Package: pipopaplot (via r-universe)

May 2, 2026

Title Seamless Data Sonification with 'ggplot2'

Version 0.0.2

Description Provides a simple framework for data sonification in R, mapping 'ggplot2'-style aesthetics to MIDI events. Designed for creative exploration rather than strict auditory data analysis.

License BSD_2_clause + file LICENSE

URL https://paithiov909.quarto.pub/pipopaplot/

BugReports https://github.com/paithiov909/pipopaplot/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, forcats, ggplot2, rlang, scales

Suggests testthat (>= 3.0.0)

LinkingTo cpp11

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository https://paithiov909.r-universe.dev

Date/Publication 2025-11-30 00:59:11 UTC

RemoteUrl https://github.com/paithiov909/pipopaplot

RemoteRef HEAD

RemoteSha 00c90268e2cfce981bc594d5918631c94a7c33a8

Contents

as_notes *Convert data to a standardized notes tibble*

Description

`as_notes()` converts an arbitrary data frame or a `ggplot` object into a tibble containing the required columns for MIDI sonification: `x`, `y`, `channel`, `group`, `duration`, and `velocity`.

Usage

```
as_notes(d, ...)

## Default S3 method:
as_notes(d, ...)

## S3 method for class 'ggplot'
as_notes(d, ..., .id = 1)
```

Arguments

`d`	A data frame or a `ggplot2` object. When a `ggplot2` object is given, the layer data is extracted using `ggplot2::get_layer_data()` with the layer index specified by `.id`.
`...`	Optional name mappings. Provide named arguments to override the default column names (e.g., `as_notes(d, duration = "fill")` will use `fill` as the duration column).
`.id`	Layer index to extract from a ggplot object. Defaults to `1`.

Details

The returned tibble serves as the standard 'notes' format accepted by `sonify()`. Numeric or logical columns are converted to numeric and missing values are replaced with `1`.

Value

A tibble with columns:

- `x`, `y` – coordinate-like values to be mapped to time and pitch
- `channel` – grouping for MIDI channels
- `group` – sub-group within each channel
- `duration`, `velocity` – numeric modifiers for note length and intensity

rollup *Aggregate or transform note data before sonification*

Description

`rollup()` provides a simple mechanism to summarize or reshape note-level data prior to sonification. By default, it performs numeric aggregation within each `(channel, group, by)` combination.

Usage

```
rollup(d, by, .fun = base::mean)
```

Arguments

`d`	A data frame returned by `as_notes()`.
`by`	A variable or expression used for grouping within each `(channel, group)` pair.
`.fun`	A summary function applied to numeric columns. Defaults to `base::mean()`.

Details

Although `rollup()` uses summarization by default, any transformation that returns a data frame with the expected columns will work in a sonification pipeline. Users can define their own custom `rollup` functions, including those that duplicate rows to generate chords or parallel voices.

Value

A data frame containing the same key columns (`x`, `y`, `channel`, `group`, `duration`, `velocity`) and suitable for use in `sonify()`.

sonify *Map note data to MIDI event timings and values*

Description

`sonify()` transforms a notes tibble into a structured data frame of note-on and note-off events that can be written to a Standard MIDI file via `write_midi()`.

Usage

```
sonify(
  notes,
  phrase_len = 4,
  tpq = 480,
  pitch_range = c(27, 102),
  vel_range = c(60, 100),
  duration_range = c(16, 4),
  offset = 0.1
)
```

Arguments

notes	A tibble containing columns x, y, channel, group, duration, and velocity. channel and group should be factors; other columns should be numeric.
phrase_len	Length of the phrase in beats.
tpq	Ticks per quarter note.
pitch_range	Range of MIDI note numbers to map y onto.
vel_range	Range of MIDI velocities to map velocity onto.
duration_range	Range of note durations (in ticks) mapped from duration. Default c(16, 4) (shorter to longer).
offset	Padding fraction for mapping x values.

Details

Value scaling is handled by scales::rescale(), so the input value ranges are arbitrary but must be finite. Note that because of this rescaling behavior, a velocity value of 0 will not be mapped to complete silence by default (i.e., the minimum input value will still be mapped to the lower bound of vel_range rather than 0).

Within each channel, duplicated timing values (x) are deduplicated using dplyr::distinct(). To produce simultaneous notes (like chords), you must assign different channel values in advance.

Value

A data frame with factor and integer columns: channel, group, tick_on, tick_off, pitch, and velocity. Suitable as input to write_midi().

---

write_midi *Write a data frame of notes to a Standard MIDI file*

Description

Write a data frame of notes to a Standard MIDI file

Usage

```
write_midi(notes, filename = "test.mid", opt = list())
```

Arguments

notes	A data frame containing columns tick_on, tick_off, pitch, and velocity. tick_on and tick_off should be integers. pitch and velocity should be integers. channel and group should be factors; other columns should be numeric.
filename	Name of the output file.
opt	A list of options.

Value

The path to the output file is invisibly returned.

Index