Supplemental Information for

`Dsuite` - fast *D*-statistics and related admixture evidence from VCF files

Milan Malinsky[1,*], Michael Matschiner[2,3], and Hannes Svardal[4,5]

[1]Zoological Institute, University of Basel, Basel, Switzerland; [2]Department of Paleontology and Museum, University of Zurich, Zurich, Switzerland; [3]Department of Biosciences, University of Oslo, Oslo, Norway; [4]Department of Biology, University of Antwerp, Antwerp, Belgium; and [5]Naturalis Biodiversity Center, Leiden, The Netherlands.

*Correspondence author. E-mail: millanek@gmail.com

Simulations for computational performance assessment

To assess computational efficiency, we produced two test datasets ("small" and "large") performing neutral coalescent simulations using `msprime 0.7.4` (Kelleher et al. 2016). To produce the input data for and convert the output of msprime, we used custom python functions available in the `pypopgen3` repository (https://github.com/feilchenfeldt/pypopgen3/; last accessed 30/07/2020, commit 759570fb58819ff6042d83d542bcd37842164fa5). In particular, we first produced random trees using the function `treetools.get_random_geneflow_species_tree` with options `n_species=20, n_geneflow_events=5` and `n_species=100, n_geneflow_events=10` for the small and large simulation set, respectively. The sample size of each species was set to two diploid individuals by `sample_sizes=4` and the `ete3` python package was used to produce random trees (option `random_tree_process='ete3'`). The other parameters were left at their default values, which means:

- The function produced a species tree with random branch lengths and sample relationships, an initial ingroup-divergence of 1 million years ago (mya) and a divergence of the ingroup from an additional outgroup lineage 2 mya.
- For each gene flow event, a time was chosen uniformly in the interval [1e6, 0], and a donor and a recipient branch were drawn from all the branches existing at this time point. The strength of the gene flow event was determined by `numpy.random.beta(2, 5) * 0.3`.
- Effective population sizes for each branch were set 50 000, except for the outgroup branch; effective population size for the outgroup was set to 1 to ensure that the outgroup was fixed for alleles segregating in the ingroups

We then used the gene flow tree as input to `msprime` directly producing VCF files via the `pypopgen3 simulate` module

```
(samples,
population_configurations,
sorted_events,
sample_to_pop, sample_names,
             id_to_name) = simulate.msprime_input_from_split_tree(tree)

tree_sequence = simulate.simulate_to_vcf(vcf_fn_chrom,
                                         samples,
                                  population_configurations,
                                         sorted_events,
                                         mutation_rate,
                                  recombination_rate,
                                  genomic_length,
                                         sample_names=sample_names,
                                  chrom_id=chrom_id  )
```

Using the above functions, we simulated 20 and 100 chromosomes for the small and large simulation runs, respectively, each of a length of one Megabase, using per site mutation and

recombination rates of 1e-8. The two simulation runs resulted in totals of 4,342,77 and 97,201,601 single nucleotide polymorphisms (SNPs), respectively (Table 2). For the first run, all VCF files were concatenated using `bcftools concat` before running the different *D* statistic calculation tools, while for the large dataset tools were run on each of the 100 chromosomes individually. See Table 3 for results and Fig. 3.

Simulations for *f*-branch power assessment

In order to assess how often *f*-branch is able to correctly identify a gene flow event, we performed coalescent simulations on random trees similar to above, but with a single gene flow event, different combinations of gene flow strength (0.01, 0.025. 0.05. 0.1, 0.2) and different numbers of SNPs used as input for `Dsuite` (1e4, 1e5, 1e6), producing 100 replicates per parameter combination. We then assessed whether the strongest signal in the *f*-branch matrix is consistent with the gene flow receptor and donor.

In particular, random species trees were produced using:
`dendropy`'s `treesim.birth_death_tree` function
with options birth_rate=1, death_rate=0, num_extant_tips=n_species, gsa_ntax=n_species+1 and `random_tree_process='birth_death'`

Gene flow events were added using the function
`treetools.get_random_geneflow_species_tree`
with options `n_species=30`, `n_gene_flow_events=1`, `sample_sizes=2` (single diploid individuals), and a variable gene flow strength between 0.01 and 0.2 (see Fig. 4).

In order to specifically assess the performance of f-branch in identifying internal branches that receive gene flow (which is not possible with regular *D* statistics or f_4-ratios), we produced for each parameter combination 100 replicates where the gene flow recipient was a terminal branch, and 100 replicates where the gene flow recipient was an internal branch.

Simulations were run using the same parameters and functions as in the simulation to assess computational performance of Dsuite (`simulate.msprime_input_from_split_tree, simulate.simulate_to_vcf`), but, here, a single chromosome of 10 Megabases was simulated per replicate. This yielded VCF files with ~ 2.5 million SNPs per replicate. We then ran `Dsuite Dtrios` and `Dsuite Fbranch` three times on each replicate, providing the original simulation tree, and using the `--region=1,{n_snps}` option of `Dsuite Dtrios` to restrict the analyses to 1e4, 1e5, and 1e6 SNPs, respectively.

We then assessed, for each replicate, whether the recipient and donor were correctly identified in the following way:

- A recipient branch was scored as correct if the row (*b*, "y-axis") of the maximum *f*-branch signal corresponded to the gene flow recipient branch.
- A donor was scored as correct if the column (P3, "x-axis") of the maximum *f*-branch signal corresponded to the gene flow donor branch or its descendant. (The latter is necessary because there are no internal branches among the columns of *f*-branch. Internal donor branches cannot be directly identified.)
- Since gene flow into a branch also results in excess allele sharing of its sister branch with their closest relatives, we also assessed cases where the row of the maximum *f*-

branch signal corresponded to the sister branch of the true gene flow recipient. These cases are shown with light colour shades in Fig. 4a,b..

- Conversely, cases where the column of the maximum *f*-branch signal corresponded to a descendant of the sister branch of the true donor are shown in light colour shades in Fig. 4c,d.

Using genotype likelihoods or probabilities in allele frequency estimates

`Dsuite` estimates allele frequencies for each biallelic SNP and each population by default from the called genotypes in the VCF file (the GT field). In addition, we provide an option to use genotype probabilities (GP field) produced for example by genotype imputation software such as BEAGLE, or genotype likelihoods (either GL or PL fields) produced by variant callers such as GATK.

For each biallelic SNPs and population, we estimate $\widehat{AF}_{GP}$, the alternative allele frequency from genotype probabilities as:

$$\widehat{AF}_{GP} = \frac{\sum_{i=1}^{n} P(G_i = 1|D) + 2 * P(G_i = 2|D)}{2n}$$

where, for each individual i among $\{1 \dots n\}$ sampled from a population, $P(G_i = 1|D)$ denotes the posterior probability of a heterozygous genotype, and $P(G_i = 2|D)$ is the posterior probability of the genotype homozygous for the alternative allele.

We base genotype priors for each individual on estimates of allele frequency obtained from called genotypes ($\widehat{AF}_{GT}$) and assume Hardy-Weinberg Equilibrium. Therefore, we set the genotype priors as follows:

$$P(G_i = 0) = \left(\widehat{AF}_{GT}\right)^2$$
$$P(G_i = 1) = \widehat{AF}_{GT} * \left(1 - \widehat{AF}_{GT}\right)$$
$$P(G_i = 2) = \left(\widehat{AF}_{GT}\right)^2$$

where $P(G_i = 0)$ denotes the prior probability for the genotype homozygous for the reference allele, $P(G_i = 1)$ for the heterozygous genotype, and $P(G_i = 2)$ for the genotype homozygous for the alternative allele.

Finally, we obtain the posterior genotype probabilities from genotype likelihoods by calculating for each possible genotype $g = \{0,1,2\}$

$$P(G_i = g|D) = \frac{P(D|G_i = g) * P(G_i = g)}{\sum_g P(D|G_i = g) * P(G_i = g)}$$

where $P(D|G_i = g)$ denotes the likelihood of genotype g.