Detecting BC(BCE) dates in electronic texts

by Karen Grigorian

October 2023

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 almost all 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.

This is not a calendar reform proposal. I am not saying that the Christian year numbering system should be officially replaced with another system. Rather, I am 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.

I present here a project (from now on "the Project") that will allow anybody to use alternative year numbering systems in existing digital texts (web pages and ebooks). A special emphasis will be made on making BC dates convertible on Wikipedia.

An alternative timeline example

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). We will call that 'previous era' Old Era.

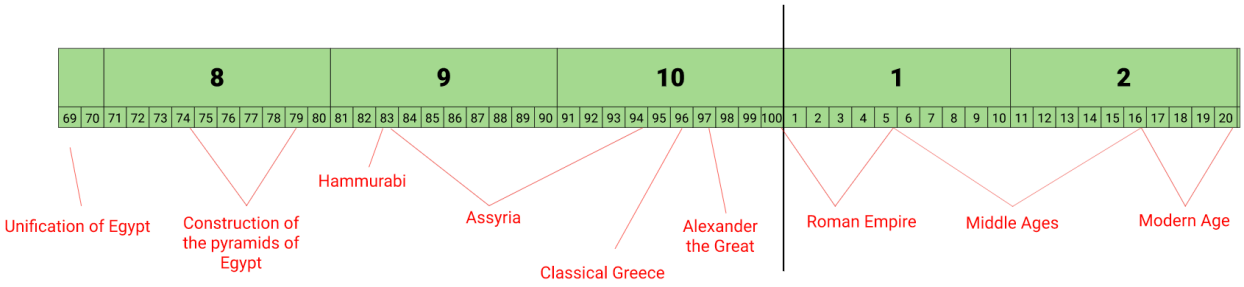

Figure1. Recorded history in an alternative timeline (Old Era)

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 Emiliany 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.

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. On Wikipedia it can be a user script or a gadget (a program one can install in the settings of their Wikipedia account), however once you have a browser extension, user scripts and gadgets become redundant. It can be a JavaScript module that a site owner can install on their website (again probably redundant). For ebooks it can be just an ebook reader app that has date converting functionality.

Converters may convert BC dates into one or multiple different alternative year numbering systems.

An example of a converter that you can use right now is a Chrome extension called "Old Era - Ancient Date Converter". It works on all websites and allows you to choose the the starting point of your timeline.

How converters detect BC dates in texts

There are two ways to detect a BC date:
- using 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.

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 through simple pattern recognition, there are a few ways it can be made detectable. Sometimes a “BC” annotation 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 the 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:
****149****
****1000****
****520s****
****4th**** century
****1st**** millennium
…

This markup may be extended with one “data” attribute:
****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.

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-s="{{{2}}}"|}}>{{{1}}}

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.

This is what the markup will look like in wiki text:
{{bc-y|149**}}**
{{bc-i|1000**}}**
{{bc-d|520s**}}**
{{bc-c|4th**}}** century
{{bc-m|1st**}}** millennium
…

{{bc-y|49|149**}}** - here we tell the converter that 49 should be replaced with 149 before conversion takes place.
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-y|681}}–{{bc-y|669}}).

We've added markup to dates 681 and 669, because they are not detectable through pattern recognition. Year 685 BC, on the other hand, will be detected through 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 possible to add markup on Wikipedia. Even though we can technically do it, the markup will soon be 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.

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

Figure 3. 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 text 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 makes it dependent on one person (or organisation) that has to maintain it. Regular edits made on websites like Wikipedia mean that all the information on the server needs to be checked from time to time, and any mismatches between information on the target web pages and the server should be eliminated. 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.

When it comes to websites other than Wikipedia, storing positions of dates on a server will be the only option probably for a very long time. While we can easily add markup on Wikipedia, the markup on other websites must be added by the 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.

Luckily, most websites are not like Wikipedia in that their content doesn't change as often, which means the information that you store on the server will remain relevant for a very long time and therefore won't require a lot of maintenance.

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 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 tools should be used. Such tools are already developed and are built into the before mentioned Chrome extension "Old Era - Ancient Date Converter".

The processing goes as follows. First, you need to highlight all dates on the page with special colors. One color for years BC, another color for centuries BC and so on.

Initially the program will highlight all the dates that it can detect on its own.

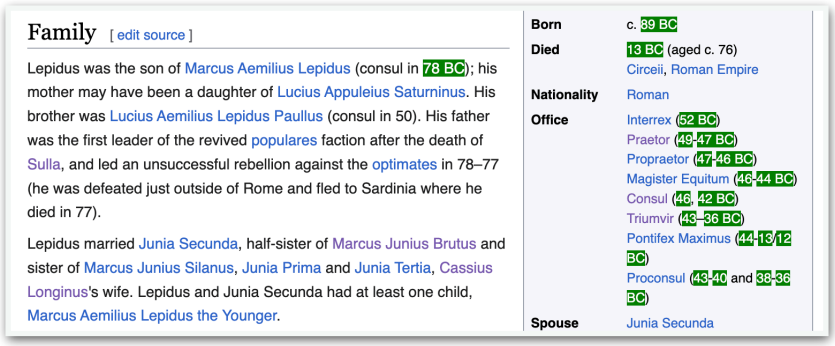

Figure 4. The program highlighted the dates it can detect on its own

You can then start highlighting dates that the program missed. There are tools built into the program that help you do that.

For example, you can first automatically highlight all the numbers of a certain length on the page.

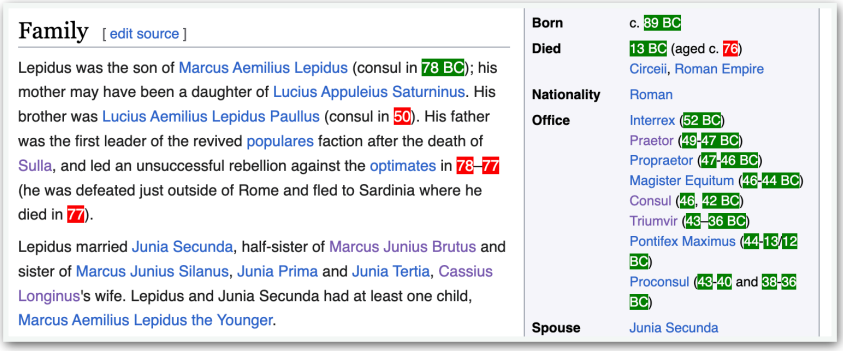

Figure 5. Highlighting all 2-digit numbers on the page to make sure we don't miss any BC date

And then select the ones that are actually BC dates and change their color to the appropriate one.

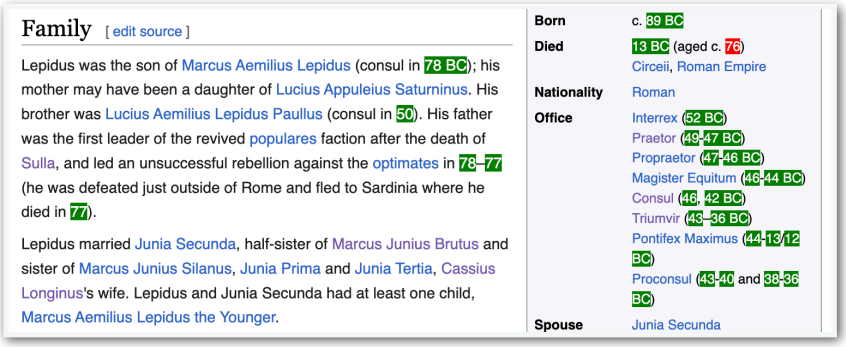

Figure 6. Selecting and changing the color of BC years

Once all the dates are highlighted with correct colors, you can open the Editor popup.

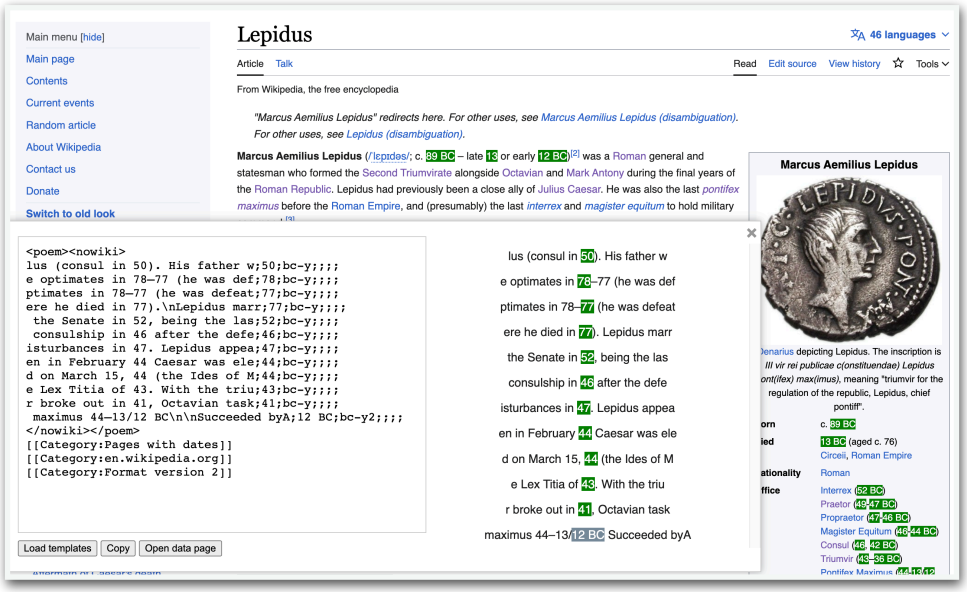

Figure 7. Editor contains list of dates that the Converter wouldn't be able to detect on its own

In the editor you can see all the dates that the Converter will need help with. On the left you see the same information in CSV format. From here you can copy that text, open data page on our server and save it there:

Figure 8. Saving data on the server

This works for any website that was added on this page. And it is what we currently have to do even for Wikipedia pages. However, once we are allowed to add markup on Wikipedia, there is a second step and a second Editor we can use - the Wikitext Editor.

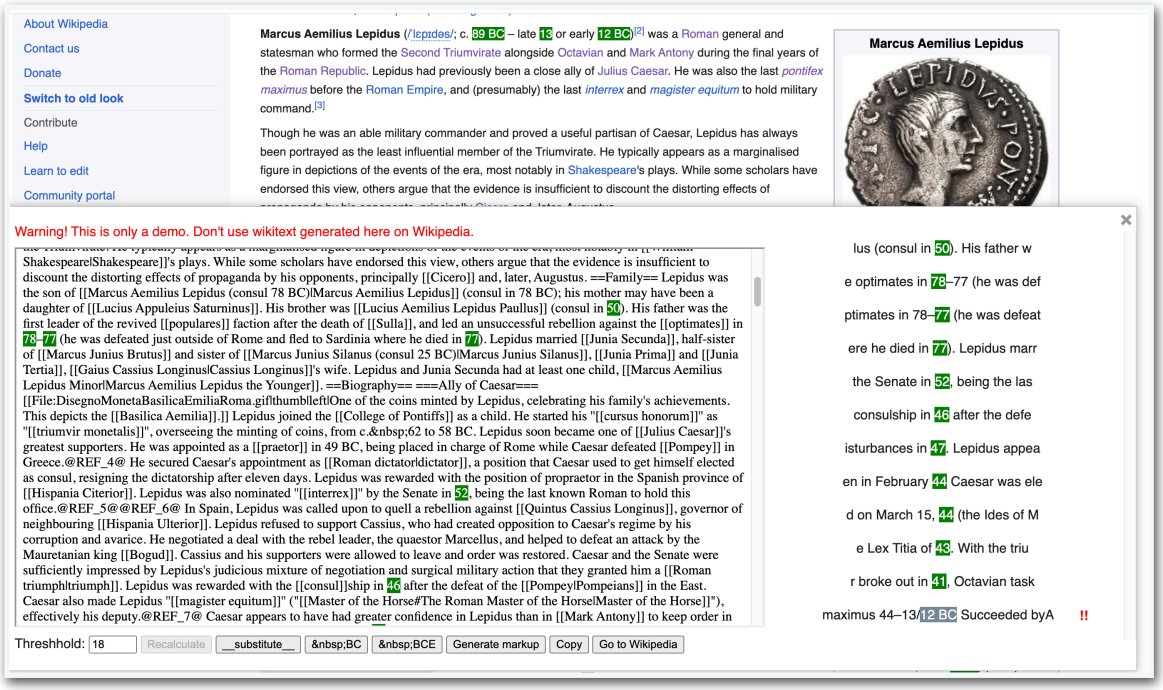

Figure 9. Wikitext Editor popup

Wikitext Editor loads wikitext for the current page from Wikipedia. On the right hand side it shows all the dates that will need to be marked up in the wikitext. It also tries to automatically highlight those dates in the wikitext. Sometimes it fails to do so, because wikitext and the text on the web page may be too different from each other in some places. In such cases it shows that with exclamation marks. You can then find and highlight those dates in wikitext yourself. For some dates you can just add BC annotation manually or using corresponding buttons in the Wikitext Editor instead of highlighting those dates. Once all dates are highlighted with corresponding colors in wikitext you can click 'Generate markup' button and the final wikitext with markup will be created. Now you can click 'Copy' and then 'Go to Wikipedia'. The current Wikipedia page will be opened for editing. Just paste the new wikitext there. Then you can click 'Show changes' on Wikipedia page to make sure all the edits are correct.

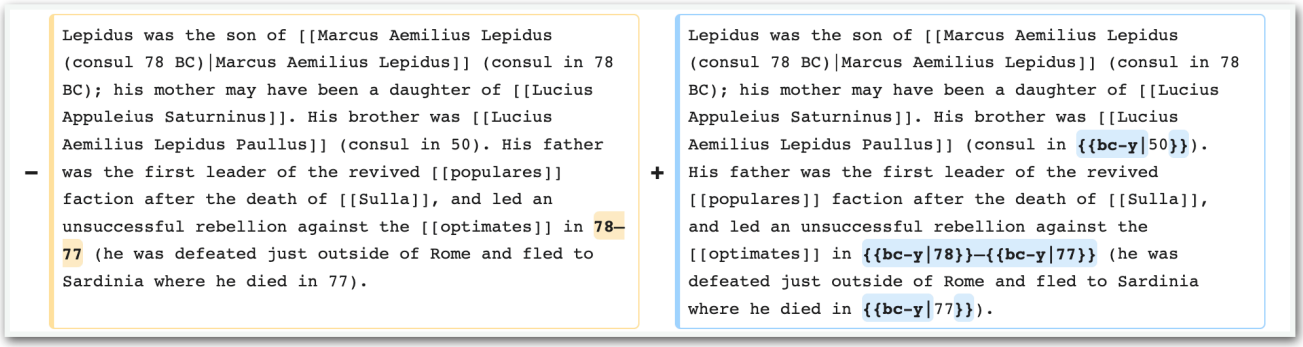

Figure 10. Making sure that all the edits are the ones you intended to make

Just don't save those changes yet. Wikipedia doesn't have the templates that are needed for this to work. For now the Wikitext editor serves only as a demo of what's possible.

How much work is it?

We'll start with Wikipedia. There are 6,5 million pages on English Wikipedia. But only about 55,000 of them contain BC dates (less than 1% of the entire Wikipedia). At this moment about 1500 pages have been processed already (although using a server, not in-place markup). Out of the remaining pages, about 28,000 contain 1 or 2 mentions of a "BC" (or "BCE") label. Most of the BC dates on those pages are probably perfectly detectable through pattern recognition. Another 25,000 pages have each 3-50 "BC"s. There are only about 760 pages containing 51-100 "BC"s, and about 160 pages containing more than 100 "BC"s.

You can check these numbers here: dates.oldera.org. Those numbers will change with time as more pages are processed.

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. You can actually fix any such issue yourself using the tools provided by the Old Era Chrome extension.

We can currently process pages on any website. And eventually the Project will be expanded 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 by 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. And the tools for processing texts already exist (at least for web pages).

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 more than 200 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 200 dates we need to markup only 8 dates.

Example 2. There are 71 instances of "BC" label in the article on Pompey. This time we only need to markup 5 dates.

Example 3. There are 72 instances of "BC" label in the article on Pyrrhus of Epirus. We need to markup only 2 dates.

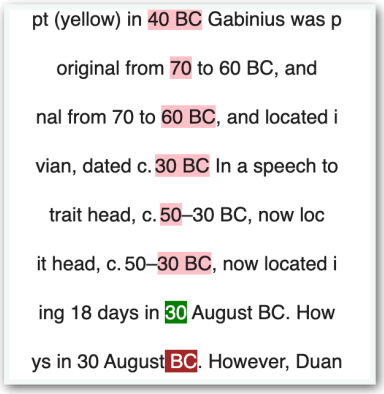

Figure 11. Dates that need to be marked up on Cleopatra page.

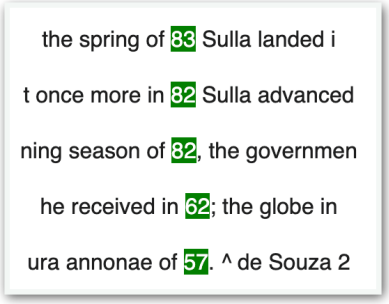

Figure 12. Dates that need to be marked up on Pompey page.

Figure 13. Dates that need to be marked up on Pyrrhus of Epirus page.

Most often you need to add markup to only a few dates even if a page contains a lot of BC dates. A lot of times you only need to mark some round years as imprecise (shown as pink in Figure 11), so they can be converted to round years (for example, with imprecise conversion you can convert 2000 BC to year 8000 of Old Era 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 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. Take for example Wikipedia page "List of Roman consuls" where you 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 cashed 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?

Proposed markup is not any more complex or confusing than a lot of other stuff you may find in the surrounding wikitext. See Figure 10 for an example. We also need to remember that we will typically be adding markup to only a few dates even on pages with lots of BC dates.

What about converting years of the 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 2023 to year 12,023 HC. The extension actually allows you to convert dates AD that have "AD" (or "CE") labels. The majority of AD

dates don't have era annotations though, so they can't be detected through simple pattern recognition. And it's not practical to try to make them all detectable using markup or storing there positions on a server.

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.

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, if you want to make all AD (CE) dates detectable you can start processing those dates the way I process BC dates, but you will never finish that project. You will need many thousands of years just to process dates in all the currently existing digital texts.

Luckily, converting only the dates that have era labels, may be just what you need in the case of AD dates. When it comes to BC dates, you want to convert them all, regardless of whether or not they have era labels. But when converting AD dates you can actually get away with using 5-digit numbers only sometimes, while using familiar 4-digit year numbers most of the time.

Original	Translation to Holocene Calendar
13 BC – AD 35	9988 – 1,0035
7 – 43 CE	1,0007 – 43
2023	2023

Figure 14. Examples of AD dates conversions

Notice how we use comma separator to separate 4 digits, showing that the first digit "1" is optional. Also notice how in the range of dates (7 - 43 CE) we only add extra digit to the first year in the range. And lastly, if a date doesn't have an "AD" label, there is no need to add an extra digit (also, we can't, because there is no reliable way of detecting such dates without producing lots of false positives).

Using machine learning to detect BC dates

Trying to detect BC dates using ML on the fly is not very practical. But some ML solution can be used at some point in the future to process texts. Using some large language model like ChatGPT it can be done as follows. We can extract text from a web page and send it to the LLM, asking it to list all BC dates in that text. Then, using the original text and the list of BC dates, we can create snapshots of text in the needed format (like on Figure 8) and store them on the server.

According to a rough calculation that I made, processing all 55,000 pages that contain BC dates on Wikipedia using ChatGPT may cost just a couple thousand dollars.

The only problem with ChatGPT is that it doesn't always produce correct answers. But I only tested version 3.5. Version 4 may be better at this task and also some prompt engineering may help improve the answers. In any case this topic needs further research.

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" annotation. 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, the page changed its title, and now you can't find the related information on our server, and so on. 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.

While it's important to add markup to Wikipedia, we are currently not allowed to do so. Technically, there is nothing that prevents us from adding the markup right now. However, if we do so, at some point our markup will be deleted by other editors.

We need to convince Wikipedia community that adding the markup is a good idea and is actually aligned with the goals of Wikipedia. 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 total number of pages that even contain BC dates in the first place is less than 1% of the total number of articles on Wikipedia. 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 want 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?

The only problem with this proposal is that it can be easily rejected without much discussion simply because currently we can't show that there is a significant number of people who need date converting functionality on Wikipedia.

So, the current plan is to temporarily treat Wikipedia as any other website and achieve convertibility of BC dates there by storing positions of dates, that are not detectable through simple pattern recognition, on our server.

This achieves three objectives. First, we get the needed functionality without having to wait for anybody's approval. That in turn helps us to grow the user base, because we have something that it usable right now. Secondly, by the time we are ready to make our proposal on Wikipedia, we will have a complete list of all the articles that will need to be edited and we will also have information about all the necessary edits. We can even collect some statistical information on the number of instances of different date patterns on all those pages: there are some tradeoffs between pattern recognition and markup that statistical information can help resolve. Basically, our proposal will be more detailed. And thirdly, when we are finally allowed to add markup, the work of highlighting the dates in articles will be done an saved on the server. We won't have to do it again. When you open a previously processed page for editing, all the dates get highlighted automatically using the information from the server, and all you have to do is make sure all those highlights are moved correctly to wikitext using Wikitext Editor.

We probably won't be ready to make a proposal on Wikipedia for the next few years. But it doesn't mean we shouldn't talk about the Project there. In fact, I plan to start a discussion with Wikipedia community soon. The main question that I have for the community is "What would it take for Wikipedia community to take this project seriously?" Should I show that a certain number of people use alternative timelines? How many? And how would I prove that number? Should I show that some historians are using alternative timelines? And so on. All those questions can be discussed in the near future. There is no need to wait until the proposal is ready.

Wikipedia has something called Wiki projects. It's a way for people to collaborate on improving different aspects of Wikipedia. I have created a page for our future Wiki project. There we'll be able to coordinate our efforts in adding markup to articles. The Wiki project is currently in draft state and will likely be in that state for a long time.

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. Over time the 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.

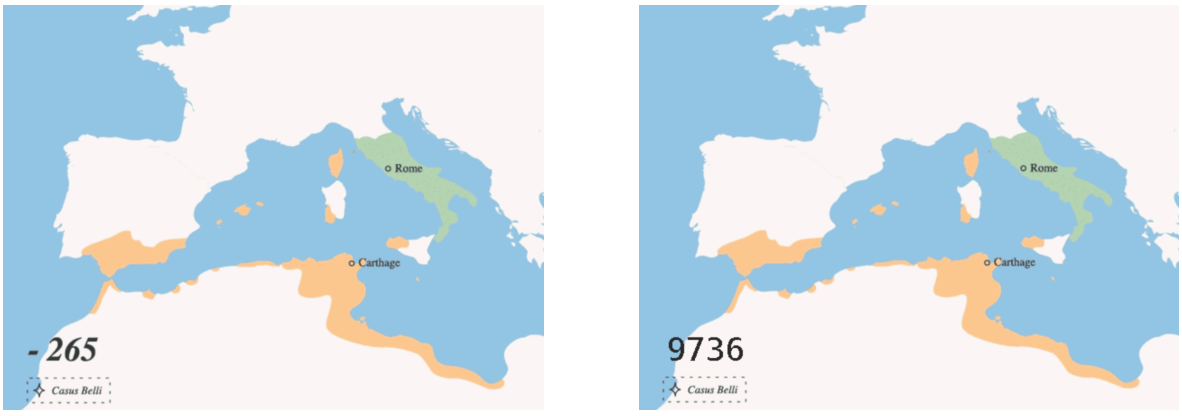

Figure 15. An example of original and edited image

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. Currently, the images are stored on the server (and there are not that many of them yet). In the 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.

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

Another way to convert dates on images is to programmatically create rectangular patches on top of an image in places where BC dates are located. The rectangles will block the original dates and inside those rectangles there will be texts with converted dates. The advantages of this approach is that you don't have to store an extra image and you can translate dates into any timeline. You will only have to store original BC dates in text form and some parameters like coordinates of patches, text sizes and background colors. All the measurements will be relative to the size of the image, so that even if the image is resized the converted dates stay in the right places on top of it. The disadvantage is that you can't use this approach for every image (all those patches may look ugly on some images), and especially you can't use it with animated images.

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, on the same server where we store information for web pages. 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 text 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.

Converting dates in videos

I also plan to add a certain feature to the main extension that adds subtitles to YouTube videos so that when a narrator mentions a BC date you can see the converted date on the screen. This functionality can be achieved by simply finding all BC dates in the transcript of the video and making a list of them, with each BC date having a timestamp. The list should be stored on the server. The Converter will use that list to show subtitles with converted dates in the appropriate moments defined by the timestamps.

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. 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 have processed them, BC dates in them will always stay convertible. We will only have to make sure that our server keeps on running.

To ensure the longevity of the Project at some point we will probably have to start a non-profit organisation that will accept donations and use them to run the server.

Current state of the Project

The main converter, that we currently have, is called "Old Era - Ancient Date Converter". It is a Chrome extension. It works on any website. It allows you to pick the starting point of your preferred timeline, and it can also convert AD dates that have era labels. While it can detect the markup we discussed earlier, it relies on a server for detecting dates that are not detectable through simple pattern recognition simply because the markup is not added to any website yet. As soon as markup appears on Wikipedia (or any other website) this program will be able to use it.

The extension contains all the tools that let anybody to process texts. If some BC-date is not detectable, you can always fix that. There is also a built-in tool that allows you to create markup for Wikipedia but it will be used only after we are allowed to add said markup to Wikipedia. The extension is open sourced.

To use this extension on mobile Android devices you just need to use a browser that allows installation of Chrome extensions on mobile devices. For example, I use Kiwi browser on my Android phone.

As for converting dates on iOS devices, I plan to create a Safari web extension soon.

An ebook-reader app that will convert dates in ebooks is yet to be developed.

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

The up-to-date list of software can be found on this page.

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 an 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 the "Old Era - Ancient Date Converter"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

I 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. I showed that most BC dates can be detected through 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.

I also showed that making all CE (AD) dates detectable everywhere would, most likely, require a prohibitively large amount of work, and therefore is rather unrealistic. But detecting only labeled CE (AD) dates can be done automatically and may be just what you need.

I 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.

I 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 : Exemplaire 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.