Tutorial 2 - Circular descriptives

Arjan Huizing

16 April 2018

Circular descriptives in R

In the previous tutorial we discussed what circular data is and how it is measured. In this tutorial we will start analyzing circular data. For the analyses of circular data we will be using the open source programming language R. R is available for Windows, MacOS and popular Linux distributions, and can be downloaded from https://cran.r-project.org/.

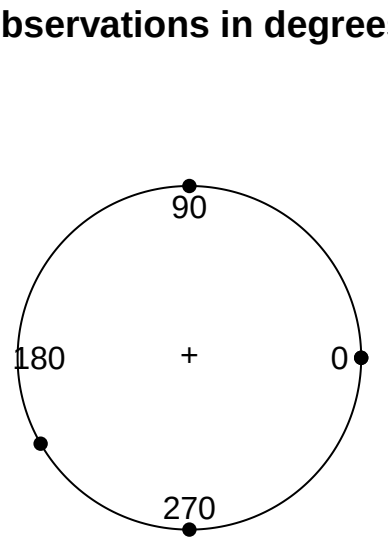
To get started, install and load the package `circular` in R. This package contains functions for analyzing as well as plotting circular data.

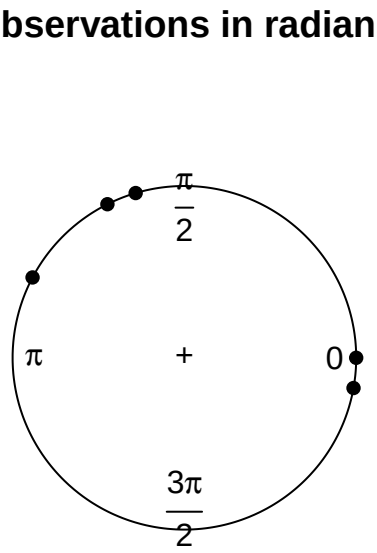
```
# install.packages("circular")
library(circular)
```

For circular data there are two units of measurement that are commonly used, which are degrees and radians. A circle has a total of 360 degrees, or equivalently, a total of 2π (≈ 6.28) radians. Values cannot exceed this total, and if they do most functions will automatically scale them back to the circular range (e.g. 361 degrees equals 1 degree). To see this in action, we specify two small datasets, one in degrees and the other in radians. For this, we use the function `as.circular` and specify the measurement type with the argument units.

```
data.deg <- as.circular(c(0, 90, 210, 270, 360), units = "degrees")
data.rad <- as.circular(c(0, 90, 210, 270, 360), units = "radians")
```

In our dataset with values in radians, all values except for 0 exceed 6.28. Thus when calculating the mean direction, all these values are rescaled. Instead of specifying the values of 0, 90, 210, 270 and 360, we specified the values 0, 2.0354057, 2.6548849, 6.1062171 and 1.8584375 radians. Remainders can be calculated in R using `%%`.





This shows that we must be very careful: the data entered in `as.circular` must match the units that are given. In fact, the way we entered our data in radians suggests that we first wanted to convert a set of data in degrees to radians and then work with them as radians. To do that, we could have done;

```
data.rad <- as.circular(c(0, 90, 210, 270, 360) * pi / 180, units = "radians")
```

or even just

```
data.rad <- conversion.circular(data.deg, units = "radians")
```

For the remainder of this tutorial, we will keep working with the dataset which interprets the values 0, 90, 210, 270 and 360 as degrees.

The circular mean

A mean value in circular data is referred to as the *mean direction*. For example, the mean direction in a dataset with observations on a compass pointing north (0∘) and east (90∘) would be northeast (45∘). Although intuitively this makes sense, it often is not as straight forward to calculate as the mean in linear data. Unlike in linear data, it is not possible to take the sum of all values and divide them by n. Imagine the same scenario as before but with the compass indicating 359∘. Where a linear mean would drastically shift, the mean direction will barely change.

To obtain the mean direction then, requires we use the sum of cosins ($C = \sum \cos\theta_i$), sum of sines ($S = \sum \sin\theta_i$) and $R = \sqrt{C^2 + S^2}$, which is the square root of the sum of squared C and S values. The mean direction is then given by

$$\overline{\theta} = \begin{cases} tan^{-1}(S/C) & S > 0, C > 0 \\ tan^{-1}(S/C) + \pi & C < 0 \\ tan^{-1}(S/C) + 2\pi & S < 0, C > 0 \end{cases}, \text{(Fisher, 1995)} \quad (1)$$

The mean direction can easily be calculated in R. The package `circular` contains the function `mean.circular`, which integrates with the base R function for calculating the mean.

```
#Obtaining the linear mean of the values we have specified.
mean(c(0, 90, 210, 270, 360))
```

```
## [1] 186
```

```
mean(data.deg)
```

```
## Circular Data:
## Type = angles
## Units = degrees
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
## [1] -23.79398
```

As we can see, the mean and the mean direction of values in degrees are very different. Where the regular mean simply takes the sum of all values and divides them by the amount of values, the mean direction takes account of the fact that data is on a circular continuum. In the circular dataset, the datapoints 90 and 270 are both the same distance from 0. Finally, the values of 0 and 360 are identical. The resultant mean direction is -23.8 degrees, which is equal to 336.2 degrees.

One peculiarity with the mean direction is that it can be undefined. For example, what is the mean direction of north and south: east or west? More on this in the third tutorial on distributional assumptions.

Measures of concentration and spread

The mean direction is a measure of location, just like the mean in linear data. Similarly, we can also obtain descriptives that are a measure of spread or concentration. These indicate how much variation there is in observations, similar to the variance in linear data.

The mean resultant length (ρ) is one such measure, and is obtained by $\overline{R} = R/n$, where $R = \sqrt{C^2 + S^2}$. This is a measure of *concentration* that can take on values between zero and one. One indicates that all values of the dataset are concentrated at a single location. A value close to zero indicates a higher spread. Interestingly enough, zero itself does not necessarily mean complete dispersion but can also be caused by having two sets of observations on opposite ends on the circle. More on this in the third tutorial on distributional assumptions.

The mean resultant length can be calculated using `rho.circular`.

```
rho.circular(data.deg)
```

```
## [1] 0.2478627
```

A mean resultant length of $\rho \approx .248$ indicates a somewhat low concentration.

Other than the mean direction and mean resultant length, we can also obtain a circular variance and a circular standard error.

Since the mean resultant length is a measure of concentration and the variance is traditionally a measure of spread, the variance in circular data is simply the inverse of the mean resultant length. The function

`var.circular` will give you this value, but it is also easily obtained by $1 - \rho$. The standard deviation is *not* obtained using the square root of the variance, but rather using $(-2 \log (1 - var))^{1/2}$.

```
var(data.deg)
```

```
## [1] 0.7521373
```

```
sd(data.deg)
```

```
## [1] 1.670258
```