Digger

Release 0.1

William Lees

Apr 06, 2023

GETTING STARTED

Digger is a toolkit for the automatic annotation of unrearranged V,D and J genes in B- and T- cell immunoglobulin receptor genomic loci. It can be used to annotate both entire assemblies or large fragments of a locus, or small fragments. It is designed for use with entire gene sequences, including leader and RSS (V-gene UTRs are not required, and are currently not annotated).

CHAPTER

ONE

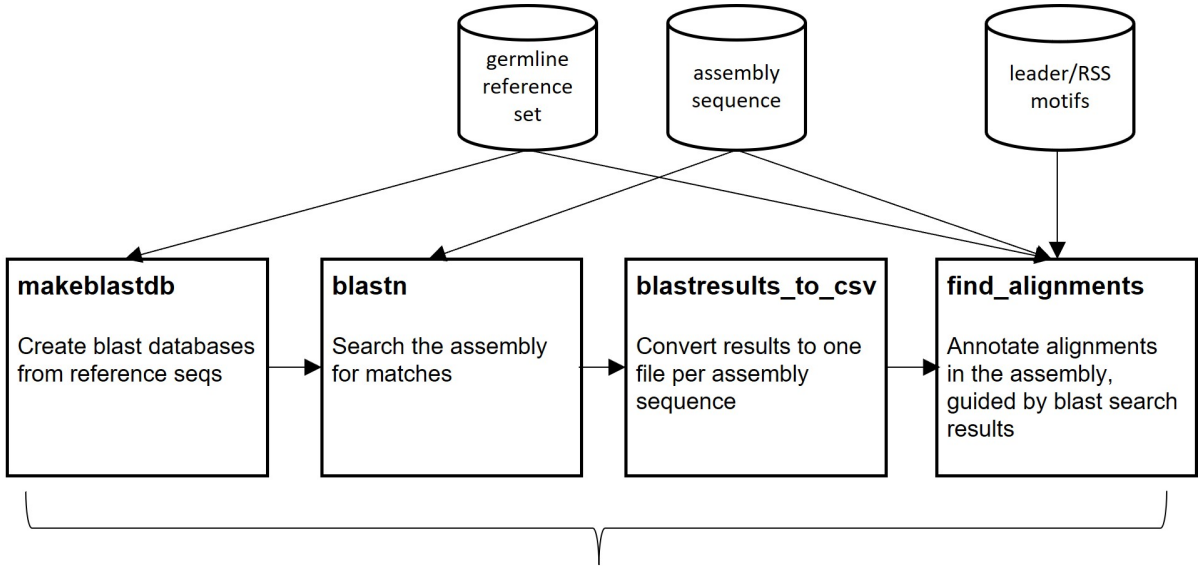
OVERVIEW

1.1 Scope and Features

Digger identifies immunoglobulin receptor germline sequences in a genomic sequence or assembly. It requires two inputs: the sequence or assembly, and a set of reference genes to use as a starting point. An existing reference set for the species under study is an ideal starting point, but if one is not available, good results can be obtained by using the reference set for another species, preferably one that is reasonably closely related. The package is intended to be straightforward to use, but some familiarity with command-line tools is expected.

Digger starts by searching the sequence for approximate matches to sequences in the reference set. Where these are found, the match is extended to the full length of the matched sequence. A window at either end is then checked for the expected flanking sequences (e.g. leader, RSS). These are identified by means of position weight matrices (PWMs). The flanking sequences have a well-established 'canonical' form that is conserved between species. Digger contains PWMs for human and rhesus macaque IG loci, and these can be used as a starting point for other species. However, as some variation is observed between species, Digger also contains tools for deriving tailored PWMs for a species of interest. These can be obtained from an existing annotation of the locus, or from an initial annotation conducted with the human or rhesus PWMs. Digger's built-in PWM's are based on analysis of selected IMGT annotations.

1.2 Pipeline



Individual steps are managed by digger for typical use cases

A brief summary of the tools contained in the package is given below. In most cases, annotation can be performed with the *digger* tool. This will call subsidiary tools as necessary. Please refer to the Examples section for more information.

Tool	Description
blastresults_to_csv	Convert the result of a BLAST search to a simple tabular format
calc_motifs	Determine PWMs from a list of gene features
compare_annotations	Compare two sets of annotations
digger	Run the annotation pipeline
find_alignments	Produce the annotation file, given tabulated BLAST results and other inputs
parse_imgt_annotations	Download and parse an annotation file from IMGT

CHAPTER

TWO

INSTALLATION

The latest stable version of Digger may be downloaded from PyPI. Digger requires Python 3.9 or above.

Development versions are available from GitHub.

Digger requires BLAST to be installed. No BLAST databases are needed: just the executable. Once BLAST has been installed, please verify by typing *blastn –help* at the command line: if everything is ok, it should provide usage instructions.

If you encounter the error `Cannot allocate memory`, this is coming from BLAST makeblastdb. Please set the environment variable `BLASTDB_LMDB_MAP_SIZE=100000000`.

The easiest way to install Digger itself is with pip:

```
> pip install receptor-digger --user
```

Digger requires the Python packages receptor-utils and biopython. These should be installed automatically by pip.

CHAPTER

THREE

RELEASE NOTES

3.1 Version 0.5: April 2023

First public version.

CHAPTER

FOUR

ANNOTATING THE HUMAN IGH LOCUS

The IGHV locus in the human reference assembly GRCh38.p12 has been annotated by IMGT. In this example, we will annotate with Digger and compare results. The comparison, and a script to reproduce it using the steps below, can be found in digger's Git repository.

4.1 Data

The locus and IMGT annotations can be downloaded and parsed with *parse_imgt_annotations*:

```
> parse_imgt_annotations \
    --save_sequence IMGT000035.fasta \
        "http://www.imgt.org/ligmdb/view?format=IMGT&id=IMGT000035" \
        IMGT000035_genes.csv \
        IGH
```

This will create two files: `IMGT000035.fasta`, containing the assembly sequence, and `IMGT000035_genes.csv`, containing the co-ordinates of the IMGT-annotated genes, and sequences of their flanking regions.

The human IGH reference set can be downloaded from IMGT with the receptor_utils command `extract_refs` (receptor_utils is installed as part of Digger's installation):

```
> extract_refs -L IGH "Homo sapiens"
```

This creates the reference sets `Homo_sapiens_IGHV_gapped.fasta`, `Homo_sapiens_IGHV.fasta`, `Homo_sapiens_IGHD.fasta`, `Homo_sapiens_IGHJ.fasta`, `Homo_sapiens_CH.fasta`

Lastly, we need to make a combined file containing all the IGH V, D and J reference genes:

```
> cat Homo_sapiens_IGHV.fasta Homo_sapiens_IGHD.fasta Homo_sapiens_IGHJ.fasta \
    > Homo_sapiens_IGHVDJ.fasta
```

4.2 Annotating the Assembly

With the data in place, we can instruct *digger* to perform the annotation:

```
> digger IMGT000035.fasta \
    -v_ref Homo_sapiens_IGHV.fasta \
        -d_ref Homo_sapiens_IGHD.fasta \
        -j_ref Homo_sapiens_IGHJ.fasta \
```

(continues on next page)

(continued from previous page)

```
        -v_ref_gapped Homo_sapiens_IGHV_gapped.fasta \
        -ref imgt,Homo_sapiens_IGHVDJ.fasta \
        -species human \
        -locus IGH \
        IMGT000035.csv
```

`-v_ref, -d_ref, -j_ref, -v_ref_gapped` provide the reference assembly sequences.

`-ref imgt,Homo_sapiens_IGHVDJ.fasta` instructs Digger to compare any sequences it identifies in the assembly with those in the combined reference set. It will create columns in the output assembly containing the nearest reference sequence found, % identity and so on. These columns will be prefixed `imgt` as specified in this argument. Multiple `-ref` arguments can be passed, allowing comparison with multiple reference sets.

`-species` tells Digger to use its internal position-weighted matrices for human loci, and `-locus` specifies the locus.

The output file will be `IMGT000035.csv`.

Digger will summarise progress as it runs. It will call the following tools:

- makeblastdb to create BLAST databases for the reference genes
- blastn to run these databases against the reference sequence
- *blastresults_to_csv* to convert the BLAST output to a simpler format
- *find_alignments* to annotate gene alignments found in the assembly

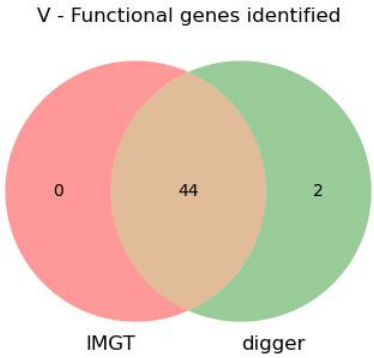
4.3 Comparing the output to IMGT's annotation

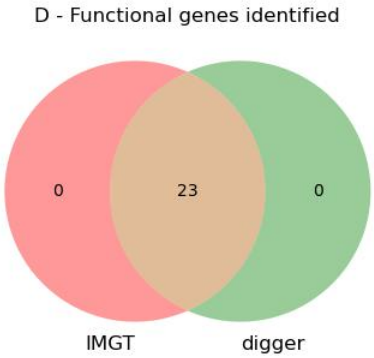
compare_annotations compares the output of a digger run with a summary annotation file. Here we use it to compare the results of digger's annotation with IMGT's:

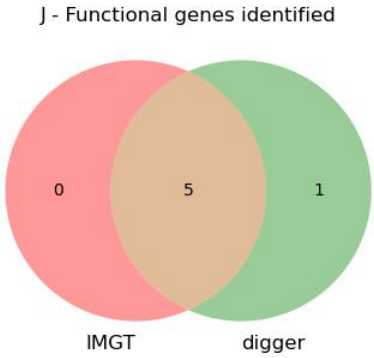
```
> compare_annotations IMGT000035.csv IMGT000035_genes.csv forward comparison_results
```

The final argument specifies that the results should be put into files named `comparison_results`. Three files are produced with differing extensions: a summary graphic (.jpg), a text file listing differences in sequences annotated as functional (.txt), and a detailed line-by-line comparison (.csv).

comparison_results.jpg summarises functional annotations found by digger and IMGT, according to which, digger annotated as functional all genes so annotated by IMGT, and annotated an additional 2 V-genes and one J-gene as functional.







comparison_results.txt lists the differences in functional analysis in detail. comparison_results_notes.txt adds some commentary: of the two additional V-genes annotated by digger as functional, one has unusual variations in the RSS, causing IMGT to annotate it as ORF. The other is annotated as ORF on the grounds that it has not been seen rearranged. The additional J-gene is currently annotated by IMGT in the assembly as ORF, although it is listed in the IMGT gene table as functional.

There is not, at present, a clear set of accepted criteria for categorisation of functionality, and minor differences of this nature are to be expected. Over the next few years, we expect to see comparisons of genomic sequencing of the loci with the expressed repertoire across multiple subjects, and this should allow a deeper understanding to develop. Overall, the comparison of digger results with the manually supervised curation at IMGT shows a good level of agreement. It is possible that results may change from those noted here, as they are based on downloaded data which may be revised over time.

4.4 References

Lefranc et al., 2015, IMGT®, the international ImMunoGeneTics information system® 25 years on. *Nucleic Acids Res.* doi: 10.1093/nar/gku1056.

Watson et al., 2013, Complete haplotype sequence of the human immunoglobulin heavy-chain variable, diversity, and joining genes and characterization of allelic and copy-number variation. *Am J Hum Genet* doi: 10.1016/j.ajhg.2013.03.004

Schneider et al., 2017, Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research* doi: 10.1101/gr.213611.116

CHAPTER

FIVE

ANNOTATING THE RHESUS MACAQUE IGH LOCUS

In this example we will see how to:

- Create position weight matrices from existing IMGT annotations
- Use the underlying commands that digger calls, and understand how they might be useful when annotating multiple contigs or scaffolds

IMGT has identified scaffolds in the 2006 rhesus macaque reference assembly, Mmul_051212, which lie within the IGH locus. Here we will bring them together in a single file and annotate them with motifs derived from the current reference assembly, rhemac10 (Mmul_10). While this example is somewhat artificial, in that the scaffolds could equally well be handled individually using the digger command, the approach is useful where the number of sequences to be processed is large. The example also serves to show how the individual commands in the package can be used. This provides some additional flexibility, for example in tuning the blast searches, and also illustrates how they work together. The comparison with IMGT's annotation of Mmul_051212, and a script to reproduce this example using the steps below, can be found in digger's Git repository.

5.1 Data

As in the previous example, the rhesus IGH germline reference set can be downloaded from IMGT with the receptor_utils command `extract_refs` (receptor_utils is installed as part of digger's installation). However, the rhesus IG gapped V-genes provided by IMGT contain additional inserted codons relative to the conventional IMGT alignment. As digger (alongside other tools) expects the conventional alignment, a further step is needed to realign the gapped sequences, using the receptor_utils tool `fix_macaque_gaps`. The following commands will prepare the reference data:

```
> extract_refs -L IGH "Macaca mulatta"
> fix_macaque_gaps Macaca_mulatta_IGHV_gapped.fasta \
    Macaca_mulatta_IGHV_gapped_fixed.fasta IGH
> cat Macaca_mulatta_IGHV.fasta Macaca_mulatta_IGHD.fasta Macaca_mulatta_IGHJ.fasta \
    > Macaca_mulatta_IGHVDJ.fasta
```

The Mmul_51212 scaffolds can be downloaded from IMGT as follows:

```
> parse_imgt_annotations --save_sequence NW_001157919.fasta \
  "https://www.imgt.org/ligmdb/view.action?format=IMGT&id=NW_001157919" \
      NW_001157919_genes.csv IGH
> parse_imgt_annotations --save_sequence NW_001122023.fasta \
  "https://www.imgt.org/ligmdb/view.action?format=IMGT&id=NW_001122023" \
  NW_001122023_genes.csv IGH
> parse_imgt_annotations --save_sequence NW_001122024.fasta \
```

(continues on next page)

(continued from previous page)

```
   "https://www.imgt.org/ligmdb/view.action?format=IMGT&id=NW_001122024" \
       NW_001122024_genes.csv IGH
> parse_imgt_annotations --save_sequence NW_001121239.fasta \
   "https://www.imgt.org/ligmdb/view.action?format=IMGT&id=NW_001121239" \
       NW_001121239_genes.csv IGH
> parse_imgt_annotations --save_sequence NW_001121240.fasta \
   "https://www.imgt.org/ligmdb/view.action?format=IMGT&id=NW_001121240" \
       NW_001121240_genes.csv IGH

>cat NW_001157919.fasta NW_001122023.fasta NW_001122024.fasta \
       NW_001121239.fasta NW_001121240.fasta > Mmul_051212.fasta
```

5.2 Preparing position-weighted matrices

Digger already has PWMs for rhesus IGH, but for the purpose of this example, we will create a set using the features listed in IMGT's annotation of the rhemac10 IGH locus, which has the IMGT accession number IMGT000064. The following commands download the annotation, determine the features, and calculate the PWMs from features of functional annotations:

```
> mkdir motifs
> cd motifs
> parse_imgt_annotations \
       "http://www.imgt.org/ligmdb/view?format=IMGT&id=IMGT000064" \
       IMGT000064_genes.csv IGH
> calc_motifs IMGT000064_genes.csv
```

`calc_motifs` will create 10 motif files in the directory.

The motifs directory may optionally contain a FASTA file `conserved_motifs.fasta` defining strongly-conserved nucleotides in the RSS and leader. Only those features with conserved residues need to be listed in the file. The names follow the filenames used for the PWMs. The following sequences were derived from Figure 3 of Ngoune et al. (2022) and will be used in this example:

```
>V-HEPTAMER
CAC---G
>V-NONAMER
-----AACC
>5'D-HEPTAMER
----GTG
>5'D-NONAMER
---T-----
>3'D-HEPTAMER
C-C---G
>3'D-NONAMER
-C----A--
>J-HEPTAMER
C--TGTG
>J-NONAMER
-GTT--TG-
```

Again, this file is provided for download at the location provided near the top of this example.

While the presence or absence of conserved residues can be a useful guide to the likely functionality of a sequence, please bear in mind that it is a guide only: exceptions can be expected, particularly where the definitions have been built on limited data.

5.3 Annotating the Assembly

The digger command is not able to handle a FASTA file containing multiple contigs, so we will call the underlying tools directly. We start by creating the blast databases and querying against the assembly, using the reference genes determined in the study:

```
> makeblastdb -in Macaca_mulatta_IGHV.fasta -dbtype nucl
> makeblastdb -in Macaca_mulatta_IGHD.fasta -dbtype nucl
> makeblastdb -in Macaca_mulatta_IGHJ.fasta -dbtype nucl

> blastn -db Macaca_mulatta_IGHV.fasta -query Mmul_051212.fasta -out mmul_IGHV.out \
   -outfmt 7 -gapopen 5 -gapextend 5 -penalty -1 -word_size 11
> blastn -db Macaca_mulatta_IGHD.fasta -query Mmul_051212.fasta -out mmul_IGHD.out \
   -outfmt 7 -gapopen 5 -gapextend 5 -penalty -1 -word_size 7 -evalue 100
> blastn -db Macaca_mulatta_IGHJ.fasta -query Mmul_051212.fasta -out mmul_IGHJ.out \
   -outfmt 7 -gapopen 5 -gapextend 5 -penalty -1 -word_size 7
```

Note that a higher evalue is used for the D genes, as they can be quite short.

Next we call `blastresults_to_csv` to convert to a more convenient format:

```
> blastresults_to_csv mmul_IGHV.out mmul_ighvdj_
> blastresults_to_csv mmul_IGHD.out mmul_ighvdj_ -a
> blastresults_to_csv mmul_IGHJ.out mmul_ighvdj_ -a
```

The commands instruct the tool to create merged files containing V,D and J hits. This is achieved by specifying the same prefix on each command (`mmul_ighvdj_`) and using the `-a` (append) option. The records created by blastn contain the name of the contig in which a hit was found. `blastresults_to_csv` will create one file per contig. The names contain the ID of the contig in `Mmul_051212.fasta`, except that they are modified where necessary to ensure file system compatibility.

We now call `find_alignments` to process the annotations:

```
> find_alignments Macaca_mulatta_IGHVDJ.fasta \
      Mmul_051212.fasta \
      "mmul_ighvdj_nw_*.csv" \
      -ref imgt,Macaca_mulatta_IGHVDJ.fasta \
      -align Macaca_mulatta_IGHV_gapped_fixed.fasta \
      -motif_dir motifs \
      Mmul_051212.csv
```

Note that the third argument, `"mmul_ighvdj_nw_*.csv"`, contains a wildcard that will match all the files produced in the previous step. It is quoted to avoid expansion by the shell. V-genes in the annotation will be annotated and gapped using the IMGT set as a template (with fixed gaps). `find_alignments` will attempt to deduce the sense in which to annotate each segment. This is helpful in this case as the contigs vary in their orientation. Note that we are specifying the location of the motifs directory created in the previous step rather than the species and locus, which would cause digger to use the built-in tables.

5.4 Comparing the output to the study's annotation

`compare_annotations` is not capable of handling the output from multiple sequences in the same file, so unfortunately we need to split the results up for the comparison:

> head -n 1 Mmul_051212.csv > mmul_header.csv

> cp mmul_header.csv NW_001157919_digger.csv > grep NW_001157919 Mmul_051212.csv >> NW_001157919_digger.csv

> cp mmul_header.csv NW_001122023_digger.csv > grep NW_001122023 Mmul_051212.csv >> NW_001122023_digger.csv

> cp mmul_header.csv NW_001122024_digger.csv > grep NW_001122024 Mmul_051212.csv >> NW_001122024_digger.csv

> cp mmul_header.csv NW_001121239_digger.csv > grep NW_001121239 Mmul_051212.csv >> NW_001121239_digger.csv

> cp mmul_header.csv NW_001121240_digger.csv > grep NW_001121240 Mmul_051212.csv >> NW_001121240_digger.csv

> compare_annotations NW_001157919_digger.csv NW_001157919_genes.csv forward NW_001157919_comp > compare_annotations NW_001122023_digger.csv NW_001122023_genes.csv forward NW_001122023_comp > compare_annotations NW_001122024_digger.csv NW_001122024_genes.csv forward NW_001122024_comp > compare_annotations NW_001121239_digger.csv NW_001121239_genes.csv forward NW_001121239_comp > compare_annotations NW_001121240_digger.csv NW_001121240_genes.csv forward NW_001121240_comp

Scaffold-by-scaffold comparisons are provided in Github. and an overall comparison is provided here. One sequence, in NW_001121240, is annotated as functional by digger but not by IMGT, who report no V-RS. Digger identifies a different start co-ordinate for the V-REGION, and finds a potentially functional RSS. Two V-sequences are identified as functional by IMGT but not by digger; one of thes has Ns in the leader, while the other lies at the extreme 5' end of the scaffold and the RSS is not fully represented: these issues caused digger not to annotate the sequences as functional.

Digger identified a total of 13 potentially functional D-genes not annotated by IMGT, across four of the five scaffolds, while IMGT annotated D-genes only in NW_001121239. The macaque IGHD genes are known to occupy a small, distinct, region towards the 3' end of the IGH locus. It would therefore be reasonable to expect them to be located in a single scaffold, and to be distinct from the V-genes. However, given the sequencing technology available for sequencing and assembly when the scaffolds were created, and bearing in mind the short length of the D-genes, it is possible that the D-locus was not correctly assembled. Another reason for suspecting this is that two of the D-sequences identified by Digger are extremely short, at 3nt and 1nt, and yet appear to be flanked by functional RSS. In contrast, in an annotation of the rhemac10 assembly, Digger identified only one D-gene not annotated by IMGT (this was also outside the D locus).

5.5 References

Ngoune et al., 2022, IMGT® Biocuration and Analysis of the Rhesus Monkey IG Loci. *Vaccines* doi: 10.3390/vaccines10030394.

Warren et al., 2020, Sequence Diversity Analyses of an Improved Rhesus Macaque Genome Enhance Its Biomedical Utility. *Science* doi: 10.1126/science.abc6617.

Gibbs et al., 2007, Evolutionary and biomedical insights from the rhesus macaque genome. *Science* doi: 10.1126/science.1139247.

CHAPTER

SIX

ADDITIONAL EXAMPLES

Additional examples, covering the IG loci of human and rhesus macaque, can be found in the tests folder of the digger Github repo. In each case, the example consists of a script file, `run_digger.bat`, which will download necessary data and conduct the analysis, and the analysis results. Despite its extension, `run_digger.bat` will run under a Linux shell (`source run_digger.bat`), or under Windows. For Windows, the Gnu core utilities, or some other implementation of simple Linux shell commands, are required.

CHAPTER

SEVEN

COMMANDLINE USAGE

7.1 blastresults_to_csv

`blastresults_to_csv` converts the output of a blast search in 'format 7' into csv format. If there were multiple query sequences, results are split into separate files. Please refer to rhesus_igh for example usage of this and the other 'individual' commands.

Convert blast file format 7 to one or more CSVs

```
usage: blastresults_to_csv [-h] [-a] infile out_prefix
```

7.1.1 Positional Arguments

infile the blast file

out_prefix prefix for csv files

7.1.2 Named Arguments

-a, --append append to existing output files

Default: False

7.2 calc_motifs

`calc_motifs` creates motif files for RSS and leader fields, based on the features produced by *parse_imgt_annotations*. Only annotations of sequences marked as 'functional' are considered. Please refer to rhesus_igh for example usage of this and the other 'individual' commands.

Given a set of gene features, create motif matrices

```
usage: calc_motifs [-h] feat_file
```

7.2.1 Positional Arguments

feat_file feature file, created, for example, by parse_imgt_annotations

7.3 compare_annotations

`compare_annotations` compares annotations produced by digger with IMGT's annotations as summarised by parse_annotations. Three files are produced: - a .jpg showing Venn diagrams of identified functional annotations - a .txt file summarising the specific functional sequences that were only identified in one annotation as opposed to both - a .csv file listing agreements and differences of all sequences annotated by either mmethod Please refer to *Annotating the human IGH locus* for example usage.

Compare digger results to an IMGT annotation

```
usage: compare_annotations [-h] [-nc] [--filter_annot FILTER_ANNOT] [--comp_name COMP_
↪NAME] digger_results annotation_file sense outfile
```

7.3.1 Positional Arguments

digger_results Digger results file

annotation_file IMGT annotation produced by parse_imgt_assembly_x.py

sense Sense of annotation compared to digger results (forward or reverse)

outfile Output file name (will create .csv, .jpg, .txt

7.3.2 Named Arguments

-nc include sequences for leader and rss

Default: False

--filter_annot filter IMGT annotations by sense (forward or reverse)

--comp_name name to use for comparison (default IMGT)

7.4 digger

`digger` annotates a single reference assembly, using BLAST to search the assembly for potential germline sequences. It requires an initial reference set for BLAST to use: this could come from a similar species, or a former annotation. Please refer to *Annotating the human IGH locus* for example usage.

Find functional and nonfunctional genes in a single assembly sequence

```
usage: digger [-h] [-species SPECIES] [-motif_dir MOTIF_DIR] [-locus LOCUS] [-v_ref V_
↪REF] [-d_ref D_REF] [-j_ref J_REF] [-v_ref_gapped V_REF_GAPPED] [-ref_comp REF_COMP] [-
↪sense SENSE] [-keepwf]
              assembly_file output_file
```

7.4.1 Positional Arguments

assembly_file	assembly sequence to search
output_file	output file (csv)

7.4.2 Named Arguments

-species	use motifs for the specified species provided with the package
-motif_dir	pathname to directory containing motif probability files
-locus	locus (default is IGH)
-v_ref	set of V reference genes to use as starting point for search
-d_ref	set of D reference genes to use as starting point for search
-j_ref	set of J reference genes to use as starting point for search
-v_ref_gapped	IMGT-gapped v-reference set used to determine alignment of novel sequences
-ref_comp	ungapped reference set(s) to compare to: name and reference file separated by comma eg mouse,mouse.fasta (may be repeated multiple times)
-sense	sense in which to read the assembly (forward or reverse) (if omitted will select automatically)
-keepwf	keep working files after processing has completed Default: False

At least one file containing reference genes must be provided. You can, for example, supply `v_ref`, `d_ref` and `j_ref`, or just `v_ref`. Digger will annotate whatever genes are discovered with the corresponding set(s). In practice, the sets do not have to be that good a match: BLAST will identify partial matches, and Digger's logic will extend the match to a full gene, including canonical RSS and leader (using the `motif` folder).

Digger requires a set of postion-weighted matrices, to identify RSS and leader. It is also possible to specify conserved locations of motifs. This *motif* data should be stored in a `motif` folder. Motifs for human and rhesus macaque IG are built in to the package, and may be used with `-species` by specifying either `human` or `rhesus_macaque`. The species is used in conjunction with `-locus` to determine the correct motifs. Alternatively, `-motif_dir` can be used to specify custom motifs created outside of the package. Please refer to *calc_motifs* and to rhesus_igh for further details on custom motifs.

`v_ref_gapped` is used to gep v-sequences correctly in order to identify conserved codons and so on. Again these sequences do not need to be that good a match in practice. The sequences **must be IMGT aligned with no extraneous codons**. Note in particular that IMGT has introduced insertions into macaque alignments in recent years. **Sets with these insertions should not be used**.

`ref_comp` allows you to specify that you would like annotated sequences to be compared with sequences in a set. You can include as many different sets as you wish. The output file will contain columns for each of these, listing the closest sequence found and the proximity (%, and number of nucleotides).

If you choose not to specify the `sense`, Digger will select the sense that elicits the highest number of hits and the highest evalue (results are shown in the output so that you can decide whether it has made the right choice, and whether you wish to annotate in both senses)

7.5 find_alignments

`find_alignments` takes the output of a blast search (as formatted by blastresults_to_csv.py), checks each identified location for the presence of a gene, and annotates if found. Please refer to rhesus_igh for example usage of this and the other 'individual' commands.

Find valid genes in a contig given blast matches

```
usage: find_alignments [-h] [-species SPECIES] [-motif_dir MOTIF_DIR] [-ref REF] [-align␣
↪ALIGN] [-locus LOCUS] [-sense SENSE] [-debug] germline_file assembly_file blast_file␣
↪output_file
```

7.5.1 Positional Arguments

germline_file	reference set used to produce the blast matches
assembly_file	assembly or contig provided to blast
blast_file	results from blast in the format provided by blastresults_to_csv (can contain wildcards if there are multiple files, will be matched by glob)
output_file	output file (csv)

7.5.2 Named Arguments

-species	use motifs for the specified species provided with the package
-motif_dir	use motif probability files present in the specified directory
-ref	ungapped reference to compare to: name and reference file separated by comma eg mouse,mouse.fasta (may be repeated multiple times)
-align	gapped reference file to use for V gene alignments (should contain V genes only), otherwise de novo alignment will be attempted
-locus	locus (default is IGH)
-sense	sense in which to read the assembly (forward or reverse) (will select automatically)
-debug	produce parsing_errors file with debug information Default: False

7.6 parse_imgt_annotations

`parse_imgt_annotations` downloads an IMGT annotation file or or uses a file already downloaded. It parses the file to provide a list of annotated features. Optionally it will also store the file downloaded, and create a FASTA file containing the annotated assembly. Please refer to *Annotating the human IGH locus* for example usage.

Given a set of IMGT annotations, build a CSV file containing gene names and co-ordinates

```
usage: parse_imgt_annotations [-h] [--save_download SAVE_DOWNLOAD] [--save_sequence SAVE_
↪SEQUENCE] [--save_imgt_annots SAVE_IMGT_ANNOTS] imgt_url outfile locus
```

7.6.1 Positional Arguments

imgt_url URL of IMGT annotation, e.g. http://www.imgt.org/ligmdb/view?format=IMGT&id=IMGT000064, or name of text file containing its contents

outfile Output file (CSV)

locus one of IGH, IGK, IGL, TRA, TRB, TRD, TRG

7.6.2 Named Arguments

--save_download Save contents of annotation to specified file

--save_sequence Save sequence to specified file

--save_imgt_annots Save IMGT annotations to specified file

CHAPTER

EIGHT

ANOTATION FORMAT

This page describes the annotation file produced by digger / find_alignments

8.1 Columns in the Annotation File

In addition to the columns in the first table, the file contains the columns in the second table, prefixed by the reference name, for each reference specified with a -ref argument.

Column Name	Meaning
contig	ID of the sequence in which the gene or pseudogene was found
start	start co-ord of the coding region
end	end co-ord of the coding region
start_rev	start co-ord in the reverse-primed sequence
end_rev	end co-ord in the reverse-primed sequence
sense	sense (relative to the input sequence)
gene_type	gene type (e.g. IGHV)
likelihood	likelihood that the RSS is that of a functional gene (compared to a random sequence)
l_part1	leader part 1 equence
l_part2	leader part 2 sequence
v_heptamer	v-heptamer sequence
v_nonamer	v-nonamer sequence
j_heptamer	j-heptamer sequence
j_nonamer	j-nonamer sequence
j_frame	coding frame of the first nucleotide of the j region (0, 1 or 2)
d_3_heptamer	3-prime d-heptamer sequence
d_3_nonamer	3-prime d-nonamer sequence
d_5_heptamer	5-prime d-heptamer sequence
d_5_nonamer	5-prime d-nonamer sequence
functional	functionality (see below)
notes	annotation notes
aa	amino acid translation of the coding region
v-gene_aligned_aa	IMGT-gapped amino acid translation of the coding sequence (for V-genes)
seq	sequence of the coding region
seq_gapped	IMGT-gapped sequence of the coding region (V-genes only)
5_rss_start	co-ordinates of the 5-prime RSS
5_rss_start_rev	
5_rss_end	
5_rss_end_rev	

continues on next page

Table 1 – continued from previous page

Column Name	Meaning
3_rss_start	co-ordinates of the 3-prime RSS
3_rss_start_rev	
3_rss_end	
3_rss_end_rev	
l_part1_start	co-ordinates of the leader part 1
l_part1_start_rev	
l_part1_end	
l_part1_end_rev	
l_part2_start	co-ordinates of the leader part 2
l_part2_start_rev	
l_part2_end	
l_part2_end_rev	
matches	number of matches to this start/end region that were produced in the BLAST analysis
blast_match	gene in the reference file with the highest match score in this start/end region
blast_score	the highest BLAST match score in this start/end region
blast_nt_diffs	the number of nucleotides differing from the most highly scoring reference sequence in this BLAST match
evalue	evalue of the most highly scoring BLAST match in this start/end region

Columns provided for each -ref:

Column Name	Meaning
_match	ID of the closest matching reference gene
_score	score of the closest match
_nt_diffs	number of nucleotides differing from the closest reference sequence

8.2 Functionality

Functionality is assigned as follows:

Functional

- RSS and leader meet or exceed position-weighted matrix threshold
- Highly-conserved residues agree with the definition for the locus, if a definition has been specified
- If a V-gene, leader starts with ATG, and spliced leader has no stop codons
- If a V-gene, coding region has no stop codons before the cysteine at IMGT position 104
- If a V-gene, conserved residues are at the expected locations
- If a J-gene, donor splice is as expected and coding region has no stop codons

ORF

- One or more of the above conditions are not met, but no stop codon has been detected
- If a V-gene, leader starts with ATG

Pseudo

- Coding region contains stop codon(s)
- Leader does not start with ATG

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search