

ARGscape: A modular, interactive tool for manipulation of spatiotemporal ancestral recombination graphs

Christopher A. Talbot^{1,2,*} and Gideon S. Bradburd¹

¹Department of Ecology and Evolutionary Biology, University of Michigan, 1105 North University Avenue, Ann Arbor, MI 48109, USA

²Department of Computational Biology, Cornell University, 350 Tower Rd, Ithaca, NY 14853, USA

*Corresponding author: cat267@cornell.edu

Abstract

Summary: Ancestral recombination graphs (ARGs) encode the complete genealogical history of a population of recombining lineages. ARGs, and their succinct representation, tree sequences, are increasingly central to modern population genetics methods, yet building an intuition for ARGs remains challenging. This is particularly true when analyzing ancestry in a geographic context, as there is a critical lack of dedicated, interactive tools capable of visualizing ARGs as spatiotemporal objects. To address this gap, we introduce ARGscape, an interactive platform for simulating, analyzing, and visualizing ARGs across space and time. ARGscape provides a user-friendly graphical interface featuring dynamic 2- and 3-dimensional visualizations to explore ARGs through space and time, as well as a novel “spatial diff” visualization for quantitative comparison of geographic inference methods. ARGscape is an innovative, unified framework that seamlessly integrates leading command-line, Python, and R-based tools for

ARG simulation, manipulation, and use in spatiotemporal inference into both graphical and command-line interfaces. By integrating these various functionalities, ARGscape facilitates novel data exploration and hypothesis generation, while lowering the barrier to entry for spatiotemporal ARG analysis in both research and education use-cases.

Availability and Implementation: ARGscape is built with a Python FastAPI backend and a React/TypeScript frontend. It is freely available as a live demo at <https://www.argscape.com> [v0.3.0] and as a Python package on PyPI (`pip install argscape`) [v0.3.0]. The source code and documentation are available on GitHub at <https://github.com/chris-a-talbot/argscape>.

Contact: cat267@cornell.edu

Supplementary information: Supplementary Information is included at the bottom of this document.

1 Introduction

The study of genetic variation across time and space is fundamental to population genetics, providing insights into processes from disease outbreaks to species evolution [Beebe et al., 2013, Breistein et al., 2022, Rehmann et al., 2025]. The record of these processes can be read from the Ancestral Recombination Graph (ARG), which traces the complete genetic ancestry, including recombination and coalescent events, for a set of samples [Rasmussen et al., 2014, Lewanski et al., 2024, Brandt et al., 2024, Nielsen et al., 2025]. These structures can thereby provide unprecedented clarity in studies of genetic variation across continuous space and time, moving beyond discrete populations and timepoints and toward a representation of real-world evolutionary processes. Recent developments in ARG inference methods have

made high-quality ARGs accessible for a wide range of datasets of varying size and quality [Kelleher et al., 2019, Speidel et al., 2019, Hubisz et al., 2020, Wohns et al., 2022, Zhang et al., 2023, Deng et al., 2024]. In particular, the succinct tree sequence data structure, implemented by `tskit`, has made the storage and analysis of very large ARGs computationally tractable [Kelleher et al., 2016, Ralph et al., 2020, Wong et al., 2024].

Consequently, ARGs are now central to a growing number of sophisticated inference methods [Hejase et al., 2022, Wohns et al., 2022, Tagami et al., 2024, Grundler et al., 2025]. In particular, a growing area of research is the use of ARGs to infer the spatial history of ancestors of a population [Wohns et al., 2022, Osmond & Coop, 2024, Grundler et al., 2025, Deraje et al., 2025]. However, despite their theoretical power, a broadly accessible and unified framework for interpreting ARGs as spatiotemporal objects remains elusive. This gap hinders their adoption by the broader genomics community. While existing tools (such as `tsbrowse` and `tskit_arg_visualizer`, [Karthikeyan et al., 2025, Kitchens & Wong, 2025]) provide powerful interfaces for inspecting the raw topology of and data stored in tree sequences and ARGs, they are not designed for the integrated spatial inference and visualization that is critical for many ecological and evolutionary questions.

To bridge this gap and enhance the accessibility of ARGs, we present ARGscape: an interactive platform for visualizing, exploring, and analyzing ARGs as spatiotemporal records of evolutionary history. ARGscape integrates simulation, analysis, and visualization into a single, intuitive workflow, making complex spatiotemporal analyses accessible to both researchers and students.

2 System and Features

ARGscape is a full-stack web application with a Python-based FastAPI backend and a React/TypeScript-based frontend, available as a publicly hosted website or a Python package for local deployment. Its workflow is designed to be modular and intuitive, guiding the user from data input to visualization and export. The backend design is intended to facilitate rapid integration of new algorithms and features, ensuring that developments in the field can be incorporated and made accessible quickly. The ARGscape Python package also features a command-line interface incorporating core functionality without the need to load the full web application.

2.1 Data Input and Simulation

Users can begin by either uploading existing tree sequences in `.trees` or compressed `.tsz` format or by simulating new ones. Custom spatial coordinates for nodes in the tree sequence can be supplied via `.csv` files. The simulation module uses `msprime` to generate tree sequences under a specified coalescent model [Kelleher et al., 2016, Nelson et al., 2020, Baumdicker et al., 2022]. For demonstrative purposes, ARGscape uses a multidimensional scaling algorithm on the genealogical distance matrix to produce “toy” spatial coordinates for simulated samples, which can be generated on a 2-dimensional grid or projected onto real-world maps.

For command-line interface users, tree sequences can be loaded and stored in temporary memory using the `argscape_load` command.

2.2 Spatiotemporal Inference

A core function of ARGscape is to provide a cohesive analytical environment that integrates multiple complex spatiotemporal inference tools, enabling direct comparison and hypothesis testing within a single interface. Once a tree sequence is loaded, users can perform:

- **Temporal Inference:** ARGscape supports the full tsdate workflow, including ARG pre-processing and temporal inference [Speidel et al., 2021].
- **Spatial Inference:** Given sample locations in continuous 1- or 2-dimensional space, ARGscape enables rapid inference of ancestral node locations in geographic space. Geographic inference may be performed using any of the following models:
 - **Wohns midpoint** [Wohns et al., 2022]
 - **Gaia** [quadratic or linear algorithms] [Grundler et al., 2025]
 - **FastGaia** (see Supplementary Information, Section S2)
 - **Sparg** [Deraje et al., 2025]

These analyses are initiated with simple button clicks in the user interface, abstracting away the need for complex command-line syntax and separate data formatting for each approach. ARGscape also offers static and interactive command-line methods for manipulating tree sequences, including persistent storage of tree sequences and the full suite of spatiotemporal inference methods, using the `argscape_infer` command.

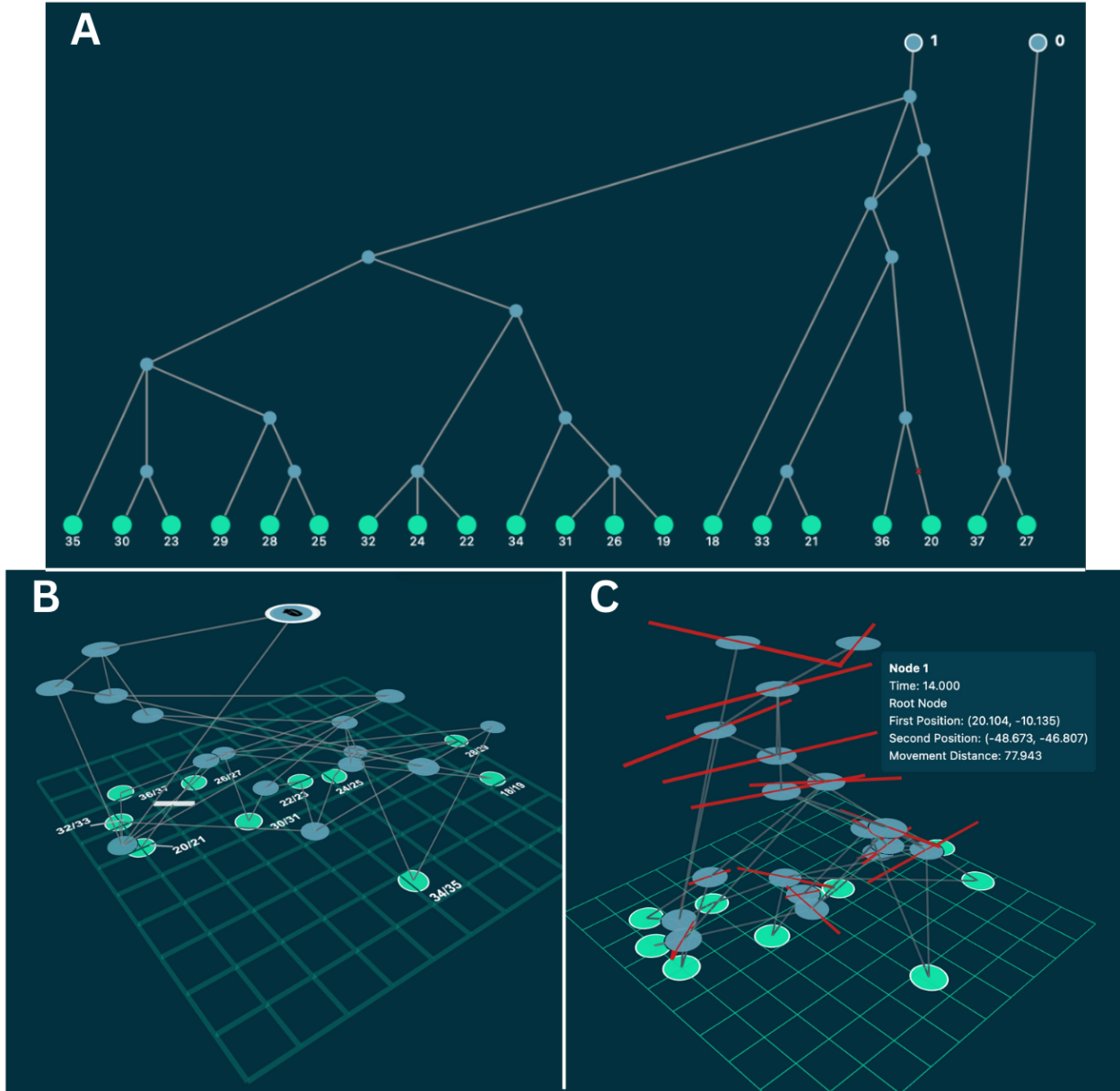


Figure 1: Visualizations of an ancestral recombination graph (ARG) generated by a spatially explicit evolutionary simulation in SLiM 5.0. **A)** A 2D visualization, displaying the topology of the ARG. Our dagre-d3 mode generates this node ordering and layout. Green nodes represent samples, and blue nodes represent ancestral nodes. Blue nodes with a white outline are roots of the ARG. Branches with mutations along them are marked with a red “x.” Generations are evenly spaced along the y-axis. In ARGscape, node locations may be adjusted, node and edge labels toggled, and colors modified. Nodes may be selected to display parent or child subARGs. **B)** A 3D visualization of the same ARG with ground truth spatial data from the simulation. Green nodes represent samples, and blue nodes represent ancestral nodes. Blue nodes with a white outline are roots of the ARG. Branches with mutations along them are marked with white bars. Generations are evenly spaced along the z-axis. Sample nodes from the same individual are labeled together because they originate from the same location. In ARGscape, spacing may be adjusted, node and edge labels toggled, and colors modified. Nodes may be selected to display parent or child subARGs. **C)** A 3D “spatial diff” visualization. Ancestral locations along the same ARG were re-inferred using the Gaia quadratic algorithm in ARGscape. Nodes are placed at the midpoint of the simulated and inferred locations, with red bars connecting the ground truth and inferred locations. This can be used to visualize the quantity and directionality of error in an inference method against ground truth; the increase in error moving back through time is clearly visible. Hovering over nodes reveals metadata, including their geographic locations in each version of the ARG and the distance between them.

2.3 Interactive Visualization

ARGscape offers multiple visualization styles, each featuring extensive interactivity, customization, and analysis capabilities. Before visualizing with any method, the user may select a subset of the sample nodes, genomic regions, or temporal regions to visualize. This is particularly useful for ARGs that extend deep in time, for which visualizing only recent ancestry may be most informative. Across all visualizations, recombination nodes are combined into single nodes with multiple node IDs, which enhances intuitive visualization and increases rendering speed. ARGscape offers three primary visualization modes for exploring tree sequences:

1. **2D Non-spatial View:** A force-directed graph rendered using D3.js displays the topological structure of the ARG. The force-directed graph visualization is fully customizable, featuring drag-and-drop node placement, toggleable node and edge labels, and a customizable color theme. To maximize interpretability, users can arrange sample nodes (tips) using multiple ordering algorithms specifically designed for tree sequences, including a novel “consensus” method that aggregates node orders across multiple local trees. Internal nodes and edges are then placed using a force simulation to ensure adequate spacing for visual clarity. Additionally, the dagre—d3 mode uses the dagre—d3 React library to place all nodes and edges using a path-crossing minimization algorithm. Users can interactively explore the graph by clicking nodes to view sub-ARGs or trace ancestral lineages.
2. **3D Spatial View:** A dynamic 3D plot rendered with deck.gl visualizes the ARG in geographic space, with time on the z-axis. The 3D spatial visualization is highly

customizable, featuring toggleable node and edge labels, as well as a customizable color theme. The tool automatically detects the coordinate reference system (CRS) to display nodes on a unit grid or a world map. Users can upload custom shapefiles for bespoke geographic contexts. Interactive sliders for genomic and temporal windows enable powerful data exploration, allowing users to isolate and visualize how specific segments of ancestry have moved across space over time. Sample nodes that occur in identical locations and are $+/-1$ node ID away from each other are assumed to represent haplotypes from the same individual, and are therefore combined into a single node for clarity. This specialized visualization mode opens doors for novel hypothesis generation and data exploration, including the visual assessment of coalescent rate through both time and space.

3. **3D Spatial “Diff” View:** A dynamic 3D plot rendered with deck.gl visualizes the difference in spatial locations on nodes in two ARGs with otherwise identical structure. Nodes are placed at the midpoint of their locations in each of two different ARGs, with a bar indicating the difference in locations between ARGs. This tool is one-of-a-kind and highly useful for comparing inferred ancestral locations with simulated ground truth, or comparing inferred locations between multiple different inference methods.

2.4 Accessibility and Data Export

To cater to different user needs, ARGscape is available as both a public web application for easy access and testing, as well as a Python package for integration into local pipelines and resource-intensive use cases. Users can download any modified tree sequence (e.g.,

after spatial/temporal inference) in standard `.trees` or `.tsz` formats for further downstream analysis. Publication-quality images of any visualization can also be exported, facilitating integration into manuscripts, presentations, and educational materials.

3 Conclusion

ARGscape provides a much-needed, user-friendly platform that integrates the simulation, analysis, and spatiotemporal visualization of ancestral recombination graphs. By integrating powerful but complex inference tools in an intuitive interface and providing novel interactive visualizations, it significantly lowers the barrier to entry for researchers and students interested in exploring the full richness of ARGs. ARGscape simplifies the process of asking and answering questions about how ancestry unfolds across the genome over time and across geographic space. Future directions will focus on enhancing performance for BioBank-scale datasets and expanding the tool’s analytical capabilities to visualize key demographic processes through time, including migration and admixture. We also aim to further develop its utility as an interactive learning platform for population genetics education by incorporating guided tutorials and visual-based lessons.

4 Acknowledgments

The authors thank the members of the Bradburd Lab at the University of Michigan, the Messer Lab at Cornell University, the `tskit-dev` team, and James Kitchens for their feedback and testing.

5 Author contributions

C.A.T. (Conceptualization [lead], Software [lead], Validation [lead], Writing – original draft [lead], Writing – review & editing [equal]). G.S.B. (Supervision [lead], Funding acquisition [lead], Conceptualization [supporting], Writing – original draft [supporting], Writing – review & editing [equal]).

6 Funding

This work was supported by the National Institutes of Health [grant R35-GM137919] to G.S.B.

7 Data availability

Source code and documentation are available at <https://github.com/chris-a-talbot/argscape>. Figure 1 uses a tree sequence generated by a spatially explicit simulation ran in SLiM 5.0 [Haller et al., 2025]. The tree sequence and SLiM simulation script used to generate visualizations for Figure 1 are available on Zenodo under doi [10.5281/zenodo.17296543](https://doi.org/10.5281/zenodo.17296543).

8 Conflict of interest

None declared.

References

- Baumdicker, F, Bisschop, G, Goldstein, D, et al. Efficient ancestry and mutation simulation with Msprime 1.0. *Genetics* 2022; 220: iyab229.
- Beebe, NW, Ambrose, L, Hill, LA, et al. Tracing the tiger: population genetics provides valuable insights into the Aedes (Stegomyia) albopictus invasion of the Australasian Region. *PLoS Negl Trop Dis* 2013, doi: 10.1371/journal.pntd.0002361
- Brandt, DYC, Huber, CD, Chiang, CWK, et al. The promise of inferring the past using the Ancestral Recombination Graph. *Genome Biol Evol* 2024; 16: evae005.
- Breistein, B, Dahle, G, Johansen, T, et al. Geographic variation in gene flow from a genetically distinct migratory ecotype drives population genetic structure of coastal Atlantic cod (Gadus morhua L.). *Evo Applications* 2022, doi: 10.1111/eva.13422
- Deng, Y, Nielsen, R, Song, YS. Robust and accurate Bayesian inference of genome-wide genealogies for large samples. *bioRxiv* 2024, doi: 10.1101/2024.03.16.585351.
- Deraje, P, Kitchens, J, Coop, G, et al. The Promise and Challenge of Spatial Inference with the Full Ancestral Recombination Graph under Brownian Motion. *bioRxiv* 2025, doi: 10.1101/2024.04.10.588900.
- Grundler, MC, Terhorst, J, Bradburd, GS. A geographic history of human genetic ancestry. *Science* 2025; 387: 1391–1397.
- Haller, BC, Ralph, PL, Messer, PW. SLiM 5: Eco-evolutionary simulations across multiple chromosomes and full genomes. *bioRxiv* 2025; doi: 10.1101/2025.08.07.669155.

- Hejase, HA, Mo, Z, Campagna, L, et al. A deep-learning approach for inference of selective sweeps from the Ancestral Recombination Graph. *Mol Biol Evol* 2022; 39: msab332.
- Hubisz, MJ, Williams, AL, Siepel, A. Mapping gene flow between ancient hominins through demography-aware inference of the Ancestral Recombination Graph. *PLoS Genet* 2020; 16: e1008895.
- Karthikeyan, S, Jeffery, B, Mbuli-Robertson, D, et al. Tsbrowse: An interactive browser for Ancestral Recombination Graphs. *bioRxiv* 2025, doi: 10.1101/2025.04.23.649987.
- Kelleher, J, Etheridge, AM, McVean, G. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Comput Biol* 2016; 12: e1004842.
- Kelleher, J, Wong, Y, Wohns, AW, et al. Inferring whole-genome histories in large population datasets. *Nat Genet* 2019; 51: 1330–1338.
- Kitchens, J, Wong, Y. tskit_arg_visualizer: interactive plotting of ancestral recombination graphs. *arXiv* 2025; doi: arXiv:2508.03958
- Lewanski, AL, Grundler, MC, Bradburd, GS. The era of the ARG: An introduction to Ancestral Recombination Graphs and their significance in empirical evolutionary genomics. *PLoS Genet* 2024; 20: e1011110.
- Nelson, D, Kelleher, J, Ragsdale, AP, et al. Accounting for long-range correlations in genome-wide simulations of large cohorts. *PLoS Genet* 2020; 16: e1008619.
- Nielsen, R, Vaughn, AH, Deng, Y. Inference and applications of Ancestral Recombination Graphs. *Nat Rev Genet* 2025; 26: 47–58.

- Osmond, M, Coop, G. Estimating dispersal rates and locating genetic ancestors with genome-wide genealogies. *eLife* 2024; doi: 10.7554/eLife.72177
- Ralph, P, Thornton, K, Kelleher, J. Efficiently summarizing relationships in large samples: A general duality between statistics of genealogies and genomes. *Genetics* 2020; 215: 779–797.
- Rasmussen, MD, Hubisz, MJ, Gronau, I, et al. Genome-wide inference of Ancestral Recombination Graphs. *PLoS Genet* 2014; 10: e1004342.
- Rehmann, CT, Small, ST, Ralph, PL, et al. Sweeps in Space: Leveraging Geographic Data to Identify Beneficial Alleles in *Anopheles gambiae*. *Mol Biol Evol* 2025; doi: 10.1093/molbev/msaf141
- Speidel, L, Forest, M, Shi, S, et al. A method for genome-wide genealogy estimation for thousands of samples. *Nat Genet* 2019; 51: 1321–1329.
- Speidel, L, Cassidy, L, Davies, RW, et al. Inferring population histories for ancient genomes using genome-wide genealogies. *Mol Biol Evol* 2021; 38: 3497–3511.
- Tagami, D, Bisschop, G, Kelleher, J. Tstrait: A quantitative trait simulator for Ancestral Recombination Graphs. *Bioinformatics* 2024; 40: btae334.
- Wohns, AW, Wong, Y, Jeffery, B, et al. A unified genealogy of modern and ancient genomes. *Science* 2022; 375: eabi8264.
- Wong, Y, Ignatieva, A, Koskela, J, et al. A general and efficient representation of Ancestral Recombination Graphs. *Genetics* 2024; 228: iyae100.

Zhang, BC, Biddanda, A, Gunnarsson, ÁF, et al. Biobank-scale inference of Ancestral Recombination Graphs enables genealogical analysis of complex traits. *Nat Genet* 2023; 55: 768–776.

Supplementary Information

Supplementary Information is included at the bottom of this document.

End of Primary Article

(Next: Supplementary Information)

Supplementary Information

ARGscape: A modular, interactive tool for manipulation of spatiotemporal ancestral recombination graphs

Christopher A. Talbot^{1,2,*} and Gideon S. Bradburd¹

¹Department of Ecology and Evolutionary Biology, University of Michigan, Ann Arbor, MI, USA

²Department of Computational Biology, Cornell University, Ithaca, NY, USA

*Corresponding author: cat267@cornell.edu

S1 Installation & Usage

S1.1 Using the Hosted Web Application

The latest production version of ARGscape is available at <https://www.argscape.com/>. The interactive interface guides users through the various modes of use and functionality, including file management, simulation, visualization, inference, and output downloading. Files stored on the hosted web server will be stored privately and securely for up to 24 hours. The web server storage is subject to being cleared, without notice, during updates.

We recommend using the hosted application only for **educational** and/or **exploratory** purposes due to the potential for data loss and the limited computational capabilities of our web server. Users are limited to a small amount of computing power for simulation, inference, and visualization, which will be insufficient for most full-scale projects.

S1.2 Installing ARGscape Locally

The latest production version of **ARGscape** is available as a Python package hosted on the Python Package Index (PyPI). To install, users must have Python version 3.8 or later. Then, from a terminal in which Python is available, users should run the following commands:

```
> python -m pip install --upgrade pip  
  
> python -m pip install argscape
```

ARGscape will be installed in the Python environment.

S1.3 Running the Web Application Locally

To run the web application hosted on a local machine, first see Supplemental Information 1.2 – Installing **ARGscape** Locally. Once **ARGscape** is installed, the web application can be started from any terminal in which Python is available and **ARGscape** is installed using the following command:

```
> argscape
```

This command will launch the complete **ARGscape** web application in a browser window, hosted on your local machine. For additional options, including customizing the port **ARGscape** is hosted on and disabling certain features, run:

```
> argscape --help
```

S1.4 Using the Command-Line Tools

To run the web application hosted on a local machine, first see Supplemental Information 1.2 – Installing **ARGscape** Locally. Once **ARGscape** is installed, the command-line tools can be

used from any terminal in which Python is available and **ARGscape** is installed. The available commands include:

S1.4.1 **argscape**

The basic **argscape** command loads the web application in a browser window.

S1.4.2 **argscape_load**

The **argscape_load** command includes features for session file management, including loading tree sequences from files using

```
> argscape_load load --file <filename>
```

or, with sample and/or node locations, using

```
> argscape_load load-with-locations --file <filename> --sample-csv <filename> --  
node-csv <filename>
```

To view the complete set of file management commands, run

```
> argscape_load --help
```

S1.4.3 **argscape_infer**

The **argscape_infer** command unifies features for running spatial and temporal inference methods on loaded tree sequences. A simple command-line interface is provided for selecting files, inference methods, and output locations, which can be accessed by running

```
> argscape_infer
```

To view the complete set of spatiotemporal inference commands, run

```
> argscape_infer --help
```

S2 FastGaia Algorithm

FastGaia is an unpublished set of algorithms for inferring the geographic locations of ancestral nodes in a tree sequence given georeferenced samples. Drawing inspiration from **Gaia** and Wohns’ midpoint approach [Wohns et al., 2022, Grundler et al., 2025], this approach aims to utilize more of the information encoded in the ARG than a simple midpoint approach, while running faster than **Gaia** by implementing a parallelizable greedy algorithm approach. The ease of incorporating **FastGaia** within the **ARGscape** framework demonstrates the flexibility of integrating new methods into **ARGscape**’s modular framework.

Like **Gaia**, **FastGaia** can operate in continuous space (no barriers to dispersal, Euclidean cost function) or discrete space (using a uniform transition cost between states or an input transition cost matrix). Note that **ARGscape** uses only the continuous-space version of the algorithm. The complete details of both **FastGaia** algorithms are provided below, along with the necessary notation.

FastGaia can be installed in Python environments running Python version 3.8 or later by running the command

```
> python -m pip install fastgaia
```

S2.1 FastGaia Algorithm Notation and Definitions

- T : Tree sequence with nodes V and edges E
- $n = |V|$: Number of nodes
- $t(u)$: Time (age) of node u

- $S \subset V$: Set of sample nodes with known locations/states
- $E(u)$: Set of edges where u is the parent
- $C(u) = \{v : (u, v) \in E\}$: Set of children of node u
- $P(u) = \{w : (w, u) \in E\}$: Set of parents of node u
- For edge $e = (u, v)$:
 - $s(e)$: Genomic span (right - left coordinates)
 - $b(e) = t(u) - t(v)$: Branch length (temporal distance)
- \mathcal{E}_u : Set of valid edges from parent u to children with known locations/states (used locally)
- ω_e : Weight assigned to edge e (continuous inference)

S2.2 FastGaia Algorithm S1: Continuous Location Inference

Input:

- Tree sequence $T = (V, E)$
- Sample locations $\mathcal{L}_S = \{\ell_u \in \mathbb{R}^d : u \in S\}$
- Boolean flags: $w_{\text{span}}, w_{\text{branch}}$

Output:

- Inferred locations $\mathcal{L} = \{\ell_u \in \mathbb{R}^d : u \in V\}$

Algorithm S1: Continuous Location Inference

```

1: Initialize  $\ell_u \leftarrow \text{NaN} \in \mathbb{R}^d$  for all  $u \in V$ 

2:  $\ell_u \leftarrow \mathcal{L}_S(u)$  for all  $u \in S$  ▷ Assign known sample locations

3: Partition nodes:  $V_\tau = \{u \in V : t(u) = \tau\}$  for each unique time  $\tau$ 

4: Sort times:  $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_k\}$  where  $\tau_1 < \tau_2 < \dots < \tau_k$ 

5: for each time  $\tau \in \mathcal{T}$  do ▷ Process from present to past

6:   for each node  $u \in V_\tau$  do ▷ Parallel processing possible

7:     if  $u \in S$  then

8:       continue ▷ Sample location already known

9:     end if

10:     $\mathcal{E}_u \leftarrow \{e = (u, v) : v \in C(u) \text{ and } \ell_v \neq \text{NaN}\}$ 

11:    if  $\mathcal{E}_u = \emptyset$  then

12:       $\ell_u \leftarrow \text{NaN}$  ▷ No valid children

13:      continue

14:    end if

15:    ▷ Compute weighted average location

16:    for each edge  $e = (u, v) \in \mathcal{E}_u$  do

17:       $\omega_e \leftarrow 1.0$ 

18:      if  $w_{\text{span}} = \text{True}$  then

19:         $\omega_e \leftarrow s(e)$ 

20:      end if

21:      if  $w_{\text{branch}} = \text{True}$  then

```

```

22:          $\omega_e \leftarrow \omega_e \cdot \frac{1}{b(e)}$  ▷ Inverse branch length
23:     end if
24: end for
25:      $W \leftarrow \sum_{e \in \mathcal{E}_u} \omega_e$ 
26:     if  $W = 0$  then
27:          $\ell_u \leftarrow \frac{1}{|\mathcal{E}_u|} \sum_{e=(u,v) \in \mathcal{E}_u} \ell_v$  ▷ Equal weights fallback
28:     else
29:          $\ell_u \leftarrow \frac{1}{W} \sum_{e=(u,v) \in \mathcal{E}_u} \omega_e \cdot \ell_v$  ▷ Weighted average location of children
30:     end if
31: end for
32: end for
33: return  $\mathcal{L} = \{\ell_u : u \in V\}$ 

```

Weight Formula: For edge $e = (u, v)$ connecting parent u to child v :

$$\omega_e = \begin{cases} s(e) \cdot \frac{1}{b(e)} & \text{if } w_{\text{span}} \wedge w_{\text{branch}} \\ s(e) & \text{if } w_{\text{span}} \wedge \neg w_{\text{branch}} \\ \frac{1}{b(e)} & \text{if } \neg w_{\text{span}} \wedge w_{\text{branch}} \\ 1 & \text{if } \neg w_{\text{span}} \wedge \neg w_{\text{branch}} \end{cases}$$

Location Update:

$$\ell_u = \frac{\sum_{v \in C(u)} \omega_{(u,v)} \cdot \ell_v}{\sum_{v \in C(u)} \omega_{(u,v)}}$$

S2.3 FastGaia Algorithm S2: Discrete State Inference

Input:

- Tree sequence $T = (V, E)$
- Sample states $\mathcal{S}_S = \{\sigma_u \in \{1, 2, \dots, m\} : u \in S\}$
- Cost matrix $M \in \mathbb{R}^{m \times m}$ (optional): M_{ij} = cost of transition from state i to state j

Output:

- Inferred states $\mathcal{S} = \{\sigma_u \subseteq \{1, \dots, m\} : u \in V\}$ (set-valued to account for ties)

Algorithm S2: Discrete State Inference

- 1: Initialize $\sigma_u \leftarrow \emptyset$ for all $u \in V$
- 2: $\sigma_u \leftarrow \{\mathcal{S}_S(u)\}$ for all $u \in S$ ▷ Assign known sample states
- 3: Determine state space: $\Sigma = \{1, 2, \dots, m\}$
- 4: Partition nodes: $V_\tau = \{u \in V : t(u) = \tau\}$ for each unique time τ
- 5: Sort times: $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_k\}$ where $\tau_1 < \tau_2 < \dots < \tau_k$
- 6: **for** each time $\tau \in \mathcal{T}$ **do** ▷ Process from leaves to root
- 7: **for** each node $u \in V_\tau$ **do** ▷ Parallel processing possible
- 8: **if** $u \in S$ **then**
- 9: **continue** ▷ Sample state already known
- 10: **end if**
- 11: ▷ Compute cost for each candidate state
- 12: **for** each candidate state $\alpha \in \Sigma$ **do**

```

13:       $\mathcal{C}_{\text{child}} \leftarrow 0$ 
14:      for each edge  $e = (u, v)$  where  $v \in C(u)$  and  $\sigma_v \neq \emptyset$  do
15:          for each state  $\beta \in \sigma_v$  do
16:               $c \leftarrow M_{\alpha, \beta}$  if  $M$  provided, else  $c \leftarrow 1$ 
17:               $\mathcal{C}_{\text{child}} \leftarrow \mathcal{C}_{\text{child}} + c \cdot s(e) \cdot b(e)$ 
18:          end for
19:      end for
20:  end for
21:   $c^* \leftarrow \min_{\alpha \in \Sigma} \mathcal{C}_{\text{child}}$ 
22:   $\sigma_u \leftarrow \{\alpha \in \Sigma : \mathcal{C}_{\text{child}} = c^*\}$  ▷ All optimal states
23:  end for
24: end for
25: return  $\mathcal{S} = \{\sigma_u : u \in V\}$ 

```

Cost Formula: For node u and candidate state α :

$$\mathcal{C}(u, \alpha) = \sum_{v \in C(u)} \sum_{\beta \in \sigma_v} c(\alpha, \beta) \cdot s_{uv} \cdot b_{uv}$$

where the transition cost function is

$$c(i, j) = \begin{cases} M_{ij}, & \text{if a cost matrix } M \text{ is provided} \\ 1, & \text{otherwise (uniform cost)} \end{cases}$$

State Assignment:

$$\sigma_u = \arg \min_{\alpha \in \Sigma} \mathcal{C}(u, \alpha)$$

Note: σ_u may include multiple states if there are ties for the minimum cost.

S3 2D Visualization Tip Ordering Algorithms

While all 2D ARGscape visualizations are fully interactive and customizable, we also provide a diverse set of tree sequence-specific tip ordering algorithms designed to clarify complex, tangled graph layouts. The available tip ordering algorithms are detailed below. Ideal choice of tip ordering algorithm will vary on a case-by-case basis, with no clear general rules for which to choose in what scenario. However, especially for very complex graphs, **dagre-d3** mode will often produce the clearest graphs.

S3.1 Numeric

The numeric tip ordering algorithm places sample nodes along the x -axis in order of increasing node ID. Ancestral nodes are placed using a force-directed simulation. This ordering will often result in complex and tangled visualizations.

S3.2 First Tree

The first-tree tip ordering algorithm places sample nodes along the x -axis in the order of a minlex postorder traversal of the first local tree in the tree sequence. This order is generated automatically by **tskit** [Kelleher et al., 2016, Ralph et al., 2020, Wong et al., 2024], and is

the same ordering algorithm used by `tskit_arg_visualizer` [Kitchens & Wong, 2025].

S3.3 Center Tree

The center-tree tip ordering algorithm places sample nodes along the x -axis in the order of a minlex postorder traversal of the middle local tree in the tree sequence. If only one local tree is available, this is the same order as in first-tree. If two trees meet the “center” criteria, the first is used.

S3.4 Consensus

The consensus tip ordering algorithm orders sample nodes along the x -axis according to a majority vote by minlex postorder traversals across K local trees in the tree sequence, where K scales with number of local trees, and is bounded by $[1, 50]$. The K local trees are selected from evenly spaced genomic positions across the tree sequence. If only one local tree is available, this is the same order as in first-tree or center-tree.

S3.5 Ancestral Path

The ancestral path tip ordering algorithm aims to group sample nodes by shared ancestry and similar time to coalescence. It creates groups of samples that coalesced at similar times, then places groups with more recent shared ancestry closer to the center of the graph. This aims to minimize path crossings by putting groups with deeper ancestry – and therefore longer edges – towards the outside of the graph.

S3.6 Coalescence

The coalescence tip ordering algorithm orders sample nodes along the x -axis in decreasing order of time to coalescence.

S3.7 Dagre-d3

When `dagre-d3` mode is enabled, all nodes in the graph are placed by the `dagre-d3` React library using an edge-crossing minimization algorithm. The force-directed simulation is disabled when this mode is active.

References

- Grundler, MC, Terhorst, J, Bradburd, GS. A geographic history of human genetic ancestry. *Science* 2025; 387: 1391–1397.
- Kelleher, J, Etheridge, AM, McVean, G. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Comput Biol* 2016; 12: e1004842.
- Kitchens, J, Wong, Y. `tskit_arg_visualizer`: interactive plotting of ancestral recombination graphs. *arXiv* 2025; doi: arXiv:2508.03958
- Ralph, P, Thornton, K, Kelleher, J. Efficiently summarizing relationships in large samples: A general duality between statistics of genealogies and genomes. *Genetics* 2020; 215: 779–797.
- Wohns, AW, Wong, Y, Jeffery, B, et al. A unified genealogy of modern and ancient genomes. *Science* 2022; 375: eabi8264.

Wong, Y, Ignatieva, A, Koskela, J, et al. A general and efficient representation of Ancestral Recombination Graphs. *Genetics* 2024; 228: iyae100.