Package 'musicXML'

February 18, 2025

Type Package

Title Data Sonification using 'musicXML'

Version 1.0.1

Description A set of tools to facilitate data sonification and handle the 'musicXML' format <https://usermanuals.musicxml.com/MusicXML/Content/XS-MusicXML.htm>.
Several classes are defined for basic musical objects such as note pitch, note duration, note, measure and score.
Moreover, sonification utilities functions are provided, e.g. to map data into musical attributes such as pitch, loudness or duration.
A typical sonification workflow hence looks like: get data; map them to musical attributes; create and write the 'musicXML' score,
which can then be further processed using specialized music software (e.g. 'MuseScore', 'GuitarPro', etc.).
Examples can be found in the blog <https://globxblog.github.io/>, the presentation by Renard and Le Bescond (2022, <https://hal.science/hal-03710340v1>) or the poster by Renard et al. (2023, <https://hal.inrae.fr/hal-04388845v1>).

License GPL-3

Encoding UTF-8

LazyData true

Imports xml2

Suggests knitr, rmarkdown, tidyr, gganimate, av

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author Benjamin Renard [aut, cre, cph]
(<https://orcid.org/0000-0001-8447-5430>),
INRAE [fnd],
European Commission [fnd] (This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 835496)

Maintainer Benjamin Renard <benjamin.renard@inrae.fr>

Repository CRAN

Date/Publication 2025-02-18 11:10:02 UTC

Contents

`duration` *duration constructor.*

Description

Creates a new instance of a 'duration' object

Usage

```
duration(
  type,
  dot = FALSE,
  triplet = FALSE,
  mxlDivision = 96,
  mxlDuration = typeToMXLDuration(type, dot, triplet)
)
```

Arguments

type	Integer in 2^(0:6), note type (longest 1=whole, shortest 64=64th). 1 = whole, 2 = half, 4 = quarter, 8 = eighth, etc. down to 64 = 64th
dot	Logical, is note dotted?
triplet	Logical, is note triplet? (play 3 for 2)
mxlDivision	Positive integer, musicXML "division" defining the time resolution, i.e. the shortest possible note. It is expressed as a fraction of a quarter note. The value of 96 allows allows using 64th notes and their triplet/dotted versions.
mxlDuration	Positive integer, music XML "duration" expressed in number of mxlDivision's. In general, mxlDivision/mxlDuration should not be modified.

Value

An object of class 'duration'.

Examples

```
d <- duration(8,dot=TRUE)
```

---

durationMapping *Duration Mapping.*

Description

Map a series of values into a series of durations

Usage

```
durationMapping(x, expMin = 0, expMax = 6, qtrans = NULL, ...)
```

Arguments

x	Vector or data frame, series to be mapped
expMin	Integer, minimum type is 2^expMin (default: 1 <=> whole note)
expMax	Integer, maximum type is 2^expMax (default: 64 <=> 16th note)
qtrans	function, quantile transformation to be applied before mapping For instance data can be "normalized" by using qnorm.
...	further arguments to be passed to qtrans.

Value

A list of duration objects.

Examples

```
d <- durationMapping(x=rnorm(100))
```

getMeasures *Get measures*

Description

Create a series of measure objects from a series of notes.

Usage

```
getMeasures(notes, beats = 4, beatType = 4, mxlDivision = 96, ...)
```

Arguments

notes	list of notes (typically created by function getNotes).
beats	number of beats (defaut signature is 4/4).
beatType	beat type (defaut signature is 4/4).
mxlDivision	Positive integer, musicXML "division" defining the time resolution, i.e. the shortest possible note. It is expressed as a fraction of a quarter note. The value of 96 allows allows using 64th notes and their triplet/dotted versions.
...	further arguments to be passed to function measure (typically, keySignature)

Value

A list of measure objects.

Examples

```
m <- getMeasures(notes=getNotes(pitches=pitchMapping(x=rnorm(100))))
```

getNotes *Get notes.*

Description

Create a series of note objects from lists of pitches / durations / loudnesses

Usage

```
getNotes(
  pitches,
  durations = durationMapping(rep(0, length(pitches)), expMin = 4, expMax = 4),
  loudnesses = loudnessMapping(rep(0, length(pitches)), lMin = 89, lMax = 89)
)
```

Arguments

pitches	list of pitches (typically created by function pitchMapping)
durations	list of durations (typically created by function durationMapping)
loudnesses	list of loudnesses (typically created by function loudnessMapping)

Value

A list of note objects.

Examples

```
n <- getNotes(pitches=pitchMapping(x=rnorm(100)))
```

---

loudnessMapping *Loudness Mapping.*

Description

Map a series of values into a series of loudnesses

Usage

```
loudnessMapping(x, lMin = 18, lMax = 141, qtrans = NULL, ...)
```

Arguments

x	Vector or data frame, series to be mapped
lMin	Integer, minimum loudness (default corresponds to ppp)
lMax	Integer, maximum loudness (default corresponds to fff)
qtrans	function, quantile transformation to be applied before mapping For instance data can be "normalized" by using qnorm.
...	further arguments to be passed to qtrans.

Value

A vector of numerics representing loudnesses.

Examples

```
l <- loudnessMapping(x=rnorm(100))
```

`measure` *Measure constructor.*

Description

Creates a new instance of a 'measure' object

Usage

```
measure(
  number,
  notes,
  beats = 4,
  beatType = 4,
  keySignature = 0,
  mode = "major"
)
```

Arguments

`number`	Integer (>0), measure number.
`notes`	List of 'note' objects. Sum of notes durations should be compatible with the measure time signature
`beats`	Integer (>0), time signature is beats/beatType (default 4/4).
`beatType`	Integer (>0), time signature is beats/beatType (default 4/4).
`keySignature`	Integer, representing the number of flats (<0) or sharps (>0).
`mode`	Character, mode. Can be one of major, minor, dorian, phrygian, lydian, mixolydian, aeolian, ionian, locrian, and none.

Value

An object of class 'measure'.

Examples

```
notes=list(note(p=pitch('Db5'),d=duration(2)),note(p=pitch('B5'),d=duration(2)))
m <- measure(number=1,notes=notes)
```

MXLDurationToType *MXL duration to (type-dot-triplet)*

Description

Convert a MusicXML duration into a (type-dot-triplet), or a list of (type-dot-triplet) summing up to the requested duration. The requested MusicXML duration may be trimmed if it cannot be expressed as a multiple of the smallest available duration.

Usage

```
MXLDurationToType(mxlDuration, mxlDivision = 96)
```

Arguments

mxlDuration	Positive integer, music XML "duration" expressed in number of mxlDivision's.
mxlDivision	Positive integer, musicXML "division" defining the time resolution, i.e. the shortest possible note. It is expressed as a fraction of a quarter note. The value of 96 allows allows using 64th notes and their triplet/dotted versions.

Value

A list with the following fields:

1. type, numeric vector of types
2. dot, logical vector of dot flags
3. triplet, logical vector of triplet flags
4. exact, logical, FALSE if the requested duration had to be trimmed

Examples

```
MXLDurationToType(972)
```

note *Note constructor.*

Description

Creates a new instance of a 'note' object

Usage

```
note(p, d = duration(4), l = 89, tie2next = FALSE, tie2previous = FALSE)
```

Arguments

p	Pitch object (step, alter, octave).
d	Duration object (type, dot, triplet).
l	Numeric (>0), loudness expressed in percentage of a MIDI velocity of 90. Effective range 0-141 (larger values are clipped). 37: pp, 54: p, 71: mp, 89: mf, 107: f, 124: ff
tie2next	Logical, is note tied with next note?.
tie2previous	Logical, is note tied with previous note?.

Value

An object of class 'note'.

Examples

```
n <- note(p=pitch('Db5'))
```

---

pitch *Pitch constructor.*

Description

Creates a new instance of a 'pitch' object

Usage

```
pitch(string)
```

Arguments

string	character string comprising: (i) one letter in ABCDEFG (step) (ii) 'b' (flat), '#' (sharp) or '' (no alteration) (iii) one integer in 0:9 (octave).

Value

An object of class 'pitch'.

Examples

```
p <- pitch('Db5')
```

pitchMapping *Pitch Mapping.*

Description

Map a series of values into a series of pitches

Usage

```
pitchMapping(
  x,
  pitches = c("A4", "C5", "D5", "E5", "G5", "A5"),
  qtrans = NULL,
  ...
)
```

Arguments

x	Vector or data frame, series to be mapped
pitches	Vector of string, pitch scale (default: A minor pentatonic)
qtrans	function, quantile transformation to be applied before mapping For instance data can be "normalized" by using qnorm.
...	further arguments to be passed to qtrans.

Value

A list of pitch objects.

Examples

```
p <- pitchMapping(x=rnorm(100))
```

score *Score constructor.*

Description

Creates a new instance of a 'score' object

Usage

```
score(parts)
```

Arguments

parts	List, either a list of measures for a single-part score or a list of 'parts' (lists of mesures) for a multi-part score

Value

An object of class 'score'.

Examples

```
m1 <- measure(number=1,notes=list(note(p=pitch('Db5'),d=duration(2)),
                                  note(p=pitch('B5'),d=duration(2))))
m2 <- measure(number=2,notes=list(note(p=pitch('A5'),d=duration(2)),
                                  note(p=pitch('B5'),d=duration(2))))
s <- score(list(m1,m2))
```

---

tieNotes *Tie notes.*

Description

Add ties to successive notes having the same pitch.

Usage

```
tieNotes(notes)
```

Arguments

notes	list of notes (typically created by function getNotes)

Value

A list of note objects.

Examples

```
n <- tieNotes(getNotes(pitches=pitchMapping(x=rnorm(100))))
```

toMXL *Generic toMXL function*

Description

Generic toMXL function

Usage

```
toMXL(x)
```

Arguments

x	Object (note, measure or score)

Value

A MusicXML string.

Examples

```
toMXL(note(p=pitch('C5'),d=duration(1),l=107))
```

toMXL.duration *Duration to MXL (MusicXML)*

Description

Convert an object of class 'duration' into the corresponding MusicXML chunk

Usage

```
## S3 method for class 'duration'
toMXL(x)
```

Arguments

x	Duration to be converted.

Value

A MXL string.

Examples

```
toMXL(duration(8,dot=TRUE))
```

toMXL.measure *Measure to MXL (MusicXML)*

Description

Convert an object of class 'measure' into the corresponding MusicXML chunk

Usage

```
## S3 method for class 'measure'
toMXL(x)
```

Arguments

x	measure to be converted.

Value

A MusicXML string.

Examples

```
notes=list(note(p=pitch('Db5'),d=duration(2)),note(p=pitch('B5'),d=duration(2)))
m <- measure(number=1,notes=notes)
toMXL(m)
```

toMXL.note *Note to MXL (MusicXML)*

Description

Convert an object of class 'note' into the corresponding MusicXML chunk

Usage

```
## S3 method for class 'note'
toMXL(x)
```

Arguments

x	Note to be converted.

Value

A MusicXML string.

Examples

```
toMXL(note(p=pitch('Db5')))
```

toMXL.pitch *Pitch to MXL (MusicXML)*

Description

Convert an object of class 'pitch' into the corresponding MusicXML chunk

Usage

```
## S3 method for class 'pitch'
toMXL(x)
```

Arguments

x	Pitch to be converted.

Value

A MusicXML string.

Examples

```
toMXL(pitch('Db5'))
```

toMXL.score *Score to MXL (MusicXML)*

Description

Convert an object of class 'score' into the corresponding MusicXML chunk

Usage

```
## S3 method for class 'score'
toMXL(x)
```

Arguments

x	score to be converted.

Value

A MusicXML string.

Examples

```
m1 <- measure(number=1,notes=list(note(p=pitch('Db5'),d=duration(2)),
                                  note(p=pitch('B5'),d=duration(2))))
m2 <- measure(number=2,notes=list(note(p=pitch('A5'),d=duration(2)),
                                  note(p=pitch('B5'),d=duration(2))))
s <- score(list(m1,m2))
toMXL(s)
```

---

typeToMXLDuration *(type-dot-triplet) to MXL duration*

Description

Convert a (type-dot-triplet) into a MusicXML duration

Usage

```
typeToMXLDuration(type, dot = FALSE, triplet = FALSE, mxlDivision = 96)
```

Arguments

type	Integer in 2^(0:6), note type (longest 1=whole, shortest 64=64th). 1 = whole, 2 = half, 4 = quarter, 8 = eighth, etc. down to 64 = 64th
dot	Logical, is note dotted?
triplet	Logical, is note triplet? (play 3 for 2)
mxlDivision	Positive integer, musicXML "division" defining the time resolution, i.e. the shortest possible note. It is expressed as a fraction of a quarter note. The value of 96 allows allows using 64th notes and their triplet/dotted versions.

Value

An integer representing the mxl duration expressed in number of mxlDivision

Examples

```
typeToMXLDuration(type=8,dot=TRUE)
```

WaggaWagga *Wagga-Wagga dataset*

Description

Times series of monthly temperatures and precipitations recorded at Wagga-Wagga, New South Wales, Australia, 1940-2018

Usage

```
WaggaWagga
```

Format

An object of class `data.frame` with 79 rows and 3 columns.

Source

http://www.bom.gov.au/cgi-bin/climate/hqsites/site_data.cgi?period=annual&variable=meanT&station=072150

http://www.bom.gov.au/cgi-bin/climate/hqsites/site_data.cgi?period=annual&variable=rain&station=072150

writeMXL *writeMXL function*

Description

Write a score to a musicXML-formatted file

Usage

```
writeMXL(s, file, ...)
```

Arguments

`s`	Score, score object to be written
`file`	Character, destination file
`...`	additional arguments passed to method xml2::write_xml

Value

No return value, called for side effects.

Examples

```
m <- getMeasures(notes=getNotes(pitches=pitchMapping(x=rnorm(100))))
s <- score(m)
tfile= file.path(tempdir(),'myMusicXML.xml')
writeMXL(s,tfile)
file.remove(tfile)
```

Index