forestDNM: *de novo* SNV discovery with Random Forests

Jake Michaelson

May 30, 2013

Introduction

`forestDNM` is an R package with a command-line executable that automates the discovery of *de novo* SNV mutations (DNMs) from family genotype data (provided in a VCF). All you need to get started is a VCF.

It works by using a classifier that has been trained to discriminate putative DNMs that were experimentally validated from putative DNMs that were invalidated, using combinations of quality metrics and other features constructed from a typical VCF produced by GATK. The result is a tool that can distill the few true DNMs from tens of thousands of candidates, with high sensitivity and specificity.

The original idea behind forestDNM was first introduced in Michaelson, et al.[1], where we used a Random Forest classifier to predict germline DNMs using families with monozygotic (MZ) twins. True DNMs detected in both identical twins can reasonably be assumed to be germline (rather than somatic) DNMs. However, in the context of trios, the results of forestDNM will include both germline and somatic DNMs (which would include cell line artifacts), as there is no reasonable way to distinguish between the two given only trio data.

In Michaelson, et al.[1] our code was tailored to the unusual scenario of quad families with MZ twins, and consequently we did not at the time have an implementation of forestDNM ready for public consumption. We have since rectified that and we hope that this tool will be useful to other scientists performing genetic studies in trios and families.

Requirements and Assumptions

The package was developed on a 64-bit Linux system with R 2.15.1, and has not been tested on other platforms. Additional required R/Bioconductor packages are `randomForest` and `VariantAnnotation`.

In terms of data, all that is required to run `forestDNM` is a VCF. However, the file must have been generated with GATK 2+ (unified genotyper), and VQSR[2] must have been run. A minimum of a father, mother and child are required in the VCF (if the file contains more family members, these must be run separately – changing `--child-col` accordingly – as `forestDNM` only works on one offspring at a time). In addition, the VCF must be sorted, compressed with `bgzip` and indexed with `tabix`. `forestDNM` assumes that in those VCF fields that contain genotype information, the father is listed first, the mother second, and the child third. If this is not the case, adjust the arguments `--pat-col`, `--mat-col`, and `--child-col` accordingly. It is critical to make sure this is correct, otherwise the results will be nonsensical.

Installation

The `forestDNM` package may be installed via the usual R mechanisms (e.g. `R CMD INSTALL` or `install.packages()`). Once installed, make sure the `forestDNM` executable in the `scripts` subdirectory in the package directory is executable, for example:

```
chmod +x forestDNM
```

The next step is to make sure that this executable can be found in your environment's path. You can either copy it to e.g. `/usr/bin`, or more preferably, create a symbolic link so that the link always points to the most recently installed executable (i.e. the one in the installed package directory):

```
sudo ln -s /home/jacob/R/x86_64/2.15/forestDNM/scripts/forestDNM /usr/bin
```

Note that the `forestDNM` executable assumes that `Rscript` exists in `/usr/bin`. If it does not, change the first line of the script to reflect the path to `Rscript`.

Quick Start

A typical invocation of forestDNM will look something like this:

```
forestDNM --file=myvcf.vcf.gz --sex=F --desc=S123 --genome=hg18
```

The path to the VCF and the sex of the offspring must be specified using the `--file` and `--sex` arguments, respectively. Also, a character string descriptor must be specified (via `--desc`) which will be incorporated into the name of any output files. The reference genome (`hg18` or `hg19`) must be specified with the `--genome` argument. `forestDNM` assumes the ordering of the family members to be father, then mother, then child. If this is not the case, these column numbers must be specified via the `--pat-col`, `--mat-col`, and `--child-col` arguments. The chromosome naming convention must be specified if 'chr' is not prepended to chromosome names in the VCF (e.g. if chromosome 1 is '1' rather than 'chr1', `--chr-conv=NA` should be specified). If you wish to change the default classifier threshold, do so using the '–cutoff' argument. If the features are to be saved, add the `--save-features` flag.

A single run of `forestDNM` (whole genome data, single thread, single trio) will take about 15 minutes (plus or minus, depending on your hardware). Exome data will take considerably less time. The output is a tab-delimited text file that indicates which sites were predicted to be true DNMs, based on the value provided to `--cutoff`. The file contains the chromosome and position, as well as the parental allele (for which both parents have a homozygous genotype) and the mutant allele (the child is heterozygous: 1 parental allele and 1 mutant allele). The classifier score is also provided in the `prd` column, with higher scores indicating higher confidence.

Package Functions

The user only interested in producing DNM calls may stop reading here – the included executable described above fulfills the intent of the tool. For those interested in the inner workings of `forestDNM`, the internal functions of the package and the details of the classifier's training are detailed below.

The package contains the lower-level functions `pDNM`, `pDNM_X_female`, `pDNM_X_male`, `pDNM_Y` that produce features for autosomes, the X chromosome in females, the X chromosome in males, and the Y chromosome, respectively. Each of these takes `vcf` as an argument, which is produced as the output of `readVcf` from the `VariantAnnotation` package. Additional arguments to these functions are `pat.col`, `mat.col`, and `child.col` which specify which column each family member occupies. These functions only return features for sites where the parents are homozygous for the same allele and the child is heterozygous.

The `dnmFeat` function is a wrapper that uses the above functions to produce features for all chromosomes of an individual. The arguments are essentially the same as those passed to the command-line executable: `filename`, `child.sex`, `pat.col`, `mat.col`, `child.col`, `genome`, and `chrom.conv`. The output is a data frame whose columns are the features that are used by the classifier to predict true DNMs (with additional columns for chromosome, position, and parental and mutant alleles).

The `predDNM` function takes a feature matrix `X` (the output of `dnmFeat`) and a trained classifier `rf` (e.g. the one located in the `classifier` package subdirectory) and makes predictions on each site. Additionally, the `cutoff` argument specifies the decision threshold on the Random Forest vote proportion (default is 0.6). Prior to running `predDNM`, rows with NA values should be removed or otherwise dealt with (this is done in the executable):

```
> ind = sapply(X,is.na)
> ind = !rowSums(ind)>0
> X = X[ind,]
```

Predictions can be made by hand (e.g. using the features saved by specifying `--save-features` on the command line) as follows:

```
>  out = predDNM(X,rf,0.6)
```

Classifier Details

The training set consisted of approximately 1 million SNVs from 20 trios (actually 10 quad families where the offspring are MZ twins; each twin was considered separately to yield two trios per family). The SNVs were labeled as validated (936), invalidated (66), or unvalidated (1019922) and a Random Forest classifier was trained as follows:

```
> rf = randomForest(x=Xtr,y=ytr,strata=ss_tr,
        sampsize=c(100000,66,936),importance=T,
        do.trace=T,ntree=100,mtry=3)
```

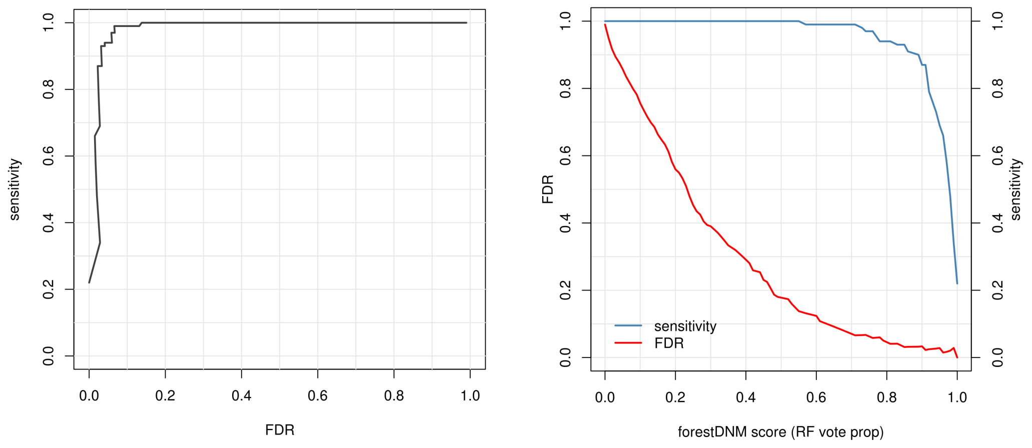

Figure 1: Relationship between sensitivity and FDR (left) and classifier threshold (right). The default threshold for `forestDNM` is 0.6, which in the test set demonstrated sensitivity of 99% and FDR of 10%.

As shown, bootstrapped sampling for each tree was performed in a stratified manner for each validation status. The outcome `ytr` however, only had two labels: validated and not validated. The 'unvalidated' SNVs are enormously enriched for false positives (e.g. discordances between MZ twins with low genotyping quality), so we treat them as equivalent to the 'invalidated' sites, and together they comprise the 'not validated' training label.

Performance

As an independent test set, 10,000 SNVs were withheld from the training set (100 validated DNMs and 9900 'not validated' SNVs). SNVs from co-twins that corresponded to SNVs in the test set were eliminated from the training set to maximize the independence of the two sets. Sensitivity and FDR were measured as the vote proportion threshold was relaxed, and the results are shown in Figure 1.

Features

The predictive features, all derived from metrics contained in the VCF, are described below. Descriptions in quotes are taken directly from information in the VCF header.

- `p_ar_max`: the maximum of the parental ratios of mutant (child) allele reads to parental allele reads

- `p_ar_min`: the minimum of the parental ratios of mutant (child) allele reads to parental allele reads
- `c_ar`: the ratio in the offspring of mutant allele reads to parental allele reads
- `p_cvg_max`: the maximum of parental log2 total coverage ratios, relative to the chromosome median coverage of the individual
- `p_cvg_min`:the minimum of parental log2 total coverage ratios, relative to the chromosome median coverage of the individual
- `c_cvg`: the log2 total coverage ratio of the child (relative to his/her chromosomal median coverage)
- `p_cg_max`: the parental maximum of the phred likelihoods for a heterozygous genotype
- `p_cg_min`: the parental minimum of the phred likelihoods for a heterozygous genotype
- `c_cg`: the child's phred likelihood for a heterozygous genotype
- `p_pg_max`: the parental maximum of the phred likelihoods for a homozygous (parental allele) genotype
- `p_pg_min`: the parental minimum of the phred likelihoods for a homozygous (parental allele) genotype
- `c_pg`: the child's phred likelihood for a homozygous (parental allele) genotype
- `FL`: FILTER, e.g. "PASS", etc.
- `QL`: QUAL, the multi-sample (family) quality score
- `VQ`: VQSLOD, the "log odds ratio of being a true variant versus being false under the trained gaussian mixture model"
- `BZ`: BaseQRankSum, the "Z-score from Wilcoxon rank sum test of Alt Vs. Ref base qualities"
- `DL`: "fraction of reads containing spanning deletions"
- `FS`: "phred-scaled p-value using Fisher's exact test to detect strand bias"
- `HS`: "consistency of the site with at most two segregating haplotypes"
- `MQ`: mapping quality
- `QD`: "variant confidence/quality by depth"
- `PZ`: ReadPosRankSum, the "Z-score from Wilcoxon rank sum test of Alt vs. Ref read position bias"

- `GQ_p_min`: minimum of the parental genotype qualities
- `GQ_p_max`: maximum of the parental genotype qualities
- `GQ_c`: genotype quality for the child
- `N_alt`: number of alternate alleles detected at the site

Feature Importance

Random Forest classifiers include an internal measure of feature importance, which indicates a feature's contribution to predictive accuracy. This is done by randomly permuting each feature and then assessing the resulting degradation of predictive performance for each class, as well as across all classes.

Here we show the feature importance for each class in Figure 2. The family QUAL score and VQSLOD, as well as variant quality by depth, are the most important features for accurate prediction of SNVs that will validate, while features derived from coverage and the genotype likelihoods are the most important features in predicting the 'not validated' class.

References

[1] Michaelson, J. J. *et al.* Whole-genome sequencing in autism identifies hot spots for de novo germline mutation. *Cell* **151**, 1431–42 (2012).

[2] DePristo, M. A. *et al.* A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.* **43**, 491–8 (2011).

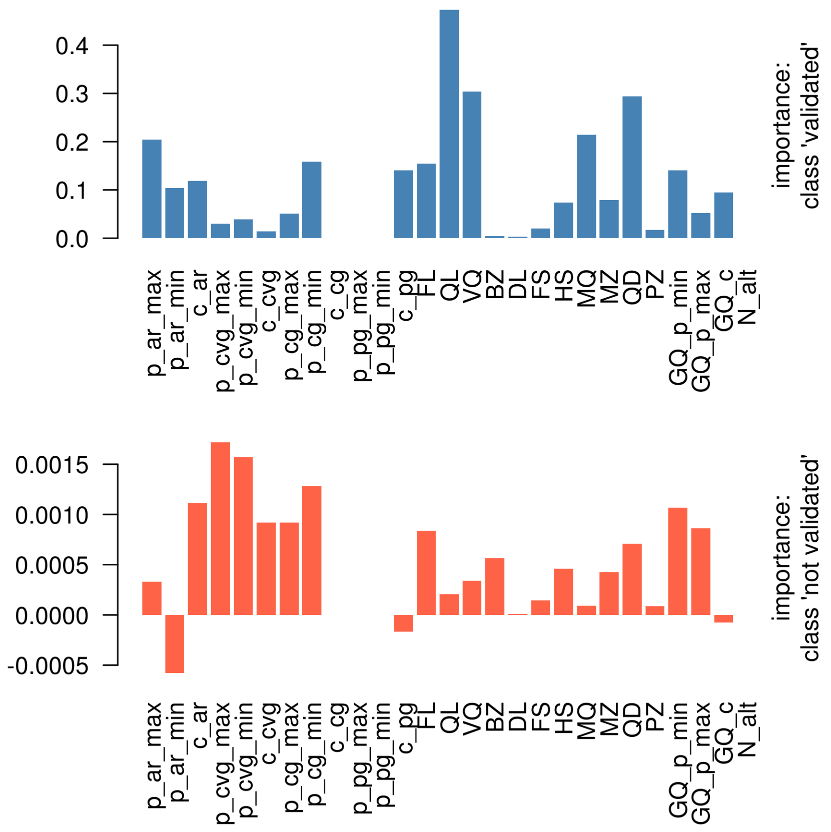

Figure 2: Feature importance for the 'validated' class (top) and the 'not validated' class (bottom).