fastDFE: fast and flexible inference of the distribution of fitness effects

Janek Sendrowski*, Thomas Bataillon*

Contact: sendrowski.janek@birc.au.dk, tbata@birc.au.dk

January 22, 2024

Abstract

Estimating the distribution of fitness effects (DFE) of new mutations is of fundamental importance in evolutionary biology, ecology, and conservation. However, existing methods for DFE estimation suffer from limitations, such as slow computation speed and limited scalability. To address these issues, we introduce fastDFE, a Python-based software package, offering fast and flexible DFE inference from site-frequency spectrum (SFS) data. Apart from providing efficient joint inference of multiple DFEs that share parameters, it offers the feature of introducing genomic covariates that influence the DFEs, and testing their significance. To further simplify usage, fastDFE is equipped with comprehensive VCF-to-SFS parsing utilities. These include options for site filtering and stratification, as well as site-degeneracy annotation and probabilistic ancestral-allele inference. fastDFE thereby covers the entire workflow of DFE inference from the moment of acquiring a raw VCF file. Despite its Python foundation, fastDFE comprises a full R interface, including native R visualization capabilities. The package is comprehensively tested, and documented at `fastdfe.readthedocs.io`.

1 Introduction

The distribution of fitness effects (DFE) of new mutations is central to our understanding of how natural selection shapes genetic variation. Like its predecessors, fastDFE infers the DFE by contrasting the site-frequency spectrum (SFS) of putatively neutral (synonymous or intergenic) and

*Bioinformatics Research Center, Aarhus University, Denmark.

selected (non-synonymous) sites (Keightley and Eyre-Walker, 2008; Galtier, 2016; Tataru et al., 2017). The intuition behind this approach is that beneficial alleles are more likely to segregate at higher frequencies than deleterious ones, implying that the SFS provides information on the fitness of new mutations. Summarizing all mutations' fitness naturally gives rise to the distribution of fitness effects (DFE), which essentially is a probability distribution of the selection coefficients of mutations at selected sites.

To account for distortions to the selected SFS due to demography, the neutral SFS is commonly used, which is assumed to be subject to demography alone. Demography is either modeled explicitly, or by the introduction of nuisance parameters which are fit so as to re-scale the observed neutral SFS to match the standard Kingman SFS (Keightley and Eyre-Walker, 2008; Tataru et al., 2017). After accounting for demography, the remaining distortions to the selected SFS are assumed to be entirely due to selection.

SFS-based methods for estimating the DFE summarize population genetic variation by quantifying the number of alleles at specific frequencies, discarding details about the specific sites where these frequencies are exhibited. As a result, we obtain one DFE for all variants collectively. This approach provides limited information when considering a single DFE, but its utility significantly increases when comparing DFEs across multiple species, populations, or genomic regions (Moutinho et al., 2022; Chen et al., 2021; Latrille et al., 2023). Existing methods that can fit more than one SFS at a time are either notoriously slow and sometimes prone to numerical instabilities (Tataru and Bataillon, 2019), or offer only limited parameter sharing capabilities (Galtier, 2016). Here we introduce fastDFE which is specifically designed to facilitate and encourage the joint inference of multiple DFEs at once, allowing for the introduction of genomic covariates, and covering the entire workflow of DFE inference from the moment of acquisition of a raw VCF file.

2 Model & Enhancements

fastDFE primarily builds upon the model of polyDFE, overcoming its constraints such as extensive computational time and limited scalability (Tataru et al., 2017; Tataru and Bataillon, 2019). As such, it provides the same functionalities such as parametric bootstrapping, customizable DFE parametrizations, nested model comparison and joint inference. Similarly, the impact of demography

is accounted for by introducing nuisance parameters (A.2).

To obtain the expected SFS given mutations having a specific selection coefficient, fastDFE makes use of the expected allele frequency sojourn times from Poisson random field (PRF) theory (Sawyer and Hartl, 1992; Sethupathy and Hannenhalli, 2008) (A.1). Let $\mathbb{E}[P_{\text{neut}}(i, S)]$ denote the expected number of derived alleles at frequency i given (population-scaled) selection coefficient S (Figure A.1). Obtaining the expected SFS given a specific DFE requires integrating the expected SFS counts over the DFE, i.e., $\int_{-\infty}^{\infty} \mathbb{E}[P_{\text{sel}}(i, S)]\phi_r(S)dS$, where $\phi_r(S)$ denotes a density describing the DFE with regards to some chosen parametrization r. The actual result, i.e., inferring the DFE given an SFS is then achieved by maximum-likelihood optimization with respect to r, where a composite Poisson likelihood for the SFS is used (A.4).

In fastDFE, the above integral is discretized, precomputed and cached, leading to significant performance improvements over polyDFE while producing very similar results (Table B.1, Figures B.1 & B.2). One issue with polyDFE is its numerical integration procedure, which has difficulties to properly capture the DFE's mass in regions where it is highly concentrated. One example of this is the oft-used Γ distribution with low shape parameters. fastDFE circumvents this problem by describing the DFE by means of its cumulative distribution function (CDF), thus providing more reliable estimates (A.5).

When inferring DFEs from multiple datasets, it is essential to standardize the derivation of the SFS to ensure results are directly comparable. To aid this process, fastDFE comes equipped with a VCF parser, enabling the extraction of the necessary SFS input data from raw VCF files. In order to obtain multiple SFS, a single VCF file can optionally be stratified with regards to some genomic properties, the stratification into a neutral and selected SFS being one example. To further aid the data preparation process, sites can be filtered and annotated with regards to degeneracy and ancestral state. Distinguishing between ancestral and derived alleles is important as it greatly improves the accuracy of DFE estimates, especially for beneficial mutations. The implemented ancestral allele annotation utility uses outgroup information, and follows the probabilistic model of EST-SFS (Keightley and Jackson (2018)). At last, utilities are provided to determine the number of mutational target sites when monomorphic sites are not present in the VCF file at hand.

3 Examples

3.1 Basic inference

In the following Python code snippet, we demonstrate how to use fastDFE to infer the DFE for a single population. Firstly, we define the spectra for both neutral (`sfs_neut`) and selected (`sfs_sel`) sites for a sample size of $n = 8$, leveraging data from a Scandinavian silver birch study (Sendrowski (2022)). We also parametrize the DFE using `GammaExpParametrization`—a common parametrization consisting of a mixture of a gamma and an exponential distribution for deleterious and beneficial selection coefficients, respectively. We use this information to instantiate a `BaseInference` object, which serves to infer a single DFE.

```python
import fastdfe as fd

# create inference object
inf = fd.BaseInference(
    sfs_neut=fd.Spectrum([66200, 410, 120, 60, 42, 43, 52, 65, 0]),
    sfs_sel=fd.Spectrum([281937, 600, 180, 87, 51, 43, 49, 61, 0]),
    model=fd.GammaExpParametrization(),  # the model to use
    n_runs=10,  # number of optimization runs
    n_bootstraps=100,  # number of bootstrap replicates
    do_bootstrap=True
)

# run inference
inf.run()
```

Upon carrying out the inference process, we proceed to visualize the results (Figure 1). Note that an equivalent R script would have significant similarities to the provided Python example.

```python
import matplotlib.pyplot as plt
```

```python
# create subplots
axs = plt.subplots(2, 2, figsize=(11, 7))[1].flatten()

# plot results
types = ['neutral', 'selected']
inf.plot_sfs_comparison(ax=axs[0], show=False, sfs_types=types)
inf.plot_sfs_comparison(ax=axs[1], show=False, colors=['C1', 'C5'])
inf.plot_inferred_parameters(ax=axs[2], show=False)
inf.plot_discretized(ax=axs[3], show=True)
```

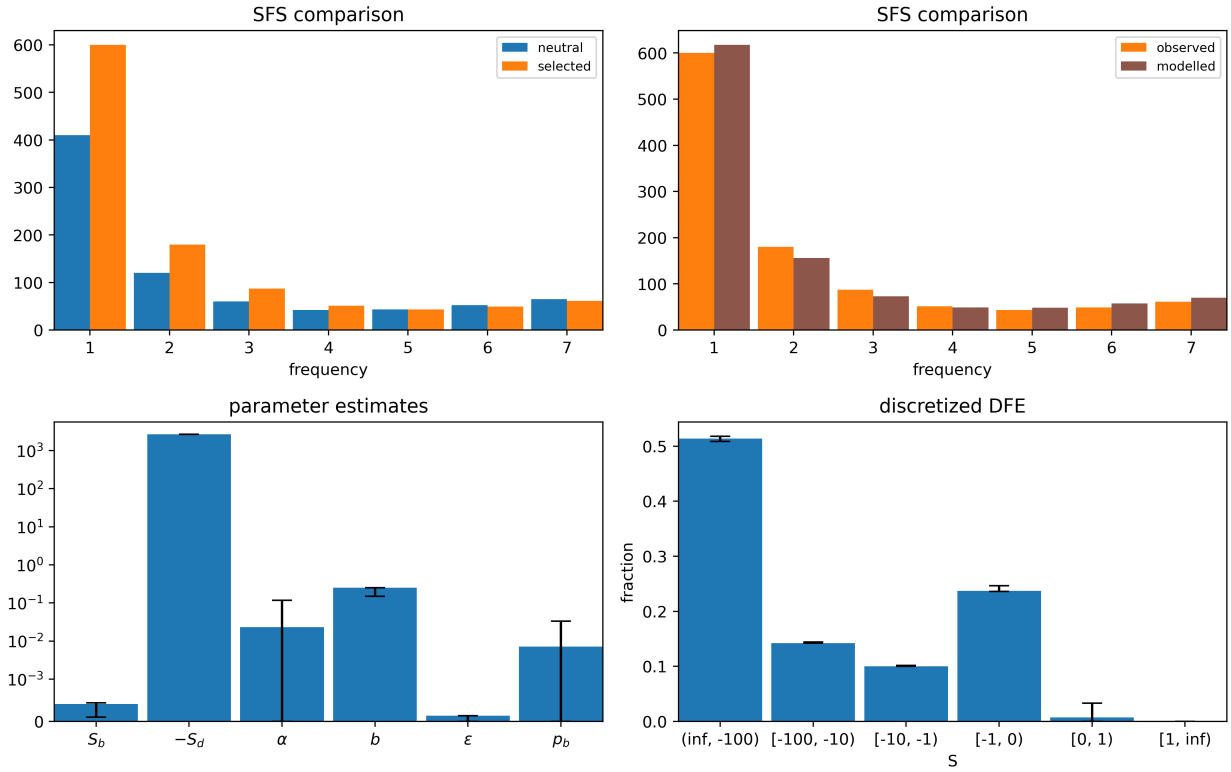

Figure 1: Output from code snippet in Section 3.1. **Top left**: observed neutral and selected SFS. **Top right**: observed (selected) and modelled (selected) SFS. **Bottom left**: parameter estimates. For `GammaExpParametrization`, S_d and S_b represent the average strength of deleterious and beneficial mutations, respectively. Parameter b denotes the shape of the gamma distribution and p_b the probability of a mutation being beneficial, whereas ϵ is the ancestral misidentification parameter (A.3). α denotes the expected proportion of beneficial non-synonymous substitutions, and is not a real parameter, but rather a function of the inferred DFE. **Bottom right**: discretized DFE estimate; The vertical bars indicate 95% confidence intervals.

3.2 VCF parsing

fastDFE offers a versatile parser extracting SFS from VCF files while allowing for stratifications of the SFS. The stratified spectra can be directly used for either joint or marginal DFE estimation. In

the following code example, we obtain the SFS used in Section 3.1. We determine the site-degeneracy on-the-fly by passing `DegeneracyAnnotation`, which requires both a FASTA and a GFF file to be specified. `DegeneracyStratification` is then used to stratify the SFS into *selected* and *neutral* sites. Ancestral alleles are determined by passing `MaximumLikelihoodAncestralAnnotation`. Since the specified VCF file only contains biallelic sites, we also provide `TargetSiteCounter`, which determines the number of monomorphic sites for each stratification by sampling sites from the provided FASTA file. The resulting SFS are depicted in Figure C.1.

```
basepath = ("https://github.com/Sendrowski/fastDFE/"
            "blob/dev/resources/genome/betula/")

# instantiate parser
p = fd.Parser(
    n=8,  # SFS sample size
    vcf=(basepath + "biallelic.with_outgroups."
                    "subset.50000.vcf.gz?raw=true"),
    fasta=basepath + "genome.subset.1000.fasta.gz?raw=true",
    gff=basepath + "genome.gff.gz?raw=true",
    target_site_counter=fd.TargetSiteCounter(
        n_target_sites=350000  # total number of target sites
    ),
    annotations=[
        fd.DegeneracyAnnotation(),  # determine degeneracy
        fd.MaximumLikelihoodAncestralAnnotation(
            outgroups=["ERR2103730"],  # use one outgroup
            n_ingroups=20  # subsample size
        )
    ],
    stratifications=[fd.DegeneracyStratification()]
)
```

```
# obtain SFS
spectra: fd.Spectra = p.parse()

spectra.plot()
```

3.3 Joint inference

In comparative analyses, the ability to stratify and associate site-frequency spectra with covariates (such as genomic features or species-specific properties) can prove very useful. Joint inference enables the sharing of an arbitrary number of parameters across multiple datasets, henceforth referred to as types. Parameters that are not shared are estimated marginally, i.e., for each type separately. However, when using many types with marginal parameters, the large number of parameters to be optimized can make it difficult to find good optima. Covariates address this by introducing a customizable, typically linear, relationship with a specific parameter across all types. This reduces the number of additional parameters to one per covariate, and allows for the testing of covariate significance through a likelihood ratio test against the null model, where the parameter in question is constant across all types.

In the example below, we access a configuration file containing data from a study testing the adaptive walk model of gene evolution in *A. thaliana* (Moutinho et al., 2022). The underlying premise is that younger genes, due to their farther distance from their fitness optimum, are more likely to experience mutations with large fitness effects compared to older genes. The study provides site-frequency spectra stratified by gene, which are associated with a number of proxies for gene age. Notably, we focus on the relative solvent accessible (RSA) as our proxy of interest. The spectra for each gene, after being downsampled to 20 individuals, were grouped into 10 quantiles based on their RSA score. The covariate for each bin was taken to be the mean RSA score over all its genes. A linear relationship is then introduced with S_d—the average strength of deleterious mutations (cf. this script). All remaining parameters are specified to be shared (i.e. constant) across all types. The significance of the included covariate is determined by comparing the likelihood against the

null model in which S_d is constant across all bins (Figure 2).

```
inf = fd.JointInference.from_config_file(
    "https://github.com/Sendrowski/fastDFE/"
    "blob/dev/resources/configs/arabidopsis/"
    "covariates_example.yaml?raw=true"
)

inf.run()

# get p-value for covariate significance
p = inf.perform_lrt_covariates()
p_str = f"p = {p:.1e}" if p >= 1e-100 else "p < 1e-100"

# plot results
_, axs = plt.subplots(1, 2, figsize=(10.5, 3.5))

inf.plot_covariate(ax=axs[0], xlabel='RSA', show=False)
inf.plot_discretized(
    title=f"DFE comparison, " + p_str, ax=axs[1],
    show_marginals=False, show=True
)
```

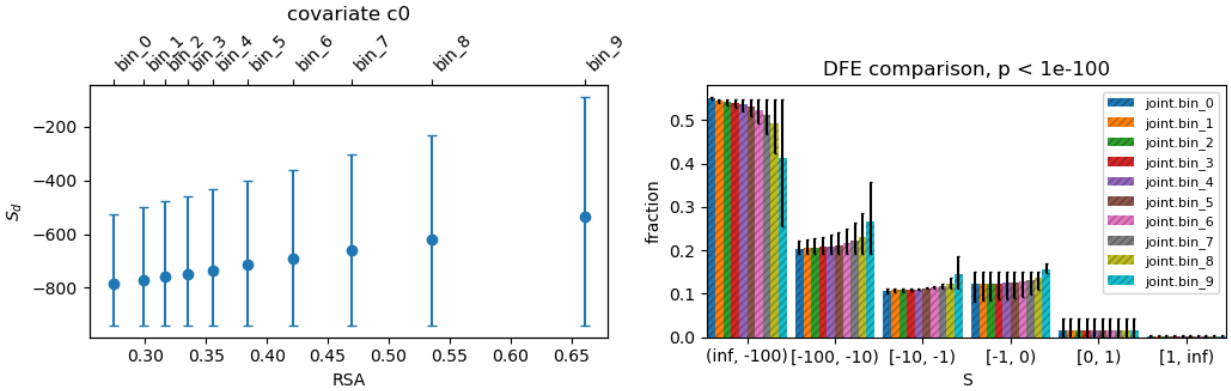

Figure 2: Output from code snippet in Section 3.3. **Left**: values of S_d vs. RSA score for the different bins. S_d is restricted to vary linearly with RSA. **Right**: inferred joint DFE. The low p-value (indicated in the plot title) confirms the high significance of covariate inclusion; Vertical bars indicate 95% confidence intervals.

4 Discussion

fastDFE aims to improve on existing DFE estimation software, particularly in the realm of joint inference and data preparation. It offers a very efficient solution for joint inference—both performance-wise and by allowing for the introduction of covariates. It is thus well-suited for comparative analyses, whether across species, populations, or varying genomic contexts. The VCF parsing and annotation features greatly facilitate the data preparation process, potentially minimizing data handling errors while making different datasets more comparable. Customizable DFE parametrizations by means of the cumulative distribution function (CDF) also offer increased flexibility and reliability. The availability of a Python and R interface instead of a traditional command line approach aims to make DFE estimation more customizable. In conclusion, we hope that fastDFE will contribute to new insights in evolutionary biology, ecology, and conservation—especially within the scope of comparative studies. The software is freely available at `github.com/Sendrowski/fastDFE`, and comprehensive documentation as well as usage examples can be found at `fastdfe.readthedocs.io`.

Acknowledgements

We would like to thank Sylvain Glémin for discussion on model implementation and Julien Dutheil for his helpful comments on the manuscript. This work has been supported by the Novo Nordisk Foundation (Data Science Collaborative Research Programme, grant 0069105). We also acknowledge Martin Lascoux and Luis Leal (Formas, grant 2020-01456) for kindly providing the birch dataset.

References

Jun Chen, Thomas Bataillon, Sylvain Glémin, and Martin Lascoux. Hunting for Beneficial Mutations: Conditioning on SIFT Scores When Estimating the Distribution of Fitness Effect of New Mutations. *Genome Biology and Evolution*, 14(1):evab151, 06 2021. ISSN 1759-6653. doi: 10.1093/gbe/evab151. URL `https://doi.org/10.1093/gbe/evab151`.

Nicolas Galtier. Adaptive protein evolution in animals and the effective population size hypothesis. *PLoS genetics*, 12:e1005774, 01 2016. doi: 10.1371/journal.pgen.1005774.

Peter Keightley and Adam Eyre-Walker. Joint inference of the distribution of fitness effects of deleterious mutations and population demography based on nucleotide polymorphism frequencies. *Genetics*, 177:2251–61, 01 2008. doi: 10.1534/genetics.107.080663.

Peter Keightley and Benjamin Jackson. Inferring the probability of the derived versus the ancestral allelic state at a polymorphic site. *Genetics*, 209:genetics.301120.2018, 05 2018. doi: 10.1534/genetics.118.301120.

Thibault Latrille, Julien Joseph, Diego Hartasánchez, and Nicolas Salamin. Mammalian protein-coding genes exhibit widespread beneficial mutations that are not adaptive. bioRxiv, 05 2023.

Ana Moutinho, Adam Eyre-Walker, and Julien Dutheil. Strong evidence for the adaptive walk model of gene evolution in drosophila and arabidopsis. *PLoS biology*, 20:e3001775, 09 2022. doi: 10.1371/journal.pbio.3001775.

S A Sawyer and D L Hartl. Population genetics of polymorphism and divergence. *Genetics*, 132 (4):1161–1176, 12 1992. ISSN 1943-2631. doi: 10.1093/genetics/132.4.1161. URL `https://doi.org/10.1093/genetics/132.4.1161`.

Janek Sendrowski. Demography of birch populations across scandinavia, 2022. URL `https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1675636&dswid=5265`.

Praveen Sethupathy and Sridhar Hannenhalli. A tutorial of the poisson random field model in population genetics. *Advances in bioinformatics*, 2008:257864, 01 2008. doi: 10.1155/2008/257864.

Paula Tataru and Thomas Bataillon. polyDFEv2.0: testing for invariance of the distribution of

fitness effects within and across species. *Bioinformatics*, 35(16):2868–2869, 01 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty1060. URL `https://doi.org/10.1093/bioinformatics/bty1060`.

Paula Tataru, Maeva Mollion, Sylvain Glémin, and Thomas Bataillon. Inference of distribution of fitness effects and proportion of adaptive substitutions from polymorphism data. *Genetics*, 207: genetics.300323.2017, 09 2017. doi: 10.1534/genetics.117.300323.