Plotting in R

Alex Potash

1/20/2018

How to make plots in R using base plot and ggplot2

R allows you to plot your data and customize the way the figure looks using base functions that are preloaded in R. These base functions make visualizing data very simple and are relatively straightforward once you've learned the basics of coding in R.

The ggplot2 package (hereafter referred to as ggplot), is a free package that also plots data, but allows for much greater customization than the base functions. ggplot can even perform statistical analyses and directly plot the results. However, the code to create a plot in ggplot can be somewhat hard to interpret, especially for beginners.

The following examples can be adapted to create the majority of plots you will need to make in this semester's course.

Start by loading the ggplot package and the sample data

```r
install.packages("ggplot2", dependencies = TRUE, repos = "http://cran.us.r-project.org")
# Only need to install the package one time
library(ggplot2) # Need to load the ggplot2 package every time you start R
data("iris") # Iris is a famous dataset that is built into R and is
# commonly used as an example dataset

# We will also use data from UF professors Dr. Earnest and Dr. White who
# have made their data freely available in the package portalr
install.packages("portalr", dependencies = TRUE, repos = "http://cran.us.r-project.org")
# Only need to install the package once
library(portalr) # Load the portalr package
download_observations() # This downloads the portal data and stores
# it on your computer
rodent=data.frame(abundance(shape="flat")) # Load the rodent abundance
# data. shape="flat" is only necessary in this instance to read in the
# data in a more useful format.
```

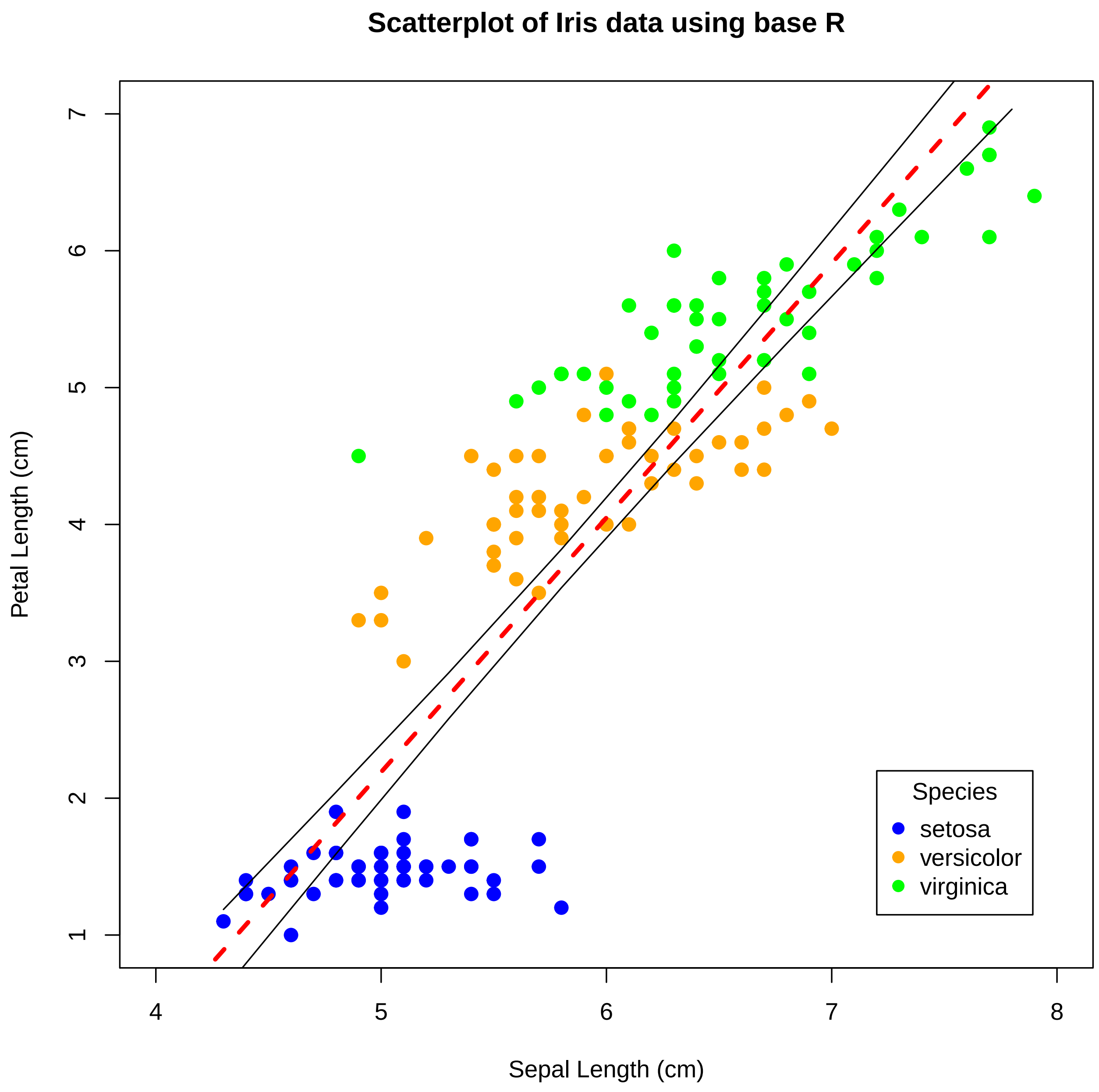
Scatterplots

Scatterplots are used to show data with a continuous y-variable and a continuous x-variable

Scatter plots using base R

```r
pdf("BaseRScatterplot.pdf", height = 8, width = 8) # Save the plot you're
# about to create as a pdf and make it 8 inches tall by 8 inches wide.
# Inches are the default unit in base R
```

```r
plot(x=iris$Sepal.Length, y=iris$Petal.Length, # Define the data for the x and y axis
     xlab = "Sepal Length (cm)", # Label the x-axis
     ylab = "Petal Length (cm)", # Label the y-axis
     main = "Scatterplot of Iris data using base R", # Give the figure a title
     xlim = c(4, 8.0), # Set the minimum and maximum values for the x-axis
     ylim = c(1, 7), # Set the minimum and maximum values for the y-axis
     col = c("blue","orange","green")[iris$Species], # Make each species a different color
     cex = 1.2, # Change the size of the points
     pch = 19) # Change the shape of the points

# Creating a 95% confidence requires calculating the linear model between
# the x and y variables and making predictions from the model.
iris_mod=lm(Petal.Length~Sepal.Length,data = iris) # Calculate the linear
# relationship (a linear model) between the x and y variables

predicted_x=seq(min(iris$Sepal.Length),max(iris$Sepal.Length),by=0.5)
# Create new values for the x axis that range from the minimum value of x to
# the maximum value of x in increments of 0.5

pred_iris_mod=predict(iris_mod, # Make predictions using the linear model
                      # for the new values of x
                      newdata = data.frame(Sepal.Length=predicted_x),
                      # These are the x-values that the predictions will be made from
                      interval = "confidence", level = 0.95)
                      # Calculate the 95% confidence interval


abline(iris_mod, # Plot the line from the linear model
       col="red", # Make the color of the line to red
       lty = "dashed", # Change the type of line to dashed
       lwd = 3) # Change the width of the line

lines(predicted_x,pred_iris_mod[,2]) # Plot the line for the lower boundary of the 95% CI
lines(predicted_x,pred_iris_mod[,3]) # Plot the line for the upper boundary of the 95% CI


# Create a legend
legend(x = 7.2, y = 2.2,# Define the x and y coordinates where the legend will be placed.
       # These numbers correspond with the scale of the axis.
       legend = levels(iris$Species), # Define the labels for the legend
       col=c("blue","orange","green"), # Give the legend colors for the points
       title = "Species", # Give the legend a title
       pch=19) # Give the shape of the points in the legend (19 is a filled circle)
```

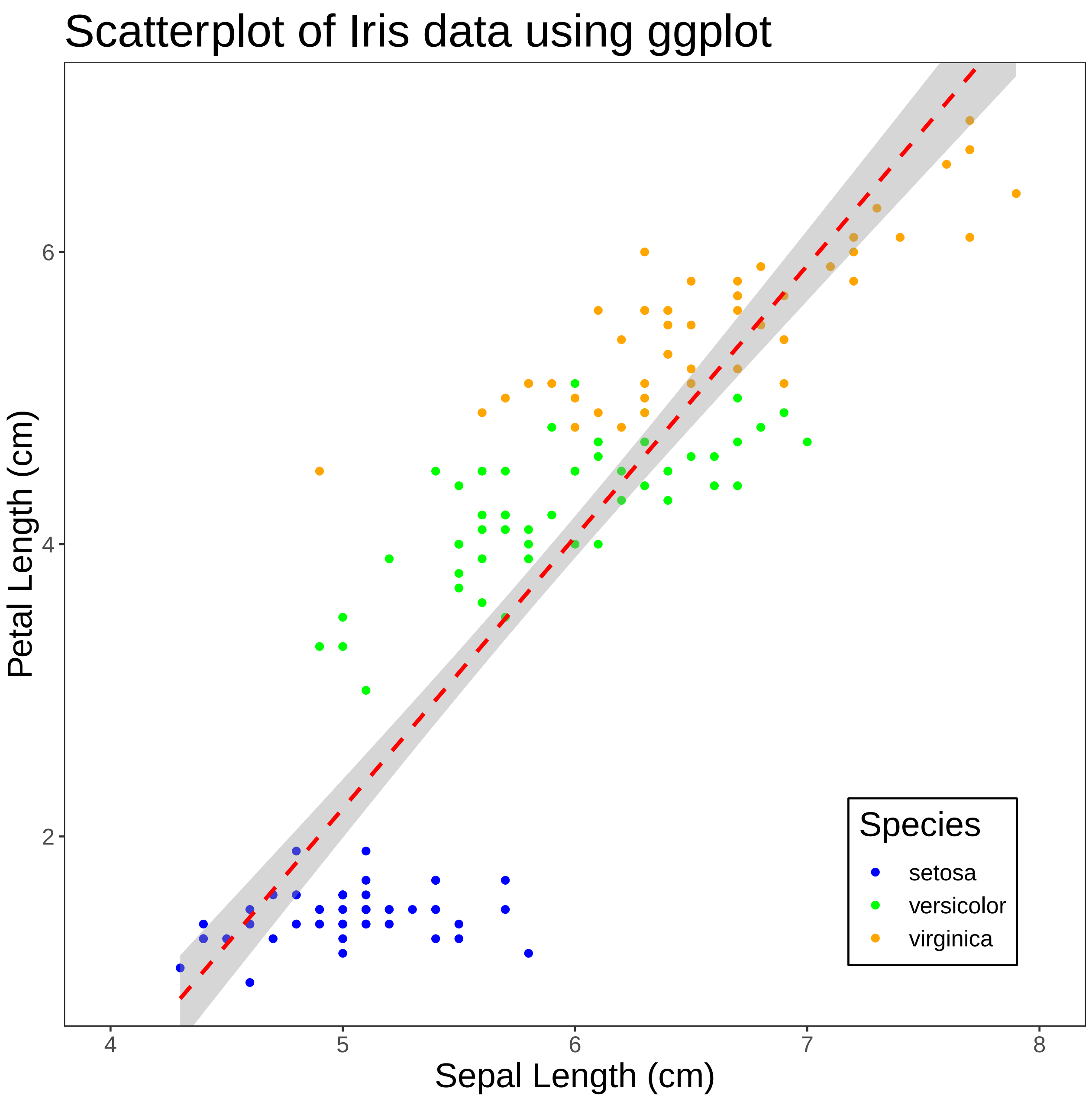


```r
dev.off() # Tell R that the plot is complete and the pdf should be saved
```

Scatterplot using ggplot

```r
ggplot(data = iris, # Define where the data comes from
       aes(x=Sepal.Length, y=Petal.Length,
           # Define the aesthetic, which is the data for the x and y axes
           color=Species))+ # Also in the aesthetic, define the variable to be colored
  geom_point()+ # Display the data as points
  geom_smooth(method = lm, fullrange = T, se = T, level = 0.95,
  # Calculate and draw the line showing the linear relationship between all of the points.
  # By setting se = T, ggplot draws the confidence interval around the line,
  # and setting level=0.95 makes it the 95% CI
```

```r
              color="red", linetype="dashed")+ # Make the line red and dashed
  theme_bw()+ # Get rid of the gray background
  xlab("Sepal Length (cm)")+ # Label the x-axis
  ylab("Petal Length (cm)")+ # Label the y-axis
  ggtitle("Scatterplot of Iris data using ggplot")+ # Give the plot a title
  # labs(x="Sepal Length", y="Petal Length", main="Scatterplot of Iris data using ggplot")
  # Could instead use this to label axes/title
  scale_color_manual(values = c("blue","green","orange"))+ # Set the colors of the different groups
  coord_cartesian(xlim = c(4,8), ylim=c(1,7))+ # Set the x and y axis limits
  theme(axis.title = element_text(size = 18), # Change the text size of
        # the axis labels (sepal length and petal width)
        axis.text = element_text(size = 12), # Change the text size of the numbers on the
        # x and y axis
        title = element_text(size = 20), # Change the text size of the title
        legend.title = element_text(size = 18), # Change the size of the legend heading
        legend.text = element_text(size = 12), # Change the size of the text in the legend
        legend.position = c(.85, .15), # Change legend position.
        # NOTE: Unlike base R, the position of the legend in ggplot is set as a
        # percentage of the x and y axes between 0-1. So, to put the legend in
        # the very middle the position would be c(0.5, 0.5). To put it near the bottom
        # right, it would be c(0.75, 0.1), etc.
        legend.background  = element_rect(colour = "black"), # Make a black rectangle
        # around the legend
        panel.grid.major = element_blank(), # Remove the major grid lines in the background
        panel.grid.minor = element_blank()) # Remove the minor grid lines in the background
```



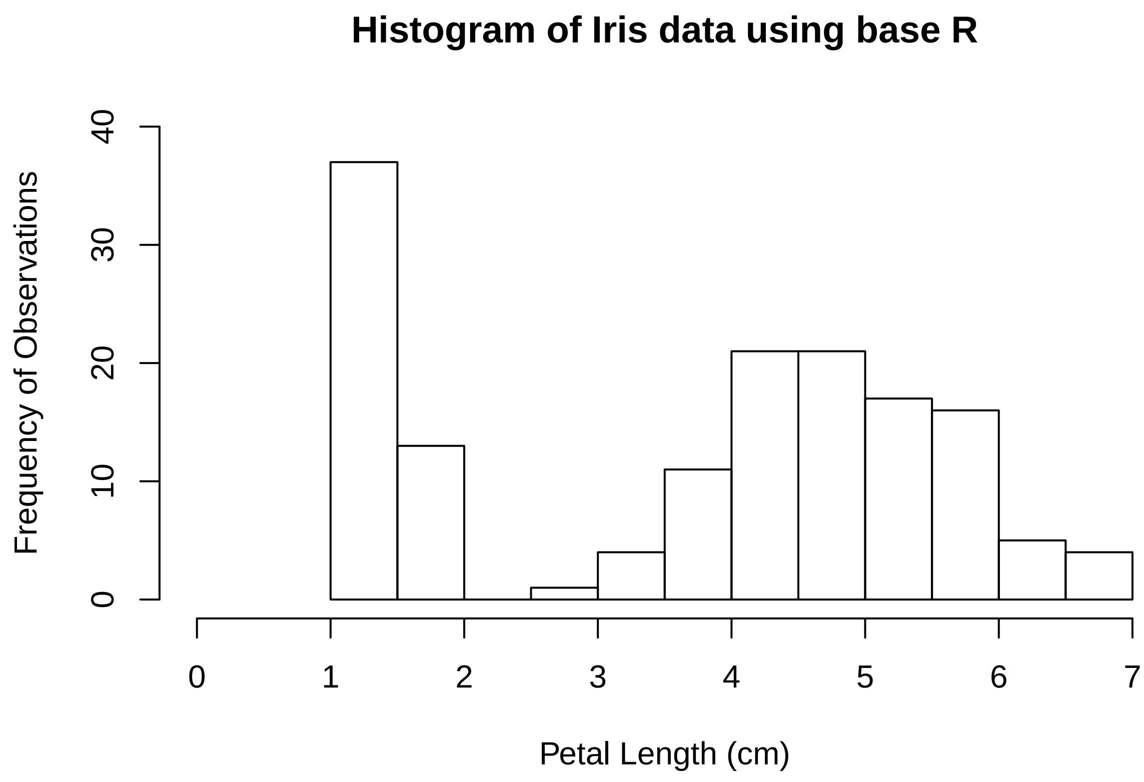
```r
ggsave("ggScatterplot.pdf", # Save the plot as a pdf
       plot=last_plot(), # Make the saved file the most recent plot you created
       width = 8, # Make the saved plot 8 units wide
       height = 8, # Make the saved plot 8 units tall
       units = "in") # Make the height/width units inches
```

Histograms and bar plots

Histograms and bar plots are very similar. Histograms are used for continuous data while bar plots are used for categorical data. Histograms are used to view the distribution of continuous data by binning the values and counting the number of data points that occur in each block. Bar plots look very similar to histograms, but show the number of observations in discrete categories.

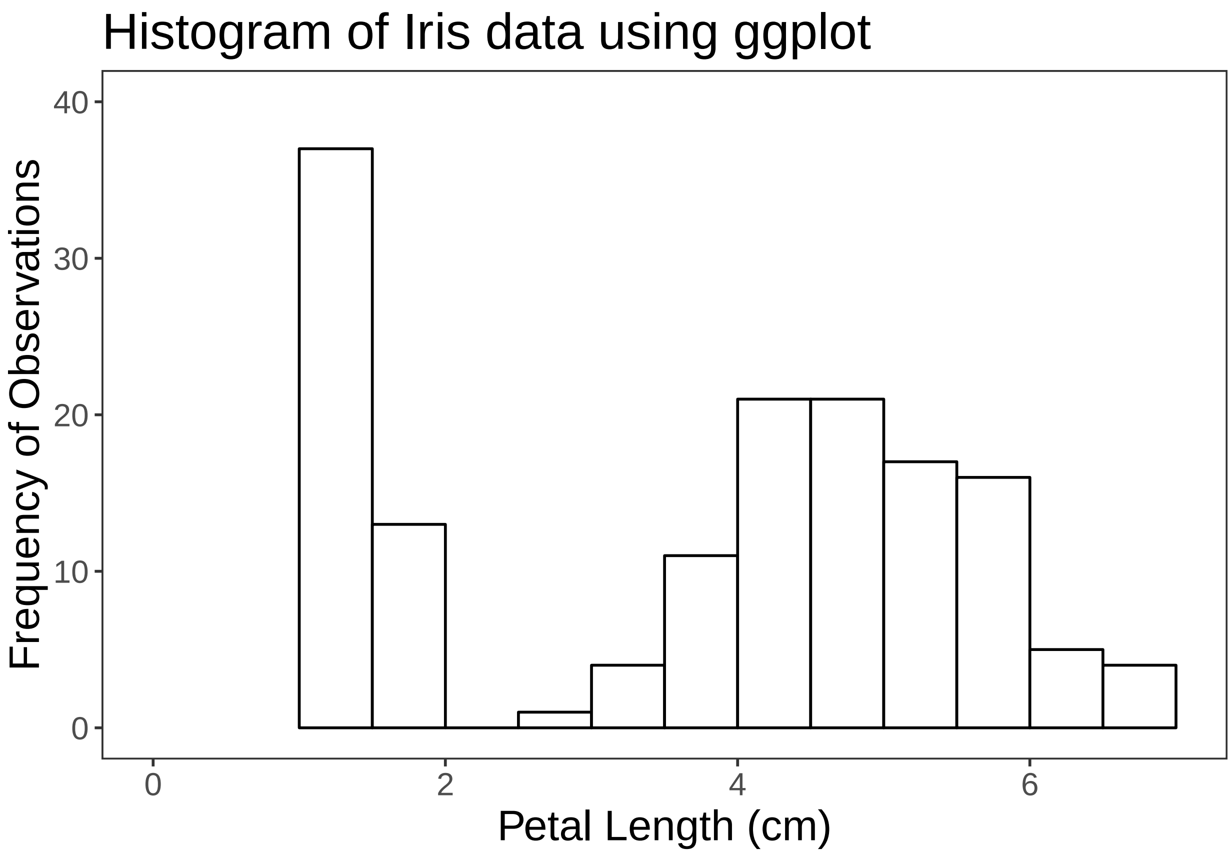
Histograms using base R

```
hist(iris$Petal.Length, # Create the histogram and define the variable to be binned
     breaks = seq(1,7, by = 0.5), # Define the bins which go from 1 to 7 in intervals of 0.5
     ylim = c(0, 40), # Define the range of the y-axis
     xlim = c(0, 7), # Define the range of the x-axis
     xlab = "Petal Length (cm)", # Label the x-axis
     ylab = "Frequency of Observations", # Label the y-axis
     main = "Histogram of Iris data using base R") # Give the plot a title
```



Histogram using ggplot

```
ggplot(data=iris,aes(x=Petal.Length))+ # Define the data and aesthetic.
  # Because a histogram just shows the distribution of a variable, only need
  # to define the x-axis variable
  geom_histogram(binwidth = 0.5, # Define the size of the bins that you want your data
                 boundary = 1, # Set the minimum value of the first bin
                 col="black", # Make the outside of each bar black
                 fill="white")+ # Make the inside of each bar white
  theme_bw()+ # Get rid of the gray background of the plot
  xlab("Petal Length (cm)")+ # Label the x-axis
  ylab("Frequency of Observations")+ # Label the y-axis
  coord_cartesian(xlim = c(0,7), ylim = c(0,40))+ # Set the range of both axes
  ggtitle("Histogram of Iris data using ggplot")+ # Give the plot a title
  theme(title = element_text(size=16), # Change the text size of the title
        axis.title = element_text(size = 16), # Change the text size of
        # the axis labels (sepal length and petal width)
        axis.text = element_text(size = 12), # Change the text size of the numbers on
        # the x and y axis
        panel.grid.major = element_blank(), # Remove the major grid lines in the background
        panel.grid.minor = element_blank()) # Remove the minor grid lines in the background
```



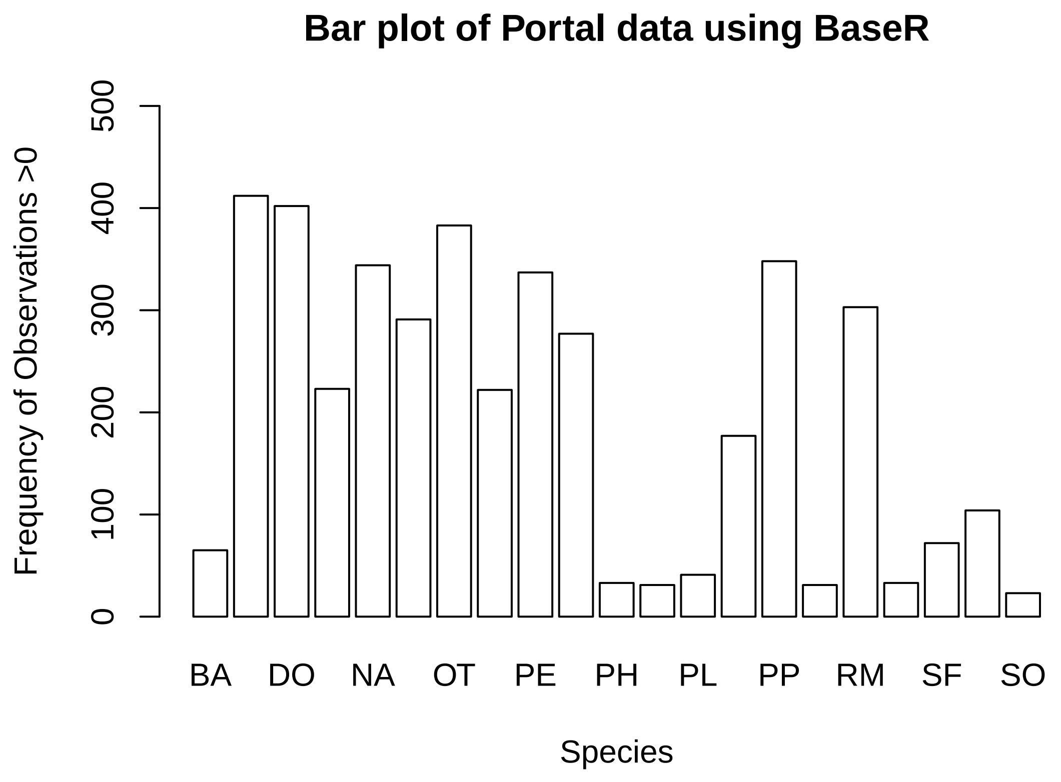
Bar plots

For more interesting results, we will use the Portal data. We will subset the data to only the data where there were >0 rodents observed

```
rodent_abundance_sub=subset(rodent,rodent$abundance>0)
```

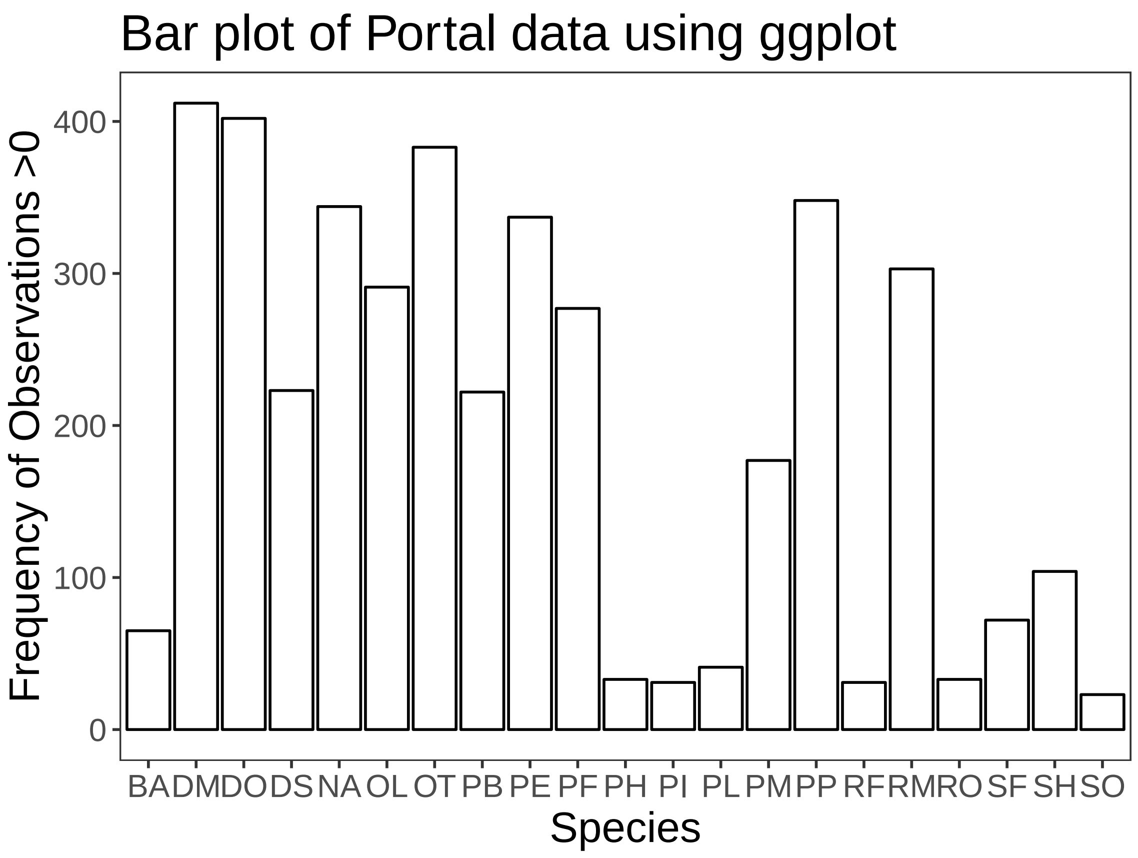
Bar plots using base R

```
plot(rodent_abundance_sub$species, # Define the data for the plot
     xlab = "Species", # Label the x-axis
     ylab = "Frequency of Observations >0", # Label the y-axis
     main = "Bar plot of Portal data using BaseR", # Give the plot a title
     col = c("white"), # Set the inside color for each bar
     ylim = c(0,500)) # Set the range for the y-axis
```



Bar plots using ggplot

```
ggplot(data=rodent_abundance_sub,aes(x=species))+ # Define the data for the plot
  stat_count(col="black", fill="white")+ # Use stat_count to create the bar plot.
  # Make the outside (col) of each bar black and the inside (fill) of each bar white
  theme_bw()+ # Get rid of the gray background in the plot
  xlab("Species")+ # Label the x-axis
  ylab("Frequency of Observations >0")+ # Label the y-axis
  ggtitle("Bar plot of Portal data using ggplot")+ # Give the plot a title
  theme(panel.grid.major = element_blank(), # Get rid of the major grid lines
        panel.grid.minor = element_blank(), # Get rid of the minor grid lines
        axis.title = element_text(size = 16), # Change the text size of
        # the axis labels (sepal length and petal width)
        axis.text = element_text(size = 12), # Change the text size of the numbers on
        # the x and y axis
        title = element_text(size = 16)) # Change the text size of the title
```



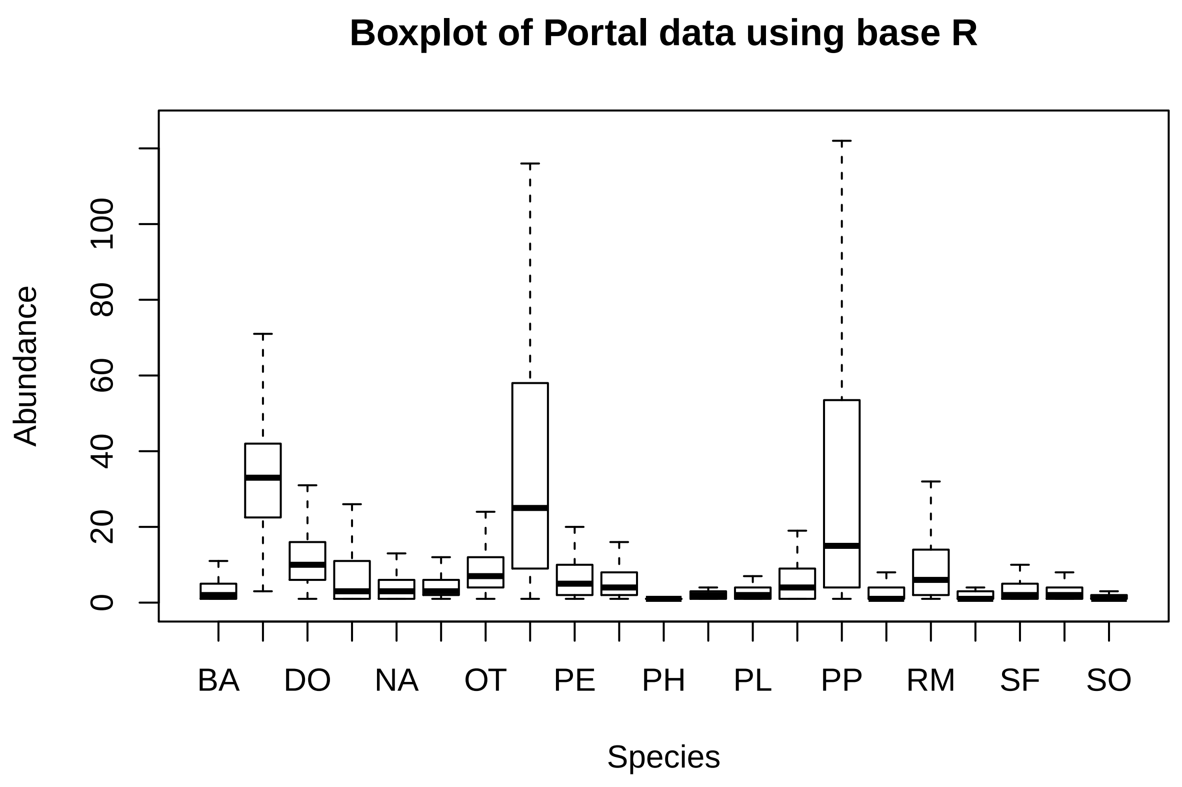
Boxplots and Point Estimates

Boxplots and point estimates can be used to show measures of central tendency for data that are measured continuously and can be broken into distinct groups.

A boxplot describes the data in quartiles; the line within the box is the median, the bottom of the box is the 25th percentile, and the top of the box is the 75th percentile of the data. The whiskers show the spread of the data that fall within a range of 1.5x outside the box. Outliers can also be shown as individual points beyond either end of the whiskers.

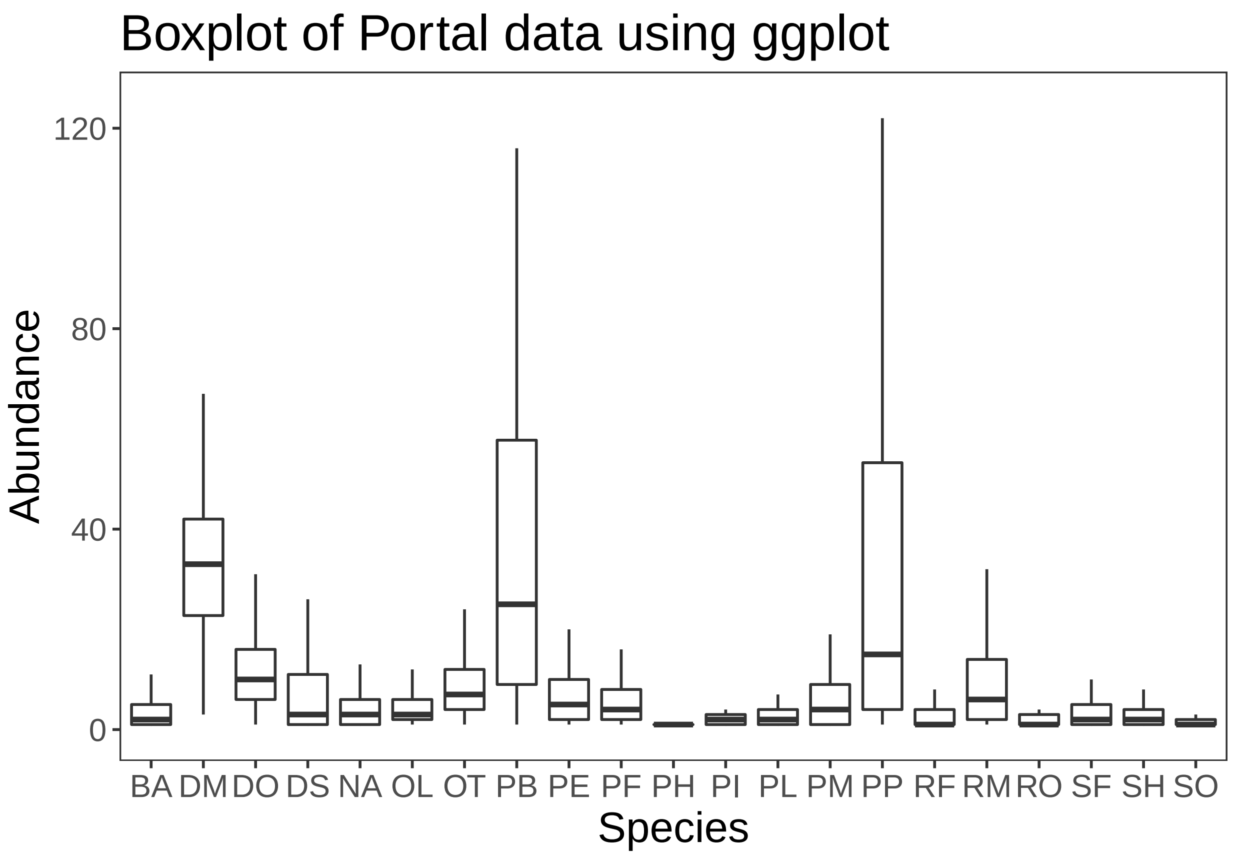
Boxplots using base R

```r
plot(x=rodent_abundance_sub$species, y=rodent_abundance_sub$abundance, # Define the data to
# plot on the x and y axes
     outline = F, # Setting outline = F prevents the outliers from appearing
     ylim = c(0,125), # Set the y-axis range
     xlab = "Species", # Label the x-axis
     ylab = "Abundance", # Label the y-axis
     main = "Boxplot of Portal data using base R") # Give the plot a title
```



Boxplots using ggplot

```r
ggplot(data=rodent_abundance_sub, aes(x=species, y=abundance))+ # Define where the data
  # come from and the aesthetic (x and y values)
  geom_boxplot(outlier.shape = NA)+ # Make the plot a box plot, and remove
  # the outliers by setting their shape to NA
  theme_bw()+ # Get rid of the gray background
  xlab("Species")+ # Label the x-axis
  ylab("Abundance")+ # Label the y-axis
  ggtitle("Boxplot of Portal data using ggplot")+ # Give the plot a title
  coord_cartesian(ylim=c(0,125))+ # Define y-axis range
  theme(axis.title = element_text(size = 16), # Change the text size of
        # the axis labels (sepal length and petal width)
        axis.text = element_text(size = 12), # Change the text size of the numbers on
        # the x and y axis
        title = element_text(size=16), # Change the text size of the title
        panel.grid.major = element_blank(), # Remove the major grid lines in the background
        panel.grid.minor = element_blank()) # Remove the minor grid lines in the background
```

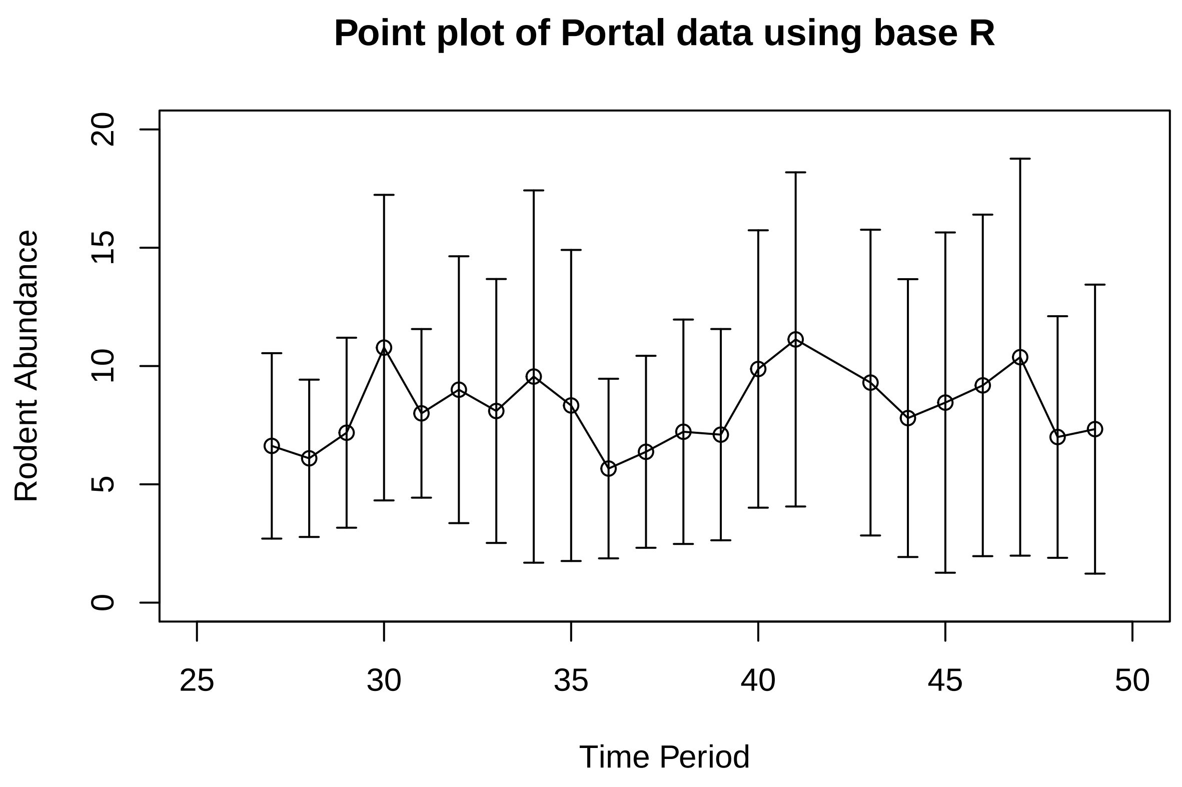


Point estimates with confidence intervals in base R

We continue to use the rodent data, but to make our figures simpler, we will only use data from the first 50 time periods

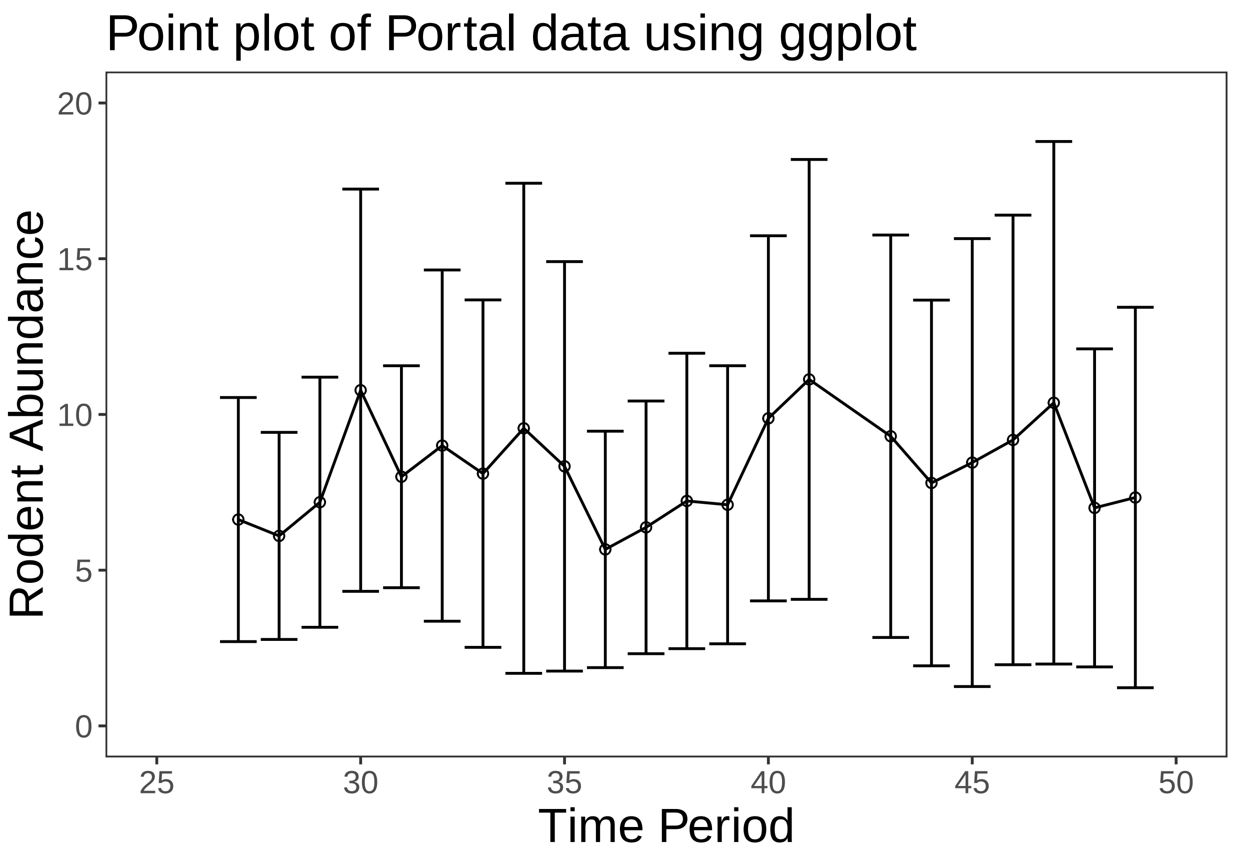
```
 # Only use datafrom first 50 time periods.
rodent_period_sub=subset(rodent_abundance_sub,rodent_abundance_sub$period<50)
# The first time period is 27, so only return 23 time periods

# To plot points and confidence intervals, we need to use the function aggregate to
# perform a function on all rows that have the same time period
rodent_agg=aggregate(rodent_period_sub$abundance, # The column we perform calculations on
                     by=list(rodent_period_sub$period), # TThe column we use to
                     #identify which points are related (i.e., all rows that have
                     #the same value for period will be grouped together for analysis)
                       function(x) c(mean=mean(x), # Define a function that calculates the mean...
                                     se=sd(x)/sqrt(length(x)))) # ...and the standard error
                                     # for all points with a shared period

# Aggregate returns the data in a strange form, so turn it back into a dataframe
# by defining the columns that we want
rodent_means=data.frame(period=rodent_agg$Group.1,
                                abundance=rodent_agg$x[,1],
                                se=rodent_agg$x[,2])
```

```r
# Calculate the 95% CI (spread of the data) according to the formula mean ± 1.96*se
rodent_means$lower_CI=rodent_means$abundance-rodent_means$se*1.96 # Make a column for the
# lower boundary of the 95% CI
rodent_means$upper_CI=rodent_means$abundance+rodent_means$se*1.96 # Make a column for the
# upper boundary of the 95% CI

# Create a blank plot that has all of the attributes of a normal plot (axis names
# and ranges, title, etc.), but does not display the data
plot(x = rodent_means$period,
     y = rodent_means$abundance,
     type = "n", # Make a blank plot so we can add the error bars, lines, and
     # points in the order that we want.
     # Could also set type = "b" to display both points and lines, but
     # this is less flexible to make the figure look how we want
     xlim = c(25, 50), # Set the x-axis range
     ylim = c(0, 20), # Set the y-axis range
     xlab = "Time Period", # Give the x-axis a name
     ylab = "Rodent Abundance", # Give the y-axis a name
     main = "Point plot of Portal data using base R") # Give the plot a name

# R hack: use arrows to plot the error bars (95% CIs that we calculated above)
arrows(rodent_means$period, rodent_means$lower_CI, # The start point of the arrow (x0, y0)
       rodent_means$period, rodent_means$upper_CI, # The end point of the arrow (x1, y1).
       # In this case the starting and ending x are the same because we only want a
       # vertical line (change in y) on each point.
       length=0.05, angle=90, code=3) # Set the length of the arrow head (.05 cm),
       # set the angle of the arrow (90 degrees is a flat line),
       # and the code to 3, which makes an arrowhead on each end of the line

# Because this is a time series, we can show the relationship between points with lines
lines(x = rodent_means$period, # x-values for the lines
      y = rodent_means$abundance, # y-values for the lines
      lwd = 1, # Define the line width
      lty = "solid", # Define the line type
      col = "black") # Define the line color

# Now add the points on top
points(x = rodent_means$period, # x-values for the points
       y = rodent_means$abundance, # y-values for the points
       cex = 1, # Size of the points
       pch = 21, # Make the shape of the points an open circle
       col = "black") # Make the border color of the points black
```



Point plot using ggplot

```
ggplot(data = rodent_means, aes(x=period, y=abundance))+ # Define the data
  # and mapping for the plot
  geom_point(shape=21)+ # Display the data as points and make the shape an open circle
  geom_line()+ # Also display the data as lines
  geom_errorbar(aes(ymin=lower_CI,ymax=upper_CI))+ # Display the error bars by defining
  # the minimum and maximum y-values of our CI.
  # Can also calculate these directly in ggplot.
  theme_bw()+ # Get rid of the gray background
  xlab("Time Period")+ # Label the x-axis
  ylab("Rodent Abundance")+ # Label the y-axis
  ggtitle("Point plot of Portal data using ggplot")+ # Give the plot a title
  coord_cartesian(xlim = c(25,50), ylim=c(0,20))+ # Define x and y-axis range
  theme(axis.title = element_text(size = 18), # Change the size of the text for the axis labels
        axis.text = element_text(size = 12), # Change the text size of the numbers on
        # the x and y axis
        title = element_text(size=16), # Change the text size of the title
        panel.grid.major = element_blank(), # Remove the major grid lines in the background
        panel.grid.minor = element_blank()) # Remove the minor grid lines in the background
```



Plotting multiple windows

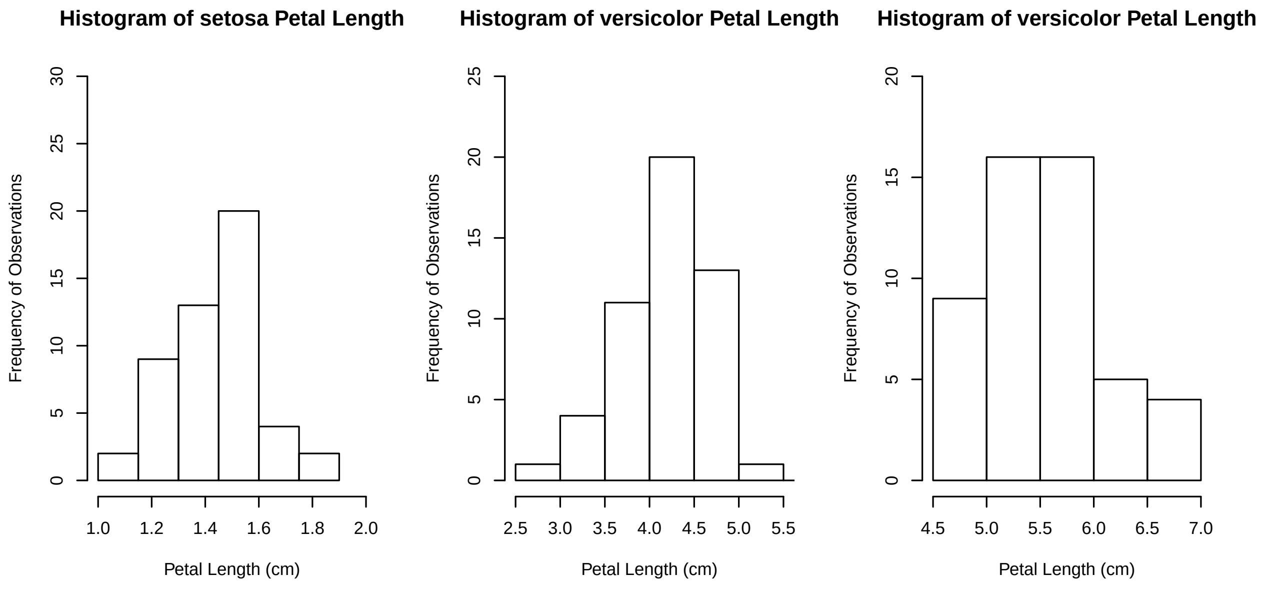
Sometimes you want to display 2 or more figures at once but in separate windows.

Be **VERY** careful when you make multiple plots; it is easy to mislead someone by using different scales for each plot and not clearly declaring these differences. See if you can spot this type of error in the plots below.

Multiple plots in base R

We display multiple plots in base R by changing the graphical parameters using the command par

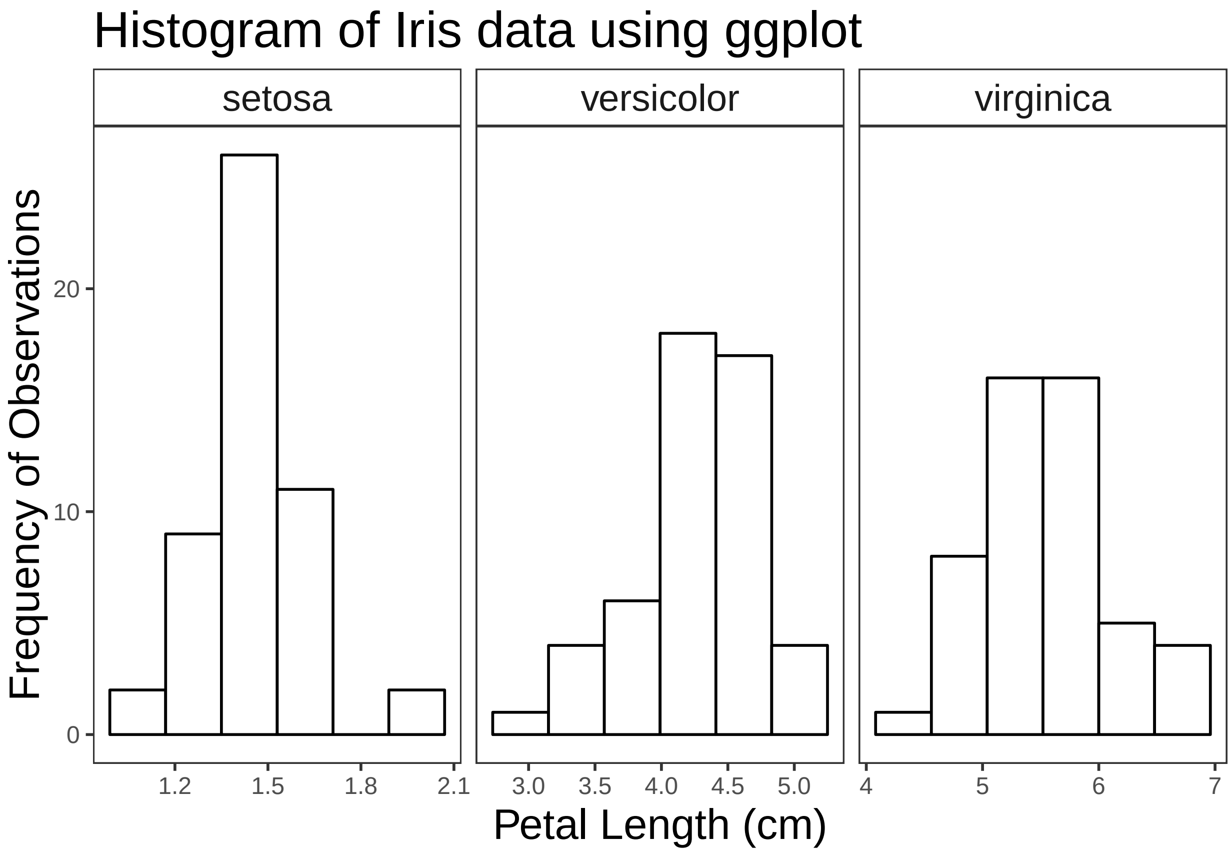
```r
par(mfrow=c(1,3)) # This creates 1 row and 3 columns of plots.
# par(mfrow=c(2,2)) would make 4 plots in 2 rows and 2 columns.
# par(mfrow=c(2,3)) would create 6 plots in 2 rows and 3 columns, etc.

# Iris histograms separated by species

hist(iris[iris$Species=="setosa",]$Petal.Length, # Subset to only setosa
     breaks = seq(1,2, by = 0.15),
     ylim = c(0, 30),
     xlim = c(1,2),
     xlab = "Petal Length (cm)",
     ylab = "Frequency of Observations",
     main = "Histogram of setosa Petal Length")

hist(iris[iris$Species=="versicolor",]$Petal.Length, # Subset to only versicolor
     breaks = seq(2.5,6, by = 0.5),
     ylim = c(0, 25),
     xlim = c(2.5,5.5),
     xlab = "Petal Length (cm)",
     ylab = "Frequency of Observations",
     main = "Histogram of versicolor Petal Length")

hist(iris[iris$Species=="virginica",]$Petal.Length, # Subset to only virginica
     breaks = seq(4.5,7, by = 0.5),
     ylim = c(0, 20),
     xlim = c(4.5,7),
     xlab = "Petal Length (cm)",
     ylab = "Frequency of Observations",
     main = "Histogram of versicolor Petal Length")
```



Multiple plots using ggplot

To make multiple plots using ggplot, we use the function facet_grid, which creates a matrix of grids based on the number of unique values in the defined column

```
ggplot(data=iris,aes(x=Petal.Length))+
  geom_histogram(bins = 6, # Automatically break up the data into 6 bins
                 col="black",
                 fill="white")+
  theme_bw()+
  xlab("Petal Length (cm)")+
  ylab("Frequency of Observations")+
  ggtitle("Histogram of Iris data using ggplot")+
  facet_grid(.~Species, # Make a different facet (window) for unique values in column Species
             scales = "free")+ # This allows ggplot to scale each axis separately.
             # Use scales = "fixed" to make all axes the same.
  theme(title = element_text(size=16),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        strip.background = element_rect(fill = "white"),
        strip.text = element_text(size = 14))
```


Histogram of Iris data using ggplot
setosa
versicolor
virginica
Frequency of Observations
Petal Length (cm)