Toward a 'Grand Unified Theory' of Archaeological Chronology

Stephen Collins-Elliott

17 Septmeber 2024

Introduction

The logic of chronology-building is familiar to every archaeologist. Relative chronologies are established on the basis of stratigraphic superimposition, contextual similarity, stylistic similarity, and seriation, which are then associated with absolute calendrical dates. Such associations typically comprise termini post and ante quem (hereafter *t.p./a.q.*) derived from external documentary or historical events, whether about a context (e.g., destruction layer) or particular artifact (e.g., coinage), or on the basis of scientific methods like radiocarbon dating. The goal of formally "combining" sources of chronological information has been a clear objective. Software exists for calibrating and conditioning radiocarbon dates upon relative constraints, such as BCal (Buck, Christen, and James 1999), OxCal (Bronk Ramsey 2009), and the R pacakge `Bchron` (Haslett and Parnell 2008), but the need to provide a comprehensive means of formal dating and all archaeological or historical events is hampered by the use of informal judgments, especially for dating artifacts, such as "around the last third of such-and-such century." This paper addresses the need to simplify the model assumptions in play and provide more basic tools that work toward a "grand unified theory" of chronology, especially to bring the dating of artifact typology–in its occurrence of production, use, and deposition—into a more formal representation.

Namely, a three-step process is advocated, in which (1) all contexts are first seriated according to their find-type (grouping like with like) into an optimal order, and then (2) constrained to any known, fixed sequences, with absolute dates treated as probability densities. The final step (3) involves marginalizing probability densities of the depositional events of all contexts and finds, those of the absolute constraints, and the production dates of find-types. Probability densities of the use of finds can then be estimated as conditional upon their production and deposition. Hence, by first creating an optimal seriation (i.e., doing what archaeologists have always done in identifying similar assemblages and the co-occurrence of artifact types), single contexts and isolated stratigraphic sequences can be incorporated into formal models of chronology, allowing for full cross-site estimation.

Optimal and Fixed Sequences

Not all sequences are of the same informational value. Some sequences may be based on ideal or optimal theoretical assumptions, as with frequency or contextual seriation, and others they may be based on physical relationships, as with soil stratigraphy. It is clear that such theoretical sequences should yield to those based on physical relationships, but this raises the problem of how to coerce one or more theoretical sequences which may contain events whose physical relationships may conflict. For example, one can produce an ideal seriation of contexts that includes discrete, single deposits on the one hand (say a number of separate pits with no overlap) and on the other hand stratified deposits. In such an optimal seriation, it is possible for those stratified deposits to be seriated "out of order" with respect to their stratified sequence (i.e., the stratigraphy may have been perturbed at one more moments so that their finds assemblages are not well stratified). Accordingly, it is desirable to take the optimal order achieved via seriation and constrain it back to agree with the soil stratigraphy.

Checking Sequence Agreement

To check whether two or more sequences of events agree with one another in their order, the `seq_check()` function is used. Sequences should run in the same direction from left to right (i.e., the earliest element of all sequences should be the first element), and be contained in a `list` object. The `seq_check()` function returns a logical output if all sequences have the same elements in the same order.

Both sequences `x` and `y` contain the same events in the same order:

```
x <- c("A", "B", "C", "D", "E")
y <- c("B", "D", "F", "E")
a <- list(x, y)
seq_check(a)
#> [1] TRUE
```

But sequence `z` contains events `"F"` and `"C"` out of order with respect to `x` and `y`:

```
z <- c("B", "F", "C")
b <- list(x, y, z)
seq_check(b)
#> [1] FALSE
```

Merging Sequences

The `synth_rank()` function will use recursion in order to produce a single, "merged" or "synthesized" sequence from two or more sequences. This is accomplished by counting the total number of elements after running a recursive trace through all partial sequences (via the `quae_postea()` function, on which see below). If partial sequences are inconsistent in their rankings, a `NULL` value is returned.

```
x <- c("A", "B", "C", "D", "H", "E")
y <- c("B", "D", "F", "G", "E")
a <- list(x, y)
synth_rank(a)
#> [1] "A" "B" "C" "D" "F" "H" "G" "E"
```

Producing a single merged or synthesized sequence is a matter of procedural convenience for the `gibbs_ad()` function (below on Gibbs Sampling). To be sure, events missing from one sequence or another could occur at different points in the merged sequence. In the example above, `"H"` could occur at any point after `"D"` and before `"E"`, but the `synth_rank()` function has situated it in between `"F"` and `"G"`.

If sequences disagree, a `NULL` value is returned for the `synth_rank()` function.

Adjusting Sequences

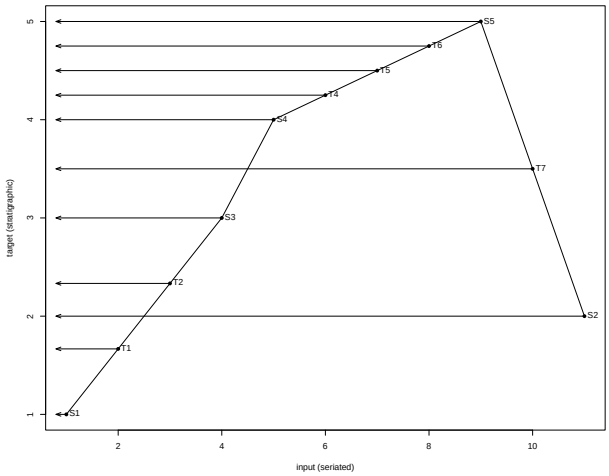
As mentioned above, one may have sequences which are derived via theoretical considerations (e.g., frequency or contextual seriation), and some which are known or fixed (e.g., soil stratigraphy or historical documentation). The `seq_adj()` function will take an "input" sequence and adjust its ordering to fit with another "target" sequence of smaller size.

For example, the input sequence might be an ordering obtained from a contextual seriation which is based on artifact types found in both tomb assemblages as well as stratified deposits, from one or more sites. And the target sequence might be a known stratigraphic sequence. One wants to maintain the ordering of the target sequence, adjusting the input into agreement:

```
# input
seriated <- c("S1", "T1", "T2", "S3", "S4", "T4", "T5", "T6", "S5", "T7", "S2")
# target
stratigraphic <- c("S1", "S2", "S3", "S4", "S5")
# input adjusted to agree with the target
```

```
seq_adj(seriated, stratigraphic)
#>  [1] "S1" "T1" "S2" "T2" "S3" "T7" "S4" "T4" "T5" "T6" "S5"
```

To achieve this the `seq_adj()` function performs a linear interpolation between jointly attested events, placing the input sequence along the x axis and the target sequence along the y axis, coercing the order of all elements to the y axis:



For multiple stratigraphic sequences, the `seq_adj()` function can be re-run, taking a new target sequence and using the previous result as the new input sequence. If both input and target sequences agree, the input sequence will be returned.

All Earlier Events, All Later Events

A core need of the `gibbs_ad()` function is to determine, for each event, *all* events which come later and earlier than that event. The functions `quae_postea()` and `quae_antea()` achieve this need for later events and earlier events respectively. Hence, there is no need for determining a single sequence or ordering as an input, which takes the form of a `list` object containing sequences. The output is a `list` indexed with each element, containing the vector of contexts which precede or follow that element.

For `quae_antea()`, a dummy element of `"alpha"` is included in all vectors, and for `quae_postea()`, a dummy element of `"omega"` is included. The elements of `"alpha"` and `"omega"` are necesary as they constitute the fixed lower and upper limits in which estimates are made in `gibbs_ad()`.

```
x <- c("A", "B", "C", "D", "H", "E")
y <- c("B", "D", "F", "G", "E")
a <- list(x, y)
quae_postea(a)
#> $A
#> [1] "omega" "B"     "C"     "D"     "H"     "E"     "F"     "G"
#>
#> $B
#> [1] "omega" "C"     "D"     "H"     "E"     "F"     "G"
#>
#> $C
#> [1] "omega" "D"     "H"     "E"     "F"     "G"
#>
#> $D
#> [1] "omega" "H"     "E"     "F"     "G"
```

```
#>
#> $H
#> [1] "omega" "E"
#>
#> $E
#> [1] "omega"
#>
#> $F
#> [1] "omega" "G"     "E"
#>
#> $G
#> [1] "omega" "E"
quae_antea(a)
#> $A
#> [1] "alpha"
#>
#> $B
#> [1] "alpha" "A"
#>
#> $C
#> [1] "alpha" "A"     "B"
#>
#> $D
#> [1] "alpha" "A"     "B"     "C"
#>
#> $H
#> [1] "alpha" "A"     "B"     "C"     "D"
#>
#> $E
#> [1] "alpha" "A"     "B"     "C"     "D"     "H"     "F"     "G"
#>
#> $F
#> [1] "alpha" "B"     "D"     "A"     "C"
#>
#> $G
#> [1] "alpha" "B"     "D"     "F"     "A"     "C"
```

Gibbs Sampling

The Gibbs sampler is a common Markov Chain Monte Carlo (MCMC) technique, widely used in estimating posterior probabilities in Bayesian inference (a mainstay of calibrating and refining radiocarbon dates; see references above) as well as in computing marginal densities. For more information, see Geman and Geman (1984), Buck, Cavanagh, and Litton (1996), and Lunn et al. (2013).

A central function of the `eratosthenes` package is the `gibbs_ad()` function, which takes in information about relative sequences and absolute dating constraints, and then samples marginal probability densities for events from the full, joint conditional density, using a Gibbs sampler. This vignette provides details on the use of this function.

The function operates on a continuous timeline. Any calendrical scale is possible, but here it is conventional for CE/AD dates to be positive and BCE/BCE dates to be negative.

Inputs

The core inputs for the `gibbs_ad()` function are the following:

- relative sequence(s) of contexts
- finds (optional), associated with a context and type/class (also optional)
- absolute constraints (*termini post/ante quem*)

Relative Sequences of Contexts Relative sequences of contexts must be in the form of a `list`, with each object in the list being a vector whose ordering of elements is in agreement with all other elements. See the vignette **Aligning Relative Sequences** for more information.

The following object `contexts` provides an example of a valid set of relative sequences:

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

seq_check(contexts) # check if the sequences are in agreement
#> [1] TRUE
```

Finds Data on finds (i.e., any elements which pertain to a given context) are optional. Each find must be in the form of a `list` with the following structure:

- `id`: An id number or string of the find, such as an inventory number or bibliographic reference.
- `assoc`: The context to which the find belongs, which must be contained in the relative sequences of contexts.
- `type`: An optional vector or element denoting any types, subtypes, classes, etc., to which the find pertains. If not present, a `NULL` value must be given.

Each find must in turn be stored in a single `list` object:

```
f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)
artifacts <- list(f1, f2, f3, f4, f5, f6)
```

Missing information on types should be supplied with a `NULL` value.

Finds should have no absolute dating constrains on them. If they do, they should be specified as an absolute constraint.

Absolute Constraints Absolute constraints are predicated on whether they provide a *terminus post quem* (*t.p.q.*) for a context or a *terminus ante quem* (*t.a.q.*) for a context. The information on these absolute dates is regarded as external or extrinsic information. For example, a radiocarbon date provides for information on when the sample died, not when its context was formed; a coin type may be known to have had a range of production dates, but the production date of *that particular coin* may be affected by the stratigraphic context in which it is found. Such constraints may take a variety of forms.

The formatting for a *t.p.q.* or a *t.a.q* is the same, as a `list` in which each constraint contains:

- `id`: An id number or string of the find, such as an inventory number or bibliographic reference.
- `assoc`: The context to which the find belongs, which must be contained in the relative sequences of contexts.
- `type`: An optional vector or element denoting any types, subtypes, classes, etc., to which the find pertains. If not present, a `NULL` value must be given.

- `samples`: A `numeric` vector or element containing potential dates of the *t.p.q.* or *t.a.q.*, i.e., a sample of the probability density function which expresses when that constraint occurred. Common densities would include:
  - A single `numeric` if the constraint is known precisely and certainly.
  - Samples of n size from a continuous uniform distribution, `runif(n, a, b)`, if known between two bounds a and b, without any more or less certainty about any one date.
  - Samples of n size from a bespoke probability density, such as a calibrated radiocarbon date.

Constraints must be contained in two separate `list` objects, one for *t.p.q.* and the other for *t.a.q.*:

```
# external
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = runif(100,37,41))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)
```

Additional Arguments

Additional arguments are necessary for the `gibbs_ad()` function:

- `samples`: the number of Gibbs samples to take (i.e., the number of estimates of any one event). By default set at `10^5`.
- `alpha`: the constraint on the earliest possible date to sample. By default set at `-5000`.
- `omega`: the constraint on the latest possible date to sample. By default set at `1950`.
- `trim`: takes a logical value as input, `trim` specifies whether to remove contexts from the result which lie earlier than the earliest given *t.p.q.* or later than the latest *t.a.q.*, i.e., contexts whose estimation depends on `alpha` and `omega`. By default set at `TRUE`.
- `rule`: the rule for how to estimate production dates for artifact types, which is described in the following section. By default set at `"naive"`.

Rules for Estimating Production Dates Since archaeologists are typically interested in dates of production and use as much as deposition, the `gibbs_ad()` function will return the marginal densities for both production and deposition (from which the estimation of a use date can then be derived).

Estimating the date of the production of a find or find-type however necessitates some assumption, since in principle the absence of evidence is not viewed as evidence of absence. Without stipulating a rule, the earliest production date of any artifact could reach back endlessly into time, since an artifact does not need to have been produced after the initial occupation of a site where it has been found.

Here, two basic rules have been included for determining production dates of finds:

- `"naive"`: The earliest potential threshold of a find-type occurs sometime before the first deposition of that type, and after the deposition of the next earliest context. A production date is then chosen uniformly at random between that threshold and the depositional date of that artifact.
- `"earliest"`: The earliest potential date of a find-type occurs sometime before the first deposition of that type, and after the deposition of the next earliest context. A production date is then chosen uniformly at random between those two dates.

The `"earliest"` option will constrain the date of production to the earliest possible instances, while the `"naive"` option (the default) will select any date between an earliest threshold and the depositional date of the particular find.

If no finds are included in the `gibbs_ad()` arguments, then only depositional dates for contexts, not production dates, are estimated.

Functionality

The `gibbs_ad()` function at its core uses a Gibbs sampler, drawing from the full joint conditional density in order to sample marginal densities for dates of deposition (of contexts and finds) and production (of finds).

First, samples are drawn from any *t.p.q* and *t.a.q.*. Then, for convenience, the Gibbs sampler proceeds in order of a sequence of contexts based on the merged ranking of all contexts (via `synth_rank()`). The sampler will identify all contexts and constraints prior and subsequent to any one context, and then will identify the largest prior date and smallest subsequent date, in between which it will uniformly sample a date. One can adjust the number of samples drawn with the `samples` argument of the function:

```
dates <- gibbs_ad(contexts, finds = artifacts, samples = 10^4, tpq = tpq_info, taq = taq_info)
```

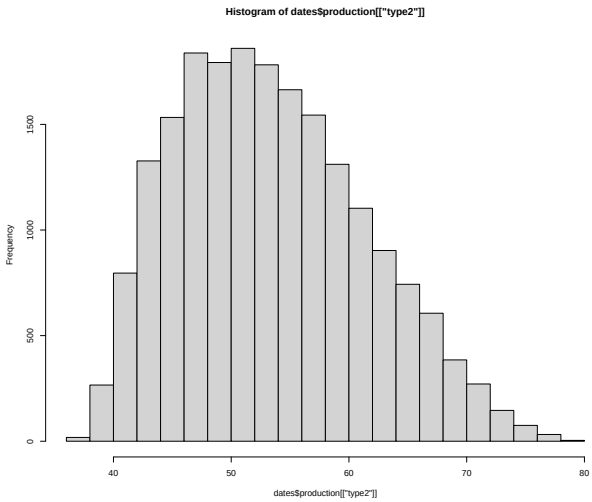
Output

The output of the `gibbs_ad()` function will be a `list` of class `marginals` containing the marginal densities of the depositional dates of contexts and finds, if included; production dates are given for finds types, again, if included. Marginal densities are also given for each *t.p.q.* and each *t.a.q.*, which expresses the probability of their dating *given* the conditions of the relative sequences of contexts (not independent of them).

- `$deposition` contains the depositional dates of contexts included in the sequences input
- `$externals` contains the dates of the absolute constraints taking the full joint conditional density into account
- `$production` contains the dates of production of artifact types

```
str(dates)
#> List of 3
#>  $ deposition:List of 9
#>   ..$ B: num [1:10000] 47.5 -185.3 -147.6 -293.1 -270 ...
#>   ..$ C: num [1:10000] 61 -42.9 -106.6 -220.6 -71.8 ...
#>   ..$ D: num [1:10000] 69.573 -24.496 -62.15 -3.912 0.343 ...
#>   ..$ E: num [1:10000] 78.44 3.61 18.81 1.47 26.41 ...
#>   ..$ F: num [1:10000] 78.8 78.2 69.4 38.9 29.2 ...
#>   ..$ G: num [1:10000] 78.8 78.5 70.6 72.5 71.1 ...
#>   ..$ H: num [1:10000] 78.9 78.7 73.2 73.4 76.1 ...
#>   ..$ I: num [1:10000] 79 78.7 78.7 77.6 77.1 ...
#>   ..$ J: num [1:10000] 79 78.9 78.8 78.4 78.2 ...
#>  $ externals :List of 3
#>   ..$ coin1: num [1:10000] -320 -316 -307 -309 -305 ...
#>   ..$ coin2: num [1:10000] 37 37.9 37.1 40 39.2 ...
#>   ..$ destr: num [1:10000] 79 79 79 79 79 79 79 79 79 79 ...
#>  $ production:List of 4
#>   ..$ type1: num [1:30000] 67.9 65 72.1 -39.5 -14.6 ...
#>   ..$ form1: num [1:30000] 69 68.6 77.8 -25 -42.8 ...
#>   ..$ form2: num [1:10000] 74.083 -4.46 -50.042 -0.123 2.485 ...
#>   ..$ type2: num [1:20000] 78.9 78.9 78.6 78.6 71.3 ...
#>  - attr(*, "class")= chr [1:2] "marginals" "list"
```

For example, a histogram of the production date of a `type2` artifact using a naive date of production will be:



In order to estimate a date of use for any one artifact, one can run the `gibbs_ad()` function again, taking the production date of the artifact type from the `marginals` object as a *t.p.q.* and its depositional context as a *t.a.q.*:

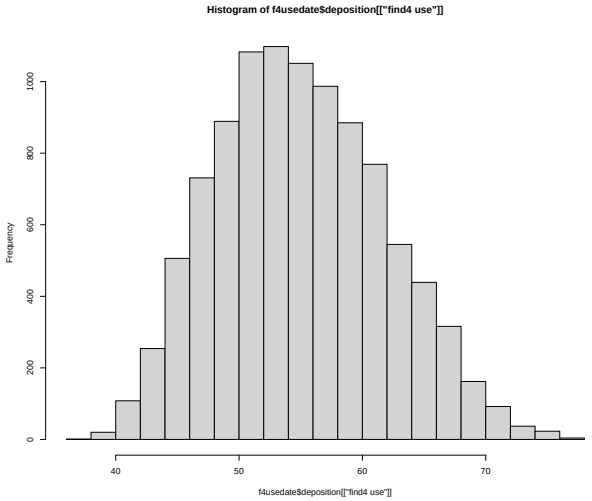
```
f4use <- list("find4 use")

find4prod <- list(id = "type2prod", assoc = "find4 use", samples = dates$production[["type2"]])
find4dep <- list(id = "find4dep", assoc = "find4 use", samples = dates$deposition[[f4$assoc]])

tpq_find4prod = list(find4prod)
taq_find4dep = list(find4dep)

f4usedate <- gibbs_ad(f4use, samples = 10^4, tpq = tpq_find4prod, taq = taq_find4dep)

hist(f4usedate$deposition[["find4 use"]])
```



References

Bronk Ramsey, C. 2009. "Bayesian Analysis of Radiocarbon Dates." *Radiocarbon* 51: 337–60.

Buck, C. E., W. G. Cavanagh, and C. D. Litton. 1996. *Bayesian Approach to Interpreting Archaeological Data*. Chichester: John Wiley & Sons.

Buck, C. E., J. A. Christen, and G. N. James. 1999. "BCal: An On-Line Bayesian Radiocarbon Calibration Tool." *Internet Archaeology* 7. https://intarch.ac.uk/journal/issue7/buck/.

Geman, S., and D. Geman. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6: 721–41.

Haslett, J., and A. C. Parnell. 2008. "A Simple Monotone Process with Application to Radiocarbon-Dated Depth Chronologies." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 57 (4): 399–418.

Lunn, D., C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter. 2013. *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Boca Raton, FL: CRC Press.