

Providing alternatives to dates BC/BCE in electronic texts using pattern recognition and machine readable markup

By K. G.

June 2022

Abstract. A collaborative project that allows people to use alternative year numbering systems in existing digital texts (ebooks and websites) instead of BC dates that are backwards and unintuitive. People who don't want to use alternative year numbering systems will not be affected by this project. The project already exists and is usable, but it is not in its final form, meaning not enough BC dates are convertible into alternative systems right now. To push it to its final form, where lots of websites and ebooks support translation of BC dates into alternative year numbering systems, a collective effort of a relatively small number of people for a number of years is needed. People working on this project don't need to agree with each other as to which alternative timeline to use instead of BC part of the Christian timeline. The project supports all possible alternative timelines simultaneously.

Introduction

The Christian timeline was not created to present history in the best possible way. Specifically there is nothing clever or helpful in numbering years of history in descending order (dates BC). An alternative year numbering system that reckons years of history in ascending order could help students of history put history into perspective.

We are not saying that the Christian year numbering system should be officially replaced with another system by means of a global calendar reform. Rather, we are saying that alternative year-numbering systems should be made easily available in existing digital texts to anyone who wants to use such systems. At the same time people who want to continue using BC dates should not be affected.

We present here a project (from now on “the Project”) that will make alternative year numbering systems accessible to all people. A special emphasis will be made on making BC dates convertible on Wikipedia.

Example of an alternative timeline

To show what the end result of date conversion in digital texts will look like, we need to pick some alternative year numbering system. Let's use a system that reckons years 10 000 BC to 1 BC in ascending order. So, for example, 10 000 BC will be year 1, and year 1 BC will be year 10 000 in that system. Years of Common era will be left unchanged. Thus, year 10 000 of the “previous era” will be followed by year 1 AD (or CE).



Figure1. Recorded history in an alternative timeline

This system is equivalent to Anterior Epoch proposed by E.R. Hope in 1963⁽¹⁾. It also resembles, at least in the first 10 000 years of it, *Calendrier nouveau de chronologie ancienne (CNCA)* proposed by Gabrielle Deville in 1924⁽²⁾, and Holocene Calendar proposed by Cesare Emiliani in 1993⁽³⁾.

This is just a system that we picked to show concrete examples of date conversions in electronic texts. Other alternative year numbering systems exist and will be supported by the proposed Project. The only caveat is that the Project will not support conversion of years in Common era for reasons that will be discussed below.

Caesar Augustus^{[1][2]} (23 September 9938 – 19 August AD 14), also known as **Octavian**, was the first **Roman emperor**, reigning from 9974 until his death in AD 14.^[a] His status as the founder of the **Roman Principate** (the first phase of the **Roman Empire**) has consolidated a **legacy** as one of the greatest leaders in human history.^[4] The reign of Augustus initiated an era of relative peace known as the **Pax Romana**. The Roman world was largely free from large-scale conflict for more than two centuries, despite continuous wars of imperial expansion on the Empire's frontiers and the year-long civil war known as the "**Year of the Four Emperors**" over the imperial succession.

Originally named **Gaius Octavius**, he was born into an old and wealthy **equestrian branch** of the **plebeian gens Octavia**. His maternal great-uncle **Julius Caesar** was **assassinated** in 9957 and Octavius was named in Caesar's will as his **adopted** son and heir; as a result, he inherited Caesar's name, estate, and the loyalty of his legions. He, **Mark Antony** and **Marcus Lepidus** formed the **Second Triumvirate** to defeat the assassins of Caesar. Following their victory at the **Battle of Philippi** (9959), the Triumvirate divided the **Roman Republic** among themselves and ruled as *de facto* **dictators**. The Triumvirate was eventually torn apart by the competing ambitions of its members; Lepidus was exiled in 9965 and Antony was defeated by Octavian at the **Battle of Actium** in 9970.

After the demise of the Second Triumvirate, Augustus restored the outward façade of the free Republic, with governmental power vested in the **Roman Senate**, the **executive magistrates** and the **legislative assemblies**, yet maintained autocratic authority by having the Senate grant him lifetime tenure

Augustus
Princeps Civitatis



Augustus of Prima Porta, 1st century

Roman emperor

Reign 16 January 9974 – 19 August AD 14

Figure 2. Wikipedia article with automatically converted dates

Converters

To convert dates in an electronic text a special program is needed. Let’s call it Converter. It may come in different forms. It may be a browser extension. If we are talking about converting dates on Wikipedia, it can be a user script or a gadget (a program one can install in the settings of their Wikipedia account). It can be a JavaScript module that a site owner can install on their website. For ebooks it can be just an ebook reader app that has date converting functionality.

Converters may convert BC dates into different alternative year numbering systems. One converter may use the system described above. Another may use 4000 BC as a starting year of the “previous era”. Non-round dates like 5342 BC may also be used as a starting point of a timeline, although this will result in not so nice conversions of centuries and millennia. For example, if we use 10 000 BC as a starting year, 4th century BC will be converted to 97th century. However, if we use 5432 BC as a starting year, then 4th century BC will be converted to “5033/5132 century”. Basically, we’d have to present centuries and/or millennia as ranges of years, which is not very convenient. Some converters may even allow users to choose their preferred timeline.

At the time of this writing there is only one Converter: a Chrome extension called “Old Era on Wikipedia”.

The above description of different converters is not a promise to develop them all. It is just a description of possibilities. The author is only interested in converting BC dates to previously described system (starting at 10 000 BC), or “Old Era” as he calls it. But some people may want to use other systems. They will have to develop their own converters. The easiest way to do so, is by forking the source code of the existing converter and changing the conversion formulas.

The goal of the Project is to make sure that converters may always perfectly detect BC dates in electronic texts. What timeline they will then convert those dates to, is up to their developers to decide. The Project must support all possible timelines.

How converters detect BC dates in texts

There are two ways to detect a BC date:

- by pattern recognition,
- using additional information prepared beforehand.

The additional information can be stored:

- in the target text in the form of special HTML markup (useful when it is possible to edit the target text),
- on a special server (useful when it is not possible to edit the target text).

Pattern recognition

Pattern recognition allows the converter to detect BC dates in most cases. For example a single BC date like “123 BC” is a simple pattern that can be easily detected. Another simple pattern is a range of dates like “149-146 BC”. In this case a converter will recognise 149 as a BC date even though it is not immediately followed by a “BC” label. There may be some variations of this pattern, for example, “149 to 146 BC” or “149 until 146 BC”.

A more complex pattern may look like this “150/49 - 145/4 BC”. Here a converter must not only detect four different year numbers, it must also understand that “49” actually stands for 149, and “4 BC” is actually 144 BC.

Another pattern is a list of dates, for example, “345, 342 and 297 BC”. Converter should be able to recognise the first two numbers in this list as BC dates. There are patterns for detecting decades like “520s BC”, centuries and millennia as well. Sometimes the lists of dates may be rather complex, like “345-344, 325-323, 233, 149-146 BC”.

In general, patterns are detected using JavaScript regular expressions.

If we wanted to be able to detect all possible patterns, the list of patterns that we’d need to take into account would be very long and constantly growing. We’d never know if we had taken into account all possible patterns. Another concern is that the more complex a pattern is, the more likely it is to produce a false positive: some numbers in target texts that are not actually BC dates may be mistakenly converted as if they were BC dates.

Therefore, it makes sense to limit the number of patterns we try to detect. We only need to use the simplest ones, that nonetheless cover a lot of dates: patterns like single dates, ranges and simple lists of dates.

If a certain date is not detectable by simple pattern recognition, there are a few ways it can be made detectable. Sometimes a “BC” label is omitted when it is obvious from the context that the date in question is actually a BC date. Because of it, pattern recognition will not work for such dates. If appropriate, an editor may simply add a missing “BC” to complete a pattern. It will make the date detectable through pattern recognition. This is often possible to do on Wikipedia.

Markup

Sometimes, adding a “BC” label to a date is not appropriate. In such cases a special markup should be used.

On the level of HTML the markup may look like this:

```
<span class="bc-y">149</span>
<span class="bc-i">1000</span>
<span class="bc-d">520s</span>
<span class="bc-c">4th</span> century
<span class="bc-m">1st</span> millennium
...
```

This markup may be extended with a couple of “data” attributes. For example:

49 - here we tell the converter that “49” stands for “149 BC”: the digit “1” was simply omitted like in the “150/49” example from above. “S” in “data-s” stands for substitute.

149 BC - here we tell the converter that this date is a part of a book title. “T” in “data-t” stands for type. The possible attribute values are “t” for book or article title and “q” for a quote. Whether or not to convert dates in book titles and/or quotes can be configured in the settings of a converter. Some converters may actually ignore this attribute and always convert all dates. Also note, how we have a year with a “BC” label, but we still put markup around it because we needed to specify that the date belongs to a book title.

There are currently 17 different markup classes:

- bc-y** - year,
- bc-y1** - year must be shortened to one digit after conversion,
- bc-y2** - year must be shortened to two digits after conversion,
- bc-i** - imprecise year,
- bc-i2** - imprecise year that should be shortened to two digits after conversion,
- bc-d** - decade,
- bc-sd** - short decade,
- bc-dp** - decades (plural),
- bc-00s** - centuries that look like 1400s BC,
- bc-000s** - millennia that look like 2000s BC,
- bc-c** - century,
- bc-m** - millennium,
- bc-r** - remove,
- bc-ig** - ignore (don’t convert whatever is inside),
- bc-tn** - timeline name (for example “Old Era”),
- bc-ot** - of timeline (for example “of the Old Era”),
- bc-at** - abbreviated timeline name(for example “OE”).

Explaining these classes in more detail is beyond the scope of this document.

You don’t usually work directly with HTML. For example, on Wikipedia you edit wiki text. So, for each class there needs to be a template. The code of a template for class ‘bc-y’ will look like this:

```
<span class="bc-y" {{#if:{{{2}}}| data-t="{{{2}}}"|}} {{#if:{{{3}}}| data-s="{{{3}}}"|}}>{{{1}}}</span>
```

Other templates will look the same, only the class name will be different each time. The templates simply create HTML markup described above based on a cleaner wiki text markup.

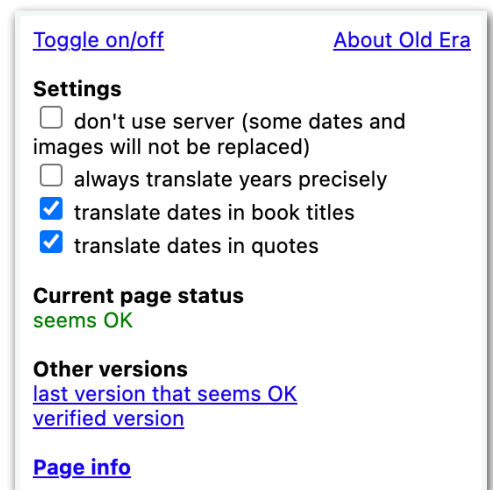


Figure 3. Users may choose not to translate some dates

This is what the markup will look like in wiki text:

```
{{bc-yl149}}
{{bc-il1000}}
{{bc-dl520s}}
{{bc-cl4th}} century
{{bc-ml1st}} millennium
...
```

`{{bc-yl49||149}}` - here we tell the converter that 49 should be replaced with 149 before conversion takes place.

`{{bc-yl149 BC|t}}` - here we tell the converter that this date is a part of a book title.

A set of similar templates or functions should be created for any content management system (Wordpress, Jumla, etc.) a particular website runs on.

Markup example

Let's look at a simple example of markup usage. Let's say that we have this sentence in wiki text:

Born c. 685 BC, Ashurbanipal was a son of his predecessor Esarhaddon (r. 681–669).

Let's add markup:

Born c. 685 BC, Ashurbanipal was a son of his predecessor Esarhaddon (r. `{{bc-yl681}}`–`{{bc-yl669}}`).

We've added markup to dates 681 and 669, because they are not detectable by pattern recognition. Year 685 BC, on the other hand, will be detected by pattern recognition and therefore doesn't need any markup. Alternatively, we can simply add the missing "BC":

Born c. 685 BC, Ashurbanipal was a son of his predecessor Esarhaddon (r. 681–669 BC).

Once dates are converted, the result will be the same regardless of which approach we've taken. If we use the before-mentioned "Old Era", the result will look like this:

Born c. 9316, Ashurbanipal was a son of his predecessor Esarhaddon (r. 9320–9332).

Storing positions of dates on a server

Sometimes it isn't possible to edit the target text. Actually, at the time of this writing it isn't clear if we even can add markup on Wikipedia. Even though we can technically add markup on Wikipedia, it may be easily deleted by other editors. First, there must be a consensus within Wikipedia community that it is OK to add machine readable markup to BC dates. We'll talk about how to reach such consensus below. Until such consensus is reached, there is another way to detect dates that are not detectable with simple pattern recognition. That is to store the positions of those dates on a special server.

There may be various ways to store positions of dates on a server. There are no rules for how a server should perform its job other than this: it should allow converters to reliably detect BC dates in texts. Basically, positions of dates stored on a server should be a complete substitute for markup that for some reason can't be added to the target text.

The server that we currently have at dates.oldera.org stores positions of BC dates like this:

inal bust from 70 to 60 BC, <a h	70	impreciseYear	
ust from 70 to 60 BC, <a href="/wik	60 BC	impreciseYear	
t Rebellion of 52 bce had notable su	52 bce	year	quote
ber of them in 52 bce to shake off t	52 bce	year	quote
he Roman Army, 31 BC–AD 337</i>. Ro	31 BC	year	bookTitle

Figure 4. This is how positions of dates are stored on the server

The converter will send URL of the page, it currently looks at, to the server, and the server returns a list of dates on that page and pieces of HTML code surrounding each date. Using this information, converter can find the position of each date on the page. Storing positions of dates this way makes the detection of dates on the page very robust: most changes to the target text don't affect the detectability of dates.

Using a server is not desirable. It centralises the Project and ties it down to one person (or organisation) that has to maintain it. Regular edits made on a website like Wikipedia mean that all the information on the server needs to be checked from time to time, and any mismatches between information in the target web pages and the server should be eliminated. Also, using a server is not always possible. For example, a Wikipedia gadget is not allowed to send requests to third party servers. And using a Wikipedia gadget is the best way to convert BC dates when viewing Wikipedia on mobile devices.

We should strive for gradual reduction of the role of a server (or servers) in the Project. Once we start adding markup on Wikipedia we will be removing the positions of corresponding dates from the server.

Once there is no data related to Wikipedia left on the server, the server will be repurposed to store positions of dates found on websites other than Wikipedia. In that capacity it will serve probably for a very long time. While we can easily add markup to Wikipedia, the markup on other websites must be added by owners of those websites. Whether they add it or not, will depend on how popular some of the alternative year numbering systems will become. And on some websites the markup will probably never be added for various reasons.

Processing digital texts


Any digital text containing BC dates should be 'processed'. It means a human editor must review the text and make sure that all the BC dates are being detected correctly. If some dates are not being detected, the editor may make them detectable by adding a missing "BC" label to a date, or marking up the date with special markup we discussed earlier. If the text is not editable, the editor may store the positions of such dates on a server. Also, the editor may mark some dates as imprecise or belonging to a book title or a quote, and/or make other small adjustments.

In many cases these edits may be done manually. But for processing large articles with lots of BC dates special software should be used. It can be one of two types: software for editing text (wiki text in case of Wikipedia), and software for finding dates on a web page and sending their positions to a server. These programs help find all the dates by highlighting candidates (for example, all numbers in case of years BC) in the source text or on the actual web page. A program for editing text has not yet been developed at this moment. So, let's look at an example of a program for finding dates on a web page and sending their positions to a server. This program is developed but is not yet published anywhere.

With this program one can, for example, highlight all numbers to make sure no BC dates are overlooked, even the ones not marked with a “BC” label. On the image below we first highlighted all the automatically detectable dates (green) then we highlighted all the remaining 2-digit numbers (red). Now we can see that there is year 52 that is not detectable by pattern recognition. We can either add a “BC” to it, using standard editing functionality of Wikipedia, or make it green and then send all the dates positions to the server.

While Caesar was in Britain his daughter Julia, Pompey's wife, had died in childbirth. Caesar tried to re-secure Pompey's support by offering him his great-niece in marriage, but Pompey declined. In 53 BC, Crassus was killed leading a failed invasion of Parthia. Due uncontrolled political violence in the city, Pompey was appointed sole consul in 52 as an emergency measure.^[71] That year, a "Law of the Ten Tribunes" was passed, giving Caesar the right to stand for a consulship *in absentia*.^[72]

From the period 52 to 49 BC, trust between Caesar and Pompey disintegrated.^[73] In 51 BC, the consul Marcellus proposed recalling Caesar, arguing that his *provincia* (here meaning "task") – due to his victory – in Gaul was complete; the proposal was vetoed.^[74]^[75] That year, it seemed that the conservatives around Cato in the Senate would seek to enlist Pompey to force Caesar to return from Gaul without honours or a second consulship.^[76] Pompey, however, at the time



A Roman bust of Pompey the Great made during the

Figure 5. Finding dates that may be overlooked by automatic pattern recognition

The server will only save information about dates that can't be automatically detected by pattern recognition. This way only a relatively small amount of data is stored on the server for each page.

Text editing software (that is yet to be developed) will work on the same principles. It will help human editors to not overlook any BC dates by highlighting all possible candidates. But instead of working with an actual web page it will be working with the source text. And the result of its work will be a marked up text. In case of Wikipedia an editor will be able to copy wiki text of a page, paste it in the editing program, add markup using different tools within the program, then paste the resulting wiki text back to the Wikipedia page.

How much work is it?

We'll start with Wikipedia. There are 6,5 million pages on English Wikipedia. But only about 60,000 of them contain BC dates (less than 1% of the entire Wikipedia). At this moment about 1400 pages have been processed already (although using a server, not in-place markup). Out of the remaining pages, about 30,000 contain 1 or 2 mentions of a “BC” (or “BCE”) label. Most of the BC dates on those pages are probably perfectly detectable by pattern recognition. Another 24,000 pages have each 3-50 “BC”s. There are only about 700 pages containing 51-100 “BC”s, and about 150 pages containing more than 100 “BC”s.

You can check these numbers here: dates.oldera.org.

How long will it take to process all BC dates on Wikipedia? It depends on how many people do the processing and how consistently they do it. It can take months, or it can take years to process everything. On the other hand, it's not an all or nothing situation. The Project is already usable, BC dates on a large number of pages are convertible because of pattern recognition. Sometimes you can see a BC date that is not convertible. These flaws will be fixed with time. In many cases you can fix them yourself (when it's appropriate to add a missing “BC” label). When we have a go-ahead from Wikipedia community to use machine readable markup we talked about earlier, you'll be able to fix any date conversion issue on Wikipedia yourself.

We'll eventually expand the Project to websites other than Wikipedia and also to ebooks. We can expect that the percentage of texts, be it web pages or ebooks, that need to be processed, compared to the amount of all the texts in the world will be the same 1% as for Wikipedia, if not a lot less.

All in all, the entire project (including other websites and ebooks) will take years, but probably, no more than a decade. However, from the point of view of an individual student of history, who wants to put history into perspective using an alternative year numbering system, the real questions should be: "Do I have enough processed texts about ancient history *for myself*?" and "If I encounter some text, where BC dates are not convertible, do I have the tools to make them convertible without much effort?"

For most people "enough texts" is just Wikipedia. As for the tools, once we can add markup to Wikipedia, the editing functionality of Wikipedia will become your tool for making dates on the website convertible. Plus, there will be a text editing program we already mentioned.

How many BC dates need to be marked up?

On most pages you usually only need to markup a small number of BC dates if any. Since no markup has been added to Wikipedia as of yet, we can use data on the server (dates.oldera.org) to get a feel for the amount of markup that will be needed.

Example 1. There are 201 instances of "BC" label in the article on Cleopatra. The number of actual BC dates is usually larger than the number of "BC" label instances, because not all BC dates are marked with a "BC" label. In any case, out of more than 201 dates we need to markup 20 dates (actually a bit less than that, because we won't need to markup "BC"s that follow centuries).

a> (yellow) in 40 BC</div></div></d	40 BC	impreciseYear		
original from 70 to 60 BC, and	70	impreciseYear		
nal from 70 to 60 BC, and located i	60 BC	impreciseYear		
pace:nowrap;"> 30 BC</div></	30 BC	impreciseYear		
pace:nowrap;"> 50–30 BC,	50	impreciseYear		
e:nowrap;"> 50–30 BC, now lo	30 BC	impreciseYear		
ing 18 days in 30 August BC. How	30	year		
ys in 30 August BC. However, <a h	BC	remove		
the end of the fourth century BC. Th	fourth	century		quote
fourth century BC. The Ptolemies	BC	remove		quote
oraries of the first century BC had	first	century		quote
e first century BC had another, p	BC	remove		quote
pire Builders (300 BC – 1 AD), Roset	300 BC	year	2.1.1.1	bookTitle
in late summer 58 BC and fearing fo	58 BC	year		quote
herself around 275 B.C. Though such an	275 B.C.	year		quote
an in the late 30s B.C.—it was essenti	30s B.C.	decade		quote
as prepared in 44 B.C. to introduce l	44 B.C.	year		quote
pire Builders (300 BC – 1 AD), Roset	300 BC	year	2.2.1.1	bookTitle
and Octavian (44–30 B.C.)", <i>	44	year		bookTitle
d Octavian (44–30 B.C.)", <i>American	30 B.C.	year		bookTitle

Figure 6. Dates that need to be marked up on Cleopatra page.

Example 2. There are 157 instances of “BC” label in the article on Julius Caesar. This time we only need to markup 8 dates.

inal bust from 70 to 60 BC, <a h	70	impreciseYear	
ust from 70 to 60 BC, <a href="/wik	60 BC	impreciseYear	
sole consul in 52 as an emergenc	52	year	
t Rebellion of 52 bce had notable su	52 bce	year	quote
ber of them in 52 bce to shake off t	52 bce	year	quote
consulship in 59 "seems to be n	59	year	
r's actions in 59 were overturne	59	year	
he Roman Army, 31 BC–AD 337</i>. Ro	31 BC	year	bookTitle

Figure 7. Dates that need to be marked up on Julius Caesar page.

Example 3. There are 72 instances of “BC” label in the article on Pyrrhus of Epirus. We need to markup only 3 dates.

p. 298: "From 288 until 284, Pyr	288	year	quote
From 288 until 284, Pyrrhus and L	284	year	quote
ise of Rome to 220 BC</i>. Vol.	220 BC	year	bookTitle

Figure 8. Dates that need to be marked up on Pyrrhus of Epirus page.

As you can see, most often you need to add markup to dates located in book titles or quotes. Sometimes you need to mark some round years as imprecise, so they can be converted to round years (for example, with imprecise conversion you can convert 2000 BC to year 8000 instead of year 8001). In most cases, if the markup is not yet added, the dates converted by pattern recognition are perfectly usable. You just can't turn off conversion of dates in book titles and quotes from the settings of your converter program, and you may see some approximate dates that may look like 8001, 8501, etc., ending with '1', while it would be better if they were rounded. So, for most pages the markup is not an absolute necessity, but rather it will be used to make the result of date conversions look nicer.

The markup becomes a necessity when “BC” labels are omitted. For example, on the page about Roman Republic they are barely used at all. We’ll have to markup 198 dates (only a fraction of them is shown on the screenshot).

e></div> <p>By 390, several <a hr	390	year
</div> <p>From 343 to 341, Rome w	343	year
<p>From 343 to 341, Rome won <a h	341	year
Latin War (340–338), Rome def	340	year
n War (340–338), Rome defeate	338	year
r began in 327.<sup id="cite_	327	year
ated, but from 314, Rome was domi	314	year
Bovianum (305). By the follo	305	year
there; but in 298 the Samnites r	298	year
pulonia in 282 Rome finished	282	year
g point was in 400, when the firs	400	year
plebeians (in 399, 396, 388, 383	399	year
eians (in 399, 396, 388, 383, and	396	year
(in 399, 396, 388, 383, and 379)	388	year

Figure 9. Some of the dates on ‘Roman Republic’ page that will need markup

Alternatively, we can add missing “BC”s and then the markup won’t be needed. There are, however, pages like “List of Roman consuls” where we have just a big table with about 500 dates BC, where it wouldn’t be appropriate to add a “BC” to every date. On such pages there is no other option but to add lots of markup. But such pages are very rare.

How will markup affect the page sizes?

Let’s look at the worst case scenario by taking the article “List of Roman consuls” as an example. We have to mark up about 500 dates with markup that on the level of HTML looks like this `<some year>`. Additional symbols are `` and there are 26 of them. They take 26 bytes. For 500 dates that would be 13 kB. That is the additional information that you’ll be downloading from Wikipedia. Also, this additional information will be cached on Wikipedia servers. Is it a lot of data? We can compare it, for example, to thumbnails of images on Wikipedia. The average size of a thumbnail (after sampling 5 random thumbnails) is 63 kB. So the size of our additional information is about one fifth of the size of an average image thumbnail (not even an actual full sized image). And that is the worst case scenario. We’ll only have a few pages like that. On most pages the additional information will be measured in bytes, not kilobytes.

Will the markup cause any inconvenience for editors on Wikipedia?

This time, instead looking at the worst case scenario, let's use a typical case. Take, for example, 'Pyrrhus of Epirus' page we looked at before. We only needed to add markup to 3 dates there. This is what the markup will look like in the wiki text:

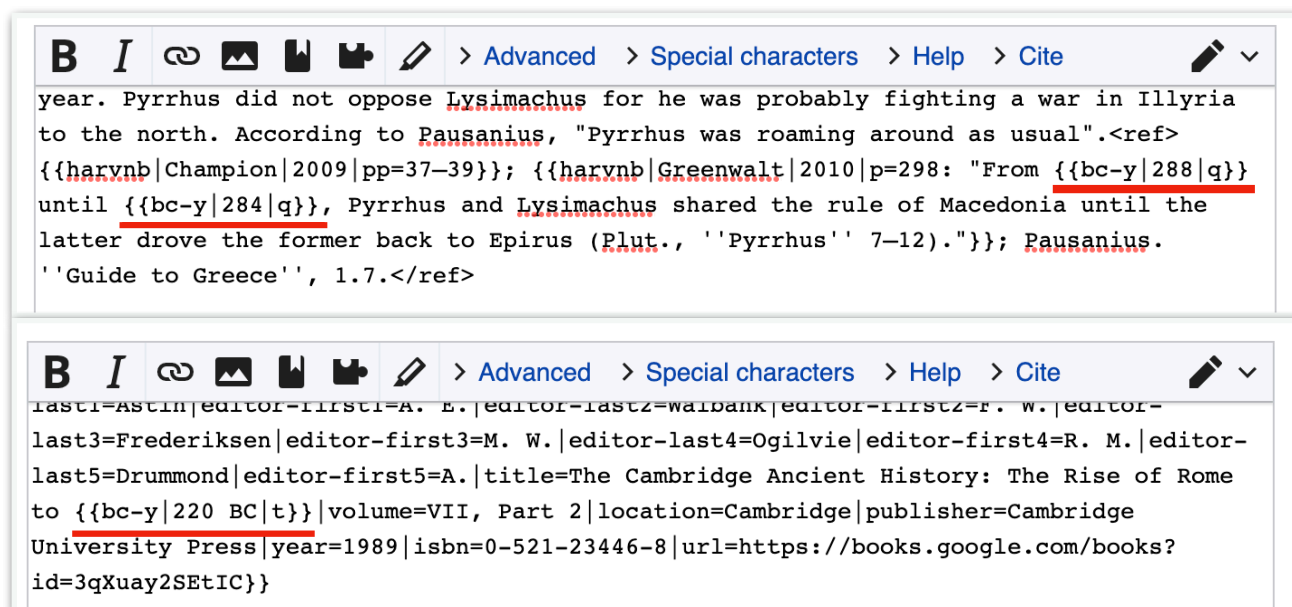


Figure 10. Example of markup in wiki-text

As you can see, wiki text was not meant to be super readable in the first place, and the proposed markup is not any more complex or confusing than a lot of other stuff you may find in the surrounding text. We also need to remember that, like in this example, we will typically be adding markup to only a few dates even on pages with lots of BC dates.

What about converting years of Common Era?

If you want to use a system like Holocene Calendar, you may want to convert not only dates BC (BCE), but dates AD (CE) as well. For example, you'd want to be able to convert year 2022 to year 12,022 HC. However, in this Project we don't even try to detect dates of Common era.

The main reason is the sheer amount of CE dates. As was mentioned before, Wikipedia pages with BC dates comprise less than 1% of all the pages on Wikipedia. CE dates, on the other hand, can be found on every Wikipedia page. Even if an article is about ancient history, it still contains bibliography section with book publication dates, which are mostly CE dates. Even on pages with lots of BC dates it is reasonable to expect as many CE dates as there are BC dates if not more.

CE dates are also harder to detect. When it comes to BC dates, if a date is detectable or not, usually comes down to whether or not it is followed by a "BC" label. With CE dates it's not as easy as that. You may try to categorise any number, that looks like a year, as a CE date, but you'd end up with lots of false positives. To fix that a lot of markup would be needed. You may as well give up on pattern recognition and just mark up all the CE dates.

When it comes to detecting BC dates, apart from 1% of Wikipedia, we only need to worry about history related websites and ebooks. And the number of those is not astronomical. If we process the top 1000 most popular websites about history and the top 1000 most popular history ebooks, we will create a new reality where it will seem to you that you can use alternative year numbering systems almost everywhere. CE dates, on the other hand, may be found not only in history related texts, but pretty much in any text regardless of its topic.

The point is, using the same techniques described here and the same resources, we'd need to do at least 1000X more work to make CE dates translatable compared to the amount of work needed for making BC dates translatable (assuming there are 100 times more texts that need to be processed and 10 times more dates that need to be marked up on each page). Let's assume that we need 10 years to achieve all of our goals when it comes to making BC dates translatable. We'd need at least 10 000 years to do the same for CE dates in existing digital texts. And the amount of texts with CE dates constantly increases. Of course, people still write books and articles about ancient history, thus increasing the number of BC dates in digital texts as well, but the rate of this increase is nothing compared to the rate at which the amount of CE dates increases. Basically, you have ancient history on one hand, and every other topic on the other hand. It's like comparing the amounts of water running through a dripping tap vs. Niagara Falls.

Can we use machine learning to detect CE dates?

If using the proposed techniques will require unreasonable amount of work to make all CE dates detectable, maybe we just need another technique? What about machine learning (ML), for example? The short answer is: it's not worth it.

One way to do it would be to use a special server that does all the date detection. Converter would send HTML of the current page to the server, and the server would tell the Converter where the CE dates are in the HTML. The server would use a neural network to do its job. If done this way, the project will be highly centralised. It wouldn't be a collaborative project. And it won't require any changes on Wikipedia, for example. That means that if someone is so inclined, they can start implementing this solution right now. Also, on websites other than Wikipedia this would be the only possible solution, since you can't add markup on those websites.

As for Wikipedia, another way to do it would be to use ML as an offline solution to automatically add markup to all 6,5 million pages on Wikipedia in a relatively short period of time. As we discussed earlier, all pages on Wikipedia contain CE dates, and you would probably want to add markup to every single CE date. You would also need to run your program periodically to add markup to newly added dates. Then the converters would use the markup to translate CE dates without having to communicate with any server.

This kind of solution will raise a lot more questions than the proposed markup of BC dates. Is it a good idea to add markup to all CE dates on all pages? Who's going to run the program that would add all this markup? How accurate will the ML solution be? Will we be able to keep up with the changes on all Wikipedia pages?

When it comes to detecting CE dates, the main challenge would probably be to find someone willing to undertake such a project. That project would require a lot of work, a lot of resources, likely, a lot of fighting, if you want to add markup to CE dates on Wikipedia. And all of that for what? So that you could convert years like 2022 to something that looks like 12,022? A conversion you can easily perform in your head.

Why not use machine learning to detect BC dates?

You may think, why bother with markup or storing positions of dates on a server? Can't we just use a ML solution to detect all possible patterns of BC dates?

Some of the reasons why it may be a bad idea have already been mentioned in the previous section about CE dates. Apart from those, there is yet another reason. To teach a neural network to detect dates in texts you need lots of texts with labeled dates. Preferably many thousands of articles. It just so happens that the kind of work we do in the Project - marking up dates in texts - is equivalent to labelling them. So we are already creating a foundation for a ML solution if we ever need to use one in the future. But, because there is only about 60,000 articles with BC dates on Wikipedia, using ML as an online solution (where date detection is done on a separate server) may be an overkill. By the time we've processed a few tens of thousands of articles it may be easier to just continue until all of them are processed. Then the markup itself will help detect dates, and no ML solution will be needed.

An offline ML solution (where we use ML to add markup to pages) may be useful. But because there are not very many pages with BC dates to begin with, the ML solution in this case will most likely be used for maintenance (finding mistakes in already marked up pages due to new edits). It won't be as important for speedy markup of pages that hasn't been previously marked up.

In short, a ML solution may be useful in the future, but we don't have to start with it, and even if we wanted to start with developing a ML solution, we'd still have to add markup to lots of dates manually (or rather using editing tools we already discussed).

Adding markup to Wikipedia

Wikipedia is by far the most important website for the proposed Project. Anyone, who wants to put history into perspective by using an alternative year numbering system, will gain a lot from simply skimming through a bunch of Wikipedia articles where the BC dates have been converted into their preferred timeline. It also just so happens that Wikipedia is editable, which potentially allows us to add markup to it and to stop using a separate server for storing positions BC dates in Wikipedia articles.

Markup requires less maintenance than a server storing positions of dates. When markup is used, the only way date conversion may become broken is if someone deliberately removes the markup from a marked-up date, or adds a new date that lacks both markup and a "BC" label. When position of a date is stored on a server, the conversion of that date may become broken for various reasons: date itself was changed, "BC" was added or removed. Oftentimes a date has simply been removed, and while the page looks perfectly fine, human editor must spend his time deleting information about such dates from the server. With markup, date conversion issues will be super rare. And once we have markup, we'll be able to use Wikipedia gadget for converting dates on mobile devices.

While it's important to add markup to Wikipedia, the question is, are we even allowed to do so? Technically, there is nothing that prevents us from adding the markup right now. However, if we start doing so, at some point some editors may delete our markup, deeming it unnecessary.

So, adding the markup covertly is probably not a good idea. On the other hand just asking for permission and not doing anything until we get a clear go-ahead is probably not a good idea either. It's not even clear who you should ask for such permission.

We need to explain what we are trying to achieve and then start adding markup right away. We can start by processing a few articles at first. Then we may stop to discuss the Project with other editors, using processed pages as examples. Then we can process some more articles, have more discussions, and so on. We must not engage in editing wars, but we can't only talk about our Project. In short, we will need to start making some changes on Wikipedia even before we have a clear go-ahead from Wikipedia community. But we'll be making those changes in a civil and non-confrontational way.

Decisions on Wikipedia are primarily made by consensus. So our goal will be to reach a positive consensus regarding our Project. What will our proposal to Wikipedia community be? It will go something like this:

There is a group of people who need a machine readable markup of dates BC on Wikipedia. They are going to add that markup themselves. This markup will be invisible to readers of Wikipedia. Editors of Wikipedia may occasionally see it in wiki text but will not be negatively affected by it in any way. The increase in page size due to addition of markup will be negligible. The majority of BC dates will not even need to be marked up. The markup of BC dates will be universal: there will never emerge another group of people, who would demand a different kind of markup for the same BC dates. Nothing out of the ordinary will be required from editors of Wikipedia due to introduction of the markup. Editors will be required to not delete the markup from BC dates arbitrarily, if it's already there. That's all. And this is not even a new rule, because you are not supposed to arbitrarily delete stuff from Wikipedia anyway.

Think about it this way. You are a reader and an editor of Wikipedia. You can go to Wikipedia right now and edit something. That makes you an editor. And so the proposal we will be presenting Wikipedia community with is addressed to you as much as it is addressed to anyone else. You are the decision maker here. And let's even assume that you don't understand why anyone would need to use an alternative year numbering system. Maybe you are not even interested in ancient history to begin with. That doesn't matter, because the proposal is not about you, or what timeline *you* prefer. Nor is it about the entire humanity. We are not talking about whether or not the humanity should convert to another calendar. It is about a relatively small group of people who for some reason want to use alternative year numbering systems on Wikipedia. This group may be comprised of different subgroups each of which uses a different year numbering system. These are unimportant details. What is important, is that you will not be affected by their edits in any way. The question to you is this: do you think these people should be allowed to get the information from Wikipedia in the format they prefer? And if not, why not?

Allowing said markup would be in the spirit of Wikipedia. Wikipedia is about providing free information to all people. That's why there are Wikipedias in many languages. Why should Wikipedia then discriminate people based on the timeline they want to use? Especially given that the proposed changes will be rare, invisible and non-destructive.

Some people may try to compare the supporters of the proposed Project to flat-earthers or some religious sect, saying that it's not Wikipedia's job to accommodate such people. But we are not trying to change the content of Wikipedia. It's not like we want Wikipedia to store different versions of truth about certain topics. We simply want to make BC dates detectable by programs and scripts. Seems kind of uncontroversial. At least it should be.

What if they say no? In that case we will continue to work on the Project using our server, and from time to time we will be coming back to Wikipedia with more and more people, asking the same questions: "Why do you discriminate against us? We only want BC dates to be detectable on Wikipedia. Why is it such a problem?" The more people we have on our side, the more ridiculous any attempts to stop us will seem, because, in the end of the day, there is no rational reason for fighting against the usage of the proposed markup on Wikipedia. But that's the worst case scenario. Hopefully, Wikipedia community will be, if not completely understanding, let alone enthusiastic about our Project, then at least indifferent enough to let us do our thing from the very first try.

Wikipedia has something called Wiki projects. It's a way for people to collaborate on improving different aspects of Wikipedia. On the pages of our Wiki project we'll be able to coordinate our efforts in adding markup to articles. Although one can just create a Wiki project, it is recommended to start with a formal proposal. And this is where we are going to present and explain our Project to Wikipedia community.

One of the pages of our Wiki project will contain detailed rules on how Converters should detect BC dates. There will be a list of patterns that every Converter should be able to recognise, and a list of all the markup classes with explanations on when to use each of them and how. It will be possible to update this list by consensus. The more Converters there are the harder it will be to change the rules. Eventually this list of rules will stabilise and turn into some sort of web standard for detecting BC dates in electronic texts.

Converting BC dates on images

Sometimes an image may contain BC dates. In such cases the image may be edited in any graphic editor. Then a converter may simply replace URL of the original image with URL of the edited image.

How does the converter know which images to replace? Just like with dates it can use a special markup or it can get that information from a server. Where are the edited images stored? Currently, they are stored on the server. In future they may be stored on Wikipedia and other target websites. While the markup of BC dates will support conversion of dates to any alternative year numbering system, when you add an edited image to Wikipedia, you add it for one specific timeline. And since anybody can invent any timeline, some vetting process will be needed to determine, which timelines deserve to have images on Wikipedia. For example,

they may need to have some notability, which may be determined by whether or not the timeline in question has already been mentioned on Wikipedia before.

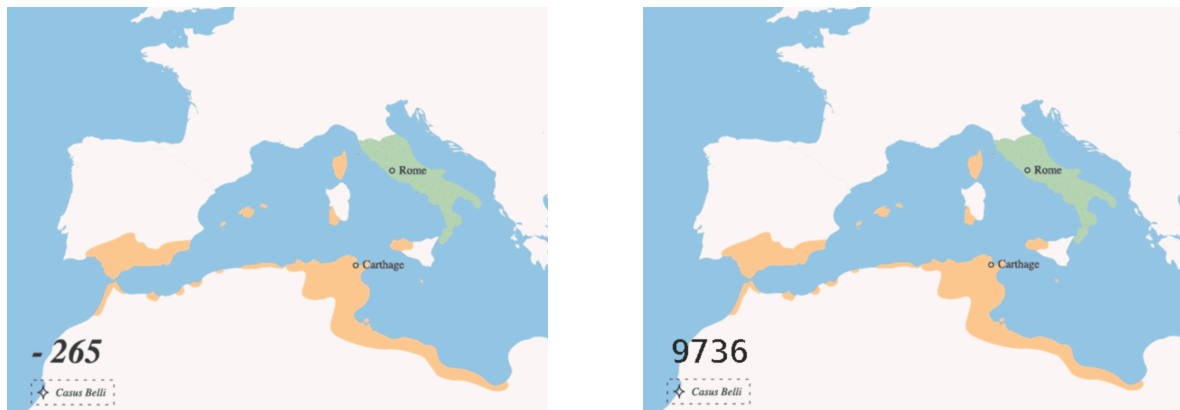


Figure 11. An example of original and edited image

Addition of edited images to Wikipedia will be proposed to Wikipedia community later and separately from the BC dates markup proposal.

Detecting BC dates on websites other than Wikipedia

Wikipedia has a set of rules about style. When developing a converter for Wikipedia one can take advantage of these rules. For example, writing down BC dates like this - “B.C. 200” - is discouraged on Wikipedia. And so, we don’t have to worry about detecting a pattern where BC goes before the year number. If we encounter something like “B.C. 200” in an article, we may simply replace it with “200 BC”.

Also, there are certain pages on Wikipedia where we can deduce what certain dates are even if they don’t visually match any pattern. For example:

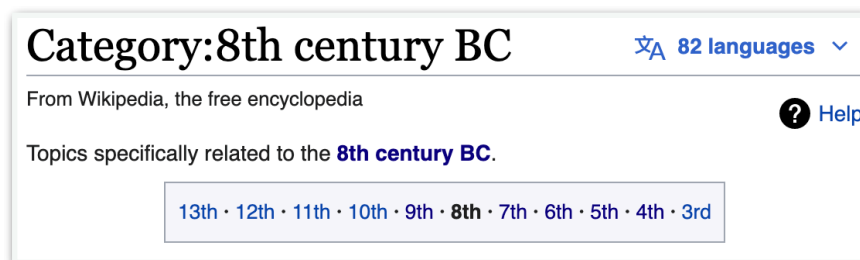


Figure 12. Example of a potentially problematic place in text

How do we know that “13th”, “12th”, etc., mean centuries BC? Adding markup would be problematic here, because this bar is generated programmatically. We may (and probably will) propose changes to the bar generation code, so that the bar is generated already having all the necessary markup. But in the meantime we may look into HTML code of the bar and find a useful pattern there.

```
<ul><li><a href="/wiki/Category:13th_century_BC" title="Category:13th century BC">13th</a></li>
<li><a href="/wiki/Category:12th_century_BC" title="Category:12th century BC">12th</a></li>
<li><a href="/wiki/Category:11th_century_BC" title="Category:11th century BC">11th</a></li>
<li><a href="/wiki/Category:10th_century_BC" title="Category:10th century BC">10th</a></li>
<li><a href="/wiki/Category:9th_century_BC" title="Category:9th century BC">9th</a></li>
<li><b>8th</b></li>
<li><a href="/wiki/Category:7th_century_BC" title="Category:7th century BC">7th</a></li>
<li><a href="/wiki/Category:6th_century_BC" title="Category:6th century BC">6th</a></li>
<li><a href="/wiki/Category:5th_century_BC" title="Category:5th century BC">5th</a></li>
<li><a href="/wiki/Category:4th_century_BC" title="Category:4th century BC">4th</a></li>
<li><a href="/wiki/Category:3rd_century_BC" title="Category:3rd century BC">3rd</a></li></ul>
```

Figure 13. Invisible patterns that may help

Now, if the converter encounters something like *title=“Category:<number> century BC><number>*, it will recognise that the number means century BC.

In short, Wikipedia has certain features that may help the converter to detect dates. But when you develop a converter that needs to work on all websites, adjusting it to peculiarities of every single site may not be practical. Therefore it may make sense to have one converter for Wikipedia, and another for all other websites. The second converter will not use patterns that exploit certain invisible features of specific websites. On the other hand it will recognise more visible patterns like before mentioned “B.C. 200” pattern.

Also, while we are going to try and make the converter for Wikipedia server-less over time, the second converter will be heavily dependent on a server, because it will deal mostly with sites that are not editable. The list of supported sites will be stored on the server, because we don’t want converters to bombard the server with requests from any random site. Converters will only call the server when they are on a site that is included in the list. Pattern recognition, on the other hand, will be available everywhere, and the user will be free to disable it on selected sites.

Detecting BC dates in ebooks

Currently, this is the least researched part of the proposed Project. There are no concrete solutions yet, only ideas. Here is how it may work. We need a special ebook reader program. Theoretically, it should be possible to use some existing open source program and add date converting functionality to it, although developing an ebook-reader from scratch is possible as well. The conversion of BC dates will work on the exact same principles we already discussed: most dates will be converted by simple pattern recognition, the rest will be either marked up in place, or their positions in the text will be stored somewhere else, probably, in special patch files. There can be a repository of patches on some server, and the ebook reader app will be able to download those patches as needed. One patch per ebook. This way it will be possible to process ebooks without modifying them. This is important, since we can’t modify and redistribute most ebooks due to copyright laws. If storing pieces of HTML inside patch files is considered a copyright violation, another format may be developed where we store only indices of dates. A patch file in this case will contain only a bunch of numbers. Such format will be possible because unlike texts on Wikipedia, texts in an ebooks never change.

To process ebooks and create patch files a special program will be needed as well.

Maintaining the Project

Once we accomplish all our goals, and BC dates are detectable and therefore convertible almost everywhere, we will have to maintain the Project in a working condition. For Wikipedia it means that we will have to keep fixing any date conversion issues that will keep emerging due to new edits. Once the markup is implemented, such issues will be super rare and will be treated just like any typos or mistakes.

How do you usually deal with typos on Wikipedia? You don’t lose your sleep over their existence and you don’t purposefully hunt for them. But when you see a typo, you can fix it right then and there. You can do likewise with date conversion issues. And if on the initial phase of processing thousands of articles we are going to use special text editing software, once the Project is done and we are in the maintenance phase, we will mostly do edits manually. Most edits will consist of adding a “BC” label or markup to a specific date that somebody has just added.

As for other websites and ebooks, maintenance of the Project there should be even easier, because texts in ebooks and on most websites are static. Once you process them, BC dates in them will always stay convertible. We will only have to make sure that our server keeps on running.

Current state of the Project

We currently have only one converter - “[Old Era on Wikipedia](#)” Chrome extension. It works only on Wikipedia and relies on a server for detecting dates that are not detectable by simple pattern recognition. The

functionality of recognising markup is built into the extension as well. So, as soon as markup appears on Wikipedia this program will be able to use it. The extension is open sourced.

The data on the server is currently editable by only one person - the author. In future this will need to change. It isn't urgent though, because currently our main objective is to add markup to Wikipedia. Markup makes server unnecessary. But when we start processing pages on websites other than Wikipedia, the server will be useful, and we will need to make sure that anybody can edit data on it.

The software for finding dates on web pages and sending their positions to server exists in the form of another Chrome extension that is currently available only to the author. This will need to change in future as well. As for text editing software, that will be used for finding and marking up dates in wiki text, its development has begun.

Wikipedia gadget that will allow the conversion of dates on mobile devices, the second converter, that will convert dates on all the sites other than Wikipedia, the ebook-reader mobile app that will convert dates in ebooks are all yet to be developed.

All of the software created for the Project will be open-sourced eventually.

Supporting other languages

Up until now we've only discussed the English version of the Project. How can we support other languages? Detecting BC dates is very language specific. Different languages will have different date patterns and will require different sets of markup classes. Therefore, for each language all the software and all the rules for detection of BC dates will have to be created independently.

Does the Project have to be popular to be successful?

The entire Project can be done to completion by a small number of enthusiasts. So, no, the Project doesn't have to be very popular. Will it lead to humanity gradually converting to some alternative year numbering system? It is possible given enough time. But, in a sense, it isn't important whether this happens or not. The goal of the Project is to give the people, who want to use alternative year numbering systems, an ability to do so. Once you have dates in your preferred year numbering system everywhere on the internet and in ebooks, and you know that they are not going to change or disappear, the fact that the entire world uses some other system becomes irrelevant to you.

However, popularity may help the Project along the way. For example, it will be easier to justify the usage of the proposed markup on Wikipedia if there are hundreds or, better still, thousands of people actually using some alternative year-numbering systems. At this time the number of such people can be roughly estimated by checking the number of users of "Old Era on Wikipedia" Chrome extension.

In general, the more popular the Project is, the better it is for each individual user of any alternative year numbering system, because he or she has to do less work when it comes to processing texts. And if you want to one day get your hands on a paper book that uses dates in your preferred alternative timeline, that timeline will have to have some noticeable popularity.

In short, popularity helps, but it is not required for the success of the Project.

Conclusion

We have proposed a project that will allow people to use alternative year numbering systems pretty much everywhere: on Wikipedia, on history related websites and in ebooks. We showed that most BC dates can be detected by pattern recognition. The ones that can't be detected this way, should be marked with a special HTML markup. If that is not possible, the positions of such dates in target texts should be stored on a special server. The amount of target texts that should be processed this way is rather small, since we want to detect only BC dates.

We also showed that making CE (AD) dates detectable everywhere would, most likely, require a prohibitively large amount of work, and therefore is rather unrealistic.

We put a special emphasis on making the Project server-less on Wikipedia. This will make detectability of BC dates a feature of Wikipedia rather than just somebody's pet-project, and will greatly simplify the maintenance of the Project. We proposed that a Wiki project dedicated to adding markup to BC dates should be created on Wikipedia. It should contain a standard for detecting BC dates. Software for marking up BC dates in wiki text should be developed as well.

We also proposed a way to make BC dates in ebooks convertible without breaking any copyright laws.

References

- (1) Hope, E.R. (1963). "The arithmetical reform of the calendar, Part I". *Journal of the Royal Astronomical Society of Canada*. **57** (1): 14–23. <https://ui.adsabs.harvard.edu/abs/1963JRASC..57...14H/abstract>
- (2) Gabriel Deville (1924), *Calendrier nouveau et chronologie ancienne : Exempleaire avec des notes et des corrections manuscrites*, Viroflay, Seine-et-Oise / Versailles: l'auteur / impr. de C. Barbier
- (3) Emiliani, Cesare (1993). "Correspondence – calendar reform". *Nature*. **366** (6457): 716.