GenoPath

Ryan Tobin, Sayaka Miura, Sudhir Kumar
Institute for Genomics and Evolutionary Medicine, Temple University

February 12, 2025

Contents

Chapter 1

Introduction

1.1 About GenoPath

We have developed GenoPath, a genomic analysis pipeline with both efficiency and user-friendliness in mind. GenoPath stands out by reducing computational overhead, and it offers an intuitive command line structure and streamlined workow, making it accessible to researchers with varying levels of expertise. By balancing computational efficiency with ease of use, GenoPath aims to democratize tumor evolution analysis, enabling both novice and experienced researchers to achieve high-quality inferences without the need for extensive computational knowledge.

GenoPath integrates a suite of cutting-edge bioinformatic tools. These include CloneFinder [7] for predicting clones, PathFinder [4] for inferring cell migration routes, PhyloSignare [8] for mutational signature inference, and other advanced tools for constructing sample trees and estimating variant allele frequencies. By leveraging these tools, GenoPath streamlines the analysis process.

1.2 Purpose of the Manual

This manual provides step-by-step instructions for installing, configuring, and using GenoPath. It is designed to help users from beginning to advanced, set up and operate GenoPath efficiently while minimizing erros and optimizing performance.

Chapter 2

Installation

2.1 System Requirements

Before downloading GenoPath, you may need to download Git, if you have not already. Git is available for download online from https://git-scm.com/download/win

2.2 Downloading GenoPath

GenoPath is freely available on the web at https://github.com/ryan-tobin/GenoPath

2.3 Installation Steps

The following code downloads the entire pipeline using command prompt:

```
1. git clone https://github.com/ryan-tobin/GenoPath

2. cd GenoPath
```

2.4 Setting Up the Environment

2.4.1 Configuring the Environment

We recommend configuring an Anaconda or Miniconda environment for running GenoPath, as it may lead to less computational strain on your system. Anaconda is available for installation from https://www.anaconda.com/download.

2.4.2 Required Libraries and Dependencies

The following libraries are required:

- `matplotlib`
- `numpy`
- `pandas`
- `graphviz`
- `Pillow`
- `requests`
- `networkx==2.8`
- `argparse`
- `scipy`
- `biopython`
- `pydot`
- `ete3`
- `PyQt5`

The following dependencies are required for `PhyloSignare`:

- SignatureEstimate
- deconstructSigs
- MutationalPatterns

When GenoPath is first run, these requirements should automatically be downloaded. If this does not work, the python packages required can be downloaded using the following code when in the GenoPath directory:

```
pip install -r requirements.txt
```

Chapter 3

Getting Started

3.1 Sample Data

Sample datasets can be found withtin the GenoPath github under the folder *Sample_Datasets*

3.2 Running GenoPath for the First Time

If you would like to test GenoPath and its functions, there is a sample dataset that can be used. The following command can be used to test GenoPath. Navigate to where you downloaded GenoPath to, and in use the files within the Sample_Datasets folder[1]:

```
python genopath.py --run_process All --target_dir [
    path/to/dir] snv /Sample_Datasets/input.tsv --
    max_graphs_per_tree 50 /Sample_Datasets/control.txt
     --abundance_weighted True --sv_file /
    Sample_Datasets/sv.txt
```

[1]Note: No Driver Mutation Analysis Will Occur When Using the Sample Datasets

Chapter 4

Pipeline Components

4.1 CloneFinder

4.1.1 Purpose and Functionality

CloneFinder is a computational tool designed to address the challenge of genomic heterogeneity observed within tumors through bulk sequencing. This heterogeneity, characterized by the presence of multiple tumor cell populations (clones), poses significant hurdles in understanding tumor biology and in the development of understanding such biology [2], CloneFinder aids in inferring the genotypes of these tumor cell populations directly from bulk sequencing data [7]. In our pipeline, CloneFinder serves a critical role at the stage of genomic analysis, where it is pivotal for separating the structure of tumor cell populations, allowing for a deeper understanding of intra-tumor genetic diversity. The outputs generated by CloneFinder, particularly the inferred genotypes and clone presence data, are instrumental for subsequence analysis in our pipeline. Specifically, these outputs are utilized by PathFinder for exploring the evolutionary trajectories of tumor clones, enhancing our understanding of tumor progression. Similarly, the data feeds into Picante for further ecological and evolutionary analysis, enabling comprehensive insights into the tumor microenvironment and its evolution over time. The implementation of CloneFinder significantly enhances our pipeline's capabilities to dissect tumor heterogeneity. By providing accurate estimates of clone genotypes and its presence data, it lays a foundational step for downstream analysis with PathFinder and Picante.

4.1.2 Usage Instructions

To run CloneFinder alone, the following command should be used:

```
python genopath.py --run_process CloneFinder --
   target_dir [path/to/dir] snv [/path/to/input.tsv]
```

4.1.3 Input

The input file is a tab-delimited text file, which contains the read counts of wild-type and mutant alleles for each sample. Normal sample should not be included. Each line in the input files gives information for each variant. Explanation of each column and specific formatting can be found below.
* "CHR" : Chromosome number, N
* "Position" : Chromosome position, N
* "Wild" : Reference/Wild Allele, X
* "Mutant" : Mutant/Alternate Allele, X
* "Trinucletide": Trinucleotide repeats for each chromosome number and position, XXX. **NOTE** Only required when performing analysis with PhyloSignare.
* "XX:ref": Reference read count for the sample, XX
* "XX:alt": Variant read count for the sample, XX

IMPORTANT
1. Low quality SNVs (e.g., low coverage) should be removed from the Input file.
2. The data needs to contain parsimony-informative sites, when the tumor profiles of presence/absence of mutations are generated. The recommended number of samples per dataset is more than 4.
3. Input file name and tumor/sector names can contain only alphabetic and numeric characters and should not begin with numbers. Total read count for each variant cannot contain values < 1.
4. cancer cell fraction (CCF) data needs to be converted into SNV read count table, under the assumption of CNA-free SNVs. Thus, variant read count is CCF that is multiplied with the total read count and divided by two. Reference read count can be computed by subtracting variant read count from total read count.

An example input file can be found in the *Sample_Datasets* in the GitHub.

4.2 PathFinder

4.2.1 Purpose and Functionality

PathFinder aims to reconstruct the routes of cancer cell migrations using a Bayesian framework [4]. It capitalized on the clone phylogeny, mutational differences among clones, and the distribution of clones across primary and metastatic tumors - data intricately analyzed and prepared by CloneFinder. By integrating this information, PathFinder can accurately infer the paths taken by tumor cells from primary sites to metastases and among metastatic sites themselves. By integrating the CloneFinder's outputs, PathFinder enriches our pipeline with the ability to decode the migration histories of cancer cells. This capability is instrumental in understanding the seeding process of new metastases. Whether from primary tumors or existing metastatic sites.

4.2.2 Usage Instructions

To use PathFinder, CloneFinder must also be ran. The following command should be used:

```
python genopath.py --run_process CloneFinder
    PathFinder --target_dir [path/to/dir] snv [/path/to
    /input.tsv] --primary [tumor] (optional) --
    max_graphs_per_tree [int] (optional)
```

4.2.3 Input

The required input files are:
1) An alignment file in FASTA or MEGA format, specifying the sequence of each tumor clone, including one sequence labeled "Normal" specifying the normal sequence(this is the germline sequence, which represents a healthy cell without any somatic mutations).
2) A tab delimited clone presence file, specifying the presence of each clone in each anatomical site (clones = columns, anatomical sites = rows), as in the provided example files.

IMPORTANT
The number of tumors cannot exceed 19.

NOTE No input files are required, these descriptions are for your information only. All files will be created within the pipeline.

4.3 PhyloSignare

4.3.1 Purpose and Functionality

Cancer cell genomes are subject to continuous change due to mutations [6], with the mutational process varying over time within patients. This dynamic landscape leaves unique signatures in the genomic variation accumulated in tumors [1], which are crucial for understanding the evolution and treatment of cancer. PhyloSignare was developed to address the challenge of accurately detecting branch-specific mutation signatures when applied to individual phylogeny branches. PhyloSignare aims to enhance the inference of branch-specific mutational signatures through a joint analysis of mutations mapped on proximal branches of the cancer cell phylogeny. This approach significantly reduces the false-positive discovery rate of branch specific signatures and is capable of detecting faint signatures that may be overlooked by other methods. Utilizing the same input as CloneFinder to create the input within the pipeline, it seamlessly integrates into our pipeline, enabling a more nuanced analysis of the mutational landscape.

4.3.2 Usage Instructions

To use PhyloSignare, CloneFinder must also be ran. There are three options:

1. `With Known Driver Mutations`
2. `With Driver Mutation Calculation`
3. `No Driver Mutation Analysis`

1. With Known Driver Mutations

```
python genopath.py --run_process CloneFinder
    PhyloSignare --target_dir [path/to/dir] --
    driver_mutation_file [path/to/file] snv [/path/to/
    input.tsv] [path/to/control_file]
```

2. With Driver Mutation Calculation

```
python genopath.py --run_process CloneFinder
    PhyloSignare --target_dir [path/to/dir] --
    ref_alt_file [/path/to/file] --tool CGI ---email [
    email] --token [token] --cancer_type_input [
    cancer_type] snv [path/to/input.tsv] [/path/to/
    control_file]
```

3. No Driver Mutation Analysis

```
python genopath.py --run_process CloneFinder
    PhyloSignare --target_dir [path/to/dir] snv [/path/
    to/input.tsv] [path/to/control_file]
```

4.3.3 Input

Please note, the only input file required to be provided by the user is the Control file. All other files will be created by the pipeline. The explanation provided for the two other input files is purely to show what those files contain.

1. Map all mutations on a clone phylogeny and make mutation count tables for each branch of a clone phylogeny.
Each mutation needs to be classified into the conventional 96 mutation types (https://cancer.sanger.ac.uk/cosmic/signatures/SBS/). A single file should be created for each branch, and an observed count of each mutation type needs to be listed by using the following format:
$A[C > A]A, 1$
$A[C > A]C, 4$

$A[C > A]G, 1$
$A[C > A]T, 0$
$A[C > G]A, 2$
$A[C > G]C, 2$
...

2. List the mutation count tables and the topology of clone phylogeny. This file needs to contain the topology of a clone phylogeny and information of branch ID and its corresponding mutation count table, which is prepared above (step 1). To describe the topology of a clone phylogeny, all ancestor and its direct descendant branch pairs should be listed by separating with "->." For example, B1->B2 means B1 is ancestral branch, and B2 is its direct descendant branch. For example,

```
#BranchID File
B4   A.csv
B5   B.csv
. . .
#Tree
B1->B2
B1->B5
B5->B3
..
```

The following figure provides visual representation of the meaning of the clone phylogeny:

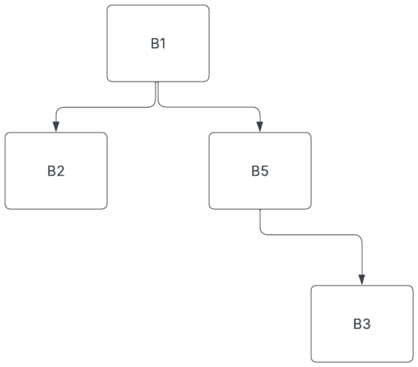

Figure 4.1: Visual Representation of Clone Phylogeny

3. Make a control file. A control file should contain the information of a signature refitting method to be used (QP, dSig, MutPat, or MutCon; see the section of R dependencies for the detail), the version of COSMIC signatures

(2 or 3; https://cancer.sanger.ac.uk/cosmic), list of expected signatures in your dataset, and ID for the output file name.
For example,
BaseMethod QP #QP, dSig, or MutPat
Signature 2 #2 or 3. 2 for V2 COSMIC signatures and 3 for V3.
Signature List all #all for using all signatures or list of signatures, e.g., S1,S2,S3,S12,S14 for COSMIC V2 and SBS1,SBS2,SBS3 for COSMIC V3
Signature ID Test

4.4 Picante

4.4.1 Purpose and Functionality

Picante is a powerful tool in the analysis of phylogenetic diversity and community structure. We leverage Picante to analyze the phylogenetic relationships and distances among tumor cell populations (clones) identified by CloneFinder. Specifically, we employ three of Picante's functions - comdist [3], comdistnt [3], and Unifrac [5] - to quantify the genetic diversity and relatedness of these clones. The comdist function calculates the inter-community mean pairwise distance, providing a measure of the average genetic distance between clones across tumor samples. This helps in understanding the overall diversity and potential evolutionary divergence among tumor populations. Comdistnt calculates the inter-community mean nearest taxon distance, offering insights into the closest genetic relationships among clones within and across tumor communities. It highlights the potential for shared evolutionary pathways or common ancestral clones. The unweighted UniFrac distance, calculated by the unifrac function, assesses the phylogenetic distance between communities based on the presence or absence of lineages. This metric is crucial for understanding the degree of phylogenetic overlap or uniqueness among tumor clones, indicating how distinct the evolutionary paths of different tumor populations are. Using the genotypic profiles and clone counts derived from CloneFinder, we apply these Picante functions to analyze the phylogenetic diversity and distances among tumor clones. This analysis feeds into the construction of a neighbor-joining (NJ) tree

4.4.2 Usage Instructions

To use Picante, CloneFinder must also be ran. There are four options:

1. `All`
2. `Comdist`
3. `Comdistnt`
4. `Unifrac`

1. Picante (All)

```
python genopath.py --run_process CloneFinder
    Picante_all --target_dir [path/to/dir] snv [/path/
    to/input.tsv] --abundance_weighted [True/False]
```

2. Picante (Comdist)

```
python genopath.py --run_process CloneFinder
    Picante_comdist --target_dir [path/to/dir] snv [/
    path/to/input.tsv] --abundance_weighted [True/False
    ]
```

3. Picante (Comdistnt)

```
python genopath.py --run_process CloneFinder
    Picante_comdistnt --target_dir [path/to/dir] snv [/
    path/to/input.tsv] --abundance_weighted [True/False
    ]
```

4. Picante (Unifrac)

```
python genopath.py --run_process CloneFinder
    Picante_unifrac --target_dir [path/to/dir] snv [/
    path/to/input.tsv] --abundance_weighted [True/False
    ]
```

4.4.3 Input

There are no input files required for any Picante function besides for the input.tsv file required for CloneFinder. Since CloneFinder is required to be run with Picante, that is the only input file needed.

4.5 Meltos

4.5.1 Purpose and Functionality

Meltos introduces a computational framework tailored to address the complex challenge constructing tumor phylogeny trees based on somatic structural variants (SVs) among multiple samples. Utilizing somatic single nucleotide variants (SNVs) derives tumor phylogeny trees as a guide, Meltos identifies high-confidence SVs to enrich and refine the construction of comprehensive tumor lineage trees [9]. This process is emphasized by an optimized formulation that does not presuppose identical evolutionary trajectories for SVs and SNVs. Meltos harnesses multiple genomic read signals to pinpoint SV breakpoints in whole-genome sequencing data, employing a probabilistic formulation to estimate variant allele frequencies (VAFs) of SV events, thereby enhancing the accuracy and reliability of SV analysis. In the culminating phase of our pipeline, Meltos leverages the SV data provided by the user, building upon the output of preceding tools such as CloneFinder. By identifying and assigning somatic SVs with high confidence, a SV tree is crafted that encapsulates the complex landscape of tumor evolution through the prism of structural variation. This SV tree offers a complementary perspective to SNV-based phylogenetic analysis, enriching our understanding of tumor heterogeneity and lineage.

4.5.2 Usage Instructions

To use Meltos, CloneFinder must also be run. This is due to the output files of CloneFinder being needed to run Meltos. The following command should be used:

```
python genopath.py --run_process CloneFinder Meltos --
    target_dir [path/to/dir] snv [/path/to/input.tsv]
    --sv_file [/path/to/sv_file]
```

4.5.3 Input and Output

The input file is a tab-delimited text file, which contains the id, chromosome number and position (SV), normal genome counts and tumor site genome counts. Normal sample should be included and set to 0. Each line in the input files gives information for each variant. Explanation of each column and specific formatting can be found below.

* "id": Numbered list of 1 to n where n is the last number of samples, XX
If there are 200 samples, "id" will be 1 to 200, increased by 1 each new line.
* "SV": Chromosome number and position, chrN:chrPosition−chrN:chrPosition
* "Normal_GenomeCounts": Normal genome counts where all values are 0, 0
* "XX_GenomeCounts": Genome counts for each tumor site which is the estimated VAF, XX

4.6 Driver Mutation Analysis

4.6.1 Purpose and Functionality

We included the option of completing analysis of driver mutation mutations within the pipeline for your tumor sites. Its primary purpose is to identify mutations that drive the initiation and progression of cancer, which we know as driver mutations. These mutations are distinguished from passenger mutations, which are incidental and do not contribute to cancer development. Knowing where driver mutations occur can lead to results that will help better understand genetic alterations.

4.6.2 Calculating Driver Mutations

To perform driver mutation analysis, we provided two separate options. The first option, which uses the Cancer Genome Interpreter (CGI), is built into the pipeline and is our primary option. If for some reason CGI does not work, we have a secondary option using OpenCRAVAT to calculate driver mutations. This secondary option is not built into the pipeline, but a script to use is still provided within the directory.
For our primary option, the flag *"−−ref_alt_file"* can be added to the pipeline code followed by a .txt file with the format below:

* chr: column with chromosome number, chrN
* pos: column with chromosome position, n
* ref: column with reference allele, XX
* alt: column with alternate or mutant allele, XX

For our secondary option, the following code can be ran to calculate driver mutations with OpenCRAVAT. The input file format and command line requirements are as follows:

* -i: Tab separated file with genomic data, example file located in *Sample_Datasets* folder on Github.
* -o: Output file path
* -c: Cancer type, 4 digit common code for cancer, PRAD

The code to run with OpenCRAVAT is as follows:

```
python driver_mutation_with_cravat.py -i [/path/to/
    input] -o [/path/to/output] -c [cancer_type]
```

4.6.3 Known Driver Mutations

For known driver mutations and no calculation, the flag *"−−driver_mutation_file"* can be added to the pipeline code followed a .txt file with the format below:

* Driver Gene: Name of gene, XX
* P-value: If p-value is known for the mutation, please use here. If not, please use 0
* ref>alt: Genetic chance for reference and alternate allele, X>X
* Sequence Ontology: Type of mutation that occurred (missense, deletion, ...), XX
* Mutation Summary: Chromosome number and position the mutation occured at, chrN:Position

Example file located in *Sample_Datasets* folder on Github.

4.6.4 No Driver Mutation Analysis

In the case where driver mutation analysis is not wanted, the flag "−−driver_mutation_file" can be excluded from the command. There will a confirmation prompt to make sure no driver mutation analysis is wanted, and then will proceed without.

Chapter 5

Comprehensive Analysis

All input and output files are as described above in the above sections. **NOTE** Meltos is not run with the All function and must be run separately.

5.1 With Known Driver Mutations

```
python genopath.py --run_process All --target_dir [
   path/to/dir] --driver_mutation_file [/path/to/file]
    snv [/path/to/input.tsv]  --primary [tumor] (
   optional) --max_graphs_per_tree [n] (optional) [/
   path/to/control_file] --abundance_weighted [True/
   False]
```

5.2 With Driver Mutation Calculation

```
python genopath.py --run_process All --target_dir [
   path/to/dir] --ref_alt_file [/path/to/file] --tool
   CGI ---email [email] --token [token] --
   cancer_type_input [cancer_type] snv [path/to/input.
   tsv] --primary [tumor] (optional) --
   max_graphs_per_tree [n] (optional) [/path/to/
   control_file] --abundance_weighted [True/False]
```

5.3 With No Driver Mutation Analysis.

```
python genopath.py --run_process All --target_dir [
   path/to/dir] snv [/path/to/input.tsv]  --primary [
   tumor] (optional) --max_graphs_per_tree [n] (
   optional) [/path/to/control_file] --
   abundance_weighted [True/False]
```

5.4 Output Files

All primary graphs will be produced by using the comprehensive analysis command(s).

Bibliography

[1] Australian Pancreatic Cancer Genome Initiative, ICGC Breast Cancer Consortium, ICGC MMML-Seq Consortium, ICGC PedBrain, Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A. J. R., Behjati, S., Biankin, A. V., Bignell, G. R., Bolli, N., Borg, A., Børresen-Dale, A.-L., Boyault, S., Burkhardt, B., Butler, A. P., Caldas, C., Davies, H. R., … Stratton, M. R. (2013). *Signatures of mutational processes in human cancer.* Nature, 500(7463), 415–421. https://doi.org/10.1038/nature12477

[2] Gao, C., Furge, K., Koeman, J., Dykema, K., Su, Y., Cutler, M. L., Werts, A., Haak, P., & Vande Woude, G. F. (2007). *Chromosome instability, chromosome transcriptome, and clonal evolution of tumor cell populations.* Proceedings of the National Academy of Sciences, 104(21), 8995–9000. https://doi.org/10.1073/pnas.0700631104

[3] Kembel, S. W., Cowan, P. D., Helmus, M. R., Cornwell, W. K., Morlon, H., Ackerly, D. D., Blomberg, S. P., & Webb, C. O. (2010). *Picante: R tools for integrating phylogenies and ecology.* Bioinformatics, 26(11), 1463–1464. https://doi.org/10.1093/bioinformatics/btq166

[4] Kumar, S., Chroni, A., Tamura, K., Sanderford, M., Oladeinde, O., Aly, V., Vu, T., & Miura, S. (2020). *PathFinder: Bayesian inference of clone migration histories in cancer.* Bioinformatics, 36(Supplement_2), i675–i683. https://doi.org/10.1093/bioinformatics/btaa795

[5] Lozupone, C., & Knight, R. (2005). *UniFrac: A New Phylogenetic Method for Comparing Microbial Communities.* Applied and Environmental Microbiology, 71(12), 8228–8235. https://doi.org/10.1128/AEM.71.12.8228-8235.2005

[6] MacConaill, L. E., & Garraway, L. A. (2010). *Clinical Implications of the Cancer Genome.* Journal of Clinical Oncology, 28(35), 5219–5228. https://doi.org/10.1200/JCO.2009.27.4944

[7] Miura, S., Gomez, K., Murillo, O., Huuki, L. A., Vu, T., Buturla, T., & Kumar, S. (2018). *Predicting clone genotypes from tumor bulk sequencing of multiple samples.* Bioinformatics, 34(23), 4017–4026. https://doi.org/10.1093/bioinformatics/bty469

[8] Miura, S., Vu, T., Choi, J., Townsend, J. P., Karim, S., & Kumar, S. (2022). *A phylogenetic approach to study the evolution of somatic mutational processes in cancer.* Communications Biology, 5(1), 617. https://doi.org/10.1038/s42003-022-03560-0

[9] Ricketts, C., Seidman, D., Popic, V., Hormozdiari, F., Batzoglou, S., & Hajirasouliha, I. (2020). *Meltos: Multi-sample tumor phylogeny reconstruction for structural variants.* Bioinformatics, 36(4), 1082–1090. https://doi.org/10.1093/bioinformatics/btz737