

# Package: music (via r-universe)

November 2, 2024

**Type** Package

**Title** Learn and Experiment with Music Theory

**Version** 0.1.3

**Maintainer** Efstathios D. Gennatas <gennatas@gmail.com>

**Description** An aid for learning and using music theory. You can build chords, scales, and chord progressions using 12-note equal temperament tuning (12-ET) or user-defined tuning. Includes functions to visualize notes on a piano using ASCII plots in the console and to plot waveforms using base graphics. It allows simple playback of notes and chords using the 'audio' package.

**Imports** graphics, utils, audio, crayon

**License** GPL (>=3)

**URL** <https://github.com/egenn/music>

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Repository** <https://egenn.r-universe.dev>

**RemoteUrl** <https://github.com/egenn/music>

**RemoteRef** HEAD

**RemoteSha** 8b0822fa987c5d00e413a8d2442e1ea1c67a3938

## Contents

music-package	2
buildChord	2
buildProgression	3
buildScale	4
cplot_piano	5
delay_time	6
formatNotation	7
formatNote	7

freq2wave . . . . .	8
mplot . . . . .	9
note2freq . . . . .	10
noteDistance . . . . .	11
playChord . . . . .	12
playFreq . . . . .	13
playNote . . . . .	14
playProgression . . . . .	15
playWave . . . . .	16
strings . . . . .	17

<b>Index</b>	<b>18</b>
--------------	-----------

---

music-package	<b>music:</b> <i>Learn and use music theory</i>
---------------	---

---

### Description

The music package allows you to build, play, and visualize scales, chords, and chord progression. For playback, **music** builds waveforms as matrices and passes them to the **audio** package which interfaces with the system's audio driver. The default notation and frequencies used throughout the package are based on twelve-tone equal temperament tuning (12ET). Custom tuning can be defined by specifying frequency ratios and a root note. See [note2freq](#). A4 defaults to 440Hz, and can be changed with the 'A4' argument.

---

buildChord	<i>Build Chord</i>
------------	--------------------

---

### Description

Build Chord

### Usage

```
buildChord(
  root,
  chord = "minor",
  plot = TRUE,
  play = FALSE,
  formatNotation = TRUE,
  ...
)
```

**Arguments**

root	String: Root note
chord	String: Chord to build. Default = "minor"
plot	Logical: If TRUE, plot chord notes using <a href="#">cplot_piano</a>
play	Logical: If TRUE, play chord using <a href="#">playChord</a>
formatNotation	Logical: If TRUE, format notes to include both flats and sharps to avoid repeating the same letter. e.g. convert c("Gb4", "G4") to c("F#4", "G4")
...	Additional arguments to be passed to <a href="#">playChord</a> if play = TRUE

**Author(s)**

E.D. Gennatas

**Examples**

```
buildChord("C4", "minor")
buildChord("A4", "sus2", plot = TRUE)
## Not run:
buildChord("B4", "sus2", play = TRUE)

## End(Not run)
```

---

buildProgression      *Build Chord Progression*

---

**Description**

Build Chord Progression

**Usage**

```
buildProgression(
  root = "A4",
  scale = "minor",
  plot = FALSE,
  play = FALSE,
  formatNotation = TRUE,
  ...
)
```

**Arguments**

root	String: Root note. Default = "A4"
scale	String: "major" or "minor". Default = "minor"
plot	Logical: If TRUE, plot each chord in the progression using <a href="#">cplot_piano</a>
play	Logical: If TRUE, play scale using <a href="#">playProgression</a>
formatNotation	Logical: If TRUE, format notes to include both flats and sharps to avoid repeating the same letter. e.g. convert c("Gb4", "G4") to c("F#4", "G4")
...	Additional arguments to be passed to <a href="#">playProgression</a> if play = TRUE

**Author(s)**

E.D. Gennatas

**Examples**

```
buildProgression("C4", "minor")
buildProgression("Bb4", "major")
## Not run:
buildProgression("Bb4", "major", play = TRUE, plot = TRUE)

## End(Not run)
```

---

buildScale

*Build Scale*

---

**Description**

Build Scale / Mode

**Usage**

```
buildScale(
  root,
  scale = "minor",
  descending = FALSE,
  plot = TRUE,
  play = FALSE,
  pairs = FALSE,
  formatNotation = TRUE,
  ...
)
```

**Arguments**

root	String: Root note. e.g. "C4"
scale	String: Scale to build. Default = "minor"
descending	Logical: If TRUE, return notes in descending order, otherwise in ascending
plot	Logical: If TRUE, plot scale notes using <a href="#">cplot_piano</a>
play	Logical: If TRUE, play scale using <a href="#">playNote</a>
pairs	Logical: If TRUE and play = TRUE, play the root note along with each other note, in sequence
formatNotation	Logical: If TRUE, format notes to include both flats and sharps to avoid repeating the same letter. e.g. convert c("Gb4", "G4") to c("F#4", "G4")
...	Additional arguments to be passed to <a href="#">playNote</a> if play = TRUE

**Author(s)**

E.D. Gennatas

**Examples**

```
buildScale("C4", "minor")
buildScale("B4", "minor", descending = TRUE, plot = TRUE)
## Not run:
buildScale("B4", "minor", descending = TRUE, play = TRUE, plot TRUE)

## End(Not run)
```

---

cplot\_piano

*Console piano plot for notes*

---

**Description**

Build an ASCII plot of notes on a piano

**Usage**

```
cplot_piano(notes = buildScale("C4", "minor"), blackKey.col = "white")
```

**Arguments**

notes	String, vector: Notes to highlight. Default = buildScale("C4", "minor")
blackKey.col	Color to use for black keys. Default = "white" for use on a dark terminal. Set to "black" for use on a light terminal.

**Author(s)**

E.D. Gennatas

**Examples**

```
cplot_piano(buildScale("B4", "minor"))
```

---

 delay\_time

*Delay and Reverb Time Calculator*


---

**Description**

Calculates delay and reverb time in milliseconds given tempo in beats per minute (BPM) and delay/reverb time in note duration

**Usage**

```
delay_time(
  bpm = 120,
  note = c("2", "1", "1/2", "1/2T", "1/4D", "1/4", "1/4T", "1/8D", "1/8", "1/8T",
    "1/16D", "1/16", "1/16T", "1/32D", "1/32", "1/32T"),
  verbose = TRUE
)
```

**Arguments**

bpm	Integer: Beats per minute. Default = 120
note	Character: Delay/Reverb time in note duration: "2", "1", "1/2", "1/2T", "1/4D", "1/4", "1/4T", "1/8D", "1/8", "1/8T", "1/16D", "1/16", "1/16T", "1/32D", "1/32", "1/32T". "2" means a double note, "1" a whole, and so on. "T" denotes a triple note, "D" denotes a dotted note. Case insensitive. Default = "1/4" (quarter note)
verbose	Logical: If TRUE print message to console

**Details**

This function originally appeared in the `rtemis` package

**Value**

Delay time, invisibly

**Author(s)**

E.D. Gennatas

**Examples**

```
delay_time(120, "1/8")
```

---

formatNotation	<i>Format Notation</i>
----------------	------------------------

---

### Description

Converts the internal note representation which uses flats, to the notation commonly used to write scales and chords, where a mix of sharps and flats is used to avoid repeating the same letter note. (e.g. "G#5" "A5", instead of "Ab5" "A5") e.g. convert the C4 Lydian from: "C4" "D4" "E4" "Gb4" "G4" "A4" "B4" "C5" to: "C4" "D4" "E4" "F#4" "G4" "A4" "B4" "C5" or convert the A4 major from: "A4" "B4" "Db5" "D5" "E5" "Gb5" "Ab5" "A5" to: "A4" "B4" "C#5" "D5" "E5" "F#5" "G#5" "A5"

### Usage

```
formatNotation(notes)
```

### Arguments

notes	String, vector: Notes to format
-------	---------------------------------

### Author(s)

E.D. Gennatas

### Examples

```
formatNotation(c("Db4", "D4", "E4", "Gb4", "G4", "A4", "B4", "C5"))
```

---

formatNote	<i>Format notes</i>
------------	---------------------

---

### Description

Format notes for use in other **music** functions

### Usage

```
formatNote(notes, default.octave = 4)
```

### Arguments

notes	Vector, String: Input notes in the form c("C4", "D4", "Eb4")
default.octave	Integer: Octave to use if missing in notes. Default = 4; i.e. "C" becomes "C4"

**Details**

Converts sharps to flats, adds octave number if missing (Default = 4), and converts (rare) "bb" notes to regular notes

**Author(s)**

E.D. Gennatas

**Examples**

```
formatNote(c("D#4", "Ebb"))
```

---

freq2wave

*Frequency to waveform*

---

**Description**

Frequency to waveform

**Usage**

```
freq2wave(
  frequency,
  oscillator = c("sine", "square", "saw", "triangle"),
  duration = 1,
  BPM = 120,
  sample.rate = 44100,
  attack.time = 50,
  inner.release.time = 50,
  plot = FALSE
)
```

**Arguments**

frequency	Float, vector: Frequency/ies to convert to waveform
oscillator	String: "sine", "square", "saw". Default = "sine"
duration	Float: Note duration in beats. Default = 1
BPM	Integer: Beats per minute. Default = 120
sample.rate	Integer: Sample rate. Default = 44100
attack.time	Integer: Attack time. Default = 50 (Helps prevent popping)
inner.release.time	Integer: Release time, that ends on note OFF (instead of beginning at note OFF). Default = 50 (Also helps prevent popping)
plot	Logical: If TRUE, plot wave(s) using <a href="#">mplot</a>



**Author(s)**

E.D. Gennatas

**Examples**

```
wave <- freq2wave(note2freq(buildChord("A4", "sus2")))
```

---

**mplot***Plot waveform*

---

**Description**

Plot waveform

**Usage**

```
mplot(  
  x,  
  type = "l",  
  main = NULL,  
  legend = TRUE,  
  lwd = 1,  
  pty = "m",  
  bg = "black",  
  fg = "gray50",  
  col = "cyan",  
  col.axis = "gray50",  
  col.lab = "gray50",  
  col.main = "gray80",  
  col.legend = "white",  
  tcl = 0.3,  
  xaxt = "s",  
  yaxt = "s",  
  new = FALSE,  
  mgp = c(2, 0, 0),  
  mar = NULL,  
  oma = NULL,  
  ...  
)
```

**Arguments**

x	Input
type	String: "l" for lines, "p" for points. Default = "l"
main	String: Plot title
legend	Logical: If TRUE, show legends on plot, if x has column names

lwd	Float: Line width. Default = 1
pty	String: "m" to fill available device space, "s" for square plot. Default = "m"
bg	Color: background color
fg	Color: foreground color
col	Color: Point/line color
col.axis	Color: Axes' color
col.lab	Color: Label color
col.main	Color: Title color
col.legend	Color: Legend color
tcl	The 'tcl' param of par
xaxt	The 'xaxt' param of par
yaxt	The 'yaxt' param of par
new	The 'new' param of par
mgp	The 'mgp' param of par
mar	Vector, length 4: Margins for par
oma	Vector, length 4: The 'oma' param of par
...	Additional parameters to pass to plot

**Author(s)**

E.D. Gennatas

---

note2freq

---

*Convert musical notes to frequencies*


---

**Description**

Convert notes to frequencies

**Usage**

```

note2freq(
  note,
  tuning = c("12ET", "custom"),
  custom.ratios = c(1, 16/15, 9/8, 6/5, 5/4, 4/3, 45/32, 3/2, 8/5, 5/3, 9/5, 15/8, 2),
  A4 = 440,
  custom.root = "C",
  default.octave = 4
)

```

**Arguments**

note	String: Note(s) to convert to frequencies
tuning	String: "12ET": 12-note equal temperament, "custom": Intonation defined by customRatios
custom.ratios	Numeric, vector, length 13: Custom ratios for a 12-note scale, starting with 1 (root) and ending in 2 (octave) to use when tuning = "custom". The A4 note will be set to A4 Hz and the rest of the frequencies will be built based on these ratios and the customRoot
A4	Float: Frequency for A4 in Hz. Default = 440
custom.root	String: Root note for just intonation (tuning = "custom"). Default = "C"
default.octave	Integer: If note is provided without octave number (e.g. "C"), default to this octave. Default = 4

**Author(s)**

E.D. Gennatas

**Examples**

```
note2freq(buildScale("B4", "minor"))
```

---

noteDistance	<i>Note distance in semitones</i>
--------------	-----------------------------------

---

**Description**

Calculates note distance in semitones

**Usage**

```
noteDistance(notes)
```

**Arguments**

notes	String, vector: Notes in form c("C4", "Eb4", "Gb4")
-------	---

**Value**

Vector of length length(notes) with semitone distances between successive notes

**Author(s)**

E.D. Gennatas

**Examples**

```
noteDistance(strings("C4 Eb4 Gb4 Bb4"))
```

---

 playChord

*Play Chord*


---

**Description**

Play Chord

**Usage**

```

playChord(
  chord,
  type = c("harmonic", "ascending", "descending"),
  oscillator = "sine",
  duration = 1,
  sample.rate = 44100,
  attack.time = 50,
  inner.release.time = 50,
  A4 = 440,
  plot = FALSE,
  ...
)

```

**Arguments**

chord	String, vector: Notes making up chord. e.g. c("A4", "C5", "E5"). e.g. output of <a href="#">buildChord</a>
type	String: "harmonic", "ascending", "descending". Default = "harmonic"
oscillator	String: "sine", "square", "saw". Default = "sine"
duration	Float: Note duration in beats. Default = 1
sample.rate	Integer: Sample rate. Default = 44100
attack.time	Integer: Attack time. Default = 50 (Helps prevent popping)
inner.release.time	Integer: Release time, that ends on note OFF (instead of beginning at note OFF). Default = 50 (Also helps prevent popping)
A4	Float: Frequency for A4 in Hz. Default = 440
plot	Logical: If TRUE, plot chord using <a href="#">cplot_piano</a>
...	Additional arguments to pass to <a href="#">note2freq</a>

**Value**

The constructed waveform (invisibly)

**Author(s)**

E.D. Gennatas

**Examples**

```
## Not run:
playChord(buildChord("E4", "minor"))

## End(Not run)
```

---

playFreq	<i>Play frequency</i>
----------	-----------------------

---

**Description**

Play frequency

**Usage**

```
playFreq(
  frequency,
  oscillator = "sine",
  duration = rep(1, length(frequency)),
  BPM = 120,
  sample.rate = 44100,
  attack.time = 50,
  inner.release.time = 50,
  plot = FALSE
)
```

**Arguments**

frequency	Numeric, Vector: Frequency / frequencies to play
oscillator	String: "sine", "square", "saw". Default = "sine"
duration	Float: Note duration in beats. Default = 1
BPM	Integer: Beats per minute. Default = 120
sample.rate	Integer: Sample rate. Default = 44100
attack.time	Integer: Attack time. Default = 50 (Helps prevent popping)
inner.release.time	Integer: Release time, that ends on note OFF (instead of beginning at note OFF). Default = 50 (Also helps prevent popping)
plot	Logical: If TRUE, plot waveform

**Author(s)**

E.D. Gennatas

**Examples**

```
## Not run:
playFreq(440)

## End(Not run)
```

---

playNote	<i>Play Note</i>
----------	------------------

---

**Description**

Play Note

**Usage**

```
playNote(
  note,
  oscillator = "sine",
  duration = rep(1, length(note)),
  BPM = 120,
  sample.rate = 44100,
  attack.time = 50,
  inner.release.time = 50,
  A4 = 440,
  plot = FALSE,
  ...
)
```

**Arguments**

note	String, Vector: Note(s) to be played, e.g. c("Ab4", "B4")
oscillator	String: "sine", "square", "saw". Default = "sine"
duration	Float: Note duration in beats. Default = 1
BPM	Integer: Beats per minute. Default = 120
sample.rate	Integer: Sample rate. Default = 44100
attack.time	Integer: Attack time. Default = 50 (Helps prevent popping)
inner.release.time	Integer: Release time, that ends on note OFF (instead of beginning at note OFF). Default = 50 (Also helps prevent popping)
A4	Float: Frequency for A4 in Hz. Default = 440
plot	Logical: If TRUE, plot notes using <a href="#">cplot_piano</a> . This supports only two octaves; do not try plotting if your notes span more than two octaves.
...	Additional arguments to pass to <a href="#">note2freq</a>

**Author(s)**

E.D. Gennatas

**Examples**

```
## Not run:
playNote("B4")

## End(Not run)
```

---

playProgression	<i>Play Progression</i>
-----------------	-------------------------

---

**Description**

Play Progression

**Usage**

```
playProgression(
  progression,
  oscillator = c("sine", "square", "saw", "triangle"),
  duration = 1,
  BPM = 120,
  sample.rate = 44100,
  attack.time = 50,
  inner.release.time = 50,
  A4 = 440,
  plot = FALSE,
  ...
)
```

**Arguments**

progression	List of string vectors: Each element of the list is a chord. e.g. output of <a href="#">build-Progression</a>
oscillator	String: "sine", "square", "saw". Default = "sine"
duration	Float: Note duration in beats. Default = 1
BPM	Integer: Beats per minute. Default = 120
sample.rate	Integer: Sample rate. Default = 44100
attack.time	Integer: Attack time. Default = 50 (Helps prevent popping)
inner.release.time	Integer: Release time, that ends on note OFF (instead of beginning at note OFF). Default = 50 (Also helps prevent popping)
A4	Float: Frequency for A4 in Hz. Default = 440
plot	Logical. If TRUE, plot each chord in the progression using <a href="#">cplot_piano</a>
...	Additional arguments to pass to <a href="#">note2freq</a>

**Author(s)**

E.D. Gennatas

**Examples**

```
## Not run:  
playProgression(buildProgression("G4", "minor"))  
  
## End(Not run)
```

---

playWave

*Minimal "Polyphonic" Wave Player*

---

**Description**

Play one or more waveforms at the same time using `audio::play`

**Usage**

```
playWave(wave, sample.rate = 44100, plot = FALSE)
```

**Arguments**

<code>wave</code>	Matrix or vector of waveforms. If a matrix, each column should be a waveform to be played simultaneously
<code>sample.rate</code>	Integer: Sample rate. Default = 44100
<code>plot</code>	Logical: If TRUE: plot wave using <a href="#">mplot</a> .

**Author(s)**

E.D. Gennatas

**Examples**

```
## Not run:  
playWave(freq2wave(440))  
  
## End(Not run)
```



---

strings	<i>Separate notes into vector of strings</i>
---------	--

---

**Description**

Convenience function to separate notes into vector of strings

**Usage**

```
strings(x, sep = " ")
```

**Arguments**

x	String: A single character object which consists of multiple notes separated by sep e.g. "C4 Eb4 G4 D5"
sep	String: the character that separates notes in x. Default = " "

**Details**

Makes it easy to copy-paste notes into other functions e.g. `playChord(strings("C4 Eb4 G4 D5"))`

**Author(s)**

E.D. Gennatas

**Examples**

```
strings("C4 Eb4 Gb4 Bb4")
```

# Index

buildChord, [2](#), [12](#)  
buildProgression, [3](#), [15](#)  
buildScale, [4](#)  
  
cplot\_piano, [3–5](#), [5](#), [12](#), [14](#), [15](#)  
  
delay\_time, [6](#)  
  
formatNotation, [7](#)  
formatNote, [7](#)  
freq2wave, [8](#)  
  
mplot, [8](#), [9](#), [16](#)  
music-package, [2](#)  
  
note2freq, [2](#), [10](#), [12](#), [14](#), [15](#)  
noteDistance, [11](#)  
  
playChord, [3](#), [12](#)  
playFreq, [13](#)  
playNote, [5](#), [14](#)  
playProgression, [4](#), [15](#)  
playWave, [16](#)  
  
strings, [17](#)