

Package ‘gm’

October 13, 2022

Type Package

Title Generate Music Easily and Show Them Anywhere

Version 1.0.2

Author Renfei Mao

Maintainer Renfei Mao <renfeimao@gmail.com>

Description Provides a simple and intuitive high-level language, with which you can create music easily. Takes care of all the dirty technical details in converting your music to musical scores and audio files. Works in 'R Markdown' documents <<https://rmarkdown.rstudio.com/>>, R 'Jupyter Notebooks' <<https://jupyter.org/>>, and 'RStudio' <<https://www.rstudio.com/>>, so you can embed generated music anywhere. Internally, uses 'MusicXML' <<https://www.musicxml.com/>> to represent musical scores, and 'MuseScore' <<https://musescore.org/>> to convert 'MusicXML'.

License MIT + file LICENSE

URL <https://github.com/flujoo/gm>, <https://flujoo.github.io/gm/>

Encoding UTF-8

RoxygenNote 7.1.1

Suggests rmarkdown, testthat

Imports base64enc, glue, htmltools, knitr, magick, magrittr, MASS, rlang, rstudioapi, stringr, utils

VignetteBuilder knitr

SystemRequirements MuseScore - <https://musescore.org/>

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-17 05:40:02 UTC

R topics documented:

+.Music	2
Clef	3
export	4
gm	5
inspect_errors	5
Key	6
Line	7
Meter	9
Music	10
show	11
Tempo	12
Tupler	13
tuplet	14
Index	15

+.Music	<i>Add Component to Music Object</i>
---------	--------------------------------------

Description

Add a component to a Music object.

Usage

```
## S3 method for class 'Music'
music + term
```

Arguments

music	A Music object.
term	A Line, Meter, Key, Clef or Tempo object.

Value

A list with class Music.

See Also

[Music\(\)](#) for initializing a Music object.

[Line\(\)](#), [Meter\(\)](#), [Key\(\)](#), [Clef\(\)](#) and [Tempo\(\)](#) for creating objects of corresponding classes.

Examples

```

# initialize a Music object
m <- Music()

# add a Line object
m <- m + Line(list("C4"), list(1))
m

# add a Meter object
m <- m + Meter(4, 4)
m

# add a Key object
m <- m + Key(1)
m

# add a Clef object
m <- m + Clef("G", to = 1)
m

# add a Tempo object
m <- m + Tempo(120)
m

```

Clef

*Create Clef Object***Description**

Create a Clef object.

Clef objects represent clefs.

Usage

```
Clef(sign, line = NULL, octave = NULL, to = NULL, bar = NULL, offset = NULL)
```

Arguments

sign	"G", "F" or "C", case insensitive.
line	Optional, 1 or 2 if sign is "G", an integer between 3 and 5 if sign is "F", or an integer between 1 and 5 if sign is "C".
octave	Optional, -1 or 1. octave can be specified only when sign is "G" and line is 2, or sign is "F" and line is 4.
to	an index or a Line name, which indicates to which Line object to add the Clef object.
bar	Optional. A positive integer which indicates the number of the measure to which to add the Clef object. By default, a Clef object will be added to the first measure.

offset Optional. A duration value, sum of duration values or 0, which indicates the position in a measure, at which to add the Clef object. The default value is 0.

Value

A list with class Clef.

See Also

[+.Music\(\)](#) for adding Clef objects to a Music object.

[vignette\("gm", package = "gm"\)](#) for details about duration values.

Examples

```
# create a Clef object
Clef("G", line = 2, octave = 1)

# add a Clef object to a Music object
Music() +
  Line(list("C4"), list(1)) +
  Clef("F", to = 1, bar = 10, offset = 1)
```

export

Export Object

Description

Export an object to various file formats.

Usage

```
export(x, dir_path, file_name, formats)
```

```
## S3 method for class 'Music'
export(x, dir_path, file_name, formats)
```

Arguments

x An object.

dir_path A single character which specifies the directory to which to export the object.

file_name A single character which specifies the name of the exported file(s).

formats A character vector which specifies the file formats. Supported file formats are "mscz", "mscx", "pdf", "png", "svg", "wav", "mp3", "flac", "ogg", "midi", "mid", "musicxml", "mxl", "xml", "metajson", "mlog", "mpos" and "spos".

Value

Invisible NULL.

Files with name `file_name` and with extensions `formats` are generated in `dir_path`.

Methods (by class)

- `Music`: export a `Music` object.

Examples

```
if (interactive()) {  
  m <- Music() + Meter(4, 4) + Line(list("C4"), list(4))  
  export(m, tempdir(), "x", c("mp3", "png"))  
}
```

gm

gm: Generate Music Easily and Show Them Anywhere

Description

Provides a simple and intuitive high-level language, with which you can create music easily. Takes care of all the dirty technical details in converting your music to musical scores and audio files. Works in R Markdown documents, R Jupyter Notebooks and RStudio, so you can embed generated music anywhere.

Author

Renfei Mao renfeimao@gmail.com

inspect_errors

See Full Error Report

Description

See a full error report when the error message is too long and thus shortened.

Usage

```
inspect_errors()
```

Value

Invisible NULL.

The full error report is printed in console.

Examples

```
## Not run:
Line(list(c, "p", NULL, 1:3, TRUE, NA_character_))

## End(Not run)
```

Key

Create Key Object

Description

Create a Key object.

Key objects represent key signatures.

Usage

```
Key(key, bar = NULL, to = NULL, scope = NULL)
```

Arguments

key	An integer between -7 and 7, which indicates the number of flat or sharp symbols in the key signature.
bar	Optional. A positive integer which indicates the number of the measure into which to insert the Key object. By default, a Key object will be inserted into the first measure(s).
to	Optional. A positive integer or a single character which indicates the Line object to which to add the Key object. By default, a Key object will be added to a whole Music object rather than to any specific Line object.
scope	Optional. "part" or "staff", which indicates whether to add the Key object to a whole part or only to a staff of a part, if the argument to is specified, or this argument will be ignored. The default value is "part".

Value

A list with class Key.

See Also

[+.Music\(\)](#) for adding Key objects to a Music object.

Examples

```
# create a Key object
Key(-7)

# insert a Key object into a specific measure
Music() + Key(7, bar = 2)

m <- Music() +
  Line(list("E5"), list(1), name = "a") +
  Line(list("C4"), list(1), name = "b", as = "staff")

# add a Key to a part
m + Key(2, to = "b")

# add a Key to a staff
m + Key(2, to = "b", scope = "staff")
```

Line	<i>Create Line Object</i>
------	---------------------------

Description

Create a Line object.
Line objects represent musical lines.

Usage

```
Line(
  pitches,
  durations,
  tie = NULL,
  name = NULL,
  as = NULL,
  to = NULL,
  after = NULL,
  bar = NULL,
  offset = NULL
)
```

Arguments

pitches A list whose members are

1. single pitch notations, like "C4", to represent the pitch contents of notes,
2. single MIDI note numbers, like 60 or "60", also to represent the pitch contents of notes,
3. single NAs to represent the pitch contents of rests, or

	4. vectors of pitch notations and MIDI note numbers, like <code>c("C4", "61")</code> , to represent the pitch contents of chords.
<code>durations</code>	A list whose members are <ol style="list-style-type: none"> 1. single duration notations or their abbreviations, like "quarter" or just "q", 2. single duration values, like 1, which is equivalent to "quarter", or 3. Duration objects returned by <code>tuplet()</code>, which is used to create complex tuplets.
<code>tie</code>	Optional. A list of indices of argument pitches, which indicates at which positions to add ties.
<code>name</code>	Optional. A single character to name the Line object.
<code>as</code>	Optional. "part", "staff" or "voice", to specify the state of the Line object. The default value is "part".
<code>to</code>	Optional. An index or a Line name, which indicates with which Line object as the reference to add the Line object.
<code>after</code>	Optional. A single logical which indicates whether to add the Line object after or before a reference Line object. The default value is TRUE.
<code>bar</code>	Optional. A positive integer which indicates the number of the measure to which to insert the Line object. By default, a Line object will be inserted to the first measure.
<code>offset</code>	Optional. A duration value, sum of duration values or 0, which indicates the position in a measure, at which to insert the Line object. The default value is 0.

Value

A list with class Line.

See Also

`+Music()` for adding Line objects to a Music object.

`vignette("gm", package = "gm")` for more details about Line objects.

Examples

```
# create a Music object
m <- Music() + Meter(4, 4) + Line(list("C4"), list(8), name = "a")

# create a Line object
l <- Line(
  pitches = list("C5", "C5", "C5"),
  durations = list(1, 1, 1),

  # tie the first two notes
  tie = list(1),

  # add the Line as a voice
  as = "voice",
```



```

# with Line "a" as reference
to = "a",

# before Line "a"
after = FALSE,

# insert the Line to bar 2 with offset 1
bar = 2,
offset = 1
)
l

# add the Line object to the Music object
m <- m + l
m

if (interactive()) {
  show(m)
}

```

Meter

Create Meter Object

Description

Create a Meter object.
 Meter objects represent time signatures.

Usage

```

Meter(
  number,
  unit,
  bar = NULL,
  actual_number = NULL,
  actual_unit = NULL,
  invisible = NULL
)

```

Arguments

number	A positive number to represent the upper numeral in a time signature symbol, which indicates how many beats are contained in each measure.
unit	1, 2, 4, 8, 16, 32 or 64 to represent the lower numeral in a time signature symbol, which indicates the duration of one beat.
bar	Optional. A positive integer which indicates the number of the measure into which to insert the Meter object. By default, a Meter object will be inserted into the first measure(s).

`actual_number`, `actual_unit`
 Optional, which defines the actual time signature rather than the time signature symbol on score. Usually used to create pickup measures. By default, these two arguments are the same with number and unit respectively.

`invisible`
 Optional. A single logical, which indicates whether to show the time signature symbol on score. The default value is FALSE.

Value

A list with class `Meter`.

See Also

[+.Music\(\)](#) for adding `Meter` objects to a `Music` object.

Examples

```
# create a 3/4 time signature
Meter(3, 4)

# insert a time signature into a specific measure
Music() + Meter(3, 4, bar = 10)

m <- Music() + Line(list("C5"), list(3))

# specify the actual time signature
ts <- Meter(3, 4, actual_number = 1, actual_unit = 4)
ts

if (interactive()) {
  show(m + ts)
}

# make a time signature invisible on score
if (interactive()) {
  ts <- Meter(3, 4, invisible = TRUE)
  show(m + ts)
}
```

Music

Initialize Music Object

Description

Initialize a `Music` object.

`Music` objects represent whole music pieces.

Usage

```
Music()
```

Details

A typical workflow with Music objects:

1. Initialize an empty Music object with `Music()`.
2. Add components to it with `+.Music()`.
3. Print it, or display it as musical score or audio file with `show()`, to check its structure.
4. Keep adding components and checking it until you get what you want.
5. Sometimes you may want to export the final Music object with `export()`.

Value

A list with class Music.

See Also

`+.Music()` for adding components to a Music object.

`show()` for displaying a Music object as musical score and audio file.

`export()` for exporting a Music object to various file formats.

Examples

```
# initialize a Music object
Music()

# print a Music object to check its structure
m <- Music() + Meter(4, 4) + Line(list("C4"), list(4))
m
```

show

Show Object

Description

Show an object as musical score or audio file.

Usage

```
show(x, to)
```

```
## S3 method for class 'Music'
show(x, to = NULL)
```

Arguments

x An object.

to Optional. A character vector which contains "score", "audio" or both, which indicates whether to show the object as musical score or audio file. The default value is "score".

Value

Invisible NULL.

The generated musical score or audio file is

1. showed in Viewer panel if show is called in RStudio,
2. included in generated HTML file if called in R Markdown document,
3. showed in output cell if called in R Jupyter Notebook, and
4. showed in user's browser if called in a normal R console.

Methods (by class)

- Music: show a Music object.

Examples

```
if (interactive()) {
  m <- Music() + Meter(4, 4) + Line(list("C4"), list(4))
  show(m, c("score", "audio"))
}
```

Tempo

Create Tempo Object

Description

Create a Tempo object.

Tempo objects represent tempo marks.

Usage

```
Tempo(tempo, unit = NULL, bar = NULL, offset = NULL)
```

Arguments

tempo	A number between 5 and 999 which indicates how many quarter notes per minute the tempo is.
unit	Optional. A duration notation, its abbreviation, or duration value corresponding to "whole", "half", "quarter", "eighth", "16th", with or without a dot. The default unit is "quarter".
bar	Optional. A positive integer which indicates the number of the measure at which to add the Tempo object. By default, a Tempo object will be added at the first measure.
offset	Optional. A duration value, sum of duration values or 0, which indicates the position in a measure, at which to add the Tempo object. The default value is 0.

Value

A list with class Tempo.

See Also

[+.Music\(\)](#) for adding Tempo objects to a Music object.

[vignette\("gm", package = "gm"\)](#) for details about duration notations and duration values.

Examples

```
# create a Tempo object
Tempo(200)

# set unit in a Tempo object
Tempo(120, unit = "half.")

# add Tempo objects to a Music object
Music() + Tempo(200) + Tempo(100, bar = 10, offset = 1)
```

Tupler

Create Tupler Object

Description

Create a Tupler object. Tupler objects are used in [tuplelet\(\)](#) to create tuplelets.

Usage

```
Tupler(n, unit = NULL, take = unit)
```

Arguments

n	A positive integer which indicates into how many parts to divide a duration.
unit, take	A duration type followed by zero to four dots, or its corresponding duration value.

Value

A list with class Tupler.

See Also

[tuplelet\(\)](#)

[vignette\("gm", package = "gm"\)](#) for a friendly guide to tuplelets.

Examples

```
# create a triplet quarter note
t <- Tupler(3, unit = "quarter", take = "quarter")
t

tuple("half", t)
```

tuple

Create Tuple

Description

Create a tuple.

Usage

```
tuple(duration, ...)
```

Arguments

duration	A duration notation, duration value, or Duration object.
...	Tupler objects returned by Tupler() , which specify how to divide the argument duration into parts, and how to take from these parts.

Value

A list with class Duration.

See Also

[Tupler\(\)](#)

`vignette("gm", package = "gm")` for a friendly guide to tuples.

Examples

```
# create a triplet quarter note
tuple("half", Tupler(3, unit = "quarter", take = "quarter"))
```

Index

`+.Music`, [2](#)
`+.Music()`, [4](#), [6](#), [8](#), [10](#), [11](#), [13](#)

`Clef`, [3](#)
`Clef()`, [2](#)

`export`, [4](#)
`export()`, [11](#)

`gm`, [5](#)

`inspect_errors`, [5](#)

`Key`, [6](#)
`Key()`, [2](#)

`Line`, [7](#)
`Line()`, [2](#)

`Meter`, [9](#)
`Meter()`, [2](#)
`Music`, [10](#)
`Music()`, [2](#), [11](#)

`show`, [11](#)
`show()`, [11](#)

`Tempo`, [12](#)
`Tempo()`, [2](#)
`Tupler`, [13](#)
`Tupler()`, [14](#)
`tuplet`, [14](#)
`tuplet()`, [13](#)