

# Package: eratosthenes (via r-universe)

November 20, 2024

**Title** Archaeological Synchronism

**Version** 0.0.2

**Description** Estimates unknown historical or archaeological dates subject to relationships with other dates and absolute constraints, derived as marginal densities from the full joint conditional distribution. Includes rule-based estimation of the production dates of artifact types. Collins-Elliott (2024) <<https://volweb.utk.edu/~scolli46/sceGUTChronology.pdf>>.

**License** GPL (>= 3)

**Imports** Rcpp, Rdpack

**RdMacros** Rdpack

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**LinkingTo** Rcpp

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Repository** <https://scollinselliott.r-universe.dev>

**RemoteUrl** <https://github.com/scollinselliott/eratosthenes>

**RemoteRef** HEAD

**RemoteSha** adef02b71500cc531e2715d02bff8fb2d7ddbcc5

## Contents

gibbs_ad . . . . .	2
quae_antea . . . . .	4
quae_postea . . . . .	5
seq_adj . . . . .	5
seq_check . . . . .	6
synth_rank . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

## Description

A Gibbs sampler for archaeological dating, to fit relative sequences to absolute, calendrical dates. Elements can be associated with *termini post quem* (*t.p.q.*) and *termini ante quem* (*t.a.q.*), which are treated as a given probability density function  $f(t)$ . This function may take any form, a single date (i.e., with a probability of 1), a continuous uniform distribution (any time between two dates), or a bespoke density (as with calibrated radiocarbon dates). Inputs of this function take samples drawn from their respective density functions.

## Usage

```
gibbs_ad(
  sequences,
  finds = NULL,
  samples = 10^5,
  tpq = NULL,
  taq = NULL,
  alpha = -5000,
  omega = 1950,
  trim = TRUE,
  rule = "naive"
)

## S3 method for class 'list'
gibbs_ad(
  sequences,
  finds = NULL,
  samples = 10^5,
  tpq = NULL,
  taq = NULL,
  alpha = -5000,
  omega = 1950,
  trim = TRUE,
  rule = "naive"
)
```

## Arguments

sequences	A list of relative sequences of elements (e.g., contexts).
finds	Optional. A list of finds related to (contained in) the elements of sequences. If one includes this ob
samples	Number of samples. Default is $10^5$ .
tpq	A list containing <i>termini post quem</i> . Each object in the list consists of:

- id A character ID of the *t.p.q.*, such as a reference or number.
- assoc The element in code to which the *t.p.q.* is associated.
- samples A vector of samples drawn from the appertaining probability density function of that *t.p.q.*

taq A list containing *termini ante quem*. Each object in the list consists of:

- id A character ID of the *t.a.q.*, such as a reference or number.
- assoc The element in code to which the *t.p.q.* is associated.
- samples A vector of samples drawn from the appertaining probability density function of that *t.p.q.*

alpha An initial *t.p.q.* to limit any elements which may occur before the first provided *t.p.q.* Default is -5000.

omega A final *t.a.q.* to limit any elements which may occur after the after the last provided *t.a.q.* Default is 1950.

trim A logical value to determine whether elements that occur before the first *t.p.q.* and after the last *t.a.q.* should be omitted from the results (i.e., to "trim" elements at the ends of the sequence, whose marginal densities depend on the selection of alpha and omega). Default is TRUE.

rule The rule for computing an estimated date of production of a find-type, either "earliest", selecting a production date between the earliest deposition of that type and the next most earliest context, or "naive" (the default), which will select a production date any time between the distribution of that "earliest" date and the depositional date of that artifact.

## Value

A list object of class `marginals` which contains the following:

- deposition A list of samples from the marginal density of each context's depositional date.
- externals A list of samples of the marginal density of each constrant (*t.p.q.* and *t.a.q.*), as conditioned upon the occurrence of other depositional
- production If a finds object has been input, samples of the marginal density of the production date of finds types will be included in the output.

## Examples

```
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)

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)

# external constraints
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)

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

---

quae\_antea

*Quae Antea*

---

## Description

For a list of multiple partial sequences (of vector objects), generate another list which, for each element, gives the elements that occur before it ("*quae antea*"), i.e., analogous to a recursive trace through all partial sequences from right to left. An element "alpha" is added to all sets to avoid empty vectors. See also [quae\\_postea](#).

## Usage

```
quae_antea(obj)
```

```
quae_antea(obj)
```

## Arguments

obj                    A list of vector objects which represent ordered sequences.

## Value

A list of vector objects, which contain the elements that occur before any one given element in the input sequences.

## Examples

```
x <- c("A", "B", "C")
y <- c("B", "D", "E", "C", "F")
z <- c("C", "G")
a <- list(x, y, z)

quae_antea(a)
```

---

`quae_postea`*Quae Postea*

---

### Description

For a list of multiple partial sequences (of vector objects), generate another list which, for each element, gives the elements that occur after it ("*quae postea*"), i.e., analogous to a recursive trace through all partial sequences from left to right. A final element "omega" is added to all sets to avoid empty vectors. See also [quae\\_antea](#).

### Usage

```
quae_postea(obj)

## S3 method for class 'list'
quae_postea(obj)
```

### Arguments

`obj`                    A list of vector objects which represent ordered sequences.

### Value

A list of vector objects, which contain the elements that occur after any one given element in the input sequences.

### Examples

```
x <- c("A", "B", "C")
y <- c("B", "D", "E", "C", "F")
z <- c("C", "G")
a <- list(x, y, z)

quae_postea(a)
```

---

`seq_adj`*Adjust Sequence to Target*

---

**Description**

Given an "input" sequence of elements and another "target" sequence that contains fewer elements in a different order, shift the order of the input sequence to match that of the target, keeping all other elements as proximate to one another as possible. This adjusted ranking is accomplished using piecewise linear interpolation between joint elements ranks. That is, joint rankings are plotted, with input rankings along the  $x$  axis and target rankings on the  $y$  axis. Remaining rankings in the input sequence are assigned a ranking of  $y$  based on the piecewise linear function between joint rankings. If the rank order of elements in the target are identical to those in the input, the result is identical to the input. A minimum number of three joint elements in both the input and target are required.

**Usage**

```
seq_adj(input, target)

## S3 method for class 'character'
seq_adj(input, target)
```

**Arguments**

input	A vector of elements in a sequence.
target	A vector of elements in a sequence, containing at least three of the same elements as input.

**Value**

A vector of the adjusted sequence.

**Examples**

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J") # the input sequence
y <- c("D", "A", "J") # the target sequence

seq_adj(x, y)
```

---

seq\_check

*Sequence Check*


---

**Description**

For a list of partial sequences (of vector objects), check to see that joint elements of each occur the same order. That is, for two sequences with elements  $A, B, C, D, E$  and  $B, D, F, E$ , all joint elements must occur in the same order to pass the check. Two sequences  $A, B, C, D, E$  and  $A, F, D, C, E$  would not pass this check as the elements  $C$  and  $D$  occur in different orders in either sequence.

**Usage**

```
seq_check(obj)

## S3 method for class 'list'
seq_check(obj)
```

**Arguments**

obj                    A list of vector objects which represent a sequence.

**Value**

TRUE or FALSE

**Examples**

```
x <- c("A", "B", "C", "D", "E")
y <- c("B", "D", "F", "E")
a <- list(x, y)

seq_check(a)

z <- c("B", "F", "C")
b <- list(x, y, z)

seq_check(b)
```

---

synth\_rank

*Synthetic Ranking*

---

**Description**

Using a list two or more partial sequences, all of which observe the same order of elements, create a single "synthetic" ranking. This is accomplished by counting the total number of elements after running a recursive trace through all partial sequences (via [quae\\_postea](#)). If partial sequences are inconsistent in their rankings, a NULL value is returned.

**Usage**

```
synth_rank(obj, ties = "average")

## S3 method for class 'list'
synth_rank(obj, ties = "average")
```

**Arguments**

obj	A list of vector objects which represent a sequence.
ties	The way in which ties are handled per the <a href="#">rank</a> function. The default is "ties = average".

**Value**

A single vector containing the synthesized ranking.

**Examples**

```
x <- c("A", "B", "C", "D", "E")
y <- c("B", "D", "F", "E")
a <- list(x, y)
```

```
synth_rank(a)
```



# Index

`gibbs_ad`, 2

`quae_antea`, 4, 5

`quae_postea`, 4, 5, 7

`rank`, 8

`seq_adj`, 5

`seq_check`, 6

`synth_rank`, 7