Intro to R: Week 6

Topics Covered: Plotting and Mapping Data

```r
# First, set up your working directory and load required packages

setwd("~/Desktop/IntroR/Week 6/")

library("ggplot2")
library("ggmap")
library("lubridate")
library("reshape2")
library("dplyr")
```

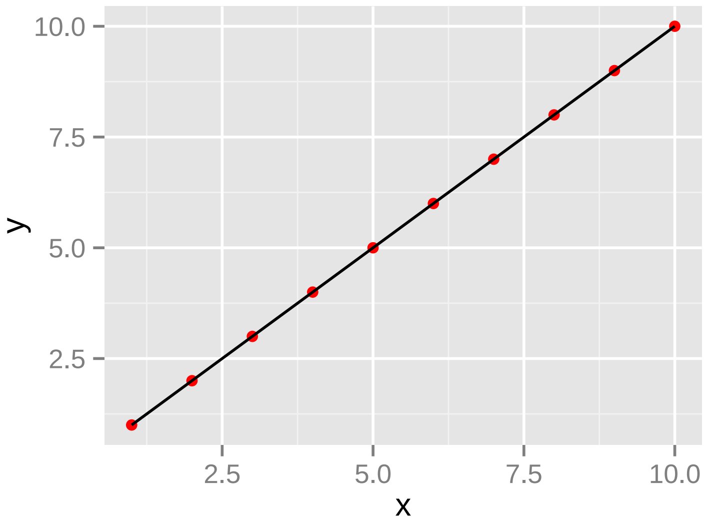
I like to think that Excel:`R`::`plot`:`ggplot2`. Like `R`, `ggplot2` requires more of an upfront time investment to learn but will save you time (and produce better results!) in the long run. This will be a very brief introduction to familiarize you with the (quirky) syntax of ggplot and some of the options available for plotting and mapping data.

Task 1: Exploring ggplot syntax

`ggplot` works by incrementally adding layers to a plot object using the `+` symbol.

Step 1.1 Create a very simple data frame with values of x from 1:10 and values of y from 1:10. Plot x versus y as points and lines and make the points red.

```r
df <- data.frame(x=1:10, y=1:10)

ggplot(data=df, aes(x=x, y=y)) + # assign global data and aesthetic mappings
  geom_point(color="red")+ # create a point layer
  geom_line() # create a line layer
```



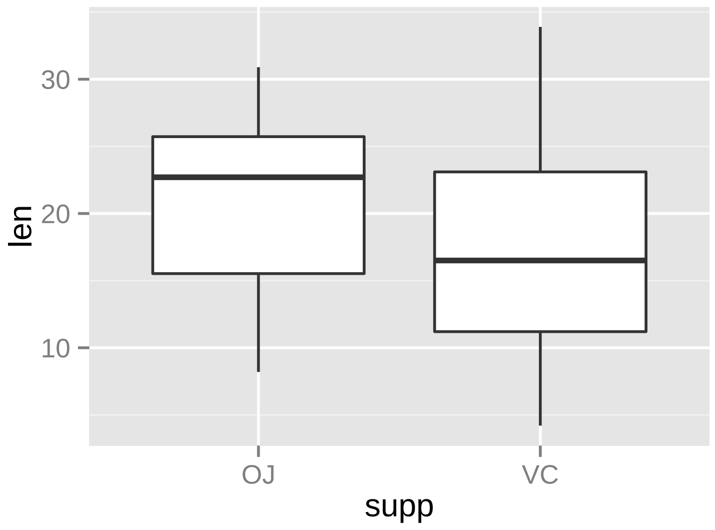
Things to notice:

- Data must be contained within a data frame and in "long" format (melted)
- Order matters. In this case, the line is plotted on top of the points
- Data assigned in the first call to ggplot are globally available to all layers
- Properties (like color or size) that depend on a variable in the data frame must be mapped to that variable within an aes() statement

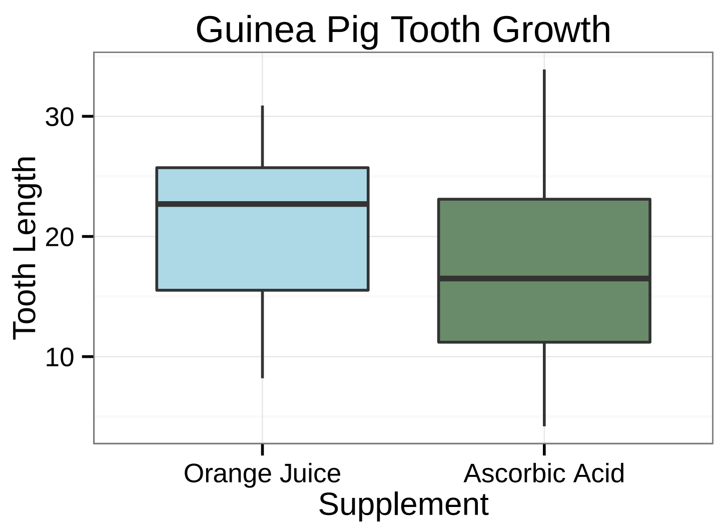
Task 2: Boxplots and Error Bars with the Guinea Pig Tooth Dataset

Step 2.1 Load the ToothGrowth dataset and make a boxplot of supplement vs. tooth length.

```
tg <- ToothGrowth

ggplot(tg, aes(x=supp, y=len))+ # map data to x and y variables
  geom_boxplot() # specify the type of plot
```

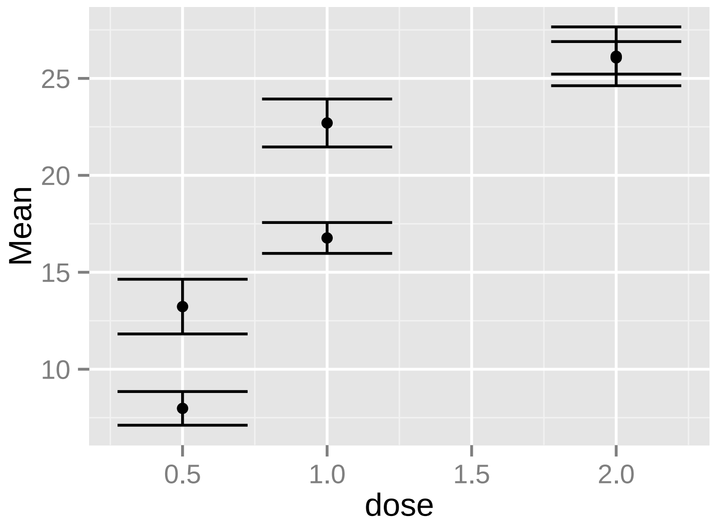


```
# now we can build on this plot to specify colors, labels, etc.

ggplot(tg, aes(x=supp, y=len, fill=supp))+ # color fill is mapped to supp
  geom_boxplot()+ #
  xlab("Supplement")+
  ylab("Tooth Length")+
  ggtitle("Guinea Pig Tooth Growth")+
  scale_fill_manual(values=c("lightblue", "darkseagreen4"))+ # specify colors
  scale_x_discrete(labels=c("Orange Juice", "Ascorbic Acid"))+ # change labels
  theme_bw()+ # get rid of the gray background
  theme(legend.position="none") # get rid of the legend
```



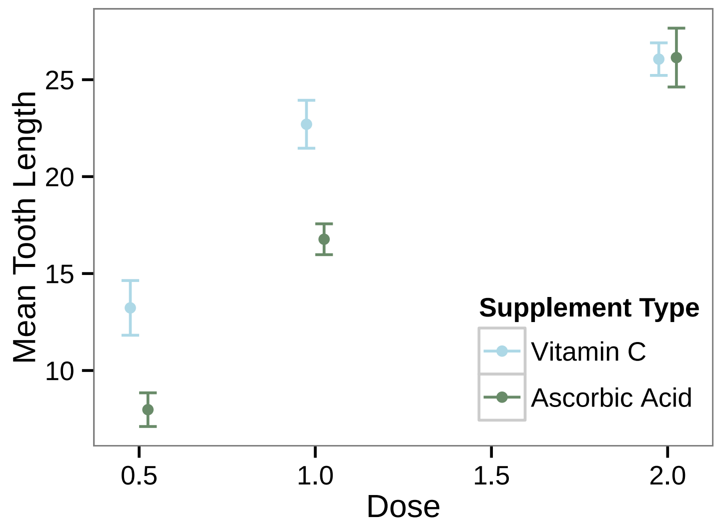
Step 2.2 Create a scatterplot of dose v. mean tooth length for both supplement types. Add bars to represent the standard error of the mean.

```
# First we need to do a bit of data manipulation to calculate the mean and SE
# for each of the supplement types at each dose level

# Define a quick funciton to calculate standard error
se <- function(x) sqrt(var(x,na.rm=TRUE)/length(na.omit(x)))

# Use the summarize function in the dplyr package to apply this function
summary <- summarize(group_by(tg, supp, dose), Mean=mean(len), SE=se(len))

# Now plot
ggplot(summary, aes(x=dose, y=Mean, group=supp))+
  geom_point()+
  geom_errorbar(aes(ymin=Mean-SE, ymax=Mean+SE))
```



```
# Note: if you want error bars in x, use geom_errorbarh.  Syntax is the same.
```

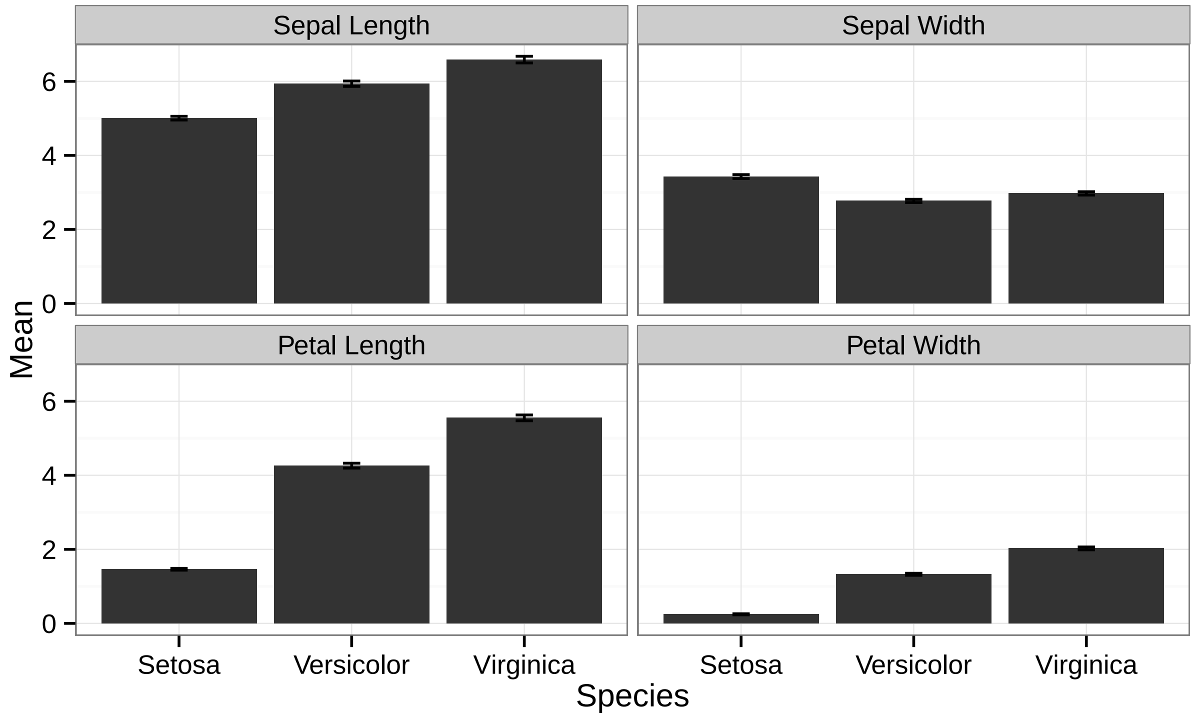
```r
# Pretty ugly, but we can fix it!
ggplot(summary, aes(x=dose, y=Mean, color=supp, group=supp))+ # data mapping
  geom_point(position=position_dodge(width=0.1))+ # points, not overlapping
  geom_errorbar(position=position_dodge(width=0.1), # avoid overlapping points
                aes(ymin=Mean-SE, ymax=Mean+SE), width=0.1)+ # errorbar mapping
  scale_color_manual(name="Supplement Type", # legend title
                     labels=c("Vitamin C", "Ascorbic Acid"), # types of supplements
                     values=c("lightblue", "darkseagreen4"))+ # colors to use
  xlab("Dose")+ # change x-axis label
  ylab("Mean Tooth Length")+ # change y-axis label
  theme_bw()+ # get rid of gray background
  theme(legend.position=c(0.8, 0.2))+ # move the legend onto the plot
  theme(panel.grid.major=element_blank(), # get rid of major
        panel.grid.minor=element_blank()) # and minor grid lines
```



Task 3: Barplots with the Iris Dataset

Step 3.1 Load the built-in iris dataset and create barplots of mean sepal length, sepal width, petal length, and petal width by species in four panels.

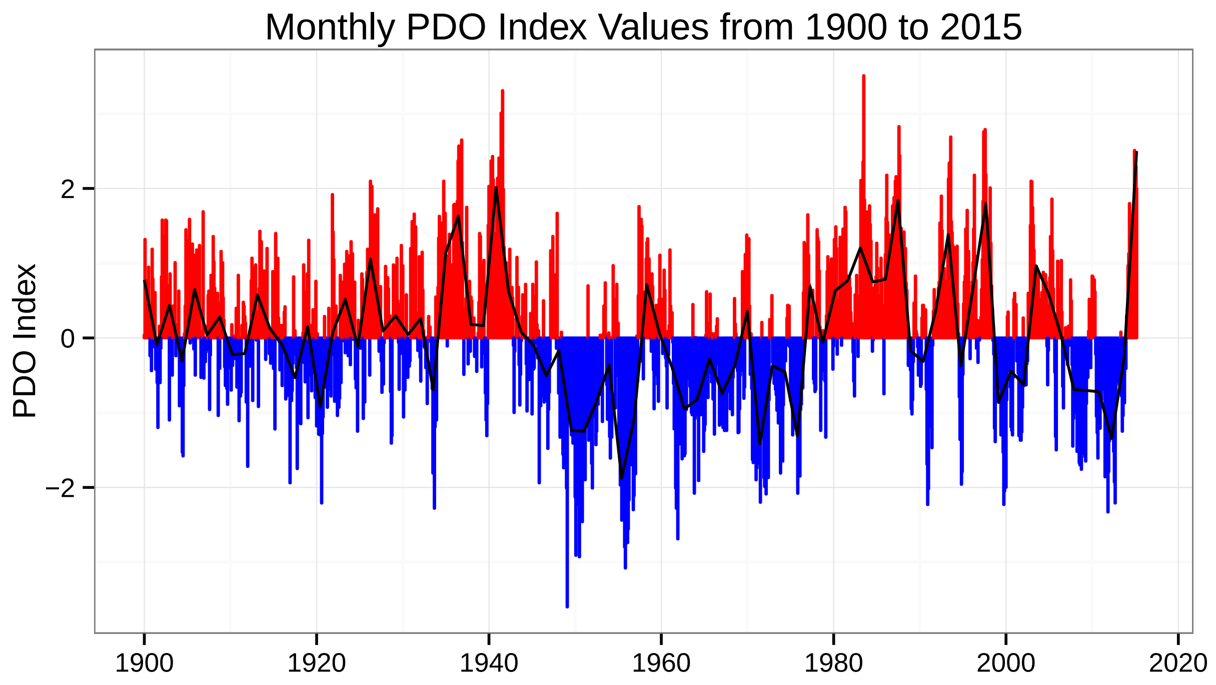
```r
names(iris)[1:4] <- c("Sepal Length", "Sepal Width",
                      "Petal Length", "Petal Width")

m.iris <- melt(iris, id.var="Species")

summary.iris <- summarize(group_by(m.iris, Species, variable),
                          Mean=mean(value), SE=se(value))

ggplot(summary.iris, aes(x=Species, y=Mean))+
  geom_bar(stat="identity")+
  geom_errorbar(aes(ymin=Mean-SE, ymax=Mean+SE), width=0.1)+
  facet_wrap(~variable)+
  scale_x_discrete(labels=c("Setosa", "Versicolor", "Virginica"))+
  theme_bw()+
  theme(legend.position="none")
```



Task 4: Plotting anomaly data

Step 4.1 Plot the PDO index in `PDO.txt` over time so that negative values are blue and positive values are red. Add a Loess smooth in black. Save the plot to a pdf.

```
# Plotting Anomaly Data

# I downloaded these data from:
# http://research.jisao.washington.edu/pdo/PDO.latest
# The goal is to make a plot that looks like this one:
# http://cses.washington.edu/cig/figures/pdoindex_big.gif

pdo <- read.delim("PDO.txt", sep="", skip=29, header=TRUE, nrow=116)

m.pdo <- melt(pdo, id.var="YEAR")

m.pdo$Timestamp <- ymd(paste(gsub("[**]", "", m.pdo$YEAR),
                       match(tolower(m.pdo$variable), tolower(month.abb)),
                       1))

PDO <- data.frame("Timestamp"=m.pdo$Timestamp,
                  "PDO"=m.pdo$value,
                  "Anomaly"=ifelse(m.pdo$value > 0, "P", "N"))

pdo.plot <- ggplot(PDO, aes(x=Timestamp, y=PDO))+
                   geom_bar(aes(color=Anomaly), stat="identity",
                   position="identity", show_guide=FALSE)+
                   geom_smooth(method="loess",
                               se=FALSE, color="black", span=0.025)+
                   scale_color_manual(values=c("blue", "red"))+
                   ggtitle("Monthly PDO Index Values from 1900 to 2015")+
                   xlab("")+
                   ylab("PDO Index")+
                   theme_bw()

pdo.plot
```

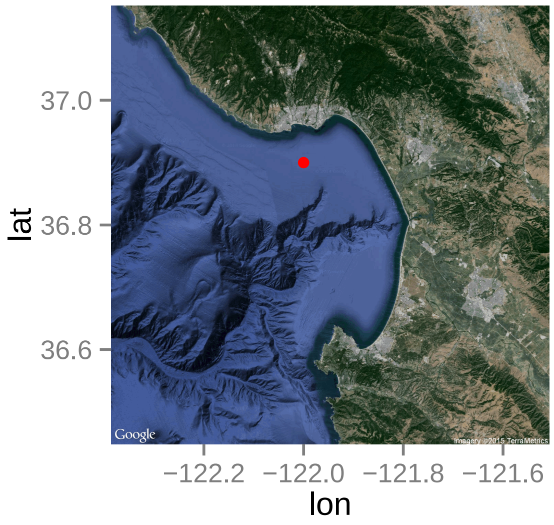


```
ggsave(filename="PDOPlot.pdf", plot=pdo.plot)

# Nailed it.
```

Task 5: Exploring ggmap

Step 5.1 Use `get_map()` and `ggmap()` to plot a satellite map of Monterey Bay. Add a red point at 36.9 deg N, -122.0 deg W.

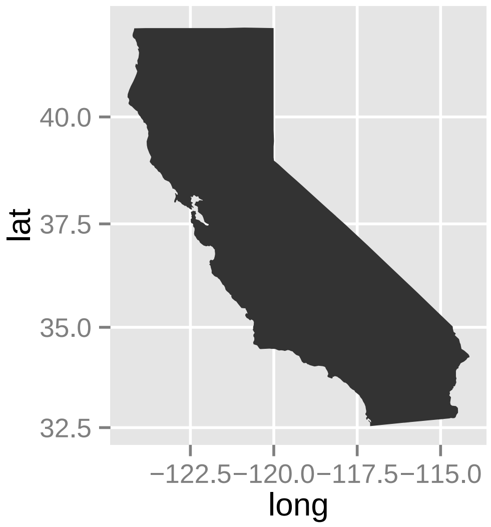
```
get.mbay <- get_map("monterey bay", maptype="satellite")

loc <- data.frame("lon"=-122.0,"lat"=36.9)

ggmap(get.mbay) +
  geom_point(data=loc,
  aes(x=lon, y=lat), size=2, color="red")
```



Task 6: Mapping with `ggplot`

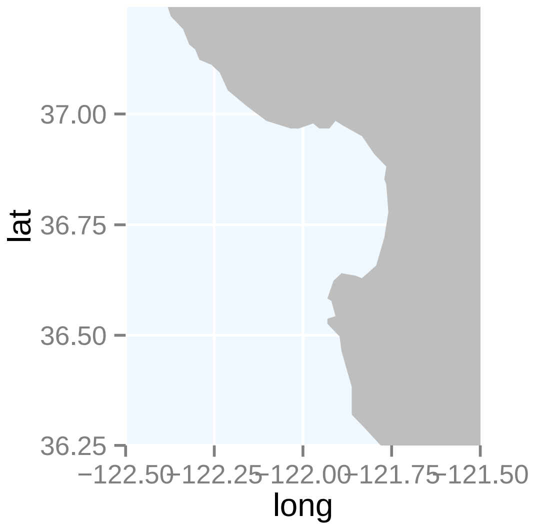
Task 6.1 Use `map_data()` and `ggplot()` to plot a map of California

```
usa<-map_data("usa")
states <- map_data("state")
california<-states[states$region=="california",]

ggplot()+
  geom_polygon(data=california,
               aes(long,lat)) +
  coord_map() # this corrects the aspect ratio
```



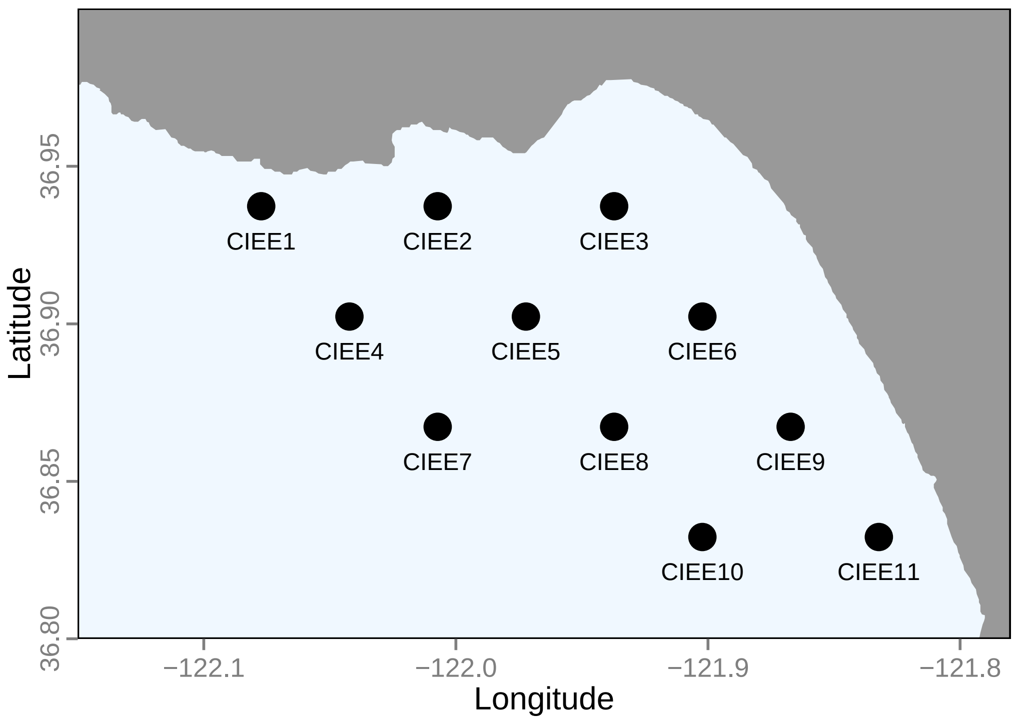
Task 6.2 Adjust `coord_map` to zoom in Monterey Bay. Make the ocean blue and the land gray.

```
ggplot()+
  geom_polygon(data=california,
               aes(long,lat),
               fill="grey") + # make California gray
  coord_map(xlim=c(-122.5,-121.5), # specify the range of longitudes
          ylim=c(36.25,37.24)) + # specify the range of latitudes
  theme(panel.background=element_rect(fill="aliceblue")) # blue background
```



Task 6.3 Use the coastline coordinates in `mbaycoast.csv` and the mooring locations in `cpods.csv` to plot a more detailed map of Monterey Bay with labeled points at the mooring locations.

```
coast <- read.csv("mbaycoast.csv", header=TRUE, stringsAsFactors=FALSE)
cpods <- read.csv("cpods.csv", header=TRUE, stringsAsFactors=FALSE)

ggplot(coast,aes(lon,lat)) + # Call ggplot and give it aesthetics
  geom_polygon(fill="grey60") + # Add a gray polygon for land
  coord_map(xlim=c(-122.15,-121.78),ylim=c(36.8,37)) + # Zoom N. Monterey Bay
  geom_point(data=cpods,aes(lon,lat),size=5) + # Add points at mooring locations
  geom_text(data=cpods, # add labels to the points
          aes(label=as.character(location),lon,lat),
          size=3,vjust=2.5)+
  xlab("Longitude") + # changing the name of the x-axis
  ylab("Latitude")+
  theme(panel.grid.minor=element_blank(), # removing gridlines
            panel.grid.major=element_blank(), # removing gridlines
            axis.text.y=element_text(angle=90,hjust=0.5), # rotating y-axis text
            panel.border = element_rect(fill=NA,colour="black"), # black border
            panel.background=element_rect(fill="aliceblue")) # blue ocean
```



CIEE1
CIEE2
CIEE3
CIEE4
CIEE5
CIEE6
CIEE7
CIEE8
CIEE9
CIEE10
CIEE11
Latitude
36.95
36.90
36.85
36.80
−122.1
−122.0
−121.9
−121.8
Longitude