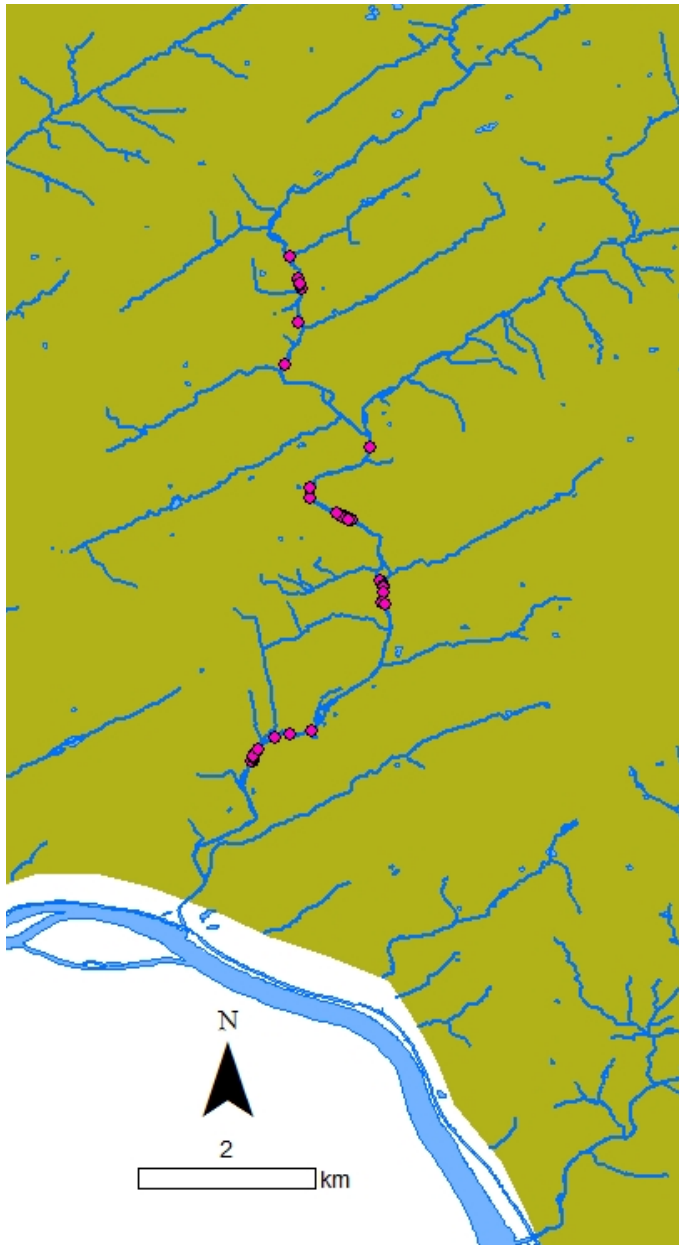
Spatial genetic patterns of rats across Manhattan



- Jupyter notebooks, ipython, R
- ipyrad, using the API and CLI
- EEMS package in R
  - Estimated Effective Migration Surfaces (EEMS) is a new method of using genomic data to illustrate barriers in the landscape (Petkova et al. 2016)

the data

- Case study
  - A threatened species lives near streams where a pipeline is planned to be built
  - Paired-end ddRAD data from 46 long-tail salamanders (*Eurycea longicauda*)
  - Need to visualize genetic structure of salamanders for state biologists to inform the permitting process



the goal

- Create a workflow in Jupyter notebooks that illustrates how to take raw genomic data and sampling locations to a visualization of genetic population structure across a landscape
- Can provide other researchers a roadmap to visualize their raw genomic data

progress

Step 1: Loading in de-multiplexed reads

Because I've already de-multiplexed my data and set the 'sorted_fastq_path', I just need to load in the files.

In [ ]:
```
ACAGTG1.run("1", ipyclient=ipyclient)
ACAGTG2.run("1", ipyclient=ipyclient)
ACAGTG3.run("1", ipyclient=ipyclient)

ATCACG1.run("1", ipyclient=ipyclient)
ATCACG2.run("1", ipyclient=ipyclient)
ATCACG3.run("1", ipyclient=ipyclient)

CGATGT1.run("1", ipyclient=ipyclient)
CGATGT2.run("1", ipyclient=ipyclient)
CGATGT3.run("1", ipyclient=ipyclient)

GCCAAT1.run("1", ipyclient=ipyclient)
GCCAAT2.run("1", ipyclient=ipyclient)
GCCAAT3.run("1", ipyclient=ipyclient)

TGACCA1.run("1", ipyclient=ipyclient)
TGACCA2.run("1", ipyclient=ipyclient)
TGACCA3.run("1", ipyclient=ipyclient)

TTAGGC1.run("1", ipyclient=ipyclient)
TTAGGC2.run("1", ipyclient=ipyclient)
TTAGGC3.run("1", ipyclient=ipyclient)
```

Step 1.5: Merging reads

The six libraries from each the three lanes can be merged before step 2. Because the three demultiplexed lanes each use the same barcodes file the samples will have identical names - ipyrad will recognize this during merging and read input files for

Step 2: Filtering using quality scores

Step 2 uses the quality score recorded in the fastQ data files to filter low quality base calls. Sites with a score below a set value are changed into “N”s (value is set by `max_Ns_consens` in Step 5), and reads with more than the number of allowed “N”s are discarded. The threshold for inclusion is set with the `phred_Qscore_offset` parameter. An optional filter can be applied to remove adapters/primers (see `filter_adapters`), and there is an optional filter to clean up the edges of poor quality reads (see `edit_cutsites`).

In [ ]:
```
lts_full_assembly.run("2", ipyclient=ipyclient) #runs step 2 with the fully merged assembly
```

Save the objects after Step 2 for future loading

In [ ]:
```
lts_full_assembly.save()
```

Loading assembly objects

When you run a `.run()` function, you will save all the directions to the results and outputs in a `.json` file. So we don't need to re-run the time-intensive steps we've already run, we load the `.json` objects that already exist using `load_json()` function.

In [ ]:
```
lts_full_assembly = ip.load_json("/rigel/edu/w4050/users/ngs2116/ipyrad/lts_full_assembly.json")
```

Step 3: Clustering within individuals

Step 3 first dereplicates the sequences from step 2, recording the number of times each unique read is observed. If the data are paired-end, which they are here, it then uses vsearch to merge paired reads which overlap. Since the data are going to be assembled denovo, the resulting data are de novo clustered using `vsearch`. If I were using a reference genome I would be

roadblocks

```
 -------------------------------------------------------------
  ipyrad [v.0.7.23]
  Interactive assembly and analysis of RAD-seq data
 -------------------------------------------------------------
  loading Assembly: lts_full_assembly
  from saved path: /rigel/edu/w4050/users/ngs2116/ipyrad/lts_full_assembly.json
  establishing parallel connection:
  host compute node: [24 cores] on node161

  Step 3: Clustering/Mapping reads
  [####################] 100%  dereplicating         | 0:44:49
  [#################### ]  99%  clustering            | 4 days, 23:07:57  slurmstepd: error: *** JOB 6184810 ON node161
CANCELLED AT 2018-04-30T08:56:26 DUE TO TIME LIMIT ***
```

next steps

- Finish running ipyrad
- Use R within jupyter notebooks to transform genomic data in proper format to run in EEMS
  - pairwise genetic dissimilarity matrix
  - Sample coordinates
  - Habitat coordinates