Intro to R: Week 5

Topics Covered: Date/Time Data and Data Manipulation

Before we begin, you will need to install several packages. Please do so before class to make sure that everything is working properly.

```
install.packages("lubridate")
install.packages("reshape2")
install.packages("dplyr")
install.packages("ggplot2")
```

Part 1: Dealing with Date/Time Data

Task 1: Introducing the `lubridate` package

Step 1.1 Load the lubridate library. Use `?lubridate` to see what this package has to offer.

```
library("lubridate")

?lubridate
```

Step 1.2 What is the current system time? What is the current time in London? Hint: use `now()` and `with_tz()`.

```
now()

class(now()) # this is a time class object

with_tz(now(), tz="GMT") # Use OlsonNames() to see all possible timezones
```

Step 1.3 On what day of the week were you born? Hint: use `ymd()` and `wday()`. On what day of the week was your 21st birthday? Hint: use `years()`.

```
bday <- ymd("1988 10 23", # This could also be 1988-10-23 or 1988 10 23 etc.
            tz="EST5EDT") # I was born on the East coast

wday(bday, label=TRUE) # Note that this function is returning an ordered factor
                       # without label=TRUE, the returned value is numeric

bday.21 <- bday + years(21)

wday(bday.21, label=TRUE)
```

Task 2: Importing, formatting, and binning data from the SIO Pier

Step 2.1 Read in the shore stations data from the SIO pier, plot the temperature measurements, then add a column with a POSIX timestamp. Hint: use `paste()` and `ymd_hms()`.

```
setwd("~/Desktop/IntroR/Week 5/")

# I downloaded these data directly from http://www.sccoos.org/query/
# SCCOOS supplies header metadata which we don't want, so I set skip=6
# The header info did specify that all times are provided in UTC

sio.pier <- read.csv("Shore Stations_Jan-Apr 2015.csv",
                     skip=6, header=TRUE, stringsAsFactors=FALSE)

plot(sio.pier$Water.Temperature..C., type="l")

# Stick all of the date time information together into a single string
tstring <- paste(sio.pier$year, sio.pier$month, sio.pier$day, sio.pier$time)

# Convert the string to a POSIX class object
timestamp <- ymd_hms(tstring, tz="UTC")

# Now add that information back into the data frame
sio.pier$Timestamp <- timestamp
```

Step 2.2 Use `plot()` or `ggplot()` to plot time v. temperature for the SIO pier record.

```
plot(sio.pier$Timestamp, sio.pier$Water.Temperature..C., type="l")

library(ggplot2)
ggplot(data=sio.pier,
       aes(x=Timestamp, y=Water.Temperature..C.)) +
  geom_line()
```

Step 2.3 These measurements were taken every six minutes, and they're noisy. Write a function that uses `interval()` and `within()` to bin data by a specified number of hours. Hint: the structure of your function will be very similar to the binning function we created last week for CTD data.

```
binPier <- function(data, bin.length){

  # Function to average SIO Pier measurements into bins of specified no. of hrs
  # data is a data frame of SCCOOS Pier data
  # bin.size is the number of hours data should be binned into
  # The function returns a data frame with the timestamp and averaged measurements

  df <- data

  # convert the date/time data into a POSIX timestamp column
  df$Timestamp <- ymd_hms(paste(df$year, df$month, df$day, df$time), tz="UTC")

  # create a sequence of time bins from the start to the end of the data record
  t.bins <- seq(range(df$Timestamp)[1],
                range(df$Timestamp)[2],
                by=bin.length*60*60) # need to specify this in secs, not hrs

  df.out <- data.frame()

  for (t in 1:(length(t.bins)-1)){
```

```
    int <- interval(t.bins[t], t.bins[t+1])

    i <- which(df$Timestamp %within% int)

    # note that you could also use
    # i <- which(df$Timestamp >= t.bins[t] & df$Timestamp < t.bins[t+1])

    nd <- data.frame("Bin.Start"=t.bins[t],
                     "Avg.Temp"=mean(df$Water.Temperature..C.[i]),
                     "Avg.Chl"=mean(df$Chlorophyll..ug.L.[i]),
                     "Avg.Sal"=mean(df$Salinity..PSU.[i]))

    df.out <- rbind(df.out, nd)

  } # end t

  return(df.out)

} # end binPier

# Run the function with a bin length of 6 hrs
binned.data <- binPier(sio.pier, 6)
```

Step 2.4 Now calculate the average temperature in 24 hour bins and plot the day of the year vs. temperature using plot or ggplot2. Hint: use `yday()`.

```
# Run the function with a bin length of 24 hrs
pier.24 <- binPier(sio.pier, 24)

# Plotting

plot(yday(pier.24$Bin.Start), pier.24$Avg.Temp, type="l")

ggplot(pier.24, aes(x=yday(Bin.Start), y=Avg.Temp))+
  geom_line()
```

Part 2: Data Manipulation

There are whole books written about data manipulation (also called data wrangling) in R, so this will be a very very brief overview to introduce you to a few of the packages and functions available.

Task 3: Handy data manipulation functions in base R

Step 3.1 Bin your SIO pier data by 1 hr, then use the `apply()` and `fivenum()` function to return the minimum, first quartile, median, third quartile, and maximum values of average temperature, average chlorophyll, and average salinity from your binned SIO pier dataset.

```
pier.1hr <- binPier(sio.pier, 1)

pier.summary <- apply(pier.1hr[,2:4], 2, fivenum)
```

This is a very minimal example of the `apply` family of functions. Basically, anything that you can do with a `for` loop you can also do with `apply`.

Step 3.2 Use `table()` to determine how many measurements were collected in each month of the SIO pier record.

```
table(month(sio.pier$Timestamp, label=TRUE))
```

Task 4: Using the `reshape2` package

Note Many of the functions in the `reshape2` package have analogs in the `dplyr` package. I'm more familiar with the `reshape2` package so that's what I'll present here, but if you're interested in data wrangling definitely check out `dplyr`!

Step 4.1 Use the `melt()` and `dcast()` functions to generate a data frame of average temperature, salinity, and chlorophyll by month using your binned one hour data frame. This should not take more than two lines of code. *Bonus:* Use ggplot to create a three-panel timeseries plot of temperature, chlorophyll, and salinity.

```
library(reshape2)

# Melt the data usin Bin.Start as the identifying variable
m.pier <- melt(pier.1hr, id.var="Bin.Start")
# now, each row = one observation

# Now recast the data with months in the rows and measurements in columns,
# using mean() to calculate the monthly averages.  Note that because there are
# missing values, we have to include na.rm=TRUE

mean.pier <- dcast(m.pier,
                   month(Bin.Start, label=TRUE)~variable,
                   mean, na.rm=TRUE)

names(mean.pier)[1] <- "Month" # Tidy up the data frame

### Plotting

ggplot(m.pier, aes(x=Bin.Start, y=value, color=variable))+
  geom_line()+
  facet_wrap(~variable, ncol=1, scales="free_y")

# Note: yes, there is something funky about the chlorophyll measurements!
```

Step 4.2 Read in the islands dataset and create a data frame of average percent hard corals, soft corals, etc. by island. Again, manipulating the data should not take more than two lines of code. *Bonus* Use `ggplot2` to create pie charts of average cover on each island.

```
# Read in the islands dataset
islands <- read.csv("IslandsData.csv", header=TRUE, stringsAsFactors=FALSE)

# Melt the data, using both the Island and the Quadrat as identifiers
l.islands <- melt(islands, id.vars=c("Island", "Quadrat"))

# Now recast the data with Island as rows and cover type in columns
avg.islands <- dcast(l.islands, Island~variable, mean)

### Plotting

# re-melt so we can plot the averages
long.avg <- melt(avg.islands)

ggplot(long.avg, aes(x="", y=value, fill=factor(variable)))+
  geom_bar(stat="identity", width=1)+
  coord_polar(theta="y")+
  facet_wrap(~Island)
```

Task 5: Regular Expressions and Partial String Matching

Step 5.1 Load the file SampleIndexes.csv and extract the samples which come from the cruise with identifier 1311COFI. Hint: use `grep()`.

```
# Read in the data
indexes <- read.csv("SampleIndexes.csv", header=TRUE, stringsAsFactors=FALSE)

# Find the location of the samples from the CalCOFI cruise
grep("1311COFI", indexes$LABID)

# Use that information to pull out the CalCOFI samples
COFI <- indexes[grep("1311COFI", indexes$LABID),]
```

Step 5.2 Which of the CalCOFI sample indexes begin with the sequence ATC?

```
grep("^ATC", COFI$INDEX)

# note that if you use grep("ATC", COFI$INDEX) the answer is different
# because grep will look for the pattern anywhere in the string rather than
# just at the beginning
```

Step 5.3 Were any samples labeled with indexes GCCGCG, GGGCCA, or CATTTT? Hint: use `%in%` or `match()`.

```
match(c("GCCGCG", "GGGCCA", "CATTTT"), COFI$INDEX)

c("GCCGCG", "GGGCCA", "CATTTT") %in% COFI$INDEX
```