Package ‘music’

April 20, 2019

Type Package
Title Learn and Experiment with Music Theory
Version 0.1.1
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
LazyData true
RoxygenNote 6.1.1
NeedsCompilation no
Author Efstathios D. Gennatas [aut, cre]
Repository CRAN
Date/Publication 2019-04-20 16:00:03 UTC

R topics documented:

music-package .................................................. 2
buildChord .................................................. 2
buildProgression ............................................. 3
buildScale .................................................... 4
cplot.piano ................................................... 5
formatNotation ................................................ 5
formatNote ..................................................... 6
freq2wave .................................................... 7
mplot .......................................................... 8
note2freq ...................................................... 9
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.

Description

Build Chord

Usage

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

Arguments

root        String: Root note
chord       String: Chord to build. Default = "minor"
play        Logical: If TRUE, play chord using playChord
plot        Logical: If TRUE, plot chord notes using cplot.piano
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 playChord if play = TRUE
Author(s)

Efstathios 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", play = FALSE,
plot = FALSE, formatNotation = TRUE, ...)

Arguments

root String: Root note. Default = "A4"
scale String: "major" or "minor". Default = "minor"
play Logical: If TRUE, play scale using playProgression
plot Logical: If TRUE, plot each chord in the progression using cplot.piano
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 playProgression if play = TRUE

Author(s)

Efstathios 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, play = FALSE, pairs = FALSE, plot = 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 order. Default = FALSE
- **play**  
  Logical: If TRUE, play scale using playNote
- **pairs**  
  Logical: If TRUE and play = TRUE, play the root note along with each other note, in sequence
- **plot**  
  Logical: If TRUE, plot scale notes using cplot.piano
- **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 playNote if play = TRUE

Author(s)

Efstathios 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

---

**cplot.piano**

*Console piano plot for notes*

---

**Description**

Build an ASCII plot of notes on a piano

**Usage**

```r
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)**

Efstathios D. Gennatas

**Examples**

```r
cplot.piano(buildscale("B4", "minor"))
```

---

**formatNotation**

*Format Notation*

---

**Description**

Converts the internal note representation which uses only 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" "Gb5" "A5"

**Usage**

```r
formatNotation(notes)
```

**Arguments**

- **notes**: String, vector: Notes to format
Author(s)

Efstathios D. Gennatas

Examples

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

formatNote

Format notes

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)

Efstathios D. Gennatas

Examples

formatNote(c("D#4", "Ebb"))
**freq2wave**

*Frequency to waveform*

**Description**

Frequency to waveform

**Usage**

```r
call2wave(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", "triangle". 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 `mplot`

**Author(s)**

Efstathios D. Gennatas

**Examples**

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

**Plot waveform**

**Description**

Plot waveform

**Usage**

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

**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)**

Efstathios D. Gennatas
Note2Freq

Convert musical notes to frequencies

Description
Convert notes to frequencies

Usage
```r
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)
Efstathios D. Gennatas

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

NoteDistance
Note distance in semitones

Description
Calculates note distance in semitones

Usage
```r
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)**

Efstathios D. Gennatas

**Examples**

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

---

**Description**

Play Chord

**Usage**

```r
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 `buildChord`

- **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 `cplot.piano`

- **...**
  Additional arguments to pass to `note2freq`
**playFreq**

**Value**

The constructed waveform (in invisibly)

**Author(s)**

Efstathios D. Gennatas

**Examples**

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

## End(Not run)
```

---

**playFreq**

*Play frequency*

**Description**

Play frequency

**Usage**

```r
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)**

Efstathios D. Gennatas
Examples

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

## End(Not run)
```

---

**Description**

Play Note

**Usage**

```r
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 `cplot.piano`. This support only two octaves; do not try plotting if your notes span more than two octaves.
- **...** Additional arguments to pass to `note2freq`

**Author(s)**

Efstathios D. Gennatas

**Examples**

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

## End(Not run)
```
**playProgression**  

**Play Progression**

### Description

Play Progression

### Usage

```r
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 `buildProgression`
- **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 `cplot.piano`
- **...**: Additional arguments to pass to `note2freq`

### Author(s)

Efstathios D. Gennatas

### Examples

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

## End(Not run)
```
playWave

**Description**

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

**Usage**

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

**Arguments**

- `wave`: Matrix or vector of waveforms. If a matrix, each column should be a waveform to be played simultaneously
- `sample.rate`: Integer: Sample rate. Default = 44100
- `plot`: Logical: If TRUE: plot wave using `mplot`.

**Author(s)**

Efstathios D. Gennatas

**Examples**

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

## End(Not run)
```

---

**strings**

**Description**

Convenience function to separate notes into vector of strings

**Usage**

```r
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)**

Efstathios D. Gennatas

**Examples**

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

buildChord, 2, 10
buildProgression, 3, 13
buildScale, 4

cplot.piano, 2–4, 5, 10, 12, 13

formatNotation, 5
formatNote, 6
freq2wave, 7

mplot, 7, 8, 14
music-package, 2

note2freq, 2, 9, 10, 12, 13
noteDistance, 9

playChord, 2, 10
playFreq, 11
playNote, 4, 12
playProgression, 3, 13
playWave, 14

strings, 14