Dates (`dates.dtx`)

Charles Duan

May 3, 2023

Nearly every reference cited in a legal document must be dated.[1] *Hereinafter* supports a wide variety of date specifications, in view of the many types of legal documents and the frequency with which historical documents are cited in legal research.

Typical legal citation systems have specific rules about the form of date to use with different reference types. *Hereinafter* does not enforce such rules, instead using a single date syntax in all places where dates are used. As a result, it is left to the discretion of writers to choose, for example, whether to include an exact date or just a year for any reference.

Dates may be entered in a variety of formats, so long as the month is spelled out in letters. The key additional feature of *Hereinafter* dates is support for "qualifier words," namely phrases that precede the date and explain its relevance. Qualifier words are considered part of the date and thus may be included in any place where a date is allowed; indeed, qualifier words are required in certain contexts.

1 Data Model

A date contains the following components:

- A year. This is the only required component. The phrase "n.d." can be used for undated references. It may be a range.
- Qualifier words preceding the date and explaining its relevance to the reference. For example, "submitted for ratification Sept. 17, 1787" includes qualifier words "submitted for ratification" that explain the relevance of the given date. Typical qualifier words are verbs ending in "ed," though any text preceding a number or keyword will be considered part of the qualifier. Qualifier words are useful, for example, when citing to cases still in litigation (e.g., "cert. granted, ⟨*date*⟩").

[1] The main exceptions are for in-force codified statutes, assumed to be dated as of the time of the writer's publication, and well-known timeless works for which a date is immaterial, such as the Bible, Blackstone, Shakespeare, and the *Federalist* papers.

- A month and date number. They may also be ranges. The month can also be a season.
- Modifiers such as “c.” for circa, “A.D.,” “A.C.E.,” “B.C.”, and “B.C.E.”

2 Input Syntax

Dates are entered, generally as parameters to references, as follows:

⟨*date*⟩ := [⟨*qualifier*⟩] (⟨*date-range*⟩ ‖ ⟨*date-word*⟩)*
⟨*date-range*⟩ := ⟨*date-word*⟩ _ ⟨*date-word*⟩
⟨*date-word*⟩ := ⟨*modifier*⟩ ‖ ⟨*month*⟩ ‖ ⟨*day*⟩ ‖ ⟨*year*⟩

Examples:

```
2009
sept-oct 1980
adopted july 4 1776
c. 538 BC
```

In other words, a date consists of qualifier words followed by one or more tokens that may be either date words or ranges of date words.[2] Date words of letters are interpreted as months or modifier words, while date words of numbers are interpreted as days or years. Punctuation is ignored other than dashes, used for ranges. Modifiers such as “A.C.E.” and “n.d.” should be entered with no dots (i.e., `ACE`, `ace`, `nd`).

Alternately, a date beginning with `!` will be used unformatted.

`\hi@date@parse[`⟨*qualifier*⟩`]{`⟨*date*⟩`}{`⟨*callback*⟩`}`: Parses a date given in ⟨*date*⟩. This macro creates temporary variables for each of the components of a date, reads the date by words, assembles the formatted date, and runs ⟨*callback*⟩. The optional `[`⟨*qualifier*⟩`]` is added to the date if no other qualifier was present.

`\hi@date@last` stores the last temporary variable where something was added. This is used when a hyphen is encountered, to put the range symbol and the next value into the right place.

```
\newcommand*\hi@date@parse[3][]{%
    \find@start{!}{#2}{#3}{%
        \def\hi@date@orig{#2}%
        \let\hi@date@qual\@empty
        \let\hi@date@pre\@empty
        \let\hi@date@day\@empty
        \let\hi@date@month\@empty
        \let\hi@date@year\@empty
        \let\hi@date@post\@empty
        \let\hi@date@last\relax
        \hi@date@readqual{#2}%
        \ifx\hi@date@qual\@empty \def\hi@date@qual{#1}\fi
        \hi@date@assemble{#3}%
    }%
}
\make@find@start{!}
```

Tests if date variable #1 is unset. If so, sets it to #2 and sets `\hi@date@last` to #1. Otherwise runs callback #3. This will be used throughout the macros below.

```
\def\hi@date@set#1#2#3{%
    \@test\ifx#1\@empty\fi{%
```

[2] Technically by this syntax, a range can consist of two unrelated date words—⟨*month*⟩-⟨*day*⟩ for example. However, the parser may interpret that construction in unexpected ways.

```
        \def#1{#2}\def\hi@date@last{#1}%
    }{%
        #3%
    }%
}
```

Reads a date, with the assumption that unknown words are part of the qualifier. #1 is the text remaining to read.

```
\def\hi@date@readqual#1{%
    \find@word{#1}%
        {\@tworun{\appto\hi@date@qual}{\hi@date@readqual}}% punct
        {\@tworun{\appto\hi@date@qual}{\hi@date@readqual}}% group
        {\hi@date@qualword} % word
        {\hi@date@nodateerr}% end
}
```

Warns that only qualifier words were found in a date, and no actual date components.

```
\def\hi@date@nodateerr{%
    \PackageWarning\hi@pkgname{%
        In date `\hi@date@orig', no date was found.\MessageBreak
        The date will not be parsed. If this is \MessageBreak
        correct, include an exclamation point \MessageBreak
        at the start of the date. This occured%
    }%
}
```

Reads one word with the assumption that it could be a qualifier. #1 is the word, #2 the remaining text to read.

```
\def\hi@date@qualword#1#2{%
    \hi@date@dateword{#1}{%
        \hi@date@readnoqual{#2}%
    }{%
        \appto\hi@date@qual{#1}%
        \hi@date@readqual{#2}%
    }%
}
```

Reads text with the assumption that all the remaining words are date words. #1 is the text remaining to read.

```
\def\hi@date@readnoqual#1{%
    \find@word{#1}%
        {\hi@date@readpunct}% punct
        {\hi@date@worderr}% group
        {\hi@date@noqualword} % word
        {}% end
}
```

Reads punctuation from a date. All punctuation is ignored other than dashes which are just added to whatever is in `\hi@date@last`.

```
\def\hi@date@readpunct#1#2{%
    \find@eq{-}{#1}{\expandafter\appto\hi@date@last{--}}{}%
    \hi@date@readnoqual{#2}%
}
\make@find@eq{-}
```

Reads a word with the assumption that all remaining words are date words. #1 is the word, #2 the remaining text.

```
\def\hi@date@noqualword#1#2{%
    \hi@date@dateword{#1}{%
        \hi@date@readnoqual{#2}%
    }{\hi@date@worderr{#1}{#2}}%
}
```

Produces an error that #1 is not a date word and then continues parsing the date.

```
\def\hi@date@worderr#1#2{%
    \PackageWarning\hi@pkgname{%
        In date `\hi@date@orig', an unexpected word \MessageBreak
        `#1' was found. Use an exclamation point \MessageBreak
        to accept an unparseable date. This occured%
    }%
    \hi@date@readnoqual{#2}%
}
```

Determines if a word is part of a date. If so, execute appropriate commands to update the date information and run the success macro; otherwise run the alternative macro. #1 is the word, #2 macro for success, #3 the macro otherwise.

```
\def\hi@date@dateword#1#2#3{%
    \ifcsdef{hi@dtmac@#1}{%
        \csname hi@dtmac@#1\endcsname #2%
    }{%
        \@ifnumber{#1}{%
            \hi@date@number{#1}#2%
        }{#3}%
    }%
}
```

Inserts a number into the date. This must determine whether the number is a day or a year. #1 is the number. (This macro always succeeds at placing the number somewhere.)

```
\def\hi@date@number#1{%
    \@test\ifx\hi@date@last\relax\fi{%
        % There was no previous number to a range. The number is either a year
        % or a day, which depends on (1) what is left to set and (2) how big
        % the number is.
        \@test\ifnum#1>\hi@date@maxday\fi{%
            % Must be a year
            \hi@date@set\hi@date@year{#1}{\hi@date@numerr{#1}}%
        }{%
            % If it could be a day, assume it is one unless the day was already
            % set
            \hi@date@set\hi@date@day{#1}{%
                \hi@date@set\hi@date@year{#1}{\hi@date@numerr{#1}}%
            }%
        }%
    }{%
        % The number could go at the end of a range. Check to see if whatever is
        % in \hi@date@last ends with a dash.
        \@expand{\find@end{-}}{\hi@date@last}{ii}{%
            \expandafter\appto\hi@date@last{#1}%
            \@gobble
        }{%
            % If it doesn't end with a dash, then unset \hi@date@last and retry
            % parsing.
            \let\hi@date@last\relax
            \hi@date@number{#1}%
        }%
    }%
}
\chardef\hi@date@maxday=31
\make@find@end{-}
```

When an unexpected excess of date parts is given, this error issues.

```
\def\hi@date@numerr#1{%
    \PackageWarning\hi@pkgname{%
        Too many date parts were found in `\hi@date@orig', \MessageBreak
        and `#1' was ignored. This occurred%
    }%
}%
```

Sets a month. #1 is the formatted month text.

```
\def\hi@date@setmonth#1{%
    \hi@date@set\hi@date@month{#1}{%
        \@expand{\find@end{-}}\hi@date@month{i}{%
            \appto\hi@date@month{#1}\@gobble
        }{\hi@date@numerr{#1}}%
    }%
}
```

Assembles a date from components, and runs callback #1 on the result.

```
\def\hi@date@assemble#1{%
    \expandafter\chop@space@then@run\expandafter{\hi@date@qual}{%
        \def\hi@date@qual
    }%
    \edef\reserved@a{%
        \ifx\hi@date@qual\@empty\else
            \noexpand\hi@fn@dateprefix{\expandonce\hi@date@qual}\space
        \fi
        \ifx\hi@date@pre\@empty\else \expandonce\hi@date@pre\fi
        \ifx\hi@date@month\@empty\else
            \expandonce\hi@date@month\space
            \ifx\hi@date@day\@empty\else \expandonce\hi@date@day,\space \fi
```

```
        \fi
        \expandonce\hi@date@year
        \ifx\hi@date@post\@empty\else \space \expandonce\hi@date@post\fi
    }%
    \@expand{#1}\reserved@a i%
}
```

Defines macros for acceptable inputs for months.

```
\def\hi@setup@month#1#2{%
    \def\do##1{%
        \@namedef{hi@dtmac@##1}{\hi@date@setmonth{#1}}%
    }%
    \docsvlist{#2}%
}
```

Code follows for defining those month inputs. It is used to generate the table for the documentation, so consult the dtx file for the source code.

2(a) Permissible Words The following are permissible words in dates.

Output	Input
Jan.	jan, Jan, January, january
Feb.	feb, Feb, February, february
Mar.	mar, mar, Mar, March, march
Apr.	apr, apr, Apr, April, april
May	may, May
June	june, June, jun, Jun
July	july, July, jul, Jul
Aug.	aug, Aug, August, august
Sept.	sept, Sept, sep, Sep, september, September
Oct.	oct, Oct, october, October
Nov.	nov, Nov, november, November
Dec.	dec, Dec, december, December
Winter	winter, win, Win
Summer	summer, sum, summ, Sum, Summ
Fall	fall, Fall
Spring	spring, spr, Spr
c. [circa]	c, circa
n.d. [no date]	nd
A.D.	ad, AD
B.C.	bc, BC
A.C.E.	ace, ACE
B.C.E.	bce, BCE

Phrases permitted in dates

```
\def\hi@dtmac@c{\appto\hi@date@pre{c.~}}
\def\hi@dtmac@circa{\appto\hi@date@pre{c.~}}
\def\hi@dtmac@nd{\hi@date@set\hi@date@year{n.d\@.}{\hi@date@numerr{n.d.}}}
\def\hi@dtmac@ad{\appto\hi@date@pre{\AD~}}
\def\hi@dtmac@AD{\appto\hi@date@pre{\AD~}}
\def\hi@dtmac@bc{\appto\hi@date@post{\BC}}
\def\hi@dtmac@BC{\appto\hi@date@post{\BC}}
\def\hi@dtmac@ace{\appto\hi@date@post{\ACE}}
\def\hi@dtmac@ACE{\appto\hi@date@post{\ACE}}
\def\hi@dtmac@bce{\appto\hi@date@post{\BCE}}
\def\hi@dtmac@BCE{\appto\hi@date@post{\BCE}}
```

The actual text for the latter few abbreviations can be redefined with `\AD`, `\BC`, `\BCE`, and `\ACE`.

Form of AD and BC

```
\def\AD{\textsc{a.d.}}
\def\BC{\textsc{b.c.}\futurelet\@let@token\hi@bc@post}
\def\BCE{\textsc{b.c.e.}\futurelet\@let@token\hi@bc@post}
\def\ACE{\textsc{a.c.e.}\futurelet\@let@token\hi@bc@post}
\def\hi@bc@post{%
    \ifx.\@let@token
        \expandafter\@gobble
    \else
        \@%
        \ifcat a\@let@token \space \fi
    \fi
}
```