User manual for *SURFDAWave*

Mehreen R. Mughal, Hillary Koch, Jinguo Huang, Francesca Chiaromonte, Michael DeGiorgio

November 6, 2019

Contents

1 Introduction

The *SURFDAWave* software package can be used to train functional multinomial and functional multi-response linear regression models to first identify selective sweeps and then learn selection parameters. *SURFDAWave* can be used to differentiate between sweeps and neutrality and then learn the selection coefficient, time of selection, and initial frequency of a mutation before it became beneficial. However, the flexible framework of this software allows the user to provide an arbitrary number of classes as well as an arbitrary number of dependent variables for prediction. We also include scripts with which users can calculate summary statistics $\widehat{\pi}$, H_1, H_{12}, H_2/H_1, and the frequency of the first to the fifth most common haplotypes as described in Mughal *et al.* (2019; *bioRxiv*). Operation of this package requires a UNIX environment with Python 2.7 and R.3.4.1.

Please cite this software as

MR Mughal, H Koch, J Huang, F Chiaromonte, and M DeGiorgio (2019) Learning the properties of adaptive regions with functional data analysis. *bioRxiv* doi:XXXXX

If you experience any issues, then please contact Mehreen Mughal at mrm79@psu.edu for further help.

2 Operation

2.1 Installation

To download *SURFDAWave*, visit

```
http://degiorgiogroup.fau.edu/surfdawave.html
```

This download will include the user manual, software, and example datasets. The scripts included are designed to perform on a UNIX system. To unpack `surfdawave` from the command line, go to the directory where it is stored, and enter

```
tar -xzvf surfdawave.tar.gz
cd surfdawave/
```

The first command will decompress the file and release the content into folder `surfdawave/` in the current directory. The second command will shift the current directory to the `surfdawave/` directory, which contains the manual, as well as the `exampledata/` and `examplefvecs/` directories.

There are a number of required packages for this software to run correctly. The `wavethresh` (Nason, 2008), and `glmnet` (Friedman et al., 2010) packages will need to be installed by the user, along with R package `matrixStats`. These can all be installed by using the following commands in R.

```
install.packages(‘‘wavethresh’’)

install.packages(‘‘glmnet’’)

install.packages(‘‘matrixStats’’)
```

2.2 Calculating summary statistics from *ms*-formatted input files

Files in *ms* (Hudson, 2002) format typically contain multiple blocks of sequence data for a set of samples at segregating sites, where each block is typically a simulated replicate when based on simulated data. To calculate summary statistics from *ms*-formatted input files, use the command

```
python calcstats.py <input_file>
```

This command will output a file `<input_file>.stats` in the same directory that `<input_file>` is located. The output file is CSV-formatted, and contains for each sequence block (*e.g.*, simulated replicate) a line with 1152 comma-separated columns. The 1152 columns represent the nine summary statistics ($\widehat{\pi}$, H_1, H_{12}, H_2/H_1, and the frequency of the first to the fifth most common haplotype) computed in 128 windows ($9 \times 128 = 1152$) used as default by *SURFDAWave*, as in Mughal et al. (2019).

Specifically, for each line, $\widehat{\pi}$ has values located in columns $1, 10, 19, \ldots, 1144$, H_1 has values located in columns $2, 11, 20, \ldots, 1145$, H_{12} has values located in columns $3, 12, 21, \ldots, 1146$, H_2/H_1 has values located in columns $4, 13, 22, \ldots, 1147$, frequency of the most common haplotpye has values located in columns $5, 14, 23, \ldots, 1148$, frequency of the second most common haplotpye has values located in columns $6, 15, 24, \ldots, 1149$, frequency of the third most common haplotpye has values located in columns $7, 16, 25, \ldots, 1150$, frequency of the fourth most common haplotpye has values located in columns $8, 17, 26, \ldots, 1151$, and frequency of the fifth most common haplotpye has values located in columns $9, 18, 27, \ldots, 1152$.

For *ms*-formatted input files, a single set of nine summary statistics across 128 windows is computed at the center of each block of sequences (*e.g.*, simulated replicate). Due to the manner in which summary statistics are computed in windows, the *ms*-formatted input file should contain at least 645 segregating sites in each sequence block (*e.g.*, simulated replicate). If a sequence block (*e.g.*, simulated replicate) does not have at least 645 segregating sites, then summary statistics will not be calculated for that block.

2.3 Calculating summary statistics from VCF files

To calculate summary statistics from data in VCF format (Danecek et al., 2011), use the command

```
python calcstat_emp.py <input_file>
```

This command will calculate summary statistics on the data from the `<input_file>`. Specifically, we first divide all the data into 10 SNP windows, each overlapping its neighbor by 5 SNPs. For each window we calculate the nine summary statistics described above. The final `<input_file>.stats` file will contain feature vectors where each line contains information from 128 contiguous windows. The `<input_file>.sites` output file will contain the same number of lines as the `<input_file>.stats` file and each line contains the genomic position of the corresponding lines feature vector.

2.4 Training classifier

In general, to train the *SURFDAWave* classifier, use the command

```
Rscript FDAclass.R <file_name> <num_stats>
```

where `<file_name>` contains summary statistics data for all classes you want to differentiate among, and `<num_stats>` is the number of different summary statistics m.

If the user chooses to use the default $m = 9$ summary statistics ($\widehat{\pi}$, H_1, H_{12}, H_2/H_1, and frequency of the first to the fifth most common haplotype), then use the command

```
Rscript FDAclass.R <file_name> 9
```

where `<file_name>` is a summary statistic file that was output from `calcstats` function (section 2.2) followed by a class label.

This `train` function will fit a multinomial logistic regression model by performing 10-fold cross validation across a grid of values to choose the elastic net penalty and the scale or detail level of the discrete wavelet transform. The discrete wavelet transform is conducted in the same manner as described in (Zhao et al., 2012). After training, this function will output a single file: `<file_name_minscale.rds>` containing the model. This `<.rds>` model file will be used to make predictions on test data. Prior to fitting the model, this function will standardize the summary statistics in `<file_name>`. Here, the `minscale` is the scale chosen through cross validation by the model. The original training data file along with the `<.rds>` model file will be necessary to apply *SURFDAWave* to test data.

2.5 Training predictor

Training of the predictor model is conducted in a similar manner to the training of a classifier as described in section 2.4 by using the command

```
Rscript FDAparam.R <file_name> <num_stats>
```

where `<file_name>` is the name of the file containing the summary statistics used to train the model, `<num_stats>` is the number of summary statistics, and the final columns are the dependent variables. For example, if the user would like to predict the time of selection, the initial frequency, and the selection strength, then the last three columns (column numbers 1153, 1154, and 1155, if using nine summary statistics calculated in 128 windows) should contain these parameter values.

2.6 Testing classifier

To apply the *SURFDAWave* classifier to a test dataset, use the command

```
Rscript predclass.R <test_file> <model_name> <num_stats>
```

where `<test_file>` is a summary statistic file, `<model_name>` is the name of the classifier assigned in section 2.4, and `<num_stats>` is the number of different summary statistics in the test file. Note that the test file must be formatted in the same manner as the file used to train the model. The `<test_file>` can be computed from *ms*-formatted data by using the `calcstat` command (section 2.2), from VCF-formatted data by using the `calcstat_emp` command (section 2.3), or can instead be computed by the user via the format specified in section 2.4 containing length p windows in which $\log_2(p)$ is a non-negative integer.

This function will standardize the summary statistics in `<test_file>` and output a file (`<test_file.pred>`) containing the predicted class for each test site (line of `<test_file>`) using the classifier specified by input `<model_name>`.

2.7 Testing predictor

Similarly, to apply the *SURFDAWave* predictor to a test dataset, use the command

```
Rscript predparam.R <test_file> <model_name> <num_stats>
```

This function will standardize the summary statistics in `<test_file>` and output a file (`<test_file.pred>`) containing the predicted selection parameters for each test site (line of `<test_file>`) using the model specified by input `<model_name>`.The number of predictions per line corresponds to the number of parameters the model was trained to predict.

3 Examples

We provide example *ms*-formatted simulated training data generated by `SLiM` (Haller and Messer, 2017) of neutral scenarios for users to test the functionality of *SURFDAWave*. The file entitled `SweepMSout` is located in the sub-directory `examplefvec/`, and contains five independent simulated replicates of sweep scenarios simulated under an African demographic history, with model parameters under as detailed in Mughal et al. (2019).

If the user is only interested in quickly testing the training functionality of *SURFDAWave*, then we provide smaller datasets entitled `Sweep_Neut.stats` and `Sweep_pred.stats` in which summary statistics have been pre-computed and are located in sub-directory `exampledata/`. We also provide a small portion of human chromosome 22 from 100 haplotypes sampled from the African Yoruban (YRI) population of the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015) in VCF file format.

To calculate summary statistics on the sample simulated file, move into the `<examplefvecs/>` directory and use the following command:

```
python calcstat.py TESTMS.ms
```

This command will output a file containing the summary statistics calculated for the simulated file titled `<input_file>.stats` and will be located in sub-directory `examplefvecs/`, which is the directory of `<input_file>`.

To calculate summary statistics from the example VCF-formatted file provided, use the command

```
python calcstat_emp.py chr22TEST.vcf
```

This command will output a file `chr22TEST.vcf.stats` located in sub-directory `examplefvecs/` containing the summary statistics calculated from the provided region of human chromosome 22 from 100 YRI haplotypes, along with a file `chr22TEST.vcf.sites` containing the genomic position for each feature vector.

To train *SURFDAWave* classifier to differentiate between neutrality and sweep scenarios using the provided summary statistic files from the smaller dataset, move into the `exampledata/` subdirectory and use the command

```
Rscript FDAclass.R  Sweep_Neut 9
```

This command will output a model named `Sweep_Neut.0.rds`, in which 0 denotes the level (scale) chosen through cross validation.

Similarly, we can train a prediction model by using the command

```
Rscript FDApred.R  Sweep_pred 9
```

This command will output a model named `Sweep_pred.1.rds`, in which 1 denotes the level (scale) chosen through cross validation.

Using the previously-trained classifier `Sweep_Neut.0.rds`, we can classify a test dataset containing summary statistics computed for sweep simulations using the command

```
Rscript predclass.R teststats Sweep_Neut.0.rds 9
```

This command will output a result file `teststats.predclass` containing the predicted class and probabilities of the classes used to train `Sweep_Neut.0.rds` for each the 200 simulated datasets. The output file teststats.predclass will be located in the same directory as the model.

Using the trained prediction model we can predict the selection parameters for the same test dataset.

```
Rscript predparam.R teststats Sweep_sels.1.rds 9
```

This command will output a result file teststats.predparam that will contain three predicted parameters for each of the 200 simulations.

Again, using the previously-trained classifier and predictor, we can classify the small region provided for human chromosome 22 based on the set of summary statistics using the commands

```
Rscript predclass.R ../examplefvec/chr22TEST.vcf.stats Sweep_Neut.0.rds 9

Rscript predparam.R ../examplefvec/chr22TEST.vcf.stats Sweep_sels.1.rds 9
```

These commands will output a result file chr22TEST.vcf.stats.predclass in which the first column contains the predicted class, followed by the probabilities for each class. The output file chr22TEST.vcf.sites contains the position information for each line in the prediction file.

References

P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, G. McVean, R. Durbin, and . G. P. A. Group. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22, 2010.

B. C. Haller and P. W. Messer. SLiM 2: flexible, interactive forward genetic simulations. *Molecular Biology and Evolution*, 34:230–240, 2017.

R. R. Hudson. Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.

M. R. Mughal, H. Koch, J. Huang, F. Chiaromonte, and M. DeGiorgio. Learning the properties of adaptive regions with functional data analysis. *biorxiv*, 2019.

G. P. Nason. *Wavelet Methods in Statistics with R*. Springer, New York, NY, 1st edition, 2008.

The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526:68–74, 2015.

Y. Zhao, R. T. Ogden, and P. T. Reiss. Wavelet-based lasso in functional linear regression. *Journal of computational and graphical statistics*, 21:600–617, 2012.