archaeometry

Data Complexity and Computer Assisted Chronology: Methods and Discoveries

Journal:	*Archaeometry*
Manuscript ID	ARCH-11-0950.R2
Wiley - Manuscript type:	Original Article
Date Submitted by the Author:	04-Jan-2025
Complete List of Authors:	Falk, David; no affiliation
Keywords:	chronology, computer analytics, permutations, methodology, ancient Near East, practice, hypothesis falsification, validation, synchronisms, software tools, consistency
Abstract:	Data complexity is one of the most formidable problems facing modern chronology. Chronologists in the past have attempted to mitigate the problem through reducing the amount of math needed by curating the data set. However, evaluating the significance of curated data can be subjective and often incorrect because choosing the most significant synchronisms is not always intuitive. For better chronologies, new methods and tools are needed that retain complexity and interdependence with cross-cultural data but eliminate the subjectivity of human data curation. The consequence of these needs would be that constructing a chronology requires not just hundreds of thousands of calculations, but also an additional number of calculations to show that a chronology is potentially valid. This paper posits that one can craft better, more accurate historical chronologies using a tool such as the Groundhog Chronological Laboratory, which was specifically designed for that purpose. Groundhog is uniquely designed to help historians and chronologists design new chronologies and test them for internal consistency.

SCHOLARONE™
Manuscripts

Data Complexity and Computer Assisted Chronology: Methods and Discoveries

By

David A. Falk

ABSTRACT:

Data complexity is one of the most formidable problems facing modern chronology. Chronologists in the past have attempted to mitigate the problem through reducing the amount of math needed by curating the data set. However, evaluating the significance of curated data can be subjective and often incorrect because choosing the most significant synchronisms is not always intuitive. For better chronologies, new methods and tools are needed that retain complexity and interdependence with cross-cultural data but eliminate the subjectivity of human data curation. The consequence of these needs would be that constructing a chronology requires not just hundreds of thousands of calculations, but also an additional number of calculations to show that a chronology is potentially valid. This paper posits that one can craft better, more accurate historical chronologies using a tool such as the *Groundhog Chronological Laboratory*, which was specifically designed for that purpose. *Groundhog* is uniquely designed to help historians and chronologists design new chronologies and test them for internal consistency.

KEYWORDS: chronology, computer analytics, permutations, methodology, ancient Near East, practice, hypothesis falsification, validation, synchronisms, software tools

For Peer Review

Data complexity is one of the most formidable problems facing modern chronology. Chronologists in the past have attempted to mitigate the problem through reducing the amount of math needed by curating the data set, i.e., only using the synchronisms that are deemed most significant. However, evaluating the significance of curated data can be subjective and often incorrect because choosing the most significant synchronisms is not always intuitive. For better chronologies, a method was needed that retains complexity and interdependence with cross-cultural data but eliminates the subjectivity of human curation. The consequence of these needs would be that constructing a chronology requires not just hundreds of thousands of calculations, but also an additional number of calculations to show that a chronology is potentially valid. This paper proposes that new methods and tools be used to craft better, more accurate historical chronologies.

THE PROBLEM OF CHRONOLOGY

New chronologies are being published on a frequent enough basis that widespread confusion has been created for museums, historians, and the public. Individual academic specializations within the historical disciplines each face unique difficulties when it comes to chronology. Yet, the history of chronology shows that there are common challenges that practitioners across the field face. With that said, I will touch upon a few brief examples that will underscore the challenge chronology faces as field of study.

For Peer Review

Beginning in the 1980s, Kitchen correlated the kings of Egypt to other kings of the ancient Near East (particularly those of Assyria and Babylon) using the Amarna Letters to derive a low chronology that was made most famous by the introductory work published by Ian Shaw (Kitchen 1987, 37-55; 1991, 201-208; 2006, 293-308; Shaw 2003, 480-489). While this low chronology was congruent with much of the cross-cultural historic data (Shaw 2003, 12), it has failed to gain traction among the majority of Egyptologists even though the unviability of the competing high chronology has been thoroughly demonstrated (Aston 2013, 297; Pruzsinszky 2009, 21). At least some of that seems to be the result of a longstanding belief which has asserted that Egyptian Chronology is "the touchstone by which all of the other chronologies are measured…" (Hornung 2006, 8).

In a similar manner, Assyriologists consider their chronologies to be a gold standard that stands independent to other dating considerations, a claim that I largely agree with. Yet, despite having a low and a high chronology, the need for museum signage and textbooks for schools drove the discipline to standardized upon a middle (convenient) chronology with many in the field agreeing that the middle chronology is not convincing (McIntosh 2005, 47).[1]

Based upon these issues, there have been those who have tried to revise chronology in order to standardize or even improve upon it. Creasman and Kamrin started an initiative to create an

[1] "Most scholars opt to use the Middle Chronology (dating Ammi-saduqua's succession at 1646 B.C.E.), not because it is the most convincing but because it is a convenient temporary fixed point until the correct date is finally established" (McIntosh 2005, 309). "The fall of the First Dynasty of Babylon to the Hittites is dated 1595 BC according to the Middle Chronology. There has been much debate as to whether this chronology, which is generally accepted for convenience, should be replaced by a higher or a lower chronology." (Collon 2007, 107).

For Peer Review

additive chronology in 2016 using the one agreed upon absolute date in Egyptian chronology (i.e., the death of Taharqa in 664 BCE) and then add the reign-lengths together in order to calculate an absolute error rate for a standard chronology (Creasman & Kamrin 2016). The method they used, however, is no more sophisticated than "dead-reckoning" (i.e., calculating the lengths of reigns backwards from an assumed fixed point in time) (Höflmayer & Cohen 2017, 2). combined with astronomic observations (i.e., lunar or Sothic data)."[2] They have intentionally avoided any cross-cultural comparisons by deprecating the use of ancient Near Eastern synchronisms, in favor of evidence that is entirely internal to their discipline. This has resulted in exaggerated error values to the extent where their chronology is impossible when viewed in light of the synchronisms of the ancient Near East. As a result of their methodology, their efforts cannot be regarded as more reliable than their predecessors.

Regrettably, most attempts to update chronology ultimately repeat the problems of the past. And the reason is because the methods used in chronology have, until recently, remained the same. Many disciplines have become a silo, in effect creating self-referential chronologies employing little more than an intuitive "dead-reckoning" methodology where absolute dates are calculated

For Peer Review

[2] The use of Sothic and lunar dating is plagued with difficulties. With Sothic dating, the observation is "not merely a matter of the geometric relationship among sun, star, and horizon" (O'Mara 2003, 25). We do not know the latitude from where the observations were taken, at what elevation they were taken, or the "atmospheric conditions and the quality or the horizon." (O'Mara, 25). We also do not know if the observation is astronomical or calendrical (O'Mara, 26). If the latter, the retrocalculation would be 2918 or 2920 years. These unknown variables make the reading unreliable for chronological work.

In the case of lunar dating, issues of reliability surround the observer. The ancient observer did not always record the phases of the moon consistently. A case in point is the multiple lunar dates that are given for the reign of Thutmosis III where "it is impossible to accept both dates as they stand without emendation." (Manuelian 1987, 11). Unlike the Sothic cycle, which is completely unreliable for dating purposes, lunar dating can be used critically to provide supporting evidence when used with other pieces of historical data.

by hand using data curated for its perceived (subjective) significance. Much of this seems to stem from a fundamental failure to understand how basic chronology works in a discipline where one does not have all the data, i.e., the problem where the length of every ruler's reign is not known with certainty (Thiele 1965, 9). This is not a criticism of the efforts of the past, but an observation that the discipline of chronology is more complex than it intuitively seems.

With all the data surrounding chronology one might think that the dates would be more secure than the previously described situation infers. However, the problem is not that of finding secure dates, often seen as the holy grail of chronology, but an omission to properly address ambiguity in the methodology. A small illustration with a fictitious chronology demonstrates the point:

[TABLE 1]

In this example, there are only three kings (looking back in time) forming a sequence of succeeding reigns with King A being the latest and King C being the earliest. And let us say we have an absolute calendar date that we can assign to the reign of one king, which for convenience we will call an *anchor date*.[3] In this example, we will say that King A was alive in 1000 BCE as an anchor date. The question is what are the absolute calendar year boundaries for this one trivial chronology?

[3] We call it an anchor date because without at least one absolute calendar year, it would be impossible to secure any dates at all. That may be common sense, and yet such assumptions and technical terminology necessitates being stated and defined.

For Peer Review

With an anchor date that is not tied to a specific regnal year, that anchor date could occur within any of the regnal years of that king's reign. This can resolve into a "low" or "high" chronology for a king. For a "low" chronology, the dates are resolved closer to the present. Thus, when the King A is synchronized with the anchor date (1000 BCE) and given a "low" chronology, his accession date would be 1000 BCE and his date of death (or retirement) would be in 996 BCE. Conversely, with a "high" chronology, dates are resolved further back in the past. Thus, when the King A is synchronized with the anchor date (1000 BCE) and given a "high" chronology, his date of death (or retirement) would be in 1000 BCE and his date of accession would be in 1004 BCE.

Likewise, where there is a variable or ambiguous reign-length (e.g., King B's 2 to 6 years), this variable can be read according to its minimal and maximal limits, which we label "contracted" and "expanded" respectively. A "contracted" reign-length for King B would be 2 years, and an "expanded" reign-length for King B would be 6 years.

Therefore, this example can be mathematically resolved into four permutations based upon the variables expressed above, which express the limits of the chronology. These four permutations are a low contracted (LC), low expanded (LE), high contracted (HC), and high expanded (HE) chronologies, which are calculated as follows:

[TABLE 2]

For Peer Review

In case it is not clear, these chronologies are boundary limits for these particular reigns. Other intermediary chronologies, e.g., King B could have reigned for 3, 4, or 5 years, are also possible. However, the optimized-permutation algorithm we use in our project does not require the calculation of intermediate chronologies because the synchronisms are tested according to the range of the boundaries.

Thus, we arrive at our proposition #1: **Any chronology that has a variable or unknown length in a sequence and is anchored to an absolute year by synchronism can be expressed as a set of boundary permutations.** The results from that set of permutations can be abbreviated using error notation. For example, if we consolidate these four chronologies into one abbreviated chronology with errors, that gives us the following:

[TABLE 3]

As can be observed, these permutations vary dramatically and, when projected over centuries, the resulting error compounds quickly. But let us take our previous example and add to it a King D from a different kingdom that reigned 3-5 years and had an anchor date of 1009 BCE. King D's chronologies look like: 1009-1006 BCE (LC), 1009-1004 BCE (LE), 1012-1009 BCE (HC), and 1014-1009 BCE (HE). And let us say there is a synchronism (like a letter sent between the two kings) showing that King A and King D were contemporaries, i.e., there was at least one day when both lived at the same time. Then the picture changes. The only possible overlap date for King A and King D is 1004 BCE. Our chronology example then becomes:

For Peer Review

[TABLE 4] [4]

So, we can see that the use of a synchronism reduced the error in the reign of King A from ±2 to a firm date, and the error of King C was reduced from ±4 years to +2/-3 years even though we do not know how the date of 1009 BCE fits into the dating of King C. Therefore, proposition #2: **Where there is uncertainty in the reign-length of a king, the use of synchronisms is a practical method of permutation reduction (so-called error reduction) in chronology.**[5] The inability to recognize the interdependence of the data is the principal failure to understand how chronology works. Traditionally, chronological method has been treated like any reductionist discipline where each king-list is like an isolated staircase of succeeding reigns which, if one determines the top step then there is a cascade effect, i.e., the "dead-reckoning" method (Höflmayer & Cohen 2017, 2). That method still works and still has its place. But the dead-reckoning method alone does not work well when the size of each "step" is only an imprecise estimate.

For Peer Review

[4] We use an asymmetric error notation that differs from strict mathematical error notation since only whole (integer) years are significant, not partial years. While the accession date for King B would be 1007.5 BCE ±2.5 years, using strict mathematical error notation in this manner would be impractical and confusing as 1007.5 would be either rounded up to 1008 (by way of strict mathematics) or rounded down to 1007 (because June 1007 BCE is still in the year 1007 BCE). If it were 1008 (±2) BCE, that would lead to a possibility for the latest potential date being 1006 BCE, when 1005 BCE is also possible. If it were 1007 (±2) BCE, that would lead to a possibility for the earliest potential date being 1009 BCE, when 1010 BCE also possible. Using asymmetric error notation has precedent when perfect symmetry of errors creates misleading results (Barlow 2003, 255).

[5] For more information about single permutation and non-permutation-based methods, see Geeraerts, Levy & Pluquet 2017, 13:1-13:18 and Levy et al. 2021, 1-27.

Instead, the king-lists should be viewed more like the gears of a chronology "clock" where the mechanism will not work unless each gear is in the correct size and of the correct tooth ratio and where each gear enables, inhibits, or limits the movement of every other gear. One "gear" of a chronology may be locked in a place by a synchronism that occurred in some other part of the chronology. It is an exercise that seems to demand not only training in the broader view of ancient Near Eastern history but also proficiency in doing large numbers of calculations. When one dates with multiple chronological sequences together with error calculation and consistency checking, efforts to attempt this by hand soon encounter the overwhelming complexity of the task. And if that chronology encompasses a broad scope of the ancient Near East involving hundreds of kings and synchronisms, the task of calculating a chronology can involve millions of calculations.

COMPUTER ASSISTED CHRONOLOGY

Given the problems that beset the modern practice of chronology and the overwhelming number of mathematical operations needed, it became clear that the problem cannot be resolved without the assistance of computers. The specifications to do modern chronology requires repetitive mathematical tasks, a function computers are ideally suited for. Moreover, the task of chronology validation is a task that few people could even perform well because of the computational burden needed. However, computers are ideally suited for the task. Those requirements provided the inspiration for the *Groundhog: Chronology Test Laboratory* project

For Peer Review

(Figure 1).[6]

[FIGURE 1]

Groundhog is an online web application software project that was designed to computationally calculate historical chronologies and check to see if they are internally consistent. The software is a suite of computer programs, written in HTML5/JavaScript, Python, and MySQL, and a distributed computing architecture that was designed specifically to calculate the absolute calendar years of ancient Near Eastern chronologies using synchronisms. The project was started in 2014 at the University of Liverpool and was completed in 2016. The software is a three-tier solution that was developed to remove much of the manual process involved in chronological calculation.

Currently, the project is hosted on an Ubuntu 16.04.7 LTS system running a GNU/Linux 4.4.0-210-generic kernel. The hardware is an AMD FX-8350 8-core processor running at 4.0 GHz with 4GB of RAM. The equipment is not exactly state-of-the-art according to 2024 standards; however, the *Groundhog* software has been optimized to be CPU intensive but memory efficient. So, it adapts well to both small platforms and large distributed computing environments.

[6] The home page for the project is at https://www.groundhogchronology.com/. The project is an online web application that can be used by the public. Details about how to access and use the application are available in the user manual found at https://www.groundhogchronology.com/GroundhogUserManual.pdf. As of this writing, *Groundhog* is currently at version 146. The software has no formal release cycle and changes are made whenever the programmer has time to update it. There are no plans to open source the software, although people are free to use the software to make chronologies of their own.

For Peer Review

On the front end, a node-based user interface written in HTML5/JavaScript was developed to streamline the task of data input. The *Groundhog* user interface allows the input of two kinds of object data: person-objects ("kings") and event-objects ("synchronisms"). Person-objects are associated by either the exact number of years reigning or a possible range of years, i.e., an *or-conditional*. Person-objects are organized inside of container-objects ("dynasties"). Container-objects in *Groundhog* are primarily used to help the human operator organize their person-objects into sequences that *Groundhog* uses to calculate a range of absolute dates, but they also provide constraints and links to other person-objects and container-objects. Finally, event-objects are the synchronisms that the software uses to create and test the proposed chronologies.

Besides the front end, the data is stored in a MySQL database and a second program written in Python 2 is run to generate the boundary limit permutations for a chronology, assigning absolute dates and checking each permutation for consistency. The consistency result is a binary outcome that is either a "consistent" or "inconsistent" value. If at least one permutation is shown to be consistent, the chronology is assigned a value that is "consistent." If all the permutations fail to be internally consistent, then the value is assigned a value that is "inconsistent." In the event where the inconsistency is only off by factors of one or two years, which can be caused by rounding errors or the difference between calendar systems, the operator is notified through either an "inconsistent (marginal=1)" or "inconsistent (marginal=2)" result. This tells the operator that the chronology is internally inconsistent but only marginally by a factor of 1 or 2 years. This allows the operator to fix the chronology where appropriate and run the test again.

For Peer Review

A third program, also written in Python 2, calculates the final abbreviated chronology, low and high chronologies,[7] and the error (the +/- difference between high and low chronologies). The third program is run whether the chronology is internally consistent or inconsistent because it does the overall consistency checking and supplies error information back to the human operator through the color-coding of the workspace (please, refer to the user manual for details on the color-coding system).[8]

Now, let us describe how the program determines consistency. Event-objects, that is, synchronisms, are assumed to be as close to objective data as possible. Dates for the person-objects are both derived from and tested by the event-objects. Base dates are established *prima facie* beginning with at least one event-object with a class 0 rule, that is, a synchronism that anchors a person-object to an absolute year. Then the person-objects are given dates that are most consistent with the synchronisms. The *Groundhog* software gives the chronology the benefit of any doubts when it comes to consistency. In other words, the software steel-mans the potential for each permutation to test as consistent. However, the software does not arbitrate when there are two event-objects that are inconsistent with each other. That requires a human operator to judge. The software only points out that there is an inconsistency. It cannot resolve an inconsistency, nor can it recommend how to fix an inconsistency. It only detects an inconsistency.

For Peer Review

[7] While any chronology where ambiguity exists produces four chronologies, we have a stated previously that those same four chronologies can be collapsed into a single abbreviated chronology with error notation (see proposition #1).

[8] https://www.groundhogchronology.com/GroundhogUserManual.pdf

For example, let us say we have a King Robert who reigned for 10 years, and two event-objects: one that places his first regnal year in the year 1000 BC and the other that places his first regnal year in the year 1001 BC. This chronology is internally inconsistent. Both years cannot be this king's first regnal year. But it is impossible for the computer to decide which is the correct event-object. The program will arbitrarily select one of the two event-objects to build the chronology and mark the other event-object as inconsistent. Once a chronology is marked as inconsistent, it is up to the human operator to decide how the chronology is to be fixed. Overall, such a chronology would be deemed "inconsistent (marginal=1)" because not all the person-objects passed a consistency check against all the event-objects. The operator will then receive the result and will then have to decide which of the event-objects is the correct one.

But that said, let us say the operator believes the first regnal year of King Robert had to be in either 1000 or 1001 BCE but the operator cannot decide which one it is. We know that the king had to reign from 1000-990 BCE or 1001-991 BCE. Instead of creating two rules that are inconsistent, the better option is to create two rules that are consistent with each other. First, create a reference person-object that reigned from 1000-991 BCE. Then create a class 3 rule[9] that shows the king to have reigned in the first year of the reference object and a second class 3 rule that shows the king to have reigned in the final year of the reference object. This will achieve the same kind of ambiguity desired by the operator without two event-objects contradicting each other.

[9] See the "Data Model" section below for the definitions of these synchronism rules.

For Peer Review

Overall, the *Groundhog* software does not take the decision-making prerogative out of the hands of the chronologist, but instead helps the chronologist to make better decisions by enforcing strict rules of consistency.

OPTIMIZED-PERMUTATION ALGORITHM

While the aforementioned information describes the optimized-permutation algorithm, we understand that this may not be an intuitive description of how the software generates a chronology. In such cases, walking through the algorithm with visual aids may explain how the software constructs chronologies better than mere words. Below is a screen shot (Figure 2) of an ambiguous chronology like the King Robert example used previously. And in this example, we have all the object-types (containers, persons, and events) and examples of some of the accepted datatypes.

[FIGURE 2]

As soon as the human operator presses the "Rebuild Chronology" button, the rebuild process begins. The first step is that the software examines all the person-objects for ambiguity in this chronology. To do this the program runs through all the person-objects to see if there are any *or-conditionals.*[10] In this case, there is one *or-conditional*. King Jones reigned for 10 to 14 years. This means that there are two permutations for this chronology (n=2).

[10] See the "Data Model" section below for the definitions of person-object datatypes.

For Peer Review

The bifurcation of permutations is how the software handles the ambiguity of contracted and expanded chronologies. In this example, Permutation 1 is the contracted chronology and Permutation 2 is the expanded chronology (Table 5).

[TABLE 5]

Once the permutations are defined, the system will assign dates to the chronology as follows. It will begin with class 0 rules from event-objects.[11] A class 0 rule ties a specific year of a person-object to an absolute calendar year.[12] In the screen shot, the "Anchor to 1000 BCE" is the only event-object with a class 0 rule. In this case, the first regnal year of the Reference Object is assigned the year 1000 BCE (Table 6).

[TABLE 6]

Following the application of class 0 rules, the system will apply the remainder of the rules in order of their class. In this example, there are no class 1 or 2 rules, but there are two event-objects with class 3 rules. King Robert is not directly linked to an anchor date (a class 0 rule). But two class 3 event-objects link King Robert to the Reference Object labeled "link to last year" and "link to first year." The rule for "link to the last year" states that King Robert ruled during the final year of the reference object, and the rule for "link to the first year" states that King

[11] For a complete description of event-object rules, please see the "Data Model" section.

[12] Multiple class 0 rules can be used within a single chronology; however, if a class 0 rule has already been used to assign dates to a person-object, any subsequent class 0 rules will be used for validation purposes only.

For Peer Review

Robert ruled during the first year of the reference object. Since the Reference Object ruled for 9 years and King Robert ruled for 10 years, a "fitting subroutine" shifts the reign of King Robert early or late to achieve the best possible fit representing high and low chronologies; ergo, either King Robert reigned from 1001 to 991 BCE (high) or from 1000 to 900 BCE (low). The shifting of person-object reigns is limited by the event-object rules and is how the software calculates the high and low chronologies. Thus, the algorithm will not shift the person-objects beyond what is permitted by the synchronisms (Table 7).

[TABLE 7]

Earlier we mentioned that chronologies where there is a point of ambiguity resolve into four fundamental boundary limit chronologies that can be aggregated down into a single abbreviated chronology with error notation (see proposition #1).[13] You should at this point be able to observe that two of the four chronologies (high and low) are being calculated within each permutation, while the permutations themselves account for the other two chronologies (contracted and expanded).

After the rule 3 event-object calculations are completed, any person-object where the dates can be implied from existing assigned dates are filled out. In this case, King Jones immediately

[13] Of course, the more points of ambiguity there are, the more permutations will result that will multiply the high, low, expanded, and contracted instances.

For Peer Review

precedes King Robert (by a yellow-link) so the dates for King Jones are calculated from the reign of King Robert to create the completed two permutations (Table 8).

[TABLE 8]

After all the rules have been applied and all the permutations are propagated with date information, the software begins the validation phase. During this phase, each permutation is tested against all the event-objects. Any permutation that is inconsistent will be marked as such, and the remaining permutations will be aggregated to create the abbreviated chronology (again, see proposition #1). In this example, both permutations are consistent so both will be used for the aggregation. In the event no permutations are consistent, all the permutations will be used to create the aggregation and the result of the chronology will be marked as "Inconsistent."

After the validation phase, the permutations are aggregated (Table 9).

[TABLE 9]

This can be further reduced to an abbreviated chronology with error notation (Table 10):

[TABLE 10]

And this is the result that is displayed on the user interface in the screen shot above. For a data model of how the person-objects and event objects work see the below section. If the data

For Peer Review

cannot produce a consistent result, *Groundhog* flags any event where an inconsistency occurs; a human operator at that point is needed to investigate in order to find where the data is flawed.

TEST RESULTS AND EXECUTION TIMES

Given the complexity of the software and the parameters that can be tested, execution times can vary dramatically. Table 1 shows a sample of test results with execution times taken from chronologies that were built and validated using *Groundhog*.

[TABLE 11]

The first row represents the minimal chronology that can be created in the software (two container-objects, one person-object, and one event-object) which results there being twelve chronological constraints (four per container-object,[14] two per person-object, and two per event-object). Software overhead accounts for most of the 16 seconds of runtime used to build this minimal chronology.

The efficiency of each build can be determined by the *constraints processed per second per core* column as that accounts for not just the total number of calculations done per the runtime but

[14] Container-objects contain three date parameters that need to be updated, propagated to other container or person-objects, and need to be checked for overall consistency. They are also linked to up to four other objects, which means that they are part of constraint checking.

For Peer Review

also accounts for the multithreading done by the software. Thus, we should note that the software set up time is a small loss in efficiency that becomes inconsequential to the overall outcome as the number of constraints being calculated increases.

However, as the data set size increases, the optimized-permutation algorithm becomes more efficient. Small builds tend to run at well under 100 constraints/sec/core. On the other hand, when the build uses over 64 permutations, the efficiency of processing can increase to up to 558.3 constraints/sec/core. This is not an unexpected outcome with large data modelling. With large data modelling, small losses of efficiency with smaller data sets are sacrificed to become much more efficient as the data set grows.

Counterintuitively, we observe that processing times are not strictly correlated to either the number of permutations or the number of objects being processed. The test where n=2 and 938 objects (120 containers, 594 persons, 224 events) had almost the same runtime as the test where n=1024 and 20 objects (120 containers, 594 persons, 224 events). Comparing the last two builds where n=8192, both chronologies use the same containers, persons, and events, and yet the two builds differ in execution time by 34%. The only difference between the two chronologies is the relationships between the objects.

Benchmarking has led us to conclude that processing time increases are not based solely upon the number of items being processed, the types of objects being processed, or the number of threads being processed, but upon the complexity of the relationships between the person-objects and the event-objects, which manifests as increased processing time in the “fitting subroutine.”

For Peer Review

The more complex[15] the relationship is between an object and its constraints (as opposed to just the number of objects alone), the more processing time needed to properly fit a chronological sequence that is not deterministically tied to an anchor date. A computational cost is attached to complexity, and a high number of permutations multiplies the cost.

Therefore, a complex chronology with a lot of ambiguity (where n>4096) could potentially take hours to build.[16] Yet, given the constraints of our use case, that build is being done effectively. We realize that execution times could be inflated by adding unnecessary ambiguity to complex chronologies. At the same time, this does not mean that the system is not doing meaningful work or that the software is functionally useless—one solution is that the human operator needs to rethink a chronology to make its build more efficient.[17] In a similar fashion, one can create fabulously inefficient SQL queries that take days to complete. That is a programmer problem—not an efficiency problem with the SQL database itself. *Groundhog* treats ambiguity (and the resulting permutations created by ambiguity) like any computing resource—a resource that helps you solve certain complex problems that cannot be done currently using polynomial-time algorithms, but it is also not an infinite resource.[18]

For Peer Review

---

[15] *Complexity* is a defined as a measure of the number and types of constraints that need to be considered when deriving a section of a chronology. The more constraints and the less deterministic those constraints, the more complex the relationship. For example, an object that is resolved by a single Year = A(y) relationship will be more deterministic and have a less complexity than an object determined by two A ⊇ B relationships.

[16] We have encountered only one chronology that took more than two hours to build. So, extended processing times appear to be an outlier in actual practice.

[17] Within *Groundhog* there are multiple ways to handle this situation should a chronology need to accommodate additional ambiguity, e.g., upgrading the current (admittedly obsolete) host system or adding parallel-compute nodes. Also, an ambiguity limiter could be added to the code letting the human operator know that their chronology needs to be disambiguated. But that is a step we have not needed to take thus far.

[18] It should go without saying that all computing resources are finite.

DATA MODEL: REDUCED SYNTAX PARADIGM

The data model used for *Groundhog* follows a reduced syntax paradigm. That means the minimum syntax is used to express complex datatypes. This works well for our particular use case in that we can roll many datatypes into a simplified instruction set without having to needlessly multiply relationship intervals. The applicable datatypes have been spread out between person-objects that define expanded and contracted chronological data, and event-objects that are used to define high and low chronological data, i.e., complexity.

For Peer Review

For person-objects, there are only three datatypes: static, inclusive or-conditional, and exclusive or-conditional.

[TABLE 12]

For event-objects, each event is defined by a rule. Rules are implemented using a four-field colon-separated notation. While this notation seems simple, these rules cover a wide range of possible synchronistic relationships. The notation uses the following fields **[class]:[king**

1]:[king 2]:[year] . Each class of synchronism uses different fields. The king fields (fields 2 and 3) can take one of three formats: the identifier number of the person (e.g., "31" where 31 equals "King Robert"),[19] the identifier anchored to accession year (e.g., "31b0" where this represents the first regnal year of King Robert), and the identifier anchored back from the death year (e.g. "31e1" where this represents the year before King Robert died). For "b" and "e" operators, it should be noted that the count begins at zero.

Table 13 maps out all the synchronistic data types used in *Groundhog*.

[TABLE 13] [20]

RELATED WORKS

The project that is most like *Groundhog* in terms of functionality is *Chronolog* (https://chrono.ulb.be/) (Levy et al. 2021, 1-27). *Chronolog* and *Groundhog* are similar in that they both take person-objects and synchronisms and generate chronologies from them. And

[19] To find the id number for a person-object in *Groundhog*, hover the mouse reticle over the person-object.

[20] Much of this notation has been used from (Levy et al. 2021, Table 2 and 5), which is based upon Allen's interval algebra (Allen 1983, 832-843). Additional notation has been added where datatypes may not be covered. The *Groundhog* project has developed its own notation; however, the reader may be more familiar with Allen's or the Levay et al. notation. While there is overlap with Allen's interval algebra, *Groundhog* event-object rules cover a wide number of situations specific to chronological work that are not expressed by interval algebra notation.

For Peer Review

while they may seem very similar on the surface, the projects differ in terms of their use cases, design philosophies, and project goals.

The *Groundhog* project is slightly older than *Chronolog,* having begun in 2014 and presenting a paper with fully operational software at the April 2017 *American Research Center in Egypt* annual conference (Falk 2017). *Chronolog* presented its concept paper at the 2017 TIME conference in Oct 2017 and completed a software product in 2019.

Chronolog was originally designed to do general chronological work on archaeological sites particularly with a focus on creating chronologies from the cross-dating of dendrochronology,[21] ceramic typologies, and stratigraphy and is therefore optimized for the archaeological discipline with an end-user experience in mind. The software is a self-contained JRE jar file that can be easily downloaded and run on any system. The GUI uses a plain widget set but with lots of help windows to assist the user. Results are instantaneous and the interface is snappy and responsive. Results and chronologies are saved to a local text file in JSON format.

In comparison, *Groundhog* was designed to create chronologies of the ancient Near East from king-lists, epigraphic sources, and synchronisms and is optimized for the work of historians and chronologists. The software is a web application run on a centralized server. The GUI uses a modern full-color HTML5/JavaScript graphical interface using a node-based design philosophy.

[21] e.g., OxCal. Nothing prevents *Groundhog* from using dendrochronological data. However, integration of this data has not been a priority for the project since dendrochronology has well-known problems when applied the archaeology of the ancient Near East (Beitak 2016, 77).

For Peer Review

Unlike *Chronolog,* the *Groundhog* GUI implements container-objects that are used to organize and minimize large groups of objects. Results are obtained by rebuilding the chronology as needed, which admittedly takes a lot longer to complete than on *Chronolog*. And the data for *Groundhog* is stored in a relational database (MySQL) allowing for the use of large data sets.

Perhaps the most important difference between the two projects is the underlying algorithms used to compute the results. *Chronolog* uses a "polynomial-time algorithm," whereas *Groundhog* uses an "optimized-permutation algorithm." These are apples and oranges descriptors. The *Chronolog* descriptor tells you that their algorithm is fast without telling you what the algorithm does, while the *Groundhog* descriptor tells you what the algorithm does without telling you how fast it is. This *prima facie* reveals a philosophical difference between the two development teams. With that said, we can infer some things about the algorithms based upon each algorithm's limitations and strengths.

The *Chronolog* polynomial-time algorithm works exceptionally well for correlating archaeological typologies to a chronological framework and with data sets where contemporaneity is presumed (Geeraerts, Levy & Pluquet 2017, 13:18). And when the datatypes have been properly curated, their algorithm is extremely fast. Yet, for some chronological applications that involve high degrees of ambiguity, when natural endpoints to a chronology are unknown, where contemporaneity cannot be presumed, and when certain datatypes are required, e.g., *or-conditonals,* the *Groundhog* optimized-permutation method is a better option.

For Peer Review

And while *Groundhog* was criticized in the *Chronolog* article, many of those criticisms seem to lack warrant. For example, they claim that "our model allows for more diverse types of chronological constraints [than Falk's model]" (Levy et al. 2021, 2), but they do not spell out what datatypes are absent from the *Groundhog* model. *Groundhog* accepts the datatypes that the *Chronolog* polynomial-time model accepts as well as several datatypes it cannot accept.

For example, Levy et al. state "Our model also presents a number of limitations. For example, it cannot model a reign of '5 or 15' years" (Levy et al. 2021, 4), and yet the *Groundhog* optimized-permutation model was explicitly designed to handle that kind of ambiguity. However, to handle such datatypes ventures into the ambiguity of contracted/expanded chronologies, which is most easily resolved using permutations. Many algorithmic methods were evaluated in the concept phase of *Groundhog* and an optimized-permutation algorithm had the scope demanded by our use case.

Another type of data that *Chronolog* cannot handle is the asynchronism. "An example of such a Chronological Relation is the *General Asynchronism*, defined as '*A* ends before the start of *B* or *A* starts after the end of *B*'. The reason for limiting ourselves to *and*-relationships is in order to be able to analyse the network using fast algorithms…" (Levy et al. 2021, 4). The *Chronolog* team essentially admits that their algorithmic speed comes at the expense of datatype acceptance. *Groundhog* accepts these datatypes (*or-conditionals* and *not-conditionals*) that *Chronolog* rejects on account of algorithmic efficiency. Again, per the *Chronolog* team "All other operators,

For Peer Review

including '**or**' and '**not**' are disallowed." (Levy et al. 2021, 11).

Moreover, the *Chronolog* team has claimed that "Second, Falk's approach relies on exhaustive search, by generating all possible combinations of dates, thus yielding exponential-time algorithms, whereas we employ a more efficient approach, using polynomial-time algorithms… this means that Falk's approach is unlikely to be able to handle networks of large sizes in short processing time." (Levy et al. 2021, 2). This criticism is curious on several levels. On the one hand, the criticism conflates our ambiguity-handling functionality (using permutations) with how the software generates each individual permutation. On the other, the *Chronolog* software is not a network-enabled application, so the appeal to "networks of large sizes" is odd. From what I can best gather, when they use "networks of large sizes," they really mean "a large assemblage of data."

Furthermore, the *Chronolog* team does not define what are "networks of large sizes" except perhaps as "several hundred chronological constraints," (Levy et al. 2021, 2) or what is an unacceptable "processing time." Regardless, our algorithm does not generate all possible dates, but the only the boundary limit dates. So, the *Groundhog* algorithm is not an exponential-time algorithm. Apart from the limited use of permutations for doing contracted and expanded ambiguity, which *Chronolog* is incapable of doing, there is little effective difference between *Chronolog* and *Groundhog* regarding how dates are calculated for a single permutation, except that *Groundhog* allows for more complexity in its datatypes. The actual processing time for a single permutation of a *Groundhog* build usually no more than a few seconds (once software

For Peer Review

overhead ceases to be a factor), which is not significantly more than with *Chronolog*.[22] Nevertheless, the philosophy of the *Groundhog* project is that longer processing times are a small price to pay for solving more difficult problems.

As stated previously by their development team, the *Chronolog* algorithm buys computational efficiency through elimination of difficult datatypes. That is a sound strategic decision for the type of work for which they are optimizing. But it does not mean that the alternative does not have utility for a use case with a broader scope. We believe an optimized-permutation algorithm is a good compromise between exponential-time algorithms and polynomial-time algorithms for use cases where tolerance for ambiguity is required.

Additionally, the *Chronolog* team claims "Our technique can scale and handle networks with several hundred chronological constraints in less than a second, allowing for a truly interactive experience for the user" (Levy et al. 2021, 2). *Groundhog* is not designed to handle hundreds of chronological constraints, but tens of thousands of constraints per permutation and millions of chronological constraints overall,[23] and is designed for serious large-data chronological work. Unlike *Chronolog* that can only be run on a single CPU and has no networking capability,

For Peer Review

[22] If one considers the "worst case" scenario from Table 1, the entire execution time was 9314 seconds. This build performed 8192 permutations or 1.13 secs per permutation. It should be noted that *Chronolog* only uses a single permutation to arrive at its outcomes. So, for this to be an apples-to-apples comparison, building a single *Groundhog* permutation should be the basis of comparison with *Chronolog*. After all, when *Groundhog* is building with multiple permutations, the *Groundhog* software is performing an extended functionality that *Chronolog* is incapable of performing.

[23] cf. Table 1, bottom line.

Groundhog's architecture is multithreaded and has facilities to allow for parallel-computing operations should the need ever arise.

When it comes to comparing *Chronolog* to *Groundhog,* they are more dissimilar than they are alike. They are designed and optimized for different kinds of work. *Chronolog* is software product designed for onsite workloads where single threading on a personal computer can suffice. As such, it is ideal for the kind of work it was designed to do, which is to create chronologies from archaeological typologies and assemblages in the field. *Groundhog* is a web application designed to carry forward ongoing research questions. As such, it was intended to calculate large data sets with processing capabilities that can be expanded to a large network of servers[24] and is designed to meet the needs of historians and chronologists though broad acceptance of ambiguous datatypes.

For Peer Review

DISCOVERIES

Having had to build the *Groundhog* project from the ground up, we ended up learning a lot about the theory and practice of chronology. We learned that not every algorithm is appropriate for the kind of work you want to specialize in. And some of what was learned may seem intuitive but has never been explained formally in academic literature, e.g., that ambiguous reign-lengths

[24] While *Groundhog's* parallel-computing capabilities have been tested across a group of five servers, the need has not yet arisen to where we must expand the computing capabilities beyond its current host system.

result in contracted/expanded chronologies that differ from the often mentioned high/low chronologies, even if the former can in practice be rolled into the latter. However, other minor discoveries have been made over the last decade.

(1) Subjective and objective data.

This was more of an observation that arose as a result of the software development project, but an observation that has not been carefully considered in the academic literature. When dealing with chronologies, as briefly alluded to in the section on "Computer Aided Chronology," we are dealing with two kinds of data objects: persons ("kings") and events ("synchronisms") that could tie two persons as being contemporary.

We observed across various proposed chronologies that the proposed reign-lengths for the persons were subjective. Proposed reign-lengths can vary among chronologists. As a result, the absolute dates for a chronology can vary considerably from one chronologist to another. However, we also observed that any discovered synchronisms generally were agreed upon by consensus, as long as (a) the source text could be read without controversy, and (b) both the sender and the recipient were mentioned by name in the source. This is, of course, not withstanding special pleading, question-begging, or stands of invincible ignorance (essentially ignoring the wealth of existing data) that has been the hallmark of revisionist chronologies;[25]

[25] "Revisionist chronologies" are chronological theories that are fringe or outside what is acceptable to the academic consensus. Examples of revisionist chronologies include the "new chronology" of David Rohl and the *Worlds in Collision* "revised chronology" of Immanuel Velikovsky.

For Peer Review

however, as noted by Hornung such hypotheses are not based upon the historical sources (Hornung 2006, 15).

Therefore, we have one type of data (persons) that is the subjective data to be tested (the hypothesis), and another type of data (events) that is a kind of objective data against which any chronological hypothesis could be tested. And in knowing that the data could be sorted according to subjective and objective types means that a testing framework for a chronology is at least possible if enough of the appropriate kinds of synchronisms were available in the archaeological and historical record.

Before the project was begun, we were not sure if enough synchronisms would be available to do viable chronology calculation and testing. At the time of the writing of this article, 190 synchronisms meeting the requisite criteria have been added to the verification database, which are being used to test against the reigns of 538 kings, with a potential for an estimated 50-200 more synchronisms that would meet the criteria stated previously yet to be added.

(2) Extended Inconsistency Effect.

One of the observations in the running *Groundhog* simulations is that, when inconsistency is observed, inconsistency is not always observed near where an initial change was made. Often the inconsistency shows up in a king-list that may be removed from the altered sequence by one or two degrees of separation. This happens because changing a single king-list moves the

For Peer Review

synchronisms in relation to other king-lists, which in turn creates inconsistencies further down the line.

An example of extended inconsistency effect is the case of Ninurta-apal-ekur where sources indicate that his reign could be either 3 or 13 years (Glassner 2004, 143 and 155 fnote 56). While most Assyriologists hold to the short date, some scholars still argue for the long date to get some wiggle room in the Mesopotamian king-lists. However, *Groundhog* testing showed that the 13-year value creates inconsistencies with the synchronisms found between Ugarit and the Hittite Empire.

Even though this raised the question of whether an alternate Hittite chronology would affect the outcome,[26] the possibility that an inconsistency can manifest two or three degrees of separation from the original chronology itself demonstrates that an inconsistency, which may not be immediately apparent to the researcher, can manifest in a proposed chronology. In order for the researcher to discover such an inconsistency would require the researcher to consider the broadest implications of any chronology. Thus, creating a chronology that is isolated within a disciplinary silo is no longer feasible given the number of synchronisms available today.

(3) Unpredictable Significance.

[26] It should be noted that testing against the other chronological possibilities for the Hittite Empire did not change the outcomes when compared to the Assyrian Standard Chronology.

For Peer Review

This is another of the discrete results that came out because of computer assisted analytics for chronology. This has shown that humans are not good at determining when synchronisms are most significant (i.e., curated data). When one asks chronologists, what are the most significant synchronisms for the interaction between the Hittites and Ugarit? Many will answer the synchronisms between Suppiluliuma II and Ammurapi, Mursili II and Niqmaddu II, or even Suppiluliuma I and Niqmaddu II. However, few if any will say that the most important synchronisms are between Tudhaliya IV and Ammittamru II or Mursili II and Niqmepa, even though *Groundhog* testing has shown that those two synchronisms lock the two king lists to a span of +/- 11 years, which is a surprisingly small amount in the field of ancient chronology.

CONCLUDING REMARKS

While the use of computers for chronology is still in its infancy, results have already emerged that could stabilize the chronological debate. *Groundhog's* role was to break the ground with new methods of doing chronology and chronological research. As such the *Groundhog* project was never intended to be a software product or an end unto itself. It is a tool designed for serious scholars with a mind towards keeping the complexities of chronology intact. The project remains a work in progress in that it is constantly changing to meet the demands and challenges that face the modern chronologist. Initial experiments have weighed in on several chronological controversies, and those results are published with the data made publicly available online for third parties to examine (Falk 2020, 99-111).

For Peer Review

Groundhog's contribution to the field is in revealing the full complexity that creating a robust, internally consistent chronology based on comparative synchronisms requires. While specialized field apps have their purpose, *Groundhog* is uniquely designed to address the ambiguities of real-world historical data. By providing an evidential approach using cross-cultural data that gives chronological method soundness, *Groundhog* invites scholars to come out of their silos to craft better, more accurate historical chronologies.

For Peer Review

BIBLIOGRAPHY

Allen, J. F. (1983) 'Maintaining Knowledge about Temporal Intervals', *Communications of the ACM,* vol 26, pp. 832-843

Aston, D. (2013) 'Radiocarbon, Wine Jars and New Kingdom Chronology', *Ägypten und Levante,* vol 22/23, pp. 289-315.

Barlow, R. (2003) 'Asymmetric Errors', *PHYSTAT2003, SLAC, Standford, California, September 8-11, 2003*, pp. 250-255.

Beitak, M. (2016) 'Antagonisms in Historical and Radiocarbon Chronology' in A. J. Shortland & C. B. Ramsey (eds), *Radiocarbon and the Chronologies of Ancient Egypt,* Oxbow Books, Oxford, pp. 76-109.

Collon, D. (2007) 'Babylonian seals' in G. Leick (ed), *The Babylonian World,* Routledge, New York, pp. 95-123.

Creasman, P. P. & Kamrin J. (2016) 'Kings, Chronologies, and Confusion' paper presented at the conference of the 2016 American Research Center in Egypt.

Falk, D. A. (2017) 'Groundhog: Preliminary Results of a Computational Method for Validating Chronologies.' paper presented at the conference of the 2017 Annual Meeting of the American Research Center in Egypt.

Falk, D. A. (2020) 'Computer Analytics in Chronology Testing and Its Implications for the Date of the Exodus' in R. E. Averbeck and K. L. Younger (eds), *"An Excellent Fortress for His Armies, a Refuge for the People": Egyptological, archaeological, and biblical studies in honor of James K. Hoffmeier,* Eisenbrauns, University Park, pp. 99-111.

Geeraerts, G., Levy E. & Pluquet F. (2017) 'Models and Algorithms for Chronology' in S. Schewe, T. Schneider and J. Wijsen (eds), *Proceedings of The 24th International Symposium on Temporal Representation and Reasoning* (TIME 2017), Dagstuhl Publishing, Dagstuhl, pp. 13:1-13:18.

Glassner, J-J. (2004) *Mesopotamian Chronicles,* Society of Biblical Literature, Atlanta.

Höflmayer, F. & Cohen S. L. (2017) 'Chronological Conundrums: Egypt and the Middle Bronze Age Southern Levant', *Journal of Ancient Egyptian Interconnections,* vol 13, pp. 1-6.

Hornung, E. (2006) 'Introduction' in E. Hornung, R. Krauss and D. Warburton (eds), *Ancient Egyptian Chronology,* Brill, Boston, pp. 1-16.

For Peer Review

Kitchen, K. A. (2006) 'The Strengths and Weaknesses of Egyptian Chronology—A Reconsideration', *Ägypten und Levante,* vol 16, pp. 293-308.

Kitchen, K. A. (1991) 'The Chronology of Ancient Egypt', *World Archaeology,* Taylor & Francis, Ltd., New York, pp. 201-208.

Kitchen, K. A. (1987) 'The basics of Egyptian chronology in relation to the Bronze Age' in P. Åström (ed), *High, middle or low? Acts of an international colloquium on absolute chronology held at the University of Gothenburg 20th - 22nd August 1987,* Paul Åströms Förlag, Gothenburg, pp. 37-55.

Levy, E., Geeraerts G., Pluquet F., Piasetzky E. & Fantalkin A. (2021) 'Chronological networks in archaeology: A formalised scheme', *Journal of Archaeological Science,* vol 127, pp. 1-27.

Manuelian, P. der (1987) *Studies in the Reign of Amenophis II,* Gerstenberg Verlag, Hildesheim.

McIntosh, J. R. (2005) *Ancient Mesopotamia: New Perspectives,* ABC-CLIO, Santa Barbara.

O'Mara, P. F. (2003) 'Censorinus, the Sothic Cycle, and Calendar Year One in Ancient Egypt: The Epistemological Problem', *Journal of Near Eastern Studies,* vol 62, pp. 17-26.

Pruzsinszky, R. (2009) *Mesopotamian Chronology of the 2nd Millennium B.C.: An Introduction to the Textual Evidence and Related Chronological Issues,* Österreichische Akademie der Wissenschaften, Vienna.

Shaw, I. (2003) 'Chronology' in I. Shaw (ed), *The Oxford History of Ancient Egypt,* new ed., Oxford University Press, New York, pp. 480-489.

Shaw, I. (2003) 'Introduction: Chronologies and Cultural Change in Egypt' in I. Shaw (ed), *The Oxford History of Ancient Egypt,* new ed., Oxford University Press, New York, pp. 1-15.

Thiele, E. R. (1965) *The Mysterious Numbers of the Hebrew Kings: A Reconstruction of the Chronology of the Kingdoms of Israel and Judah,* 2nd ed, Paternoster Press, Exeter.

For Peer Review

TABLE 1: A FICTITIOUS EXAMPLE CHRONOLOGY

King A reigned 4 years
King B reigned from 2 to 6 years
King C reigned 6 years

For Peer Review

TABLE 2: CHRONOLOGICAL LIMITS OF THE FICTITIOUS CHRONOLOGY

Chronology 1 (Low Contracted)	Chronology 2 (Low Expanded)
King A from 1000-996 BCE	King A from 1000-996 BCE
King B from 1002-1000 BCE	King B from 1006-1000 BCE
King C from 1008-1002 BCE	King C from 1012-1006 BCE
Chronology 3 (High Contracted)	**Chronology 4 (High Expanded)**
King A from 1004-1000 BCE	King A from 1004-1000 BCE
King B from 1006-1004 BCE	King B from 1010-1004 BCE
King C from 1012-1006 BCE	King C from 1016-1010 BCE

For Peer Review

TABLE 3: CONSOLIDATED FICTITIOUS CHRONOLOGY WITH ERRORS

King A from 1002(±2)-998(±2) BCE
King B from 1006(±3)-1002(±2) BCE
King C from 1012(±4)-1006(±3) BCE

For Peer Review

TABLE 4: THE FICTITIOUS CHRONOLOGY CORRELATED TO A SYNCHRONISM

King A from 1004(±0)-1000(±0) BCE
King B from 1008(+2/-3)-1004(±0) BCE
King C from 1014(+2/-3)-1008(+2/-3) BCE

For Peer Review

TABLE 5: INITIAL PERMUTATIONS

Permutation 1
Reference Object=9 years
King Robert=10 years
King Jones=10 years

Permutation 2
Reference Object=9 years
King Robert=10 years
King Jones=14 years

For Peer Review

TABLE 6: PERMUTATIONS AFTER RULE 0 (ANCHOR DATE)

Permutation 1
Reference Object=1000-991 BCE (9 yr)
King Robert=10 years
King Jones=10 years

Permutation 2
Reference Object=1000-991 BCE (9 yr)
King Robert=10 years
King Jones=14 years

For Peer Review

TABLE 7: PERMUTATIONS AFTER RULE 3 SYNCHRONISMS

Permutation 1
Reference Object=1000-991 BCE (9 yr)
King Robert=1001-991 BCE [high] or 1000-900 BCE [low] (10 yr)
King Jones=10 years

Permutation 2
Reference Object=1000-991 BCE (9 yr)
King Robert=1001-991 BCE [high] or 1000-900 BCE [low] (10 yr)
King Jones=14 years

For Peer Review

TABLE 8: COMPLETED PERMUTATIONS

Permutation 1
Reference Object=1000-991 BCE (9 yr)
King Robert=1001-991 BCE [high] or 1000-900 BCE [low] (10 yr)
King Jones=1011-1001 BCE [high] or 1010-1000 BCE [low] (10 yr)

Permutation 2
Reference Object=1000-991 BCE (9 yr)
King Robert=1001-991 BCE [high] or 1000-900 BCE [low] (10 yr)
King Jones=1015-1001 BCE [high] or 1014-1000 BCE [low] (14 yr)

For Peer Review

TABLE 9: PERMUTATION AGGREGATION

Reference Object=1000-991 BCE (9 yr)
King Robert=1001-991 BCE [high] or 1000-900 BCE [low] (10 yr)
King Jones=1011-1001 BCE [high] or 1010-1000 BCE [low] (10 yr)
King Jones=1015-1001 BCE [high] or 1014-1000 BCE [low] (14 yr)

For Peer Review

TABLE 10: ABBREVIATED CHRONOLOGY WITH ERROR NOTATION

Reference Object=1000-991 BCE
King Robert=1001(0/-1)-991(0/-1) BCE
King Jones=1013(+2/-3)-1001(0/-1) BCE

For Peer Review

TABLE 11: EXECUTION TIMES ON THE HOST SYSTEM

Ambiguity/ Permutations (n)	Number of objects	Number of chronological constraints	Total number of constraints calculated
1	4 (2 containers, 1 person, 1 event)	12	12
1	133 (2 containers, 1 person, 130 events)	270	270
2	6 (2 containers, 2 persons, 2 events)	16	32
2	134 (2 containers, 2 persons, 130 events)	272	544
2	938 (120 containers, 594 persons, 224 events)	2,116	4,232
4	135 (2 containers, 3 persons, 130 events)	274	248
8	136 (2 containers, 4 persons, 130 events)	276	2,208
16	137 (2 containers, 5 persons, 130 events)	278	4,448
32	138 (2 containers, 6 persons, 130 events)	279	8,928
64	139 (2 containers, 7 persons, 130 events)	280	17,920
64	800 (112 containers, 501 persons, 112 events)	2,900	185,600
512	925 (141 containers, 545 persons, 239 events)	1,893	969,216
1,024	20 (4 container, 12 persons, 4 events)	80	81,920
1,024	844 (116 containers, 538 persons, 190 events)	3,376	3,457,024
8192	838 (114 containers, 537 persons, 187 events)	3,352	27,459,584
8192	838 (114 containers, 537 persons, 187 events)	3,352	27,459,584

For Peer Review

Constraints processed per second per core	Time to completion (sec)
0.75	16
12.27	22
0.7272	22
16	17
78.37	27
6.545	22
12.55	22
32.7	17
69.75	16
101.8	22
95.87	242
367.1	330
465.5	22
361.3	1196
558.3	6148
368.5	9314

For Peer Review

TABLE 12: PERSON-OBJECT "KING" DATATYPES

Datatype	Description	Ambiguity
Static	Defines the exact number of years a person reigned	No
Inclusive or-conditional	Defines a range of years from x to y.	Yes
Exclusive or-conditional	Defines a reign as either x or y years.	Yes

For Peer Review

Example	Image
34 "34 Years"	
10t14 "10 to 14 years"	
10v14 "10 or 14 years"	

For Peer Review

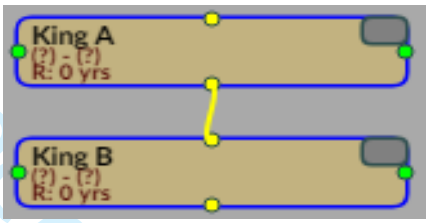
TABLE 13: EVENT-OBJECT "SYNCHRONISM" DATATYPES

Class	Description	Example
0	Anchor one regnal year to an absolute calendar year. Third field not used.	**0:31b5::1000** "The sixth year (5+1) of King A (id:31) was in 1000 BCE"
1	An absolute year occurs during the reign of King A. "b"/"e" notation is not supported. Third field not used.	**1:31::1000** "1000 BCE occurs during the reign of King A (id:31)"
2	The specific year of King A is equal to the specific year of King B. This is a unidirectional synchronism that can be used for flow control. The second field is the source, and the third field is the target. Hence, A(y) →= B(y), but not A(y) =← B(y). Fourth field not used.	**2:21b18:97b1:** "The 19th year of King A (id:21) is equal to the 2nd year of King B (id:97)."

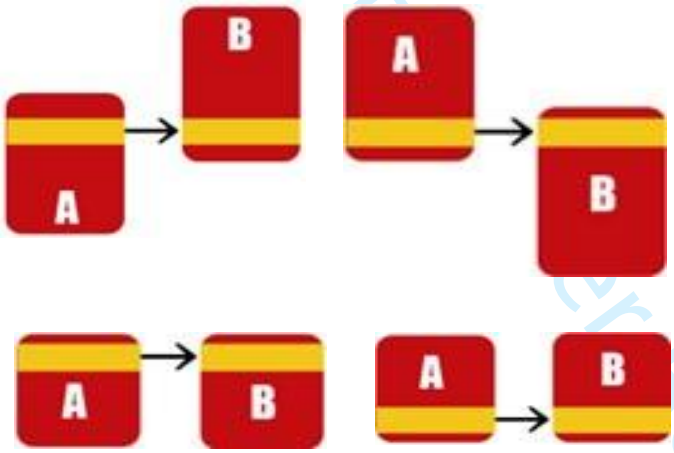
For Peer Review

		3:21:91:
		"King A (id:21) lived during at least one year of King B (id:91)."
		3:21b9:91:
		"King B (id:91) lived during the 10th year (9+1) of King A (id:21)."
3	Links two kings as being living during the same time. A class 3 synchronism with both kings defined to exact regnal years is a bidirectional alternative to a class 2 synchronism. Fourth field is not used.	**3:21b0:91e0:**
		"The final year of King B (id:91) is the same year as the first year of King A (id:21)."
4	An absolute year occurs during the reign of King A. Only used for validation testing and is not used to establish dates. Third field not used.	**4:31::1000**
		"1000 BCE occurs during the reign of King A (id:31)"
50	Asynchronism to an absolute year. Asynchronisms are not-conditionals or negated synchronisms. So instead of an event occurring in a specific year, with an asynchronism, an event does not occur in that year.	**50:31b5::1000**
		"The sixth year (5+1) of King A (id:31) was not in 1000 BCE"
		51:31::1000
51	Asynchronism where the absolute year does not occur during the reign of King A.	"1000 BCE does not occur during the reign of King A (id:31)"

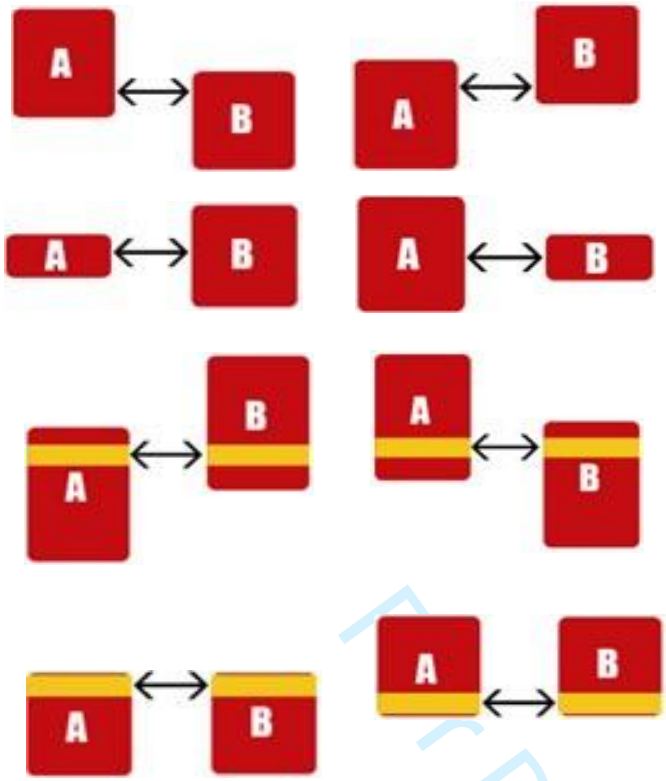
For Peer Review

52	Asynchronism where the specific year of King A is not equal to the specific year of King B.	**52:21b18:97b1:** "The 19th year of King A (id:21) is not equal to the 2nd year of King B (id:97)."
53	Asynchronism where the two kings did not live during the same time.	**53:21:91:** "King A (id:21) did not live during the time of King B (id:91)."
*	A link where one king precedes a second king. There is no event rule for this. The relationship is done manually in the GUI by connecting one yellow-node to another yellow-node with a link.	

For Peer Review

Image	Notations
	Year = A(y)
	Year ⊆ A
	A(y) →= B(y), A →⊤ B, A →⊥ B

For Peer Review



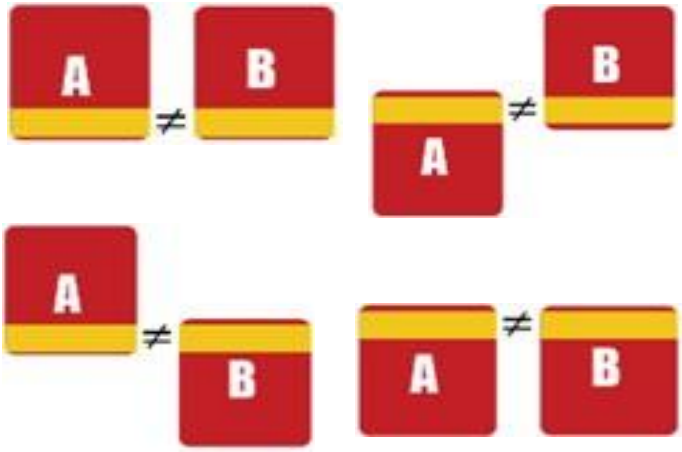
$A \leq B$,
$A \geq B$,
$A \leftharpoondown B$,
$A \rightharpoonup B$,
$A \subseteq B$,
$A \supseteq B$,
$A(y) = B(y)$,
$A \top B$,
$A \perp B$

$Year \subseteq A$

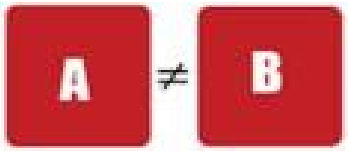
$Year \neq A(y)$

$Year \neq A$

For Peer Review



A(y) ≠ B(y)



A ≠ B

A(end)=B(start),
B(end)=A(start)

For Peer Review

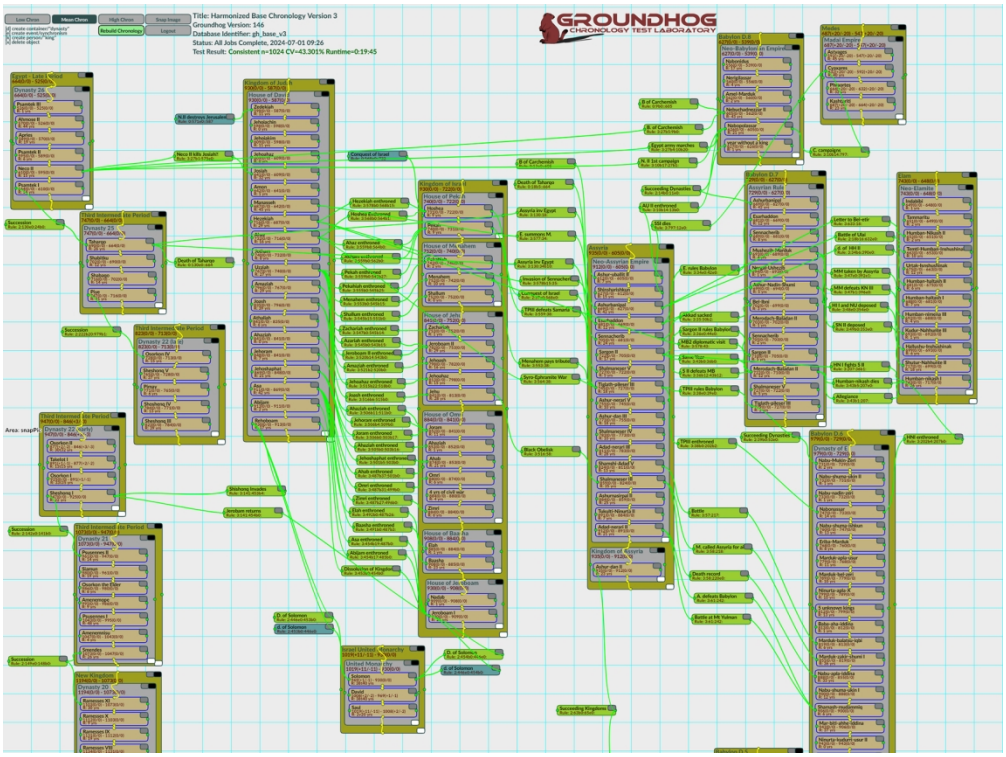


Figure 1: The Groundhog workspace with a populated chronology.

1741x1303mm (28 x 28 DPI)

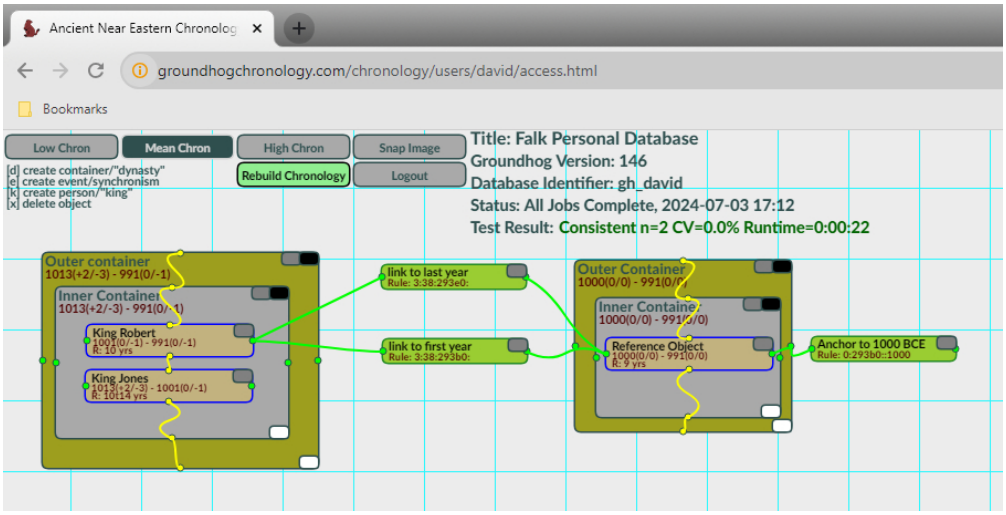


Figure 2: Screen shot of Groundhog with a short chronology.

820x415mm (28 x 28 DPI)