chronos

Version v0.9 (SVN Rev: 10885)

Clea F. Rees*

2025/02/28

Abstract

chronos is a LaTeX 2ε package, based on PGF/TikZ, designed to support the typesetting of diagrams illustrating timelines or chronologies. Externalisation is supported out-of-the-box with memoize. The package has developed from two sources: first, the creation of a timeline for use in teaching[1] and, second, questions on tex.stackexchange.com concerning obstacles encountered in using existing packages. This package might be considered an attempt to use the former to partially remedy the latter. It also means both the code and the user-interface contain strange and tangled regions where the wild errors may grow.

*Bug tracker: codeberg.org/cfr/chronos/issues | Code: codeberg.org/cfr/chronos | Mirror: github.com/cfr42/chronos

[1]See this answer on TeX StackExchange or view the PDF.

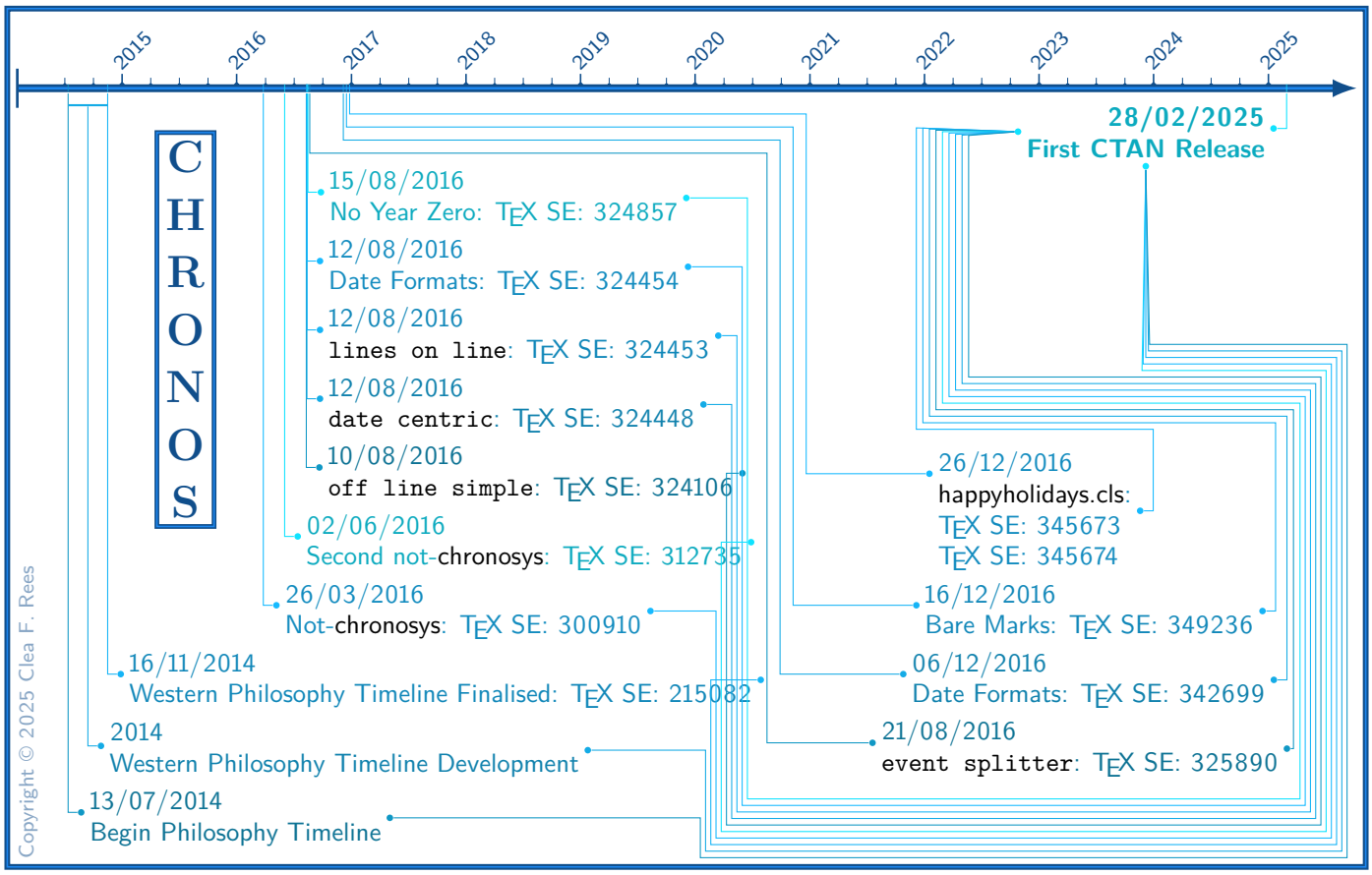


Figure 1: Chronos development: a chronos timeline (sections 6 and 8.4) with chronos style `blues below` (section 7.1.2) and custom styles `tag left`, `tag post` and `tag right` (section 13.3).

Contents

1 Raison d'être

Chronos aims to make it easy

- to specify timelines covering from days to centuries;
- to customise a timeline's appearance using the standard key-value syntax familiar to users of Ti*k*Z;
- to define new timeline styles in a straightforward manner;
- to utilise a range of timeline styles provided out-of-the-box, including some based on those offered by other packages and/or featured on tex.stackexchange.com.

2 Caveats, Assumptions & Limitations

First, the caveats ...

Chronos is *experimental*. Future releases will not make significant backwards-incompatible changes to documented features of the user interface without good reason. If such changes are made, a compatibility option will be offered, unless there is extremely good reason not to do so. *This applies* only *to documented features. It does applies to* neither *undocumented features* nor *the implementation details of those documented.*

Chronos makes some use of undocumented internal PGF/Ti*k*Z commands.

Chronos uses etoolbox to patch certain internal PGF/Ti*k*Z commands. While some of these changes, such as modifications to `rectangle`[2] are applied only locally, others, including changes to the `tikzpicture` initialisation code[3], are made globally.

Chronos has known incompatibilities with certain standard PGF/Ti*k*Z libraries (section 15).

Chronos has unknown incompatibilities with other standard and non-standard PGF/Ti*k*Z libraries and packages. These will be documented when discovered.

Chronos differs substantially from code previously published as chronos on TEX StackExchange. In particular, the user interface has changed: chronos now uses a key-value interface rather than multiple arguments when adding things to the timeline and the timeline itself is now created by the environment `chronos`[4]. See section 15.1 for guidance on converting existing timelines.

Caveat emptor ...

Second, (some of) the assumptions ...

Within the `chronos` environment, chronos assumes control over PGF/Ti*k*Z layers, disregarding any configuration setup by the user or other packages (section 6.4). This means you cannot use additional, custom layers in `chronos` environments unless you integrate them appropriately with chronos's changes. These changes are made locally and do not affect the use of whatever layers you please in a non-chronos environment, such as a regular `tikzpicture`.

Caveat emptor ...

Third, (some of) the limitations ...

The most serious limitation, given chronos's aims (section 1), is that you cannot define styles involving chronos keys using the standard PGF/Ti*k*Z interface, if you want to use them to configure individual additional elements (sections 6 and 9). Moreover, the alternative mechanism provided has serious shortcomings (section 13.3).

Chronos cannot produce timelines covering hundreds of thousands of years or which need to distinguish temporal units less than a day. It does days, months, years and centuries; it does not do (many) millennia, hours, minutes or seconds.

In particular, chronos is not designed to deal with dates outside the current Julian period. In theory, this means any date from 24th November, 4714 BCE should be permissible, but in fact, 24th November, 4713 BCE is the first date for which the package's behaviour should be relatively well-defined[5]. Matters are a little different when it comes to dates in the *next* Julian period. The cut off date for these is sometime in 3268 CE, according to Wikipedia, but pgfcalendar appears to be unaware of this. This means you may be able to get away with later dates, even though they are officially beyond the scope of this package[6].

[2] I am grateful to Symbol 1 for providing the code implementing this at TEX StackExchange: 385953.
[3] I am grateful to Martin Scharrer for for this at TEX StackExchange: 56405.
[4] Early versions on TEX SE actually used an environment, so this difference applies only to some chronos-based answers there.
[5] pgfcalendar says it uses the Wikipedia method, but appears to return dates 1 year later than some Wikipedia specifies e.g. day 0 gives a date in 4713 exactly a year after Wikipedia's one in 4714. But Wikipedia itself seems inconsistent, sometimes suggesting a date in 4713 and sometimes the previous year. For current purposes, the right answer doesn't matter: what matters is that pgfcalendar's answer is consistent. This means quibbles about the start date are unimportant (unless you're drawing a timeline starting with Winter Solstice 4714 BCE, of course. If you are, you might want to look into the matter.)
[6] That is, it may work, but it isn't a bug if it doesn't.

Chronos draws horizontal timelines. It does not support alternative orientations. In particular, vertical timelines are not currently supported.

Caveat emptor . . .

Finally, the code lacks both the virtues of sophistication and simplicity, while the user interface is characterised by confusion and complexity, the documentation is spotted with lacunae and unclarities, and the index is a conglomeration of misdirection and bull shit[7].

Caveat emptor . . .

3 Typesetting a Timeline

Further details concerning loading and invocation are explained in sections 4 and 5. The overall structure chronos provides is outlined in sections 6 and 6.4. Section 7 covers simple customisation using colour schemes and chronos styles. Detailed configuration of the timeline is explained in section 8. Section 9 covers the addition of elements such as lives, events, periods, theories, info boxes and titles to timelines. In this section, we begin by looking at a simple example.

After loading chronos in the document preamble:

```
% in document's preamble
\usepackage{chronos}
```

the `chronos` environment is available for typesetting timelines.

```
\begin{chronos}
  []
\end{chronos}
```

This takes an optional argument used to configure the timeline. This determines the size, appearance and duration of the timeline, as well as the use of headings, subheadings and frame. The body of the environment should consist of material to be added to the timeline itself, typically using chronos's commands for adding lives, events, periods, theories, theory circles, info boxes and/or main titles. It is also possible to include arbitrary Ti*k*Z code in the body of the environment, but commands need to be added to the appropriate chronos layer if they are to have their intended effects.

Suppose that we wish to typeset a timeline illustrating developments in the history of writing and printing. Having done exhaustive research utilising a single Wikipedia page, we decide our timeline should begin around 3,100BCE and end in the present. We're going to use the chronos style `cronoleg`, which puts year markers on the timeline itself. We decide we'd like large markers every 500 years and a smaller marker halfway between each pair of larger ones. We might, therefore, try

```
\begin{chronos}
  [
    cronoleg,% load chronos style
    timeline={% configure the timeline 'line' itself
      start date={-3100},
      end date=2100,
      minor step=250,
      major step=500,
    },
    levels=10:10,
  ]
```

This will result in 'major' markers (marks and years) at 3,000BCE, 2,500 BCE etc. and 'minor' at 2,750BCE, 2,250BCE and so on. Note that chronos starts the timeline at 3,100BCE, but assumes we'd like the first marker at 3,000BCE. `levels=10:10` will create a series of invisible nodes above and below the timeline named `level`

[7] In what sense 'bull shit'? Take your pick from any of several technical philosophical senses.

1,...,level 10 and level -1,...,level -10 respectively. The nodes are constructed so they take the same space as a 'standard' text tag of 'tag' type life created with \chronoslife. We can refer to these nodes when placing items to facilitate stacking, spacing and packing.

Based on our exhaustive seconds-long research, we now want to add some items of interest onto our timeline. We decide we'd like to note the lives of significant figures in the development of contemporary typography, most notably Donald Knuth, as well as a few luminaries from the modern era[8]. We'd also like to note certain specific events, such as key publication dates, and processes of longer duration.

```
\chronosevent{%
  name=\emph{jikji},
  date=1377
}
```

This will create an event in the default style in the default location, just off the timeline. Note that the text displayed in the event's node is '*Jikji*'. The coordinate jikji is placed at the point the element is added on the border of the timeline. The circular connector created at this point is the node chronos connector jikji. The circular connector on the event's text tag is the node main connector jikji. The text tag itself is the node tag jikji. As it stands, we may not be able to actually see all these elements if the event's text tag is placed right on the border of the timeline. If text tag yshift is non-zero, chronos will shift the node but, in general, it is necessary to tell chronos where to place the text tag. This doesn't affect the placement of the event on the timeline itself.

```
\chronosevent{%
  name=\emph{jikji},
  date=1377,
  yshift=20pt,
}
```

This will place the text tag node due north of the circular connector on the timeline with a straight line connecting the circular connector nodes main connector jikji and chronos connector jikji. However, we might also want to shift the text tag node horizontally and have the connection drawn to the west or east of the text tag.

```
\chronosevent{%
  name=\emph{jikji},
  date=1377,
  yshift=20pt,
  xshift=-5pt,
  anchor=east,
}
```

will shift the text tag 5pt to the left and draw the connection up and left from the timeline to main connector jikji which is now drawn east rather than the default south.

We decide to place a second event, for which we have a precise date. This time, we use as is to tell chronos not to attempt to capitalise the text. This is necessary because we have an \emph{⟨*word*⟩ ⟨*word*⟩} and chronos's capitalisation command can't cope with this. This also means we need to add appropriate capitalisation ourselves.

```
\chronosevent{%
  date={868-05-11},
  name={Publication of \emph{Diamond Sutra}},
  yshift=-40pt,
  xshift=20pt,
  anchor=west,
  as is,
  connectors={east,south},
}
```

[8] In my discipline, 'modern' means roughly the sixteenth to nineteenth centuries.

Note that this event is placed below the timeline.

We decide to add some notable figures next. For this, we create elements of tag type life, beginning with the inventor of movable type, Bi Sheng.

```
\chronoslife{%
  name=bi sheng,
  birth=972,
  death=1051,
  at=tag jikji.north -| bi sheng,
  connectors={east,north},
}
```

Note the use of at to place the text tag detailing the name and dates. Since this node is placed above the timeline, its anchor is south by default. at=tag jikji.north -| bi sheng aligns this anchor directly above the relevant point on the timeline (bi sheng) and just on top of tag jikji. If you want to fit many items onto your timeline, fitting them closely together is useful but you could, of course, lift the box higher if you want a bit more space.

Leaping ahead, we now want to add Donald Knuth.

```
\chronoslife{%
  name=donald knuth,
  birth={1938-01-10},
  text tag yshift=40pt,
  connectors={west,north},
}
```

Note the omission of death for a living person. Chronos assigns today's date internally for placement purposes, but will not typeset it when constructing the text tag[9] This works reasonably, but the connection from the timeline crosses the text node for the publication of the *Diamond Sutra* because chronos has placed this item below the timeline, even though there is plenty of space above. This is because, by default, chronos alternates between placement above and below the line. In this case, we decide to override the default choice.

```
\chronoslife{%
  name=donald knuth,
  birth={1938-01-10},
  text tag yshift=40pt,
  connectors={west,north},
  place above,
}
```

Note that the cronoleg rotates the colours used for elements belonging to tag types life, event and period, but not theory, but colour lists are rather subdued for events and periods. For each type of elements, one set of colours is used below and another above the timeline. These colours can be accessed later as colour ⟨*name*⟩[10].

Colour rotation can be switched on or off for particular kinds of elements, overridden for individual elements and configured by altering the colour lists chronos cycles through. These colours are tracked by copying them to new names for each element created and may be accessed using these names later. This means you can draw something in the colour assigned to Donald Knuth, say, without knowing which colour that is. If you add an element to the timeline or change the colour lists later, the drawing will use the appropriate colour. For example,

```
  \node (pi) [colour donald knuth, font=\Huge, right=5pt of tag donald knuth.base east, anchor=base
west] {$\pi$};
```

will add a large π in the colour (automatically or otherwise) assigned to Knuth.

[9] chronos is not the most optimistic of packages.

[10] In most cases, you can also access items using American spelling. So color would work here. So would lliw.

```
\draw [colour donald knuth] (tag donald knuth.north) ++(0pt,20pt) circle (10pt);
```

would draw a circle above Donald Knuth's text tag in the colour automatically assigned to Donald Knuth.

We next decide to indicate the period when woodblock printing was used to produce books. This is a *circa* date, so we can't use chronos's automatic production of the date information, though we still need to specify dates for placement on the timeline. We'd still like chronos to format the name of the text tag, though, so we use `dates content` to override the automatic production of date labels.

```
\chronosperiod{%
  name=woodblock printing,
  start=600,
  end=1450,
  yshift=-20pt,
  xshift=10pt,
  anchor=west,
  dates content={c600--1450\ceyearlabel},
  place below,
}
```

If we wanted to override the formatting of the name rather than the dates, we could use

```
  name=woodblock printing,
  name content={WoOdBlOcK pRiNtInG},
```

If we wanted something completely different in place of the name and date information, we could instead use

```
  text content={something entirely different\\--- not even about woodblocks!},
```

BCE dates require special consideration. In general, a minus indicates BCE, but chronos needs to be able to distinguish this from the hyphen between years and months or months and days in standard date specifications (section 8.2). This means either providing a full date of the form `-YYYY-MM-DD`, for example, or ensuring chronos expects only a partial date such as a year.

```
\chronosperiod{%
  name=proto-Elamite use of cylinder seals,
  start={{-3100}-01-01},
  end={{-2700}-12-31},
  dates content={c3000\,\bceyearlabel},
  yshift=20pt,
  connectors=north,
  connectors=east,
}
```

Here, we protect the BCE year with curly brackets, specify a default month and day. If we specified only a year, chronos would assign a month and day; if we assigned only a year and month, chronos would assign a day. (The outer set of curly brackets is standard and cannot be omitted for full date specifications, regardless of era.)

We've now added examples of each of the three basic types chronos supports connecting to our timeline. However, the package also offers some complementary elements. These are not connected to the timeline, though theories are designed to be connected to the types which are.

```
\chronostheory {%
  name=TeX,
  text content=\TeX,
  at=donald knuth-text.north west,
  xshift=-10pt,
  anchor=south east,
  connectors={east},
}
```

We also want to indicate Knuth's connection with TEX, so we join the connector we made when creating the text tag for Knuth to the connector we've just created for TEX. Chronos supports the addition of such connectors on most text tags created with its commands and the drawing of connections between connectors.

```
    \draw [chronos connect=life:donald knuth] (connector donald knuth) -- ++(-5pt,0pt) |- (connector
TeX);
```

This makes it possible to connect multiple people to the same theory, for example, as well as connecting a single person to multiple theories. In a more complete chronology, several different font designers or book publishers, for example, might be connected with a particular approach to typography. Elements which support connectors out-of-the-box are those belonging to tags of types life, event, period and theory.

When `cronoleg` is used, connectors are small circular nodes on the timeline's border and the borders of text tags i.e. the nodes containing information about the chronos elements presented in the chronology illustrated.

In contrast, theory circles, info (information boxes), copyleft or copyright notices and main titles are freestanding objects without ready-made connectors.

Headings and subheadings are designed to label stretches of time and are placed in relation to the timeline, though no connecting lines are drawn.

When we've finished adding material to the timeline, of course, we need to complete it.

```
\end{chronos}
```

4 Loading the Package

Chronos requires a LaTeX 2$_\varepsilon$ format no older than 2021–11–15. To load the package simply add the following to your document's preamble.

```
\usepackage{chronos}
```

Chronos will load the following packages and libraries automatically:

Packages:

- calc
- chronos-lib-colschemes (part of chronos)
- chronos-lib-styles (part of chronos)
- etoolbox
- expl3 (if required)
- fp
- pgfcalendar
- svn-prov
- tikz
- xcolor
- xparse (for LaTeX 2$_\varepsilon$ formats prior to 2020–10–01)

PGF/Ti*k*Z libraries:

- arrows.meta
- backgrounds
- calc
- decorations.text
- fit
- fixedpointarithmetic
- positioning
- shadows

`simple colour names` = `true|false`
`no simple colour names`
`simple color names`
`no simple color names`
boolean key

The only two options currently supported are `simple colour names` or `simple color names` and its complement `no simple colour names` or `no simple color names`. The following are equivalent:

```
\usepackage{chronos}
\usepackage[simple colour names]{chronos}
\usepackage[simple colour names=true]{chronos}
\usepackage[no simple colour names=false]{chronos}
```

In these cases, chronos will create an additional colour for each additional element of tag-type life, event, period, theory or info named ⟨*name*⟩, where ⟨*name*⟩ is the value given to `name` when creating the element.

Since chronos creates these `colours` globally, this is potentially problematic. To disable it use any of the following

```
\usepackage[no simple colour names]{chronos}
\usepackage[no simple colour names=true]{chronos}
\usepackage[simple colour names=false]{chronos}
```

If you want to disable such names later, perhaps for specific timelines, see section 8.8.

5 Invocation

chronos [⟨*chronos preamble*⟩]
environment

The ⟨*chronos preamble*⟩ is a ⟨*key-value list*⟩ setting any non-default options which should be applied to the timeline and any other macro-level elements of the picture to be constructed. At a minimum, most users will want to specify start and end dates, but the majority will likely want to customise the timeline further. (If you do not much care about customisation, there are simpler packages to typeset timelines!)

Some options can be given only *in or before* the ⟨*timeline specification*⟩ in the optional ⟨*chronos preamble*⟩. Others will have no effect or unwanted effects at this point and must be specified later.

The environment `chronos` ***is a wrapper for a*** `tikzpicture`***. It can neither include, nor be included in, another*** `tikzpicture`***. Additional drawing commands must, therefore, be included in*** `chronos` ***itself.***

6 Chronos Anatomy

Figure 2 provides an overview of the configuration and anatomy of a chronos timeline.

As explained in section 5, the timeline itself is constructed by the `chronos` environment, as determined by the ⟨*chronos preamble*⟩, any prior use of `\chronosset` and fallback defaults.

In addition to configuring the timeline itself, the ⟨*chronos preamble*⟩ and any prior use of `\chronosset` determine the use and configuration of any frame, headings and subheadings, as well as the default configuration of any additional elements.

The body of the `chronos` environment is the ⟨*timeline additions specification*⟩. The ⟨*timeline additions specification*⟩ specifies what should be added to the `tikzpicture` besides the timeline itself and any frame, headings or subheadings. It will typically consist of a series of chronos commands specifying the items to be connected to the timeline and any non-connected elements (section 9). However, it may include any code valid in a `tikzpicture` environment or be entirely empty.

Section 6.1 provides a breakdown of the various elements of which the timeline is composed. Section 6.2 provides an overview of the additional elements which may be added in the ⟨*timeline additions specification*⟩.

If your timeline uses non-chronos commands, you will need to read sections 6.4 and 10, which explains the layers chronos uses. If your commands are not having their usual effects, you should first check whether they are simply hidden by another layer.

6.1 Chronos Timeline

The timeline itself is a horizontal line consisting of some or all of the following elements

- Timeline line refers to the main line, which is drawn or filled by default depending on height and configuration. The `height`, `width` and `timeline border height` are responsible for the total size of the timeline.
- Borders are (potentially) filled with a gradient above and below the main line. By default, borders are added when marks are placed on the timeline itself, which necessitates a taller timeline.
- Era labels are (potentially) placed at each end of the line, depending on the time period covered.
- Timeline years, minor years, marks, minor marks and bare marks may be placed above, below or on the main timeline line.

Some elements must be specified in the ⟨*chronos preamble*⟩, but are constructed only at the end of the `chronos` environment. These include optional headings and subheadings to be placed at the top of the `chronos` environment and an optional frame.

Headings and subheadings are constructed after and above most other elements `on chronos foreground layer`. As explained in section 8.7, headings and subheadings may be used to roughly indicate named stretches of time such as 'Tudors' or 'Bronze Age'.

- Headings are placed in a single row at the top.
- Subheadings are placed just below the headings in two rows:
  - The upper subheadings are placed in a single row just beneath the headings.
  - The lower subheadings are placed in a single row just beneath the upper subheadings.

The frame is constructed even later, but drawn behind most other elements `on chronos background layer`.

6.2 Chronos Additional Element Types

Aside from the timeline itself, its headings and subheadings and frame, chronos provides six primary types of element which may be added to the timeline: life, event, period, theory, info and theory circle. In this documentation, these are referred to as 'tags' or 'tag types'. Three further tags encompass one-off elements:

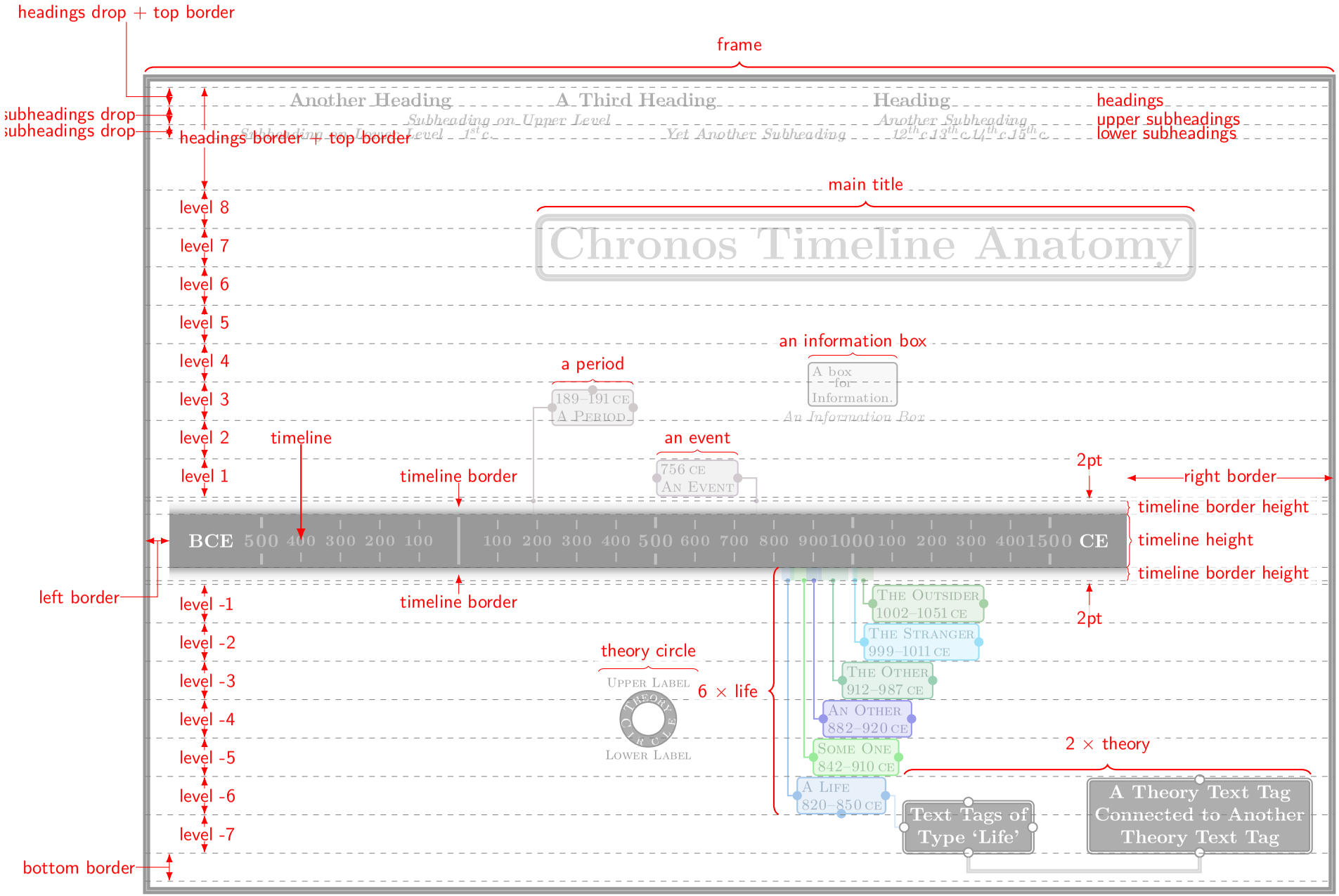


Figure 2: chronos anatomy

main covers the main title and frame, while copyleft and copyright account for any declaration of copyleft or copyright.

For example, all elements created using `\chronoslife` are said to belong to tag type life.

6.2.1 Primary Types

6.2.1.1 Timeline-Connectable Elements Elements belonging to the first three tags (life, event , period) are (potentially) connected to the timeline and are placed according to date of occurrence.

- These elements are assigned colours and colour names are created so they may easily be reused. These colours may (and, by default, are) used to create connections, connectors, lines and text tags.
- These elements are connected to the timeline by default using connections which join chronos connectors to text tag connectors on the elements' text tags.
- Dates/periods are (potentially) drawn or filled on, above or below the timeline using lines.
- Text tags are created for the elements[11]. By default, these typically include a name and date or date-range, though arbitrary content is permissible. The location of text tags is configurable, though it usually makes sense to place them in relation to their chronos connectors.
- Life and period use two dates for placement. A line is (potentially) drawn and/or filled on, above or below the timeline, by default in the element's associated colour.
- Event uses a single date for placement. A line is (potentially) drawn on the timeline, by default in the element's associated colour.

Timeline-connectable elements are also connectable (note 6.2.1.2).

6.2.1.2 Connectable Elements Elements belonging to the first four tags (life, event, period , theory) are (potentially) connectable to each other.

- These elements (potentially) feature connectors which may be used to connect elements together. When the first three are connected to the timeline, one such connector is created by default[12].
- Elements belonging to the theory tag are connectable, but not timeline-connectable. Unlike timeline-connectable elements (note 6.2.1.1), they cannot be connected to the timeline and may be freely placed; unlike non-connectable elements (note 6.2.1.3), they may be connected to each other and/or timeline-connectable elements.

6.2.1.3 Non-Connectable Elements

Elements belonging to the remaining tags (info, theory circle, main, copyleft and copyright) are non-connectable and, with the exception of frame may be located according to user preference.

- Like connectable-but-not-timeline-connectable elements, non-connectable elements are not connected to the timeline and may involve no date information at all, but unlike theories they do not feature connectors so may not easily be connected to other elements.
- Info and theory circle elements are standalone items for providing content. The former (potentially) have captions below; the latter (potentially) have labels above and/or below. The first are basically just text nodes with arbitrary content; the second can display two small chunks of text arranged in semicircles with a hole in the middle for a letter or symbol.
- Theory circles are *slow* and their use should be limited to avoid excessive compilation times. They are also arguably the most difficult to read and should be used only for items of minor or secondary importance.

[11] I am grateful to Symbol 1 for enabling connectors to be centred correctly on the borders of text tags at TEX StackExchange: 385953.

[12] Connectors may be customised to 'disappear', but even invisible connectors can be used in connections.

- The standalone elements are best created last and are most useful for filling in 'holes' in a timeline which would otherwise look unbalanced. If chiropody didn't develop much in the twelfth century or not much is known about the finer points of tortoise-raising in the second, these elements may be used to plug the unsightly gaps left by inconvenient histories.

6.2.2 Secondary (Sub-)Elements

Orthogonal to the primary elements explained above, chronos uses the following (sub-)elements:

- Connectors are small elements drawn on the boundaries of text tags and the timeline which can be used as connection points. By default, they are small and circular, but they may be rendered invisibly or otherwise according to preference.
- Connections are drawn between connectors. The package draws a connection between the timeline and date-placed elements by default, but occasionally you may prefer to specify this connection manually. Other connections can be added to link elements.
- Text tags hold information associated with all elements except theory circles.
- Lines are marked on the timeline to indicate the date and/or duration of dated elements.

6.3 Chronos Coordinate and Node Names

Figure 3 shows key coordinate and node names. Those available by default can be shown on any timeline using the option `debug`. Examples of different tags have been added with labels to illustrate how chronos names their coordinates and nodes. Detailed documentation is provided in sections 8 and 9.

6.4 Chronos Layers

In addition to loading the backgrounds library, which defines the layer `background`, and the default layer `main`, chronos defines another four layers, for a total of six: `chronos background` and `chronos middle ground`, which are layered between `background` and `main`, and `chronos foreground` and `chronos overlay`, which are layered above `main`. From top to bottom:

```
chronos overlay
chronos foreground
main
chronos middle ground
chronos background
background
```

Section 10 explains how to draw directly on different layers. You may wish to do this if you are using non-chronos code in the ⟨*timeline additions specification*⟩ or the facilities explained in section 12 for deferring code.

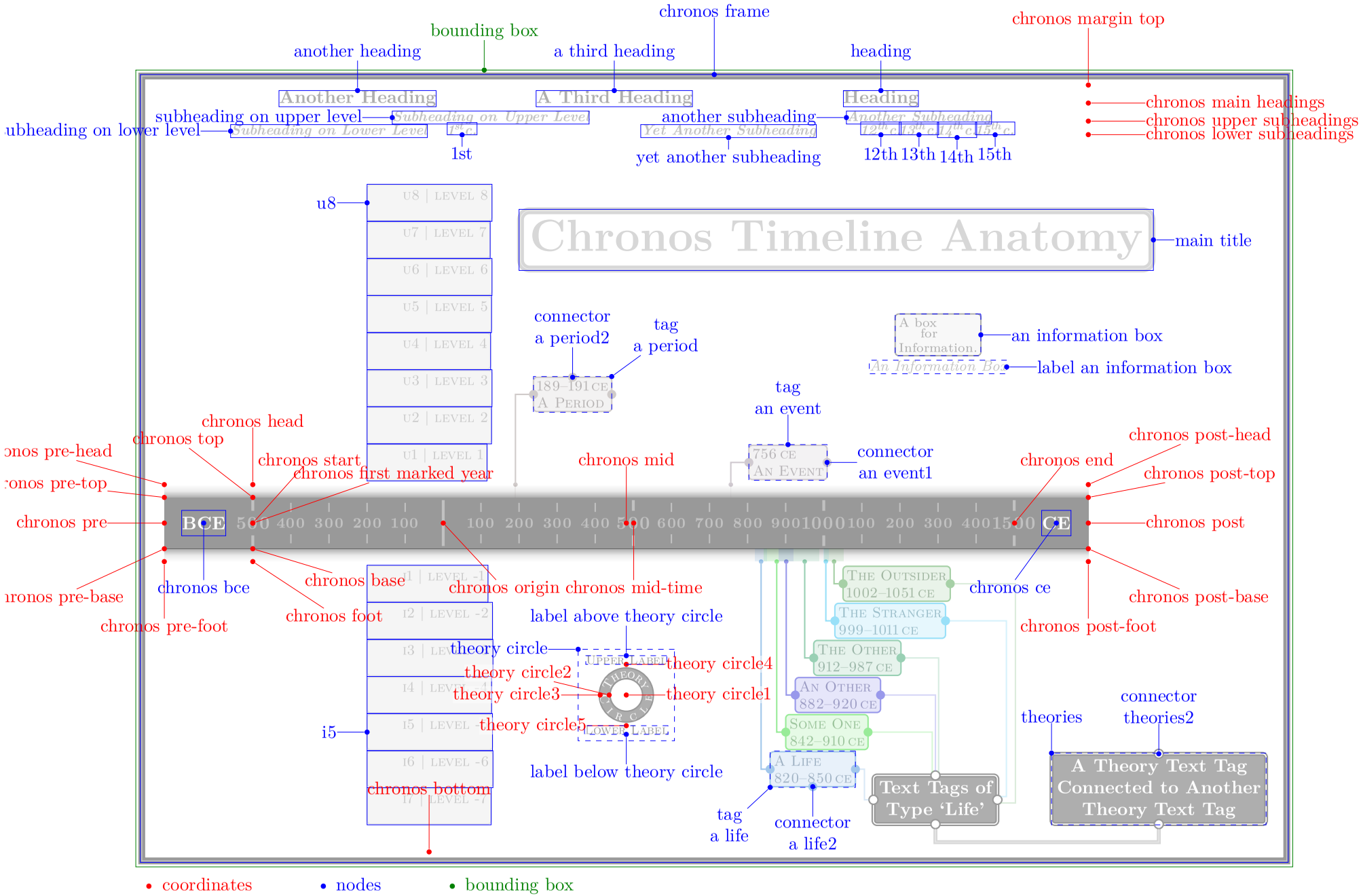


Figure 3: chronos anatomy: key coordinate and node names

7 Chronos Schemes and Styles

Two simple methods for applying, defining and reusing chronos styles are provided: chronos styles and colour schemes. If using both, load the chronos style first, since it may already load a colour schemes.

7.1 Chronos Styles

By far the easiest way to customise a timeline is simply to load a chronos style in the ⟨*chronos preamble*⟩. This section illustrates a basic timeline typeset with each of chronos's standard styles.

Note that you will typically need to set `start date` *and* `end date` *and perhaps adjust how often* years *and* marks *appear on your* timeline. *Chronos styles such as key[chronosstyle]event splitter set highly idiosyncratic dates by default, simply by way of example.* chronos *will not warn you if you don't override options set by a chronos style.*

In selecting a chronos style, bear in mind that some things are easy to change, while others are harder. At a minimum, you should pick an 'on line' chronos style if you want `timeline years on line` and an 'off line' one if you want them `above` or `below`. `event years on line` requires an 'on line' chronos style; `event dates split` is designed for an 'off line' one.

You should also think about how much information you need to display. `date centric` won't work for a densely packed timeline, so if you have a lot of things to pack in, don't choose this unless you're drawing an extremely long timeline. Likewise, `cronoleg` will look rather silly if you only want to represent the lives of Socrates and Plato.

7.1.1 'On Line' Styles

All 'on line' styles are designed to support adding elements both above and below the timeline. This includes the default settings. See table 1 and fig. 4.

`cronoleg` *chronos style*
The most developed and best tested, if somewhat idiosyncratic, chronos style, based on the code used to construct my Western Philosophy Timeline. It constructs a 235mm timeline and uses a colour scheme highlighting elements of type life, but the colours may be adjusted or the same colour scheme applied to event and period as well. By default, it is designed to produce a picture occupying an entire A4 page and has a wide right-hand margin for additional elements, in addition to ten levels above and below the timeline. See table 1 and fig. 5. By default, this chronos style does *not* use the bounding box for the frame.

`date centric` *chronos style*
A chronos style with a monochrome appearance and sans-serif fonts of 150mm[13]. Intended for timelines highlighting relatively few dates. See table 1 and fig. 6. This style demonstrates the use of `event years on line` and `special date`.

`lavender menace` *chronos style*
A variant of `modern` with a muted colour scheme and sans-serif fonts. By default, it produces a timeline covering the modern era (1500–1900 CE). See table 1 and fig. 7a.

`modern` *chronos style*
A chronos style with a monochrome appearance and sans-serif fonts. By default, it produces a timeline covering the modern era (1500–1900 CE). See table 1 and fig. 7b.

`rainbow serif` *chronos style*
A colourful variant of `serif on line` utilising xcolor colour series and serif fonts. See table 1 and fig. 8a.

`serif on line` *chronos style*
A chronos style with a monochrome appearance and serif fonts. See table 1 and fig. 8b.

`sober judge` *chronos style*
A somewhat subdued chronos style with a monochrome appearance, sans-serif fonts and boxed text tags. See table 1 and fig. 9.

[13] Based on my answer at TEX StackExchange: 324448.

Table 1: Summary of chronos styles.

Name	Timeline Year Style	Defaults				
		Levels	Dates	Colour Scheme	Rotation	Arrow
-	on line	0:0	1800–2050 CE	default	✓	–
cronoleg	on line	10:10	500 BCE–2050 CE	cronoleg	✓	–
date centric	[on line]	-	1935–2010 CE	default	–	–
lavender menace	on line	3:3	1500–1900 CE	lavender+chronosSilver	✓	–
modern	on line	3:3	1500–1900 CE	modern	–	–
rainbow serif	on line	3:3	1500–2100 CE	xcolseries	✓	–
serif on line	on line	3:3	1800–1900 CE	default	–	–
sober judge	on line	3:3	1/10/1001–14/6/1003 CE	default	–	–
blues below	off line, below	0:3	1550–2050 CE	blues	✓	✓
flipping blues	off line, above	3:0	1550–2050 CE	blues	✓	✓
contemporary 90	off line, above	0:3	2002-2016 CE	contninety	–	✓
off line colour	off line, below	-	3000–2000 BCE	offlinebasic	✓	✓
off line colour alt	off line, below	-	3000–2000 BCE	offlinealt	✓	✓
off line simple	off line, below	-	3000–2000 BCE	offlinebasic	–	✓
rotated 45	off line, above	-	25 BCE–20 CE	default	–	–
simple arrow	off line, above	-	1–2000 CE	default	–	✓
somewhat plain	off line, above	0:3	500 BCE–2050 CE	default	–	–
event splitter	[above]	-	01/13–02/22/2014 CE	default	–	–
lines on line	none	-	1–2016 CE	default	✓	✓
plain arrow	none	-	1–2016 CE	default	✓	✓

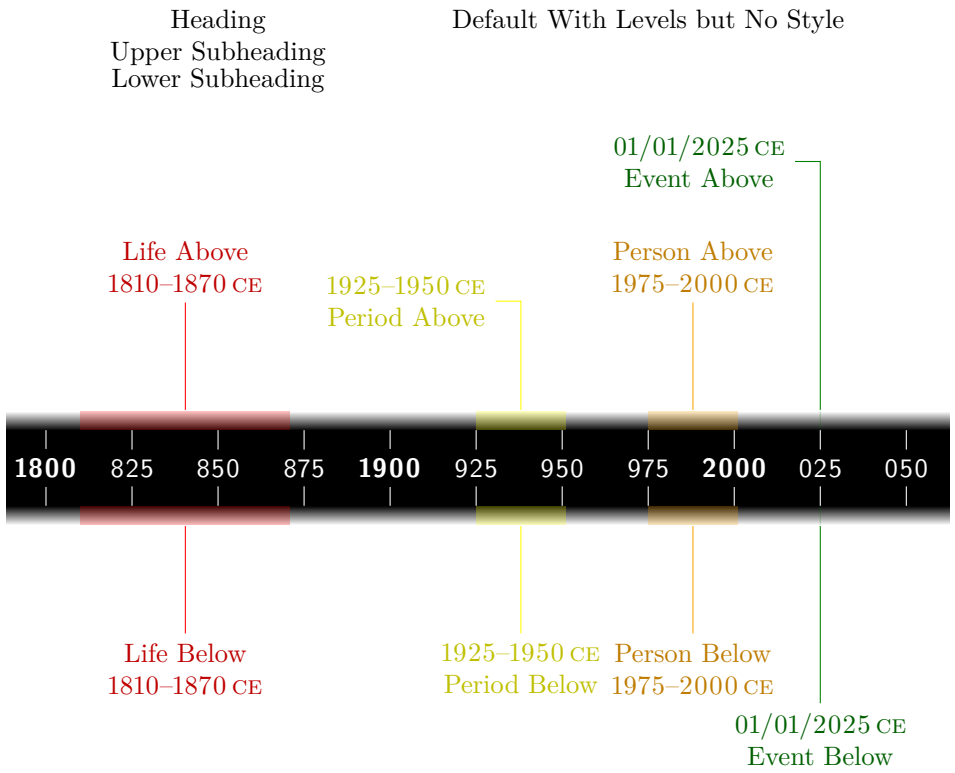


Figure 4: Chronos style: none.

7.1.2 'Off Line' Styles

`blues below`
chronos style

A chronos style featuring the `blues` colour scheme, off-set lines and year labels rotated through 45°. Intended for timelines which add elements below. See table 1 and fig. 10a. This style demonstrates how to rotate year labels.

`contemporary 90`
chronos style

A chronos style with a monochrome appearance, sans-serif fonts and rotated year labels, which produces a relatively short timeline of 90mm by default. Intended for timelines which add elements below. See table 1 and fig. 11.

`flipping blues`
chronos style

A variation of `blues below` featuring year labels rotated through -45°. Intended for timelines which add elements above. See table 1 and fig. 10b. This style demonstrates how to utilise an existing chronos style to produce a variant.

`off line colour` = ⟨*length*⟩
chronos style

A straightforward style utilising scientific dates in which the line tapers to form an arrow. Intended for timelines which add elements above and/or below. The optional ⟨*length*⟩ specifies the length of the tapering.

Default: `20mm`

See table 1 and fig. 12a. This style demonstrates the use of `chronos middle ground layer` to reduce visual clutter where connections cross timeline marks. Although the connections are drawn after the timeline, they are placed on a lower layer, with a partially transparent rectangle in between.

`off line colour alt` = ⟨*length*⟩
chronos style

A variant of `off line colour` which uses a different colour scheme.

Default: `20mm`

Cronoleg

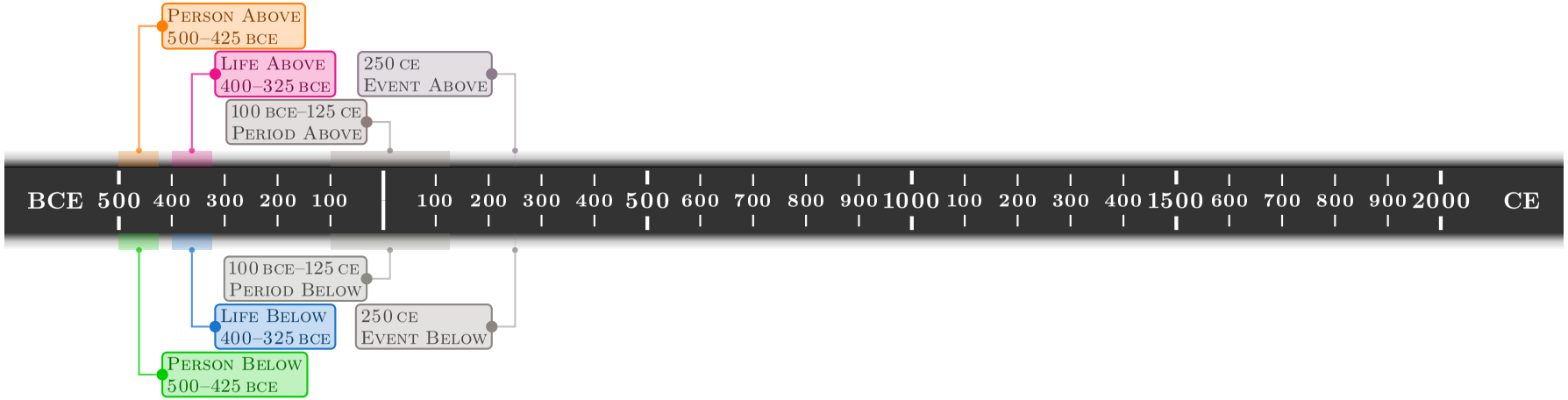


Figure 5: Chronos style: `cronoleg`.

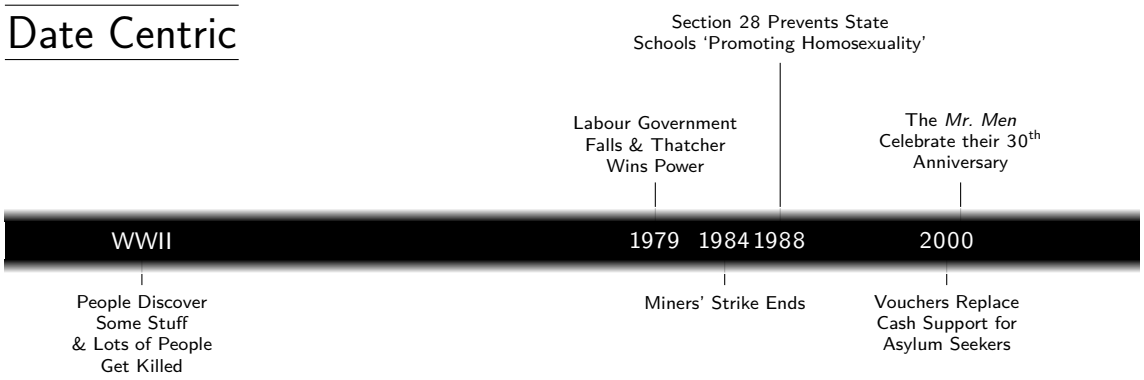


Figure 6: Chronos style: `date centric`.

See table 1 and fig. 12b.

`off line simple` = ⟨*length*⟩
chronos style

A less colourful variant of `off line colour` utilising only two colours[14].

Default: `20mm`

See table 1 and fig. 12c.

`rotated 45` *chronos style*
A chronos style featuring the off-set lines and text tags rotated through 45°. Intended for timelines which add elements below. See table 1 and fig. 13. This style demonstrates how to rotate text tags.

`simple arrow` = ⟨*length*⟩
chronos style

A monochrome appearance with a plain 200mm arrow timeline and years and marks above[15]. ⟨*length*⟩ determines the length of the taper comprising the arrow.

Default: `10mm`

Intended for timelines which add elements below. See table 1 and fig. 14.

`somewhat plain` *chronos style*
A chronos style with a monochrome appearance and sans-serif fonts which produces a relatively short timeline of 100mm by default. Intended for timelines which add elements below. See table 1 and fig. 15. This style demonstrates how to create a style to draw lines above and below the main title node, without drawing the left and right sides of the node.

7.1.3 'No Year' Styles

`event splitter` *chronos style*
A 150mm timeline with no year labels which demonstrates the use of `event dates split`[16]. *Intended for* timelines *with* connected elements *solely of* tag *type* event. See table 1 and fig. 16.

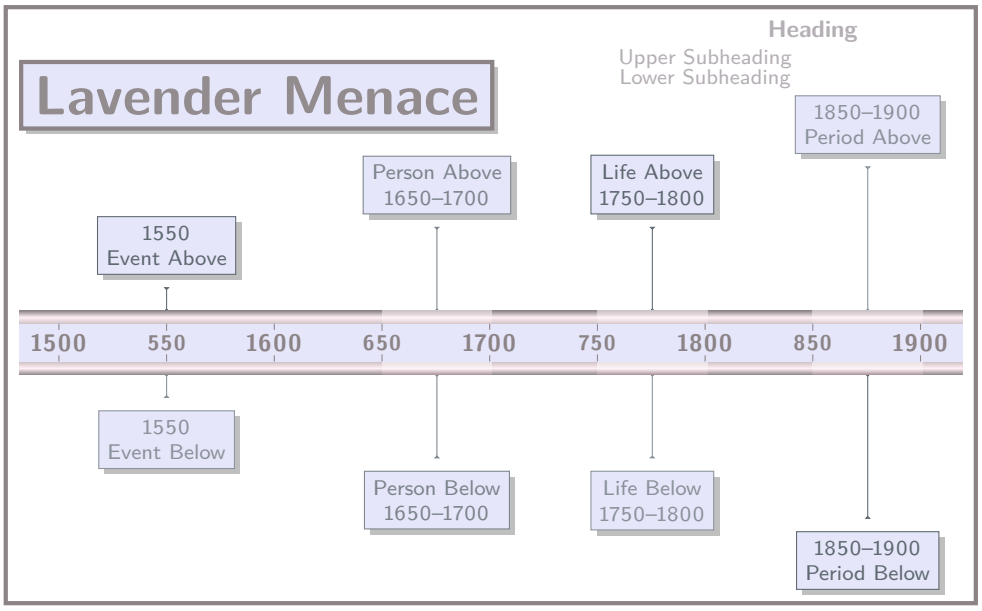
`lines on line` = ⟨*dimension*⟩
chronos style

A 120mm timeline arrow, ⟨*dimension*⟩ high, with no year labels and life, event and period lines drawn on the timeline itself[17]. Date information is confined to text tags. Out-of-the-box, this chronos style adds elements of tag type event above and those of type life and period below.

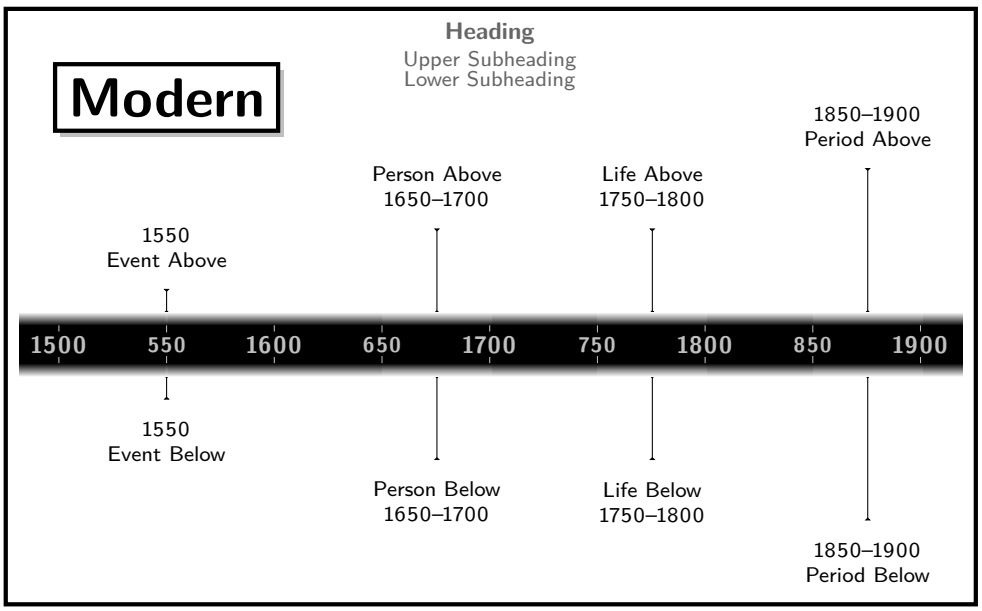
Default: `5mm`

See table 1 and fig. 17.

[14] In fact, this version is closest to the original. See my answer at TEX StackExchange: 324106.
[15] Based on my answer at TEX StackExchange: 342699.
[16] Based on my answer at TEX StackExchange: 325890.
[17] Based on my answer at TEX StackExchange: 324453.

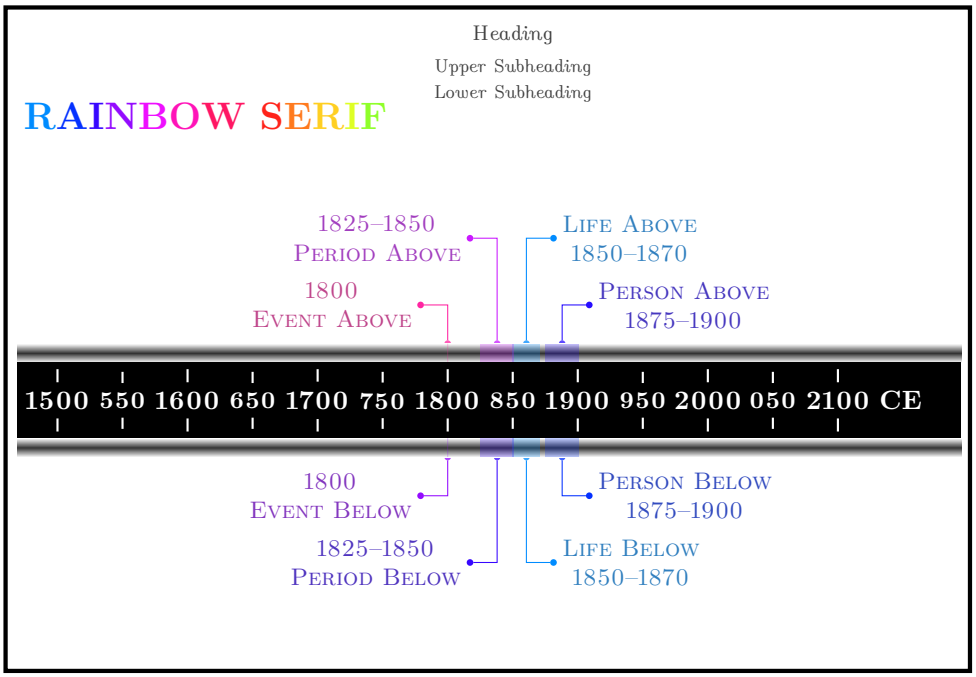


(a) Chronos style: `lavender menace`

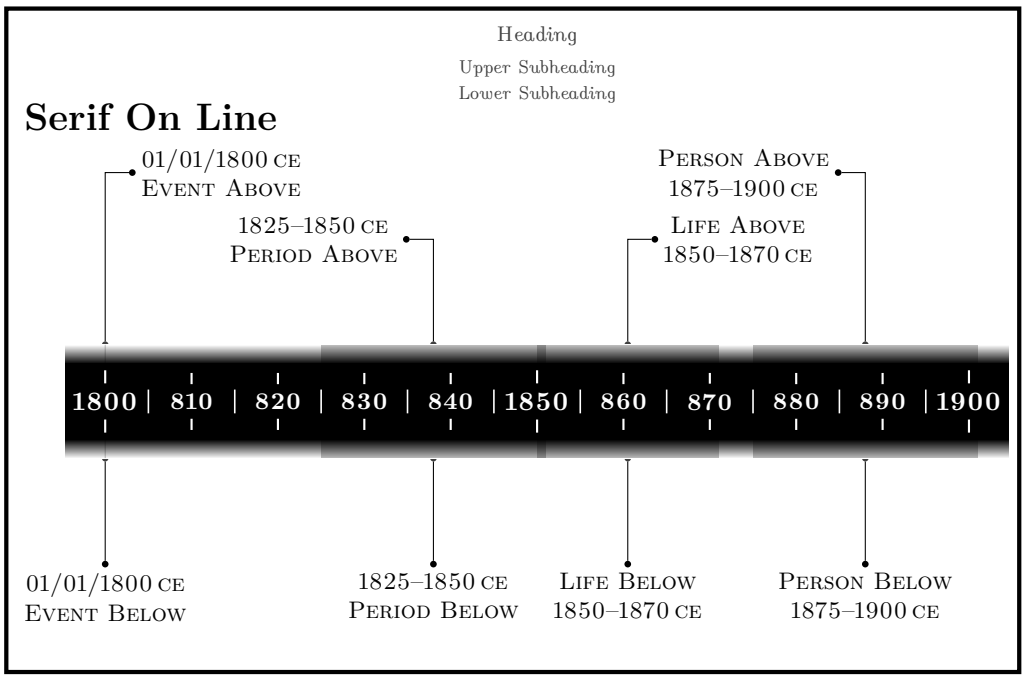


(b) Chronos style: `modern`

Figure 7: Figure 7a is a variant of fig. 7b.



(a) Chronos style: `rainbow serif`.



(b) Chronos style: `serif on line`.

Figure 8: Figure 8a is a variant of fig. 8b.

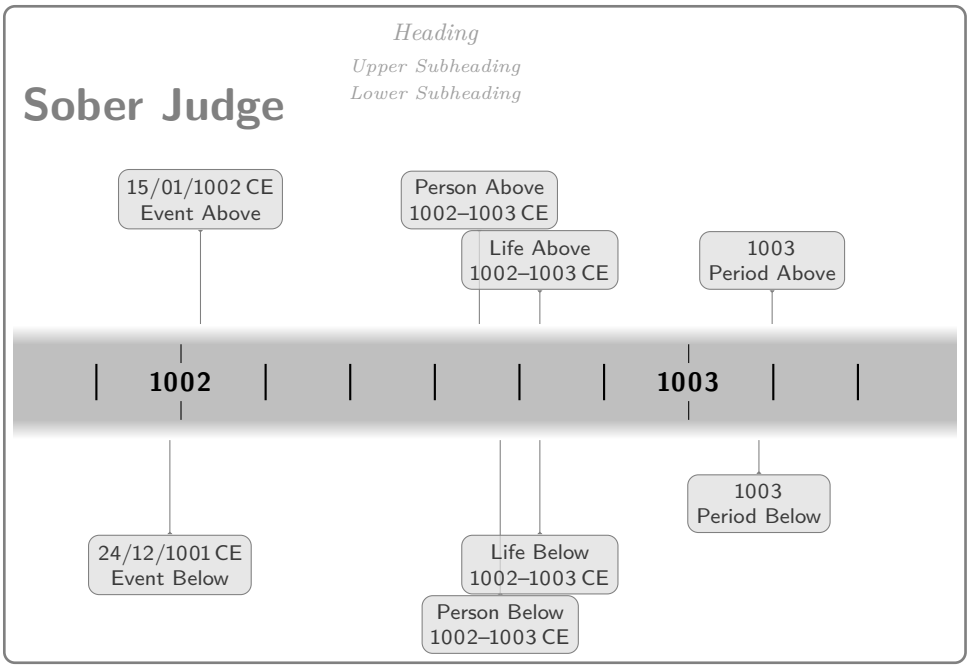


Figure 9: Chronos style: `sober judge`.

`plain arrow` = ⟨*dimension*⟩
chronos style

A variant of `lines on line` (fig. 17) which draws a 120mm timeline arrow with no year labels and life, event and period lines drawn on the timeline itself[18]. Date information is confined to text tags.

Default: `5mm`

Intended for timelines which add elements of tag type event above and those of type life and period below. See table 1 and fig. 17b.

7.2 Chronos Colour Schemes

As explained in section 8.8, chronos utilises a somewhat complex system for colour customisation. In many cases, however, you will not need to delve into the mechanisms used. Instead, you can simply load an existing colour scheme. If none of the provided schemes meet your needs, see section 13.1.

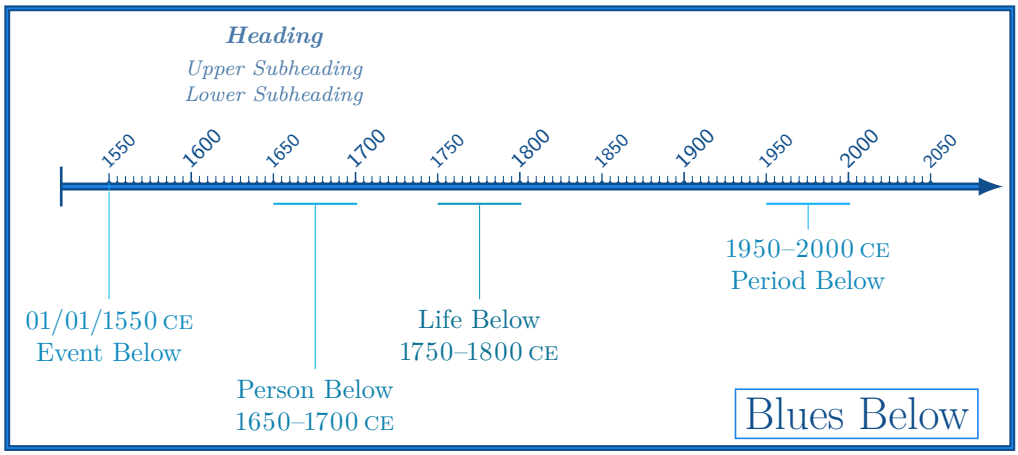
To load a colour schemes, you just write

```
\begin{chronos}
  [
    modern,
    colour scheme=blues,
  ]
\end{chronos}
```

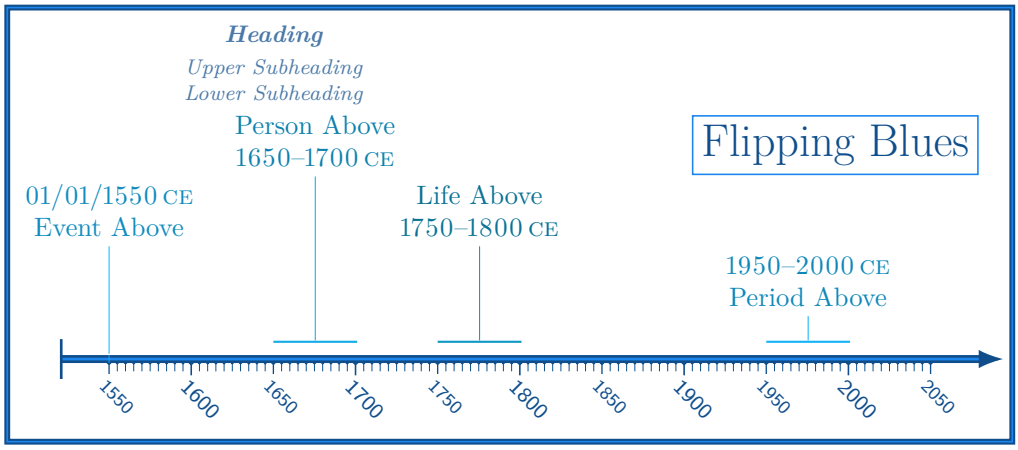
which would load the chronos style `modern` followed by the colour schemes `blues`. Since chronos styles may legitimately load colour schemes, but colour schemes may not load chronos styles, always load any chronos style *before* any colour scheme. Then make any further modifications you wish.

```
\begin{chronos}
  [
```

[18] Based on my answer at TEX StackExchange: 324453.



(a) Chronos style: `blues below`.



(b) Chronos style: `flipping blues`.

Figure 10: Figure 10b is a variant of fig. 10a.

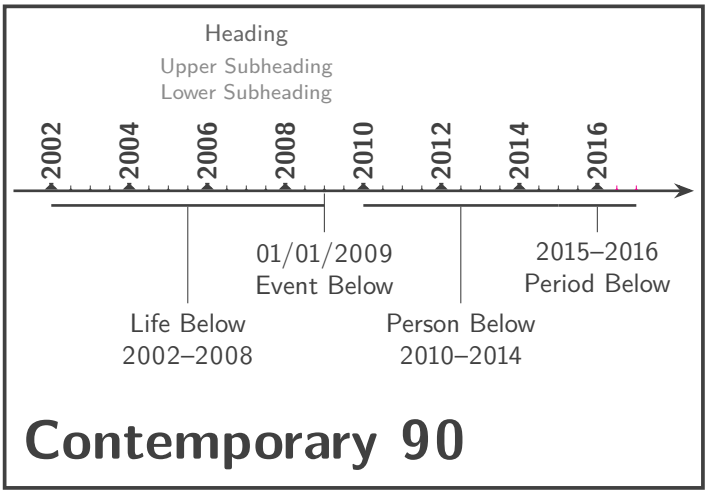
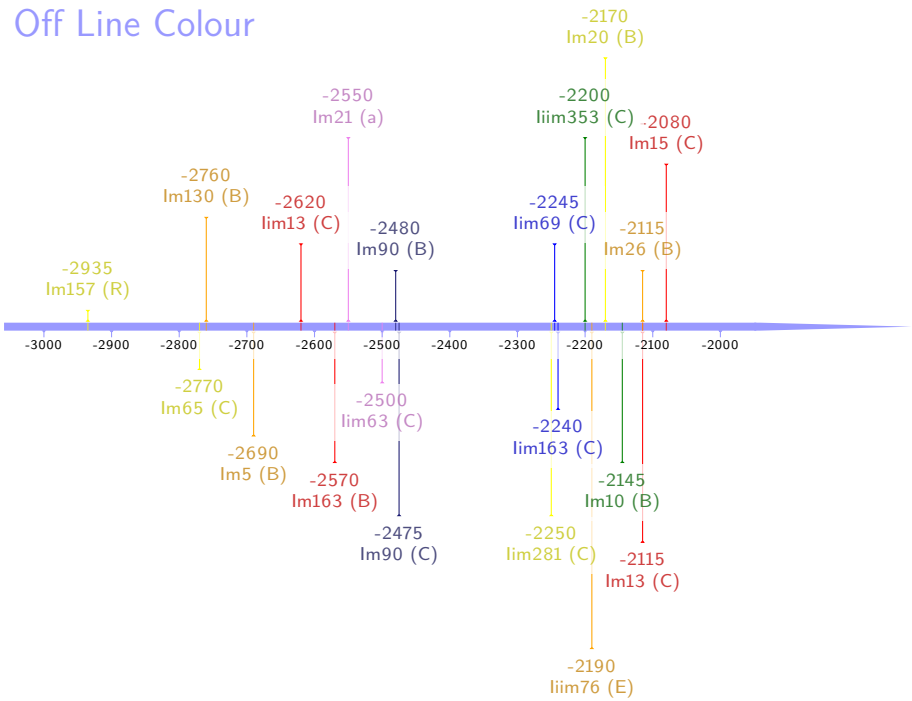
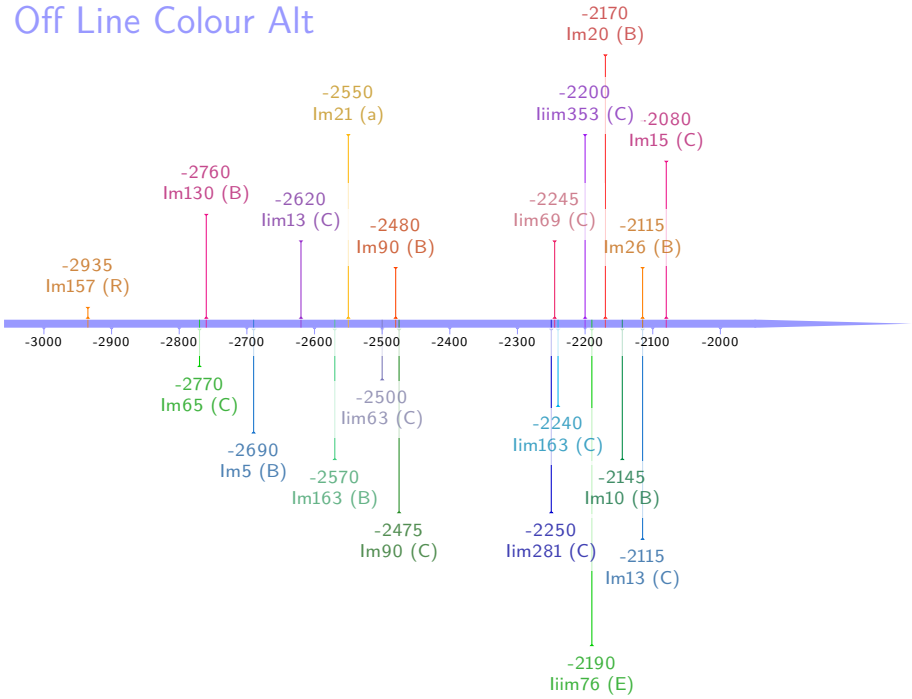


Figure 11: Chronos style: `contemporary 90`.

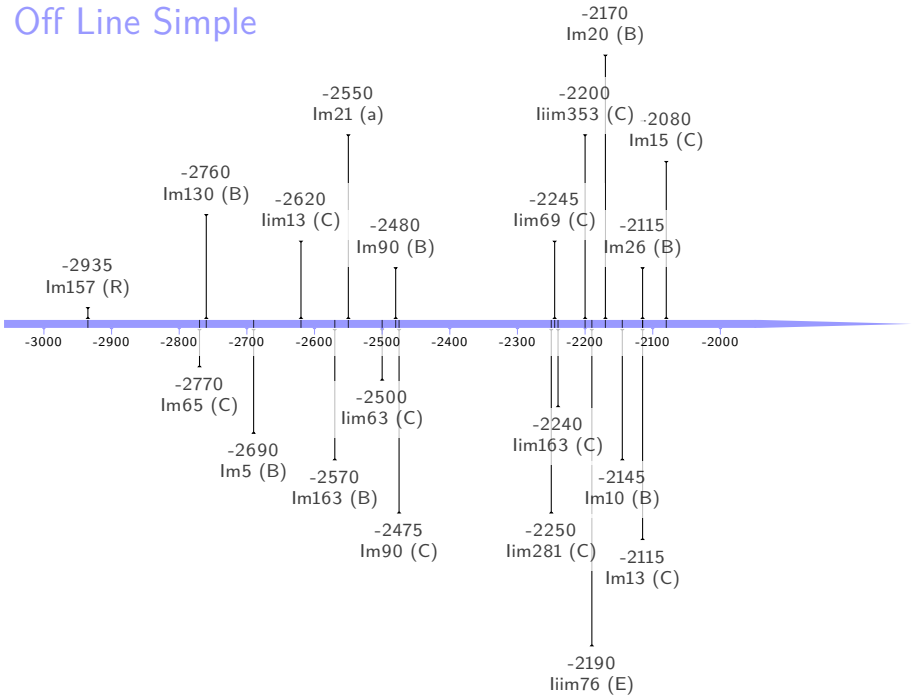


(a) Chronos style: `off line colour`.



(b) Chronos style: `off line colour alt`.

Figure 12: Figures 12b and 12c are variants of fig. 12a.



(c) Chronos style: `off line simple`.

Continued Figure 12: Figures 12a and 12c are variants of fig. 12b.

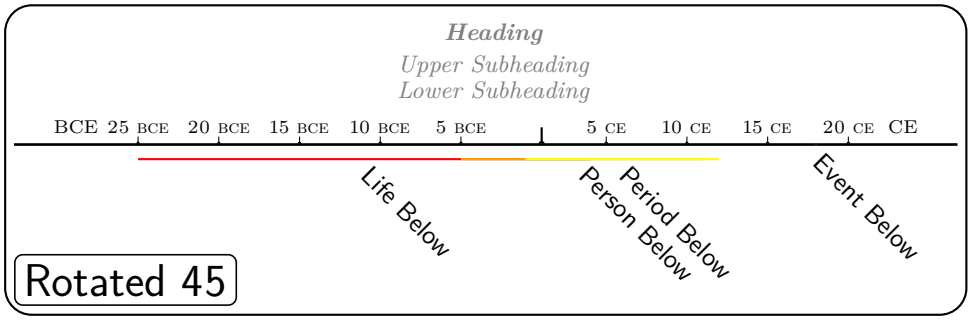


Figure 13: Chronos style: `rotated 45`.

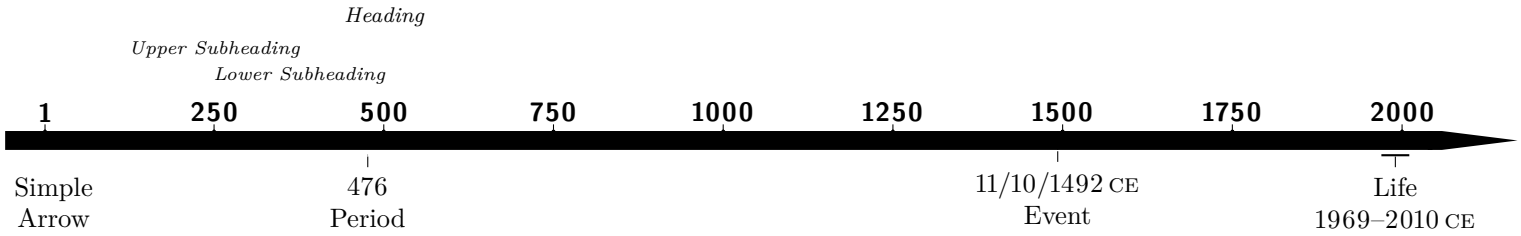


Figure 14: Chronos style: `simple arrow`.

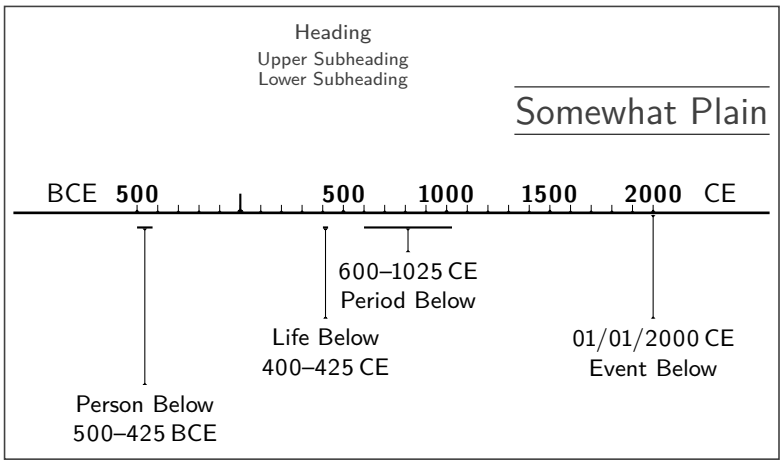


Figure 15: Chronos style: `somewhat plain`.

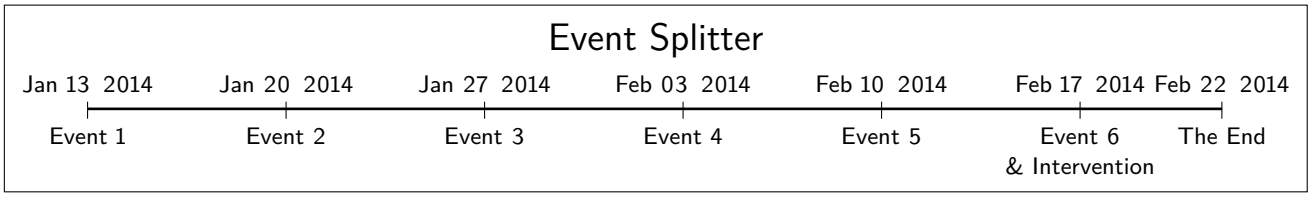
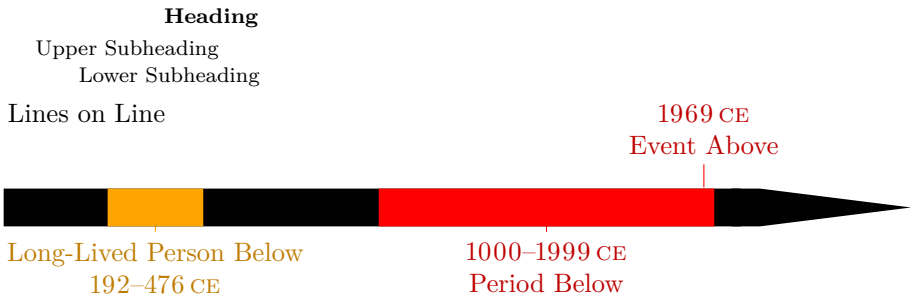
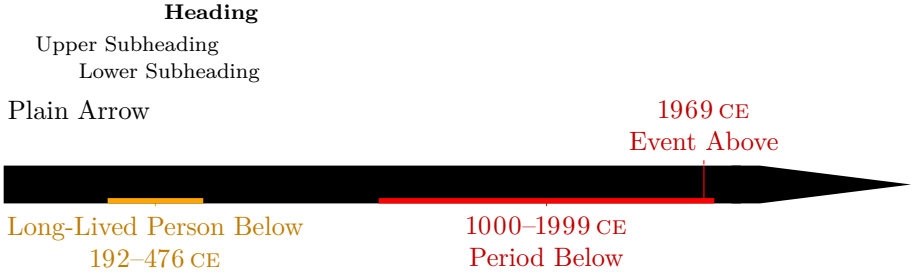


Figure 16: Chronos style: `event splitter`.



(a) Chronos style: `lines on line`.



(b) Chronos style: `plain arrow`.

Figure 17: Figure 17b is a variant of fig. 17a.

Table 2: Chronos Colour schemes.

Colour scheme	Variant Of	Default For	Examples
- (default)	-	rotated 45, serif on line and somewhat plain	figs. 4, 8b, 13 and 15
blues	-	blues below and flipping blues	figs. 1, 10a and 10b
contninety	-	contemporary 90	fig. 11
cronoleg	-	cronoleg	fig. 5
lavender	-	lavender menace	fig. 7a
modern	-	modern	fig. 7b
offlinebasic	-	off line colour and off line simple	figs. 12a and 12c
offlinealt	cronoleg	off line colour alt	fig. 12b
sobriety	-	sober judge	fig. 9
xcolseries	-	rainbow serif	fig. 8a

```
    modern,
    colour scheme=blues,
    timeline={%
      dates=1066:1946,
    },
    event/default colour=ForestGreen,
    every text tags+={draw=##1},
  ]
\end{chronos}
```

`colour scheme` = {⟨*name*⟩}
`color scheme`
key

⟨*name*⟩ should be the name of a colour scheme. A small number of colour schemes are provided by chronos (section 7.2); others may be defined using the method explained in section 13.1.

Default: the default set of colours.

Example: `colour scheme=cronoleg`

chronos styles may load colour schemes and typically should if they wish to make significant changes.

In addition to the default colours, chronos currently provides `blues`, `contninety`, `cronoleg`, `lavender`, `modern`, `offlinebasic`, `offlinealt`, `sobriety` and `xcolseries` (table 2). New colour schemes may be created using the interface explained in section 13.1.

8 Configuration

Chronos was designed to be highly configurable. However, by far the *easiest* way to customise a timeline is to load a chronos style. See section 7.1.

Most configuration uses the standard key/value interface provided by Ti*k*Z. In addition, a `\chronosset` is provided for configuring defaults.

Most chronos options have local scope. That is, changes do not survive the current group.

However, a small number of options are set *globally*. In these cases, chronos keeps track of a list of defaults, as well as the current options, and restores the defaults at the beginning and end of each `chronos` environment. By default, `\chronosset` changes the default values of globalised options, whereas the ⟨*chronos preamble*⟩ does not.

Globalised options saved as default are stored in expl3 variables named with a package-specific prefix. A similar prefix is used for globalised colours.

`\chronosset` {⟨*key-value list*⟩}
macro

`\chronosset*` {⟨*key-value list*⟩}
macro

This should be used to configure chronos *outside* the `chronos` environment. It should *not* be used within that environment. The starred version does *not* make any global changes. In general, there is no reason to use the starred version as altering these variables non-globally will have no effect and other variables are not set globally in any case. It is provided 'just in case', even though I can't think of a use-case for it.

Chronos sets the following options globally. At the end of the preamble, the active values are saved. These are then restored at the end of each `chronos` environment. This means the results of typesetting a timeline should not depend on earlier timelines in the same document, a phenomenon which may otherwise result in changes of position and colour, for example. Options set globally:

- the list of `century subheadings` (but neither other subheadings nor headings are globalised);
- most colours and lists of colours;
- whether the last `text tag` of a particular kind (event or period) was placed above or below the timeline.

All other settings should behave as usual for PGF/Ti*k*Z as they are not handled specially and all other LaTeX 3 variables are declared locally.

This approach is intended to ensure that things behave as I expect you to expect, but it is obviously not unlikely you may expect something I don't expect you to expect. For this reason, it is strongly recommended that document-wide settings be configured in the preamble of your document. `\chronosset` should be used in the document body *only* when you wish to change the document defaults partway through your document. If at all possible, I recommend the use of styles, configured in the preamble, instead, but there will be cases where such an approach may be sub-optimal. `\chronosset` may be used later in such cases.

In particular, you are urged to configure default colours and colour lists, in your preamble. See sections 8.3, 8.8 and 9.5. If you get unexpected colours, please remember that chronos defines most colours *globally*. They are *not* limited to the current `chronos` environment. That is, chronos lets you customise the colours in many different ways, including many you might wish it did not.

8.1 Documentation Notes

The following notes apply throughout this document.

8.1.1 Font Conventions

This document uses the following typographic conventions.

Bold/***Bold Italics*** are used to emphasise important points, especially ones which might be overlooked.

Italics are used with < and > for {⟨*mandatory arguments*⟩}, [⟨*optional arguments*⟩] and ⟨*parameterised values*⟩. When used in the text without delimiters, they are used for emphasis in accordance with standard typographic conventions for English language texts.

`Monowidth Typewriter` is used for `\macros` (e.g. `\commands`), `environments`. `key names` and `code`.

Sans Serif is used for concepts, elements, package names and class names.

The distinction between a 'concept', an 'element' and a 'key' is not always obvious. Where discussion meanders through the borderlands of fuzzy concepts[19], the font in which a word appears

[19] A 'fuzzy concept' is one whose extension cannot be precisely defined without arbitrariness. For example, there are clear cases where 'bald' applies and equally clear cases where it does not, but there is no non-arbitrary point at which non-baldness becomes baldness. 'Bald' is clear in the middle and clear well beyond its scope, but decidedly fuzzy at its edges.

is sometimes arbitrary and the choice should not be taken too seriously. Moreover, some words, such as 'timeline', are used for all three.

8.1.2 Keys and Values

Chronos provides a user interface for customisation based almost exclusively on pgfkeys.

8.1.2.1 Keys In case you have somehow come across this package shortly after landing in contemporary TEXland, the basic idea is that the package provides a set of keys which you use selectively to customise the output. Some of these keys are simple keywords.

Example: `no connections,`

8.1.2.2 Values When keys permit or require arguments, the arguments are called `values`. A given key will generally require a ⟨*value*⟩ of some particular sort, as explained for each key below.

Some chronos keys permit an argument, but don't require it.

Example: `frame,`

Example: `frame=true,`

Example: `frame=false,`

The above are all valid (with the first two being equivalent).

Other chronos keys require one or more arguments.

Example: `colour=Cerulean,`

Example: `heading={chronos year -150}{chronos year 250}{past},`

Chronos frequently requires multiple arguments to be separated by colons, because this often seemed less error-prone than multiplying curly brackets in complex cases.

Example: `dates={{-100}-01-12}:{900-12-24},`

In some instances, where a proliferation of colons seemed no less an invitation to error than one of curly brackets, the colon cases are convenience keys, which you can avoid through the use of two or more alternate keys to specify items separately.

8.1.2.3 Key-Value Lists ⟨*key-value list*⟩s are comma-separated lists of items, each of which is either a simple ⟨*key-name*⟩ or a ⟨*key-name*⟩`= {`⟨*comma-separated list of values*⟩`}`. In general, the ⟨*comma-separated list of values*⟩ will be a Ti*k*Z ⟨*key-value list*⟩, though it may sometimes be appropriate to include further chronos keys.

Example: `event/line={draw=blue,draw opacity=.75}`

8.1.3 Key Specifications

Key specifications in this document look like this:

`key name` = ⟨*argument specification*⟩ *tag1, tag2, tag3, ...*
key type

⟨*Description of key and explanation of usage.*⟩

Default: ⟨*key's default value*⟩

Initially: ⟨*key's initial value*⟩

Example: ⟨*example of usage*⟩

⟨*Commentary.*⟩

Table 3: chronos key types.

Key type	Description	Example
boolean key	Controls a boolean or toggle i.e. a conditional.	
choice key	Selects from a list of possible options.	
comma-separated list key	Processes or stores a comma-separated list of things.	
colour key	Specifies a colour.	
colour list key	Special kind of comma-separated list key which stores a list of colours.	
date key	Specifies a date or dates.	
date format key	Specifies one or more date output formats.	
dimension key	Specifies a TEX dimension.	
key	Some other kind of key.	
style	A PGF/Ti*k*Z style.	

Here, `key name` is the name of the key, *key type* is the type of key, ⟨*argument specification*⟩ specifies the number, kind and format of the value or values the key expects and *tag1, tag2, tag3, ...* indicates to elements of which tag or tags the key applies. See table 3 for an explanation of the types of key chronos uses. See sections 6 and 6.2 for information about tags.

If no initial value is specified, the default value is also the initial value. Where both an initial and a default value are specified, the default is the value used if the ⟨*key name*⟩ is given without an argument and the initial value is the value used if ⟨*key name*⟩ is not used at all. This terminology follows the usage in pgfkeys and is especially prevalent in the handling of boolean keys, where it is common for the initial value to be `false`, but the default value to be `true`.

Schematically,

```
\begin{chronos}% initial value used
  [
    % other keys
  ]
\end{chronos}
\begin{chronos}% default value used
  [
    % other keys
    key name,
  ]
\end{chronos}
\begin{chronos}% new value used
  [
    % other keys
    key name=new value,
  ]
\end{chronos}
```

8.1.4 Syntax Notes

See section 8.1.5 for the syntax of dimension keys, where plus and prime have different meanings.

8.1.4.1 Slash (/) Where a forward slash (/) occurs in a key, it indicates a context-specific key. For those familiar with PGF keys, this corresponds to a path under `/chronos`.

Example: `life/connection`

indicates a key affecting connection(s) belonging to elements of type life.

8.1.4.2 Plus (+) A plus sign (+) at the end of a key indicates that the key *adds* to any pre-existing list. This form is generally available when the base key replaces, rather than adding

to, any pre-existing list.

```
timeline line={draw=black,fill=green},
timeline line+={opacity=.8},
```

is equivalent to

```
timeline line={draw=black,fill=green,opacity=.8},
```

A plus at the end of a dimension key indicates that the dimension key *adds* the value given to the current value of the dimension.

8.1.4.3 Prime (') A prime (') at the end of a key indicates that the key *replaces* any pre-existing list. This form is generally available when the base key adds to, rather than replacing, any pre-existing list.

```
century subheadings={15,17,19}{th},
century subheadings'={13,14}{th},
century subheading={21}{st},
```

is equivalent to

```
century subheadings'={13,14}{th},
century subheading={21}{st},
```

and will result in subheadings being created for the 13th, 14th and 21st centuries (assuming the timeline covers these time periods and the relevant coordinates exist).

A prime at the end of a dimension key, or at the end except for a plus ('+), indicates that the dimension key expects a TEX dimension, as opposed to an expression to be evaluated by pgfmath.

8.1.5 Dimension Notes

8.1.5.1 Dimensions Each key described as a dimension keys is available in six forms[20]:

⟨*dimension key*⟩ = {⟨*pgfmath-parsable dimension*⟩}
dimension key

The dimension key parses the ⟨*specified value*⟩ using pgfmath and assigns the result in points as the dimension. This base form, which is typically the only form explicitly listed in this documentation, is slow but flexible. Unless otherwise noted, the existence of the base form implies the availability of all six variants.

⟨*dimension key*⟩' = {⟨*dimension*⟩}
dimension key

The dimension key expects a TEX ⟨*dimension*⟩, complete with units, which it assigns directly. This is faster but less flexible.

⟨*dimension key*⟩+ = {⟨*pgfmath-parsable dimension*⟩}
dimension key

The dimension key parses the expression (⟨*specified value*⟩ + ⟨*existing value*⟩) with pgfmath and assigns the result in points. This is slower but more flexible.

⟨*dimension key*⟩'+ = {⟨*dimension*⟩}
dimension key

The dimension key expects a TEX ⟨*dimension*⟩, complete with units, which it adds to the ⟨*existing dimension value*⟩ directly. This is faster but less flexible.

⟨*dimension key*⟩- = {⟨*pgfmath-parsable dimension*⟩}
dimension key

[20] Occasionally, a convenience key may only support the prime, prime-plus and prime-minus forms. Where this applies, the limitation is noted in the description.

The dimension key parses the expression (⟨*specified value*⟩ − ⟨*existing value*⟩) with pgfmath and assigns the result in points. This is slower but more flexible.

⟨*dimension key*⟩`'-` = {⟨*dimension*⟩}
dimension key

The dimension key expects a TEX ⟨*dimension*⟩, complete with units, which it subtracts from the ⟨*existing dimension value*⟩ directly. This is faster but less flexible.

When dimension keys end in prime, prime-plus or prime-minus, ⟨*dimension*⟩s must be given as TEX dimensions complete with units and may not require calculation.

Example: `timeline height'=10mm`

Example: `timeline border height'+=20pt`

Example: `timeline width'-=2em`

When dimension keys do not include prime, any value which can be parsed by pgfmath is valid.

Example: `timeline height=.01\textheight`

Example: `timeline border height+=1.5\headrulewidth`

Example: `timeline width-=0.05\linewidth+1.5pt`

8.1.6 Date Specification Notes

8.1.6.1 Date Format Specifications A ⟨*date format specification*⟩ (⟨*date format spec.*⟩) is an expression using the syntax explained in section 8.2.2.

Example: `date format={!d !B !Y !E}`

8.1.6.2 Dates ⟨*date*⟩s must be specified using the syntax explained in section 8.2.1.

Example: `dates={{-200}-04-05}:{200-12-31}`

8.1.7 Colour Notes

8.1.7.1 Colours ⟨*colour*⟩s should be colour names or mixtures supported by xcolor.

Example: `colour=WildStrawberry`

Example: `foreground=WildStrawberry!50!black`

8.1.7.2 Colour Lists ⟨*colour list*⟩s are comma-separated lists of colour names or mixtures supported by xcolor.

Example: `life/colours above={blue,green,blue!50!green}`

8.1.7.3 Colour `colour` and `color` are synonyms in key names.

Example: `colours below={black,gray}`

Example: `colors below={black,gray}`

8.2 Dates

Chronos uses a fixed format for date input and offers a flexible format for date output.

8.2.1 Input

All date keys expect one or two arguments specifying a date or dates in the format `{{Y}-M-D}`. `Y`, `M` and `D` must be integers. If `Y` is negative, the date is interpreted as BCE; otherwise CE is assumed. The additional curly brackets around `Y` are *mandatory* for negative values.

Table 4: Date and year format specification codes.

code	meaning	example output	date format specifier?	year format specifier?
`!a`	short weekday name	Mon	✓	–
`!A`	full weekday name	Monday	✓	–
`!b`	short month name	Jan	✓	–
`!B`	full month name	January	✓	–
`!c`	semi-shortened year	900	✓	✓
`!d`	day of the month	23	✓	–
`!E`	era	BCE or CE label	✓	✓
`!m`	month number	01	✓	–
`!q`	minus if year is BCE	-	✓	✓
`!Q`	minus if year is BCE; plus for CE	+	✓	✓
`!y`	last two digits of year	66	✓	✓
`!Y`	year	1066	✓	✓

```
start date={{-3000}-05-23},
end date={1500-12-04},
```

It is also permissible to specify only a year, in which case chronos will specify values for the month and day. Hence,

```
dates={-245}:789,
```

is also valid. Where two dates are required, `dates` offers a more concise syntax, but dates may always be specified singly if this is preferred.

8.2.2 Output

All date format keys expect one or three arguments using the syntax specified in table 4.

Example: `date format={ B d, Y}`

This would result in a full month name followed by the day of the month, then a comma and finally the year.

Each character in the format is either translated into an element of the date format or passed through as is. This includes punctuation and spaces. (Note that macros etc. won't work here because the macro will be broken down and 'translated' token-by-token.)

The format codes, listed in table 4, are mostly a subset of the format codes provided by GNU's date command, with a few extras not relevant to GNU[21].

A subset of the date-specification codes (as indicated in table 4) is available to customise the formatting of years on the timeline itself. In the case of the timeline, era labels may instead be added at each end to avoid the clutter of including BCE or CE with every year.

`date format` = {⟨*date format specification*⟩}
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format for dates.

Default: `!d/!m/!Y\thinspace !E` (with eras)

Default: `!d/!m/!Y` (without eras)

`event/date format` = {⟨*date format specification*⟩} *event*
date format key

[21] I am grateful to Joseph Wright for providing the code implementing this at TEX StackExchange: 327642.

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format for event dates. *This key overrides* `show eras`, `without eras`, `full dates` *and* `only years` *for elements of* tag *type* event.

Default: `!d/!m/!Y\thinspace !E` (with eras)

Default: `!d/!m/!Y` (without eras)

The following keys set `event/date format` conditionally. This may be used to switch between formats showing eras or only years and no eras or full dates while ensuring uniformity of all formats with or without eras, for example. For instance, it may make little sense to use full dates for events where only the year is known or which occurred when different calendars were used, but you might still want full dates for other cases. *These keys override* `show eras`, `without eras`, `full dates` *and* `only years` *for elements of* tag *type* event.

`event/show eras/full` = {⟨*date format specification*⟩} *event*
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format to use for event when showing full dates with eras.

Default: `!d/!m/!Y\thinspace !E`

`event/show eras/only years` = {⟨*date format specification*⟩} *event*
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format to use for event when showing only years with eras.

Default: `!Y\thinspace !E`

`event/without eras/full` = {⟨*date format specification*⟩} *event*
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format to use for event when showing full dates without eras.

Default: `!d/!m/!Y`

`event/without eras/only years` = {⟨*date format specification*⟩} *event*
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format to use for event when showing only years without eras.

Default: `!Y`

life and period are more complex as date ranges are involved, but the basic structure works in the same way.

`life/date formats` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
`period/date formats`
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default formats for life or period dates. In these cases, we have two dates — either a birth and death or a start and end. You might want different formats for the two and you might want different formats when the first date is BCE and the second CE. Hence, we need to specify three formats. The first argument specifies the format to use for the birth or start date when the death or end date occurs in the same era. The second specifies the format to use for the first date when the eras differ. The third specifies the format to use for the death or end date. *These keys override* `show eras`, `without eras`, `full dates` *and* `only years` *for elements of* tag *types* life *and* period *respectively.*

Default: `{!d/!m/!Y}:{!d/!m/!Y\thinspace !E}:{!d/!m/!Y\thinspace !E}` (with eras)

Default: `{!d/!m/!Y}:{!d/!m/!Y}:{!d/!m/!Y}` (without eras)

The following keys override `date formats` *for elements of* tag *types* life *and* period *respectively.* They work in the same way as those explained above for event.

`life/show eras/full` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
`period/show eras/full`
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default formats to use for life or period when showing full dates with eras.

Default: `{!d/!m/!Y}:{!d/!m/!Y\thinspace !E}:{!d/!m/!Y\thinspace !E}`

`life/show eras/only years` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
`period/show eras/only years`
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default formats to use for life or period when showing only years with eras.

Default: `{!Y}:{!Y\thinspace !E}:{!Y\thinspace !E}`

`life/without eras/full` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
`period/without eras/full`
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default formats to use for life or period when showing full dates without eras.

Default: `{!d/!m/!Y}:{!d/!m/!Y}:{!d/!m/!Y}`

`life/without eras/only years` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
`period/without eras/only years`
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default formats to use for life or period when showing only years without eras.

Default: `{!Y}:{!Y}:{!Y}`

`every date format` = {⟨*date format specification*⟩}
date format key

Sets *all* date formats for *all* tags and the default format to ⟨*date format specification*⟩. This key does not affect the formatting of years, minor years or eras on the timeline itself.

Default: none

Initially: none

`bce year label` = ⟨*text*⟩
key

The label to use if showing the BCE era in text tags. Note this is not the label used if marking eras on the timeline, unless including them as part of year labels.

Default: `\textsc{bce}`

```
\begin{chronos}
  [
    bce year label=BCE,
  ]
\end{chronos}
```

The label is available as `\bceyearlabel` inside the environment `chronos`. In addition, it is made available at the end of the preamble if the command is not otherwise defined.

`ce year label` = ⟨*text*⟩
key

The label to use if showing the CE era in text tags. Note this is not the label used if marking eras on the timeline, unless including them as part of year labels.

Default: `\textsc{ce}`

```
\begin{chronos}
  [
    ce year label=\textsc{ad},
  ]
\end{chronos}
```

The label is available as `\ceyearlabel` inside the `chronos` environment. In addition, it is made available at the end of the preamble if the command is not otherwise defined.

The timeline itself features only years (but see `event years on line` for a limited exception).

`year format` = {⟨*year format specification*⟩}
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format for years. This is the format used to format 'major' years on the timeline.

Default: `!Y\thinspace !E` (with eras)

Default: `!Y` (without eras)

`minor year format` = {⟨*year format specification*⟩}
date format key

When used in the ⟨*chronos preamble*⟩ or in `\chronosset`, sets the default format for 'minor' years.

Default: `!c`

The idea is that you might want, say, four-digit years every half century and three-digit years every hundred years in between.

`timeline/timeline mark eras` = `true|false`
boolean key

Should era labels be included at the end(s) of the timeline? Note that a label will only be shown if the dates the timeline covers include some in the relevant era. So if your timeline starts at 500 CE, the BCE will be omitted and if it ends at 200 BCE, the CE will be omitted.

Default: `true`

Initially: `false`

`timeline bce label` = ⟨*text*⟩
key

The label to use if marking the BCE era on the timeline. Note this is not the label used if showing eras in text tags.

Default: `BCE`

```
\begin{chronos}
  [
    timeline bce label=BC,
  ]
\end{chronos}
```

The label is available as `\celabel` inside the `chronos` environment. In addition, it is made available at the end of the document preamble for general use if the command is not otherwise defined.

`timeline ce label` = ⟨*text*⟩
key

The label to use if marking the CE era on the timeline. Note this is not the label used if showing eras in text tags.

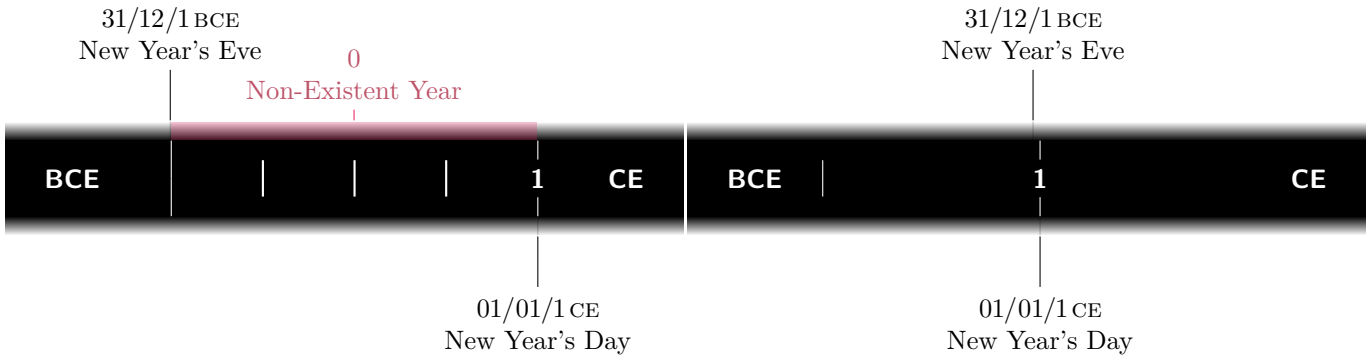
Default: `CE`

```
\begin{chronos}
  [
    timeline ce label=AD,
  ]
\end{chronos}
```

The label is available as `\bcelabel` inside the `chronos` environment. In addition, it is made available for general use at the end of the document preamble if the command is not otherwise defined.

8.2.3 The Problem of the Non-Existent Year

Chronos uses pgfcalendar to calculate Julian day numbers from dates when constructing the timeline. Generally, this works well, but an issue occurs if your timeline spans the two eras (BCE



(a) pgfcalendar default. (b) chronos adjustment.

Figure 18: The problem of the non-existent year.

and CE). Pgfcalendar assumes there was a year zero (fig. 18a), which historians will assure you there was not.

By default, chronos corrects for this (fig. 18b), but the correction can be switched off if desired (fig. 18a).

`timeline/year zero` = `true|false`
boolean key

Whether to tolerate the year zero.

Default: `true`

Initially: `false`

If there is no year zero, certain complications arise. First, what should be marked on the timeline at the 'era switch'? Second, if you ask chronos to mark every hundredth year, say, you probably do not expect it to mark 200 BCE, 100 BCE, 1 CE, 101 CE and so on. Moreover, you might want to do something such as this

```
\foreach \i in {-100,-50,...,300} \node [red,inner sep=2.5pt] at (chronos year \i) {};
```

This seems reasonable, but will fail if `chronos year 0` doesn't exist.

Chronos attempts to solve these problems by handling the 'era switch' as a special case. First, if there is no year zero, it will create *two* coordinates at the switch, provided you have asked it to mark something at this point. `chronos year 0` will exist, as far as chronos is concerned, at the same point as `chronos year 1`. This means you can loop over the era switch in the normal way and expect sensible output, but you can *also* refer to `chronos year 1`, even if you only asked every hundredth year to be marked from 100 BCE.

Second, chronos provides a special option for configuring what is marked on the timeline at the switch of eras.

`timeline/mark at era switch` = `true|false`
boolean key

Whether to use a mark rather than a year at the era switch. If false, the year (e.g. '1') is used; if true, a mark is used instead (illustrated in fig. 18b, though the format will depend on how the timeline is configured).

Default: `true`

Initially: `false` (if showing every year)

Initially: `true` (otherwise)

Note that this option only configures what is marked if something is. If you ask chronos to mark every hundredth year from 150 BCE to 400 CE, nothing will be marked at the era switch (but

chronos will write a warning to the log). Chronos won't do that by default, but, if you insist, it will take you at your word.

`timeline/year at era switch` = `true|false`
boolean key

Whether to use a year rather than a mark at the era switch. This is simply a convenience key which does the opposite of `mark at era switch`.

Default: `true`

Initially: see `mark at era switch`.

8.3 Basic Colours

Chronos uses (or may use) two basic colours: one for foreground and one for background elements.

`background` = ⟨*colour name*⟩
colour key

This is the 'main background colour' for the picture as a whole. This colour is accessible within the `chronos` environment as `chronos main background colour` or `chronos main background color`. Whether it is used and, if so, how, depends on other settings. By default, it is used to determine the colours for the timeline itself and is the basis for the colours used in some tags. It is also used in some standard `chronos styles`.

Default: `white`

```
\begin{chronos}
  [
    background=magenta,
  ]
\end{chronos}
```

`foreground` = ⟨*colour name*⟩
colour key

This is the 'main foreground colour' for the picture as a whole. This colour is accessible within the `chronos` environment as `chronos main colour` or `chronos main color`. Whether it is used and, if so, how, depends on other settings. By default, it is used to determine the colours for the timeline itself and is the basis for the colours used in some tags. It is also used as the `default colour` for connections, lines and text tags and in some standard `chronos styles`.

Default: `black`

```
\begin{chronos}
  [
    foreground=red,
  ]
\end{chronos}
```

For other colours, see sections 8.4.5 and 8.8.

8.4 Timeline

See section 6.1 for an overview of the timeline's components and construction.

Placing different elementson different layers enables the same basic building blocks to result in different styles, but the blocks may also be configured directly. The layers on which the connections and lines of items connected to the timeline are drawn also affects the appearance. For example, putting connections behind the border results in circular chronos connectors appearing as semicircles. Chronos's use of layers is explained in sections 6.4 and 10.

`connections on` = `background|middle ground|main|foreground|overlay`
`lines on`
`timeline/timeline on`
`timeline/border on`
choice key

Which `layer` each type of element should be placed on. Aside from `main` these are not standard layers. In particular, `background` is not the standard Ti*k*Z `background` layer, but instead refers to the `chronos background` layer.

Default: dependent on other options

See section 6.4.

The timeline should be configured using the following key.

`timeline` = {⟨*key-value list*⟩}
key

⟨*key-value list*⟩ should be a list of chronos keys from the timeline configuration options. These keys may also be accessed more verbosely as `/chronos/timeline/`⟨*key name*⟩ or, in the ⟨*chronos preamble*⟩ or in `\chronosset` as `timeline/`⟨*key name*⟩. Some may also work without the `timeline/` prefix, but *this is not guaranteed and may break without notice in future releases.*

```
\begin{chronos}
  [
    timeline={% timeline configuration
      dates={1310-02-03}:{1350-06-07},
      timeline foreground=black,
      timeline background=gray,
      minor years,
      timeline height=5pt,
      timeline width=\textwidth,
      timeline era margin=10pt,
      major step font=\sffamily\bfseries,
      minor step font=\sffamily\bfseries\small,
      timeline minor marks,
      timeline marks,
      timeline years=above,
    },
  ]
\end{chronos}
```

Timeline configuration keys are prefixed with `timeline/` in this manual.

8.4.1 Timeline Dates

`timeline/dates` = ⟨*start date*⟩:⟨*end date*⟩
date key

The fist and last date to be represented on the timeline. Dates must be specified as explained in section 8.2. This key offers a more compact syntax as an alternative to the keys `start date` and `end date` (or `start` and `end`) explained below. That is

```
\begin{chronos}
  [
    timeline={%
      dates={1310-02-03}:{1350-06-07},
%     equivalent to
      start date={1310-02-03},
      end date={1350-06-07},
%     equivalent to
      start={1310-02-03},
      end={1350-06-07},
    },
  ]
\end{chronos}
```

`timeline/start date` = {⟨*date*⟩}
`timeline/start`
date key

The first date to represent on the timeline, specified as explained in section 8.2.

```
\begin{chronos}
  [
    timeline={%
      start date={1310-02-03},
%     equivalent to
      start={1310-02-03},
    },
  ]
\end{chronos}
```

timeline/end date = {⟨*date*⟩}
timeline/end date key

The last date to represent on the timeline, specified as explained in section 8.2.

```
\begin{chronos}
  [
    timeline={%
      end date={1350-06-07},
%     equivalent to
      end={1350-06-07},
    },
  ]
\end{chronos}
```

8.4.2 Timeline Dimensions

See note 8.1.5.1.

The dimensions of the timeline line and border are illustrated in fig. 19.

The total height of the timeline is a function of the dimensions `timeline height` and `timeline border height`:

$$\text{timeline height} + 2 \cdot \text{timeline border height}$$

The total width is `timeline width`. The width includes the width used to represent the time covered by the timeline and twice the `timeline margin`. If era labels are used, the width also includes the space used for these[22] and the `timeline era margins`.

For example,

```
\begin{chronos}
  [
    timeline={%
      timeline height=10mm,
      timeline border height=2.5mm,
      timeline width=200mm,
      timeline mark eras,
      timeline margin=5mm,
      timeline era margin=2.5mm,
      dates={-200}:2000,
    },
  ]
\end{chronos}
```

would result in a total timeline height of 15mm and a total timeline width of 200mm. The width used to represent the years from 200 BCE to 2000 CE would be

$$200\text{mm} - 2 \cdot 5\text{mm} - 2 \cdot 2.5\text{mm} - \text{width of BCE label} - \text{width of CE label}$$

[22] I am grateful to Martin Scharrer for providing the code implementing this at TEX StackExchange: 56405.

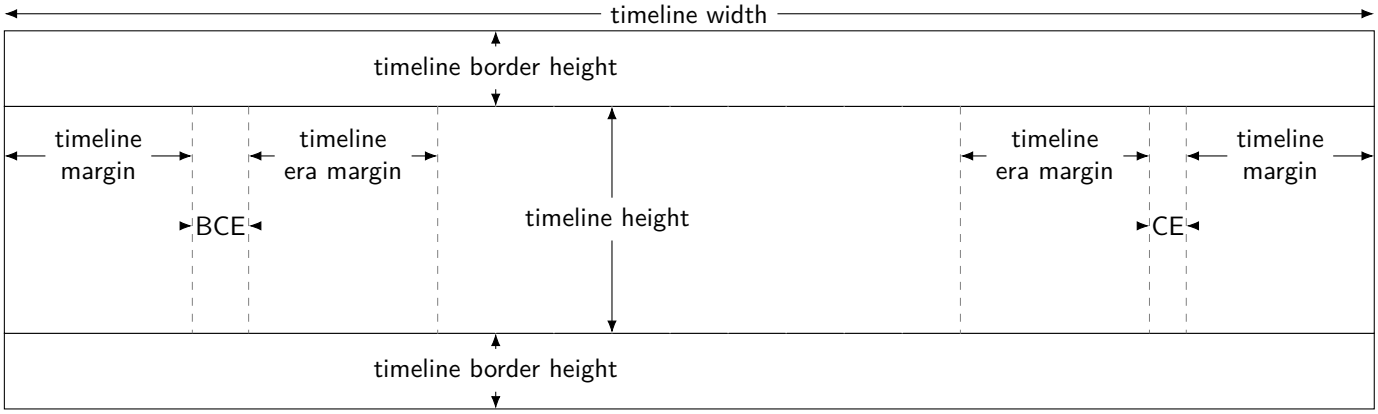


Figure 19: Timeline dimensions.

that is,

185mm − width of BCE label − width of CE label

`timeline/timeline height` = ⟨*dimension*⟩
`timeline/height`
dimension key

The height of the timeline excluding any border.

Default: dependent on other options

For example,

```
timeline={
  timeline height'=10mm,% we can use ' here
},
```

`\timelineht`
macro

The `height of the timeline`. This macro is available *only at the end of the* ⟨*chronos preamble*⟩ *and can be considered reliable only within the* ⟨*timeline specification*⟩[23]. Despite its unreliability, early availability is essential to some chronos styles definitions. In these cases, the chronos style is responsible for ensuring accuracy (or compensating for inaccuracy). In standard cases, this happens automatically, even though it is not guaranteed. However, if you neither load a chronos style nor configure dimensions explicitly, you should not try to use this macro before the timeline is constructed.

`timeline/timeline border height` = ⟨*dimension*⟩
dimension key

The height of each of the upper and lower borders.

Default: dependent on other options

For example,

```
timeline={
  timeline border height'+=2.5pt,% we can use ' here
},
```

`\timelineborderht`
macro

The `height of the border`. This macro is available *only within the* ⟨*timeline specification*⟩.

`timeline/timeline width` = ⟨*dimension*⟩
`timeline/width`
dimension key

The total width of the timeline, including margins.

Default: `\textwidth`

For example,

[23] Note that the unreliability applies to the internal macro, too.

```
timeline={
  timeline width=.75\paperheight,% we cannot use ' here
  timeline width'-=10mm,% we can use ' here
},
```

`\timelinewd` *macro*
The `width of the timeline`. This macro is available *only within the* ⟨*timeline specification*⟩.

`timeline/timeline margin` *dimension key*
= ⟨*dimension*⟩

The horizontal space to allow at each of the two ends of the timeline.

Default: `15pt`

For example,

```
timeline={
  timeline margin'+=-2.5pt,% we can use ' here
},
```

`timeline/timeline era margin` *dimension key*
= ⟨*dimension*⟩

The horizontal space to allow between the first/last point on the timeline and the era labels.

Default: `15pt`

For example,

```
timeline={
  timeline era margin+=0.05,% we can't use ' here
},
```

The following keys determine dimensions of the chronos picture as a whole. They do not affect the dimensions of the timeline itself.

`headings border` *dimension key*
= ⟨*dimension*⟩

The distance between the top of the highest `level` and the top of the space used for headers.

Default: `15pt` + ⟨*headings drop*⟩ + ⟨*upper subheadings drop*⟩ + ⟨*lower subheadings drop*⟩ (if there are one or more levels above the timeline)

Default: `5pt` + ⟨*headings drop*⟩ + ⟨*upper subheadings drop*⟩ + ⟨*lower subheadings drop*⟩ (otherwise)

`headings drop` *dimension key*
= ⟨*dimension*⟩

The distance between the top of the border and the headings.

Default: `0pt` (if headings are omitted)

Default: `15pt` (if headings are used)

Note that you should set this explicitly to `0pt` *if using* subheadings *without* headings.

`subheadings drops` *dimension key*
= `{`⟨*dimension 1*⟩`}:{`⟨*dimension 2*⟩`}`

The distances between the headings and upper subheadings and between the tops of the upper subheadings and lower subheadings.

Default: `0pt:0pt` (if headings are omitted)

Default: `12pt:10pt` (if headings are used)

Note that you should set this explicitly to `0pt:0pt`, ⟨*dimension*⟩`:0pt` *or* `0pt:`⟨*dimension*⟩ *if using* headings *without* upper subheadings *and/or* lower subheadings *or only one of* upper subheadings *or* lower subheadings.

`headings drops'` = {⟨*dimension 1*⟩}:{⟨*dimension 2*⟩}:{⟨*dimension 3*⟩}
`headings'+`
`headings'-`
dimension key

A convenience key equivalent to setting `headings drop'` to ⟨*dimension 1*⟩ and `subheadings drops'` to ⟨*dimension 2*⟩ and ⟨*dimension 3*⟩. *Note that only the ' forms are available.* For pgfmath support, use `headings drop` and `subheadings drops`.

`outer border` = ⟨*dimension*⟩
dimension key

If a frame is created, this is the outer border. In effect, the bounding box will be set to be this distance from the frame, less half the line width used to draw it.

Default: `5pt`

`borders'` = {⟨*dimension*⟩}:{⟨*dimension*⟩}:{⟨*dimension*⟩}:{⟨*dimension*⟩}:{⟨*dimension*⟩}:{⟨*dimension*⟩}
`borders'+`
`borders'-`
dimension key

Sets the `headings border`, `top border`, `right border`, `bottom border`, `left border` and `outer border` in one go. *Note that only the ' forms are available.* For pgfmath support, use `top border`, `right border`, `left border`, `bottom border` and `headings border`.

If you're not sure what this key does or uncertain whether to use it, it is not the key you are looking for. Setting the `outer border` and `headings border` suffices in most cases.

`top border` = ⟨*dimension*⟩
`right border`
`bottom border`
`left border`
dimension key

If the frame does not use the bounding box, these dimensions determine the internal margin between each of the top of the headings, the timeline's right end, the bottom of the lowest `level`, the timeline's left end and the frame, less half the line width used to draw the frame.

Default: `0pt`

Most people should let the frame use the bounding box, which is the default, and leave these dimensions alone.

8.4.3 Timeline Marks and Years

Chronos offers two primary styles of timeline. In one, the line has sufficient vertical depth (`timeline height`) for years, era labels and marks to be drawn on the timeline itself. In the other, the timeline may be much thinner, with marks, era labels and years drawn above or below the line. In this case, the marks appear to grow out from the line and the year labels float slightly above or below.

It is also possible to use chronos to draw a line with neither marks nor years. Alternatively, you might want to create 'invisible' marks or years, which may be useful for placement purposes[24]. Figure 19 shows a timeline in which this has been done by setting the foreground and background colours equal. The nodes are used to place the arrows and labels illustrating the various dimension keys.

`timeline/timeline years` = `on line|off line|above|below|none`
choice key

Whether years (and any era labels and marks) should be created on the timeline, off it or not at all and, if they should be off the timeline, whether they should be above or below it. The options are mutually exclusive, except that `off line` implies either `above` or `below`. See also `minor years`, `timeline marks`, `timeline minor marks` and `timeline bare marks`, which further determine what exactly is shown.

Default: none

Initially: `on line`

it may actually make sense to write something like

```
\begin{chronos}
  [
```

[24] You don't need this simply to connect elements to the timeline. chronos doesn't depend on the creation of marks or years for that purpose.

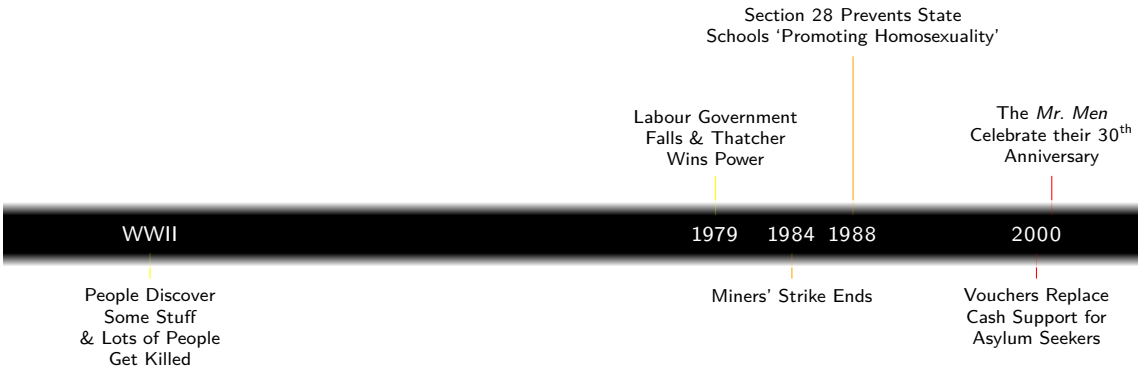


Figure 20: Illustration of `event years on line`.

```
    timeline={%
      timeline years=off line,
      timeline years=none,
    },
  ]
\end{chronos}
```

if one wants an off-line style of line with no years or marks. I don't know why one *would* want such a thing, but the possibility is there.

`none` is actually intended to support a particular style of event-only timeline, in which the dates are created on the line itself.

`event years on line` *key*

Don't create regular year labels or marks on the timeline itself. Instead, put the years of subsequently added events onto the line. This option creates a timeline suitable for showing years on the timeline, but doesn't create any labels when drawing the line itself.

Assuming `timeline years` is not set to `none`, as it is if `event years on line` is enabled, the following keys determine how and where chronos represents time on (or off) the timeline itself. The primary concepts here are those of major steps and minor steps. The space available to represent time on the timeline (see section 8.4.2) is divided into major steps and, optionally, further divided into minor steps. These can be highlighted with `timeline marks` and `timeline minor marks` and are set using `step major year` and `step minor year`.

In addition to years, `timeline bare marks` may be used to create unlabelled subdivisions at intermediate points. In the standard case, the value of `step divisions` is used to divide the distance equally. For example, if you specify 5, chronos will use 4 lines to subdivide each. No attempt is made to place these so they correspond to any particular date: if you request 12, chronos will not make the division for February smaller than the one for December.

However, if a timeline is short, chronos proceeds differently. 'Short' refers to temporal duration rather than dimension and includes any timeline which begins and ends in the same year or in consecutive years.

`timeline/minor years` = `true|false`
boolean key

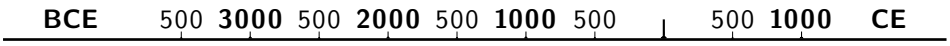
Whether to label minor years, in addition to major years.

Default: `true`

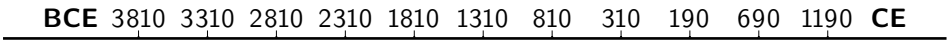
Initially: `true`

`timeline/step major year` = {⟨*positive integer*⟩}
`timeline/step major years`
key

How often to label major years on the timeline if showing them. Use this key if you want a larger or bolder font and/or a different date format and/or thicker or longer marks to be used for some



(a) Default selection of major yearss and minor yearss.



(b) Selected major yearss and minor yearss with `step from year=3810`.

Figure 21: Default (fig. 21a) and non-default (fig. 21b) selection of major years and minor years when `dates={-3814}:1213`, `step major year=1000` and `step minor year=500`.

year labels. You can also use this key if you want all year labels on the timeline to use the same format. For example, you might want to print the full 4 digits of the year each thousand years.

Default: dependent on other options

`timeline/step minor year` = {⟨*positive integer*⟩}

`timeline/step minor years` *key*

How often to label minor years on the timeline if displaying them. The idea is that you might want a smaller or lighter font and/or a different date format and/or thinner or shorter marks to be used for intermediate year labels. For example, you might want to print full years only every millennium and the last 3 digits of the year each century.

Default: dependent on other options

Chronos labels minor years only if labelling major years. Although the package attempts to correct the result if only `minor years` are requested, it is better to use `step minor year` only in conjunction with major years.

`timeline/step year` = {⟨*positive integer*⟩}

`timeline/step years` *key*

How often to label years on the timeline, if you want them all to be formatted in the same way. This key sets `step major years` internally and unsets `step minor year`.

Default: dependent on other options

Chronos tries to label years *modulo* the `step major year` and `step minor year` (or `step year`). This means you can start the timeline at 3,814 BCE, request major years every millennium and minor years every half millennium without worrying about which year should be the first (labelled) year. Figure 21 illustrates chronos's default choices in this case. Note that the first year is *not* determined by the start date alone in fig. 21a, but is determined in conjunction with `step major year` and `step minor year` so that -1 BCE ends (and 1 CE begins) at a major year and the turn of millennia generally occur at major years, while the first minor year is 3,500 BCE.

`timeline/step from year` = {⟨*integer*⟩} *key*

Do not use this key unless chronos *produces undesirable results by default.* If for some reason you do *not* want years on the timeline to be determined modulo `step major year` and `step minor year`, you may tell chronos where to begin stepping from. In this case, chronos will issue a warning, but it will implement your choice.

Default: dependent on other options

Note that fig. 21b effectively includes no major yearss because chronos tests whether the current year is modulo the `step major year` when deciding how to format the year label and markss.

`chronos year` ⟨*YYYY*⟩
`chronos year -`⟨*YYYY*⟩ *coordinate*
`chronos year` ⟨*YYYY*⟩
`chronos year -`⟨*YYYY*⟩ *node*

Every major year and minor year receives a name: a `node` or `coordinate` is created with the name `chronos year YYYY` for CE and `chronos year -YYYY` for BCE. No zeros are added, so years with fewer than four digits get nodes or coordinates with names such as `chronos year -1`. Chronos creates all years at the beginning of the year i.e. 1st January. (This is analogous to a ruler which marks each centimetre at its beginning.)

`chronos origin` *coordinate*

If the timeline spans the switch of eras from BCE to CE *and* the years represented on the timeline

are modulo an additional coordinate named `chronos origin` is created at the era switch point, `chronos year 1`.

`chronos year 0` *coordinate*

If `year zero` is `false`, as it is by default, a third coordinate named `chronos year 0` is created at `chronos origin`[25].

`timeline/step divisions` = {⟨*positive integer*⟩} *key*

Whether the timeline should be further subdivided between major and/or minor years using bare marks and, if so, how many sub-divisions should be made. These are simple subdivisions of the distance between points. Unlike the labels/marks made for years, they do not involve calculations involving dates and are not named.

Default: dependent on other options

`timeline/timeline year` = {⟨*key-value list*⟩} *key*

Adds ⟨*key-value list*⟩ to the common style used when putting major years and minor years onto the timeline. Do not specify `font` or `anchor` here as they will be overridden. Although both major and minor years use the same general style, they may and, by default do, use different fonts and date format keys.

Example: `timeline/timeline year=fill=chronos timeline background colour`

Default: `text=`⟨*timeline foreground*⟩`, text opacity=1, align=center, fill opacity=.75` (off line)

Default: `text=`⟨*timeline foreground*⟩`, anchor=center` (on line)

`timeline/timeline years` = {⟨*text*⟩}

`anchor` *key*

The Ti*k*Z `anchor` to use when creating the `nodes` for years on or off the timeline. *Do not set this option unless you know you need to.* In most cases, chronos will pick a sensible default. The key is provided primarily for cases where you want to rotate the year labels in styles which place them off the line. Even then, you should not need to change the setting if using a style designed for rotation, unless you need to change the angle.

Default: dependent on other options

`timeline/timeline marks` = `true|false` *boolean key*

Whether to draw vertical marks on or off the timeline at major years using the style set with `timeline mark`.

Default: `true`

Initially: `true`

`timeline/timeline minor marks` = `true|false` *boolean key*

Whether to draw vertical marks on or off the timeline at minor years using the style set with `timeline minor mark`.

Default: `true`

Initially: `true`

`timeline/timeline show years` = `true|false` *boolean key*

Whether to represent years on or off the timeline at all. If false, neither labels nor marks will be added when the timeline is constructed. This is useful if you wish to use a style such as `event years on line`, but is the nuclear option otherwise.

Default: `true`

Initially: `true`

[25] So the non-existent year zero is marked at the same point as the existent year one. This avoids complications in `\foreach` loops.

`timeline/timeline bare marks` = `true|false`
boolean key

Whether to draw bare marks on or off the timeline in between years[26] using the style set by `timeline bare mark`. If you specify `step divisions`, this key will be automatically enabled. If you don't want bare marks, don't set/set to zero `step divisions`.

Default: `true`

Initially: `false`

`timeline/timeline mark` = {⟨*key-value list*⟩}
key

Adds to the style used for the vertical lines drawn when chronos labels a major year on or off the timeline and `timeline marks` is true. These correspond to the major steps at which chronos puts years.

Example: `timeline mark=thick`

Default: `draw=`⟨*timeline foreground*⟩`, Triangle[width=0pt 3,reversed,length=0pt 1.5]-, thin, shorten >=-2.5pt` (off line)

Default: `draw=`⟨*timeline foreground*⟩ (on line)

`timeline/timeline minor mark` = {⟨*key-value list*⟩}
key

Adds ⟨*key-value list*⟩ to the style used for the vertical lines drawn when chronos labels a minor year on or off the timeline and `timeline minor marks` is true. These correspond to the minor steps at which chronos puts years.

Example: `timeline mark=thin, shorten >=-2pt`

Default: `draw=`⟨*timeline foreground*⟩`, Triangle[width=0pt 3,reversed,length=0pt 1.5]-, very thin, shorten >=-2.5pt` (off line)

Default: `draw=`⟨*timeline foreground*⟩`, thin` (on line)

`timeline/timeline bare mark` = {⟨*key-value list*⟩}
key

Adds ⟨*key-value list*⟩ to the style used to draw lines at `step divisions`, provided `timeline marks` is true.

Example: `timeline bare mark=thin, <-`

Default: `draw=`⟨*timeline foreground*⟩`, Triangle[width=0pt 3,reversed,length=0pt 1.5]-, very thin, shorten >=-1.5pt` (off line)

Default: `draw=`⟨*timeline foreground*⟩`, thick` (on line)

`timeline/timeline all marks` = {⟨*key-value list*⟩}
key

Adds to the styles used to draw lines at major years, minor years and step divisions. This is equivalent to passing ⟨*key-value list*⟩ to each of `timeline mark`, `timeline minor mark` and `timeline bare mark`.

`event year on line` *style*

The style used to mark years on the timeline if `event years on line` is enabled. By default, the style otherwise used for years when on the line is used. Redefine this if you wish, but you could also use `timeline years`, since no other years will be set on the line anyway.

`event year on line skip` *key*

Don't put this particular event's year on the timeline. This can be used if the line would otherwise become too crowded. See section 9.3.

`timeline/era switch off line` *style*

The style to use if years are 'off line' and `mark at era switch` is `true`. With the standard settings, you would get a small mark at the switch, no different from other intermediate marks. Likely you want something more similar in stature to the year labels. Redefine or supplement using standard Ti*k*Z techniques.

[26] If your timeline is very short and 12 `step divisions` are set, chronos will actually mark months. In other cases, marks simply divide the available space and are not placed by date.

Default: `thick, shorten >=0pt`

```
\begin{chronos}
  [
    timeline={%
      era switch off line/.append style={ultra thick},% retain undoing of shortening
in default, but make mark thicker
      era switch off line/.style={ultra thick, shorten>=-2pt},% make mark thicker and
longer
      era switch off line/.style={shorten>=-2pt},% make mark longer but use whatever
thickness is used for other marks
    },
  ]
\end{chronos}
```

8.4.4 Timeline Fonts

`major step font` = {⟨*key-value list*⟩}
key

The font used for major years.

Default:

```
\begin{chronos}
  [
    timeline={%
      major step font=\sffamily,
    },
  ]
\end{chronos}
```

`timeline/minor step font` = {⟨*key-value list*⟩}
key

The font used for minor years.

Default:

```
\begin{chronos}
  [
    timeline={%
      minor step font=\sffamily\small,
    },
  ]
\end{chronos}
```

`timeline/eras font` = {⟨*key-value list*⟩}
key

The font used for era labels on the timeline.

Default:

```
\begin{chronos}
  [
    timeline={%
      eras font=\sffamily\bfseries\large,
    },
  ]
\end{chronos}
```

8.4.5 Timeline Colours

timeline/timeline border inner colour = ⟨*colour name*⟩
timeline/timeline border inner color *colour key*

The innermost colour used for the gradient used to shade the timeline borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border inner colour` or `chronos timeline border inner color`.

Default: the `timeline background` colour, which is itself `black` by default.

```
\begin{chronos}
  [
    timeline={%
      timeline border inner colour=blue,
    },
  ]
\end{chronos}
```

timeline/timeline border outer colour = ⟨*colour name*⟩
timeline/timeline border outer color *colour key*

The outermost colour used for the gradient used to shade the timeline borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border outer colour` or `chronos timeline border outer color`.

Default: the `background` colour, which is itself `white` by default.

```
\begin{chronos}
  [
    timeline={%
      timeline border outer colour=green!5!white,
    },
  ]
\end{chronos}
```

timeline/timeline border middle colour = ⟨*colour name*⟩
timeline/timeline border middle color *colour key*

The middle colour used for the gradient used to shade the idx post=colour configuration[type=element,idx as=timeline border]timeline borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border middle colour` or `chronos timeline border middle color`.

Default: a 50-50 mix of the `timeline border outer colour` and `timeline border inner colour`.

```
\begin{chronos}
  [
    timeline={%
      timeline border middle colour=blue!20!green,
    },
  ]
\end{chronos}
```

timeline/timeline background = ⟨*colour name*⟩
colour key

The colour used for the background of the central part of the timeline. This colour is accessible within the `chronos` environment as `chronos timeline background colour` or `chronos timeline background color`.

Default: the `foreground` colour, which is itself `black` by default (if putting years/marks on the line).

Default: the `background` colour, which is itself `white` by default (otherwise).

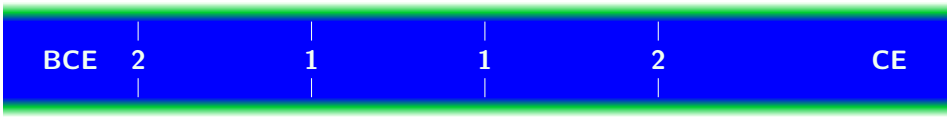


Figure 22: Cumulative effect of colour settings given as examples in sections 8.4.5 and 8.8.

```
\begin{chronos}
  [
    timeline={%
      timeline background=blue,
    },
  ]
\end{chronos}
```

`timeline/timeline foreground colour` *key* = ⟨*colour name*⟩

The colour used for the foreground of the central part of the timeline. This colour is accessible within the `chronos` environment as `chronos timeline foreground colour` or `chronos timeline foreground color`.

Default: the `background` colour, which is itself `white` by default (if putting years/marks on the line).

Default: the `foreground` colour, which is itself `black` by default (otherwise).

```
\begin{chronos}
  [
    timeline={%
      timeline foreground=green!5!white,
    },
  ]
\end{chronos}
```

The cumulative effect of the colour settings given in the examples in this section, together with the `background` and `foreground` from section 8.8 is shown in fig. 22.

8.4.6 Timeline Style

The timeline's overall style can be customised using the following keys, which should (and, by default, do) utilise colours from the colour scheme (see section 13.2). Unless you are creating a chronos style, it is best to *add to* rather than *replacing* the existing configuration. For example, if you wish the line to take the form of an arrow, you can simply add the use of an appropriate arrow tip, without modifying the colours, dimensions or markings.

`timeline/timeline line` = {⟨*key-value list*⟩}
`timeline/timeline line'`
`timeline/timeline line+` *key*

The style of the timeline line. `timeline/timeline line+` adds to the current list; `timeline/timeline line` and `timeline/timeline line'` replace it.

Default: `empty`

Initially: dependent on other options

This key makes it possible to override the default drawing or filling of the timeline lines.

For example, `blues below` includes the following in its timeline configuration,

```
timeline={%
  ...
```

```
  timeline line={Bar-Latex,chronos timeline foreground colour,double=chronos timeline
background colour,line width=\timelineht/3,double distance=\timelineht/3,shorten <=-\
timelineht/3,shorten >=-3pt-2.1\timelineht},
  timeline config+={\pgfqkeys{/chronos/timeline}{timeline width-={3pt+2.43\timelineht}}},

  ...
}
```

To make the timeline line into an arrow, without otherwise modifying the existing style, use, for example,

```
timeline={%
  ...
  timeline line+={shorten >={-10mm}, -{Triangle Cap[length=10mm]}},
  timeline config+={\pgfqkeys{/chronos/timeline}{timeline width-=10mm}},
  ...
}
```

The adjustments are required to ensure that the tapered part is not counted when `chronos` calculates how much of the total `timeline width` is available to represent time.

`timeline/timeline arrow` = `true|false`
boolean key

Whether the timeline should be or have an arrow or arrows.

Default: `true`

Initially: `false`

Whether this has any effect depends entirely on the chronos style. With the default settings, it does nothing but trigger a warning, since `on line` styles cannot have arrows.

`timeline/no timeline arrow` *key*

A convenience key which sets `timeline/timeline arrow false`. *Whether this has any effect depends entirely on the chronos style.*

`timeline/timeline border` = {⟨*key-value list*⟩}
`timeline/timeline border'`
`timeline/timeline border+`
key

The style of the timeline border. `timeline/timeline border+` adds to the current list; `timeline/timeline border` and `timeline/timeline border'` replace it.

Default: `empty`

Initially: dependent on other options

This key makes it possible to override the default gradients used to fill the borders.

8.5 Frame

`frame` = `true|false`
boolean key

Whether to draw a frame. This is initially false, but use of `main/frame` will automatically set it to true.

Default: `true`

Initially: `false`

`frame uses bb` = `true|false`
boolean key

Whether the bounding box should be used to determine any frame at the end of the `chronos` environment. This is true by default and almost certainly what you want unless you are smuggling code into the end of the environment or using the frame for nefarious purposes.

Default: `true`

Initially: `true`

`main/frame` = {⟨*key-value list*⟩}
`main/frame'`
`main/frame+`
key

The style of the TikZ `node` used to draw the `frame`. This may be freely redefined as desired.

Default: dependent on other options (empty)

Example: `main/frame={draw=black,ultra thick,inner sep=5pt}`

Example: `main/frame+={double=blue}`

The second form may be useful if you wish to modify, rather than replace, a style defined by a chronos style. `main/frame` and `main/frame'` replace any current list; `main/frame+` adds to it.

8.6 Placing Things: Levels & Coordinates

Knowing where to put things may get tricky in complicated or densely-packed timelines. Chronos offers several techniques to help. The simplest is to simply use existing items as reference points. Chronos names coordinates and nodes routinely and predictably, as explained throughout this documentation. However, sometimes this isn't quite enough. Levels and chronos coordinates offer additional help with vertical and horizontal placement respectively.

8.6.1 Levels

Levels are not (generally) visible elements. They are instead part of the structure behind-the-scenes. They are, if you like, minimal stage-hands.

The idea is to tell chronos how many tiers (approximately) of elements you will create above and below the timeline. For each of these levels, chronos creates a standardised node or `placeholder` based on the settings used for elements of type life when the timeline is constructed. Each of these nodes is named: `level 1`, `level 2`, ... above the timeline and `level -1`, `level -2`, ... below[27]. The first node in each direction is shifted `2pt` from the timeline. Subsequent nodes are created directly above each other, with no separation between.

Together with points on the timeline, you then have a crude system for placing things horizontally and vertically. It also enables you to 'stack' text tags, but create them in any order.

`levels` = {⟨*number above*⟩}:{⟨*number below*⟩}
key

⟨*number above*⟩ and ⟨*number below*⟩ should be non-negative integers specifying how many levels to create above and below the timeline respectively.

Default:

no number of `levels` are created by default (not even zero).

```
\begin{chronos}
  [
    levels=4:4,
  ]
\end{chronos}
```

`levels at` = {⟨*coordinate*⟩}
key

Although they are not intended to be visible in the timeline, placeholder nodes may be rendered visible for debugging or development purposes. As such, it may be useful to move them from their default location.

Default: `chronos mid`

```
\begin{chronos}
  [
    levels at=chronos year -200,% make sure this exists!
  ]
```

[27] You can also refer to the nodes above as `u1`, `u2` etc. and those below as `i1`, `i2` etc.

```
\end{chronos}
```

To render the nodes temporarily visible, see section 14.

8.6.2 Chronos Coordinates

In addition to the coordinates and nodes shown in fig. 3, chronos names a coordinate or node `chronos year` ⟨*year*⟩ for each year represented on the timeline. However, depending on your preferred style, this may not provide sufficient horizontal reference points. In that case, you can create additional coordinates. Like levels, chronos coordinates are not ordinarily visible; unlike levels, there is nothing there to see[28].

`chronos coords` = {⟨*comma-separated list of years*⟩}
comma-separated list key

For each ⟨*year*⟩ in ⟨*comma-separated list of years*⟩, chronos will place a single coordinate named `chronos year` ⟨*year*⟩ at the appropriate point on the timeline. These may be used together with levels to specify coordinates e.g. `(chronos year` ⟨*year*⟩ `|- level` ⟨*n*⟩`)` is the point vertically aligned with `level` ⟨*n*⟩ and horizontally aligned with `chronos year` ⟨*year*⟩.

Default: `empty`

8.6.3 Miscellaneous

`\chronosbaselineskip`
macro

The `chronos` environment sets this macro equal to the current `\baselineskip`. It may be used to fine-tune placement in the same way you might use `\baselineskip` outside a `tikzpicture`.

8.7 Headings

`headings` = {⟨*text*⟩/⟨*coordinate 1*⟩/⟨*coordinate 2*⟩,⟨*text*⟩/⟨*coordinate 1*⟩/⟨*coordinate 2*⟩,...}
`headings+`
`headings'`
comma-separated list key

List of value triplets in the format used by PGF's `\foreach`. The list should consist of one or more triplets where ⟨*text*⟩ is used in capitalised form for the content of a node which will be aligned with `chronos main headings` vertically and placed midway between the horizontal positions of ⟨*coordinate 1*⟩ and ⟨*coordinate 2*⟩. `headings` and `headings+` add to the current list; `headings'` replaces it.

Default: none

See section 8.7.1 for an example.

`heading` = {⟨*text*⟩}{⟨*coordinate 1*⟩}{⟨*coordinate 2*⟩}
`heading+`
`heading'`
key

Add or set a single heading. These forms require the same information as `headings`, `headings+` and `headings'` but as three separate arguments.

Default: none

See section 8.7.1 for an example.

`subheadings` = {⟨*text*⟩/⟨*coordinate 1*⟩/⟨*coordinate 2*⟩/⟨*coordinate 3*⟩,⟨*text*⟩/⟨*coordinate 1*⟩/⟨*coordinate 2*⟩/⟨*coordinate 3*⟩,...}
`subheadings+`
`subheadings'`
comma-separated list key

List of value quadruplets in the format used by PGF's `\foreach`. The list should consist of one or more quadruplets where ⟨*text*⟩ is used in capitalised form for the content of a node which will be aligned with ⟨*coordinate 4*⟩ vertically and placed midway between the horizontal positions of ⟨*coordinate 1*⟩ and ⟨*coordinate 2*⟩. ⟨*coordinate 4*⟩ should be either `chronos upper subheadings` or `chronos lower subheadings`. `subheadings` and `subheadings+` add to the current list; `subheadings'` replaces it.

Default: none

See section 8.7.1 for an example.

[28] You could label them, of course, but they are just regular PGF/Ti*k*Z coordinates and so naturally invisible.

`subheading` = {⟨*text*⟩}{⟨*coordinate 1*⟩}{⟨*coordinate 2*⟩}{⟨*coordinate 3*⟩}
`subheading+`
`subheading'`
key

Add or set a single subheading horizontally aligned with the midpoint between the horizontal positions of ⟨*coordinate 1*⟩ and ⟨*coordinate 2*⟩ and vertically aligned with ⟨*coordinate 3*⟩. ⟨*coordinate 3*⟩ should be either `chronos lower subheadings` or `chronos upper subheadings`, though this is not enforced. These forms require the same information as `subheadings`, `subheadings+` and `subheadings'` but as four separate arguments.

Default: none

See section 8.7.1 for an example.

`century subheadings` = {⟨*number list*⟩}{⟨*text*⟩}
`century subheadings+`
`century subheadings'`
comma-separated list key

Create a subheading aligned with `chronos lower subheadings` for each of the centuries specified in =⟨*number list*⟩, using ⟨*text*⟩ as the superscript for each. Note that for the nth century `chronos year` coordinates much exist for both the year $n00$ and the year $(n+1)00$. `century subheadings` and `century subheadings+` add to the current list; `century subheadings'` replaces it.

Default: none

See section 8.7.1 for an example.

`century subheading` = {⟨*number*⟩}{⟨*text*⟩}
`century subheading+`
`century subheading'`
key

Add or set a single century subheading. These forms require the same information as `century subheadings`, `century subheadings+` and `century subheadings'` but expect a single ⟨*number*⟩.

Default: none

See section 8.7.1 for an example.

8.7.1 Example

For example, here's an excerpt from the code used for fig. 2 which demonstrates the use of keys to create headings and subheadings.

```
\begin{chronos}
  [
    timeline={%
      dates={-500}:1500,
    },
    chronos coords={-500,-450,...,1500},
    headings={heading/chronos year 800/chronos year 1500,another heading/chronos year
-450/chronos year 1,a third heading/chronos year 100/chronos year 800},
    subheadings={subheading on upper level/chronos year -250/chronos year 500/chronos
upper subheadings,subheading on lower level/chronos start/chronos year -100/chronos
lower subheadings,another subheading/chronos year 1000/chronos year 1500/chronos upper
subheadings,yet another subheading/chronos year 500/chronos year 1000/chronos lower
subheadings},
    century subheadings={12,13,...,15}{th},
    century subheading={1}{st},
  ]
\end{chronos}
```

Note the use of `chronos coords` to add coordinates for years which may not be visibly represented on the timelines. This ensures the `chronos year` coordinates needed to place headings, subheadings and century subheadings exist. It is permissible for coordinates to lie beyond the timeline's `end date`, though you may get strange results if you create coordinates too distant from the endpoint.

8.7.2 Headings Configuration

`headings style` = {⟨*key-value list*⟩}
`headings style+`
`headings style'`
key

PGF/Ti*k*Z options to apply to headings. `headings style` and `headings style'` replace the current list; `headings style+` replaces it.

Default: dependent on other options (empty)

Example: `headings style={align=center, anchor=base, inner sep=0pt, outer sep=0pt, color=chronos main colour, opacity=.8, font=\bfseries}`

Although the style is empty by default, `anchor=base` is passed to the node prior to the style. If you do not want this alignment, therefore, you must specify an alternative anchor.

`subheadings style` = {⟨*key-value list*⟩}
`subheadings style+`
`subheadings style'`
key

PGF/Ti*k*Z options to apply to subheadings. `subheadings style` and `subheadings style'` replace the current list; `subheadings style+` replaces it.

Default: dependent on other options (empty)

Example: `subheadings style={align=center, anchor=base, inner sep=0pt, outer sep=0pt, font=\bfseries\itshape\footnotesize, color=chronos main colour!75!chronos main backgrou colour, opacity=.8}`

Although the style is empty by default, `anchor=base` is passed to the node prior to the style. If you do not want this alignment, therefore, you must specify an alternative anchor.

8.8 Colours

For timeline colours, see section 8.4.5. For basic colours, see section 8.3.

The *easiest* way to customise colours is to load a colour scheme as explained in section 7.2.

The *simplest* way to make use of colours is to specify colours for elements manually. Defaults can be configured in the timeline setup.

`life/default colour` = ⟨*colour name*⟩
`event/default colour`
`period/default colour`
`theory/default colour`
`info/default colour`
`life/default color`
`event/default color`
`period/default color`
`theory/default color`
`info/default color`
colour key

Sets the default colour for elements of the specified type. This provides a fall-back colour and ensures some colour is always found, even when none is specified.

Default: `chronos main colour`

See `foreground` in section 8.4.5. For example,

```
\begin{chronos}
  [
    life/default colour=chronos timeline foreground colour,
    event/default colour=chronos timeline foreground colour!50!chronos main colour,
    period/default colour=chronos main colour,
    theory/default colour=chronos timeline background colour,
    info/default colour=chronos main colour!50!chronos main background colour,
  ]
\end{chronos}
```

Alternatively or in addition, colours can be set on a per-element basis (sections 9.3 to 9.5).

8.8.1 Colour Rotation

More complex configuration can be achieved using lists of colours from which chronos selects when adding elements to the timeline. If you wanted to typeset all elements of type life in the

colours of the rainbow taken in order, for example, it would be error prone and inflexible to assign colours manually. Instead, we would like chronos to select the colours in turn, keep track of which colour is used for which element and automatically adjust the assignments if items are inserted or removed from the timeline.

To achieve this, chronos supports colour rotation for text tags, connections and lines of type life, event, period and theory.

Chronos assigns all elements belonging to tags life, event, period, theory and info a colour with a predictable colour name. Chronos determines the colour to assign to the element as follows.

1. First, chronos checks whether a `colour` has been specified for the element.
   ↳ If it has, that `colour` is assigned.
2. If not, chronos checks whether colour rotation is enabled for the relevant type of element.
   ↳ If it is, chronos assigns the next colour from the specified colour list for the type of element in question and according to whether the element will be placed above or below the timeline. That colour is then moved to the bottom of the list.
3. If rotation is not enabled, a configurable `default colour` is assigned instead.

8 sets of colours can be configured which correspond to material placed above and below the timeline for each of default, life, event and period. See section 8.8.3 for details.

8.8.2 Using Colours

There are at least two things you might want chronos to tell you about elements' colours. First, you might want to know the `colour` assigned to a particular element *after* the element is created. Second, you might want to know the `colour` assigned to the current element during creation. Note 8.8.2.1 addresses the first, note 8.8.2.2 the second.

8.8.2.1 Colours by Element Name Regardless of how the colour assigned to an element ends up being determined, chronos assigns the colour a name derived from the element so that it can be used later, if required.

The result of this is that, assuming we have created an element of type life with `name=donald knuth`, we can write

```
\draw [chronos connect=life:donald knuth] (text tag connector donald knuth1) -- (text
tag connector metafont2);
```

to connect Donald Knuth with an element named `metafont`, which might be of type theory. The code used to draw the connection will use the same style and colour as any connection drawn between Donald Knuth and the timeline[29]. This colour can also be (and, by default, is) passed to the text tag. For example, a darker shade might be used for the text and outline of the node, and a paler one as a filling. The colour may also be accessed directly using `colour donald knuth`, `color donald knuth` or, if `simple colour names` are enabled[30], simply `donald knuth`.

`colour` ⟨*name*⟩ Colour names assigned to the element created with `name=` ⟨*name*⟩. *life, event, period, theory, info*
`color` ⟨*name*⟩
colour
Note these names cannot be used during the element's creation in `\chronos`⟨*tag*⟩.

⟨*name*⟩ An additional name for `colour` ⟨*name*⟩. *life, event, period, theory, info*
colour
Requires `simple colour names`.

[29] See section 9.6
[30] See sections 5 and 8.8.4.

8.8.2.2 The Current Tag Colour

You may also wish to refer to an element's assigned colour while creating it.

`chronos current tag colour` / `chronos current tag color` *colour*

The colour assigned to the current element during creation. *life, event, period, theory, info*

This colour is available when creating an element belonging to an appropriate tag i.e. inside the tag context setup when using `\chronoslife`, `\chronosevent`, `\chronosperiod` or `\chronostheory`. Outside a tag context, `chronos current tag colour` and `chronos current tag color` are equivalent to `chronos main colour`.

Example: `\hypersetup{urlcolor=chronos current tag colour}`

Figure 1 uses this code within a `figure` to override the colour of URL links locally in such a way that each hyperlink's colour is the colour of the text tag to which it belongs.

8.8.3 Colour Lists

The lists of colours for colour rotation (section 8.8.1) may be loaded from provided styles, specified directly.

No specific lists are provided for theory, but you can obviously reserve the default lists for this type, if you want distinct lists for everything.

`colours above` = ⟨*list of colour names*⟩

`colors above` *colour list key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, sets the default colour list for use above the timeline to ⟨*list of colour names*⟩.

Default: `Red,Orange,Yellow,Green,Blue,MidnightBlue,Violet`

`colours below` = ⟨*list of colour names*⟩

`colors below` *colour list key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, sets the default colour list for use below the timeline to ⟨*list of colour names*⟩.

Default: `Red,Orange,Yellow,Green,Blue,MidnightBlue,Violet`

`colour rotation` = `true|false`

`color rotation` *boolean key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, determines whether colours are rotated by default or not.

Default: `true`

This key does not override tag-specific settings. Depending on other settings, therefore, using this key may have no effect or it may enable colour rotation for everything.

`rotate all colours` / `rotate all colors` *key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, enables both default colour rotation and colour rotation for all supported tags. This key overrides tag-specific settings.

`no colour rotation` / `no color rotation` *key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, disables default colour rotation. This key does not override tag-specific settings. Depending on other settings, therefore, using this key may have no effect or it may prevent colour rotation completely.

`rotate no colours` / `rotate no colors` *key*

When given in the ⟨*chronos preamble*⟩ or to `\chronosset`, disables both default colour rotation and colour rotation for all tags. This key overrides tag-specific settings.

Note that, like many chronos keys, the effect of setting these depends on the current key path. That means that using a key when creating a tag of type life, for example, the key will have a different effect from using in in the ⟨*chronos preamble*⟩.

`life/colours above` = ⟨*list of colour names*⟩

`life/colors above` *colour list key*

Sets the colour list for use with elements of type life placed above the timeline to ⟨*list of colour names*⟩.

Default: dependent on other options (empty)

`life/colours below` = ⟨*list of colour names*⟩
`life/colors below`
colour list key
Sets the colour list for use with elements of type life placed below the timeline to ⟨*list of colour names*⟩.

Default: `empty`

`event/colours above` = ⟨*list of colour names*⟩
`event/colors above`
colour list key
Sets the colour list for use with elements of type event placed above the timeline to ⟨*list of colour names*⟩.

Default: `empty`

`event/colours below` = ⟨*list of colour names*⟩
`event/colors below`
colour list key
Sets the colour list for use with elements of type event placed below the timeline to ⟨*list of colour names*⟩.

`period/colours above` = ⟨*list of colour names*⟩
`period/colors above`
colour list key
Sets the colour list for use with elements of type period placed above the timeline to ⟨*list of colour names*⟩.

Default: `empty`

`period/colours below` = ⟨*list of colour names*⟩
`period/colors below`
colour list key
Sets the colour list for use with elements of type period placed below the timeline to ⟨*list of colour names*⟩.

Default: `empty`

8.8.4 Simple Colour Names

If you wish to enable or disable `simple colour names` (see sections 5 and 8.8) for a particular timeline, use one of the following two options.

`simple colour names` = `true|false`
`simple color names`
boolean key
Enable or disable `simple colour names`.

Default: `true`

Initially: `true`

Example: `simple colour names=false,`

See section 5 for details, but note that the keys here are implemented differently.

`no simple colour names`
`no simple color names`
key
Disable `simple colour names`.

Example: `no simple colour names,`

See section 5 for details, but note that the keys here are implemented differently. In particular, unlike both `simple colour names` and the load-time option, `no simple colour names` does *not* take an argument.

9 Adding Elements to the Timeline

See section 6.2 for an overview of the components available for use in the timeline's ⟨*timeline additions specification*⟩.

Seven macros are provided for adding elements to the timeline. Conceptually, these are always 'above' or 'below', though they could also be created to the left or right. For an overview of the way these commands work, see section 6.

9.1 Adding Connectable Elements

The most important kinds of additions chronos supports are those which can be connected to the timeline itself.

9.1.1 Timeline-Connectable Elements

`\chronoslife` {⟨*key-value list*⟩} *life*
macro

Create an element of type life. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary Ti*k*Z keys. At a minimum, `name` and `birth` must be specified for a living person. If the person is dead, both `birth` and `death` or `dates` should be given. If no date of death is specified, chronos assumes the person is living and uses the current date when placing the element on the timeline.

Table 5 summarises the chronos keys supported by elements of type life, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 6 for a typical case, but note that additional connectors require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronoslife{%
  name=leslie lamport,
  birth={1941-02-07},
  at=leslie lamport |- u1.north,
  connectors=east,
  tag anchor=west,
  xshift=10pt,
}
```

This will create a text node (text tag) named `tag leslie lamport` with two connectors, `10pt` to the right of coordinate `(leslie lamport |- u1.north)`, using the settings for life. The main connector, named `main connector leslie lamport` or `connector leslie lamport0`, will be at the Ti*k*Z anchor `west`. This will be used as the Ti*k*Z `anchor` when placing the node and used to connect it to the timeline. A second connector, named `connector leslie lamport1` will be created at the `east`, which may be used to connect the text tag to other elements.

A chronos connector, named `chronos connector leslie lamport` will be created on the timeline at the midpoint between `1941-02-07` and today's date. A line will also be marked on the timeline border, on the timeline or near the timeline, between these dates.

Note that the coordinate `leslie lamport` need not (and generally should not) exist when this command is given. of coordinates A coordinate of this name will be created on the timeline midway between the birth and death dates (or, in this case, between the birth date and today's date) prior to creation of the text tag. However, `u1` must exist. In this case, it refers to a node created using the `levels` option. `u1` is also known as `level 1` and refers the the first level above the timeline. Lamport will be a bit higher because the text tag's `west` anchor will be aligned with the `north` of node `level 1`.

Since the text tag is shifted right, the connection will be drawn using `|-` rather than `--`. If more complex paths are required, `connect=false` may be used and the text tag connected to the timeline manually. A chronos connector, `chronos connector leslie lamport`, would then be created on the timeline, as would the connectors on the text tag, but the connection itself would be omitted.

In addition, a colour named `colour leslie lamport` or `color leslie lamport` will be created. This is typically used in the styles responsible for the appearance of the text tag, line, connection

Table 5: Keys which are enabled (✓) and disabled (–) for tag contexts associated with chronos macros.

	Option	life	event	period	theory	theory circle	info	main	copyleft copyright
primarily per item configuration	name	✓	✓	✓	✓	✓	✓	✓	✓
	as is	✓	✓	✓	✓	–	–	–	–
	at	✓	✓	✓	✓	✓	✓	✓	✓
	tag anchor	✓	✓	✓	✓	–	✓	✓	✓
	colour | color	✓	✓	✓	✓	–	✓	–	–
	connect	✓	✓	✓	–	–	–	–	–
	connectors | connectors+ | connectors'	✓	✓	✓	✓	–	–	–	–
	place above	✓	✓	✓	✓	–	–	–	–
	place below	✓	✓	✓	✓	–	–	–	–
	dates	✓	–	✓	–	–	–	–	–
	date	–	✓	–	–	–	–	–	–
	birth	✓	–	–	–	–	–	–	–
	death	✓	–	–	–	–	–	–	–
	start	–	–	✓	–	–	–	–	–
	end	–	–	✓	–	–	–	–	–
	dates content	✓	✓	✓	–	–	–	–	–
	name content	✓	✓	✓	✓	–	✓	✓	✓
	text content	✓	✓	✓	✓	–	✓	–	–
	event year on line skip	–	✓	–	–	–	–	–	–
	caption	–	–	–	–	–	✓	–	–
	labels	–	–	–	–	✓	–	–	–
	circle texts	–	–	–	–	✓	–	–	–
	sizes	–	–	–	–	✓	–	–	–
	author	–	–	–	–	–	–	–	✓
	copyleft	–	–	–	–	–	–	–	✓
	notice	–	–	–	–	–	–	–	✓
	rotate	–	–	–	–	–	–	–	✓
	year	–	–	–	–	–	–	–	✓
primarily all-of-type-tag configuration	date format	–	✓	–	–	–	–	–	–
	date formats	✓	–	✓	–	–	–	–	–
	full dates	✓	✓	✓	–	–	–	–	–
	only years	✓	✓	✓	–	–	–	–	–
	show eras	✓	✓	✓	–	–	–	–	–
	without eras	✓	✓	✓	–	–	–	–	–
	only text	✓	✓	✓	–	–	–	–	–
	tag | tag+	✓	✓	✓	✓	–	✓	–	–
	connection | connection+	✓	✓	✓	✓	–	–	–	–
	line | line+	✓	✓	✓	–	–	–	–	–
	text tag | text tag+	✓	✓	✓	✓	–	✓	–	–
	default colour | color	✓	✓	✓	✓	–	✓	–	–
	colours | colors above	✓	✓	✓	✓	–	–	–	–
	colours | colors below	✓	✓	✓	✓	–	–	–	–
	colour | color rotation	✓	✓	✓	✓	–	–	–	–
	text tag yshift	✓	✓	✓	✓	–	–	–	–

Table 6: Components of elements of tag types life and period.

Element	Name	Description	TikZ Type
–	⟨*name*⟩	Point on timeline midway between ⟨*birth*⟩ and ⟨*death*⟩ (life) or ⟨*start*⟩ and ⟨*end*⟩ (period).	`coordinate`
line	–	Line or rectangle on or near timeline or timeline border from ⟨*birth*⟩ to ⟨*death*⟩ (life) or ⟨*start*⟩ to ⟨*end*⟩ (period).	`\path`
chronos connector	`chronos connector` ⟨*name*⟩	Connection point midway along line.	`node`
text tag	`tag` ⟨*name*⟩	Main box representing element. By default, contains dates above capitalised ⟨*name*⟩ (life) or capitalised ⟨*name*⟩ above dates (period).	`node`
main connector	`main connector` ⟨*name*⟩	Connection point at TikZ anchor of text tag.	`node`
connection	–	Line between the chronos connector and main connector.	`\draw`
connectors	`connector` ⟨*name*⟩n	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \ldots$.	`node`

Table 7: Components of an element of tag type event.

Element	Name	Description	TikZ Type
–	⟨*name*⟩	Point on timeline at ⟨*date*⟩.	`coordinate`
line	–	Line from timeline to the edge of timeline border at ⟨*date*⟩.	`\path`
chronos connector	`chronos connector` ⟨*name*⟩	Connection point at end of line.	`node`
text tag	`tag` ⟨*name*⟩	Main box representing element. By default, contains the date above the capitalised ⟨*name*⟩.	`node`
main connector	`main connector` ⟨*name*⟩	Connection point at TikZ anchor of text tag.	`node`
connection	–	Line between the chronos connector and main connector.	`\draw`
connectors	`connector` ⟨*name*⟩n	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \ldots$.	`node`

and connectors and may be referenced and reused later. If `simple colour names` or `simple color names` are used, it may also be referenced as `leslie lamport`.

`\chronosevent` {⟨*key-value list*⟩} *event*
macro

Create an element of type event. This is intended for events spanning no more than a day. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary TikZ keys. At a minimum, `name` and `date` should be specified.

Table 5 summarises the chronos keys supported by elements of type event, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 7 for a typical case, but note that additional connectors require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronosevent {%
  name=\emph{Common Sense},
  as is,
  yshift=5pt,
  date=1776,
```

```
    text=WildStrawberry,% will affect text for the element itself but not drawing,
filling or the assigned colour
    place below,% does nothing because the positive yshift pushes the element above the
timeline
  }%
```

Note the use of `as is` to prevent errors trying to capitalise `\emph`. `place below` has no effect here: the item still ends up above the timeline due to `yshift=5pt`. Note the use of only a year in `date`. If you only specify years, you probably want to configure your timeline to avoid printing full dates or you will end up with everything happening on January 1st. See section 8.2.2.

`\chronosperiod {`⟨*key-value list*⟩`}` *period*
macro

Create an element of type period. This is intended for extended events spanning more than one day. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary Ti*k*Z keys. At a minimum, `name` and `start` must be specified for an ongoing period. If the extended event has ended, both `start` and `end` or `dates` should be given. If no end date is specified, chronos assumes the period is ongoing and uses the current date when placing the element on the timeline.

Table 5 summarises the chronos keys supported by elements of type period, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 6 for a typical case, but note that additional connectors require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronosperiod {%
  dates={476-01-01}:{476-10-31},
  name=Fall of the\\Roman Empire,
  colour=blue,
  line+={draw=gray},% draw ugly grey border around line
}
```

This will construct an element analogous to the one created for Lamport. Note that the names of nodes and coordinates will be based on `Fall of theRoman Empire` because chronos will remove the `\\` and the capitalisation won't change. `colour Fall of theRoman Empire` will be `blue` and the line representing the period on the timeline will be drawn in `gray` but potentially filled in `blue`. This is because `line+` adds to any existing style rather than replacing it.

9.1.2 Adding Other Connectable Elements

Of the remaining elements, only those of type theory are connectable. While they cannot be connected to the timeline[31], chronos can create connectors for them to enable easy connections to other elements.

`\chronostheory {`⟨*key-value list*⟩`}` *theory*
macro

Create an element of type theory. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary Ti*k*Z keys. At a minimum, `name` must be specified, but `at` is required for placement. If left unspecified, chronos will place the theory at `chronos origin` and issue a warning.

Table 5 summarises the chronos keys supported by elements of type theory, with detailed usage information provided in sections 9.3 and 9.5.

[31] At least, chronos won't connect them for you.

Table 8: Components of an element of tag type theory.

Element	Name	Description	TikZ Type
–	⟨*name*⟩	Alias for text tag.	node
text tag	tag ⟨*name*⟩	Main box representing element. By default, contains the capitalised ⟨*name*⟩.	node
main connector	main connector ⟨*name*⟩	Connection point at TikZ anchor of text tag.	node
connectors	connector ⟨*name*⟩*n*	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \ldots$.	node

Table 9: Components of an element of tag type theory circle.

Element	Name	Description	TikZ Type
–	⟨*name*⟩	A (rectangular!) box containing all other components.	node
–	label above ⟨*name*⟩	Label above the ring.	nodes
–	label below ⟨*name*⟩	Label below the ring.	
–	⟨*name*⟩1	Centre of the ring.	coordinate

Creating the element involves constructing and naming components of up to two kinds. These are described in table 8 for a typical case, but note that a connector requires tag anchor or connectors to be set. Connectors may be rendered invisibly.

9.2 Adding Non-Connectable Elements

The remaining elements are non-connectable.

\chronostheorycircle {⟨*key-value list*⟩} *theory circle*
macro

Create a theory circle. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary TikZ keys. At a minimum, name must be specified, but at is required for placement.

Table 5 summarises the chronos keys supported by elements of type theory circle, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing and naming components of several kinds. Depending on the style, the element is intended to consist of a ring with text placed on the upper and lower semicircles and labels above and below. A symbol or picture can then be placed at the centre. The components are described in table 9 for a typical case, but note that these are style-dependant. In practice, this element could be used in other ways since it depends primarily on re-definable styles. However, in that case, there's no reason to avoid — and every reason to prefer — a new name.

For example,

```
  \chronostheorycircle{
    name=gutenberg revolution,
    at=chronos end |- printing press.center,
    sizes=15pt:9pt,
    circle texts=Gutenberg:Revolution,
    labels=15\textsuperscript{th}c.\thinspace \celabel:21\textsuperscript{st}c.\
thinspace \celabel,
  }
```

\chronosinfo {⟨*key-value list*⟩} *info*
macro

Create an element of type info i.e. an information box with a distinct caption. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary TikZ keys. At a minimum, name and at must be specified.

Table 10: Components of an element of tag type info.

Element	Name	Description	TikZ Type
–	⟨*name*⟩	Alias for text tag.	node
text tag	tag ⟨*name*⟩	Main box representing element. Empty by default.	node
caption	caption ⟨*name*⟩	By default, contains the capitalised ⟨*name*⟩.	node

Table 11: Components of an element of tag type main.

Element	Name	Description	TikZ Type
text tag	⟨*name*⟩	By default, contains the capitalised ⟨*name*⟩.	node

Table 5 summarises the chronos keys supported by elements of type info, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing and naming two components. These are described in table 10 for a typical case.

For example,

```
\chronosinfo{%
    name=syllogism,
    at=chronos year 200 |- u4,
    text content={All men are\\[-.25em]\hspace*{1.5em}mortal.\\Socrates is a\\[-.25em
]\hspace*{1.5em}man.\\$\therefore$ Socrates is\\[-.25em]\hspace*{1.5em}mortal.},
    anchor=north,
    caption=A Syllogism,
  }
```

Note the use of caption to override the default reuse of name. This allows the box to be captioned 'A Syllogism', while allowing references simply to syllogism.

`\chronosmaintitle` {⟨*key-value list*⟩} *main*
macro

Create the main title. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary TikZ keys. At a minimum, name and at must be specified.

Table 5 summarises the chronos keys supported by elements of type main, with detailed usage information provided in sections 9.3 and 9.5.

The result is simply a TikZ node, as described in table 11.

`\chronoscopyright` {⟨*key-value list*⟩} *copyleft, copyright*
macro

Create a copyleft or copyright notice. The ⟨*key-value list*⟩ should specify values for chronos keys and may include arbitrary TikZ keys. At a minimum, at should be specified to avoid a warning.

Table 5 summarises the chronos keys supported by elements of type copyleft and copyright, with detailed usage information provided in sections 9.4 and 9.5.

The result is simply a TikZ node, as described in table 12.

`\chronoscopyleft` {⟨*key-value list*⟩} *copyleft, copyright*
macro

Table 12: Components of an element of tag type copyleft and copyright.

Element	Name	Description	TikZ Type
text tag	⟨*name*⟩	By default, contains a standard copyright or copyleft notice utilising whatever details are provided or default values and dummy texts.	node

Create a copyleft notice. Sets `copyleft true` before passing `{`⟨*key-value list*⟩`}` to `\chronoscopyright`.

9.3 Additional Elements: Local Configuration

These keys are designed for use when creating specific elements. That is, they should be used in the argument of a chronos command such as `\chronoslife`, `\chronosevent`, `\chronosperiod`, `\chronostheory`, `\chronosinfo`, `\chronostheorycircle`, `\chronosmaintitle`, `\chronoscopyleft` or `\chronoscopyright`. If used globally (e.g. in `\chronosset` or the ⟨*chronos preamble*⟩), they will determine defaults for all elements (belonging to the relevant tag). Where this makes sense, the possibility is noted below; where it is not noted, global usage is unsupported.

`name` = ⟨*text*⟩ *life, event, period, theory, info, theory circle, main, copyleft, copyright*
key

The base name of the element. Except for `\chronosmaintitle`, `\chronoscopyleft` and `\chronoscopyright`, **this key is required**.

Default: `main title` (main)

Default: `copyleft` *and* `copyright` (copyleft and copyright)

By default, ⟨*text*⟩ is used multiple times.

First, it is capitalised and used for (part of) the content created for the element added to the timeline. `as is` prevents capitalisation. In the case of life, event and period, it is used for the non-date part of the content. In the case of theory and main, it is used for the whole content of the title. In the case of info, it is used to create the caption. In the case of copyleft and copyright, it is used as the author's name if `author` is unset. It is not used to create content in the case of theory circle.

Second, it is processed to create multiple names for different parts of the element e.g. names for connectorss , text tags etc. Processing attempts to remove some things which would be problematic when used as part of the names for coordinates and nodes, but markup can still cause problems. In this case, use `name content` or `text content` for the marked-up version and give ⟨*name*⟩ a suitably simplified version.

`as is` = `true|false` *life, event, period, theory*
boolean key

Whether to skip capitalisation of `name` if using it in the textual content of the element. If true, the `name` will *not* be capitalised; if false, it will be. Capitalisation is never used when setting the names of coordinates, nodes etc.

Default: `false`

`at` = ⟨*coordinate*⟩ *life, event, period, theory, info, theory circle, main, copyleft, copyright*
key

Where to place the element. This key is mandatory for theory circle, info, main, copyleft and copyright.

For life, event, period and theory, the key is optional. By default, the text tag will be placed at ⟨*name*⟩, which is a point on the timeline calculated according to date, offset vertically by either `yshift` or `text tag yshift`. Since theory text tags do not have dates, they are placed at the `(chronos origin)` and a warning is issued.

Example: `at=`⟨*name*⟩ `|- level -2`

This will align ⟨*name*⟩ horizontally with its placement point on the timeline and vertically with `level -2`, assuming at least two levels exist below the timeline. See section 8.6.

`tag anchor` = ⟨*node anchor*⟩ *life, event, period, theory, info, main, copyleft, copyright*
key

The PGF/Ti*k*Z anchor to use for the element's main connector. This is the point chronos uses to connect life, event and period text tags to the timeline. By default, this anchor is also used when placing the text tag. That is, `tag anchor` is used as the Ti*k*Z `anchor`. If you want different anchors to be used for the connection point and for placement, you can use both keys.

```
\chronoslife{%
  name=friedrich gottlob koenig,
  dates={1774-04-17}:{1833-01-17},
  at=friedrich gottlob koenig |- i1.north,
  tag anchor=east,
  anchor=north east,
  xshift=-5pt,
}
```

Default[for elements below the timeline]north Default[for elements above the timeline]south These defaults may be overridden on a per-tag basis by setting the key globally. For example,

```
\begin{chronos}[%
    life/tag anchor=50,
    event/tag anchor=north east,
    period/tag anchor=south,
  ]
\end{chronos}
```

`colour` = ⟨*colour name*⟩ *life, event, period, theory, info*

`color` *colour key*

The colour to assign to the element. The effect depends on the type of element being created and other settings. To modify the default colours, see sections 8.8 and 9.5.

`connect` = `true|false` *life, event, period*

boolean key

Whether to connect the element to the timeline.

Default: `true`

`connectors` = ⟨*list of node anchors*⟩ *life, event, period, theory*

`connectors+` `connectors'` *key*

Connection points to create on the element's text tag. Applies to life, event, period and theory. `connectors` and `connectors+` *add* to the existing list (if any). `connectors'` *replaces* it.

Default: dependent on other options (empty)

```
connectors={north,south,east,west},
connectors'={north},
connectors+={south},
connectors={east},
```

This code would result in connection points at the node's `north`, `south` and `east` anchors.

Note that one connection point is always created if the element is of a kind which could be connected to the timeline.

`default colour` Use the `default colour` assigned to elements of this tag type. *life, event, period, theory, info, main*

`default color` *key*

This key does something quite different if used in a global configuration context. See section 9.5 and section 8.8 for details. For example,

```
\begin{chronos}
  [
    life/colour rotation=true,
    life/default colour=gray,
  ]
  \chronoslife{% use colour from life's colours above colour list
    name=chris,
    dates={1038-01-10}:{1066:11-19},
    at=u2 -| chris,
  }
  \chronoslife{% use gray
```

```
    name=sandy,
    dates={1345-11-23}:{1378-12-24},
    at=u3 -| sandy,
    default colour,
  }
  \chronoslife{% use blue
    name=alex,
    dates={1246-09-22}:{1295-02-07},
    at=u5 -| alex,
    colour=blue,
  }
  \chronoslife{% use colour from life's colours below colour list
    name=hilary,
    dates={1156-06-12}:{1201-04-01},
    at=i4 -| hilary,
  }
\end{chronos}
```

Note the lack of an argument when used locally.

Note that there is no reason to use this key unless you wish to override colour rotation for a particular element. It suffices not to specify a `colour`.

`place below` = `true|false` *life, event, period, theory*

boolean key

By default, chronos alternates putting elements of a particular type above and below the timeline, but you may wish to put everything above or below, all elements of particular type above or below. Furthermore, you may wish to override the default for particular elements. Densely-packed timelines, especially, can require considerable intervention in order to make best use of the space while arranging things in a clear and (hopefully) visually appealing way.

```
  \chronosevent {%
    name=red letter day,
    date=1750,
    place below=false,
  }
```

Default: `true`

Initially: dependent on other options

`place above` *key* A convenience key equivalent to `place below=false`. *life, event, period, theory*

Thus the previous code could be rewritten as

```
  \chronosevent {%
    name=red letter day,
    date=1750,
    place above,
  }
```

`dates` = {⟨*birth date*⟩}:{⟨*death date*⟩} *life*

date key

={⟨*start date*⟩}:{⟨*end date*⟩} *period*

Dates of a life or period, specified as explained in section 8.2. The second date may be empty for a living person or ongoing occurrence. This key offers a more compact syntax as an alternative to the keys `birth` and `death` or `start` and `end` explained below. That is

```
  dates={1310-02-03}:{1350-06-07},
```

is equivalent to

```
birth={1310-02-03},
death={1350-06-07},
```

for life or

```
start={1310-02-03},
end={1350-06-07},
```

for period.

By default, these dates are used for both placement on the timeline and the date content of the element's text tag, but see `dates content`.

`birth` = {⟨*birth date*⟩} *life*
date key

The date of birth for a life, specified as explained in section 8.2. See `dates` above.

`death` = {⟨*death date*⟩} *life*
date key

The date of death for a life, specified as explained in section 8.2. See `dates` above.

`start` = {⟨*start date*⟩} *period*
date key

The start date of a period, specified as explained in section 8.2. See `dates` above.

`end` = {⟨*end date*⟩} *period*
date key

The end date of a period, specified as explained in section 8.2. See `dates` above.

`date` = {⟨*date*⟩} *event*
date key

The date of an event, specified as explained in section 8.2. By default, the date is used for both placement on the timeline and the date content of the element's text tag, but see `dates content`.

`event year on line skip` Don't put this particular event's year on the timeline. *event*
key

This can be used if the line would otherwise become too crowded when using `event years on line`. Cf. `special date`. See section 8.4.3. Figure 20 illustrates the effect of using this key.

`special date` = {⟨*text*⟩} *event*
key

Use ⟨*text*⟩ rather than the `date` for a particular event when using `event years on line`. Cf. `event year on line skip`. See section 8.4.3. Figure 20 illustrates the effect of using this key.

`dates content` = {⟨*text*⟩} *life, event, period*
key

Override the use of specified dates when creating content for the element's text tag. This is intended for 'special' cases e.g. uncertain, approximate or non-standardly specified dates. By default, the value is derived from `dates` or `date`.

Example: `dates content={c600-1450\,\celabel}`

`name content` = {⟨*text*⟩} *life, event, period, theory, info, main*
key

Override the use of the element's `name` when creating content for the element's text tag. This might be necessary if special markup is required. For example,

```
name content=\LaTeX3 Hummingbird,
```

It may also be desirable where longer content would render reuse of a `name` unwieldy.

`text content` = {⟨*text*⟩} *life, event, period, theory, info*
key

Override the use of both element's `name` and `dates` when creating content for the element's text tag.

```
  name=block printing,
  text content={Block printing, originally used to print pictures and text onto cloth,
developed into a method of printing books on paper.},
```

phantom = true|false *life, event, period*
boolean key

Create a 'phantom' element. Phantoms have assigned colours, require names and potentially feature lines, but they do not have text tags or connections. Note that these componentss are not invisible; *they are not constructed at all*.

Default: `true`

Initially: `false`

Example: `\chronosperiod{`⟨*name=c17,dates=1600:1699,colour=cyan,phantom*⟩`}`

This key may be used globally to set a different tag-specific default.

```
\begin{chronos}[%
    period/phantom,% make periods are phantoms by default
    event/phantom=true,% make events are phantoms by default
    life/phantom=false,% make lives non-phantoms by default (this matches the package
default)
  ]
\end{chronos}
```

For example, this key may be used to colour stretches of time without visibly labelling them, in conjunction with non-phantom lifes or events[32].

```
\begin{chronos}[% https://tex.stackexchange.com/a/701743/
    ...
    period={%
      phantom,
      colours below={orange,cyan,green,green},
    },
    ...
  ]
  % these **must** be named, even though they invisible, detached phantoms
  \chronosperiod{dates=2018:2019,name={n1}}
  \chronosperiod{dates=2019:2022,name={n2}}
  \chronosperiod{dates=2022:2023,name={n3}}
  \chronosperiod{dates=2023:2024,name={n4}}
  ...
\end{chronos}
```

caption = {⟨*text*⟩} *info*
key

The caption for an element of type info.

labels = {⟨*upper label*⟩}:{⟨*lower label*⟩} *theory circle*
key

Labels to be placed above and below a theory circle.

circle texts = {⟨*upper text*⟩}:{⟨*lower text*⟩} *theory circle*
key

The text to place in the upper and lower parts of a theory circle. By default, this uses `text effects along path`, so the content must be consistent with the restrictions imposed by use of this Ti*k*Z `decoration`.

sizes = ⟨*outer circle dimension*⟩:⟨*inner circle dimension*⟩ *theory circle*
dimension key

The sizes of the inner and outer circles used to create a theory circle.

[32] Based on my answer at TEX StackExchange: 701743.

Default: `15pt:9pt`

The difference between the two dimensions gives the thickness of the ring around which text is placed; the size of the inner circle gives the dimension of the hole in which a symbol or similar may be placed. This key may be used globally to set defaults.

```
\begin{chronos}[%
    theory/circles/sizes'+=10pt:5pt,
  ]
\end{chronos}
```

9.4 Additional Elements: Local/Global Configuration

Although you will generally want to use the following keys in the ⟨*chronos preamble*⟩ or in `\chronosset`, they can also be used to influence the format of a particular element.

⟨*tag*⟩`/date format` = {⟨*date format specification*⟩} *event*
date format key

Use ⟨*date format specification*⟩ to format date.

```
\chronosevent{%
  ...,
  date format={!a, !d !b},% show short day of week, day of month and short month
}
\end{chronos}
```

See section 8.2 for details and defaults.

⟨*tag*⟩`/date formats` = {⟨*date format spec.*⟩}:{⟨*date format spec.*⟩}:{⟨*date format spec.*⟩} *life, period*
date format key

Use ⟨*date format spec.*⟩s to format date range.

```
\chronosevent{%
  ...,
  date formats={!d}:{!d !B},% show day of month for start/birth date and day of month
and month name for end/death date
}
\end{chronos}
```

See section 8.2 for details and defaults.

`full dates` Show full dates. *life, event, period*
⟨*tag*⟩`/full dates`
key

```
\chronoslife{%
  ...,
  full dates,
}
\end{chronos}
```

See section 8.2 for details and defaults.

`only years` Show only years. *life, event, period*
⟨*tag*⟩`/only years`
key

```
\chronoslife{%
  ...,
  only years,% use only years in all dates
  event/full dates,% override to use full dates for events
}
\end{chronos}
```

See section 8.2 for details and defaults.

show eras Show eras. *life, event, period*
⟨*tag*⟩**/show eras**
key

```
\chronoslife{%
  ...,
  show eras,% show eras in all text tags
}
\end{chronos}
```

See section 8.2 for details and defaults.

without eras Omit eras. *life, event, period*
⟨*tag*⟩**/without eras**
key

```
\chronoslife{%
  ...,
  without eras,% omit eras in all text tags
  life/show eras,% override to show eras in life text tags
}
\end{chronos}
```

See section 8.2 for details and defaults.

only text Omit all date information. *life, event, period*
⟨*tag*⟩**/only text**
key

Default: disabled

```
\chronoslife{%
  ...,
  only text,% omit all dates from all tags
}
\end{chronos}
```

The following six sets of keys all work in the same way[33]. If used when creating a specific element, they affect that element. If set in the ⟨*chronos preamble*⟩ or `\chronosset` with a tag prefix, they set the tag-specific setting and will affect all elements belonging to that tag unless overridden locally.

Note these keys require *a* tag *prefix if used in a global context, such as the* ⟨chronos preamble⟩. *They do* not *need a prefix if used when creating a particular element.* For example,

```
\begin{chronos}
  [
    event/line+={semithick},% prefix required ; event/  explicit
  ]
  \chronosevent{%
    name=dydd dewi sant,
    date={1982-03-01},
    line+={double},% no prefix ; event/ implicit
  }
\end{chronos}
```

⟨*tag*⟩**/connection** = {⟨*key-value list*⟩} *life, event, period, theory*
⟨*tag*⟩**/connection+**
⟨*tag*⟩**/connection'**
key

⟨*key-value list*⟩ to apply to this element's connection. This affects the line drawn between the element's connector on the timeline and the text tag's main connector. This is intended for arbitrary Ti*k*Z keys; it should *not* be used for chronos keys as they may not be processed correctly.

[33] There is a seventh set, ⟨*tag*⟩`/tag`, ⟨*tag*⟩`/tag+ and` ⟨*tag*⟩`/tag'`, which may be of interest to advanced users. These keys are also potentially destructive. Not only ⟨*tag*⟩`/tag'`, but also ⟨*tag*⟩`/tag` and even ⟨*tag*⟩`/tag+`, can overwrite default settings for such things as colour rotation.

⟨*tag*⟩`/connection` and ⟨*tag*⟩`/connection'` replace any current list; ⟨*tag*⟩`/connection+` adds to it.

⟨*tag*⟩`/line` = {⟨*key-value list*⟩} *life, event, period*
⟨*tag*⟩`/line+`
⟨*tag*⟩`/line'`
key

⟨*key-value list*⟩ to apply to this element's line on or parallel to the timeline. This is the line representing the temporal extension of a life or period. This is intended for arbitrary Ti*k*Z keys; it should *not* be used for chronos keys as they may not be processed correctly. ⟨*tag*⟩`/line` and ⟨*tag*⟩`/line'` replace any current list; ⟨*tag*⟩`/line+` adds to it.

Default: `fill=##1,fill opacity=.25,draw=none` (on line, life/period)

Default: `draw=##1,fill=none,opacity=.25` (on line, event)

Default: `draw=##1,thick,fill opacity=.75` (off line, life/period)

Default: `draw=##1,draw opacity=.75,fill=none` (off line, event)

⟨*tag*⟩`/line yshift` = {⟨*dimension*⟩} *life, period*
dimension key

Default vertical displacement of lines from the timeline. Whether the displacement is reckoned from the centre or border of the timeline depends on the default placement.

`\lineyshift` The `line yshift`. This macro is available *only within the* ⟨*timeline specification*⟩.
macro

⟨*tag*⟩`/text tag` = {⟨*key-value list*⟩} *life, event, period, theory, info*
⟨*tag*⟩`/text tag+`
⟨*tag*⟩`/text tag'`
key

⟨*key-value list*⟩ to apply to this element's text tag. This is intended for arbitrary Ti*k*Z keys; it should *not* be used for chronos keys as they may not be processed correctly. ⟨*tag*⟩`/text tag` and ⟨*tag*⟩`/text tag'` replace any current list; ⟨*tag*⟩`/text tag+` adds to it.

```
\chronosset{%
  life/text tag+={font=\scshape\small},
  event/text tag+={font=\scshape\footnotesize},
  period/text tag+={font=\itshape\footnotesize},
}
```

See also ⟨*tag*⟩`/date font` and ⟨*tag*⟩`/text font`.

⟨*tag*⟩`/chronos connector` = {⟨*key-value list*⟩} *life, event, period*
⟨*tag*⟩`/chronos connector+`
⟨*tag*⟩`/chronos connector'`
key

Specify Ti*k*Z settings to be used when creating chronos connectors on the timeline. Note that ⟨*tag*⟩`/chronos connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

⟨*tag*⟩`/text tag connector` = {⟨*key-value list*⟩} *life, event, period, theory*
⟨*tag*⟩`/text tag connector+`
⟨*tag*⟩`/text tag connector'`
key

Specify Ti*k*Z settings to be used when creating text tag connectors on the timeline. Note that ⟨*tag*⟩`/text tag connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

⟨*tag*⟩`/main text tag connector` = {⟨*key-value list*⟩} *life, event, period, theory*
⟨*tag*⟩`/main text tag connector+`
⟨*tag*⟩`/main text tag connector'`
key

Specify *additional* Ti*k*Z settings to be used when creating the main connectors on text tags. ⟨*tag*⟩`/main text tag connector` and ⟨*tag*⟩`/main text tag connector'` replace any current list; ⟨*tag*⟩`/main text tag connector+` adds to it. The 'main' connector is the one which connects (or would connect) the text tag to the timeline. These keys are rarely needed because, usually, you want all the text tag connectors to look the same. Only use one of these three keys rather than one from the previous set if you *don't* want ⟨*key-value list*⟩ to apply to all of them. You do *not* need to duplicate settings here.

Note that ⟨*tag*⟩`/main text tag connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

⟨*tag*⟩`/label` = {⟨*key-value list*⟩} *info, theory circle*
⟨*tag*⟩`/label'`
⟨*tag*⟩`/label+`
key

Style to apply to the caption of an element of tag type info or the labels of an element of type theory circle. In the latter case, the style applies to both the upper and lower label.

Default: dependent on other options (empty)

`label` and `label'` replace the current list; `label+` replaces it.

⟨*tag*⟩`/title` = {⟨*key-value list*⟩} *main*
⟨*tag*⟩`/title'`
⟨*tag*⟩`/title+`
key

Style to apply to the main title, an element of tag type main.

Default: dependent on other options (empty)

`main/title` and `main/title'` replace the current list; `main/title+` replaces it.

⟨*tag*⟩`/title lines` *style*

Place main title between two parallel lines aligned to the width of the text. *main*

This style is available when creating a text tag of type main and draws lines along the northern and southern sides of the node. It is used in `somewhat plain` and `date centric`.

⟨*tag*⟩`/author` = {⟨*text*⟩} *copyleft, copyright*
key

The author's name for a copyleft or copyright notice. This is used only if `name content` is unset.

Default: `Author` (as a last resort)

If `author` and `name content` are unset, chronos first tries to figure out a suitable author. If `name` is set, a capitalised version is used. Otherwise, if `\svnauthor` is defined, `\svnFullAuthor{`⟨`\svnauthor`⟩`}` is used, if `\svnFullAuthor` is available, or `\svnauthor`, if it is not. If chronos still hasn't found an author, `Author` is used.

⟨*tag*⟩`/copyleft` = `true|false` *copyleft, copyright*
boolean key

Whether a copyleft or copyright notice should specify copyleft or copyright.

Default: `false` (`\chronoscopyright`)

Default: `true` (`\chronoscopyleft`)

`\chronoscopyright` respects the global default, so if you set ⟨*tag*⟩`/copyleft true` with `\chronosset`, both macros will make copyleft notices unless overridden in the ⟨*key-value list*⟩ of options they absorb when executed. `\chronoscopyleft` always creates a copyleft notice, regardless of any global settings, unless `copyleft` is explicitly set `false` when invoked.

⟨*tag*⟩`/notice` = {⟨*macro definition*⟩} *copyleft, copyright*
key

Template for a copyleft or copyright notice. It is used as the definition of the macro used for the content of the notice and should absorb two arguments: year and author.

Default: `{Copyleft \textcopyleft{} #1 #2}` (if ⟨*tag*⟩`/copyleft` is `true`)

Default: `{Copyright \textcopyright{} #1 #2}` (if ⟨*tag*⟩`/copyleft` is `false`)

For example,

```
\begin{chronos}
  [
    copyright/notice={Created by #2 in the year #1 of the Great Debacle at the behest of
 His Gracious Grasp Full Acre Fanfare the Nineteenth.},
  ]
```

⟨*tag*⟩/rotate = ⟨*angle*⟩ *copyleft, copyright*
key

The angle to rotate the node containing a copyleft or copyright notice.

Default: 90

⟨*tag*⟩/year = ⟨*text*⟩ *copyleft, copyright*
key

The year of publication for a copyleft or copyright notice.

Default: `\svnyear` (if available)

Default: `\today` (otherwise)

9.4.1 Specialist Fonts for Text Tags

⟨*tag*⟩/date font = {⟨*font commands*⟩} *life, event, period*
key

Set font macros to be applied to the date content of text tags.

Default:

```
\chronosset{%
  ...
  event/date font=\itshape\bfseries\small,
  life/date font=\sffamily\large,
  period/date font=\upshape\normalsize\mdseries,
}
```

Note that if you want to alter the font for the entire contents of the text tag, it is better to just use ⟨*tag*⟩/text tag+=font={⟨⟩}. Use date font to modify those settings specifically for date(s). Note that if era label are included, they will not be affected.

⟨*tag*⟩/text font = {⟨*font commands*⟩} *life, event, period*
key

Set font macros to be applied to the text content of text tags.

Default:

```
\chronosset{%
  ...
  event/text font=\uishape\large,
  life/text font=\sffamily\Large,
  period/text font=\small\bfseries,
}
```

Note that if you want to alter the font for the entire contents of the text tag, it is better to just use ⟨*tag*⟩/text tag+=font={⟨⟩}. Use text font to modify those settings specifically for names.

9.5 Additional Elements: Global Configuration

Except where otherwise noted, the keys in this section should not be used locally. The following keys are intended for use in the ⟨*chronos preamble*⟩ or in `\chronosset`. They are not intended for use when creating particular elements. For example, `default colour` should *not* be used for particular elements, unless you wish to *use* the existing default, as opposed to setting it. Instead, use `colour` to override default settings.

See section 8.8 for further information about colour keys and colour list keys.

life = {⟨*key-value list*⟩} *life, event, period, theory*
event
period
theory
key

Equivalent to prefixing each item in ⟨*key-value list*⟩ with ⟨*tag*⟩.

```
\begin{chronos}
  [
    life={%
      full dates,
      without eras,
      text tag+={font=\sffamily},
      text font=\bfseries,
      date font=\small,
      colours above={red,orange,blue},
      colours below={darkgray,gray,black,magenta},
    },
    period={%
      only years,
      text tag+={opacity=.75},
    },
    event={%
      text tag+={double=blue},
    },
  ]
\end{chronos}
```

⟨*tag*⟩/default colour = ⟨*colour name*⟩ *life, event, period, theory, info*
⟨*tag*⟩/default color
colour key

The default colour to use for all elements of type ⟨*tag*⟩, as explained in section 8.8. *This key does something quite different if used when creating a specific element. See section 9.3 for details.* For example,

```
\begin{chronos}[
  life/default colour=blue,
  event/default colour=green,
  period/default colour=red,
]
\end{chronos}
```

See section 8.8 for details and defaults.

colours above = {⟨*colour list*⟩} *life, event, period, theory*
colors above
⟨*tag*⟩/colours above
⟨*tag*⟩/colors above
colour list key

The default and tag-specific colour lists for all susceptible elements above the timeline. *These keys should never be used when creating specific elements.*

```
\begin{chronos}[
  colours above={gray,blue,green},
  life/colours above={magenta,pink,purple},
]
\end{chronos}
```

See section 8.8 for details and defaults.

colours below = {⟨*colour list*⟩} *life, event, period, theory*
colors below
⟨*tag*⟩/colours below
⟨*tag*⟩/colors below
colour list key

The default and tag-specific colour lists for all susceptible elements below the timeline. *These keys should never be used when creating specific elements.*

```
\begin{chronos}[
  colours below={red,orange,magenta},
  theory/colours below={black,gray},
]
\end{chronos}
```

See section 8.8 for details and defaults.

`colour rotation` = `true|false` *life, event, period, theory*
`color rotation`
⟨*tag*⟩`/colour rotation`
⟨*tag*⟩`/color rotation`
boolean key

Whether colour rotation is enabled by default.

Default: `true`

```
\begin{chronos}[
  colour rotation=false,
]
\end{chronos}
```

See section 8.8 for details and defaults.

`copyleft` = {⟨*key-value list*⟩} *copyleft, copyright*
`copyleft'`
`copyleft+`
`copyright`
`copyright'`
`copyright+`
key

Style to apply to the copyleft or copyright, an element of tag type copyleft / copyright.

Default: dependent on other options (empty)

`copyleft`, `copyleft'`, `copyright` and `copyright'` replace the current list; `copyleft+` and `copyright+` replace it.

`event dates split` = `true|false` *event*
boolean key

Create two text tags for each event, one above and one below the timeline. The formatted `date` or `dates content` goes into one and the formatted `name` or `name content` goes into the other. *This key has no effect on* text tags *belonging to other* tags*, such as* life *or* period*.*

Default: `true`

Initially: `false`

`event date split` Additional style applied to text tags of type event if `event dates split` is `true`. *event*
style

This style is provided primarily for use *outside* the `chronos` environment, in case you want some timelines with split events and some without. It is *not* intended to support both split and unsplit events on the same timeline.

Default: dependent on other options (empty)

The next twelve sets of keys fall into two groups, corresponding to the five sets of corresponding keys explained in section 9.4. ***None of these keys should be used when creating specific elements.***

The first set of six consists of plural forms, as opposed to the singular forms used for tag-specific configuration. These are available in the ⟨*chronos preamble*⟩ and `\chronosset`.

`text tags` = {⟨*key-value list*⟩} *life, event, period, theory, info*
`text tags+`
`text tags'`
key

Set or modify the global default ⟨*key-value list*⟩ to be applied to text tags in the absence of a tag-specific setting (section 9.4). `text tags` and `text tags'` replace the current value; `text tags+` replaces it.

Default: `outer sep=0pt,text=#1!75!black`

The key are passed a single argument specifying the current element's assigned colour, which may be used in the usual way i.e. by writing `#1` everywhere you would like the colour to be used.

Note that, when checking if a more fine-grained value is set, *the lists of* ⟨*key-value*⟩ *pairs are regarded as a whole. They are* not *treated on a* ⟨*key*⟩*-by-*⟨*key*⟩ *basis.* So if you write

```
\begin{chronos}
  [
    event/text tag={},
    text tags+={fill=green},
  ]
\end{chronos}
```

you will *not* get green text tags for events. Nor will you get the package option default. Instead, no style whatsoever will be applied when creating event text tags.

`connections` = {⟨*key-value list*⟩} *life, event, period, theory*

`connections+` `connections'` *key*

Set or modify the global default ⟨*key-value list*⟩ to be applied to connections in the absence of a tag-specific setting (section 9.4). `connections` and `connections'` replace the current value; `connections+` replaces it.

Default: `draw=#1`

These keys are related to the tag-specific ⟨*tag*⟩`/connection`, ⟨*tag*⟩`/connection+` and ⟨*tag*⟩`/connection'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to ⟨*tag*⟩`/text tag`, ⟨*tag*⟩`/text tag+` and ⟨*tag*⟩`/text tag'`. Please see above for details.

`lines` = {⟨*key-value list*⟩} *life, event, period*

`lines+` `lines'` *key*

Set or modify the global default ⟨*key-value list*⟩ to be applied to lines in the absence of a tag-specific setting (section 9.4). `lines` and `lines'` replace the current value; `lines+` replaces it.

Default: none (see section 9.4 for tag-specific defaults.)

These keys are related to the tag-specific ⟨*tag*⟩`/line`, ⟨*tag*⟩`/line+` and ⟨*tag*⟩`/line'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to ⟨*tag*⟩`/text tag`, ⟨*tag*⟩`/text tag+` and ⟨*tag*⟩`/text tag'`. Please see above for details.

`chronos connectors` = {⟨*key-value list*⟩} *life, event, period, theory*

`chronos connectors+` `chronos connectors'` *key*

Set or modify the global default ⟨*key-value list*⟩ to be applied to chronos connectors in the absence of a tag-specific setting (section 9.4). `chronos connectors'` replaces the current value; `chronos connectors` and `chronos connectors+` replace it.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

These keys are related to the tag-specific ⟨*tag*⟩`/chronos connector`, ⟨*tag*⟩`/chronos connector+` and ⟨*tag*⟩`/chronos connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to ⟨*tag*⟩`/text tag`, ⟨*tag*⟩`/text tag+` and ⟨*tag*⟩`/text tag'`. Please see above for details.

`text tag connectors` = {⟨*key-value list*⟩} *life, event, period, theory*

`text tag connectors+` `text tag connectors'` *key*

Set or modify the global default ⟨*key-value list*⟩ to be applied to text tag connectors in the absence of a tag-specific setting (section 9.4). `text tag connectors'` replaces the current value; `text tag connectors` and `text tag connectors+` replace it.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

These keys are related to the tag-specific ⟨*tag*⟩`/text tag connector`, ⟨*tag*⟩`/text tag connector+` and ⟨*tag*⟩`/text tag connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to ⟨*tag*⟩`/text tag`, ⟨*tag*⟩`/text tag+` and ⟨*tag*⟩`/text tag'`. Please see above for details.

`main text tag connectors` = {⟨*key-value list*⟩} *life, event, period, theory*

`main text tag connectors+` `main text tag connectors'` *key*

Set or modify the global default ⟨*key-value list*⟩ to be applied to main text tag connectors in the absence of a tag-specific setting (section 9.4). `main text tag connectors'` replaces the current value; `main text tag connectors` and `main text tag connectors+` add to it.

Default: dependent on other options (empty)

These keys are related to the tag-specific ⟨*tag*⟩`/main text tag connector`, ⟨*tag*⟩`/main text tag connector+` and ⟨*tag*⟩`/main text tag connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to ⟨*tag*⟩`/text tag`, ⟨*tag*⟩`/text tag+` and ⟨*tag*⟩`/text tag'`. Please see above for details.

The next six sets of keys are convenience keys which set or modify the global defaults and the corresponding keys for all tags at once.

`every text tags` = {⟨*key-value list*⟩} *life, event, period, theory, info*
`every text tags+`
`every text tags'`
key

A convenience key equivalent to setting the same ⟨*key-value list*⟩ for all of `text tags`, `life/text tag`, `event/text tag`, `period/text tag`, `theory/text tag` and `info/text tag` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every connections` = {⟨*key-value list*⟩} *life, event, period, theory*
`every connections+`
`every connections'`
key

A convenience key equivalent to setting the same ⟨*key-value list*⟩ for all of `connections`, `life/connection`, `event/connection`, `period/connection` and `theory/connection` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every lines` = {⟨*key-value list*⟩} *life, event, period*
`every lines+`
`every lines'`
key

A convenience key equivalent to setting the same ⟨*key-value list*⟩ for all of `lines`, `life/line`, `event/line` and `period/line` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every chronos connectors` = {⟨*key-value list*⟩} *life, event, period, theory*
`every chronos connectors+`
`every chronos connectors'`
key

A convenience key equivalent to setting ⟨*key-value list*⟩ for all of `chronos connectors`, `life/chronos connector`, `event/chronos connector`, `period/chronos connector` and `theory/chron connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every text tag connectors` = {⟨*key-value list*⟩} *life, event, period, theory*
`every text tag connectors+`
`every text tag connectors'`
key

A convenience key equivalent to setting the same ⟨*key-value list*⟩ for all of `text tag connectors`, `life/text tag connector`, `event/text tag connector`, `period/text tag connector` and `theory/text tag connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every main text tag connectors` = {⟨*key-value list*⟩} *life, event, period, theory*
`every main text tag connectors+`
`every main text tag connectors'`
key

A convenience key equivalent to setting the same ⟨*key-value list*⟩ for all of `main text tag connectors`, `life/main text tag connector`, `event/main text tag connector`, `period/main text tag connector` and `theory/main text tag connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every theory circle circle` = {⟨*key-value list*⟩} *theory circle*
`every theory circle circle'`
`every theory circle circle+`
key

Configuration of the base ring for elements of tag type theory circle. The ring consists of two circles with the smaller forming a hole in the centre by default. Changing or deleting the filling rule will eliminate the hole.

Default: `fill=`⟨*chronos main colour*⟩`, draw=`⟨*chronos main colour*⟩`, even odd rule`

`every theory circle circle` and `every theory circle circle+` add to the current ⟨*key-value list*⟩; `every theory circle circle'` replaces it.

`every theory circle text` = {⟨*key-value list*⟩} *theory circle*
`every theory circle text'`
`every theory circle text+`
key

Style applied to the texts used in constructing elementsof tag type theory circle. By default the texts are placed along the semicircular paths corresponding to the upper and lower halves of the ring formed by the theory circle circles. This means the colour used here should differ from that used to fill the circles, given the default styles.

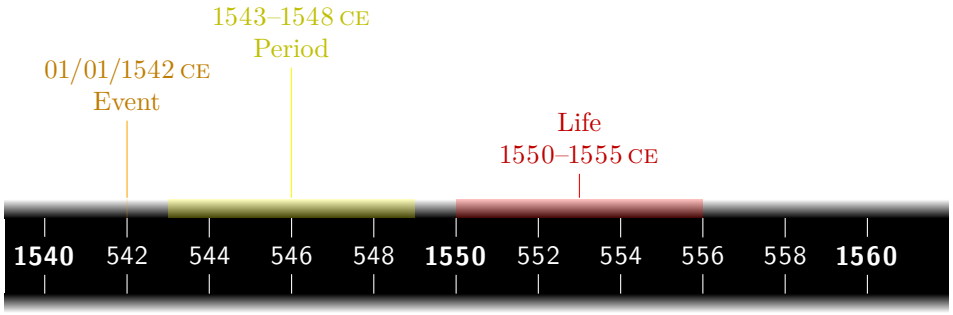
Default: `decoration={text effects along path, text={##1}, text effects/.cd, fit text to path, text=chronos@prifliw@cefndir, characters={text along path, font=\scriptsize\sc decorate`

`every theory circle text` and `every theory circle text+` add to the current ⟨*key-value list*⟩; `every theory circle text'` replaces it.

`text tag yshift` = ⟨*dimension*⟩ *life, event, period, theory*
dimension key

The `yshift` to apply when placing the `text tag` if `yshift` is otherwise `0pt` and `at` is unset. You should probably never use this key in the context of a particular element, because `yshift` works just as well and will probably be more reliable and certainly faster. Moreover, unlike `yshift`, which can be used to adjust a position set with `at`, `text tag yshift` cannot. If `at` is used, `text tag yshift` is ignored. It makes sense to set this globally if you want all elements or all elements belonging to a particular tag to be shifted by some specified distance from the timeline. For example,

```
\begin{chronos}[
  life/text tag yshift=10pt,
  event/text tag yshift=30pt,
  period/text tag yshift=50pt,
  theory/text tag yshift=70pt,
]
\end{chronos}
```



Theory

The following keys take the form {`every`} ⟨*tag*⟩, optionally followed by prime or plus. *They should not be used to configure elements for which other global keys exist, such as* colours, connections, connectors, date formats, lines *or* text tags. Generally, these keys should be unnecessary and are best avoided, although they may occasionally be convenient.

`every life` = {⟨*key-value list*⟩} *life*
`every life'`
`every life+`
key

Additional configuration for all elements of tag type life. These do much the same as `life/tag`, `life/tag+` and `life/tag'`, but should *never* be used when creating a specific element. `every life` and `every life+` add to the current ⟨*key-value list*⟩; `every life'` replaces it.

`every event` = {⟨*key-value list*⟩} *event*
`every event'`
`every event+`
key

Additional configuration for all elements of tag type event. These do much the same as `event/tag`, `event/tag+` and `event/tag'`, but should *never* be used when creating a specific element. `every event` and `every event+` add to the current ⟨*key-value list*⟩; `every event'` replaces it.

`every period` = {⟨*key-value list*⟩} *period*
`every period'`
`every period+`
key

Additional configuration for all elements of tag type period. These do much the same as `period/tag`, `period/tag+` and `period/tag'`, but should *never* be used when creating a specific element. `every period` and `every period+` add to the current ⟨*key-value list*⟩; `every period'` replaces it.

`every theory` = {⟨*key-value list*⟩} *theory*
`every theory'`
`every theory+`
key

Additional configuration for all elements of tag type theory. These do much the same as `theory/tag`, `theory/tag+` and `theory/tag'`, but should *never* be used when creating a specific element. `every theory` and `every theory+` add to the current ⟨*key-value list*⟩; `every theory'` replaces it.

`every info` = {⟨*key-value list*⟩} *info*
`every info'`
`every info+`
key

Additional configuration for all elements of tag type info. These do much the same as `info/tag`, `info/tag+` and `info/tag'`, but should *never* be used when creating a specific element. `every info` and `every info+` add to the current ⟨*key-value list*⟩; `every info'` replaces it.

9.6 Adding Connections, Using Colours and Accessing Styles

To access the colour list used for the timeline etc., see sections 8.3 and 8.4.5. For details of the way colour list are assigned to elements, see section 8.8.

Life, event, period and theory elements are designed to be connected not only, in the case of those which are connectable, to the timeline, but also to each other. To ensure consistent styling, this requires the use of chronos styles in Ti*k*Z commands.

In addition, densely-packed timelines sometimes require non-standard paths be used to connect a minority of elements to the timeline in an efficient way. Again, this requires access to chronos styles.

`chronos connect` = {⟨*tag*⟩}:{⟨*element name*⟩} *life, event, period, theory*
style

This sets the style used for connections belonging to elements of type ⟨*tag*⟩ with the colour assigned to ⟨*element name*⟩ (section 8.8). For example,

```
\draw [chronos connect=life:johannes gutenberg] (connector johannes gutenberg) -- (
connector printing press) (connector johannes gutenberg2) -|- (connector movable type) (
connector johannes gutenberg3) -- ++(5pt,0pt) |-| (connector gutenberg bible);
```

This will draw a line using the style for connections of tag type life and the colour assigned to the element named `johannes gutenberg`. Note the use of connectors on both the element's own text tag and on other elements' text tags. In this case, `tag johannes gutenberg` is being connected to `tag printing press`, `tag movable type` and `tag gutenberg bible`.

The following four keys provide analogous access to the styles and colour list used for chronos connectors, text tag connectors, lines and text tags and are used in the same way.

`chronos create chronos connector` = {⟨*tag*⟩}:{⟨*element name*⟩} *life, event, period*
style

This sets the style used for chronos connectors belonging to elements of type ⟨*tag*⟩ with the colour assigned to ⟨*element name*⟩.

`chronos create text tag connector` = {⟨*tag*⟩}:{⟨*element name*⟩} *life, event, period, theory*
style

This sets the style used for text tag connectors belonging to elements of type ⟨*tag*⟩ with the colour assigned to ⟨*element name*⟩.

`chronos mark line` = {⟨*tag*⟩}:{⟨*element name*⟩} *life, event, period*
style

This sets the style used for lines (on or near the timeline) belonging to elements of type ⟨*tag*⟩ with the colour assigned to ⟨*element name*⟩.

`chronos text tag` = {⟨*tag*⟩}:{⟨*element name*⟩} *life, event, period, theory, info*
style

This sets the style used for text tags belonging to elements of type ⟨*tag*⟩ with the colour assigned to ⟨*element name*⟩.

We can also use the colour assigned to `johannes gutenberg` directly. Perhaps, for example, we'd like to put a book symbol near this element in the appropriate colour.

Example: `\node [colour johannes gutenberg, above left=5pt and 10pt of tag johannes gutenberg.north west, anchor=south east, inner sep=0pt] {⟨book-symbol⟩};`

10 Drawing on Chronos Layers

See section 6.4.

`on chronos background layer` *style*

Apply to a `scope` to draw everything inside on layer `chronos background`.

```
\begin{scope}[on chronos background layer]
  \node {Something};% in between the regular background and chronos middle ground
\end{scope}
```

`on chronos middle ground layer` *style*

Apply to a `scope` to draw everything inside on layer `chronos middle ground`.

```
\begin{scope}[on chronos middle ground layer]
  \node {Something};% behind the main layer and chronos background
\end{scope}
```

`on chronos foreground layer` *style*

Apply to a `scope` to draw everything inside on layer `chronos foreground`.

```
\begin{scope}[on chronos foreground layer]
  \node {Something};% above the main layer but behind chronos overlay.
\end{scope}
```

`on chronos overlay layer` *style*

Apply to a `scope` to draw everything inside on layer `chronos overlay`.

By default, chronos puts only debugging information on `chronos overlay`, which means drawing on this layer should always draw on top of anything constructed by the package code.

```
\begin{scope}[on chronos overlay layer]
  \node {Something over everything else.};
\end{scope}
```

In addition, chronos never puts anything on the non-chronos PGF/Ti*k*Z `background` layer and it would be difficult to persuade it to do so without rewriting internal code. Drawing on *this* layer, therefore, is almost guaranteed to end up behind everything constructed by the package code[34].

```
\begin{scope}[on background layer]% fill area below the timeline
  \fill [blue!25!white] (chronos pre |- chronos bottom) rectangle (chronos post-foot);
\end{scope}
```

11 Externalising Chronos Timelines with Memoize

As explained in section 15, `chronos timelines` cannot be externalised with PGF/Ti*k*Z's external. Since PGF/Ti*k*Z, in general, and chronos, in particular, can be rather slow to compile, this is serious issue. If you only have a two or three small timelines, the compilation time will be negligible. But if you have a large, densely packed timeline or many timelines, compilation time will quickly become excessive.

Fortunately, `chronos` environments *can* be externalised. Moreover, they can be externalised more conveniently, more robustly and more securely, without the need for a separate compilation for each `chronos`. This means compilation is only a little slower when the timelines are being compiled (whereas compilation would be far slower with the external pgf/ti\emphkz library, even if it worked) and subsequent compilations are fast.

[34] Unless nefarious TEXnicians have interfered with your installation. It is even quite unlikely a bug would cause this kind of problem, though bugs will doubtless cause many others.

Sašo Živanović's memoize has no trouble compiling this documentation and externalising its timelines. Memoize is a little more trouble to set up initially than the external pgf/ti\emphkz library, but requires far less fine-tuning once configured.

To externalise** chronos timelines**, you must first setup memoization as explained in** memoize**'s documentation*[35]*.

Chronos supports automemoization out-of-the-box[36]: to enable automatic memoization of `chronos` environments, simply load memoize early in your preamble. Chronos will then enable 'automemoization' for all timelines[37].

All chronos styles (except default) and all colour schemes (except `default`) are defined so that modification will automatically trigger the recompilation of all chronos timelines which use them.

12 Deferring Code

If you don't know why you might want to use the keys in this section, you don't need to use them.

`timeline config` = {⟨*code*⟩}
`timeline config'`
`timeline config+`
key

Execute additional ⟨*code*⟩ after chronos has processed the keys at the start of the `chronos` environment, but before further processing the resulting configuration and constructing the timeline. These keys are provided primarily for use in chronos style definitions, but may occasionally be useful elsewhere. `timeline config` and `timeline config+` add to the current code; `timeline config'` replaces it. Note that `timeline config'` is destructive: it obliterates any existing code chronos has installed, which may be entirely unrelated to the code now being stored. Chronos style authors should never use this form. Even if the code is for purely private use in a locked room with no internet access, you should stick to the additive forms unless your memory is infallible *and* you always remember to use it.

12.1 Additional Ti*k*Z

Generally, you can mix arbitrary Ti*k*Z code freely into the body of the `chronos` environment. For example, this is how to add connections between text tags or to decorate your timeline with symbols or ornaments.

However, sometimes you might want to add something *after* `chronos` has finished. You might, for example, want to do something after the frame is drawn or place something relative to headings or subheadings Two sets of keys are provided for this purpose. One set enable you to execute arbitrary Ti*k*Z code within the picture's bounding box; the other enables you to do so outside. Generally, it is the first set you will want to use; the second are useful in a narrower range of cases and for debugging purposes.

`chronos tikz'` = {⟨*TikZ commands*⟩}
`chronos tikz`
`chronos tikz+`
key

Commands to execute after the ⟨*timeline additions specification*⟩ and any frame, headings and subheadings are drawn, but before debugging information is added (see section 14). `chronos tikz` and `chronos tikz+` add to current material; `chronos tikz'` replaces it. Material added with these keys is included in the final picture's bounding box. If you draw outside the frame and outer border, for example, the final bounding box expands to accommodate it. *If you aren't sure which set of keys to use, choose these.*

[35] By default, memoize uses `perl` and requires the installation of a couple of libraries. If you use Linux or have `python` already installed, I'd recommend using this method as it requires only a single extra library, is faster and more robust. If you do not wish to use either `perl` or `python`, you can use TEX, but I have not personally tested this method as it is slower and less secure.

[36] This fantastic feat was accomplished by copying a line of code from memoize's manual and substituting `chronos` for the appropriate word. Even I managed to achieve this without major incident.

[37] Of course, memoization can be disabled permanently or temporarily for some or all timelines. See memoize's documentation for details or look at the code for this document, which disables memoization for fig. 1 to prevent destruction of hyperlinks.

`chronos tikz outside bb'` = {⟨*TikZ commands*⟩}
`chronos tikz outside bb`
`chronos tikz outside bb+`
key

Commands to execute after the ⟨*timeline additions specification*⟩ and any frame, headings and subheadings are drawn, but before debugging information is added (see section 14). `chronos tikz outside bb` and `chronos tikz outside bb+` add to current material; `chronos tikz outside bb'` replaces it. Material added with these keys is excluded when the final picture's bounding box is determined. If you draw outside the frame and outer border, for example, TEX will treat it as if it didn't exist and you will need to ensure adequate space is available to accommodate it manually. *If you aren't sure which set of keys to use, avoid these.*

Finally, you might want to add material at some specific point in the construction of the picture (e.g. after headings but before the frame). The following sets of keys facilitate such additions.

`before headings'` = {⟨*TikZ commands*⟩}
`before headings`
`before headings+`
key

Commands to execute after the ⟨*timeline additions specification*⟩, but before constructing any headings. `before headings` and `before headings+` add to current material; `before headings'` replaces it.

`before drawing frame'` = {⟨*TikZ commands*⟩}
`before drawing frame`
`before drawing frame+`
key

Commands to execute after the ⟨*timeline additions specification*⟩ and any headings and subheadings are drawn, but before constructing any frame. `before drawing frame` and `before drawing frame+` add to current material; `before drawing frame'` replaces it.

13 Custom Schemes and Styles

The macros and keys explained in this section enable you to define custom colour schemes and chronos styles. These may then be used in the same way as those provided by chronos (section 7).

Customisation is a two-stage process. Chronos styles should not define colours definable by colour schemes.

Colour schemes are straightforward to define; chronos styles are a bit trickier.

13.1 Defining Chronos Colour Schemes

As explained in section 7.2, in addition to the default colours, chronos currently provides `blues`, `contninety`, `cronoleg`, `lavender`, `modern`, `offlinebasic`, `offlinealt`, `sobriety` and `xcolseries`[38] (table 2). `xcolseries` demonstrates the use of xcolor colour series in chronos colour lists. `contninety`, `modern`, `offlinebasic` and `offlinealt` illustrate the use of colour schemes to support chronos styles which require minimal modifications of other colour schemes.

New colour schemes should follow the examples in chronos-lib-colschemes.sty[39]. For instance, here's the code to set up `blues`:

```
\chronosnewcolourscheme{blues}{% chronos-lib-colschemes.sty
  timeline foreground=DodgerBlue4,
  timeline background=DodgerBlue2,
  default below={Cerulean!50!DodgerBlue4,Cerulean!50!DodgerBlue3,Cerulean!50!DodgerBlue2,
Cerulean!50!DodgerBlue1,Cerulean},
  default above={Cerulean!50!DodgerBlue4,Cerulean!50!DodgerBlue3,Cerulean!50!DodgerBlue2,
Cerulean!50!DodgerBlue1,Cerulean},
  foreground=DodgerBlue4,
```

[38] Note that `xcolseries` uses the `hsb` colour model, which is not supported by PGF/Ti*k*Z. If loading this set of colours directly, add `/utils/exec=` to `chronos`'s optional argument. This is not necessary if loading a chronos style which utilises `xcolseries`.In either case, all colours in the current `chronos` environment will be converted to `rgb`.

[39] For historical reasons, `cronoleg` is non-standardly defined as it was the default scheme during most chronos development. The current implementation of this scheme is officially internal. The implementation — as opposed to the scheme — is highly likely to change in backwards-incompatible ways without notice. This warning does not apply to *usage* of the colour scheme, but you should not take it as a model for a new scheme, except to pass it as an option to `\chronosnewcolourscheme`.

```
  background=white,
}
```

This is intended for 'off line' timelines so it doesn't include colours for a timeline border, though chronos will derive such colours anyway, as explained below.

There are two pitfalls in defining a colour scheme. First, definitions cannot utilise other chronos colours at this stage. You cannot, therefore, define the middle border colour, for example, in terms of the outer and inner colours.

Second, scheme names must consist of letters only as they are used to create new macros.

`\chronosnewcolourscheme` `[`⟨*existing scheme*⟩`]{`⟨*name*⟩`}{`⟨*key-value list*⟩`}`
macro
`\chronosnewcolorscheme` `[`⟨*existing scheme*⟩`]{`⟨*name*⟩`}{`⟨*key-value list*⟩`}`
macro

If ⟨*existing scheme*⟩ is specified, it should be the name of an existing colour scheme; otherwise, a default set of colours is loaded. ⟨*name*⟩ is the name of the new colour scheme and must be a unique string of alphabetic characters suitable for use in a macro name. ⟨*key-value list*⟩ is a list of key-value pairs from the list in table 13.

Schemes need not use all keys[40]. It is sufficient to specify the required deviations from ⟨*existing scheme*⟩. For example, here's the code to set up `offlinealt`,

```
\chronosnewcolourscheme[cronoleg]{offlinealt}{%
  timeline foreground=blue!40,
}
```

13.1.1 How Colour Schemes are Processed

When a colour scheme is loaded, chronos processes the settings in six stages.

1. The specified ⟨*existing scheme*⟩ or defaults are loaded.
2. Keys for the 'core' colours `foreground` and `background` are set and flipped to provide default settings for the 'core derivative' colours `timeline foreground` and `timeline background`.
3. Keys for the 'core derivative' colours `timeline foreground` and `timeline background` are set and the resulting four colours used to derive default settings for the 'core border' colours `timeline border inner`, `timeline border middle` and `timeline border outer`. In particular, `timeline border inner` is set to match `timeline background`, `timeline border outer` is set to `background` and `timeline border middle` is set to a 50-50 mix of the two.
4. Keys for the 'core border' colours `timeline border inner`, `timeline border middle` and `timeline border outer` are set. The main `foreground` colour is assigned to the 'elemental' default colours `life/default`, `event/default`, `period/default`, `theory/default` and `info/default`.
5. Keys for the 'elemental' default colours `life/default`, `event/default`, `period/default` and `theory/default` are set.
6. *Much later*, after the user configuration for the `chronos` environment has been read, chronos potentially flips the 'core derivative' colours `timeline foreground` and `timeline background`. See section 13.2.

Only after this sixth stage are the 'public' names listed in table 14 assigned to the final set of colour scheme-definable colours.

[40] In fact, they need not use any, though a colour scheme which uses none would serve no purpose.

Table 13: Keys for `\chronosnewcolourscheme`. Note that neither 'colour' nor 'color' appears in any key.

Key	Expected Argument Type	Example
`foreground`	⟨*colour name*⟩	`chronosblack`
`background`	⟨*colour name*⟩	`chronoswhite`
`timeline foreground`	⟨*colour name*⟩	`chronosCerulean`
`timeline background`	⟨*colour name*⟩	`chronosDodgerBlue4!50!chronosblack`
`timeline border outer`	⟨*colour name*⟩	`chronoswhite`
`timeline border inner`	⟨*colour name*⟩	`chronosCerulean`
`timeline border middle`	⟨*colour name*⟩	`chronosDodgerBlue4!50!chronosblack`
`life/default`	⟨*colour name*⟩	`chronosDodgerBlue4`
`event/default`	⟨*colour name*⟩	`chronosDodgerBlue4`
`period/default`	⟨*colour name*⟩	`chronosDodgerBlue4`
`theory/default`	⟨*colour name*⟩	`chronosDodgerBlue4`
`info/default`	⟨*colour name*⟩	`chronosDodgerBlue4`
`default above`	⟨*list of colour names*⟩	`chronosRed, chronosOrange, chronosYellow, chronosGreen, chronosBlue, chronosMidnightBlue, chronosViolet`
`default below`	⟨*list of colour names*⟩	`chronosCerulean!50!chronosDodgerBlue4, chronosCerulean!50!chronosDodgerBlue3, chronosCerulean!50!chronosDodgerBlue2, chronosCerulean!50!chronosDodgerBlue1, chronosCerulean`
`life/above`	⟨*list of colour names*⟩	`chronosDeepPink2, chronosDarkOrange1, chronosFirebrick1, chronosPurple0, chronosWildStrawberry, chronosOrangeRed1, chronosDarkGoldenrod1, chronosDarkOrchid3`
`life/below`	⟨*list of colour names*⟩	`chronosDodgerBlue3, chronosGreen3, chronosBlue3, chronosSpringGreen4, chronosDeepSkyBlue2, chronosForestGreen, chronosPeriwinkle, chronosSeaGreen3`
`event/above`	⟨*list of colour names*⟩	`chronosThistle4, chronosThistle4!.5!chronosThistle3, chronosThistle3, chronosThistle3!.5!chronosThistle2, chronosThistle2`
`event/below`	⟨*list of colour names*⟩	`chronosSeashell4, chronosSeashell4!.5!chronosSeashell3, chronosSeashell3, chronosSeashell3!.5!chronosSeashell2, chronosSeashell2`
`period/above`	⟨*list of colour names*⟩	`chronosMistyRose4, chronosMistyRose4!.5!chronosMistyRose3, chronosMistyRose3, chronosMistyRose3!.5!chronosMistyRose2, chronosMistyRose2`
`period/below`	⟨*list of colour names*⟩	`chronosIvory4, chronosIvory4!.5!chronosIvory3, chronosIvory3, chronosIvory3!.5!chronosIvory2, chronosIvory2`
`theory/above`	⟨*list of colour names*⟩	`xcolor s2!![0],xcolor s2!![1],xcolor s2!![2],xcolor s2!![3],xcolor s2!![4],xcolor s2!![5],xcolor s2!![6],xcolor s2!![7],xcolor s2!![8],xcolor s2!![9],xcolor s2!![10],xcolor s2!![11], xcolor s2!![12],xcolor s2!![13],xcolor s2!![14],xcolor s2!![15]`
`theory/below`	⟨*list of colour names*⟩	`xcolor g2!![0],xcolor g2!![1],xcolor g2!![2],xcolor g2!![3],xcolor g2!![4],xcolor g2!![5],xcolor g2!![6],xcolor g2!![7],xcolor g2!![8],xcolor g2!![9],xcolor g2!![10],xcolor g2!![11],xcolor g2!![12],xcolor g2!![13],xcolor g2!![14],xcolor g2!![15]`

13.2 Defining Chronos Styles

The current method for creating chronos styles is straightforward in theory, but potentially hazardous in practice. Here's an example from chronos-lib-styles.sty.

```
\pgfqkeys{/chronos}{%
  blues below/.style={%
    /chronos/.cd,
    blues below/.meaning to context,
    colour scheme=blues,
    rotate all colours,
    timeline={%
      timeline years=above,
      timeline marks,
      timeline minor marks,
      step minor year=50,
      step divisions=10,
      step major year=100,
      dates=1550:2050,
      timeline height'=3pt,
      timeline line={chronos timeline foreground colour,double=chronos timeline
background colour,line width=\timelineht/3,double distance=\timelineht/3},
      timeline arrow,
      conditional timeline arrow={%
        timeline/timeline line+={Bar-Latex,shorten <=-\timelineht/3,shorten >=-3pt-2.1\
timelineht},
        timeline/timeline width-={3pt+2.43\timelineht},
        before headings+={\path (chronos post) -- ++(3pt+2.1\timelineht,0pt) (chronos
pre) -- ++(-\timelineht/3,0pt);},
      }{},
      timeline mark={chronos timeline foreground colour,line width=.6pt,shorten >=-4pt},
      timeline minor mark={chronos timeline foreground colour,line width=.5pt,shorten
>=-3.5pt},
      timeline bare mark={chronos timeline foreground colour,line width=.3pt,shorten
>=-2.5pt},
      timeline year={fill=none,text=chronos timeline foreground colour,rotate around
={45:(chronos year \chronosyeari |- chronos top)}},
      major step font=\sffamily\footnotesize\tlstyle,
      timeline years anchor=south west,
      minor step font=\sffamily\scriptsize\tlstyle,
      timeline margin'=17.5pt,
    },
    minor year format={!Y},
    every event below,
    every life below,
    every period below,
    levels=0:3,
    headings style+={text=chronos main colour!75!chronos main background colour,font=\
small\itshape\bfseries},
    subheadings style+={text=chronos main colour!75!chronos main background colour,font
=\footnotesize\itshape},
    main/title+={font=\LARGE,text=chronos timeline foreground colour,draw=chronos
timeline background colour,semithick},
    main/frame+={thick,draw,chronos timeline foreground colour,double=chronos timeline
background colour},
    copyright={font=\footnotesize\sffamily, inner sep=0pt, outer sep=0pt, text=chronos
timeline foreground colour!50!chronos main background colour},
    copyright/rotate=90,
    copyright/tag anchor=north west,
  },
}
```

This definition is chosen because it is one of the most technically complex examples. This complexity is a function of several factors: it uses *off-line* years; the year labels are rotated; the line involves two arrow tips; and the line is drawn with `double`.

Note the following:

1. colours listed in table 13 are used but not defined;
2. instead, a custom colour configuration is set by loading an appropriate colour scheme;
3. there is a weird looking `\chronosyeari` in the definition of `timeline year`;
4. `timeline/timeline arrow` and `timeline/conditional timeline arrow` enables use of arrow tips to be toggled off;
5. `dates` are defined, even though they are almost certainly wrong in most cases;
6. `.meaning to context` is used, even though the user might not have loaded memoize, which defines it.
7. some fonts use a non-standard command `\tlstyle`.

Item 7 need not concern us here. If certain packages are loaded, it ensures tabular, lining figures; if not, chronos provides a command with this name at the end of the preamble by simply `\let`ting it to `\upshape`.

Regarding item 5, the standard chronos styles all define `dates`, but whether they should do so is another question. On the one hand, if they are not defined (as they are not if no chronos style is loaded), chronos will generate an error, alerting the user to the deficiency. Since it is highly unlikely any default choice will suit any user, let alone most of them, an error might be considered appropriate. On the other hand, some chronos styles are far more suitable for some temporal ranges than others. For example, consider this excerpt from the definition of `contemporary 90`:

```
timeline={%
  timeline marks,
  timeline minor marks,
  timeline mark={ultra thick},
  timeline minor mark={thick},
  step divisions=4,
  step major years=2,
},
```

This is fine for a timeline of a decade or two, but quite unsuitable for one representing either the period 3,000 BCE–2025 CE or the first half of 1857. While a user can always modify these settings, along with the `dates`, a default range provides a sense of the temporal duration the chronos style is suitable for 'out-of-the-box'.

The author of this package has found a comfortable spot on a convenient fence and intends to stay there, whatever the provided chronos styles might suggest. The reader is warned to make the most of the fences available here, as there are none whatsoever in the next section.

13.2.1 How (Not) to Customise Colours

Items 1 and 2 are the most important. ***Chronos styles* MUST NOT *set core, core derivative or core border colours,* where 'core, core derivative and core border colours' refer to those listed in tables 13 and 14.** In many cases, violating this rule may appear to work, but in others doing so will produce weird results or errors.

Moreover, ***chronos styles* should not *set any other*** colour key ***or*** colour list ***directly***. In many cases, violating this rule may appear to work, but in others doing so will cause things not to work as expected.

To summarise, ***if it* can *be done by a*** colour scheme***, it* should *be done by a*** colour scheme[41].

[41] That is, 'can implies ought'.

Table 14: Keys and names for chronos colours. Note that neither 'colour' nor 'color' appears in any key in the first column, but in every key in the second. In the second column, 'color' may be substituted for 'colour' in any name.

		Colour Schemes Key	Later Accessible As	
MUST NOT define!	CORE	core: `foreground`	`chronos main colour`	CORE
		core: `background`	`chronos main background colour`	
		core derivative: `timeline foreground`	`chronos timeline foreground colour`	
		core derivative: `timeline background`	`chronos timeline background colour`	
		core border: `timeline border outer`	`chronos timeline border outer colour`	
		core border: `timeline border inner`	`chronos timeline border inner colour`	
		core border: `timeline border middle`	`chronos timeline border middle colour`	
Should NOT touch!	ELEMENTAL	default colours: `life/default`	-	
		default colours: `event/default`	-	
		default colours: `period/default`	-	
		default colours: `theory/default`	-	
		default colours: `info/default`	-	
		colour lists: `default above`	-	
		colour lists: `default below`	-	
		colour lists: `life/above`	-	
		colour lists: `life/below`	-	
		colour lists: `event/above`	-	
		colour lists: `event/below`	-	
		colour lists: `period/above`	-	
		colour lists: `period/below`	-	
		colour lists: `theory/above`	-	
		colour lists: `theory/below`	-	

The reason for this restriction is that the colours are not finalised and the public colour names are not defined when the colour scheme and/or chronos style are read. Initially, chronos assigns colours only to internal names. When the user configuration in the ⟨*chronos preamble*⟩ has been read, chronos starts the `tikzpicture` environment and further processes the configuration before drawing the timeline. As part of this processing, chronos makes changes to colours in specified circumstances.

In particular, the colours assigned to the timeline foreground and background are switched if three conditions are satisfied.

1. The internal colour names for `chronos timeline foreground colour` and `chronos timeline background colour` evaluate to the same colour specification.
2. One of the specifications is identical to the colour specification for `white`.
3. `timeline years` is not `on line`.

Condition 3 cannot be determined until the complete configuration has been read. In particular, it is not known when colour schemes and chronos styles are read. While it is recommended users select a chronos style congruent with their preferred setting for `timeline years`, this is intended to make configuration easier and is not a requirement.

Only *after* colours are potentially switched are the public names listed in table 14 assigned, long after colour schemes and chronos styles have been read.

It is nonetheless possible, indeed recommended, to *use* the public names in chronos styles, though they cannot be used in colour schemes. It is only *defining* them at this stage which is problematic.

Here is an example from the definition of `modern` in chronos-lib-styles:

✔
```
    timeline line={chronos timeline background colour, opacity=1},
    period/line={fill=chronos timeline foreground colour, blend mode=overlay},
    life/line={fill=chronos timeline foreground colour, blend mode=overlay},
    event/line={draw=chronos timeline foreground colour, thick, blend mode=overlay},
    every text tags={fill=chronos main background colour, text=####1, fill opacity=.75,
text opacity=1, draw=none, rounded corners, align=center, font=\sffamily\footnotesize},
```

This is perfectly proper[42]. However, if you were to include something such as

✘
```
  timeline border middle colour=chronos timeline border inner colour!50!chronos timeline
 border outer colour,
```

you would get an error complaining about the use of undefined colours. The definition of `timeline border middle colour` is the prerogative of the colour scheme and shouldn't feature in a chronos style at all, but this particular definition is illegitimate in any case because neither `chronos timeline border inner colour` nor `chronos timeline border outer colour` yet exists.

But why shouldn't chronos styles include colour definitions of the kind permitted in colour schemes? Because chronos processes the definitions in colour schemes as they are read (section 13.1.1). If you put

✘
```
  foreground=SlateBlue4,
  background=Snow1,
```

in a chronos style, *only* these colours will be set. In particular, neither the timeline nor any default colours will be affected at all. But if you put this into a colour scheme, chronos will derive colours for the timeline and set default colours for elements belonging to the various tags. If no other changes are made, the result will be a white-on-blue timeline with blue-to-white timeline borders and blue as the fallback colour for tag elements. (This is probably wrong for `off line` and chronos won't correct you because `Snow1` isn't exactly `white`, but that's why colour schemes should do either a bit more or a bit less than this.)

[42] At least, it is fine as far as chronos goes. Whether it is proper Ti*k*Z code is not for me to judge.

If you wish, your chronos style can load a colour scheme of its own. This is what many of the standard chronos styles do. For instance, here is the sum total of `modern`'s `modern` colour scheme,

✔
```
\chronosnewcolourscheme{modern}{%
  timeline foreground=chronosSilver,
}
```

13.2.2 How to Rotate Years

Item 3 is a function of this style's rotation of the year labels created for the timeline. The easiest way to do this is to `rotate around` one of the anchors belonging to the node containing the relevant year. Obviously, we can't do this for each node. We don't know how many there are or what they are named. Instead, we need a hook into the `\foreach` loop chronos uses when creating the year labels.

`\chronosyeari` *macro* refers to the current year *inside the* `\foreach` *loop used to mark years on the* timeline. `(chronos year \chronosyeari)` isn't actually the node, but the point representing the date on the timeline, but the node starts there, so we can use it provided `timeline years anchor` is set appropriately.

```
  timeline year={rotate around={45:(chronos year \chronosyeari |- chronos top)}},
  timeline years anchor=south west,
```

13.2.3 Hashes

You may have noticed the following line in the excerpt from `modern`'s definition above.

```
  every text tags={fill=chronos main background colour, text=####1, fill opacity=.75,
text opacity=1, draw=none, rounded corners, align=center, font=\sffamily\footnotesize},
```

Anywhere you'd normally use a single hash (e.g. #1) in defining a Ti*k*Z style, you need two (##1) because you're nesting that definition within the definition of another style. So it is not surprising to find lines such as

```
  connections={draw=##1, {Triangle[width=0pt 3,reversed,length=0pt 1.5]}-{Triangle[width
=0pt 5,reversed,length=0pt 2.5]}},
```

in `modern`'s definition, but why *four*?

Certain keys require one or more additional doublings of hashes. Anytime you use an `every` key, you need to double. Double double makes four, so we get `text=####1`[43].

Elsewhere, a single doubling is generally sufficient, as shown in these lines from the definition of `plain arrow`

```
    period/line+={line width=2pt,draw=##1},
    life/line+={line width=2pt,draw=##1},
```

Incidentally, PGF doesn't complain if you quadruple the hashes here, though it does so if you make the same mistake elsewhere. So silence does not always indicate correctness. This is important if you're debugging: don't assume because a pattern generates no error in one case, it cannot be the source of an error in another.

Note also that if you say

✘
```
    text tags={draw=####1,sharp corners,text opacity=1,fill opacity=1,draw opacity=1,
drop shadow},
```

[43] For real fun with hashes, may I recommend chronos or forest?

TEX will give you an error suggesting you haven't used *enough* hashes,

```
! Illegal parameter number in definition of \tikz@temp.
<to be read again>
                   1
l.113   ]

? h
You meant to type ## instead of #, right?
Or maybe a } was forgotten somewhere earlier, and things
are all screwed up? I'm going to assume that you meant ##.

?
```

If you double the hashes *again* (`########1`), you'll get the same error. The actual problem is that you've used too many.

✔
```
    text tags={draw=##1,sharp corners,text opacity=1,fill opacity=1,draw opacity=1,drop shadow},
```

is correct in a chronos style definition i.e. twice the number required in the ⟨*chronos preamble*⟩. If you reduce the hashes to one (`#1`), you'll get no error but the wrong output as the element's colour won't be used.

Despite this, chronos styles should always use chronos ***keys and hashes for colours.***

Hashes are essential for two reasons.

1. Hard-coding colours breaks colour rotation. In order for colours to be not just assigned in rotation, but used for the elements they are assigned to, chronos style definitions must use the colour names passed to them. So hashes are essential when defining the properties of tag elements subject to colour rotation.
2. Chronos ***cannot track colours it doesn't know about and it doesn't know about colours passed directly to PGF/TikZ keys.*** Hard-coding colours breaks the system of colour names chronos provides. Chronos will assign colour names to colours regardless, but the names will not refer to the colours actually used. They will merely refer to the colours assigned. Chronos styles are responsible for ensuring assigned colours are used so chronos colour names work correctly. Suppose a chronos style includes `event/text tag+={text=red},event/connection+={draw=red}`. Chronos will keep assigning colours to elements of tag type event, but it will not assign 'red' except by happy chance.

   Example: `\draw [chronos connect=period:red letter day] ...`

   will still work, but may well use black or navy blue rather than the pillar box red expected. Since this referencing system works for some elements not subject to colour rotation at all, such as those belonging to tag info and applies even when colour rotation is disabled completely, it constitutes a more general reason to avoid hard-coding colours, even if the effects may be less immediately noticeable in some timelines.

13.2.4 Timeline Arrow

Chronos styles must decide whether to support timelines with and/or without one or more arrow tips and/or line caps. In deciding this, note the following points.

- Only `off line` styles can support these features.
- Adding, removing or modifying a tip or cap requires adjusting the `timeline width`. This is because the length available for representing time is reduced when some proportion of the timeline line is used for a tip or cap. Chronos adjusts automatically for `timeline margins` and `timeline era margins`, but styles are responsible for other adjustments.

- Supporting both arrowed and non-arrowed variants therefore requires conditionalised code.
- Each arrow tip and line cap requires a bespoke adjustment, even if used in default form.
- Users may legitimately use `timeline/timeline arrow` and `timeline/no timeline arrow` after loading a chronos style.
- Chronos styles may legitimately ignore these keys.
- Chronos styles must delay finalising the content of `timeline` until the end of the ⟨*chronos preamble*⟩ if they wish to support variants with and without tips and/or caps.

See `timeline/timeline arrow` and `timeline/no timeline arrow`.

`timeline/conditional timeline arrow` *key*

= {⟨*=key-value list if arrow/cap*⟩}key-value list otherwise

This key expects two arguments: ⟨*key-value list if arrow/cap*⟩ should be a list of key-values to be executed if `timeline/timeline arrow` is `true`; ⟨*key value list otherwise*⟩ should be a list of key-values to be executed if it is `false`. Chronos will switch the key path to `/chronos/` prior to using the list, but the `timeline` prefix must be specified if required. The effect is to add code to the style `timeline/do timeline arrow` which executes ⟨*key-value list if arrow/cap*⟩ if `timeline arrow` is `true` and ⟨*key-value list otherwise*⟩ otherwise. More specifically, the code used to implement this mechanism is equivalent to

```
conditional timeline arrow/.code 2 args={%
  \pgfqkeys{/chronos}{%
    llinell amser/.cd,
    timeline@arrow/.style={/chronos/.cd,#1},
    no@timeline@arrow/.style={/chronos/.cd,#2},
    do timeline arrow/.add code={%
      \ifchronostimelinearrow
        \tikzset{/chronos/llinell amser/timeline@arrow}%
      \else
        \tikzset{/chronos/llinell amser/no@timeline@arrow}%
      \fi
    },
  }%
},
```

If the timeline uses off line yearss, `\pgfqkeys{`⟨*/chronos/timeline*⟩`}{`⟨*do timeline arrow*⟩`}` is executed after `timeline/timeline height` is finalised.

Example: See below.

`timeline/do timeline arrow` *key*

Chronos styles are expected to set this *via* `timeline/conditional timeline arrow`, which causes it to be executed in `timeline config`, but they could also execute it explicitly if required.

Default: dependent on other options

For example, `lines on line` supports arrowed and non-arrowed variants using

```
lines on line/.style={% https://tex.stackexchange.com/a/324453/
  /chronos/.cd,
  ...
  timeline={%
    timeline width'=120mm,
    ...
    timeline arrow,
    conditional timeline arrow={%
      timeline/timeline width'-=20mm,
      timeline/timeline line+={shorten >=-20mm, -{Triangle Cap[length=20mm]}},
      before headings+={%
        \path (chronos post) -- +(20mm,0pt);
```

```
      },
    }{},
  },
  ...
},
```

`timeline arrow` requests an arrow by default, but does nothing else. `conditional timeline arrow` sets up the style keys to execute if `timeline arrow` is still enabled when `do timeline arrow` is executed. At this stage, then, no actual changes are applied to the style to be applied to the timeline.

The actual effects on the timeline's style are determined only at the end of ⟨*chronos preamble*⟩ when `timeline/do timeline arrow` is executed. Hence, the user may override the style's use of `timeline arrow` by writing `timeline/timeline arrow=false` or `timeline/no timeline arrow` after loading `lines on line`.

Styles which support timeline arrows must do the following to ensure correct results[44].

1. Set `timeline/timeline arrow` if an arrow, non-default line-cap or similar is to be default.
2. Use `timeline/conditional timeline arrow` if a non-arrow is to be supported and configure the arrow/cap/spacer(s) *only* using this conditional.
3. Decrease `timeline/timeline width` by the total length of arrows, caps and spacers. At the beginning of the `chronos` environment, this dimension must equal the actual length available for the `timeline era margins`, `timeline margins` and the representation of time, else marks and years may be placed onto arrows or caps.

   The recommended way to do this at present is to

   (a) calculate the total length of arrows, caps and spacers by hand and use `timeline/timeline width'= {`⟨*total length*⟩`}` to subtract it from the user-specified `width`[45];

   (b) add `shorten >=` and/or `shorten <=`, as appropriate, to increase the length of the line just while it is being drawn.
4. Ensure the bounding box includes any arrows, caps and spacers.

   One way to achieve this is to

   (a) use `before headings+` to place coordinates at the tip and very tail of the arrow/-cap/spacer(s).
5. Calculations must account for `\pgflinewidth` and, if applicable, any use of `double`, in order to avoid overfull boxes.

13.2.5 Styles and Automemoization

It is recommended that chronos styles are configured so that externalised `chronos` timelines which use them are automatically recompiled if the styles' definitions change. This can be achieved by adding ⟨*name of style*⟩`/.meaning to context` to each chronos style's definition. For example, the packaged styles all use the following template to begin their definitions.

```
\pgfqkeys{/chronos}{%
  <name of style>/.style={%
    /chronos/.cd,
```

[44] This is necessary because

`chronos` (environment) discards the bounding box which includes the arrows immediately after drawing them and it is not possible (as far as I can tell) to extract the required information, even though PGF has just performed all these calculations itself.

[45] Accurate calculation requires knowledge of `\pgflinewidth`, any use of `double`, custom options passed to the arrow and details of the formula PGF uses to calculate the length for the specific types of arrow tips and/or line caps configured. In some cases, this information is included in the Ti*k*Z manual but, in most cases, you must consult the source of the arrows.meta pgf/ti\emphkz library.

```
    <name of style>/.meaning to context,
    ...
  },
}
```

This is safe, even if memoize isn't used, because chronos provides a fallback key handler, `.meaning to context` which does nothing.

13.3 Defining Styles for Additional Elements

Due to the way chronos manages tag contexts, creating custom styles to apply to the additional elements explained in section 9 is not necessarily straightforward.

If you only want to use non-chronos keys in your style, however, it *is* straightforward. Simply create whatever PGF/Ti*k*Z styles you wish and add them to particular elements as you deem appropriate.

The trouble starts if you want to define style which include chronos keys. More particularly, difficulties arise if you want to use keys which are specific to tag contexts such as `at` or `tag anchor`. For example, the timeline in fig. 1 uses three custom styles, `tag left`, `tag post` and `tag right` to place text tags. Consider the definition of `tag right`,

```
  at/.expand once=level -##1.south -| ##2,
  tag anchor=north west,
  anchor=south west,
  xshift=5pt,
  connectors=east,
```

It uses `at` and `tag anchor`, which are tag-specific chronos keys, as well as the `anchor` and `xshift` PGF/TikZ keys. A naïve approach would suggest

✘
```
  tag right/.style 2 args={%
  at/.expand once=level -##1.south -| ##2,
  tag anchor=north west,
  anchor=south west,
  xshift=5pt,
  connectors=east,
},
```

but this will fail. Less naïvely, you might fiddle with path prefixes, but this won't work reliably either because chronos effectively activates some tag-specific settings by installing them temporarily under `/chronos`. Meanwhile, it redefines a subset of both the global and tag-specific keys to ensure local element-specific settings don't 'leak'[46].

The result of all this is that you cannot generally use standard PGF/Ti*k*Z techniques to define styles involving chronos keys for use in creating chronos elements belonging to tags. Given the aims of chronos, this is a significant limitation only partially mitigated by the following workaround.

Chronos provides a PGF/Ti*k*Z key handler to facilitate the creation of straightforward styles, but the current version has significant limitations I've not been able to solve.

`.chronos key maker` = {⟨*key name*⟩}{⟨*pgf key handler*⟩}{⟨*value*⟩}
key handler

⟨*key name*⟩ should be a name suitable for a PGF/Ti*k*Z key. ⟨*pgf key handler*⟩ should be a PGF key handler, without the leading dot, such as `style 2 args` or `ecode`. ⟨*value*⟩ should be the value or definition for ⟨*key name*⟩. *Only handlers which expect a single argument may be used.* This limits the maximum number of arguments ⟨*key name*⟩ can absorb to two, since the only PGF key handlers capable of absorbing three or more arguments themselves require two or more.

[46] PGF/Ti*k*Z has this type of containment down to a fine art. Chronos's approach is altogether cruder.

The key handler is available in the ⟨*chronos preamble*⟩ and in `\chronosset`. It requires a single doubling of hashes.

Example:

Here are the definitions of `tag left`, `tag post` and `tag right` mentioned above.

```
tag right/.chronos key maker={tag right}{style 2 args}{%
  at/.expand once=level -##1.south -| ##2,
  tag anchor=north west,
  anchor=south west,
  xshift=5pt,
  connectors=east,
},
tag left/.chronos key maker={tag left}{style 2 args}{%
  at/.expand once=level -##1.south -| ##2,
  tag anchor=north east,
  anchor=south east,
  xshift=-5pt,
  text tag+={align=right},
},
tag post/.chronos key maker={tag post}{style}{%
  at=level -##1.south -| chronos end,
  tag anchor=north west,
  anchor=south east,
  connect=false,
  connectors=east,
},
```

Note `tag post`'s use of the standard `coordinate chronos end` (fig. 3).

14 Debugging

Note that many keys in this section draw `on chronos overlay layer`*. They will typically draw* ***over*** *content you've created.* This should not be a concern as they are not intended for use in the final document.

`placeholders` = `on`|`off`
choice key

If enabled, any helper nodes created with `levels` will be visible rather than invisible[47] and vertical lines corresponding to headings will be drawn. This option is intended to assist in the creation of complex timelines.

Default: `on`

Initially: `off`

`placeholder lines` = {⟨*key-value list*⟩}
style

The style used to draw any lines created when `placeholders` is enabled. The style may be modified or replaced using the usual Ti*k*Z techniques, but the settings for nodes should not be altered in a way which changes their size e.g. by setting `line width` or similar.

```
  \begin{chronos}
    [
      placeholders,
      placeholder lines/.append style={thick},% for the default nodes and similar lines,
 but thicker
      placeholder lines/.style={thin,draw=magenta,<->},% for magenta double-arrowed
lines with no changes to nodes
    ]
```

[47] I am grateful to Qrrbrbirlbel for providing the code implementing this at TEX StackExchange: 694967.

```
\end{chronos}
```

Default: `help lines, every node/.append style=rotate=-90,anchor=south,pos=.25,inner sep=0pt`

The following were created for use in developing the package, but some may be more generally useful. Those which seem most likely to be helpful are listed first.

Note that all of the keys which follow ignore the picture's bounding box. This means they will disappear (or partially disappear) with no warning if there is insufficient space. This may be a concern, but having half the timeline disappear from view is worse.

`show coords` = `true|false`
boolean key

Labels a selection of chronos coordinates, which may be useful for placement or trouble-shooting purposes.

Default: `true`

Initially: `false`

`show bounding box` = `true|false`
boolean key

Draws the bounding box of the `tikzpicture` containing the timeline.

Default: `true`

Initially: `false`

`show nodes` = `true|false`
boolean key

If, and only if, `timeline mark eras` is explicitly enabled (as opposed to being enabled just because a timeline spans BCE and CE), draws and labels the nodes containing the era labels on the timeline.

Default: `true`

Initially: `false`

`debug`
key

A convenience key which switches on all four of the options above.

```
\begin{chronos}
  debug,
\end{chronos}
```

The following keys are available to customise the output of the options in this section.

`show coordinate colour` = ⟨*colour name*⟩
`show coordinate color`
colour key

Default: `red`

`show bb colour` = ⟨*colour name*⟩
`show bb color`
colour key

Default: `green`

`show node colour` = ⟨*colour name*⟩
`show node color`
colour key

Default: `blue`

`show coordinate`
style

A style used to show coordinates. It is used both directly and indirectly by both `show coord` and `show node coord`. If you want to redefine it, it should take 5 arguments: a colour name, an angle, the name of the coordinate, a dimension and a (possibly empty) key-value list.

Default: `fill=#1, circle, anchor=center, inner sep=1pt, text=#1, pin={[#1, inner sep=0pt, pin edge={draw=#1}, pin distance=#4, #5]#2:#3}`

`show coord`
style

A style used to show coordinates. If you want to redefine it, it should take 2 arguments: the

name of the coordinate and an angle.

Default: `/chronos/show coordinate={`⟨*chronos show coordinate colour*⟩`}{#1}{#2}{30pt}{}`

`show node coord style` A style used to show particular points on nodes. If you want to redefine it, it should take 2 arguments: the name of the coordinate and an angle.

Default: `/chronos/show coordinate={`⟨*chronos show node colour*⟩`}{#1}{#2}{30pt}{}`

`\chronosshowcolour` `[`⟨`\macroname`⟩`]{`⟨*colour name*⟩`}` *macro*
`\chronosshowcolour*` `[`⟨`\macroname`⟩`]{`⟨*colour name*⟩`}` *macro*
`\chronosshowcolor` `[`⟨`\macroname`⟩`]{`⟨*colour name*⟩`}` *macro*
`\chronosshowcolor*` `[`⟨`\macroname`⟩`]{`⟨*colour name*⟩`}` *macro*

Extract the colour specification of ⟨*colour name*⟩ to the macro ⟨`\macroname`⟩. The starred forms show ⟨`\macroname`⟩; the remainder merely (re)define it. In case it is not obvious, don't use a ⟨`\macroname`⟩ you care about as it will be overwritten without warning. By default, an internal macro is used and reused, so, if you don't specify ⟨`\macroname`⟩, you can only inspect one colour specification at a time.

Example: `\chronosshowcolour*{white}`

will show the colour specification of `white` on the terminal.

The remainder are unlikely to be helpful except in debugging chronos and no attempt has been made to render their output intelligible.

`\chronosshowpreset` *macro* Show non-default globalised options. This shows the properties[48] currently recorded as set by the user. This includes selected options set by chronos styles and options set with `\chronosset`, but not defaults set by chronos when loading. This list is used in deciding whether to change the current setting of an option during timeline configuration. For example, if a user specifically requests `off line` years with a `timeline height` of 50mm in white-on-blue, chronos won't override those settings. But if a user asks for `off line` years without specifying `timeline height` or changing the default colours, chronos will try to select something reasonable for `timeline height` and assume the user wants black-on-white rather than white-on-white.

The output of `\chronosshowpreset` is unlikely to prove especially enlightening unless debugging chronos. Here, for example, is the output when used at the start of a sample `chronos` environment,

```
  The sequence \l__chronos_gosod_seq is empty
> .
```

and right after the optional argument has been processed,

```
The sequence \l__chronos_gosod_seq contains the items (without outer braces):
>  {angor@blynyddoedd}
>  {timeline@years}
>  {@digwyddiad@llawn}
>  {@byw@llawn}
>  {@parhad@llawn}
>  {markeras}
>  {llinell}
>  {cysylltiad}
>  {llinell amser}
>  {border}.
```

So this user didn't specify any non-default settings in the document preamble or with `\chronosset`, but has either set or specified a chronos style which set various options for this particular `chronos` environment, which chronos should respect. Note that the output tells us nothing about what has

[48] Specifically, the contents of the expl3 sequence used to record the names of chronos properties.

been chosen, but only *that* an explicit choice has been made. For example, `markeras` means the user has decided eras should or should not be marked on the timeline, but does not tell us which.

`\chronosshowfeatures` [⟨*tag*⟩] *life, event, period, theory, info*
macro

Shows properties[49] assigned to either the current or ⟨*tag*⟩ context. Note that the output uses the original names for tags, which differ from those documented in this manual. `life, event, period, theory` and `info` correspond to `byw, digwyddiad, parhad, theori` and `gwybodaeth`.

Without an argument, the default list of properties is shown if the command is executed outside a tag context; otherwise, the list for the current context is shown. With an argument, the list of properties for ⟨*tag*⟩ is shown regardless of execution context.

There is no list of properties associated with tag main.

Here's the output from `\chronosshowfeatures` inside a `chronos` environment, but outside any tag context,

```
The property list \l__chronos_prop contains the pairs (without outer braces):
> {@tag} => {{,/chronos/troi lliwiau=false,/chronos/blynyddoedd yn
unig,/chronos/heb gyfnodau,/chronos/troi lliwiau=true}}
> {@cysylltwr@chronos} => {{coordinate}}
> {@cysylltwr@testun} => {{anchor=center,inner sep=0pt,outer
sep=0pt,circle, anchor=center, draw=none, fill=none, minimum
size=\pgflinewidth }}
> {@llinell} => {{}}
> {@testun} => {{fill=chronos main background colour, text=####1, fill
opacity=.75, text opacity=1, draw=none, rounded corners, align=center,
font=\sffamily \footnotesize ,draw=####1,sharp corners,text opacity=1,fill
opacity=1,draw opacity=1,drop shadow}}
> {@cysylltiad} => {{draw=##1, {Triangle[width=0pt 3,reversed,length=0pt
1.5]}-{Triangle[width=0pt 5,reversed,length=0pt 2.5]}}}.
```

and from `\chronosshowfeatures[event]`,

```
The property list \l__chronos_digwyddiad_prop contains the pairs (without
outer braces):
> {@cysylltwr@chronos} => {{coordinate}}
> {@cysylltwr@testun} => {{circle, anchor=center, draw=none, fill=none,
minimum size=\pgflinewidth }}
> {@testun} => {{fill=chronos main background colour, text=##1, fill
opacity=.75, text opacity=1, draw=none, rounded corners, align=center,
font=\sffamily \footnotesize ,draw=##1,sharp corners,text opacity=1,fill
opacity=1,draw opacity=1,drop shadow}}
> {@tag} => {{/chronos/blynyddoedd yn unig,/chronos/heb
gyfnodau,/chronos/troi lliwiau=true}}
> {@llinell} => {{draw=chronos timeline foreground colour, thick, blend
mode=overlay}}.
```

[49] Specifically, expl3 property lists.

Table 15: Public names for chronos internal macros defined locally within the ⟨*timeline specification*⟩.

Public name	Chronos internal name
\ceyearlabel	\chronos@yearce
\bceyearlabel	\chronos@yearbce
\celabel	\chronos@ce
\bcelabel	\chronos@bce
\timelineht	\chronos@height
\timelineborderht	\chronos@borderheight
\timelinewd	\chronos@width
\lineyshift	\chronos@llinell@yshift

Table 16: Public names for chronos internal macros defined if undefined at the end of the preamble.

Public name	Chronos internal name
\ceyearlabel	\chronos@yearce
\bceyearlabel	\chronos@yearbce
\celabel	\chronos@ce
\bcelabel	\chronos@bce

15 Compatibility

Chronos timelines cannot be externalised using Ti*k*Z's external pgf/ti\emphkz library[50].

Ti*k*Z's spy pgf/ti\emphkz library also appears to be incompatible.

Arrow tips and line caps from Ti*k*Z's arrows pgf/ti\emphkz library are not supported in `timeline`. Please use arrows.meta instead.

Chronos defines some commands without either marking them as internal or using a package-specific prefix. These commands are of the following kinds.

- They use Welsh rather than English (`\byw`, `\digwyddiad`, `\parhad`, `\gwybodaeth`, `\theori`, `\cylchtheori` and `\prifdeitl`). These all use `\NewDocumentCommand`. Should they already be defined, LaTeX 2$_\varepsilon$ will produce an error and existing definitions will not be overwritten.
- They are defined only locally within the ⟨*timeline specification*⟩. These provide local access to `chronos` internals and do not use a package-specific prefix for reasons of convenience. These macros are listed in table 15. *Note that some of these macros are also defined conditionally at the end of the preamble. The local definitions described here are unconditional.*
- They are 'throwaway', extremely temporary macros such as `\tempa`. These are used only very, very locally. Any macro which needs to retain its definition for more than a few lines uses a `chronos@` prefix unless it is a variable in a PGF `\foreach` loop.
- They are defined only if undefined at the end of the preamble, so existing definitions are maintained without warning or error. This applies to cases where either chronos uses a command if it is available (e.g. `\uishape`), but needs a fallback otherwise, or a public macro is made available as a convenience, if the user is not using the name already (e.g. `\celabel`). These macros are listed in tables 16 and 17.
- They are differently-named replacements for a subset of etoolbox macros and tests[51], which are defined only if they do not exist. If they already exist, chronos produces a warning and continues, hoping for the best. This set of macros is compatible with etoolbox, which chronos depends on for patching purposes.

[50] However, chronos pictures *can* be 'memoized'. Moreover, if memoize is loaded, chronos will set up 'automemoization' by default. See section 11.

[51] They are a response to advice not to mix expl3 and etoolbox. Since I'd originally thought it was better to use etoolbox functions than create a slew of wrappers for expl3 functions, these are the products of the resulting rewrite. Despite my best efforts, the dependency on etoolbox remains, but usage is confined to cases where expl3 does not offer equivalent functionality.

Table 17: Fallback definitions for macros undefined at the end of the preamble.

Functionality used if defined	Chronos fallback definition
\tlstyle	\let\tlstyle\upshape
\plstyle	\let\plstyle\upshape
\uishape	\let\uishape\itshape
\textui	\DeclareTextFontCommand{\textui}{\uishape}
\sishape	\DeclareRobustCommand\sishape{\itshape\scshape}
\textsi	\DeclareTextFontCommand{\textsi}{\sishape}

Table 18: Approximate replacements for etoolbox macros.

etoolbox	chronos expl3 wrapper
\ifundef	\IfFreeTF, \IfFreeT and \IfFreeF
\ifdef	\IfExistTF, \IfExistT and \IfExistF
\ifcsundef	\IfCSFreeTF, \IfCSFreeT and \IfCSFreeF
\ifcsdef	\IfCSExistTF, \IfCSExistT and \IfCSExistF
\undef	\Undefine
\csletcs	\CSletCS
\cslet	\CSlet
\ifboolexpr	\IfBooleanExprTF, \IfBooleanExprT and \IfBooleanExprF
bool	\LegacyBoolean
test	\CSFreeBoolean
\ifnumcomp	\IntCompareBoolean, \IfIntCompareTF, \IfIntCompareT and \IfIntCompareF

However, they may be incompatible with packages I'm unaware of or which are not yet published, in which case the warnings may prove informative. These macros are listed in table 18.

15.1 Compatibility with Code from TEX SE Answers

The CTAN release of chronos is not backwards compatible with versions published on TEX StackExchange. However, there are several methods you can use to update most timelines produced using code from answers there. Which approach is best depends on the specific case.

I suggest four possible approaches below. Of these, methods 1 and 2 are strongly recommended. The remaining methods 3(a) and 3(b) are for those keen for adventures in the typesetting hinterlands, desperate souls suffering in imminent-deadline hells and the perilously inquisitive with too much time on their hands. They are included because most of us, at one time or another, find ourselves in situations of the second type, even if we are too home-loving and incurious to dare the others.

Method 1: If you intend to develop work utilising code from TEX SE answers further, I strongly recommend taking the time to switch to the new key-value interface and `chronos` environment. This method is the most work, but also the most reliable and flexible. There is no guarantee that either of the alternative methods methods 3(a) and 3(b) will work or continue to work with future chronos releases. Method 2 is an option, but if you are actively developing a timeline, the flexibility of chronos should make things easier and provide options otherwise unavailable. If you put more work in and then find the code you have insufficient to your needs, you will only have delayed and expanded the task of updating.

Method 2: If you don't intend to develop existing timelines further, I strongly recommend not loading chronos, renaming any existing file to avoid conflicts and doing an ultra-simple update so existing documents load the renamed file. This is the simplest, most straightforward option. Why fix what ain't broke? If the code you have works and you're satisfied with the results, you need this package like a head needs an ache. The only thing you should do — and you really *should* do this — is rename any conflicting package you created locally. That is, if you've stuck code from an SE answer in a file named `chronos.sty`, I strongly recommend renaming it to, for example, `chronos-se.sty` to avoid conflicts. Then you can use chronos in new documents and just change the `\usepackage` invocation to chronos-se in old ones.

Method 3: If methods 1 and 2 aren't options — if, say, you want to use this package for a new timeline in a document with existing timelines and you don't have time to update those, then one of the following pairs of definitions *may* produce more-or-less the same output from existing or slightly modified code. Note that there is no guarantee this will work in any particular case or, if it does, that it will continue to work with future releases of chronos. It may, however, provide a quick-and-dirty fix if you are stuck.

(a) This requires minimal changes to existing code. You will need to modify existing timelines to use the `chronos` environment if they are currently in `tikzpicture` environments. Then place the following code *into the preamble* of your document:

```
\usepackage{chronos}
\makeatletter
% The following definitions **MUST** be in the premable.
% They will **NOT** work if placed after \begin{document}
% or before \usepackage{chronos}.
% BEGIN \chronosevent
\NewDocumentCommand \chronosevent { O {} m O {} +m D () { \chronos@testun@yshift } }
{%  #1 [<connection options>]
%   #2 {<date>}
%   #3  [<text tag options>]
%   #4 {<text>}
%   #5 (<yshift>)
  \digwyddiad{%
    date=#2,
    name=#4,
    yshift=#5,
    text tag+={#3},
    connection+={#1},
  }%
}
% END \chronosevent
% BEGIN \chronosperiod
\NewDocumentCommand \chronosperiod { O {} m O {} m O {} +m D () { \chronos@testun@yshift } }
{%  #1 [<line options>]
%   #2 {<start date>}
%   #3 [<connection options>]
%   #4 {<end date>}
%   #5 [<text tag options>]
%   #6 {<text>}
%   #7 (<yshift>)
  \parhad{%
    start=#2,
    end=#4,
    name=#6,
    yshift=#7,
    connection+={#3},
    text tag+={#5},
    line+={#1},
  }%
}
% END \chronosperiod
\makeatother
```

If you use this method, you *cannot* use the key-value versions of `\chronosevent` and `\chronosperiod`. Instead, you will need to use `\digwyddiad` for events and `\parhad` for periods when you wish to make use of the new features.

(b) Alternatively, update all existing environments to use `chronos` as explained in method 3(a), if re-

quired. Then replace every occurrence of \chronosevent and \chronosperiod with \chronoslegacyevent and \chronoslegacyperiod and place the following in your document preamble[52]:

```
\usepackage{chronos}
\makeatletter
% BEGIN \chronoslegacyevent
\NewDocumentCommand \chronoslegacyevent { O {} m O {} +m D () { \chronos@testun@yshift } }
{%  #1 [<connection options>]
%   #2 {<date>}
%   #3  [<text tag options>]
%   #4 {<text>}
%   #5 (<yshift>)
  \chronosevent{%
    date=#2,
    name=#4,
    yshift=#5,
    text tag+={#3},
    connection+={#1},
  }%
}
% END \chronoslegacyevent
% BEGIN \chronoslegacyperiod
\NewDocumentCommand \chronoslegacyperiod { O {} m O {} m O {} +m D () { \chronos@testun@yshift } }
{%  #1 [<line options>]
%   #2 {<start date>}
%   #3 [<connection options>]
%   #4 {<end date>}
%   #5 [<text tag options>]
%   #6 {<text>}
%   #7 (<yshift>)
  \chronosperiod{%
    start=#2,
    end=#4,
    name=#6,
    yshift=#7,
    connection+={#3},
    text tag+={#5},
    line+={#1},
  }%
}
% END \chronoslegacyperiod
\makeatother
```

This allows you to use \chronosevent and \chronosperiod with the key-value interface in new timelines.

16 Version History

16.1 0.8

First CTAN release.

16.2 Pre-0.8

Earlier versions were published informally on TEX StackExchange.

[52] The location isn't crucial in this case, provided the definitions are read before you use them and after chronos is loaded, but it is bad practice to define new commands in the body of documents.

Index

Features are sorted by kind. All references are to pages. ***Bold italics*** *indicate the main explanation of the macro, environment or key.* Upright numbers *refer to additional comments, discussion or examples*[53]. † *indicates an example.*

[53] I am grateful to David Carlisle and Ulrike Fischer for help with indexing at TEX StackExchange: 695555.

E

L

M

N

P