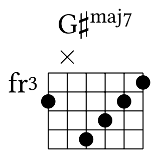


# Chordbox

**A LaTeX package for drawing string instrument chord diagrams**

Steven Franzen

v0.3 December 13, 2018
`https://github.com/sfranzen/chordbox`

## Contents

## 1 Package introduction

This package is the result of a search similar to the one undertaken by Clemens Niederberger for his `leadsheets`[1] package: over the years I have collected many textual guitar tabs and chord sheets that I finally wanted to put together and typeset properly using LaTeX, including guitar chord diagrams. The first part of my requirements is now more than fulfilled by `leadsheets`, which provides all the tools for putting together chords and lyrics. For the second part I found that there were only two relevant existing packages: `guitarchordschemes`[2] and `gchords`.[3] Neither of these achieved exactly what I wanted, so this package is my own attempt at providing that tool.

Note that I am far from a TeX guru; I know a bit of LaTeX and also have some familiarity with Ti*k*Z, so I have gratefully used the higher level interfaces provided by said package as well as `xifthen`[4] and `xstring`[5] for inspecting arguments. The PGF/Ti*k*Z packages provide a powerful and versatile framework for creating vector graphics using LaTeX commands paired with a special drawing syntax. One advantage of this is that you can fully integrate graphics into your document that will always look great: they will use the document fonts and can be rendered at any size you desire, avoiding some issues associated with the use of external images. This does not come without a cost, however; although the syntax introduced by Ti*k*Z is concise and easy to read, composing images will require some thought, multiple commands and adjustments and can be labour-intensive. Also, there is simply so much functionality that the official manual[6] currently spans almost 1200 pages of dense, though amply illustrated information.

The main aim of the present package is therefore to provide a way of typesetting chord diagrams using Ti*k*Z, while sparing the user from the need to plough through its documentation to learn the required drawing syntax. It does this by defining two commands that require only a few arguments and encapsulate the necessary drawing instructions. Their basic usage is introduced in Section 2, along with examples illustrating the various possibilities. Further information needed by the drawing algorithm is stored using PGF/Ti*k*Z' native key-value management system, which also makes it quite easy to expose settings for modification by the user, allowing you to influence the appearance of the resulting chord boxes. These are documented in Section 3, with some examples of the commands you can use to change them.

### 1.1 Loading the package

`chordbox` currently does not support LaTeX options, so it is loaded simply by putting

```
\usepackage{chordbox}
```

in your document's preamble.

[1]Niederberger, *leadsheets.*
[2]Niederberger, *guitarchordschemes.*
[3]Peeters, *gchords.*
[4]Noirel, *xifthen.*
[5]Tellechea, *xstring.*
[6]Tantau, *The TikZ and PGF Packages.*

### 1.2 License

## 2 Usage

The package provides two similar commands, `\chordbox` and `\bchordbox`. The latter extends the former to draw barred chords.

### 2.1 The `\chordbox` command

Syntax: `\chordbox[`⟨*base fret*⟩`]{`⟨*chord name*⟩`}{`⟨*fret positions*⟩`}`

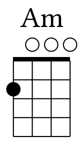
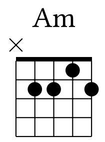
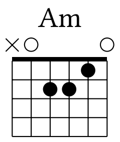
The simplest form of the command requires you to specify only the ⟨*chord name*⟩ and ⟨*fret positions*⟩. The former can contain any text and math symbols (see below), the latter should be a comma-separated list of elements, one for each string, which may take the following values:

- ⟨*fret number*⟩[:⟨*fingering text*⟩]
  A (positive) ⟨*fret number*⟩ marks the current string as fretted at that position. It may optionally be followed by a colon and a ⟨*fingering text*⟩ that can be displayed on the fret symbol or below the chord box.

- *0* (zero)
  Marks the current string as open (unfretted).

- *Anything else*
  An empty element or one starting with anything other than a number marks the current string as muted (not played).

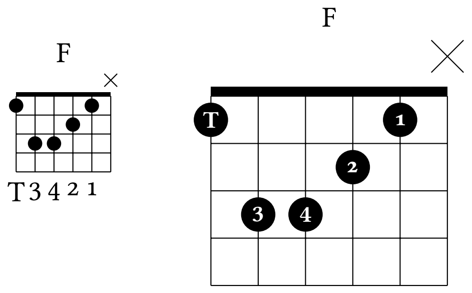
It draws a grid resembling vertical strings crossing horizontal frets, with a black bar at the top figuring as the instrument's nut. The number of strings drawn is determined by the number of ⟨*fret positions*⟩ passed to the command, so generating chord diagrams for different instruments is no problem:

| Ukulele | 5-string banjo | 6-string guitar |
|---|---|---|
| `\chordbox{Am}{2,0,0,0}` | `\chordbox{Am}{x,2,2,1,2}` | `\chordbox{Am}{x,0,2,2,1,0}` |
|  |  |  |

The number of frets drawn is initially 4, but this can be changed through the configuration key `/chordbox/numfrets`.

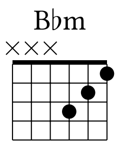
As mentioned, you can optionally provide fingering information for a chord, which is particularly useful for writing chord charts and other training materials. Because text on the fret symbol has to be scaled to fit inside it and the default scale is small, `chordbox` is initially configured to put this text below the strings, where it can be rendered at a larger size. However, the other option may be more appealing for larger diagrams, so both options can be used, see `/chordbox/text below` and `/chordbox/text on node`. They are illustrated below, where the letter T is used for the thumb and the digits for the other fingers, starting at 1 for the index finger. The larger chord box on the right has been given a scale of 2.5 times the default.

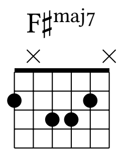
```
\chordbox{F}{1:T,3:3,3:4,2:2,1:1,x}
```



By default, the ⟨*chord name*⟩ is processed by a command that typesets it in math mode and roman type (see `/chordbox/name`). This allows the use of math symbols like `\flat` and `\sharp` as well as the `^` (superscript) operator to typeset decorated chord names:

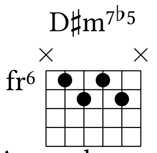
```
\chordbox{B\flat m}{x,x,x,3,2,1}   \chordbox{F\sharp^{maj7}}{2,x,3,3,2,x}
```





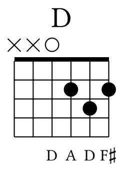
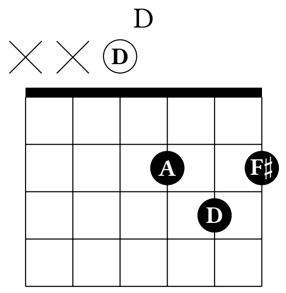
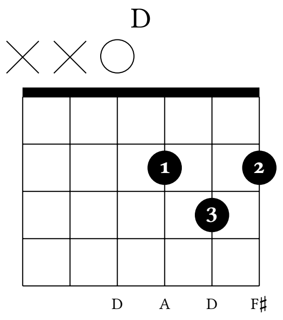
For best results, make sure to select the same typeface for both text and math typesetting. Of course, if you are also using the `leadsheets` package, you could opt to use its `\writechord` command for typesetting chord names, see also subsection 3.2.

The ⟨*base fret*⟩ is the number that, although formally an optional argument, must be provided for chords extending past the number of frets in the chord box. It is used to position the box and fretted notes relative to this fret, for example `\chordbox[6]{D\sharp m^{7\flat 5}}{x,6,7,6,7,x}`:



As can be seen, the nut is not drawn in these cases.

Finally, `chordbox` can also display the pitch of each note in a chord, although the default is not to. The pitches do not need to be specified manually, but are determined from the fret positions given to the command and the configured tuning (`/chordbox/tuning`). Just like fingering information, you may choose to display it inside the symbols or underneath the chord box, or combine both types of information:

| Command | `\chordbox{D}{x,x,0,2:1,3:3,2:2}` | | |
|---|---|---|---|
| Scale | default | 2.5 × larger | 2.5 × larger |
| `text on node` | `none` | `pitch` | `fingering` |
| `text below` | `pitch` | `none` | `pitch` |
| Result |  |  |  |

Chords may of course feature accidentals on some or all of their notes, depending on their associated musical key. Typesetting such pitches clearly in limited space is more challenging than just single characters, such as finger positions. With Ti*k*Z, text is most conveniently positioned using so-called *nodes*, which can also be scaled to display their contents at any size. This leads to the following two possible choices:

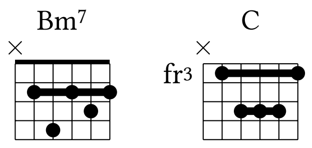
1. Scale the nodes to the chord box, so they are always as large as possible without overlapping;
2. Do not scale the nodes, but choose a fixed font size for the text, small enough to avoid overlap at the chosen scale.

The first option has the advantage that overlap is precluded entirely, but the node text is scaled as well and will therefore need to be changed to a size near that of the unscaled node. Additionally, the scaling operation is a purely graphical transformation, which yields worse results than selecting a similar font size in the first place. This is because a well-designed font has different (namely slightly thicker) glyphs for smaller sizes to maintain readability. The second option instead has the disadvantage that a font size must be chosen manually and calibrated to the size of the chord boxes one wishes to render. However, because documents will not usually contain chord boxes of many different sizes, I consider this to be a minor point and have, at least for now, chosen the second option. The font size initially configured specifically for pitch names below the strings is `\tiny`, which should produce readable results even for the default chord box size.

## 2.2 The `\bchordbox` command

Syntax: `\bchordbox[`⟨*base fret*⟩`]{`⟨*chord name*⟩`}{`⟨*fret positions*⟩`}{`⟨*barre frets*⟩`}`

As mentioned, this command is for drawing barre chords, for which it requires an additional comma-separated list of fret numbers. A thick line is drawn over the string symbols at these frets. The other arguments are identical to those of `\chordbox`, see for example `\bchordbox{Bm^7}{x,2,4,2,3,2}{2}` and `\bchordbox[3]{C}{x,3,5,5,5,3}{3,5}`:



## 3 Settings

Because `chordbox` relies on PGF/Ti*k*Z for drawing, it makes sense to use the powerful PGF key management system that comes with it. Therefore this package stores its code and style settings as keys, some of which may be modified to customise the output. This can be done in the preamble or in any part of the document by means of `\tikzset` and the more generic `\pgfkeys` command. Some examples of usage will be given below, but for more reading about PGF keys and their handling please refer to Section 82 of the PGF manual[7] where everything is documented in full detail. In any case, modifying a key stores its value for the LaTeX group where the command is issued and its child groups, which can override it again, but it does not propagate up to parent groups.

### 3.1 Ti*k*Z keys

The items described in this section are called *styles* in Ti*k*Z terminology and contain (lists of) keys and values that can be applied to various drawing commands. Most importantly, the `chordbox` style applies to the whole `{tikzpicture}` environment of every chord box produced. The default setting is to scale all coordinates by a quarter, because the drawing is done in PGF's "natural" units that, though convenient to use, result in an impractically large picture. Note that this scaling only influences the Ti*k*Z coordinates; in particular, fonts are not affected by default[8].

Because these keys are all stored in the `/tikz` path, it is most convenient to change them using the `\tikzset` command, for example `\tikzset{chordbox/.style={scale=0.4, thick}}`. Setting options like this through the `.style` *key handler*, as it is called, replaces any previous contents. Other handlers, like `.prefix style` and `.append style`, can be used to insert additional keys before or after the existing style, respectively. For example, calling `\tikzset{chordbox/.append style={scale=2}}` results in an effective `chordbox` style of `{baseline, scale=0.5}`, because the `scale` factor was set twice. These are special in that multiple values are multiplied together, whereas for most other keys only the last occurrence applies.

`/tikz/chordbox` (style, initially `{baseline, scale=0.25}`)

This style is applied to every `{tikzpicture}` environment produced by this package. The `baseline` key sets the base line of the picture at a height of 0 pt, which is the top of the grid and the bottom of the nut (if drawn). It ensures that multiple chord boxes in a row align properly.

[7] Tantau, *The TikZ and PGF Packages*.

[8] This is because text is put in nodes, which are scaled independently of the picture and require the `scale` and/or `transform shape` options to be supplied.

`/tikz/fret node text` (style, initially `{font=\Large\bfseries,text=white}`)
: This style is applied to text drawn on fretted or open string nodes, with the colour switched to black for the latter. It aims to make the text as clear and as big as possible without increasing the node size, which will occur if the text is too tall or wide.

`/tikz/pitch text below` (style, initially `{font=\tiny}`)
: This affects only pitch names typeset below the chord box. The initial setting is `\tiny` to keep neighbouring notes with accidentals from overlapping each other at the default scale.

The rest of the styles define the three symbol shapes used for the fretted (●), open (○) and muted (×) string positions.

`/tikz/string/base` (style, initially `{circle, draw, inner sep=0, minimum size=20, transform shape}`)
: Common options for all three symbols. The `transform shape` option is required to correctly scale the symbols, which are implemented as Ti*k*Z nodes, with the rest of the picture.

`/tikz/string/fretted` (style, initially `{string/base, fill}`)

`/tikz/string/open` (style, initially `{string/base, above}`)

`/tikz/string/muted` (style, initially `{string/open, cross out, minimum size=19}`)

## 3.2 Other keys

The following keys are not passed directly to Ti*k*Z commands, but used to store various other settings and code. They are all in the `/chordbox` path and should be set using the `\pgfkeys` command, for example `\pgfkeys{/chordbox/name/.code=\writechord{#1}}`. Keys that execute code, like this example, produce output with occurrences of `#1` replaced by the relevant value. If you want to change more than one of these settings at once, it is shorter to issue a command like the following: `\pgfkeys{/chordbox/.cd, numfrets=5, fingering text=on node}`.

`/chordbox/numfrets=`⟨*number*⟩ (initially 4)
: The number of frets drawn in each chord box.

`/chordbox/base fret` (initially `{fr\raisebox{.5ex}{\scriptsize#1}}`)
: This key stores the code used to typeset the base fret position of a chord box, if provided and greater than 1.

`/chordbox/name` (initially `\ensuremath{\mathrm{#1}}`)
: The code that is used to typeset the name of the chord.

The next two keys govern the placement of extra information in the chord box. In both cases, the `fingering` text is only displayed if actually present in the input.

`/chordbox/text below=none|fingering|pitch` (initially `fingering`)

Selects what to display below each string.

`/chordbox/text on node=none|fingering|pitch` (initially none)

Selects what to display inside the string symbols.

The remaining keys determine how pitch information is calculated and displayed. Pitches should be given enclosed in quotation marks and accidentals should be specified as `b` (flat) or `\#` (sharp); these will be replaced by their respective symbols.

`/chordbox/flat symbol=⟨symbol⟩` (initially `\flat`)

`/chordbox/sharp symbol=⟨symbol⟩` (initially `\sharp`)

`/chordbox/pitch names=⟨list of pitches⟩` (initially `{"A","A\#","B","C","C\#","D","D\#","E","F","F\#","G","G\#"}`)

The names of the pitches to be used for display. The list need not start at A, but it must contain twelve sequential semitones.

`/chordbox/tuning=⟨list of pitches⟩` (initially `{"E","A","D","G","B","E"}`)

The tuning of the instrument for which chord boxes are to be drawn. It is used to determine the pitches of the chord and must consist of elements present in `pitch names`.

## References

Clemens Niederberger. *guitarchordschemes*. Version 0.7. Aug. 16, 2016. URL: `https://ctan.org/pkg/guitarchordschemes/`.

Clemens Niederberger. *leadsheets*. Version 0.5b. Sept. 26, 2017. URL: `https://ctan.org/pkg/leadsheets/`.

Josselin Noirel. *xifthen*. Version 1.4.0. Nov. 5, 2015. URL: `https://ctan.org/pkg/xifthen/`.

Kasper Peeters. *gchords*. Version 1.20. Feb. 3, 2008. URL: `https://ctan.org/pkg/gchords/`.

Till Tantau. *The TikZ and PGF Packages. Manual for version 3.0.1a*. Aug. 29, 2015. URL: `http://sourceforge.net/projects/pgf/`.

Christian Tellechea. *xstring*. Version 1.7c. Oct. 13, 2013. URL: `https://ctan.org/pkg/xstring/`.