Intro to R: Week 9

Topics Covered: Oce package and bathymetry data

This week, we will explore some functions of the `Oce` package. This package was created by a physical oceanographer, Dr. Dan Kelley, and is designed for oceanography applications. In order to complete the exercises, you will need to install the following packages:

Task 1: Sun angle
This example was based on a code created by Dan Kelley, available at http://dankelley.github.io/r/2014/03/22/sun-moon.html.

Step 1.1 Run the code below to generate a plot of the sun angle in Halifax, NS today.

```
angles <- function(day=Sys.Date(), lon=-63.61, lat=44.67, tz="America/Halifax", sun=TRUE)
{
  tUTC <- t <- seq(as.POSIXct(paste(day, "00:00:00"), tz=tz), length.out=240, by="6 min")
  attributes(tUTC)$tzone <- "UTC"
  a <- if (sun) sunAngle(tUTC, lon=lon, lat=lat) else moonAngle(tUTC, lon=lon, lat=lat)
  invisible <- a$altitude < 0
  a$altitude[invisible] <- NA
  a$azimuth[invisible] <- NA
  list(t=t, altitude=a$altitude, azimuth=a$azimuth)
}

day <- Sys.Date()
tz <- "America/Halifax"
s <- angles()
m <- angles(sun=FALSE)
max <- 1.04 * max(c(s$altitude, m$altitude), na.rm=TRUE)

par(mar=rep(0.5, 4))
theta <- seq(0, 2*pi, length.out=24 * 10)
radiusx <- cos(theta)
radiusy <- sin(theta)

# Horizon and labels+lines for EW and NS
plot(radiusx, radiusy, type='l', col='gray', asp=1, axes=FALSE, xlab="", ylab="")
lines(c(-1, 1), c(0, 0), col='gray')
lines(c(0, 0), c(-1, 1), col='gray')
D <- 1.06
text( 0, -D, "S", xpd=TRUE) # xpd so can go in margin
text(-D,  0, "W", xpd=TRUE)
text( 0,  D, "N", xpd=TRUE)
text( D,  0, "E", xpd=TRUE)

## Moon
mx <- (90 - m$altitude) / 90 * cos(pi / 180 * (90 - m$azimuth))
my <- (90 - m$altitude) / 90 * sin(pi / 180 * (90 - m$azimuth))
lines(mx, my, col='blue', lwd=3)
t <- s$t
mlt <- as.POSIXct(sprintf("%s %02d:00:00", day, 1:24), tz=tz)
```

```r
ti <- unlist(lapply(mlt, function(X) which.min(abs(X-t))))
points(mx[ti], my[ti], pch=20, cex=3, col='white')
text(mx[ti], my[ti], 1:24, cex=3/4)

## Sun
sx <- (90 - s$altitude) / 90 *  cos(pi / 180 * (90 - s$azimuth))
sy <- (90 - s$altitude) / 90 *  sin(pi / 180 * (90 - s$azimuth))
lines(sx, sy, col='red', lwd=3)
slt <- as.POSIXct(sprintf("%s %02d:00:00", day, 1:24), tz=tz)
si <- unlist(lapply(slt, function(X) which.min(abs(X-t))))
points(sx[ti], sy[ti], pch=20, cex=3, col='white')
text(sx[ti], sy[ti], 1:24, cex=3/4)

mtext(paste("Halifax NS", day, sep='\n'), side=3, adj=0, line=-2)
mtext("Sun angle", side=3, adj=1, line=-2)
```

Step 1.2 Modify the code above to plot the sun angle today in San Diego, CA. Remember the function `OlsonNames()` and that San Diego has coordinates 32.72° N and 117.16° W. You can look up the sun's zenith on your plot and compare it to the one at the following website (http://www.timeanddate.com/astronomy/usa/san-diego).

Step 1.3 Modify the code above to plot the sun angle on December 12, 2014 in San Diego, CA.

Step 1.4 Pretend you are planning a field trip in July to monitor spawning in Little Cayman (19.68 ° N, 80.05 ° W). You know that the moon plays an important role in this process. Create a data frame with hourly values of the illuminated fraction of the moon for the entire month of July 2015. To make your life easier for sampling and data processing, create a column with time in UTC and another one with local time (the Cayman Islands use Eastern Standard Time year round).

Step 1.5 Use `ggplot()` to plot the illuminated fraction of the moon for the month of July 2015, using Little Cayman local time.

Task 2: Quick functions

Now, we will quickly explore some functions of the `Oce` package that could be useful to you.

Step 2.1 Use the `Oce` package to calculate the Coriolis parameter at 45° N.

Step 2.2 Create the data below and use the function `binAverage()` in the `Oce` package to calculate an average value for each 5 meters. Plot the original data in grey, and the averages in red. Note: ggplot() did not work for me, but maybe it will for you.

```r
# Data
z <- seq(1, 100)  # depth, in [m]
y <- 5 + 2*z
```

Step 2.3 The `Oce` package has functions that allows you to look at certain data format quickly. After calling current meter data, use `plot(cm)` to look at it. How do you find information on this function?

```r
# Data
data(cm)

# Quick look at the data
summary(cm)
plot(cm)
```

Step 2.4 Use `data(ctd)` to obtain sample ctd data and plot it on a TS diagram. Change the color of the isopycnals to purple.

Task 3: Working with bathymetry data

Step 3.1 Import the central California bathymetry data from 20050622cacentral3sec.asc and the harbor porpoise sighting information from PpSightings.RData. Use the bathymetry data to determine the water depth of each harbor porpoise sighting, then plot a histogram of harbor porpoise sighting depths.