airr-standards Documentation

Release 0.1.0

AIRR Community

Oct 10, 2019

Table of Contents

The AIRR Community is developing a set of standards for describing, reporting, storing, and sharing adaptive immune receptor repertoire (AIRR) data, such as sequences of antibodies and T cell receptors (TCRs). Some specific efforts include:

- The MiAIRR standard for describing minimal information about AIRR datasets, including sample collection and data processing information.
- Data representations (file format) specifications for storing large amounts of annotated AIRR data.
- APIs for exposing a common interface to repositories/databases containing AIRR data.
- A community standard for software tools which will allow conforming tools to gain community recognition.

CHAPTER 1

MiAIRR Standard

1.1 Introduction to MiAIRR

1.1.1 Summary

One of the primary initiatives of the Adaptive Immune Receptor Repertoire (AIRR) Community has been to develop a set of metadata standards for the submission of AIRR sequencing datasets. This work has been carried out by the AIRR Community Minimal Standards Working Group. In order to support reproducibility, standard quality control, and data deposition in a common repository, the AIRR Community has agreed to six high-level data sets that will guide the publication, curation and sharing of AIRR-Seq data and metadata: Study and subject, sample collection, sample processing and sequencing, raw sequences, processing of sequence data, and processed AIRR sequences. The detailed data elements within these sets are defined here.

1.1.2 Implementations

- NCBI-based - see this document
- AIRR Common Repositories - *in development*

1.1.3 References

1.2 MiAIRR-to-NCBI Implementation

Authors Christian E. Busse, Florian Rubelt and Syed Ahmad Chan Bukhari

1.2.1 Guide for submission of AIRR-seq data to NCBI

This site provides a detailed "how-to" guide for submission of AIRR-seq data to **NCBI repositories** (BioProject, BioSample, SRA and GenBank). For other implementations of the MiAIRR standard see here.

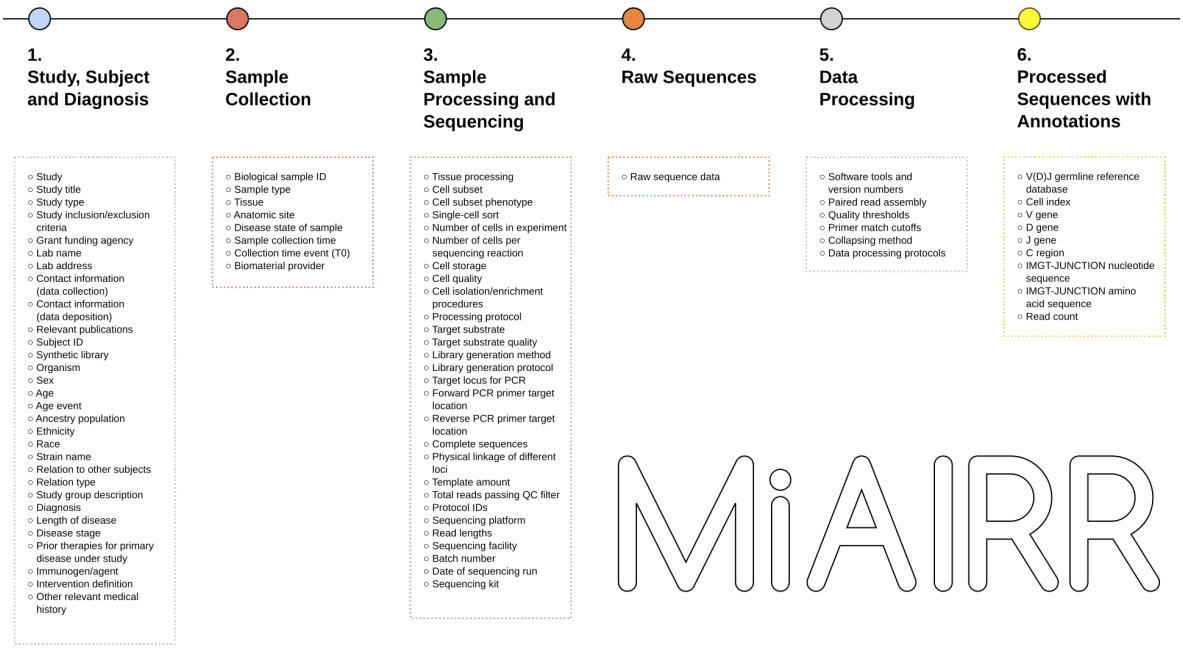


Fig. 1: Schema of MiAIRR data sets and the individual data elements of each set.

One of the primary initiatives of the AIRR (Adaptive Immune Receptor Repertoire) Community has been to develop a set of metadata standards for the submission of immune receptor repertoire sequencing datasets. This work has been carried out by the AIRR Community Standards Working Group. In order to support reproducibility, standard quality control, and data deposition in a common repository, the AIRR Community has agreed to six high-level data sets that will guide the publication, curation and sharing of AIRR-Seq data and metadata: Study and subject, sample collection, sample processing and sequencing, raw sequences, processing of sequence data, and processed AIRR sequences. The detailed data elements within these sets are defined here. The association between these AIRR sets, the associated data elements, and each of the NCBI repositories is shown below:

Submission of AIRR sequencing data and metadata to NCBI's public data repositories consists of five sequential steps:

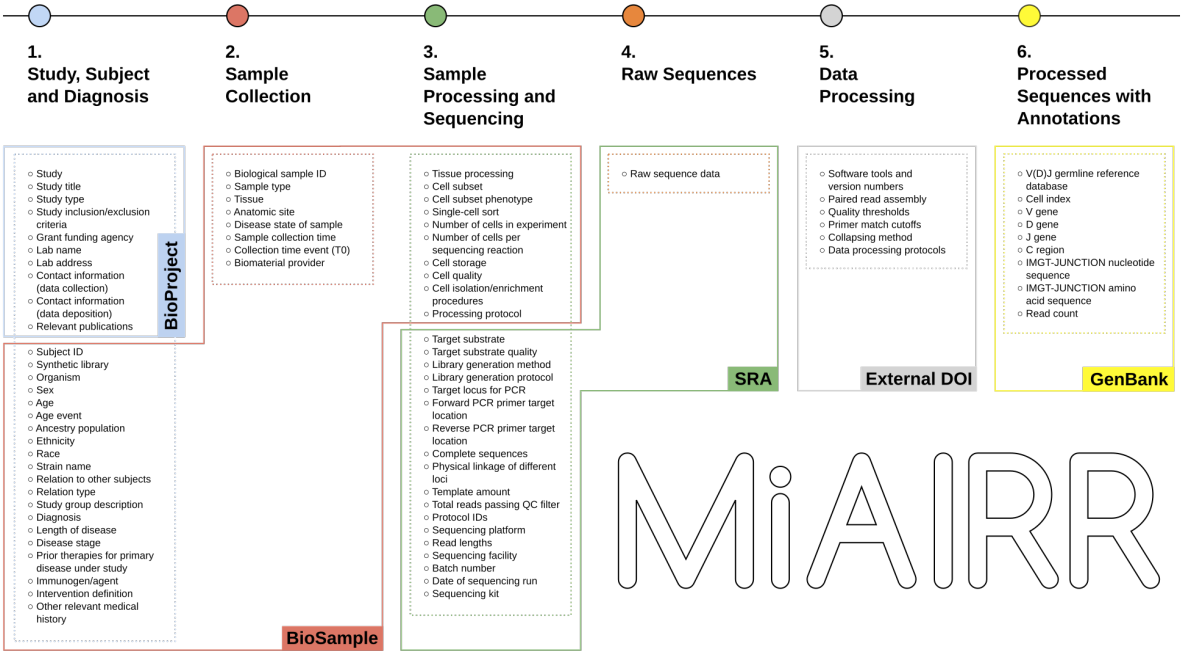
1. Submit study information to NCBI BioProject using the NCBI web interface.
2. Submit sample-level information to the NCBI BioSample repository using the AIRR-BioSample templates.
3. Submit raw sequencing data to NCBI SRA using the AIRR-SRA data templates.
4. Generate a DOI for the protocol describing how raw sequencing data were processed using Zenodo.
5. Submit processed sequencing data with sequence-level annotations to GenBank using AIRR feature tags.

For step-by-step instructions on carrying out theses steps an AIRR study submission, see here.

1.2.2 MiAIRR-to-NCBI Submission Manual

Scope of this document

Provide a user manual describing the submission of AIRR data using the NCBI reference implementation described in [Rubelt_2017]. This implementation uses NCBI's BioProject, BioSample, Sequence Read Archive (SRA) and GenBank repositories and metadata standards to report AIRR data.



Data Submission Manual

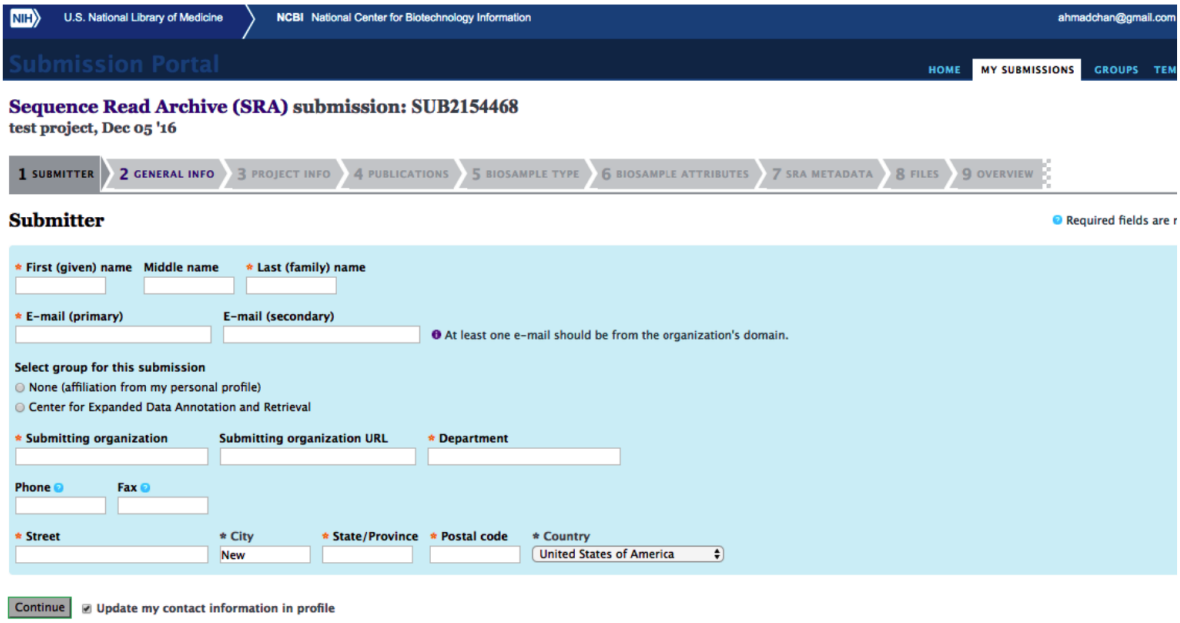
To facilitate AIRR data submissions to NCBI repositories, we have developed the NCBI-compliant metadata submission templates both for single and bulk AIRR data submissions. NCBI provides a web-based interface to create a BioProject and allows to BioSample, Sequence Read Archive (SRA) and GenBank metadata via tab-delimited files for single BioProject related data files submission. To support the bulk submission of metadata through the FTP, NCBI also has established an XML schema. This will promote the standard and provide important feedback for its iterative improvement. Since we propose to include a combination of raw and processed sequence data, the AIRR standard will sometimes need to be distributed and linked across multiple repositories (e.g., data in SRA linked to related data in GenBank). In addition, the data elements that comprise the standard will be mapped to ontologies in BioPortal through NIH CDE (Common Data Element) terms. These linkages will support more sophisticated validation and logical inference.

MiAIRR data submission to BioProject, BioSample and SRA

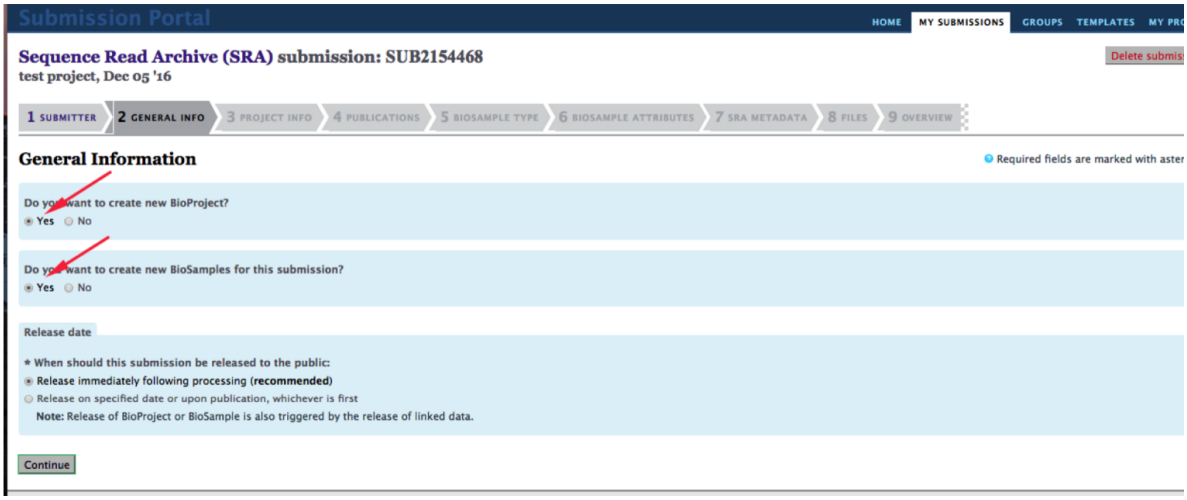
Submissions via the web interface

Submitting AIRR data and associated metadata to the Bioproject, BioSample and SRA repositories via NCBI's web interface follows in general the submission procedure described in [NCBI_NBK47528], but uses AIRR-specific template for metadata submission:

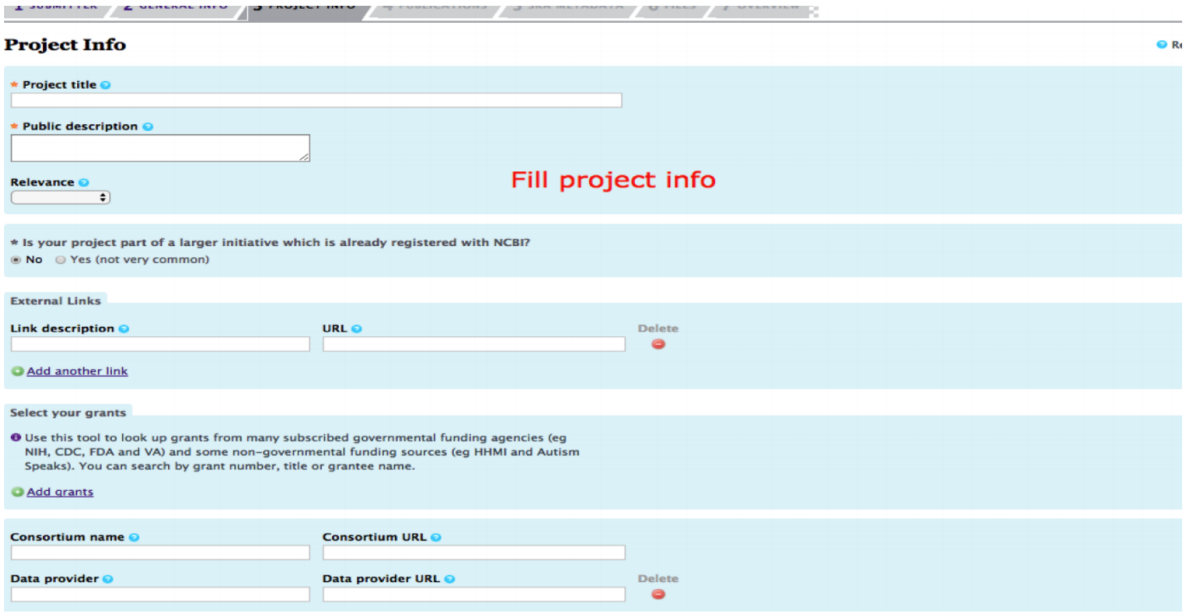
1. Go to https://submit.ncbi.nlm.nih.gov/subs/sra/ and login with your NCBI account (create an account if necessary).
2. Click on "create new submission". You will see a form as below. Fill the form with required information and click on "continue".



3. If you are submitting for the first time, check "Yes" on the "new BioProject" or "new BioSample" options to create a new project or sample, respectively.



4. Fill in the project information. Add as much relevant information you can add in description. It will help later in searching the particular submission.



5. The AIRR BioSample template is not yet listed on the NCBI website. The template sheet `AIRR_BioSample_V1.0.xls` can be downloaded from https://github.com/airr-community/airr-standards/tree/master/NCBI_implementation/templates_XLS. Fill in the required field and save the file as *tab-delimited* text file (.TSV format), then upload it.
6. To submit the SRA metadata use the `AIRR_SRA_v1.0.xls` file. Make sure that the column `sample_name` uses sample names that match the record in the BioSample template (if new BioSamples are being submitted) or a previously entered record. Also this file must be saved as *tab-delimited* text file for upload.
7. Submit the raw sequence file.
8. Complete the submission.

Submissions via an XML template

In addition to the web interface, NCBI provides an FTP-based solution to submit bulk metadata. The corresponding AIRR XML templates can be found under https://github.com/airr-community/airr-standards/tree/master/NCBI_implementation/templates_XLS. Otherwise users should refer to the current SRA file upload manual https://www.ncbi.nlm.nih.gov/sra/docs/submitfiles/. Users planning to frequently submit AIRR-seq data to SRA using scripts to generate the XML files MUST ensure that the templates are identical to the current upstream version on Github.

MiAIRR data submission to GenBank/TLS

Processed sequence data will be submitted to the "Targeted Locus Study" (TLS) section of GenBank. The details of this submission process are currently still being finalized. Basically the procedure is identical to a conventional GenBank submission with the exception of additional keywords marking it as TLS submission.

Non-productive records should be removed before the data submission or use an alternative annotation as described in the specification document.

GenBank provides multiple tools (GUI and command-line) to submit data:

- BankIt, a web-based submission tool with wizards to guide the submission process

- Sequin, NCBI's stand-alone submission tool with wizards to guide the submission process is available by FTP for use on for Windows, macOS and Unix platforms.
- Tbl2asn is the recommended tool for the bulk data submission. It is a command-line program that automates the creation of sequence records files (.sqn) for submission to GenBank, driven by multiple tabular unput data files. Documentation and download options can be found under https://www.ncbi.nlm.nih.gov/genbank/tbl2asn2/.

1.2.3 MiAIRR-to-NCBI Specification

Outline of INSDC reporting procedure

TODO: Outline the reporting procedure for data sets 1-4

In terms of standard compliance it is currently REQUIRED[1] to deposit information for MiAIRR data sets 5 and 6 in general-purpose sequence repositories for which an AIRR-accepted specification on information mapping MUST exist. However, users should note that in the future additional AIRR-sanctioned mechanisms for data deposition will become available as specified by the AIRR Common Repository Working Group. The mapping of data items in MiAIRR data sets 5 and 6 differs substantially in size and structure and therefore requires distinct reporting procedures:

- Set 5: This is free text information describing the work flow, tools and parameters of the sequence read processing. It is REQUIRED that this information is deposited as a freely available document, permanently linked via a DOI. Note that is currently neither a specific format for this document nor a recommended service provider for obtaining the DOI.
- Set 6: This is specified to contain the consensus sequence and the following information obtained from the initial analysis: V, D and J segment, C region and IMGT-JUNCTION[2] [LIGMDB_V12]. These will be deposited in a general-purpose INSDC repository, using the record structure described below.

INSDC records were originally designed to hold individual Sanger sequences. Therefore each record will contain a header with information largely identical between all records in an AIRR sequencing study. Records can be concatenated for uploading.

The INSDC feature table (FT) [INSDC_FT] is a sequence annotation standard used within the INSDC records and assigns information to specified positions on the reported sequence string. In regard to the correct location of the provided annotation, it should especially be noted that some V(D)J inference tools will return coordinates referring to the reference instead of the query sequence. As the sequence submitted in a record MUST be identical to the query sequence, the positions provided by the V(D)J inference tool MUST, if necessary, be translated back onto the query sequence. In case the start and/or end of a feature cannot be reliably determined or is not present in the reported sequence[3], open intervals CAN be used for reporting. However, open intervals MUST NOT be used to deliberately obfuscate known positions.

In addition to the required information specified in *Table_1*, users CAN use all valid FT keys/qualifiers to provide further annotation for the reported sequences. However, a record MUST still be compliant with this specification, if such OPTIONAL information would be removed, meaning that it is FORBIDDEN to move REQUIRED information into OPTIONAL keys/qualifiers. In addition, users MUST NOT use keys/qualifiers that could create ambiguity with the keys/qualifiers specified here.

[1] See the "Glossary" section on how to interpret term written in all-caps.
[2] Note that according to IMGT definition this is a superset of the CDR3.
[3] This can occur e.g. in paired-end sequencing of head-to-head concatenated transcripts, where the 5' end of the V segment is present in the amplicon, but cannot be precisely determined.

element	FT key	FT qualifier	FT value	REQUIRED (if used by original study)
V segment	`V_segment`	`/gene`	see [Feature table]	yes
D segment	`D_segment`	`/gene`	see [Feature table]	yes; if *IGH*, *TRB* or *TRD* sequence
J segment	`J_segment`	`/gene`	see [Feature table]	yes
C region	`C_region`	`/gene`	see [Feature table]	yes
JUNCTION	`CDS`	`/function`	"JUNCTION"	yes

Table 1: Summary of the mapping of mandatory AIRR MiniStd data set 6 elements to the INSDC feature table (FT). Note that the overall record will contain additional information, such as cross-references linking the deposited sequence reads and metadata.

Element mapping

The broad strategy of element mapping to the various repositories is depicted in *Table_2*.

MiAIRR data set / subset	target repository
1 / study	BioProject
1 / subject	
1 / diagnosis & treatment	
2 / sample	BioSample
3 / processing (cells)	
3 / processing (nucleic acids)	
4 / raw sequences	SRA
5 / processing (data)	user-defined DOI
6 / Processed sequences & annotations	Genbank

Table 2: Summary of the mapping of MiAIRR data sets to the various repositories

Mapping of data sets 1-4 to BioProject/BioSample/SRA

TODO: Include item-by-item mapping [NCBI_NBK47528]

Mapping of data set 5 to a user-defined repository

While several mandatory item have been defined in this data set, there is currently no mapping as the reporting procedure is implemented as a free text document. AIRR RECOMMENDS to use Zenodo for deposition of these documents, as it is hosted by CERN and supports versioned DOIs (termed "concept" DOI). Users SHOULD use the existing `AIRR` tag when submitting documents to increase the visiblity of their study.

Mapping of data set 6 to INSDC

Users should note that while the FT is standardized, the overall sequence record structure diverges between the three INSDC repositories. The following section refers to items at or above the hierarchy level of the FT using the GenBank specification [GENBANK_FF], the corresponding designations of ENA [ENA_MANUAL] are provided in parenthesis[11].

[11] Note that there is currently no submission specification for ENA. This information is provided for reference only and will be moved to a separate document in the future.

Record header

The header MUST contain all of the following elements:

- REQUIRED: header structure as specified by the respective INSDC repository [ENA_MANUAL] [GENBANK_FF] [GENBANK_SR].
- FORBIDDEN: The `DEFINITION` entry will be autopopulated by information provided in the FT part (`misc_feature`, `/note`).
- REQUIRED: identifier of the associated SRA record (MiAIRR data set 4) as `DBLINK` (ENA: `DR` line). Note that it is **not** possible to refer to individual raw reads, only the full SRA collections can be linked.
- REQUIRED: in the `KEYWORDS` field (ENA: `KW` line):
  - the term "TLS"
  - the term "Targeted Locus Study"
  - the term "AIRR"
  - the term "MiAIRR:<x>.<y>" with <x> and <y> indicating the used version and subversion of the MiAIRR standard.
- REQUIRED: DOI of the associated free-text record containing the information on data processing (MiAIRR data set 5) as `REMARK` within a `REFERENCE`[4] (ENA: `RX` line).
- OPTIONAL: The use of structured comments is currently evalutated for use in future versions of the MiAIRR standard.

Feature table

The feature table, indicated by `FEATURES` (ENA: `RX` line), MUST or SHOULD contain the following keys/qualifiers:

General sequence information

- REQUIRED: key `source` containing the following qualifiers:
  - REQUIRED: qualifier `/organism` (required by [INSDC_FT]).
  - REQUIRED: qualifier `/mol_type` (required by [INSDC_FT]).
  - REQUIRED: qualifier `/citation` pointing to the reference in the header (`REFERENCE`, ENA: `RN` line) that links to the data set 5 document.
  - REQUIRED: qualifier `/rearranged`[5].
  - REQUIRED: qualifier `/note` containing the `AIRR_READ_COUNT` keyword to indicate the read number used for the consensus. The criteria for selecting these reads and the procedure used to build the consensus SHOULD be reported as part of data set 5.
  - OPTIONAL: qualifier `/note` containing the `AIRR_INDEX_CELL` keyword for single-cell experiments. The value of the keyword SHOULD only contain alpha-numeric characters and MUST be identical for sequences derived from the same cell of origin.

[4] The current GenBank record specification does not include a separate key for DOIs.

[5] Although FT does specify a */germline* qualifier for non-rearranged sequences it has not been included in this specification as there is no obvious use case for it. In addition, non-rearranged transcripts would lack a number of other features that are assumed to be present, first of all the JUNCTION.

- RECOMMENDED: qualifiers `/assembly_gap` and `/linkage_evidence` to annotate non-overlapping paired-end sequences.
- RECOMMENDED: qualifier `/strain`, if `/organism` is "Mus musculus".

Note that additional qualifiers might be REQUIRED by GenBank to harmonize the GenBank record with the BioSample referenced by it in the header. A list of known BioSample keyword and GenBank qualifiers that MUST contain the same information can be found below. Whether (and in which direction) the existence of a keyword/qualifiers triggers a requirement in the corresponding record is currently unknown. Please report any undocumented requirements surfacing during submission to the MiAIRR team.

BioSample keyword	GenBank FT qualifier
`cell type`	`/cell_type`
`isolate`	`/isolate`
`sex`	`/sex`
`tissue`	`/tissue_type`

Segment and region annotation

The following keys MUST be used for annotation according to their FT definition, if the respective item has been reported by the original study:

- REQUIRED: key `V_region`. Note that this key MUST NOT be used to annotate V segment leader sequence[6][7].
- REQUIRED: key `misc_feature` with coordinates identical to those given in `V_region`. This key MUST contain a `/note` qualifier that contains a string as value, which describes the general type of variable region described by the record. The string MUST match the regular expression

```
/^(immunoglobulin (heavy|light)|T cell receptor (alpha|beta|gamma|delta)) chain␣
↪variable region$/
```

  This string will be used as record heading upon import into Genbank. Note that while this behavior of Genbank is undocumented, the procedure has been approved by NCBI.
- REQUIRED: key `V_segment`, both coordinates MUST be within `V_region`. Note that this key MUST NOT be used to annotate V segment leader sequence[6][7].
- REQUIRED: key `D_segment`, both coordinates MUST be within `V_region`. This key is only REQUIRED for sequences of applicable loci (*IGH*, *TRB*, *TRD*[8]).
- REQUIRED: key `J_segment`, both coordinates MUST be within `V_region`.
- REQUIRED: key `C_region`, both coordinates MUST NOT overlap with `V_region`. If the region can be unambiguously identified, the respective official gene symbol MUST be reported using the `/gene` qualifier. If only the isotype (e.g. IgG) but not the subclass (e.g. IgG1) can be identified, a truncated gene symbol (e.g. IGHG instead of IGHG1) SHOULD be reported instead[9].

Each `[VDJ]_segment` key MUST or SHOULD contain the following qualifiers:

- REQUIRED: qualifier `/gene`, containing the designation of the inferred segment, according to the database in the first `/db_xref` entry. This qualifier MUST NOT contain any allele information.

[6] The FT explicitly states that *V_segment* does **not** cover the leader sequence. The definition of *V_region* is slightly more ambiguous, however in combination with the *V_segment* definition, it becomes clear that the leader is also not considered to be a part of *V_region*. Therefore the leader sequence should be implicitly annotated as the region between the start of *CDS* and the start of *V_region*.

[7] Previously the leader was implicitly annotated as the region between *CDS* start and *V_region* start. As it was decided to drop the "global" CDS to make it easier to accommodate for INDELs, this is currently not an option anymore.

[8] For simplicity, this document only uses human gene symbols. For non-human species the specification pertains to the respective orthologs.

[9] This approach has been approved by NCBI.

- RECOMMENDED: qualifier `/allele`, containing the designation of the inferred allele, according to the database in the first `/db_xref` entry. Note that while INSDC does not specify any format for this qualifier, AIRR compliance REQUIRES that this field only contains the allele string, i.e. without the gene name or separator characters.
- REQUIRED: qualifier `/db_xref`, linking to the reference record of the inferred segment in a germline database [INSDC_XREF]. This qualifier can be present multiple times, however only the first entry is mandatory and MUST link to the database used for the segment designation given with `/gene` and (if present) `/allele`.

  Note on referencing IMGT databases: There are two IMGT database available in the controlled vocabulary [INSDC_XREF]:

  - `IMGT/GENE-DB`: This is the genome database, which requires that a reference sequence has been mapped to genomic DNA. When using this database as reference, note that you can only refer to the gene symbol **not** the allele. In the case of ambiguous allele calls (see below) this means that you MUST NOT annotate any `/allele` at all. Nevertheless, this SHOULD be the default database for applications using IMGT as reference, as the sequence for each gene/allele is unique.
  - `IMGT/LIGM`: This database collects sequences described in INSDC databases (GenBank/ENA/DDBJ). As it might contain multiple entries representing a given gene/allele, it is NOT RECOMMENDED to use it unless that inference gene/allele is only present in `IMGT/LIGM` and not in `IMGT/GENE-DB`.
- RECOMMENDED: `/inference` to indicate the tool used for segment inference. The description string SHOULD use `COORDINATES` as category and `aligment` as type [INSDC_FT].

Annotation of sequences producing multiple hits with identical scores is problematic and is ultimately at the discretion of the depositing researcher. However, the algorithms used for tie-breaking SHOULD be documented in data set 5. In addition, the following procedures MUST be followed:

- Certain gene, ambiguous allele: If multiple alleles of the same gene match to the sequence, the `/allele` qualifier MUST NOT be used. As the REQUIRED `/db_xref` qualifier will ofter refer to a specific allele, all equal hits SHOULD be annoted via this qualifier (which can be use multiple times). Also see the note on the limitations of the IMGT/GENE-DB reference database above.
- Ambiguous gene: Pick one, annotate using the qualifiers as noted for ambiguous allele.

JUNCTION annotation

INSDC does currently not define a key to annotate JUNCTION[10]. Therefore the following procedure MUST be used:

- REQUIRED: key `CDS`, indicating the positions of

  1. the first bp of the first AA of JUNCTION
  2. the last bp of the last AA of JUNCTION as determined by the utilized V(D)J inference tool.

  Open coordinates MUST be used for both coordinates to allow for automated creation of the `/translated` qualifier providing the peptide sequence. Further note that a non-productive JUNCTION can have a length not divisible by three. This key contains the following qualifiers:

  - REQUIRED: qualifier `/codon_start` with the assigned value "1".
  - REQUIRED: qualifier `/function` with the assigned value "JUNCTION".
  - REQUIRED: qualifier `/product` with an assigned value matching the regular expression

```
/^(immunoglobulin (heavy|light)|T cell receptor (alpha|beta|gamma|delta))␣
↪chain junction region$/
```

[10] NCBI confirmed that once there would be enough datasets using the *JUNCTION* tag as specified here, a motion for an INSDC-sanctioned key could be initiated.

The variable region referred to in the string MUST be the same as the one given in the `misc_feature` key.

- RECOMMENDED: qualifier `/inference`, indicating the tool used for positional inference. The description string SHOULD use `COORDINATES` as category and `protein motif` as type [INSDC_FT].
- FORBIDDEN: qualifier `/translated`, which will be automatically added by Genbank.

Note that the complete `CDS` key will be removed by Genbank if the translation contains stop codons or to many "N" (exact number unknown). As such a record will lack a central piece of REQUIRED information it is RECOMMENDED that submitters either

- remove the complete record or
- replace the `CDS` with a `misc_feature` key while at the same time removing the `/codon_start` and `/product` qualifiers

upfront, as described in the submission manual. If the submitter chooses the replacement option, it has to be ensured that the annotated coordinates are actually valid and not affect by the frame- shift.

Record body

The record body starts after `ORIGIN` (ENA: `SQ` line) and MUST contain:

- the consensus sequence

References

Footnotes

Appendix

Example record (GenBank format)

```
LOCUS       AB123456                420 bp    mRNA    linear   EST 01-JAN-2015
DEFINITION  TLS: Mus musculus immunoglobulin heavy chain variable region,
            sequence.
ACCESSION   AB123456
VERSION     AB123456.7
KEYWORDS    TLS; Targeted Locus Study; AIRR; MiAIRR:1.0.
SOURCE      Mus musculus
  ORGANISM  Mus musculus
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata;
            Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Glires;
            Rodentia; Sciurognathi; Muroidea; Muridae; Murinae; Mus.
REFERENCE   1  (bases 1 to 420)
  AUTHORS   Stibbons,P.
  TITLE     Section 5 information for experiment FOO1
  JOURNAL   published (01-JAN-2000) on Zenodo
  REMARK    DOI:10.1000/0000-12345678
REFERENCE   2  (bases 1 to 420)
  AUTHORS   Stibbons,P.
  TITLE     Direct Submission
  JOURNAL   Submitted (01-JAN-2000) Center for Transcendental Immunology,
            Unseen University, Ankh-Morpork, 12345, DISCWORLD
DBLINK      BioProject: PRJNA000001
```

(continues on next page)

(continued from previous page)

```
            BioSample: SAMN000001
            Sequence Read Archive: SRR0000001
FEATURES             Location/Qualifiers
     source          1..420
                     /organism="Mus musculus"
                     /mol_type="mRNA"
                     /strain="C57BL/6J"
                     /citation=[1]
                     /rearranged
                     /note="AIRR_READ_COUNT:123”
     V_region        1..324
     misc_feature    1..324
                     /note="immunoglobulin heavy chain variable region"
     V_segment       1..257
                     /gene="IGHV1-34"
                     /allele="01"
                     /db_xref="IMGT/LIGM:AC073565"
                     /inference="COORDINATES:alignment:IgBLAST:1.6"
     D_segment       266..272
                     /gene="IGHD2-2"
                     /allele="01"
                     /db_xref="IMGT/LIGM:AJ851868"
                     /inference="COORDINATES:alignment:IgBLAST:1.6"
     J_segment       291..324
                     /gene="IGHJ4"
                     /allele="01"
                     /db_xref="IMGT/LIGM:V00770"
                     /inference="COORDINATES:alignment:IgBLAST:1.6"
     CDS             <258..>290
                     /codon_start=1
                     /function="JUNCTION"
                     /product="immunoglobulin heavy chain junction region"
                     /inference="COORDINATES:protein motif:IgBLAST:1.6"
                     /translated="CARAGVYDGYTMDYW"
     C_region        325..420
                     /gene="Ighg2c"
ORIGIN
        1 agcctggggc ttcagtgaag atgtcctgca aggcttctgg ctacacattc actgactata
       61 acatacactg ggtgaagcag agccatggaa agagccttga gtggattgca tatattaatc
      121 ctaacaatgg tggttatggc tataacgaca agttcaggga caaggccaca ttgactgtcg
      181 acaggtcatc caacacagcc tacatggggc tccgcagcct gacctctgag gactctgcag
      241 tctattactg tgcaagagcg ggagtttacg acggatatac tatggactac tggggtcaag
      301 gaacctcagt caccgtctcc tcagccaaaa caacagcccc atcggtctat ccactggccc
      361 ctgtgtgtgg aggtacaact ggctcctcgg tgactctagg atgcctggtc aagggcaact
//
```

Glossary

- MUST / REQUIRED: Indicates that an element or action is necessary to conform to the standard.
- SHOULD / RECOMMENDED: Indicates that an element or action is considered to be best practice by AIRR, but not necessary to conform to the standard.
- CAN / OPTIONAL: Indicates that it is at the discretion of the user to use an element or perform an action.
- MUST NOT / FORBIDDEN: Indicates that an element or action will be in conflict with the standard.

Abbreviations

- AA: amino acid
- bp: base pair
- DOI: digital object identifier
- FT: INSDC Feature Table
- INSDC: International Nucleotide Sequence Database Collaboration
- SRA: sequence read archive

1.2.4 Introduction

The MiAIRR standard

The MiAIRR standard (minimal information about adaptive immune receptor repertoires) is a minimal reporting standard for experiments using sequencing-based technologies to study adaptive immune receptors (e.g. T cell receptors or immunoglobulins). It is developed and maintained by the Minimal Standards Working Group of the Adaptive Immune Receptors Repertoire (AIRR) Community [Breden_2017]. The current version (1.0) of the standard has been recently published [Rubelt_2017] and was passed by the general assembly at the annual AIRR Community meeting in December 2017. MiAIRR requires researchers to report six sets of information:

1. study, subject, diagnosis & intervention
2. sample collection
3. sample processing and sequencing
4. raw sequencing data
5. data processing
6. processed sequences with a basic analysis results

However, MiAIRR only describes the mandatory data items that have to be reported, but neither provides details how and where to deposit data nor specifies data types and formats. Therefore this document aims to provide both a submission manual for users as well as a detailed data specification for developers.

1.2.5 References

CHAPTER 2

CAIRR Pipeline

The CAIRR pipeline for submitting standards-compliant B and T cell receptor repertoire sequencing studies to the NCBI

Quick Summary

Just want to get to it? Here is a 2-minute YouTube video.

1- Go to http://cairr.miairr.org to start a metadata instance. Create an account/log in to CEDAR if you need to.

2- Fill out your metadata.

3- Return to your Workspace and select the metadata you just created.

4- To submit your metadata and associated data files, click on the Submit to Repository button in the toolbar .

5- Choose your computer files to submit, then click “SUBMIT”.

You should see the files load into CEDAR, which will immediately upload them into NCBI. (Note: CEDAR does not save your data files, only your metadata.) Error messages will be reported initially via CEDAR, and later via the email you provided.

CREATE AIRR METADATA

Clicking on the following link will open up a metadata form in CEDAR for you to enter your AIRR metadata.

http://cairr.miairr.org

For more details, read on.

Introduction

AIRR sequencing (AIRR-seq) has tremendous potential to understand the dynamics of the immune repertoire in vaccinology, infectious disease, autoimmunity, and cancer biology. The adaptation of high-throughput sequencing (HTS) for AIRR (Adaptive Immune Receptor Repertoire) studies has made possible to characterize the AIRR at unprecedented depth and the outcome of such sequencing produces big data. Effective sharing of AIRR-seq big data could

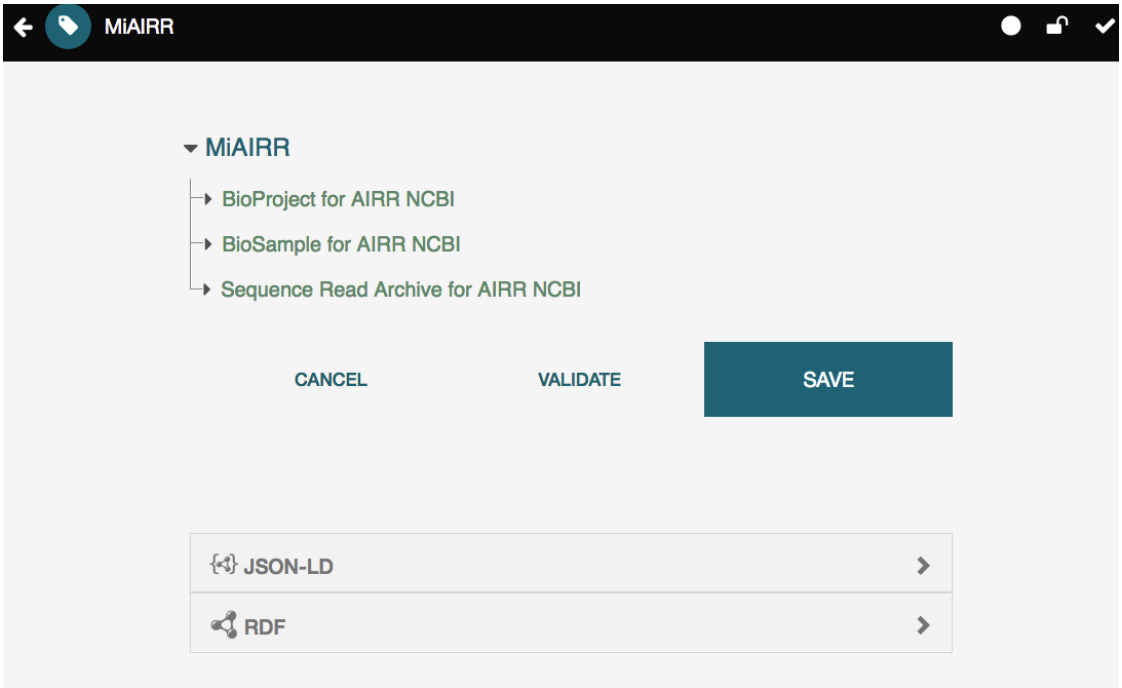
potentially reveal amazing scientific insights. The AIRR Community has proposed MiAIRR (Minimum information about an Adaptive Immune Receptor Repertoire Sequencing Experiment), a standard for reporting AIRR-seq studies. The MiAIRR standard has been implemented using the National Center for Biotechnology Information (NCBI) repositories. Submissions of AIRR-seq data to the NCBI repositories typically use a combination of web-based and flat-file templates and include only a minimal amount of terminology validation. As a result, AIRR-seq studies at the NCBI are often described using inconsistent terminologies, limiting scientists' ability to access, find, interoperate, and reuse the data sets and to understand how the experiments were performed. CEDAR (Center for Expanded Data Annotation and Retrieval) develops technologies involving the use of data standards and ontologies to improve metadata quality. In order to improve metadata quality and ease AIRR-seq study submission process, we have developed an AIRR-seq data submission pipeline named CEDAR-AIRR (CAIRR). CAIRR leverages CEDAR's technologies to: i) create web-based templates whose entries are controlled by ontology terms, ii) generate and validate metadata and iii) submit the ontology-linked metadata and sequence files (FASTQ) to the NCBI BioProject, BioSample, and Sequence Read Archive (SRA) databases. Thus, CAIRR provides a web-based metadata submission interface that supports compliance with MiAIRR standards. The interface enables ontology-based validation for several data elements, including: organism, disease, cell type and subtype, and tissue. This pipeline will facilitate the NCBI submission process and improve the metadata quality of AIRR-seq studies.

Submission Process

You will need a CEDAR system account; you can self-register at https://cedar.metadatacenter.org. You will also need the identifier of a BioProject already entered in the NCBI BioProject database. (Soon CEDAR will allow you to create a BioProject, but not quite yet!)

Submission Steps

Create your metadata. Go to http://cairr.miairr.org, and CEDAR should open in your browser. (If you are not already logged in, you may need to log in before being redirected to the metadata page.) It will look something like this.



If you do not want your metadata to be public immediately in NCBI, fill out the Submissions Release Data field at the top of the form. Then click on any of the three metadata sections to open them up.

Note that our BioProject metadata you enter can not be submitted to NCBI yet, but soon we will enable that service; meanwhile we are saving this information in CEDAR.

Click on the SAVE button often; if you navigate away from the page or close the page your unsaved changes will be

lost (after a warning). Use VALIDATE to validate your metadata via NCBI's validation service. When done, use the left arrow at the top to navigate back to your Workspace. You should see your latest saved metadata there.

Submit your metadata

From your Workspace in CEDAR, select your metadata instance. You should now be able to click on the activated Submit to Repository button. You will be prompted to specify your data files to upload. (Their names should match the names you entered in the SRA section of the form.) Finally, click on the SUBMIT button. When you complete the submission process, CEDAR will display messages indicating completion results as they are logged by NCBI. (If the upload icon is gray instead of white, you probably haven't selected an NCBI-eligible metadata form.)

Cite MiAIRR Pipeline

Bukhari, Syed Ahmad Chan, Martin J. O'Connor, Marcos Martínez-Romero, Attila L. Egyedi, Debra Debra Willrett, John Graybeal, Mark A. Musen, Florian Rubelt, Kei H. Cheung, and Steven H. Kleinstein. "The CAIRR pipeline for submitting standards-compliant B and T cell receptor repertoire sequencing studies to the NCBI." Frontiers in Immunology 9 (2018): 1877. DOI: 10.3389/fimmu.2018.01877 (now in press)

Tell Us About It

Please let us know how it went! If you are willing, we'd love to have your comments in a short survey, it should just take 5 minutes or so.

We also welcome entry of issues and requests in our github repository issues, and emails can be sent to cedar-users@lists.stanford.edu. Both of these resources are publicly visible.

CHAPTER 3

AIRR Data Representations

3.1 Field Definitions

3.1.1 Rearrangement Schema

See the *format overview* for details on how to structure this data.

Definition Clarifications

Junction versus CDR3

We work with the IMGT definitions of the junction and CDR3 regions. Specifically, the IMGT `JUNCTION` includes the conserved cysteine and tryptophan/phenylalanine residues, while `CDR3` excludes those two residues. Therefore, our `junction` and `junction_aa` fields which represent the extracted sequence include the two conserved residues, while the coordinate fields (`cdr3_start` and `cdr3_end`) exclude them.

Productive

The schema does not define a strict definition of a productive rearrangement. However, the IMGT definition is recommended:

1. Coding region has an open reading frame
2. No defect in the start codon, splicing sites or regulatory elements.
3. No internal stop codons.
4. An in-frame junction region.

Locus names

A naming convention for locus names is not strictly enforced, but the IMGT locus names are recommended. For example, in the case of human data, this would be the set: IGH, IGK, IGL, TRA, TRB, TRD, or TRG.

Gene and allele names

Gene call examples use the IMGT nomenclature, but no specific gene or allele nomenclature is mandated. Species denotations may or may not be included in the gene name, as appropriate. For example, "Homo sapiens IGHV4-59*01", "IGHV4-59*01" and "AB019438" are all valid entries for the same allele.

Alignments

There is no required alignment scheme for the nucleotide and amino acid alignment fields. These fields may, or may not, include numbering spacers (e.g., IMGT-numbering gaps), variations in case to denote mismatches, deletions, or other features appropriate to the tool that performed the alignment. The only strict requirement is that the query ("sequence") and reference ("germline") **must** be properly aligned.

Fields

Download as TSV.

Name	Type	Priority	Description
sequence_id	string	**required**	Unique query sequence identifier within the file. Most often this will b
sequence	string	**required**	The query nucleotide sequence. Usually, this is the unmodified input s
sequence_aa	string	optional	Amino acid translation of the query nucleotide sequence.
rev_comp	boolean	**required**	True if the alignment is on the opposite strand (reverse complemented)
productive	boolean	**required**	True if the V(D)J sequence is predicted to be productive.
vj_in_frame	boolean	optional	True if the V and J segment alignments are in-frame.
stop_codon	boolean	optional	True if the aligned sequence contains a stop codon.
locus	string	optional	Gene locus (chain type). For example, IGH, IGK, IGL, TRA, TRB, TF
v_call	string	**required**	V gene with allele. For example, IGHV4-59*01.
d_call	string	**required**	D gene with allele. For example, IGHD3-10*01.
j_call	string	**required**	J gene with allele. For example, IGHJ4*02.
c_call	string	optional	C region gene with allele. For example, IGHM*01.
sequence_alignment	string	**required**	Aligned portion of query sequence, including any indel corrections or
sequence_alignment_aa	string	optional	Amino acid translation of the aligned query sequence.
germline_alignment	string	**required**	Assembled, aligned, fully length inferred germline sequence spanning
germline_alignment_aa	string	optional	Amino acid translation of the assembled germline sequence.
junction	string	**required**	Junction region nucleotide sequence, where the junction is defined as t
junction_aa	string	**required**	Junction region amino acid sequence.
np1	string	optional	Nucleotide sequence of the combined N/P region between the V and D
np1_aa	string	optional	Amino acid translation of the np1 field.
np2	string	optional	Nucleotide sequence of the combined N/P region between the D and J
np2_aa	string	optional	Amino acid translation of the np2 field.
cdr1	string	optional	Nucleotide sequence of the aligned CDR1 region.
cdr1_aa	string	optional	Amino acid translation of the cdr1 field.
cdr2	string	optional	Nucleotide sequence of the aligned CDR2 region.
cdr2_aa	string	optional	Amino acid translation of the cdr2 field.
cdr3	string	optional	Nucleotide sequence of the aligned CDR3 region.
cdr3_aa	string	optional	Amino acid translation of the cdr3 field.
fwr1	string	optional	Nucleotide sequence of the aligned FWR1 region.
fwr1_aa	string	optional	Amino acid translation of the fwr1 field.
fwr2	string	optional	Nucleotide sequence of the aligned FWR2 region.
fwr2_aa	string	optional	Amino acid translation of the fwr2 field.
fwr3	string	optional	Nucleotide sequence of the aligned FWR3 region.
fwr3_aa	string	optional	Amino acid translation of the fwr3 field.
fwr4	string	optional	Nucleotide sequence of the aligned FWR4 region.
fwr4_aa	string	optional	Amino acid translation of the fwr4 field.

Name	Type	Priority	Description
v_score	number	optional	Alignment score for the V gene.
v_identity	number	optional	Fractional identity for the V gene alignment.
v_support	number	optional	V gene alignment E-value, p-value, likelihood, probability or other sim
v_cigar	string	**required**	CIGAR string for the V gene alignment.
d_score	number	optional	Alignment score for the D gene alignment.
d_identity	number	optional	Fractional identity for the D gene alignment.
d_support	number	optional	D gene alignment E-value, p-value, likelihood, probability or other sim
d_cigar	string	**required**	CIGAR string for the D gene alignment.
j_score	number	optional	Alignment score for the J gene alignment.
j_identity	number	optional	Fractional identity for the J gene alignment.
j_support	number	optional	J gene alignment E-value, p-value, likelihood, probability or other sim
j_cigar	string	**required**	CIGAR string for the J gene alignment.
c_score	number	optional	Alignment score for the C gene alignment.
c_identity	number	optional	Fractional identity for the C gene alignment.
c_support	number	optional	C gene alignment E-value, p-value, likelihood, probability or other sim
c_cigar	string	optional	CIGAR string for the C gene alignment.
v_sequence_start	integer	optional	Start position of the V segment in the query sequence (1-based closed
v_sequence_end	integer	optional	End position of the V segment in the query sequence (1-based closed i
v_germline_start	integer	optional	Alignment start position in the V gene reference sequence (1-based clo
v_germline_end	integer	optional	Alignment end position in the V gene reference sequence (1-based clos
v_alignment_start	integer	optional	Start position in the V segment in both the sequence_alignment and ge
v_alignment_end	integer	optional	End position in the V segment in both the sequence_alignment and ger
d_sequence_start	integer	optional	Start position of the D segment in the query sequence (1-based closed
d_sequence_end	integer	optional	End position of the D segment in the query sequence (1-based closed i
d_germline_start	integer	optional	Alignment start position in the D gene reference sequence (1-based clo
d_germline_end	integer	optional	Alignment end position in the D gene reference sequence (1-based clos
d_alignment_start	integer	optional	Start position of the D segment in both the sequence_alignment and ge
d_alignment_end	integer	optional	End position of the D segment in both the sequence_alignment and ger
j_sequence_start	integer	optional	Start position of the J segment in the query sequence (1-based closed i
j_sequence_end	integer	optional	End position of the J segment in the query sequence (1-based closed in
j_germline_start	integer	optional	Alignment start position in the J gene reference sequence (1-based clos
j_germline_end	integer	optional	Alignment end position in the J gene reference sequence (1-based closs
j_alignment_start	integer	optional	Start position of the J segment in both the sequence_alignment and ger
j_alignment_end	integer	optional	End position of the J segment in both the sequence_alignment and gern
cdr1_start	integer	optional	CDR1 start position in the query sequence (1-based closed interval).
cdr1_end	integer	optional	CDR1 end position in the query sequence (1-based closed interval).
cdr2_start	integer	optional	CDR2 start position in the query sequence (1-based closed interval).
cdr2_end	integer	optional	CDR2 end position in the query sequence (1-based closed interval).
cdr3_start	integer	optional	CDR3 start position in the query sequence (1-based closed interval).
cdr3_end	integer	optional	CDR3 end position in the query sequence (1-based closed interval).
fwr1_start	integer	optional	FWR1 start position in the query sequence (1-based closed interval).
fwr1_end	integer	optional	FWR1 end position in the query sequence (1-based closed interval).
fwr2_start	integer	optional	FWR2 start position in the query sequence (1-based closed interval).
fwr2_end	integer	optional	FWR2 end position in the query sequence (1-based closed interval).
fwr3_start	integer	optional	FWR3 start position in the query sequence (1-based closed interval).
fwr3_end	integer	optional	FWR3 end position in the query sequence (1-based closed interval).
fwr4_start	integer	optional	FWR3 start position in the query sequence (1-based closed interval).
fwr4_end	integer	optional	FWR4 end position in the query sequence (1-based closed interval).
v_sequence_alignment	string	optional	Aligned portion of query sequence assigned to the V segment, includir

Name	Type	Priority	Description
v_sequence_alignment_aa	string	optional	Amino acid translation of the v_sequence_alignment field.
d_sequence_alignment	string	optional	Aligned portion of query sequence assigned to the D segment, includi
d_sequence_alignment_aa	string	optional	Amino acid translation of the d_sequence_alignment field.
j_sequence_alignment	string	optional	Aligned portion of query sequence assigned to the J segment, includin
j_sequence_alignment_aa	string	optional	Amino acid translation of the j_sequence_alignment field.
c_sequence_alignment	string	optional	Aligned portion of query sequence assigned to the constant region, inc
c_sequence_alignment_aa	string	optional	Amino acid translation of the c_sequence_alignment field.
v_germline_alignment	string	optional	Aligned V gene germline sequence spanning the same region as the v_
v_germline_alignment_aa	string	optional	Amino acid translation of the v_germline_alignment field.
d_germline_alignment	string	optional	Aligned D gene germline sequence spanning the same region as the d_
d_germline_alignment_aa	string	optional	Amino acid translation of the d_germline_alignment field.
j_germline_alignment	string	optional	Aligned J gene germline sequence spanning the same region as the j_s
j_germline_alignment_aa	string	optional	Amino acid translation of the j_germline_alignment field.
c_germline_alignment	string	optional	Aligned constant region germline sequence spanning the same region a
c_germline_alignment_aa	string	optional	Amino acid translation of the c_germline_aligment field.
junction_length	integer	optional	Number of nucleotides in the junction sequence.
np1_length	integer	optional	Number of nucleotides between the V and D segments or V and J segn
np2_length	integer	optional	Number of nucleotides between the D and J segments.
n1_length	integer	optional	Number of untemplated nucleotides 5' of the D segment.
n2_length	integer	optional	Number of untemplated nucleotides 3' of the D segment.
p3v_length	integer	optional	Number of palindromic nucleotides 3' of the V segment.
p5d_length	integer	optional	Number of palindromic nucleotides 5' of the D segment.
p3d_length	integer	optional	Number of palindromic nucleotides 3' of the D segment.
p5j_length	integer	optional	Number of palindromic nucleotides 5' of the J segment.
consensus_count	integer	optional	Number of reads contributing to the (UMI) consensus for this sequenc
duplicate_count	integer	optional	Copy number or number of duplicate observations for the query seque
cell_id	string	optional	Identifier defining the cell of origin for the query sequence.
clone_id	string	optional	Clonal cluster assignment for the query sequence.
rearrangement_id	string	optional	Identifier for the Rearrangement object. May be identical to sequence_
rearrangement_set_id	string	optional	Identifier for grouping Rearrangement objects.
germline_database	string	optional	Source of germline V(D)J genes with version number or date accessed

3.1.2 Alignment Schema (Experimental)

See the *format overview* for details on how to structure this data.

Note, this schema definition is still experimental and should not be considered final.

Fields

Download as TSV.

Name	Type	Priority	Description
sequence_id	string	required	Unique query sequence identifier within the file. Most often this will be the input sequence header or a substring thereof, but may also be a custom identifier defined by the tool in cases where query sequences have been combined in some fashion prior to alignment.
segment	string	required	The segment for this alignment. One of V, D, J or C.
rev_comp	boolean	optional	Alignment result is from the reverse complement of the query sequence.
call	string	required	Gene assignment with allele.
score	number	required	Alignment score.
identity	number	optional	Alignment fractional identity.
support	number	optional	Alignment E-value, p-value, likelihood, probability or other similar measure of support for the gene assignment as defined by the alignment tool.
cigar	string	required	Alignment CIGAR string.
sequence_start	integer	optional	Start position of the segment in the query sequence (1-based closed interval).
sequence_end	integer	optional	End position of the segment in the query sequence (1-based closed interval).
germline_start	integer	optional	Alignment start position in the reference sequence (1-based closed interval).
germline_end	integer	optional	Alignment end position in the reference sequence (1-based closed interval).
rank	integer	optional	Alignment rank.
rearrangement_id	string	optional	Identifier for the Rearrangement object. May be identical to sequence_id, but will usually be a univerally unique record locator for database applications.
rearrangement_set_id	string	optional	Identifier for grouping Rearrangement objects.
germline_database	string	optional	Source of germline V(D)J genes with version number or date accessed. For example, 'IMGT/GENE-DB 3.1.18 (15 March 2018)'.

3.2 Format Specification

Data for `Rearrangement` or `Alignment` objects are stored as rows in a *tab-delimited* file and should be compatible with any TSV reader. A dataset is defined in this context as: a TSV file, a TSV with a companion YAML file containing metadata, or a directory containing multiple TSV files and YAML files.

Encoding

- The file should be encoded as ASCII or UTF-8.
- Everything is case-sensitive.

Dialect

- The record separator is a newline `\n` and the field separator is a tab `\t`.

- Fields or data should not be quoted.
- A header line with the AIRR-specified column names is always required.
- Values must not contain tab or newline characters.
- Values should avoid `@`, `#`, and quote (`"` or `'`) characters, as the result may be implementation dependent.
- Nested delimiters are not supported by the schema explicitly and should be avoided. However, if multiple values must be reported in a single column for an application specific reason, then the use of a comma as the delimiter is recommended.

File names

AIRR formatted TSV files should end with `.tsv`.

3.2.1 Structure

The data file has two sections in this order:

1. Header. A single line with column names.
2. Data values. One record per line.

A comment section preceding the header (e.g., `#` or `@` blocks) is not part of the specification, but such a section is reserved for potential inclusion in a future release. As such, a comment section should not be included in the file as it *may* be incompatible with a future specification.

Header

A single line containing the column names and specifying the field order. Any field that corresponds to one of the defined fields should use the specified field name.

Required columns

Some of the fields are defined as `required` and therefore must always be present in the header. Note, however, that all columns allow for null values. Therefore, required columns exist to define a core set of fields that are always present in the table structure, but do not mandate that a value be reported.

Custom columns

There are no restrictions on inclusion of additional custom columns in the Rearrangements file, provided such columns do not use the same name as an existing required or optional field. It is recommended that custom fields follow the same naming scheme as existing fields. Meaning, `snake_case` with narrowing scope when read from left to right. For example, `sequence_id` is the "*identifier* of the *query sequence*".

Consider submitting a pull request for a field name reservation to the airr-standards repository if the field may be broadly useful.

Ordering

There are no requirements that fields or records be sorted or ordered in any specific way. However, the field ordering provided by the schema is a recommended default, with top-to-bottom equating to left-to-right.

3.2.2 Data Values

The possible data types are `string`, `boolean`, `number` (floating point), `integer`, and `null` (empty string).

Boolean values

Boolean values must be encoded as `T` for true and `F` for false.

Null values

All fields may contain null values. This includes columns that are described as `required`. A null value should be encoded as an empty string.

Coordinate numbering

All alignment sequence coordinates use the same scheme as IMGT and INSDC (DDBJ, ENA, GenBank), with the exception that partial coordinate information should not be used in favor of simply assigning the start/end of the alignment. Meaning, coordinates should be provided as 1-based values with closed intervals, without the use of > or < annotations that denoted a partial region.

CIGAR specification

Alignments details are specified using the CIGAR format as defined in the SAM specifications, with some vocabulary restrictions on the use of clipping, skipping and padding operators. The following table defines the valid operator set.

Operator	Description
=	An identical non-gap character.
X	A differing non-gap character.
M	A positional match in the alignment. This can be either an identical (=) or differing (x) non-gap character.
D	Deletion in the query (gap in the query).
I	Insertion in the query (gap in the reference).
S	Positions that appear in the query, but not the reference. Used exclusively to denote the start position of the alignment in the query. Should precede any N operators.
N	A space in the alignment. Used exclusively to denote the start position of the alignment in the reference. Should follow any S operators.

Note, the use of either the =/`X` or `M` syntax is valid, but should be used consistently. While leading `S` and `N` operators are required, tailing `S` and `N` operators are optional.

CHAPTER 4

Software Tools Standard

4.1 AIRR Software WG - Guidance for AIRR Software Tools

Version 1.0

4.1.1 Introduction

The Adaptive Immune Receptor Repertoire (AIRR) Community will benefit greatly from cooperation among groups developing software tools and resources for AIRR research. The goal of the AIRR Software Working Group is to promote standards for AIRR software tools and resources in order to enable rigorous and reproducible immune repertoire research at the largest scale possible. As one contribution to this goal, we have established the following standards for software tools. Authors whose tools comply with this standard will, subject to ratification from the AIRR Software WG, be permitted to advertise their tools as being AIRR-compliant.

4.1.2 Requirements

Tools must:

1. Be published in source code form, and hosted on a publicly available repository with a clear versioning system.
2. Support community-curated standard file formats and strive for modularity and interoperability with other tools. In particular, tools must read and write *AIRR Data Representations* standards corresponding to their tool.
3. Include example data (in AIRR standard formats where applicable) and checks for expected output from that data, in order to provide a minimal example of functionality allowing users to check that the software is performing as described.
4. Provide information about run parameters as part of the output.
5. Provide a container build file that can be used to create an image which encapsulates the software tool, its dependencies, and required run environment. This needs to be remotely and automatically built. We currently recognize two software solutions, although we will adapt as software evolves:
    a. A Dockerfile that automatically builds a container image on Docker Hub.

b. A Singularity recipe file that automatically builds a container image on Singularity Hub.

6. Provide user support, clearly stating which level of support users can expect, and how and from whom to obtain it.

4.1.3 Recommendations

We suggest software tools be published under a license that permits free access, use, modification, and sharing, such as GPL, Apache 2.0, or MIT. However, we understand that this depends on institutional intellectual property restrictions, thus it is a recommendation rather than a requirement.

4.1.4 Explanatory Notes

Open Source Software and Versioned Repositories

Software tools in the AIRR field are evolving rapidly. In the interests of reproducibility and transparency, published work should be based on tools (and versions of tools) that can be obtained easily by other researchers in the future. To that end, AIRR compliant tools must be published in open repositories such as GitHub or Bitbucket, and we encourage publishing users to provide specifics on the version and configuration of tools that have been employed.

Community-Curated File Formats

The AIRR Data Representation Working Group has defined standards for immune receptor repertoire sequencing datasets. Software tool authors are requested to support these standards as much as possible, for applicable data sets. The currently implemented standard covers submission of reads to NCBI repositories (BioProject, BioSample, SRA and Genbank) and annotated immune receptor rearrangements. Tool authors can assist by easing/guiding the process of submission as much as possible.

Example Data and Checks

Because the installation and operation of the tools in this field may be complex, we require example data and details of expected output, so that users can confirm that their installation is functioning as expected. Furthermore, metadata (for example, germline gene libraries) and other software dependencies should be checked when the tool runs, and informative error messages issued if necessary.

Dependencies and Containers

Containers encapsulate everything needed to run a piece of software into a single convenient executable that is largely independent of the user's software environment. For the following purposes, providers of AIRR-compliant tools must provide a containerized implementation (based on a published build script as described above) as one download option that users can choose:

- Containers allow users to use and evaluate a tool easily and reproduce results, without the need to resolve dependencies or configure the environment.
- Having these containers be automatically built also provides a self-validated way to understand the fine details of installation from a known starting point.

To ensure that containers are up to date, they must be built automatically when the current release version of the tool is updated. We will use automated builds on Docker Hub and Singularity Hub for this purpose. The corresponding build files document dependencies clearly, and make it easy for the maintainer to keep the container's dependencies up to date in subsequent releases.

An example Docker container is provided on the Software WG GitHub repository. This example encapsulates IgBLAST, and implements the bioboxes command-line standard.

Support Statements

Tool authors must provide support for the tool. They must state explicitly what level of support is provided, and explain how support can be obtained. We recommend a method such as the issues tracker on Github, that publishes support requests transparently and links resolutions to specific versions or releases. Users are advised to check that the level of support and the frequency of software updates matches their expectations before committing to a tool.

Analysis Workflows

- At the moment, we do not endorse a specific workflow technology standard:
  - Technology is evolving too rapidly for us to commit to a particular workflow.
  - Typically, AIRR analysis tools have many options and modes, which would make it difficult to support a "plug and play" environment without unduly restricting functionality.
- As tools and workflows evolve, we will keep the position under review and may make stronger technology recommendations in the future.
- We strongly encourage authors of tools to provide concrete, documented, examples of workflows that employ their tools, together with sample input and output data.
- **Likewise we encourage authors of research publications to provide** documented workflows that will enable interested readers to reproduce the results.

4.1.5 Ratification

Authors may submit tools to the AIRR Software WG requesting ratification against the standard. The submitter should provide a completed copy of the *AIRR Software WG - Compliance Checklist for AIRR Software Tools* to evidence reviewable and itemised evidence of compliance with each Requirement listed above.

The Software WG will, where appropriate, issue a Certificate of Compliance, stating the version of the tool reviewed and the version of the Standard with which compliance was ratified. After receiving a Certificate, authors will be entitled to claim compliance with the Standard, and may incorporate any artwork provided by AIRR for that purpose.

The Software WG will maintain and publish a list of compliant software.

If a tool does not achieve ratification, the Software WG will provide an explanation. The Software WG encourages resubmission once issues have been resolved.

Authors must re-submit tools for ratification following major upgrades or substantial modifications. The Software WG may, at its discretion, request resubmission at any time. If a certified tool subsequently fails ratification, or is not re-submitted in response to a request from the Software WG, AIRR compliance may no longer be claimed and the associated artwork may no longer be used.

The Software WG may, at its discretion, issue a new version of this standard at any time. Tools certified against previous version(s) of the standard may continue to claim compliance with those versions and to use the associated artwork. Authors wishing to claim compliance with the new version must submit a new request for certification and may not claim compliance with the new version, or use associated artwork, until and unless certification is obtained.

4.2 AIRR Software WG - Compliance Checklist for AIRR Software Tools

Version 1.0 (when finalised)

This questionnaire should be read in conjunction with the *AIRR Software WG - Guidance for AIRR Software Tools*.

To submit your tool for ratification against the standard, please send the completed questionnaire to software@airrc.antibodysociety.org.

Please provide comments in italics in each response box where these would be helpful to facilitate understanding. We kindly ask for a brief explanatory comment if your answer to a question is *no* or *not applicable*.

Name of Tool:

Contact Name/Institution:

Contact email:

Requirement Ref.	Question	Response
1	Where is the source code published (please provide a link)?	
2	Does the tool support *AIRR Data Representations* standards? Please list any other standard data formats that are supported	yes/no
3	Does the distribution include example data? Is the example data in MiAIRR format, where applicable? Does the tool support or provide checks for expected output from example data?	yes/no yes/no/not applicable yes/no
4	Does the output of the tool include a summary of the run parameters?	yes/no
5	Is a container build file provided? Container technology used? Is the container automatically built as new versions are released?	yes/no Docker/Singularity/Other (please specify) yes/no
6	Where can users see what level of support is available? (Please provide a link)	
7	Under what software licence is the tool published? (please provide the name of the licence (e.g. GPL, MIT) or a link	

4.3 Evaluation Data Sets

The Software WG is working on the development and evaluation of simulated data sets.

Lists of published real-world datasets are maintained in the AIRR Forum Wiki.

CHAPTER 5

AIRR Python Reference Library

Installation

Install in the usual manner from PyPI:

```
> pip3 install airr --user
```

Or from the downloaded source code directory:

```
> python3 setup.py install --user
```

Reading AIRR formatted files

The `airr` package contains functions to read and write AIRR data files as either iterables or pandas data frames. The usage is straightforward, as the file format is a typical tab delimited file, but the package performs some additional validation and type conversion beyond using a standard CSV reader.

```
import airr

# Create an iteratable that returns a dictionary for each row
reader = airr.read_rearrangement('input.tsv')

# Load the entire file into a pandas data frame
df = airr.load_rearrangement('input.tsv')
```

Writing AIRR formatted files

Similar to the read operations, write functions are provided for either creating a writer class to perform row-wise output or writing the entire contents of a pandas data frame to a file. Again, usage is straightforward with the *airr* output functions simply performing some type conversion and field ordering operations.

```
import airr

# Create a writer class for iterative row output
writer = airr.create_rearrangement('output.tsv')
for row in reader:  writer.write(row)
```

(continues on next page)

(continued from previous page)

```
# Write an entire pandas data frame to a file
airr.dump_rearrangement(df, 'file.tsv')
```

5.1 API Reference

5.1.1 Inferface

airr.**read_rearrangement**(*filename*, *validate=False*, *debug=False*)
Open an iterator to read an AIRR rearrangements file

Parameters

- **file** (*str*) – path to the input file.
- **validate** (*bool*) – whether to validate data as it is read, raising a ValidationError exception in the event of an error.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns iterable reader class.

Return type *airr.io.RearrangementReader*

airr.**create_rearrangement**(*filename*, *fields=None*, *debug=False*)
Create an empty AIRR rearrangements file writer

Parameters

- **filename** (*str*) – output file path.
- **fields** (*list*) – additional non-required fields to add to the output.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns open writer class.

Return type *airr.io.RearrangementWriter*

airr.**derive_rearrangement**(*out_filename*, *in_filename*, *fields=None*, *debug=False*)
Create an empty AIRR rearrangements file with fields derived from an existing file

Parameters

- **out_filename** (*str*) – output file path.
- **in_filename** (*str*) – existing file to derive fields from.
- **fields** (*list*) – additional non-required fields to add to the output.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns open writer class.

Return type *airr.io.RearrangementWriter*

airr.**load_rearrangement**(*filename*, *validate=False*, *debug=False*)
Load the contents of an AIRR rearrangements file into a data frame

Parameters

- **filename** (*str*) – input file path.

- **validate** (*bool*) – whether to validate data as it is read, raising a ValidationError exception in the event of an error.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns Rearrangement records as rows of a data frame.

Return type pandas.DataFrame

airr.**dump_rearrangement**(*dataframe*, *filename*, *debug=False*)
Write the contents of a data frame to an AIRR rearrangements file

Parameters

- **dataframe** (*pandas.DataFrame*) – data frame of rearrangement data.
- **filename** (*str*) – output file path.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns True if the file is written without error.

Return type bool

airr.**merge_rearrangement**(*out_filename*, *in_filenames*, *drop=False*, *debug=False*)
Merge one or more AIRR rearrangements files

Parameters

- **out_filename** (*str*) – output file path.
- **in_filenames** (*list*) – list of input files to merge.
- **drop** (*bool*) – drop flag. If True then drop fields that do not exist in all input files, otherwise combine fields from all input files.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns True if files were successfully merged, otherwise False.

Return type bool

airr.**validate_rearrangement**(*filename*, *debug=False*)
Validates one or more AIRR rearrangements files

Parameters

- **filename** (*str*) – path of the file to validate.
- **debug** (*bool*) – debug flag. If True print debugging information to standard error.

Returns True if files passed validation, otherwise False.

Return type bool

5.1.2 Classes

class airr.io.**RearrangementReader**(*handle*, *base=1*, *validate=False*, *debug=False*)
Iterator for reading Rearrangement objects in TSV format

fields
field names in the input Rearrangement file.

Type list

external_fields
list of fields in the input file that are not part of the Rearrangement definition.

Type list

__init__(*handle*, *base=1*, *validate=False*, *debug=False*)
Initialization

Parameters

- **handle** (*file*) – file handle of the open Rearrangement file.
- **base** (*int*) – one of 0 or 1 specifying the coordinate schema in the input file. If 1, then the file is assumed to contain 1-based closed intervals that will be converted to python style 0-based half-open intervals for known fields. If 0, then values will be unchanged.
- **validate** (*bool*) – perform validation. If True then basic validation will be performed will reading the data. A ValidationError exception will be raised if an error is found.
- **debug** (*bool*) – debug state. If True prints debug information.

Returns reader object.

Return type *airr.io.RearrangementReader*

__iter__()
Iterator initializer

Returns airr.io.RearrangementReader

__next__()
Next method

Returns parsed Rearrangement data.

Return type dict

close()
Closes the Rearrangement file

next()
Next method

class airr.io.**RearrangementWriter**(*handle*, *fields=None*, *base=1*, *debug=False*)
Writer class for Rearrangement objects in TSV format

fields
field names in the output Rearrangement file.

Type list

external_fields
list of fields in the output file that are not part of the Rearrangement definition.

Type list

__init__(*handle*, *fields=None*, *base=1*, *debug=False*)
Initialization

Parameters

- **handle** (*file*) – file handle of the open Rearrangements file.
- **fields** (*list*) – list of non-required fields to add. May include fields undefined by the schema.
- **base** (*int*) – one of 0 or 1 specifying the coordinate schema in the output file. Data provided to the write is assumed to be in python style 0-based half-open intervals. If 1,

then data will be converted to 1-based closed intervals for known fields before writing. If 0, then values will be unchanged.

- **debug** (*bool*) – debug state. If True prints debug information.

Returns writer object.

Return type *airr.io.RearrangementWriter*

close()
Closes the Rearrangement file

write(*row*)
Write a row to the Rearrangement file

Parameters **row** (*dict*) – row to write.

class airr.schema.**Schema**(*definition*)
AIRR schema definitions

properties
field definitions.

Type collections.OrderedDict

info
schema info.

Type collections.OrderedDict

required
list of mandatory fields.

Type list

optional
list of non-required fields.

Type list

false_values
accepted string values for False.

Type list

true_values
accepted values for True.

Type list

from_bool(*value*, *validate=False*)
Converts a boolean to a string

Parameters

- **value** (*bool*) – logical value.
- **validate** (*bool*) – when True raise a ValidationError for an invalid value. Otherwise, set invalid values to None.

Returns conversion of True or False or 'T' or 'F'.

Return type str

Raises airr.ValidationError – raised if value is invalid when validate is set True.

spec (*field*)
Get the properties for a field

Parameters **name** (*str*) – field name.

Returns definition for the field.

Return type collections.OrderedDict

to_bool (*value*, *validate=False*)
Convert a string to a boolean

Parameters

- **value** (*str*) – logical value as a string.
- **validate** (*bool*) – when True raise a ValidationError for an invalid value. Otherwise, set invalid values to None.

Returns conversion of the string to True or False.

Return type bool

Raises `airr.ValidationError` – raised if value is invalid when validate is set True.

to_float (*value*, *validate=False*)
Converts a string to a float

Parameters

- **value** (*str*) – float value as a string.
- **validate** (*bool*) – when True raise a ValidationError for an invalid value. Otherwise, set invalid values to None.

Returns conversion of the string to a float.

Return type float

Raises `airr.ValidationError` – raised if value is invalid when validate is set True.

to_int (*value*, *validate=False*)
Converts a string to an integer

Parameters

- **value** (*str*) – integer value as a string.
- **validate** (*bool*) – when True raise a ValidationError for an invalid value. Otherwise, set invalid values to None.

Returns conversion of the string to an integer.

Return type int

Raises `airr.ValidationError` – raised if value is invalid when validate is set True.

type (*field*)
Get the type for a field

Parameters **name** (*str*) – field name.

Returns the type definition for the field

Return type str

validate_header (*header*)
Validate header against the schema

Parameters **header** (*list*) – list of header fields.

Returns True if a ValidationError exception is not raised.

Return type bool

Raises airr.ValidationError – raised if header fails validation.

validate_row(*row*)
Validate Rearrangements row data against schema

Parameters **row** (*dict*) – dictionary containing a single record.

Returns True if a ValidationError exception is not raised.

Return type bool

Raises airr.ValidationError – raised if row fails validation.

5.1.3 Schema

airr.schema.**RearrangementSchema Schema object for the Rearrangement definition**
AIRR schema definitions

airr.schema.**properties**
field definitions.

Type collections.OrderedDict

airr.schema.**info**
schema info.

Type collections.OrderedDict

airr.schema.**required**
list of mandatory fields.

Type list

airr.schema.**optional**
list of non-required fields.

Type list

airr.schema.**false_values**
accepted string values for False.

Type list

airr.schema.**true_values**
accepted values for True.

Type list

airr.schema.**AlignmentSchema Schema object for the Alignment definition**
AIRR schema definitions

airr.schema.**properties**
field definitions.

Type collections.OrderedDict

airr.schema.**info**
schema info.

Type collections.OrderedDict

airr.schema.**required**
list of mandatory fields.

Type list

airr.schema.**optional**
list of non-required fields.

Type list

airr.schema.**false_values**
accepted string values for False.

Type list

airr.schema.**true_values**
accepted values for True.

Type list

5.2 Commandline Tools

5.2.1 airr-tools

AIRR Community Standards utility commands.

```
usage: airr-tools [-h] [--version]  ...
```

-h, --help
show this help message and exit

--version
show program's version number and exit

airr-tools merge

Merge AIRR rearrangement files.

```
usage: airr-tools merge [--version] [-h] -o OUT_FILE [--drop] -a AIRR_FILES
                        [AIRR_FILES ...]
```

--version
show program's version number and exit

-h, --help
show this help message and exit

-o <out_file>
Output file name.

--drop
If specified, drop fields that do not exist in all input files. Otherwise, include all columns in all files and fill missing data with empty strings.

-a <airr_files>
A list of AIRR rearrangement files.

airr-tools validate

Validate AIRR rearrangement files.

```
usage: airr-tools validate [--version] [-h] -a AIRR_FILES [AIRR_FILES ...]
```

--version
show program's version number and exit

-h, --help
show this help message and exit

-a <airr_files>
A list of AIRR rearrangement files.

5.3 Release Notes

5.3.1 Version 1.2.1: October 5, 2018

- Fixed a bug in the python reference library causing start coordinate values to be empty in some cases when writing data.

5.3.2 Version 1.2.0: August 17, 2018

- Updated schema set to v1.2.
- Several improvements to the `validate_rearrangement` function.
- Changed behavior of all *airr.interface* functions to accept a file path (string) to a single Rearrangement TSV, instead of requiring a file handle as input.
- Added `base` argument to `RearrangementReader` and `RearrangementWriter` to support optional conversion of 1-based closed intervals in the TSV to python-style 0-based half-open intervals. Defaults to conversion.
- Added the custom exception `ValidationError` for handling validation checks.
- Added the `validate` argument to `RearrangementReader` which will raise a `ValidationError` exception when reading files with missing required fields or invalid values for known field types.
- Added `validate` argument to all type conversion methods in `Schema`, which will now raise a `ValidationError` exception for value that cannot be converted when set to `True`. When set `False` (default), the previous behavior of assigning `None` as the converted value is retained.
- Added `validate_header` and `validate_row` methods to `Schema` and removed validations methods from `RearrangementReader`.
- Removed automatic closure of file handle upon reaching the iterator end in `RearrangementReader`.

5.3.3 Version 1.1.0: May 1, 2018

Initial release.

CHAPTER 6

AIRR R Reference Library

An R library providing AIRR schema definitions and read, write, and validation functions for AIRR standard formatted data files.

Download & Installation

To install the latest release from CRAN:

```
install.packages("airr")
```

To build from the source code, first install the build dependencies:

```
install.packages(c("devtools", "roxygen2"))
```

To install the latest development code via devtools:

```
library(devtools)
install_github("airr-community/airr-standards/lang/R@master")
```

Note, using `install_github` will not build the documentation. To generate the documentation, clone the repository and build as normal. Then run the following R commands from the package root `lang/R`:

```
library(devtools)
install_deps(dependencies=T)
document()
install()
test()
```

6.1 About

6.1.1 AIRR Data Representation Reference Library

`airr` is an R package for working with data formatted according to the AIRR Data Representation schemas. It includes the full set of schema definitions along with simple functions for read, write and validation.

6.1.2 Dependencies

Imports: methods, readr, stats, stringi, yaml
Suggests: knitr, rmarkdown, testthat

6.1.3 Authors

Jason Vander Heiden (aut, cre)
Susanna Marquez (aut)
AIRR Community (cph)

6.2 Usage Vignette

6.2.1 Introduction

Since the use of High-throughput sequencing (HTS) was first introduced to analyze immunoglobulin (B-cell receptor, antibody) and T-cell receptor repertoires (Freeman et al, 2009; Robins et al, 2009; Weinstein et al, 2009), the increasing number of studies making use of this technique has produced enormous amounts of data and there exists a pressing need to develop and adopt common standards, protocols, and policies for generating and sharing data sets. The Adaptive Immune Receptor Repertoire (AIRR) Community formed in 2015 to address this challenge (Breden et al, 2017) and has stablished the set of minimal metadata elements (MiAIRR) required for describing published AIRR datasets (Rubelt et al, 2017) as well as file formats to represent this data in a machine-readable form. The `airr` R package provide read, write and validation of data following the AIRR Data Representation schemas. This vignette provides a set of simple use examples.

AIRR Data Representation Standards

The AIRR Community's recommendations for a minimal set of metadata that should be used to describe an AIRR-seq data set when published or deposited in a AIRR-compliant public repository are described in Rubelt et al, 2017. The primary aim of this effort is to make published AIRR datasets FAIR (findable, accessible, interoperable, reusable); with sufficient detail such that a person skilled in the art of AIRR sequencing and data analysis will be able to reproduce the experiment and data analyses that were performed.

Following this principles, V(D)J reference alignment annotations are saved in standard tab-delimited files (TSV) with associated metadata provided in accompanying YAML formatted files. The column names and field names in these files have been defined by the AIRR Data Representation Working Group using a controlled vocabulary of standardized terms and types to refer to each piece of information.

6.2.2 Reading AIRR formatted files

The `airr` package contains the function `read_rearrangement` to read and validate files containing AIRR Rearrangement records, where a Rearrangement record describes the collection of optimal annotations on a single sequence that has undergone V(D)J reference alignment. The usage is straightforward, as the file format is a typical tabulated file. The argument that needs attention is `base`, with possible values `"0"` and `"1"`. `base` denotes the starting index for positional fields in the input file. Positional fields are those that contain alignment coordinates and names ending in "_start" and "_end". If the input file is using 1-based closed intervals (R style), as defined by the standard, then positional fields will not be modified under the default setting of `base="1"`. If the input file is using 0-based coordinates with half-open intervals (python style), then positional fields may be converted to 1-based closed intervals using the argument `base="0"`.

```
library(airr)

example_data <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")
basename(example_data)
```

```
## [1] "rearrangement-example.tsv.gz"
```

```
airr_rearrangement <- read_rearrangement(example_data)
class(airr_rearrangement)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
head(airr_rearrangement)
```

```
## # A tibble: 6 x 33
##   sequence_id sequence rev_comp productive vj_in_frame stop_codon v_call
##   <chr>       <chr>    <lgl>    <lgl>      <lgl>       <lgl>      <chr>
## 1 SRR765688.... NNNNNNN... FALSE    TRUE       TRUE        FALSE      IGHV2...
## 2 SRR765688.... NNNNNNN... FALSE    TRUE       TRUE        FALSE      IGHV5...
## 3 SRR765688.... NNNNNNN... FALSE    TRUE       TRUE        FALSE      IGHV7...
## 4 SRR765688.... NNNNNNN... FALSE    TRUE       TRUE        FALSE      IGHV7...
## 5 SRR765688.... NNNNNNN... FALSE    TRUE       TRUE        FALSE      IGHV7...
## 6 SRR765688.... NNNNNNN... FALSE    FALSE      TRUE        TRUE       IGHV2...
## # ... with 26 more variables: d_call <chr>, j_call <chr>, c_call <chr>,
## #   sequence_alignment <chr>, germline_alignment <chr>, junction <chr>,
## #   junction_aa <chr>, v_cigar <chr>, d_cigar <chr>, j_cigar <chr>,
## #   v_sequence_start <int>, v_sequence_end <int>, v_germline_start <int>,
## #   v_germline_end <int>, d_sequence_start <int>, d_sequence_end <int>,
## #   d_germline_start <int>, d_germline_end <int>, j_sequence_start <int>,
## #   j_sequence_end <int>, j_germline_start <int>, j_germline_end <int>,
## #   junction_length <int>, np1_length <int>, np2_length <int>,
## #   duplicate_count <int>
```

6.2.3 Writing AIRR formatted files

The `airr` package contains the function `write_rearrangement` to write Rearrangement records to the AIRR TSV format.

```
out_file <- file.path(tempdir(), "airr_out.tsv")
write_rearrangement(airr_rearrangement, out_file)
```

6.2.4 References

1. Breden, F., E. T. Luning Prak, B. Peters, F. Rubelt, C. A. Schramm, C. E. Busse, J. A. Vander Heiden, et al. 2017. Reproducibility and Reuse of Adaptive Immune Receptor Repertoire Data. *Front Immunol* 8: 1418.
2. Freeman, J. D., R. L. Warren, J. R. Webb, B. H. Nelson, and R. A. Holt. 2009. Profiling the T-cell receptor beta-chain repertoire by massively parallel sequencing. *Genome Res* 19 (10): 1817-24.
3. Robins, H. S., P. V. Campregher, S. K. Srivastava, A. Wacher, C. J. Turtle, O. Kahsai, S. R. Riddell, E. H. Warren, and C. S. Carlson. 2009. Comprehensive assessment of T-cell receptor beta-chain diversity in alphabeta T cells. *Blood* 114 (19): 4099-4107.

4. Rubelt, F., C. E. Busse, S. A. C. Bukhari, J. P. Burckert, E. Mariotti-Ferrandiz, L. G. Cowell, C. T. Watson, et al. 2017. Adaptive Immune Receptor Repertoire Community recommendations for sharing immune-repertoire sequencing data. *Nat Immunol* 18 (12): 1274-8.

5. Weinstein, J. A., N. Jiang, R. A. White, D. S. Fisher, and S. R. Quake. 2009. High-throughput sequencing of the zebrafish antibody repertoire. *Science* 324 (5928): 807-10.

6.3 Reference Topics

6.3.1 read_airr

Read an AIRR TSV

Description

`read_airr` reads a TSV containing AIRR records.

Usage

```
read_airr(file, base = c("1", "0"), schema = RearrangementSchema, ...)
```

```
read_rearrangement(file, base = c("1", "0"), ...)
```

```
read_alignment(file, base = c("1", "0"), ...)
```

Arguments

file input file path.

base starting index for positional fields in the input file. If `"1"`, then these fields will not be modified. If `"0"`, then fields ending in `"_start"` and `"_end"` are 0-based half-open intervals (python style) in the input file and will be converted to 1-based closed-intervals (R style).

schema `Schema` object defining the output format.

... additional arguments to pass to read_delim.

Value

A data.frame of the TSV file with appropriate type and position conversion for fields defined in the specification.

Details

`read_rearrangement` reads an AIRR TSV containing Rearrangement data.

`read_alignment` reads an AIRR TSV containing Alignment data.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)
```

See also

See Schema for the AIRR schema object definition. See write_airr for writing AIRR data.

6.3.2 write_airr

Write an AIRR TSV

Description

`write_airr` writes a TSV containing AIRR formatted records.

Usage

```
write_airr(data, file, base = c("1", "0"),
schema = RearrangementSchema, ...)
```

```
write_rearrangement(data, file, base = c("1", "0"), ...)
```

```
write_alignment(data, file, base = c("1", "0"), ...)
```

Arguments

data data.frame of Rearrangement data.

file output file name.

base starting index for positional fields in the output file. Fields in the input `data` are assumed to be 1-based closed-intervals (R style). If `"1"`, then these fields will not be modified. If `"0"`, then fields ending in `_start` and `_end` will be converted to 0-based half-open intervals (python style) in the output file.

schema `Schema` object defining the output format.

... additional arguments to pass to write_delim.

Details

`write_rearrangement` writes a data.frame containing AIRR Rearrangement data to TSV.

`write_alignment` writes a data.frame containing AIRR Alignment data to TSV.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)

# Write a Rearrangement data file
outfile <- file.path(tempdir(), "output.tsv")
write_rearrangement(df, outfile)
```

See also

See Schema for the AIRR schema object definition. See read_airr for reading to AIRR files.

6.3.3 validate_airr

Validate AIRR data

Description

`validate_airr` validates compliance of the contents of a data.frame to the AIRR data standards.

Usage

```
validate_airr(data, schema = RearrangementSchema)
```

Arguments

data data.frame to validate.

schema `Schema` object defining the data standard.

Value

Returns `TRUE` if the input `data` is compliant and `FALSE` if not.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)

# Validate a data.frame against the Rearrangement schema
validate_airr(df, schema=RearrangementSchema)
```

```
[1] TRUE
```

6.3.4 load_schema

Load a schema definition

Description

`load_schema` loads an AIRR object definition from the internal definition set.

Usage

```
load_schema(definition)
```

Arguments

definition name of the schema definition.

Value

A Schema object for the `definition`.

Details

Valid definitions include:

- `"Rearrangement"`
- `"Alignment"`
- `"Study"`
- `"Subject"`
- `"Diagnosis"`
- `"Sample"`
- `"CellProcessing"`
- `"NucleicAcidProcessing"`
- `"RawSequenceData"`
- `"SoftwareProcessing"`

Examples

```
# Load the Rearrangement definition
schema <- load_schema("Rearrangement")

# Load the Alignment definition
schema <- load_schema("Alignment")
```

See also

See Schema for the return object.

6.3.5 Schema-class

S4 class defining an AIRR standard schema

Description

`Schema` defines a common data structure for AIRR Data Representation standards.

Usage

```
"names"(x)
```

```
"["(x, i)
```

```
"$"(x, name)
```

```
AlignmentSchema
```

```
RearrangementSchema
```

Arguments

x `Schema` object.

i field name.

name field name.

Format

A `Schema` object.

Details

The following predefined Schema objects are defined:

`AlignmentSchema`: AIRR Alignment `Schema`.

`RearrangementSchema`: AIRR Rearrangement `Schema`.

Slots

required `character` vector of required fields.

optional `character` vector of non-required fields.

properties `list` of field definitions.

See also

See load_schema for loading a `Schema` from the definition set. See read_airr, write_airr and validate_airr schema operators.

6.3.6 ExampleData

Example AIRR data

Description

Example data files compliant with the the AIRR Data Representation standards.

Format

`extdata/rearrangement-example.tsv.gz`: Rearrangement TSV file.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)
```

6.4 Release Notes

6.4.1 Version 1.2.0: August 17, 2018

- Updated schema set to v1.2.
- Changed defaults to `base="1"` for read and write functions.
- Updated example TSV file with coordinate changes, addition of `germline_alignment` data and simplification of `sequence_id` values.

6.4.2 Version 1.1.0: May 1, 2018

Initial release.

CHAPTER 7

Applications Supporting AIRR Standards

7.1 Rearrangement Schema

The following list of software tools and databases support the TSV format of v1.2 of the *AIRR Rearrangement schema*.

Software	Version	Support
AIRR Python Library	1.2	Input, output and validation
AIRR R Library	1.2	Input, output and validation
IgBLAST	1.10	Output
IGoR	TBD	Input and output
Immcantation:Change-O	0.4.2	Input, output and conversion
ImmuneDB	0.24.0	Output
iReceptor	2.0	Input, output and conversion
MiXCR	2.2.1	Output
OLGA	TBD	Input and output
Partis	TBD	Output
SONAR	3.0	Output
TRIgS	2	Input
VDJServer	1.2.0	Input and output
Vidjil-algo	2018.10	Output
Vidjil Web Platform	TBD	Input and conversion

CHAPTER 8

Examples & Workflows

Example workflows, tutorials and use cases for AIRR Standards.

8.1 AIRR Rearrangement TSV Interoperability Example

The example that follows illustrates the interoperability provided by the AIRR Rearrangement schema. The code provided demonstrates how to take AIRR formatted data output by IgBLAST and combine it with data processed by IMGT/HighV-QUEST that has converted to the AIRR format by Change-O. Then, the merged output of these two distinct tools is used to (a) create MiAIRR compliant GenBank/TLS submission files, and (b) perform a simple V gene usage analysis task.

8.1.1 Data

We've hosted a small set of example data from BioProject PRJNA338795 (Vander Heiden et al, 2017. J Immunol.) containing both input and output of the example. It may be downloaded from:

Example Data

8.1.2 Walkthrough

Environment setup

We'll use the Immcantation docker image for this example, which comes loaded with all the tools used in the steps that follow:

```
# Download the image
docker pull kleinstein/immcantation:devel

# Invoke a shell session inside the Immcantation docker image
```

(continues on next page)

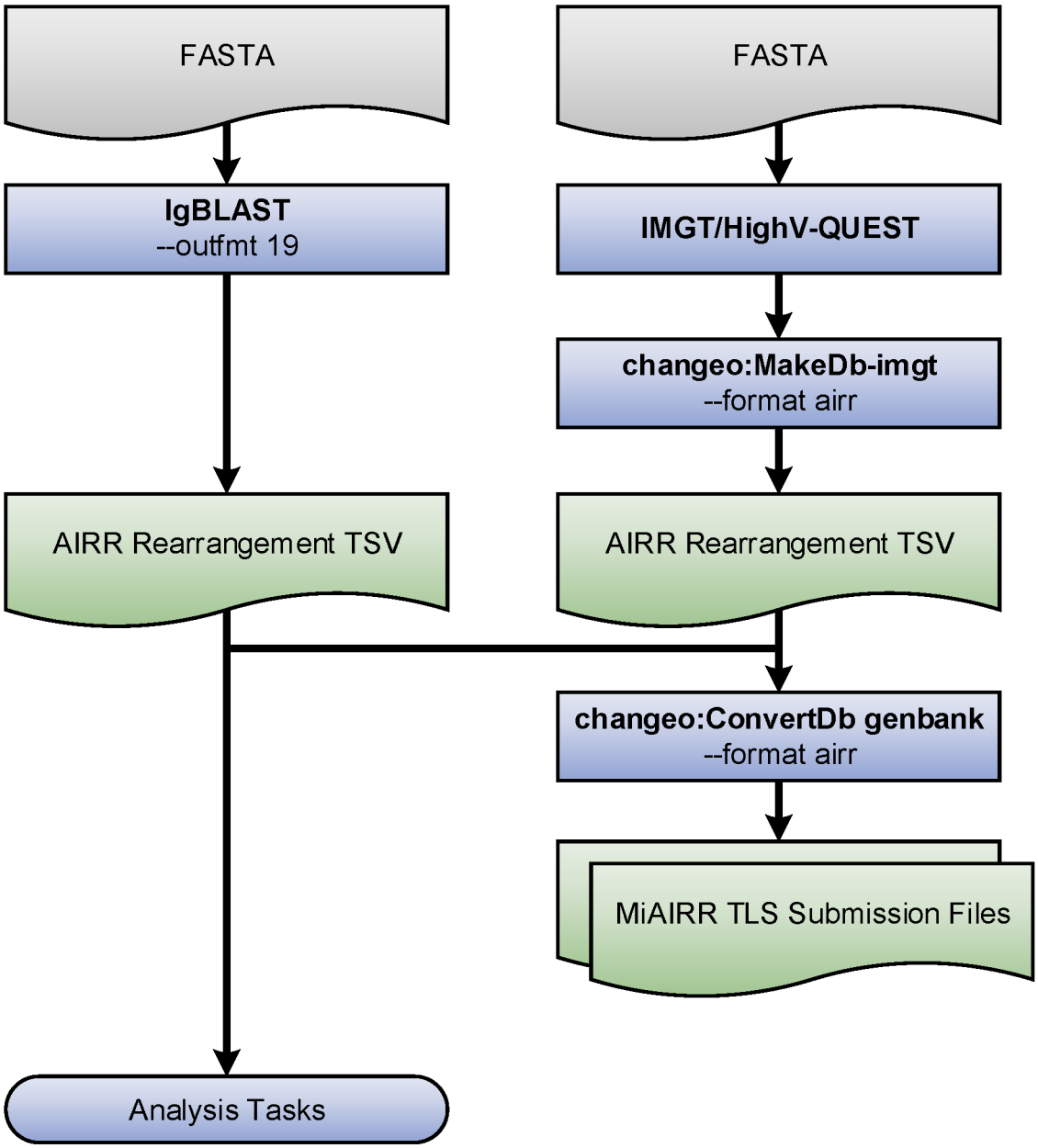


Fig. 1: **Flowchart of the example steps.**

(continued from previous page)

```
# Map example data (~/data) to the container's /data directory
$> docker run -it -v ~/data:/data:z kleinstein/immcantation:devel bash
```

Generate AIRR formatted TSV files

TSV files compliant with the AIRR Rearrangement schema may be output directly from IgBLAST v1.9+ or generated from IMGT/HighV-QUEST output (or IgBLAST <=1.8 ouput) using the MakeDb parser provided by Change-O:

```
# Generate TSV directly with IgBLAST
$> cd /data
$> export IGDATA=/usr/local/share/igblast
$> igblastn -query HD13M.fasta -out HD13M_fmt19.tsv -outfmt 19 \
      -germline_db_V $IGDATA/database/imgt_human_ig_v \
      -germline_db_D $IGDATA/database/imgt_human_ig_d \
      -germline_db_J $IGDATA/database/imgt_human_ig_j \
      -auxiliary_data $IGDATA/optional_file/human_gl.aux \
      -ig_seqtype Ig -organism human \
      -domain_system imgt

# Generate TSV from IMGT/HighV-QUEST results using changeo:MakeDb
$> MakeDb.py imgt -i HD13N_imgt.txz -s HD13N.fasta  \
      --scores --partial --format airr
```

Generate GenBank/TLS submission files

AIRR TSV files can be input directly in Change-O's ConvertDb-genbank tool to generate MiAIRR compliant files for submission to GenBank/TLS:

```
# Generate ASN files from IgBLAST output
$> ConvertDb.py genbank -d HD13M_fmt7_db-pass.tsv --format airr \
      --inf IgBLAST:1.7.0 --organism "Homo sapiens" \
      --tissue "Peripheral blood" --cell "naive B cell" \
      --id --asn -sbt HD13M.sbt

# Generate ASN files from IMGT/HighV-QUEST output
$> ConvertDb.py genbank -d HD13N_imgt_db-pass.tsv --format airr \
      --inf IMGT/HighV-QUEST:1.5.7.1 --organism "Homo sapiens" \
      --tissue "peripheral blood" --cell "naive B cell" \
      --cregion c_call --id --asn -sbt HD13M.sbt
```

Merge files and count V family usage

AIRR TSV files from different tools and easy combined to perform analysis on data generated using different software. Below is shown a simple V family usage analysis after merging the IgBLAST and IMGT/HighV-QUEST outputs into a single table:

```
# Count V family usage in R
# Imports
$> R
R> library(alakazam)
R> library(dplyr)
R> library(ggplot2)
```

(continues on next page)

(continued from previous page)

```
# Merge IgBLAST and IMGT/HighV-QUEST results
R> db_m <- read.delim("HD13M_fmt7_db-pass.tsv")
R> db_n <- read.delim("HD13N_imgt_db-pass.tsv")
R> db_m$cell_type <- "memory"
R> db_n$cell_type <- "naive"
R> db <- bind_rows(db_m, db_n)

# Subset to heavy chain
R> db <- subset(db, grepl("IGH", v_call))

# Count combined V gene usage
R> v_usage <- countGenes(db, "v_call", groups="cell_type",
                         mode="family")

# Plot V family usage
R> ggplot(v_usage, aes(x=GENE, y=SEQ_FREQ, fill=cell_type)) +
    geom_col(position="dodge") +
    scale_fill_brewer(name="Cell type", palette="Set1") +
    xlab("") +
    ylab("Fraction of repertoire")
```

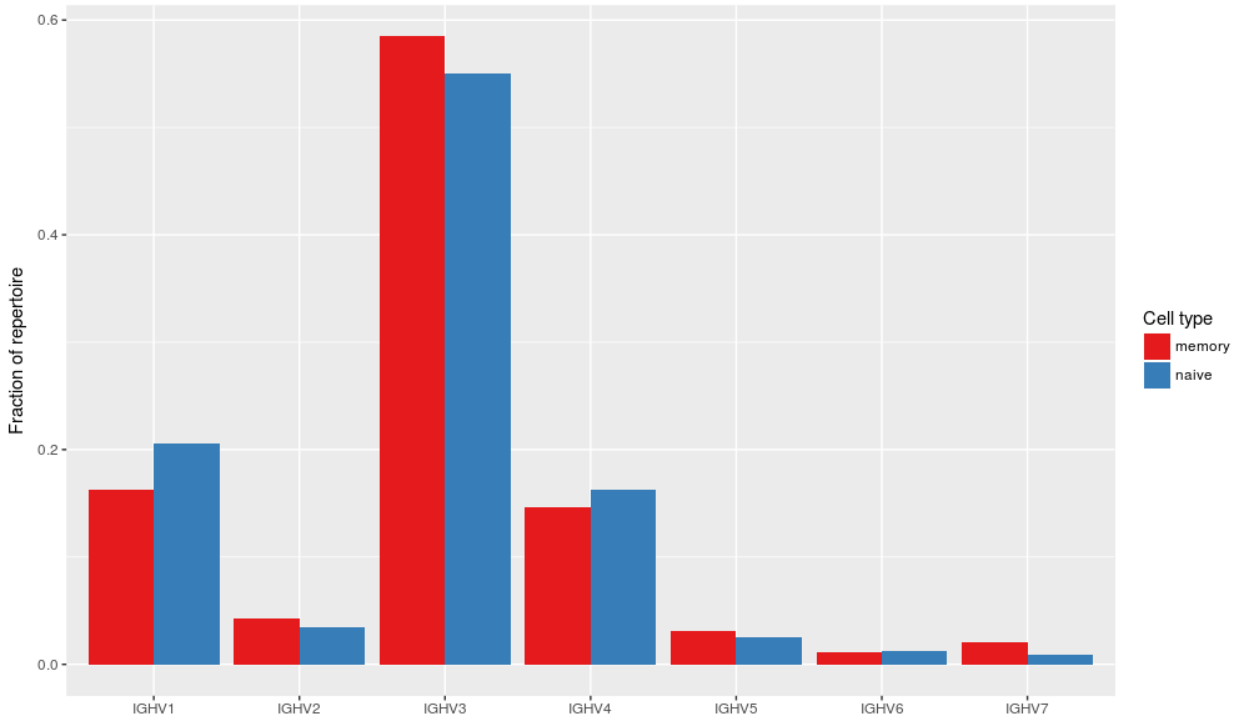


Fig. 2: **V family usage for the combined data set.**

Bibliography

[Rubelt_2017] Rubelt F *et al*. AIRR Community Recommendations for Sharing Immune Repertoire Sequencing Data. Nat Immunol 18:1274 (2017) DOI: 10.1038/ni.3873

[Breden_2017] Breden F *et al*. Reproducibility and Reuse of Adaptive Immune Receptor Repertoire Data. Front Immunol 8:1418 (2017) DOI: 10.3389/fimmu.2017.01418

[Zenodo_1185414] Release archive of the AIRR Standards repository. (2015-2018) DOI: 10.5281/zenodo.1185414

[LIGMDB_V12] IMGT-ONTOLOGY definitions. <http://www.imgt.org/ligmdb/label#JUNCTION>

[INSDC_FT] The DDBJ/ENA/GenBank Feature Table Definition. <http://www.insdc.org/documents/feature-table>

[ENA_MANUAL] European Nucleotide Archive Annotated/Assembled Sequences User Manual. <http://ftp.ebi.ac.uk/pub/databases/ena/sequence/release/doc/usrman.txt>

[GENBANK_FF] GenBank Flat File Format. <https://ftp.ncbi.nih.gov/genbank/gbrel.txt>

[GENBANK_SR] GenBank Sample Record. <https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html>

[INSDC_XREF] Controlled vocabulary for `/db_xref` qualifier. <http://www.insdc.org/documents/dbxref-qualifier-vocabulary>

[NCBI_NBK47528] SRA Handbook. <https://www.ncbi.nlm.nih.gov/books/NBK47528/>

[Rubelt_2017] Rubelt F *et al*. AIRR Community Recommendations for Sharing Immune Repertoire Sequencing Data. Nat Immunol 18:1274 (2017) DOI: 10.1038/ni.3873

[Breden_2017] Breden F *et al*. Reproducibility and Reuse of Adaptive Immune Receptor Repertoire Data. Front Immunol 8:1418 (2017) DOI: 10.3389/fimmu.2017.01418

[Zenodo_1185414] Release archive of the AIRR Standards repository. (2015-2018) DOI: 10.5281/zenodo.1185414

Index

Symbols

A

C

D

E

F

I

L

M

N

O

P

R

S

T

V

W