Why ReFernment

Why ReFernment

ReFernment was created in response to the drudgery of manually annotating a plastome sequence that has levels of RNA editing. These editing sites result in a genomic sequence that contains features that look like errors (e.g., internal stop codons) so the annotations will be rejected by NCBI unless they are anotated correctly. ReFernment corrects the annotation to produce files ready for GenBank submission ReFernment takes as input a GenBank flatfile (.gb) and a gff file and generates new annotations for nonsense mutations that can reasonably be explained by RNA editing and provides conceptual translations for coding sequences with RNAediting. It should be made clear that ReFernment is not intended to predict every RNA editing site, as can other available software packages. These packages rely on RNA seq data to compare the the genomic DNA, and thus determine if a nucleotide has been edited, but these data are not always available to researchers. We made ReFernment as a simple tool to save time for those who donâ€™t have RNA seq data available to them. We hope that ReFernment also saves the time of GenBank staff, as well has helping improve the quality of annotations available in GenBank.

How it works

ReFernment operates by refining existing annotations. What this means is that ReFernment uses an annotation generated by programs such as DOGMA, CpGAVAS, Verdant or AGORA (Wyman et al. 2004; Liu et al. 2012; McKain et al. 2017; Jung et al. 2018), and adjusts these annotations to account for RNA editing. The basic operation or ReFernment is extremely simple. First, ReFernment checks the start and stop codons of each gene. In both cases ReFernment initially checks whether the codon is a valid start or stop, if the codon is not valid it checks whether an RNA editing event would result in the restoration of the codon to a valid start or stop (e.g. ACG -> AUG). If the codon is not valid, even after checking for possible RNA editing, ReFernment checks whether nearby codons (within 5 nucleotides) represent valid codons; if so, refernment changes the gene boundaries to start or stop at those valid sites. Next, ReFernment checks whether a gene has any internal stops, and if so, checks whether RNA editing would restore these nonsense translations, adjusting the translation on the GenBank flatfile to account for this. Finally, ReFernment edits the imputed GenBank flat file (appendix), adding annotations indicating the sites where RNA editing occurred with â€˜misc_featureâ€™ flags, adding necessary RNA editing flags to the relevant genes, and providing a conceptual translation for each gene.

ReFernment operates under the assumption that only U-to-C or C-to-U RNA editing is occuring in the plastome. Additionally, ReFernment assumes that all nonsense mutations are the result of RNA editing. Since most of the genes that reside within the plastome are vital to photosynthetic function, it is assumed that these genes will remain operational. There may be cases where internal stops, bad starts, or missing stops are actually the result of a real mutation, especially in parasitic lineages (citation maybe rafflasia). In most cases, however, ReFerment annotations should reflect real RNA editing sites. A major limitation of ReFerment is that the annotations it produces are only as good as the annotations it is provided. If a gene annotation is frameshifted, if there are assembly errors, or if an annotation has the incorrect start and stop sites ReFernment will not know. In other words, ReFerment is not a substitute for manually checking gene annotations, nor is ReFerment a fix for sloppy annotation. If there are more than 5 detected internal stops in a gene ReFernment will suggest to the user that they manually check that gene. There are cases where real genes see more than 5 RNA edited internal stops, but these are relatively rare, so users should use best judgement.

How to use it

ReFernment requires both a GenBank flatfile and a gff file (without sequence), to produce annotations for each genome. ReFernment takes as input the following:

- `gbFolderPath`: directory containing the gb files
- `gffFolderPath`: directory containing the gff files
- `outputFolderPath`: the directory you would like to output the newly generated GenBank flatfiles
- `genomes`: the names of the files being used

The `gbFolderPath` and `gffFolderPath` can refer to the the same directory, but unless you are *totally confortable* with ReFernment overwriting the original gb files, then `outputFolderPath` should refer to a directory seperate from the other two. When preparing to use ReFernment you should keep the names of the gb files and gff files identical, except for their extensions. For example `Hemionitis_subcordata.gb` and `Hemionitis_subcordata.gff`. This makes it easy for refernment to loop through many files all at once if you have large numbers of plastomes that need to be annotated. Along those lines, `genomes` should not contain the file extension. ReFernment will figure add the file extensions later.

Below we want ReFernment to annotate the the plastomes of Asplenium pekinense and Woodwardia unigemmata. To start off, we declare a vector named `genomes` which contains all of the plastomes which you would like to annotate.

```
genomes <- c("Asplenium_pek", "Woodwardia_uni")
```

Next, we'll declare another vector containing the path to both the input folders (`gb` and `gff`)

```
gbFolder <- "C:\\Users\\Me\\Location\\Of\\GB\\Files\\"

gffFolder <- "C:\\Users\\Me\\Location\\Of\\gff\\Files\\"

outputFolder <- "C:\\Users\\Me\\Location\\Of\\Output\\Folder\\"
```

Note that if we wanted to `gbFolder` and `gffFolder` could refer to the same directory

```
gbFolder <- "C:\\Users\\Me\\Location\\Of\\input\\Files\\"

gffFolder <- "C:\\Users\\Me\\Location\\Of\\input\\Files\\"

outputFolder <- "C:\\Users\\Me\\Location\\Of\\Output\\Folder\\"
```

Finally, we simply call the `ReFernment` function and wait for it to finish. This can take several minutes if you have a large number of plastomes.

```
ReFernment(gbFolder, gffFolder, outputFolder, genomes)
```

As ReFernment runs, it may produce the following warning message:

```
There are a high number of edited Stops ( [number] ) in [geneName] manually check to make su
```

This message is intended to warn the user of two common problems:

1. potential assembly errors that would result in an entire gene appearing to be frameshifted
2. annotation errors where the proper frame for a gene was not selected to begin with

The warning will appear if a given coding sequence has more than 5 internal stops. There are, of course, cases where this happens and it is not the result of assembly error, but these are relatively rare. Especially if there are more than 10 internal stops the user should take a close look at this coding sequecne to check for assembly error or annotation errors.

Conclusions

While other programs such as PREPACT or **ChloroSeq** are excellent at predicting RNA editing sites given the user has access to cDNA, they do not provide RNA editing for users who donâ€™t have such data. ReFernment offers a viable alternative when cDNA is not available. Furthermore, ReFernment offers easy annotation of RNA edited sites and automatic conceptual translation, easing the process of GenBank submission and saving the user valuable time.