src/abif

April 28, 2025

Contents

This module provides a parser for ABIF (Applied Biosystems Input Format) files, commonly used for DNA sequencing data (e.g., .ab1 files).

The parser can read the binary format, extract key information such as sequences, quality values, and other metadata, and export the data to common bioinformatics formats like FASTA and FASTQ.

Example:

```
import abif

# Open an ABIF file
let trace = newABIFTrace("example.ab1")

# Get sequence data
echo trace.getSequence()

# Export to FASTA format
trace.exportFasta("output.fa")

# Close the file when done
trace.close()
```

1 Types

```
ABIFTrace = ref object
  stream*: FileStream        ## File stream
  fileName*: string          ## Path to the file
  version*: int              ## ABIF format version
  numElems*: int             ## Number of directory entries
  dataOffset: int            ## Offset to the directory
  tags*: TableRef[string, DirectoryEntry] ## Directory entries indexed by tag
  data*: TableRef[string, string] ## Extracted data values
```

Main object representing an ABIF file. Contains methods to extract and process the data.

```
DirectoryEntry = object
  tagName*: string           ## Four-character tag name
  tagNum*: int               ## Tag number
  elemType*: ElementType     ## Type of data stored
  elemSize*: int             ## Size of one element in bytes
  elemNum*: int              ## Number of elements
  dataSize*: int             ## Total size of data in bytes
  dataOffset*: int           ## Offset to the data in the file
  dataHandle*: int           ## Data handle (reserved)
```

Represents a directory entry in the ABIF file. Each entry contains metadata about a data element.

```
ElementType = enum
  etByte = 1, etChar = 2, etWord = 3, etShort = 4, etLong = 5, etRational = 6,
  etFloat = 7, etDouble = 8, etDate = 10, etTime = 11, etThumb = 12,
  etBool = 13, etPoint = 14, etRect = 15, etVPoint = 16, etVRect = 17,
  etPString = 18, etCString = 19, etTag = 20
```

Represents the data types that can be stored in an ABIF file.

2 Procs

```
proc abifVersion(): string {.raises: [], tags: [].}
```

Returns the version of the abif library.

```
proc close(trace: ABIFTrace) {.raises: [Exception, IOError, OSError],
                               tags: [WriteIOEffect].}
```

Closes the file stream associated with the trace.

```
proc exportFasta(trace: ABIFTrace; outFile: string = "") {.
    raises: [KeyError, IOError, OSError, ValueError],
    tags: [ReadIOEffect, WriteIOEffect].}
```

Exports the sequence to a FASTA format file.

Parameters: outFile: Path to the output file. If empty, ”trace.fa” is used.

```
proc exportFastq(trace: ABIFTrace; outFile: string = "") {.
    raises: [KeyError, IOError, OSError, ValueError],
    tags: [ReadIOEffect, WriteIOEffect].}
```

Exports the sequence and quality values to a FASTQ format file.

Parameters: outFile: Path to the output file. If empty, ”trace.fq” is used.

```
proc getData(trace: ABIFTrace; tag: string): string {.
    raises: [IOError, OSError, ValueError, KeyError], tags: [ReadIOEffect].}
```

Retrieves data for a specific tag.

Parameters: tag: The tag name to retrieve data for

Returns: The data as a string, or an empty string if the tag does not exist

```
proc getQualityValues(trace: ABIFTrace): seq[int] {.
    raises: [KeyError, IOError, OSError, ValueError], tags: [ReadIOEffect].}
```

Returns the sequence quality values as a sequence of integers.

Each value represents the quality score for the corresponding base in the sequence.

```
proc getSampleName(trace: ABIFTrace): string {.
    raises: [KeyError, IOError, OSError, ValueError], tags: [ReadIOEffect].}
```

Returns the sample name from the trace.

Uses the pre-extracted ”name” data or retrieves it from the SMPL1 tag.

```
proc getSequence(trace: ABIFTrace): string {.
    raises: [KeyError, IOError, OSError, ValueError], tags: [ReadIOEffect].}
```

Returns the DNA sequence from the trace.

Uses the pre-extracted ”sequence” data or retrieves it from the PBAS2 tag.

```
proc getTagNames(trace: ABIFTrace): seq[string] {.raises: [], tags: [].}
```

Returns a sequence of all tag names in the ABIF file.

```
proc newABIFTrace(filename: string; trimming: bool = false): ABIFTrace {.
    raises: [IOError, OSError, KeyError, ValueError], tags: [ReadIOEffect].}
```

Creates a new ABIFTrace object from the specified file.

Parameters: filename: Path to the ABIF file trimming: If true, low quality regions are trimmed (not implemented)

Returns: A new ABIFTrace object

Raises: IOError: If the file cannot be opened or is not a valid ABIF file