

---

# **ortho\_seqs**

***Release 1.0.1***

**Saba Nafees**

**Sep 16, 2021**



# CONTENTS

<b>1</b>	<b>Introduction:</b>	<b>1</b>
<b>2</b>	<b>Guide</b>	<b>3</b>
2.1	Background & Quickstart . . . . .	3
2.1.1	ortho_seqs . . . . .	3
2.1.2	Usage . . . . .	3
2.1.3	Results & Outputs . . . . .	6
2.1.4	Results & Outputs . . . . .	7
2.1.5	To run the GUI (currently in development) . . . . .	7
2.1.6	Support . . . . .	8
2.1.7	Roadmap . . . . .	9
2.1.8	Contribution . . . . .	9
2.1.9	Authors and acknowledgements . . . . .	9
2.1.10	License . . . . .	9
2.2	Lincense . . . . .	9
2.3	Get Further Help . . . . .	10
2.3.1	Contact . . . . .	10
<b>3</b>	<b>Indices and tables</b>	<b>11</b>



## **INTRODUCTION:**

This documentation accompanies the `ortho_seqs` python command line tool that computes multivariate tensor-based orthogonal polynomials based on DNA, RNA or protein sequence data and maps corresponding phenotypic information onto the sequence space.

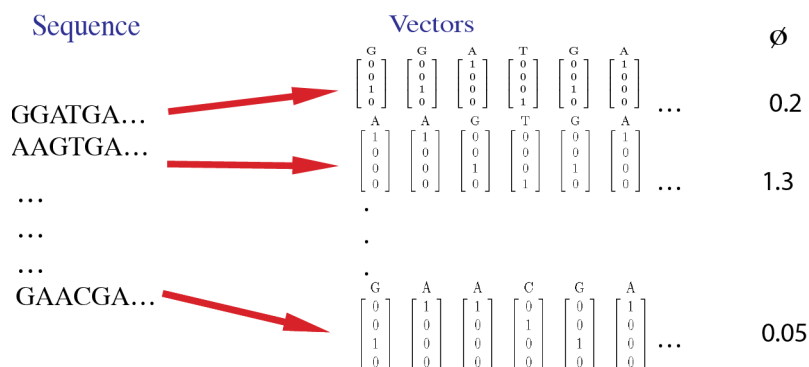


## 2.1 Background & Quickstart

### 2.1.1 ortho\_seqs

Ortho\_seqs is a command line to convert sequence data (DNA/protein) to tensor-valued orthogonal polynomials and project phenotypes onto the polynomial space. We do this by first converting the sequence information into 4-dimensional (for DNA) or 20-dimensional (for amino acids) vectors. The method can also be used for padded sequences to deal with unequal sequence lengths. Find out more about the approach in this paper [Analyzing genomic data using tensor-based orthogonal polynomials with application to synthetic RNAs](#). The paper gives an example of this method as applied to a case of synthetic RNA from a previously published dataset. Another manuscript detailing the use of this method to understand binding affinities of transcription factors (TFs) is currently in progress.

For example, the sample data inputs for this tool are shown in this image. Here, each site in a sequence is first converted to a 4-dimensional vector. The input data includes phenotype values for each sequence.



### 2.1.2 Usage

First, install an environment with dependencies for this package:

```
conda create -n ortho_seqs pip
pip install -r requirements.txt
conda activate ortho_seq
```

or

```
conda env create -f conda_environment.yml
conda activate ortho_seq
```

**Then, install the package:**

```
python setup.py install
```

**Gather the input files needed.**

1. You'll have one file with sequence data as seen in the first column in the image above (.txt or .dat or .csv). See repo's data folder for examples of what these look like for DNA or proteins.
2. You'll have one file with corresponding phenotypes as seen in the last column in the image above (.txt or .dat or .csv) with the same length as the number of sequences. Phenotypes here are defined as real numbers (see paper linked above for more background on this).

**Then, to run the commandline tool:**

To start with a test example, you can run the sample command below:

```
ortho_seq orthogonal-polynomial ./ortho_seq_code/tests/data/nucleotide/first_order/test_
↪ seqs_2sites_dna.txt --molecule DNA --pheno_file ./ortho_seq_code/tests/data/nucleotide/
↪ first_order/trait_test_seqs_2sites_dna.txt --poly_order second --out_dir ../results_
↪ ortho_seq_testing/DNA_2sites_test_run/
```

The above sample command line is building the tensor-valued orthogonal polynomial space based on the sequence data which consists of 12 sequences, each with two sites. Since these are DNA sequences, the vectors are 4-dimensional. These used to be flags for sites, dimensions, and population size, but new functionality will automatically calculate these. Corresponding to each sequence is a phenotype value (a real number) as given in the phenotype file. For DNA, the tool can run first and second order analyses currently. We'll implement third order in a future version. For amino acids, the current version supports first order analysis and we hope to expand this in the future.

Amino acids/nucleotides that do not appear in any sequence will be removed from the alphabet when the letters are being converted to first order vectors. For example, if the residue 'R' (Arginine) never occurs in the sequence dataset, the first order vectors will now have 19 dimensions (instead of 20) and 20 dimensions (instead of 21) if the sequences are padded with 'n'. This is done to greatly reduce runtime for larger sequence datasets and for longer sequences. When the program will run, it will return this sentence:

```
Will be computing p sequences with s sites, and each vector will be d-dimensional.
```

Where p represents the population size (number of rows in sequence file), s represents the number of sites, and d represents the number of amino acids/nucleotides detected in the sequence file (adds on 1 for lowercase n's). For the above example, the program will return

```
Will be computing 12 sequences with 2 sites, and each vector will be 4-dimensional.
```

Along with regressions on each site independent of one another and onto two sites at a time, the above command also computes *Fest* which is the phenotype estimated by the regressions. This shows that the mathematical calculations are done correctly as we now have an equation that accurately captures our initial data points. This only works here for sequences with 2 sites. If we had more sites, we'd need to do higher order calculations in order to capture all our combinations. Therefore, when running the tool with more sites, as will probably be the case for most users, even just going up to second order gives us useful information about our system. First order tells us the importance of each



site (independent of any correlations it might have with another site) and second order tells the importance of pairs of nucleotides independent of other pairs. Please take a look at the paper linked above to learn more about this method.

## Flags & Functionality

`--pheno_file`

Input a file with phenotype values corresponding to each sequence in the sequence file.

`--molecule`

Currently, you can provide DNA **or** protein sequences. Here, you can also provide ↪ sequences of unequal lengths.

`--sites`

The number of sites **in** a sequence. If you have sequences **with** unequal lengths, please ↪ pad them **with** a lowercase 'n'. See examples **in** the ortho\_seq\_code/tests/ folder.

`--dm`

The dimension of the vector corresponding to a site along a sequence. This **is 4 for** DNA ↪ **and 20 for** protein. For protein, you can provide a dimension of **21 in** the case that ↪ you have padded protein sequences. See test data **in** the repo **for** examples.

`--pop_size`

The number of total sequences.

`--poly_order`

The order of the polynomials that will be constructed. Currently, one can do first and second order for DNA and first order for protein.

`--out_dir`

Directory where results can be stored.

`--precomputed`

Let's say you have a case where you have the same set of sequences but two different corresponding sets of phenotypes. You can build your sequence space and then project the first set of phenotypes onto this space. Then, if you wish to see how the other set of phenotypes maps onto the same sequence space, you can use this flag so that you're not wasting time and memory to recompute the space. When doing this, be sure to add your results from the first run to the **out\_dir** when rerunning the command with the **precomputed** flag.

## 2.1.3 Results & Outputs

The tool will provide updates as the run is progressing regarding which parts of the calculations are done being computed. For example, when the mean is computed, it'll say "computed mean". All the different elements that it is computing are different parts of building the multivariate tensor-valued orthogonal polynomial space based on the sequence information. To get a general idea of what the calculations mean, please refer to the supplementary methods in the paper linked above.

```
--alphbt_input
```

Used to group amino acids/nucleotides together, or specify certain amino acids/nucleotides. For example, putting *ASGR* will tell the program to have 6 dimensions: one for each amino acid specified, and one for *z*, where every unspecified amino acid will be converted to *z*, and one for *n* (whenever sequences have unequal lengths, *ortho\_seqs* will pad the shorter sequences with *n*). You can also comma-separate amino acids/nucleotides to group them. For example, putting *AS,GR* will make the vectors 4-dimensional, one for *AS*, one for *GR*, one for every other amino acid, and one for *n*.

There are also built-in groups:

**protein\_pnp** will group by polar and non-polar amino acids, every other amino acid, and *n*.

**essential** groups by essential and non-essential amino acids, every other amino acid, and *n*. Group 1: Essential - ILVFWHKTM Group 2: Non-Essential - Everything else Group 3: *n* (Source: <https://www.ncbi.nlm.nih.gov/books/NBK557845/>)

**alberts** groups by categories set by Alberts. Group 1: Basic - KRH Group 2: Acidic - DE Group 3: AVLIPFMWGC Group 4: Everything else Group 5: *n* (Source: <https://www.ncbi.nlm.nih.gov/books/NBK21054/>)

**sigma** groups by categories set by Sigma. Group 1: Aliphatic - AILMV Group 2: Aromatic - FYV Group 3: Polar Neutral - NQCST Group 4: Acidic - KRH Group 5: Basic - DE Group 6: Other - G Group 7: Other - P Group 8: *n* (Source: <https://www.sigmaaldrich.com/US/en/technical-documents/technical-article/protein-biology/protein-structural-analysis/amino-acid-reference-chart>)

**hbond** groups by strength of hydrogen bond attractions. Group 1: Can Make Hydrogen Bonds - NQSTDERKYHW Group 2: Can Not Make Hydrogen Bonds - Everything else Group 3: *n* The first group is able to make hydrogen bonds, whereas the second group is not.

**hydrophobicity** groups by hydrophobicity. Group 1: Very Hydrophobic - LIFWVM Group 2: Hydrophobic - CYA Group 3: Neutral - TEGSQD Group 4: Hydrophilic - Everything else Group 4: *n* The first group is very hydrophobic, the second group is slightly hydrophobic, the third group is neutral, and the last group is hydrophilic.

```
--min_pct
```

When *ortho\_seqs* is run, a .csv file of covariances will be saved in the specified path. This matrix of covariances is one of the main results of the program (as shown in {sequence\_file\_name}.npz output below). The csv file will contain the covariance of each nucleotide at each site with another nucleotide at another site (or amino acids at each site). Suppose there are 5 covariance values of 2, 1, 0, 0, -1. For the percentiles, all unique *magnitudes* will be considered when assigning covariances, which will be 2, 1, and 0. 0 will be the 0th percentile (therefore, assigning 0 to the *--min\_pct* flag will return every covariance), 1 (and -1) will be 33.33..., and 2 will be 66.66... Specifying 50 as *--min\_pct* will only return the row with the covariance of 2, since only 66.6... > 50. The *min\_pct* flag is short for minimum percentile, which will remove any covariances from the .csv file that are below the given percentile. The default value is 75.

## 2.1.4 Results & Outputs

The tool will provide updates as the run is progressing regarding which parts of the calculations are done being computed. For example, when the mean is computed, it'll say "computed mean". All the different elements that it is computing are different parts of building the multivariate tensor-valued orthogonal polynomial space based on the sequence information. To get a general idea of what the calculations mean, please refer to the supplementary methods in the paper linked above. The program will save outputs in [npz format](#). See below for what is stored.

```
{sequence_file_name}.npz
```

This will store the calculations that went into constructing the polynomial space. This also includes information about the statics of our sequence space, such as mean, variance and a matrix of covariances. See figures 4 and for ideas on how mean and the matrix of covariances can be visualized. All of these calculations go into building the orthogonal polynomial space based on sequence information and at this point of the program, we have not connected the phenotype (the functional variable) with the sequence information.

```
{sequence_file_name}_covs_with_F.npz
```

This will store the covariance of the phenotype (or trait) with the polynomials. This is when we start connecting the phenotype with the sequence space.

```
{trait_file_name}_Fm.npz
```

This contains the mean trait value. This is a scalar.

```
{trait_file_name}_regressions.npz
```

This set of files contains the main results which includes the following:

1. **rFon1D**: This is the regression of the trait onto the first order conditional polynomial orthogonalized within. This tells us the regression of the phenotype onto each site and onto each nucleotide (or amino acid) at that site independent of any correlations that site might have with other sites. For the case of nucleotides, this can be visualized as bar plots as shown in Figure 6 in the paper linked above.
2. **rFon2D**: This gives 4 matrices which give the regression of the phenotype onto (site1)x(site1), (site 1)x(site 2), (site 2)x(site 1) and (site 2)x(site 2), in that order. The second matrix here is the important one and it is the same as rFon12. See description of rFon12.
3. **rFon12**: This is the regression of the trait onto *pairs* of sites for given nucleotides at each site. These are regressions on (site 1)x(site 2) independent of first order associations. Since we're looking at 2 sites at a time and there's a possibility of having 4 nucleotides at each site (for the case of DNA), we can visualize this via a 4x4 matrix as shown in Figure 8 in the paper linked above.


## 2.1.5 To run the GUI (currently in development)


A GUI version of the CLI is being actively developed in order to make it easier for users to utilize the tool. The GUI allows the user to upload the sequence and phenotype files via an upload button, specify the molecule, the polynomial order they wish to run, and whether the sequence space was already computed or not (via the precomputed button). The GUI is in its primitive form and will include further updates resembling the cli in future versions.

ortho\_seqs GUI: User interface to compute tensor-based orthogonal polynomials for sequence data

Upload sequence file:

Upload phenotype file:

Molecule:  
 

poly\_order:  
 

Precomputed?  
☐ Yes  
☐ No

cov\_hist\_{trait\_file\_name}.png

This is a histogram of all non-zero covariances. It's bin width is 0.5.

cov\_data\_frame\_{trait\_file\_name}.csv

This file is a csv file of covariances between every item at every site. This includes the item ID and site for both items in the pair used to calculate the covariance, the covariance value, the covariance magnitude, and an ID for the pair (s1-g2,s3-g4 represents the pairing of an element from the first group in the alphabet at the second site, and an element from the third group at the fourth site).

rFon1D\_graph\_{trait\_file\_name}.png

This is a bar plot of all nonzero rFon1D values of every item at every site.

## 2.1.6 Support

If you have specific or general questions, feel free to open an issue and we'll do our best to address them. If you have any comments, suggestions or would like to chat about this method or similar ideas, feel free to reach out via email at [saba.nafees314@gmail.com](mailto:saba.nafees314@gmail.com).

## 2.1.7 Roadmap

We hope to implement third order analysis for DNA in the near future. For amino acids, we hope to implement second order analysis. We'll add visualization ideas soon but if you have any thoughts on this, please feel free to reach out.

## 2.1.8 Contribution

We hope to make the tool run faster as with higher dimensions and higher order analysis of longer sequence data, we can run into memory and time issues. Any thoughts on this or visualization are welcome.

## 2.1.9 Authors and acknowledgements

The derivation of the method and the construction of an initial version of the program was done by Dr. Sean Rice who served as Saba's Ph.D. advisor. Thank you to Isaac Griswold-Steiner for helping write the function to compute generalized inner and outer products. Thank you to Pranathi Vemuri for helping with the very initial draft of the CLI, adding CI integration testing, and to Phoenix Logan for helping write unit-tests. Thank you to AhmetCan for helping initiate the first GUI version. Thank you, Aaron, for always being ready to review PRs and for your insights/help in the development process. Thank you to Vijayanta Jain and Saugato Rahman Dhruba for being the guinea pigs and running lots of sample commands, discussing the mathematics with me, and for their ideas on visualizations. Their efforts are deeply appreciated!

## 2.1.10 License

MIT

## 2.2 Lincense

MIT License

Copyright (c) 2020 Saba Nafees

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 2.3 Get Further Help

If you have more questions, please refer to the github repo at [https://github.com/snafees/ortho\\_seqs](https://github.com/snafees/ortho_seqs) for existing issues or start an issue there. You can also refer to the PyPI page at <https://pypi.org/project/ortho-seq-code/>

### 2.3.1 Contact

If you have more questions or comments, please feel free to reach out to me at [saba.nafees314@gmail.com](mailto:saba.nafees314@gmail.com). We look forward to hearing from you!

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`