Spatial Analysis and Data Exploration in History and Archaeology, Spring 2021
LDA-H313

WEEK 6 EXERCISE: Aoristic Analysis

This week we will look at aoristic analysis, a method for creating aggregated chronological views from fuzzily dated data. While originally developed for criminological research, it is particularly well suited for examining archaeological (and other historical) data. The exercise will take you through basic aoristic analysis, and further refinements with Monte Carlo probability simulation and beta-distribution.

First, please place the **week6** folder, which contains the data we will examine, in the **sade2021** folder and set your workspace. Make sure that the path is exactly as indicated, for example without a space in the folder name:

Windows:

```
setwd("C:\\sade2021\\week6")
```

Mac:

```
setwd("~/Documents/sade2021/week6")
```

1. Simple Aoristic Analysis

1.1 Setting up archSeries

The R code that we will be using was developed by David Orton[1] and is not available as a formal R package. Instead, we need to download his *archSeries* package from his github site. Like with installing a package, you will need to do the initial download only, after that you can simply load it like a regular library.

First though, you need to install and load the devtools package, which will enable the initial download and installing of archSeries. Install *devtools* with **install.packages()**.

```
library(devtools)
```

If you are on a Windows machine, run the following command:

```
devtools::install_github("davidcorton/archSeries")
```

[1] See Orton et al 2017, Further reading folder for week 6 on Moodle.

If you are on a Mac, run the following command:

```
install_github("davidcorton/archSeries")
```

You will need to do the above only once. In the future, you can just load archSeries like a regular package. Do so now:

```
library(archSeries)
```

1.2 Data

As an example, we will be working with metal-detected data taken from the Portable Antiquities Scheme in England and Wales (PAS) database[2] and recovered in the vicinity of the village of Kingston Deverill in the county of Wiltshire, England. As the name "kingston" (king's village or estate) might suggest, Kingston was a settlement of some local importance during the Early Middle Ages. It was located at the interjection of two former Roman roads and was probably a nexus of regional travel. The great Domesday Survey of 1086, evaluating estates owned by the secular and ecclesiastical elite following the division of spoils from the Norman Conquest that brought in a new king and a new aristocratic twenty years earlier, has it as one of larger manors in the region. It may well have been an early market centre.

The data consists of a CSV file containing abbreviated records of almost 2000 objects that have been recovered within approximately one mile of the present-day village centre.

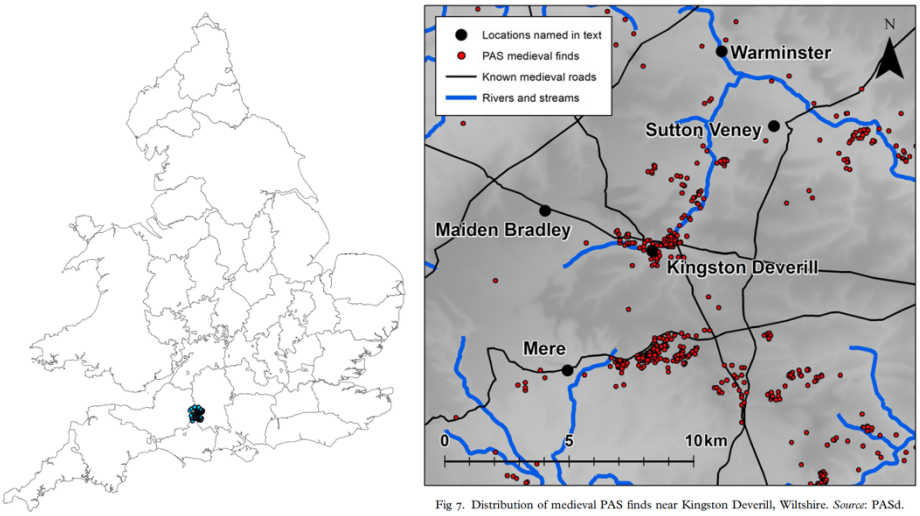

Fig 7. Distribution of medieval PAS finds near Kingston Deverill, Wiltshire. *Source*: PASd.

Metal-detected finds near Kingston Deverill. Oksanen & Lewis (2020). 'Medieval commercial sites: as seen through Prtable Antiquities Scheme Data', Antiquity 100*, 1-32*

[2] finds.org.uk

The data table consists of a couple of columns: one that gives the general object type of a find, a second one that gives the probably broad historical period of the find (Roman, medieval, etc), third and a fourth that give an estimate of the earliest and latest dates for the manufacture of the object, and finally the objects ID on the PAS online database.

Load the data, and examine it using various functions to get an idea of what is inside it. How many records does it have - use **nrow()**. Make sure that the data table loads alright by e.g. using the **`head()`** command (see picture below).

```
mydata <- read.csv(file="mydata.csv", header=TRUE, sep=",")
```

```
> head(mydata)
        objecttype broadperio fromdate todate  old_findID
1      END SCRAPER  NEOLITHIC    -3500   -800 WILT-7FC4B2
2      END SCRAPER  NEOLITHIC    -2500  -2100 WILT-7FA0D7
3      END SCRAPER  NEOLITHIC    -2500  -2100 WILT-7F7A04
4              AWL BRONZE AGE    -2150   -800 WILT-77CB78
5           ARMLET    UNKNOWN    -1200    410 WILT-BB14B5
6 SOCKETED AXEHEAD BRONZE AGE    -1150   -800 WILT-CB58F4
```

2.3 Editing and cleaning up data on the fly

Like a lot of large datasets that have been built over many years by the work of many hands, PAS data is needs cleaning before analysis. In a research project you would go through the original dataset with more care, perhaps in Excel, correcting obvious typos so as to retain as many records as possible, but in this section we will look at some commands to clean up the bulk of it quick in an R workspace.

First, though, archSeries assumes that the “no earlier than” and “no later than” columns are called Start and End, so lets rename them.

```
names(mydata)[names(mydata)=="fromdate"] <- "Start"
names(mydata)[names(mydata)=="todate"] <- "End"
```

The analysis works best if all rows have actual dates in the Start and End columns. If a cell is empty (it has a NA value), we can change that value to 0.

```
mydata[is.na(mydata)] <- 0
```

Then remove all rows, in which either Start or End date is 0.

```
mydata<-subset(mydata, Start!="0")
mydata<-subset(mydata, End!="0")
```

What if there has been a typo, and an End data that is earlier than Start date has been entered? Lets create a new column called “subtract” that gives a value that is equal to End date minus Start date, reorder the whole data frame in numerical order by the values of that column, and inspect the top of the data frame.

```
mydata$subtract <- (mydata$End - mydata$Start)
mydata <- mydata[order(mydata$subtract),]
head(mydata)
```

There does seem to be a handful of typos. You could go back to the original CSV file and correct these by hand, but for now lets just delete all rows where this is the case.

```
mydata <- mydata[!mydata$subtract < 0, ]
```

Finally, the script for aoristic analysis seems to like it that the data frame is as simple as possible. Large data frames with extraneous information can slow down the calculation, so lets create a vector containing only those columns that we need the most -- object types and the dates -- and overwrite the original data with it.

```
mydata <- mydata[,c("objecttype", "Start","End")]
```

At your leisure, study the above commands for how they work. You can adopt the principles underlying these simple editing commands to manipulating data frames in general. You can look for guidance on R “language” here, look for instance under the “R Operators” and “R If… Else” sections: https://www.w3schools.com/r

2.4 Aoristic weighing

Now that we have our data we can start doing analysis with it. The basic aoristic weighing is straightforward. In the below function you need to set the relevant parameters with the start/end.date and bin.width arguments. Lets look at the medieval data, setting the timeframe from 1000 to 1600 AD and a temporal bin of 20 years.

```
aorist<-aorist(mydata, start.date=1000, end.date=1600,
bin.width=20)
```

We can plot this as a bar chart.

```
aorist.plot(aorist, opacity=80, ylab="Aoristic Sum")
```

If we assume that the amount of material culture deposit stands as a loose proxy for population or economic activity, what does this suggest about the waxing and waning of the settlement's fortunes?

2.5 Different object types

This is a very rough overview, however, and since different object types encapsulate different biases or processes, we should look a bit deeper into the data. Using the **summary()** function, you can check that coins are by far the most common object type. We could separate them into another object and look at coins and other artefacts separately.

```
coin <- mydata[mydata$objecttype == "COIN", ]
notcoin <- mydata[mydata$objecttype != "COIN", ]
```

Now rerun the aoristic weighing and plot the two graphs for comparison. What do you think? How does this change the picture?

3. Monte Carlo Simulation

While very useful, simple aoristic weighing can produce a misleading impression of the spread of the data. As it produces an overview of the average distribution of temporal observations, the specificity of the bar chart format does not reflect uncertainty in the data. It is much better to represent this with a probability envelope.

We will run a Monte Carlo simulation on the data, using the original dataset that combines all object finds. First, the code requires each row to have a unique ID number, so create a new vector of sequential numbers equal to the number of rows and attach it as a new column called *ID*. Can you see how it is done?

```
mydata$ID <- c(1:nrow(mydata))
```

3.1 Simulate distributions

Then, simulate the distributions. Make note of the arguments: you must define the bin width you want to use (here 20 years), the number of repetitions and the bookend dates for the simulation (here AD 1000 and AD 1600). As with most functions there are many other arguments that automatically default to certain value we can accept, and we therefore do not need to explicitly type them out (check **?date.simulate**).

```
mydata_sim<-date.simulate(mydata, start.date=1000,
end.date=1600, bin.width=20, reps=1000)
```

3.2 Plotting the results

You have now created a MC simulation with 1000 repetitions. You can plot these as lines that combine together (excluding the top and bottom 2.5% as outliers) as probability envelope with a 95% confidence level. This is similar in principle to the exercise you conducted in week 3. Give it a few moments to draw.

```
lines.chron(mydata_sim)
```

The varying width of the aggregate envelope consequently demonstrates the relative degree of temporal uncertainty per chronological period. It is naturally possible that within the same dataset for some period we can be more confident about our level of dating.

A more visually appealing rendering of the same data is to plot the graph not as a bundle of lines, but as a block of solid colour between the upper and lower boundaries. The darker line through the middle gives the median value at each bin. If you want to change the colour, or do not want to have the line depicted, you can edit the default values in the function argument - take a look with **`?poly.chron`**.

```
poly.chron(mydata_sim)
```

Finally, you can also express the distributions as boxplots per temporal bin. This can make it, for example, easier to examine the probability envelope at specific places.

```
box.chron(mydata_sim)
```

3.3 Coins and artefacts

Now, carry out the same exercise, but do it the coin and non-numismatic artefact datasets you created earlier. In addition to peaking at different times, can we say that one of the two sets is dated with greater confidence? What might this indicate about the probable real pattern of depositional activity around Kingston Deverill? Examine the data again (you can also use Excel) to see what biases might lurk in it. Hint: the year AD 1400 seems to be a popular boundary date between the notional periods "Central Middle Ages" and "Late Middle Ages". For objects that are difficult to date more closely than, say, a few hundred years, how do such conventions skew distributions?

4. Beta Distribution

A further refinement to the technique considers beta distribution. This is a method for working probability distributions in data, which yields itself also to aoristic analysis and was introduced to archaeological work in Hilary Cool and Mike Baxter's 2016 case study of Roman brooches.[3]

4.1 Modelling probability distribution

First, lets look at different beta distribution shapes. We'll create data (a sequence of numbers) and then apply a probability distribution function to it with different parameters of α and β.

.

```
data = seq(0,1, length=100)
plot(data, dbeta(data, 1, 1), ylab="density", type ="l",
col="black")
```

Here α = 1 and β = 1. This describes an even probability distribution. If you think of the intervals in the x-axis as the temporal bins of an aoristic weighing, this would mean that each bin is given equal weight.

We could prioritise the middle at the expense of the edge "bins" with α = 2 and β = 2:

```
plot(data, dbeta(data, 2, 2), ylab="density", type ="l",
col="red")
```

[3] Cool and Baxter (2016). 'Brooches and Britannia', *Britannia* 47 in the Further reading folder.

Try out a few different parameters, and think about what kind of data and analysis they might pertain to.

4.2 PAS data

Now we apply this to our object data. We'll simply run the MC simulation again, now adding arguments containing the desired parameters. Do $\alpha = 2$, $\beta = 2$.

```
mydata_sim_2<-date.simulate(mydata, start.date=1000,
end.date=1600, bin.width=20, reps=1000, a=2, b=2)
```

You can replot the graphs using the functions from section 3.2; some depict the difference more clearly than others.

This could be argued to model *recorder* behaviour. Is it reasonable to assume that a person entering uncertainly dated records into a table is more likely to be certain that the object falls into the central portion of the timeline? This is certainly not a clear-cut issue, but it is worth thinking about. You can also try other parameter values.

You are now finished with this week's exercise. You can save your workspace and quit R.