Package ‘ReVAMP’

December 19, 2025

Type Package

Title Interface to 'Vamp' Audio Analysis Plugins

Version 1.0.0

Date 2025-12-03

Copyright See the file COPYRIGHTS

Description Provides an interface to the 'Vamp' audio analysis plugin system <https://www.vamp-plugins.org/> developed by Queen Mary University of London's Centre for Digital Music. Enables loading and running Vamp plugins for various audio analysis tasks including tempo detection, onset detection, spectral analysis, and audio feature extraction. Supports mono and stereo audio with automatic channel adaptation and domain conversion.

License GPL (>= 2)

Imports Rcpp

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), tuneR, knitr, rmarkdown, spelling

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.3

Encoding UTF-8

URL https://revamp.ebaker.me.uk/, https://github.com/edwbaker/ReVAMP

BugReports https://github.com/edwbaker/ReVAMP/issues

Language en-US

NeedsCompilation yes

Author Ed Baker [aut, cre] (ORCID: <https://orcid.org/0000-0002-5887-9543>),
Authors of vamp-sdk [cph] (Authorship and copyright of included
vamp-sdk as detailed in inst/COPYRIGHTS),
Mark Borgerding [cph] (Authorship and copyright of included kiss_fft as
detailed in inst/COPYRIGHTS)

Maintainer Ed Baker <ed@ebaker.me.uk>

Repository CRAN

Date/Publication 2025-12-19 14:20:08 UTC

Contents

`ReVAMP-package` *R Interface to Vamp Audio Analysis Plugins*

Description

Provides an R interface to the Vamp audio analysis plugin system developed by Queen Mary University of London's Centre for Digital Music. Enables loading and running Vamp plugins for Music Information Retrieval (MIR) tasks including tempo detection, onset detection, spectral analysis, and audio feature extraction.

Details

The ReVAMP package allows R users to access the extensive library of Vamp audio analysis plugins. Key functions include:

- `vampPlugins` - List all available Vamp plugins
- `runPlugin` - Execute a plugin on audio data
- `vampPluginParams` - Get plugin parameter information
- `vampPaths` - List plugin search paths
- `vampInfo` - Get Vamp SDK version information

See the individual function documentation for usage examples.

Author(s)

Your Name

Maintainer: Your Name <your@email.com>

References

Vamp Plugins: `https://www.vamp-plugins.org/`

Cannam, C., Landone, C., & Sandler, M. (2010). Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In Proceedings of the 18th ACM international conference on Multimedia (pp. 1467-1468).

See Also

vampPlugins, runPlugin, vampInfo

Examples

```
## Not run:
# List available plugins
plugins <- vampPlugins()
head(plugins)

# Get plugin search paths
vampPaths()

# Get info about a specific plugin
params <- vampParams("vamp-example-plugins:amplitudefollower")

# Get SDK version information
vampInfo()

## End(Not run)
```

`runPlugin` *Run a Vamp Plugin on Audio Data*

Description

Executes a Vamp audio analysis plugin on a Wave object and returns all outputs produced by the plugin. This is the main function for performing audio feature extraction and analysis.

Usage

```
runPlugin(
  wave,
  key,
  params = NULL,
  useFrames = FALSE,
  blockSize = NULL,
  stepSize = NULL,
  verbose = FALSE,
  dropIncompleteFinalFrame = TRUE
)
```

Arguments

`wave`	A Wave object from the `tuneR` package containing the audio data to analyze, or a character string specifying the path to a WAV file. Using a file path avoids loading the entire audio file into R memory, which can be more efficient for large files. Can be mono or stereo.

`key`	Character string specifying the plugin in "library:plugin" format (e.g., "vamp-example-plugins:amplitudefollower", "vamp-aubio-plugins:aubioonset"). Use `vampPlugins` to see available plugins and their keys.
`params`	Optional named list of parameter values to configure the plugin. Parameter names must match the parameter identifiers from `vampPluginParams`. Values will be coerced to numeric. If NULL (default), plugin default parameter values are used.
`useFrames`	Logical indicating whether to use frame numbers (TRUE) or timestamps (FALSE) in the output. Default is FALSE.
`blockSize`	Optional integer specifying the analysis block size in samples. If NULL (default), the plugin's preferred block size is used. For frequency domain plugins, this determines the FFT size and frequency resolution. Larger values give better frequency resolution but worse time resolution.
`stepSize`	Optional integer specifying the step size (hop size) in samples between successive analysis blocks. If NULL (default), the plugin's preferred step size is used. For frequency domain plugins, this defaults to blockSize/2. Smaller values give better time resolution but increase computation time.
`verbose`	Logical indicating whether to print progress messages and diagnostic information during plugin execution. Default is FALSE for quiet operation.
`dropIncompleteFinalFrame`	Logical indicating whether to drop the final analysis frame if it would require zero-padding due to insufficient samples. Default is TRUE, meaning only complete frames are processed. Set to FALSE to include zero-padded incomplete final frames.

Details

Many Vamp plugins produce multiple outputs. For example, an onset detector might output both "onsets" (discrete event times) and "detection_function" (a continuous measure). This function returns ALL outputs, allowing you to access whichever ones you need.

The plugin will automatically adapt to the audio characteristics:

- Channel mixing/augmentation if plugin requirements differ from input
- Time/frequency domain conversion as needed
- Buffering to handle different block sizes

Block Size and Step Size:

These parameters control the time/frequency resolution trade-off:

- **blockSize**: Size of each analysis window. For frequency domain plugins, this is the FFT size. Typical values: 512, 1024, 2048, 4096. Larger = better frequency resolution, worse time resolution.
- **stepSize**: Number of samples to advance between blocks (hop size). Typical values: blockSize/2 (50\ Smaller = better time resolution, more computation.

Each output data frame typically includes:

- **timestamp**: Time or frame number of the feature
- **duration**: Duration of the feature (if applicable, otherwise NA)
- **value/value1/value2/...**: Feature values (number of columns varies)
- **label**: Text label for the feature (if applicable, otherwise empty)

The function supports all three Vamp output sample types:

- **OneSamplePerStep**: Regular intervals based on step size
- **FixedSampleRate**: Output at a fixed rate (may differ from input)
- **VariableSampleRate**: Sparse output at irregular intervals

Value

A named list of data frames, one for each output produced by the plugin. The names correspond to the output identifiers (e.g., "amplitude", "onsets"). Each data frame contains columns for timestamp (or frame), duration, values, and labels (if applicable). If the plugin has only one output, the list will have one element.

See Also

`vampPlugins` to list available plugins, `vampPluginParams` to get plugin parameters

Examples

```
## Not run:
library(tuneR)

# Load audio file
audio <- readWave("myaudio.wav")

# Run amplitude follower plugin - returns list with one output
result <- runPlugin(
  wave = audio,
  key = "vamp-example-plugins:amplitudefollower"
)

# Access the amplitude output
amplitude_data <- result$amplitude
head(amplitude_data)

# Run onset detection - may return multiple outputs
result <- runPlugin(
  wave = audio,
  key = "vamp-aubio-plugins:aubioonset"
)

# See what outputs were produced
names(result)

# Access specific outputs
onsets <- result$onsets
```

```
detection_fn <- result$detection_function

# Run plugin with custom parameters
# First check what parameters are available
params_info <- vampPluginParams("vamp-aubio-plugins:aubioonset")
print(params_info)

# Set specific parameter values
result <- runPlugin(
  wave = audio,
  key = "vamp-aubio-plugins:aubioonset",
  params = list(threshold = 0.5, silence = -70)
)

# Run with custom block and step sizes for better time resolution
result <- runPlugin(
  wave = audio,
  key = "vamp-aubio-plugins:aubioonset",
  blockSize = 512,   # Smaller blocks for better time resolution
  stepSize = 128     # 75% overlap for smoother detection
)

# Run frequency domain plugin with larger FFT for better frequency resolution
result <- runPlugin(
  wave = audio,
  key = "qm-vamp-plugins:qm-chromagram",
  blockSize = 4096,  # Larger FFT for better frequency resolution
  stepSize = 2048    # 50% overlap (typical for frequency domain)
)

## End(Not run)
```

vampInfo *Get Vamp API and SDK Version Information*

Description

Returns information about the Vamp API and SDK versions used by ReVAMP.

Usage

```
vampInfo()
```

Value

A list with two elements:

API version Integer indicating the Vamp API version (typically 2)

SDK version Character string indicating the Vamp SDK version (e.g., "2.10")

Examples

```
## Not run:
# Get version information
vampInfo()

## End(Not run)
```

vampPaths *Get Vamp Plugin Search Paths*

Description

Returns the list of directories where ReVAMP searches for Vamp plugins. The search paths are determined by the Vamp Host SDK and typically include system-wide plugin directories and user-specific directories.

Usage

```
vampPaths()
```

Value

A character vector of directory paths

Examples

```
## Not run:
# List plugin search paths
vampPaths()

## End(Not run)
```

vampPluginParams *Get Parameters for a Specific Vamp Plugin*

Description

Returns detailed information about the configurable parameters for a given Vamp plugin. Parameters can be adjusted to customize plugin behavior.

Usage

```
vampPluginParams(key)
```

Arguments

`key`	Character string specifying the plugin key in "library:plugin" format (e.g., "vamp-aubio-plugins:aubionotes"). Use `vampPlugins` to get plugin IDs.

Value

A data frame with one row per parameter and columns including:

identifier Parameter identifier

name Human-readable parameter name

description Parameter description

unit Unit of measurement (if applicable)

min_value Minimum allowed value

max_value Maximum allowed value

default_value Default value

quantized Logical indicating if parameter is quantized to discrete values

Returns an empty data frame if the plugin has no configurable parameters.

See Also

`vampPlugins` to list available plugins

Examples

```
## Not run:
# Get parameters for aubio onset detector
params <- vampPluginParams("vamp-aubio-plugins:aubioonset")

## End(Not run)
```

`vampPlugins` *List All Available Vamp Plugins*

Description

Enumerates all Vamp plugins found in the plugin search paths and returns detailed information about each plugin including metadata, parameters, and audio processing requirements.

Usage

```
vampPlugins()
```

Value

A data frame with one row per plugin and columns including:

library Plugin library name (without extension)

name Human-readable plugin name

id Unique plugin identifier (library:plugin format)

plugin.version Plugin version number

vamp.api.version Vamp API version the plugin uses

maker Plugin author/creator

copyright Copyright information

description Detailed plugin description

input.domain Input domain: "Time Domain" or "Frequency Domain"

default.step.size Default step size in samples

default.block.size Default block size in samples

minimum.channels Minimum number of audio channels required

maximum.channels Maximum number of audio channels supported

See Also

`vampPluginParams` to get parameter information for a specific plugin

Examples

```
## Not run:
# List all installed plugins
plugins <- vampPlugins()

# Filter for specific library
aubio_plugins <- plugins[plugins$library == "vamp-aubio-plugins", ]

## End(Not run)
```

Index