# src/abimetadata

April 26, 2025

## Contents

This module provides a command-line tool for displaying and modifying metadata in ABIF files.

The abimetadata tool allows users to:

1. List all human-readable metadata fields in an ABIF file
2. View the full content of a specific tag
3. Edit the value of a tag (currently limited to string-type tags)

Command-line usage:

```
abimetadata <input.ab1> [options]
```

Options:

| | |
|---|---|
| `-h, --help` | Show help message |
| `-l, --list` | List all metadata fields (default) |
| `-t, --tag=STRING` | Tag name to view or edit |
| `-v, --value=STRING` | New value for tag when editing |
| `-o, --output=STRING` | Output file for modified ABI file |
| `--limit=INT` | Limit number of tags displayed |
| `--version` | Show version information |
| `--debug` | Show additional debug information |

Examples:

```
# List all tags
abimetadata input.ab1

# View a specific tag's full content
abimetadata input.ab1 -t SMPL1

# Edit a tag
abimetadata input.ab1 -t SMPL1 -v "New Sample Name" -o modified.ab1
```

# 1 Imports

abif

# 2 Types

```
Config = object
  inputFile*: string          ## Path to the input ABIF file
  outputFile*: string         ## Path to the output file (when editing)
  tag*: string                ## Tag name to view or edit
  value*: string              ## New value for tag (when editing)
  listTags*: bool             ## Whether to list all tags (default behavior)
  debug*: bool                ## Whether to show debug information
  limit*: int                 ## Maximum number of tags to display (0 = no limit)
  showVersion*: bool          ## Whether to show version information
```

Configuration for the abimetadata tool. Contains command-line options and settings.

# 3 Procs

```
proc canDisplayTag(tagName: string; entry: DirectoryEntry): bool {.raises: [],
    tags: [], forbids: [].}
```

Determines if a tag can be displayed based on its name and properties.

Parameters: tagName: The name of the tag entry: The DirectoryEntry for the tag

Returns: true if the tag can be displayed, false otherwise

```
proc displaySingleTag(trace: ABIFTrace; tagName: string; debug: bool) {.
    raises: [KeyError], tags: [ReadIOEffect], forbids: [].}
```

Displays the full content of a single tag.

Parameters: trace: The ABIFTrace containing the tag tagName: The name of the tag to display debug: Whether to show debug information

```
proc formatTagValue(tagName: string; entry: DirectoryEntry; trace: ABIFTrace):
↪  string {.
    raises: [], tags: [ReadIOEffect], forbids: [].}
```

Formats a tag's value for display, with possible truncation for long values.

Parameters: tagName: The name of the tag entry: The DirectoryEntry for the tag trace: The ABIFTrace containing the tag

Returns: The tag's value as a string, possibly truncated for display

```
proc getFullTagValue(tagName: string; entry: DirectoryEntry; trace: ABIFTrace):
↪  string {.
    raises: [], tags: [ReadIOEffect], forbids: [].}
```

Gets the full, untruncated value of a tag.

Parameters: tagName: The name of the tag entry: The DirectoryEntry for the tag trace: The ABIFTrace containing the tag

Returns: The tag's value as a string, formatted according to its data type

```
proc isHumanReadableType(tagType: ElementType): bool {.raises: [], tags: [],
    forbids: [].}
```

Determines if a tag type can be displayed in a human-readable format.

Parameters: tagType: The ElementType to check

Returns: true if the type is human-readable, false otherwise

```
proc listMetadata(trace: ABIFTrace; debug: bool; limit: int = 0) {.
    raises: [IOError, KeyError], tags: [WriteIOEffect, ReadIOEffect],
    forbids: [].}
```

Lists all human-readable metadata fields in the ABIF file.

Parameters: trace: The ABIFTrace to list metadata from debug: Whether to show debug information limit: Maximum number of tags to display (0 = no limit)

```
proc main() {.raises: [ValueError, OSError],
              tags: [ReadIOEffect, ReadDirEffect, WriteIOEffect, ExecIOEffect],
              forbids: [].}
```

Main entry point for the abimetadata program.

Handles command-line parsing and executes the appropriate action based on the provided options (list, view, or edit tags).

```
proc modifyTag(trace: ABIFTrace; tagName: string; newValue: string;
               outputFile: string): bool {.raises: [KeyError],
    tags: [WriteIOEffect, ExecIOEffect, ReadDirEffect, ReadIOEffect],
    forbids: [].}
```

Modifies the value of a tag in an ABIF file.

```
proc parseCommandLine(): Config {.raises: [ValueError, OSError],
                                  tags: [ReadIOEffect], forbids: [].}
```

Parses command line arguments and returns a Config object.

This procedure:

- Initializes Config with default values
- Processes command-line arguments
- Handles special flags like –version and –help
- Validates required parameters based on operating mode

Returns: A Config object with settings based on command-line arguments

```
proc printHelp() {.raises: [], tags: [], forbids: [].}
```

Displays the help message for the abimetadata tool. Exits the program after displaying the message.

```
proc verifyTagUpdate(inputFile, outputFile, tagName: string): bool {.raises: [],
    tags: [ReadIOEffect, WriteIOEffect], forbids: [].}
```

Verifies that a tag was properly updated by comparing original and modified files.

Parameters: inputFile: Path to the original ABIF file outputFile: Path to the modified ABIF file tagName: The name of the tag that was modified

Returns: true if the tag was successfully updated, false otherwise

```
proc verifyTagUpdateBasic(inputFile, outputFile, tagName: string;
                          newValue: string; offset: int): bool {.discardable,
    raises: [], tags: [ReadIOEffect], forbids: [].}
```

Verifies tag update by directly checking the binary content at the specified offset This simpler method just checks if we can find the expected value at the offset

# 4 Exports

exportFastq, newABIFTrace, getQualityValues, getData, DirectoryEntry, ElementType, close, ABIFTrace, getTagNames, abifVersion, getSequence, getSampleName, exportFasta