TOPIC 5
PLOTTING IN R
PART 2

Outline

- Last time, we covered background information on basic plotting in R.
  - Low vs. High Level plotting
  - Scatter/line plots and histograms
  - Change plotting parameters

- Today, we will cover commands that will allow us to refine the basic plotting skills we covered last time.
  - Changing plotting character, color, and sizes based on another variable
  - Adding information to the plotting window of a figure
  - Adding reference lines to a figure
  - Changing the labels and position of axis ticks marks
  - Adding overall title, labels, and legend for a figure with multiple plots

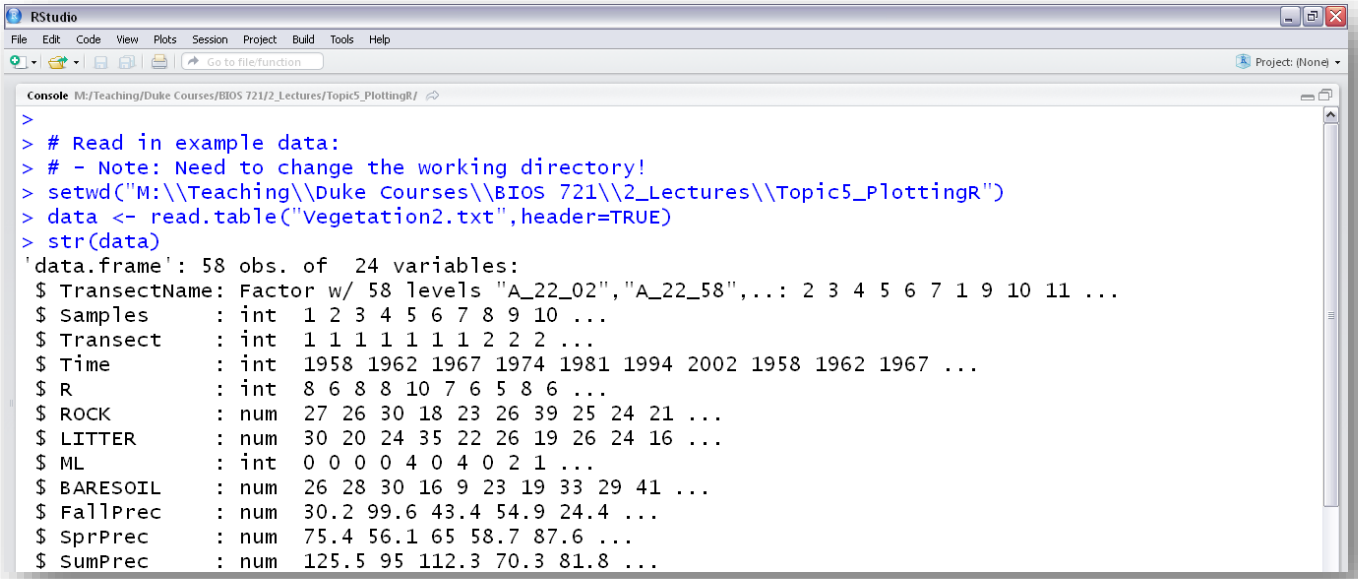
Plotting in R

- Today we will be working with the 'Vegetation' data set in the file Vegetation2.txt.
- Sikkink et al. (2007) analyzed grassland data from a monitoring program conducted in two temperate communities, Yellowstone National Park and National Bison Range, USA. The aim of the study was to determine whether biodiversity of these bunchgrass communities changed over time, and if so, whether the changes in biodiversity were related to particular environmental factors.
  - Biodiversity was quantified as species richness, defined as the number of different species per site.
  - Data was measured in 8 transects(a path along which a biological phenomenon is observed and recorded), with each being measured in 4 to 10 year intervals.
  - We will also work with the variables time (year when the observations were made) and soil exposure (proportion of bare soil along the transect).

Plotting in R

- Code: Read in data and then check structure

```
setwd("M:\\R Seminar")
data <- read.table("Vegetation 2.txt",header=TRUE)
str(data)
```



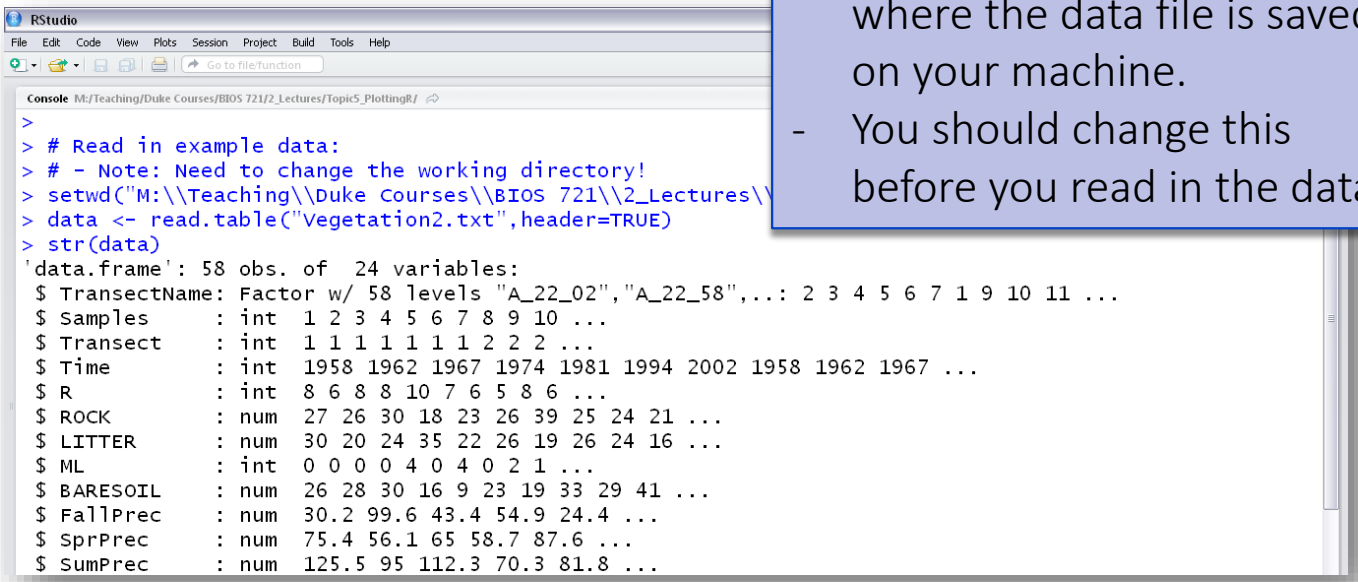
Plotting in R

- Code: Read in data and then check structure

```
setwd("M:\\R Seminar")
data <- read.table("Vegetation 2.txt",h
str(data)
```

The setwd() function "sets the working directory" in R.

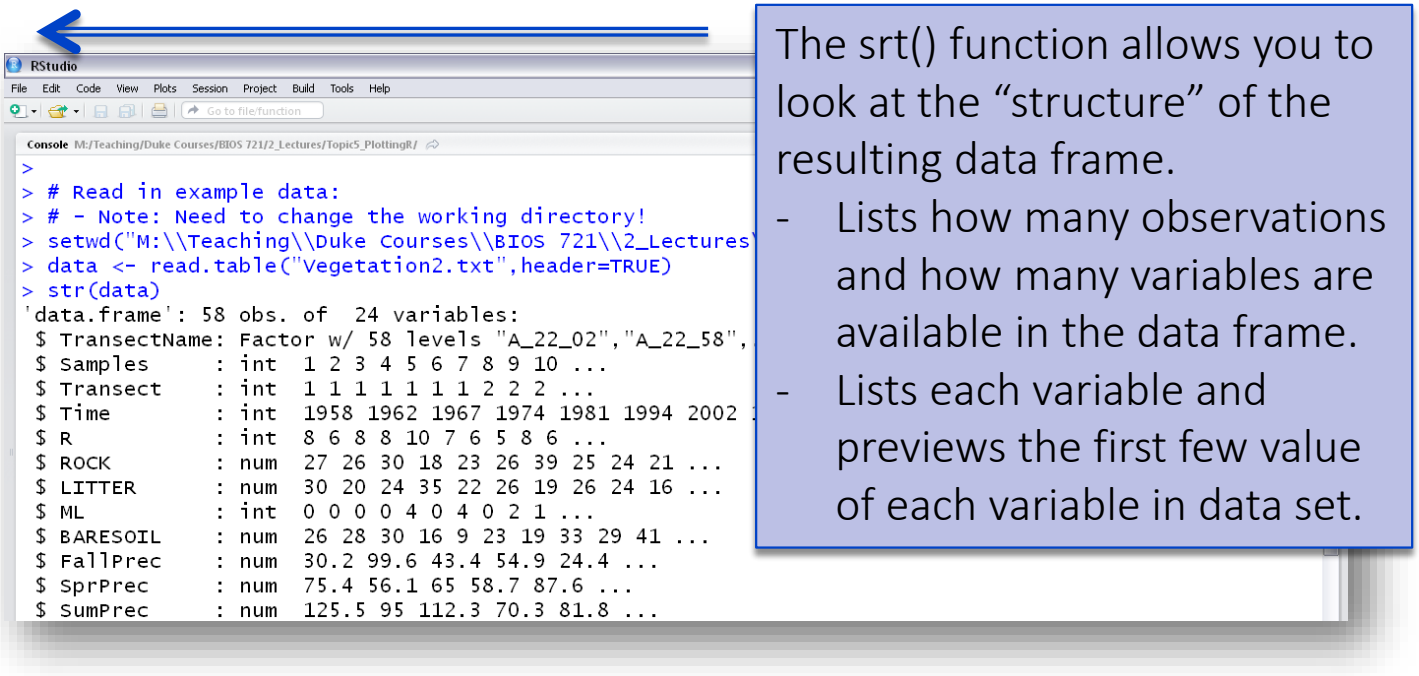
- Allows you to point R to where the data file is saved on your machine.
- You should change this before you read in the data.



Plotting in R

- Code: Read in data and then check structure

```
setwd("M:\\R Seminar")
data <- read.table("Vegetation 2.txt",header=TRUE)
str(data)
```



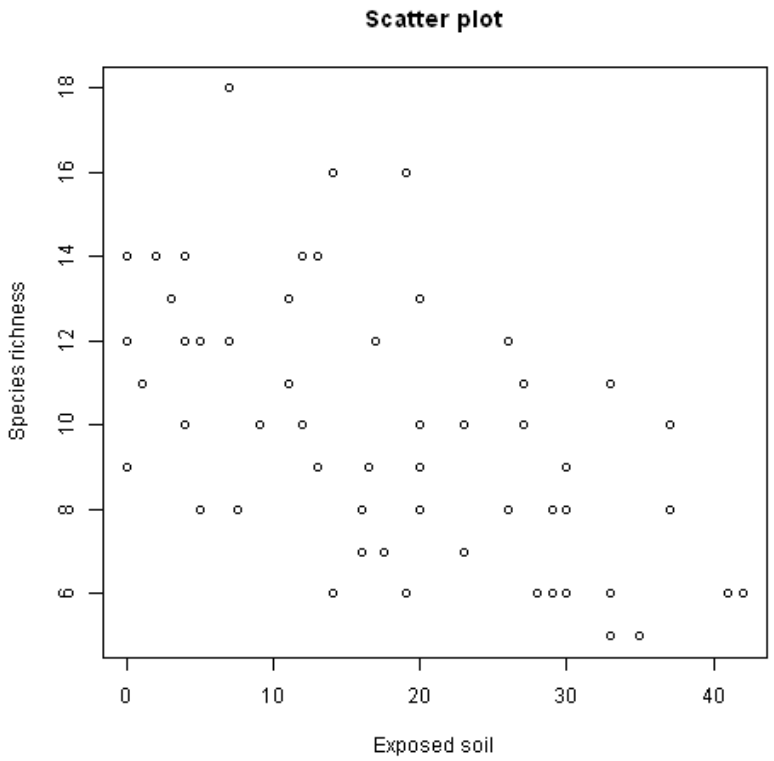
The srt() function allows you to look at the "structure" of the resulting data frame.

- Lists how many observations and how many variables are available in the data frame.
- Lists each variable and previews the first few value of each variable in data set.

Plotting in R

- Suppose we wanted to examine the relationship between the Species Richness and Soil Exposure:

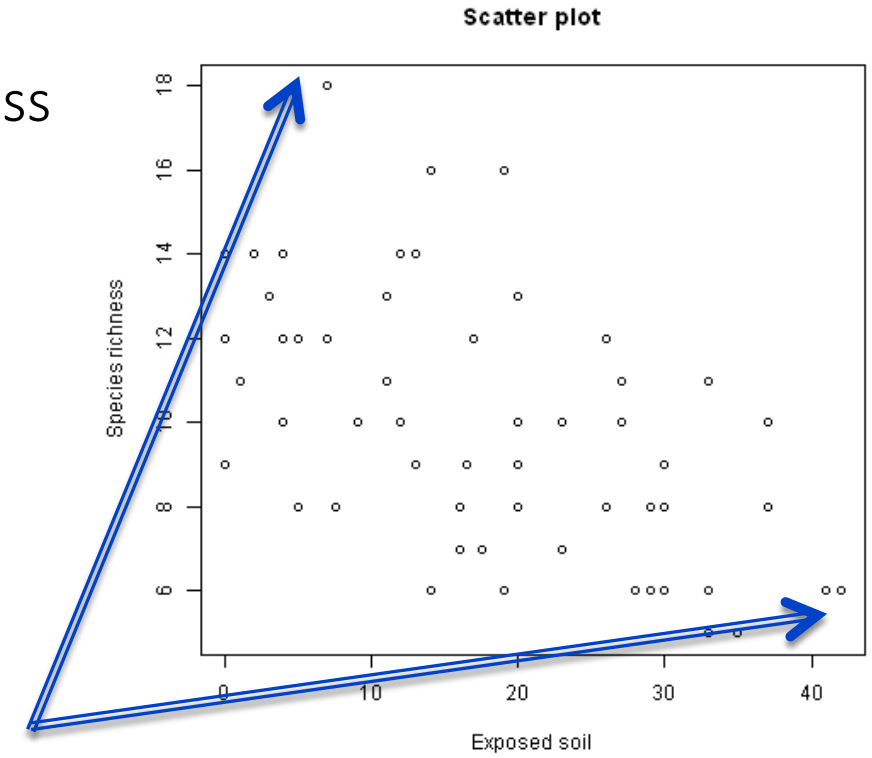
```
# Standard Scatter Plot
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot")
```



Plotting in R

- Suppose we wanted to examine the relationship between the Species Richness and Soil Exposure:

```
# Standard Scatter Plot
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot")
```

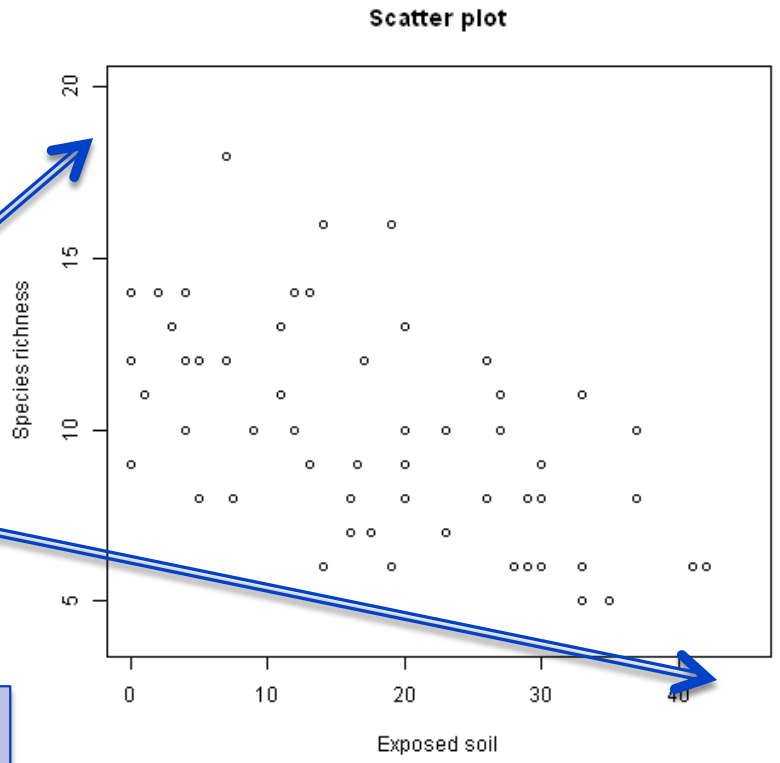


The plotting window is a little tight – let's give it a little wiggle room.

Plotting in R

- Wiggle Room:

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45))
```

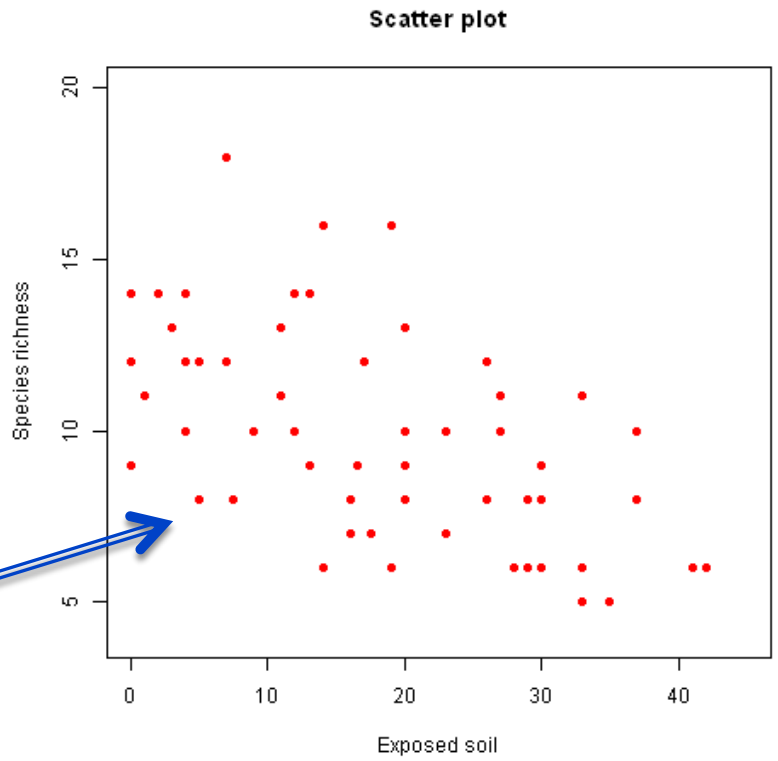


Use the ylim and xlim options to increase the range of the x and y axis.

Plotting in R

- Review: Change plotting character to red, shaded dot:

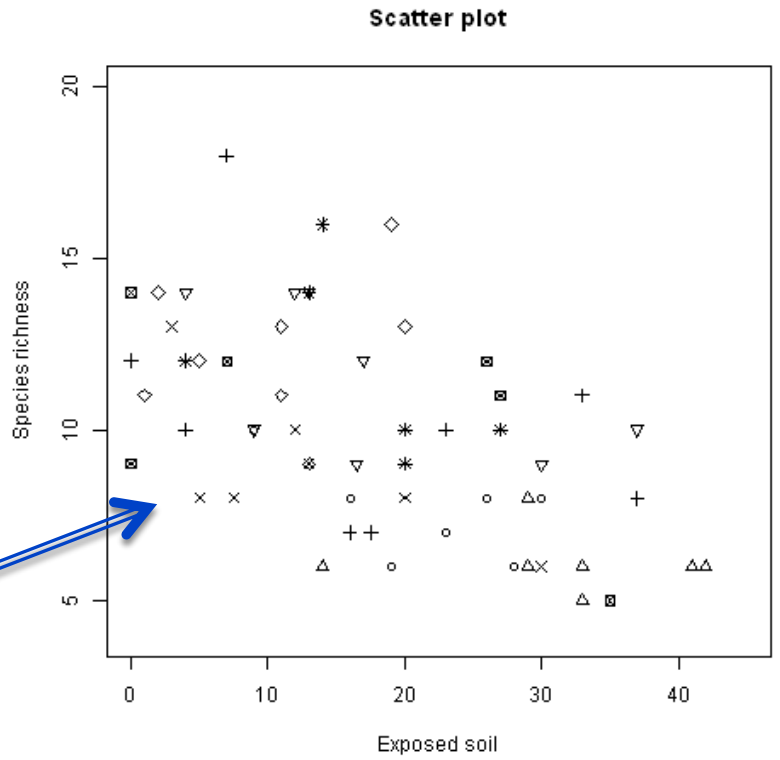
```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=16,col= "red")
```



Plotting in R

- Change plotting character by the transect variable:

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=data$Transect)
```

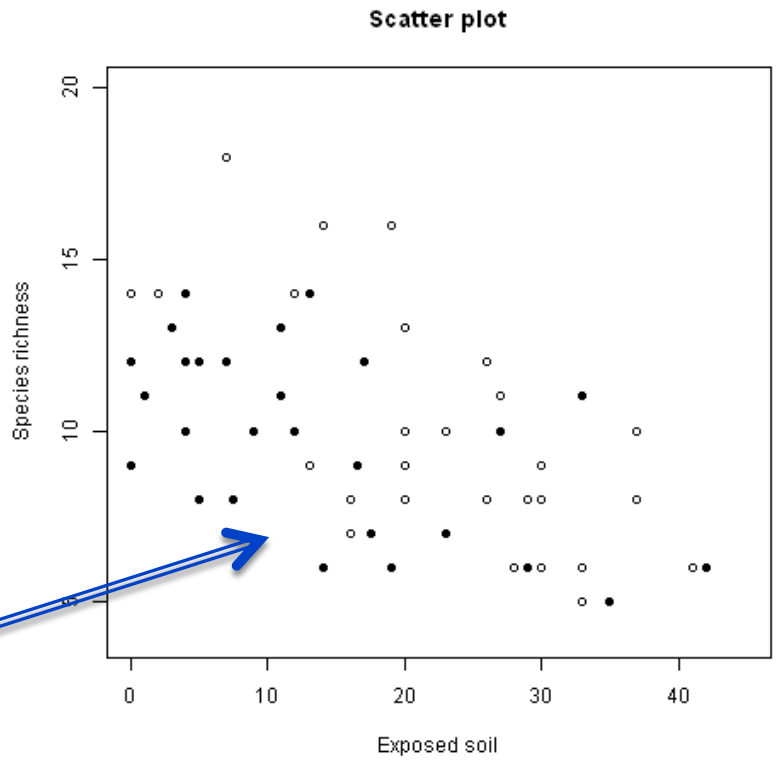


Data$Transect is numeric variable, ranging from 1 to 8, and denotes the transect ID for each observation. Can use pch = 1:8 to denote each transect with a different plotting character.

Plotting in R

- Change plotting character by Time observation measured:

```
pch.time <- rep(1,dim(data)[1])
pch.time[data$Time > 1974] <- 16

plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=pch.time)
```



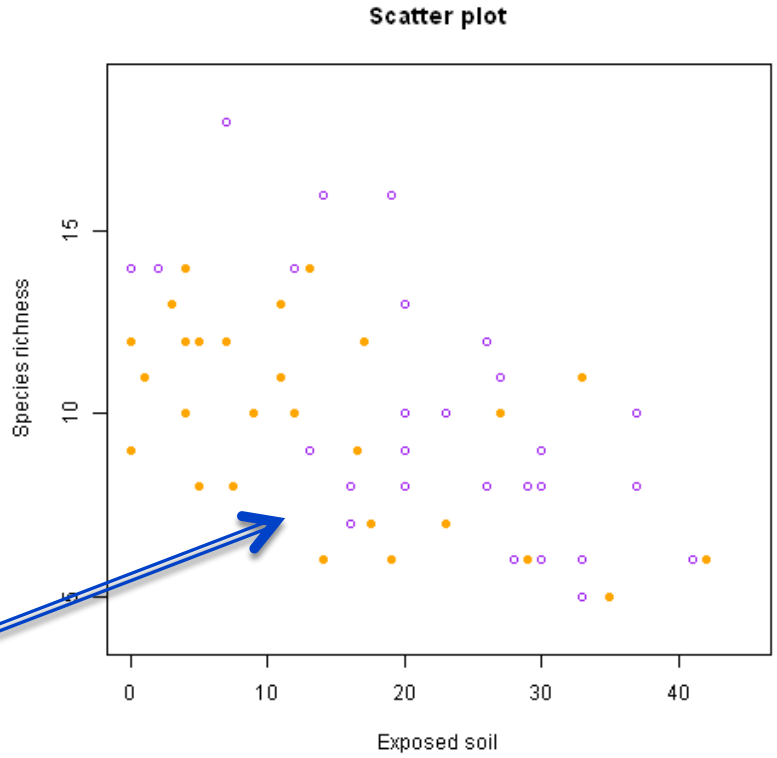
Open dot for Time ≤ 1974 and Shaded dot for Time > 1974

Note: Have to create vector of plotting characters because it does NOT already exist in data set.

Plotting in R

- Change plotting character and color by Time observation measured:

```
pch.time <- rep(1,dim(data)[1])
pch.time[data$Time > 1974] <- 16

col.time <- rep('purple',dim(data)[1])
col.time[data$Time > 1974] <- 'orange'

plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=pch.time,col=col.time)
```



Open Purple dot for Time ≤ 1974 and Shaded Orange dot for Time > 1974

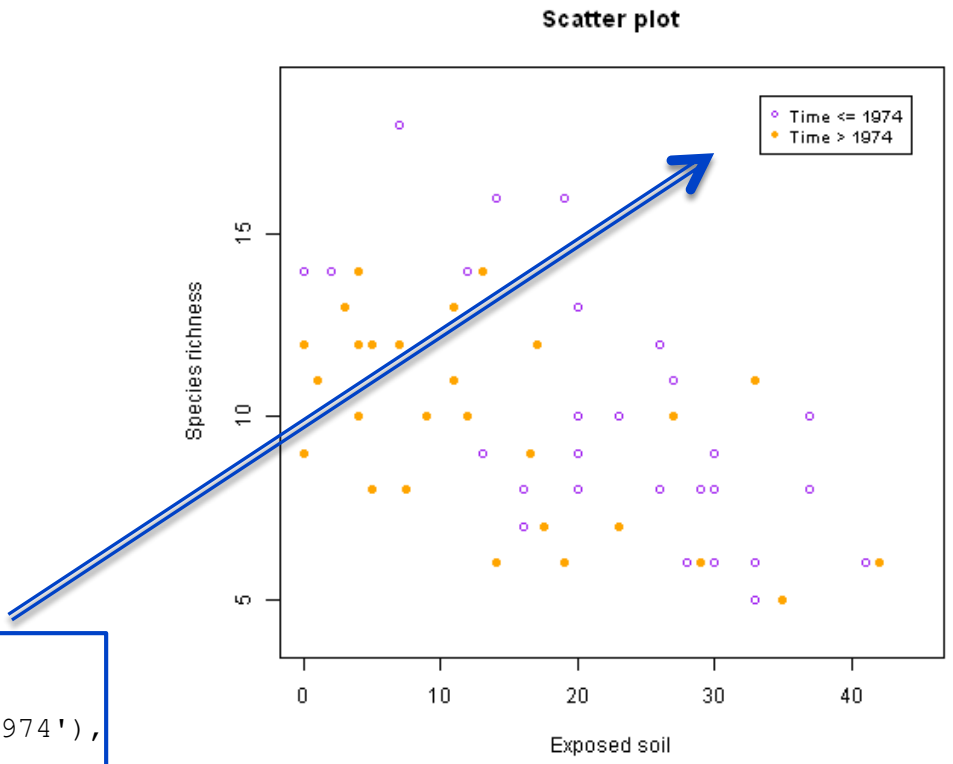
Note: Have to create vector of plotting characters/colors because it does NOT already exist in data set.

Plotting in R – You Try It!

- Probably good to add a legend:

```
pch.time <- rep(1,dim(data)[1])
pch.time[data$Time > 1974] <- 16
col.time <- rep('purple',dim(data)[1])
col.time[data$Time > 1974] <- 'orange'

plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=pch.time,col=col.time)
```

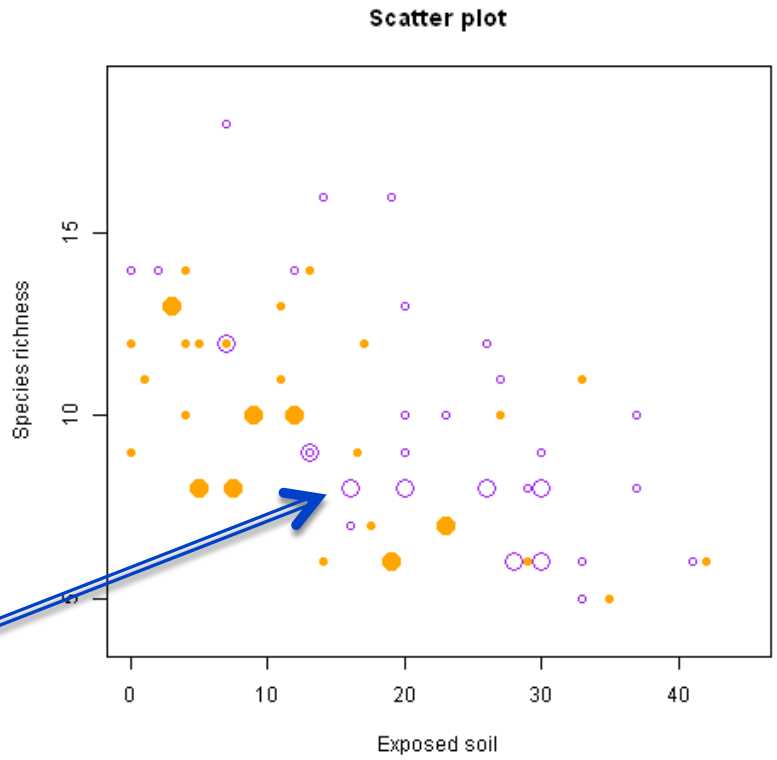
```
legend('topright',inset=0.05,cex=0.8,
       legend=c('Time <= 1974','Time > 1974'),
       pch=c(1,16),col=c('purple','orange'))
```



Plotting in R – You Try It!

- Change size of plotting character by Transect (1 and 4 are of interest):

```
pch.time <- rep(1,dim(data)[1])
pch.time[data$Time > 1974] <- 16

col.time <- rep('purple',dim(data)[1])
col.time[data$Time > 1974] <- 'orange'

cex.tran <- rep(1,dim(data)[1])
cex.tran[data$Transect==1 | data$Transect==4] <- 2

plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),
     pch=pch.time,col=col.time,cex=cex.tran)
```

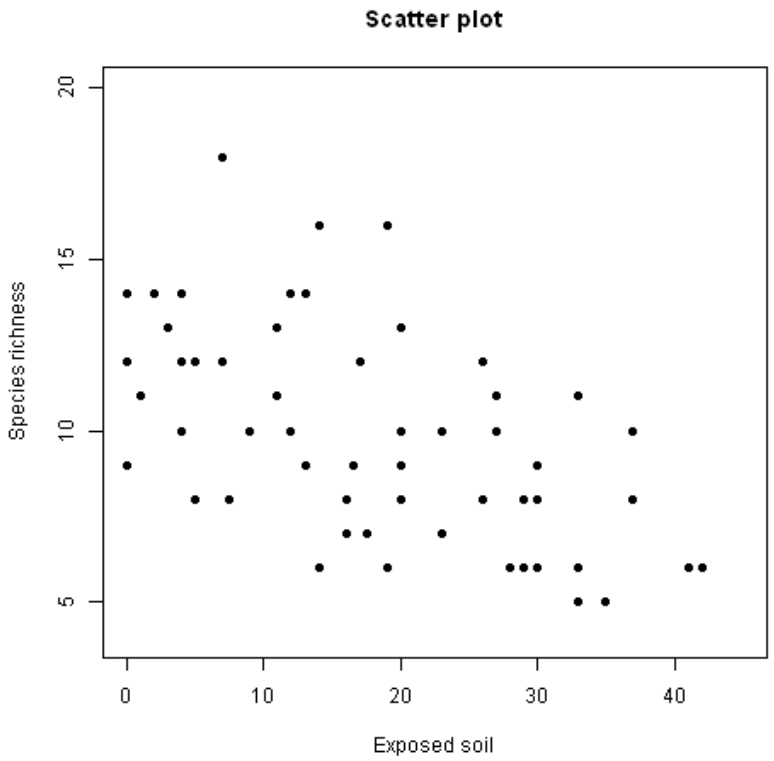


Open large dot for Transect 1 or 4 and Small dot for Transect ≠ 1 or 4

Note: Have to create vector of plotting characters sizes because it does NOT already exist in data set.

Plotting in R

- There appears to be a negative linear trend between species richness and soil exposure.
- Suppose we wanted to fit a linear regression model and the add the fitted regression line to the plot.
  - We can easily add statistical information to plots in R using R functions for fitting models and then using the lines() function.

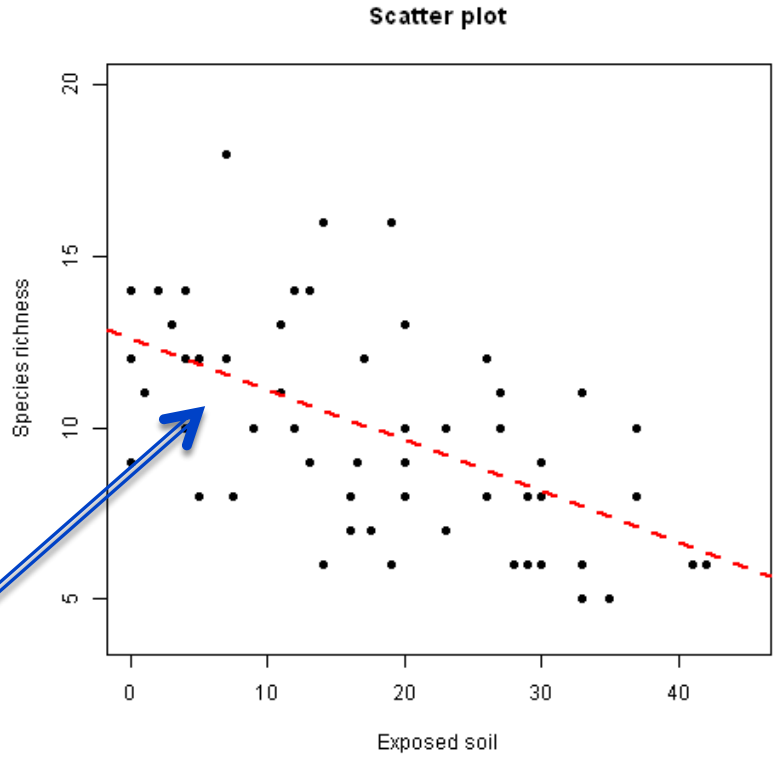


Plotting in R

- Add linear regression line to the scatter plot:

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16)
```

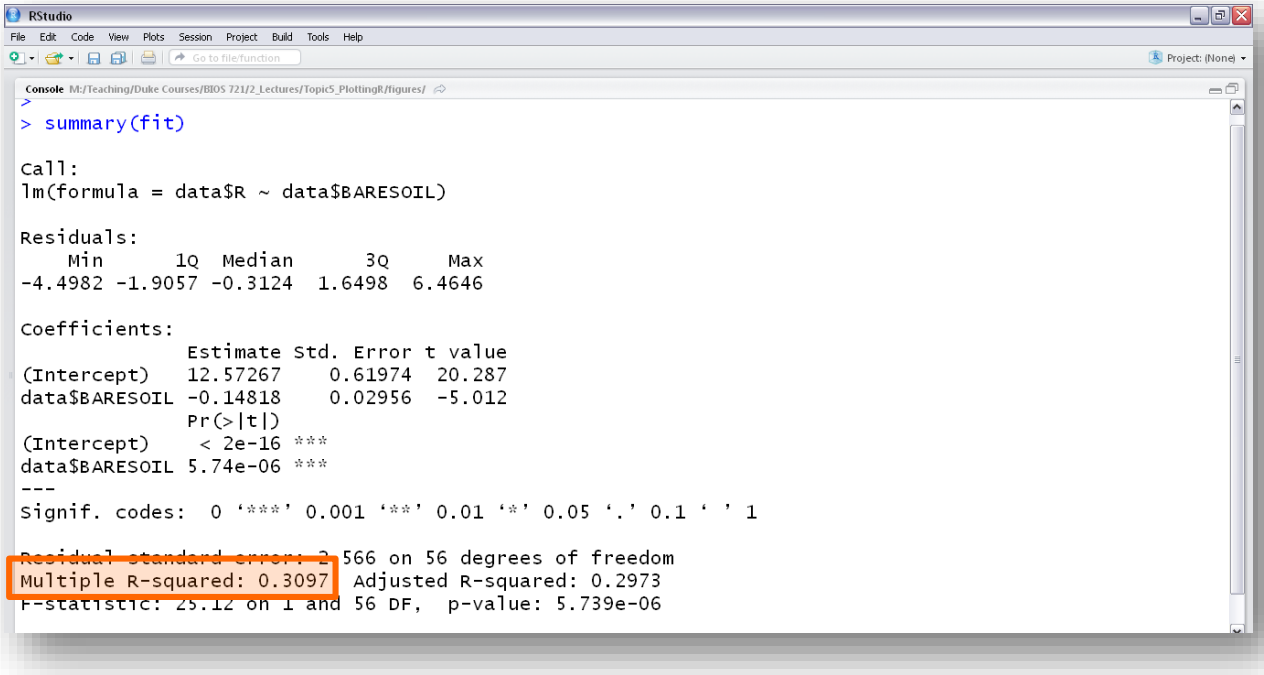
```
fit <- lm(data$R ~ data$BARESOIL)
abline(fit,lty=2,lwd=2,col='red')
```



- Use the lm() function to fit the linear regression between the two variables.
- Then use the abline() function to add the fitted line to the scatter plot.

Plotting in R

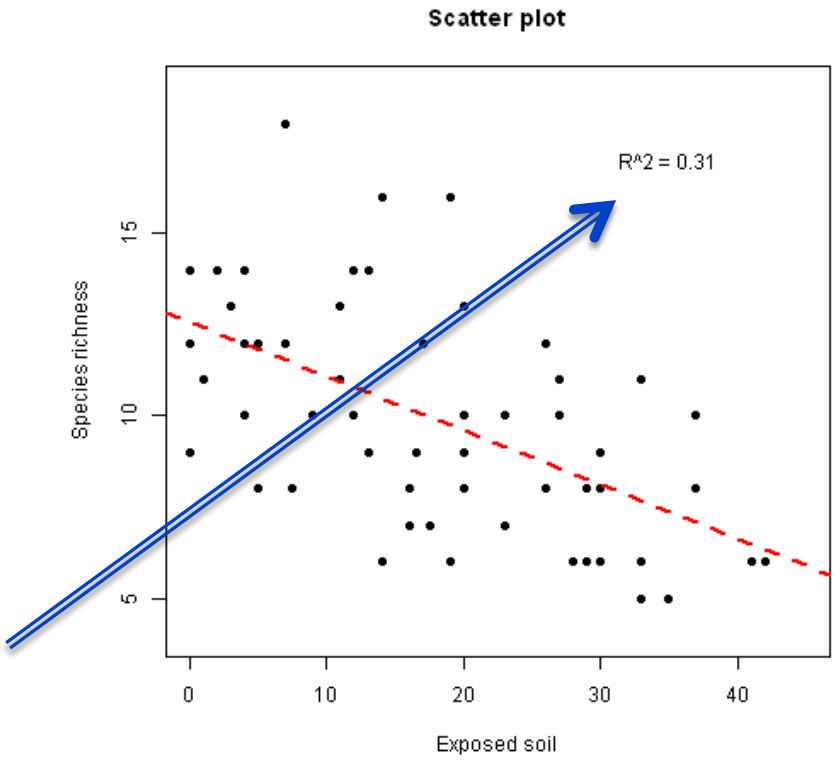
- This plots look great, but often we would like to add more statistical summaries to these types of plots.
  - For example, researchers often report the correlation between two variables in scatter plots OR the R^2 value associated with the fitted linear regression model.
- We can easily add this info to the plot.
  - To determine the R^2 value, use the summary() function in the lm() fit object.



Plotting in R

- Add R^2 value to plot:

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16)

fit <- lm(data$R ~ data$BARESOIL)
abline(fit,lty=2,lwd=2,col='red')

text(x=35,y=17,labels='R^2 = 0.31')
```



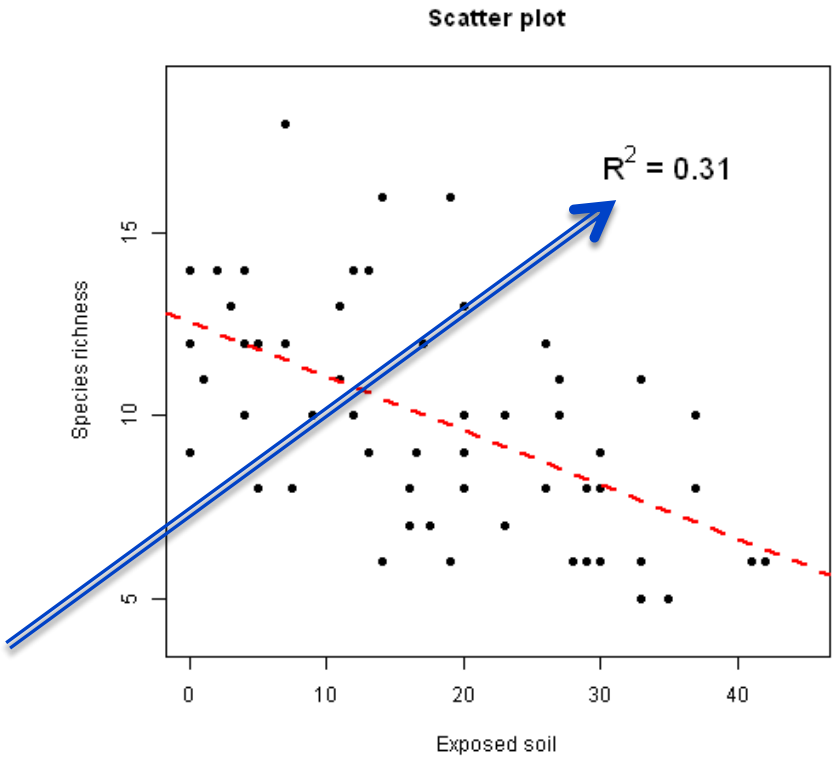
Use the text() function to add text to the plotting window.
- Give the x and y coordinates of text, and character string that you want to add.

Plotting in R

- Can make this look better – bigger and use legit math symbols:

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16)

fit <- lm(data$R ~ data$BARESOIL)
abline(fit,lty=2,lwd=2,col='red')
```

```
text(x=35,y=17,font=2,cex=1.5,
     labels=expression(paste(R^{2},' =
     0.31',sep='')))
```



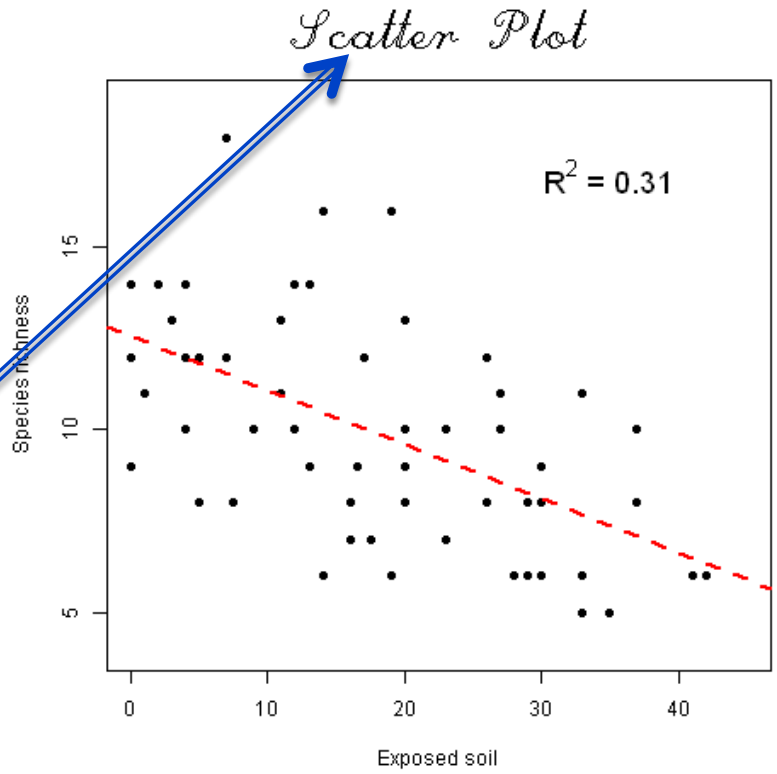
Use the expression() function to present "correct" mathematical symbols/notation. Use cex and font options to make text bigger and bold/italic.

Plotting in R

- Can make change the font and size of the plot title/labels:

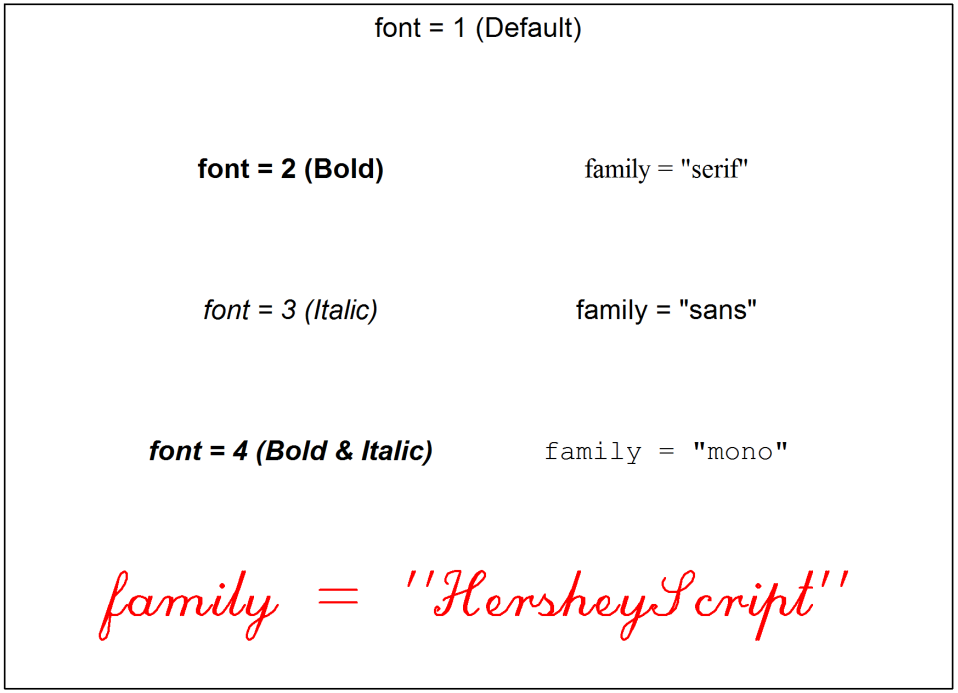
```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16)

fit <- lm(data$R ~ data$BARESOIL)
abline(fit,lty=2,lwd=2,col='red')

text(x=35,y=17,font=2,cex=1.5,
     labels=expression(paste(R^{2},' =
     0.31',sep='')))
```

```
title(main='Scatter Plot',
      cex.main=3,
      family='HersheyScript')
```



To make alterations to the plot title/labels, usually easiest to leave them "missing" in the main plotting function and then use a low-level plotting command to make changes.

Plotting in R

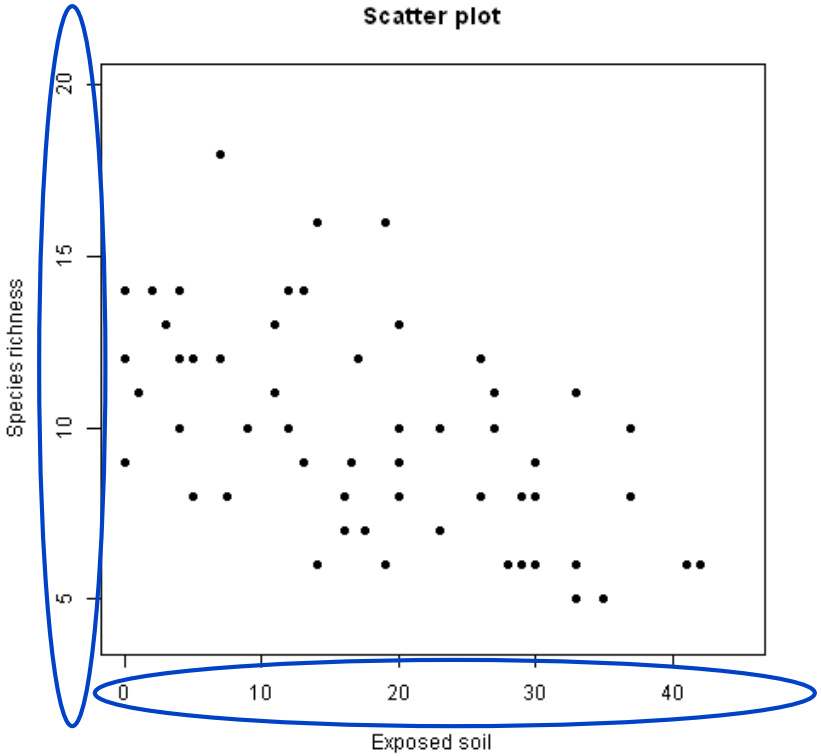


Available Fonts

- Using the text options in text() and title() functions you can change the font size, color, style, and script using the cex, col, font, and family options, respectively.
- Can mix all elements together to create highly customized and publication ready figures!

Plotting in R

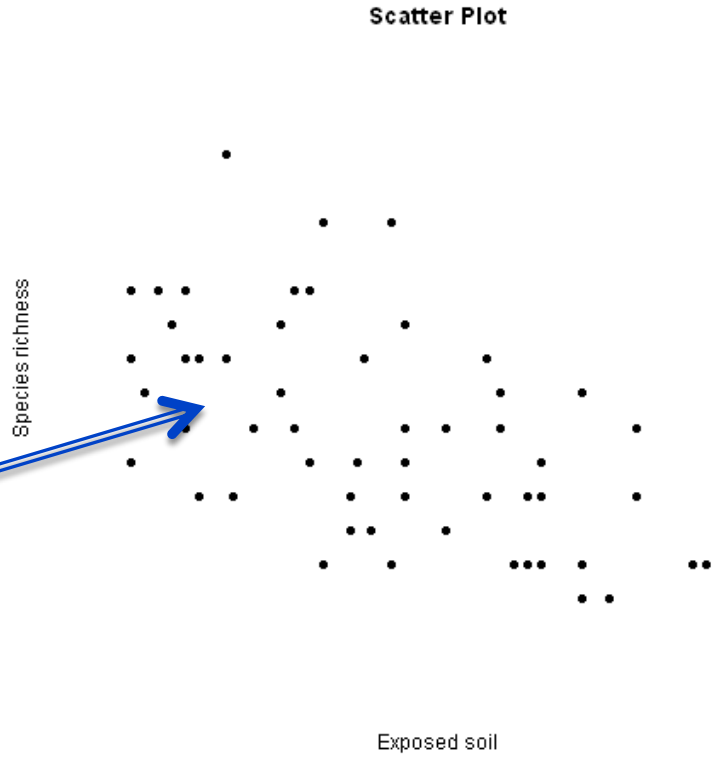
- Suppose we did not like the location or label of the axes tick marks.
- We can change these.
- However, the process is a little more involved.
  - Need to first turn 'off' axes in high level plotting function.
  - Then re-build them with the tick mark locations and labels that you want.
  - Can replace numeric labels with character labels!



Plotting in R

- Step 1: Turn 'off' axes

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16,
     axes=FALSE)
```

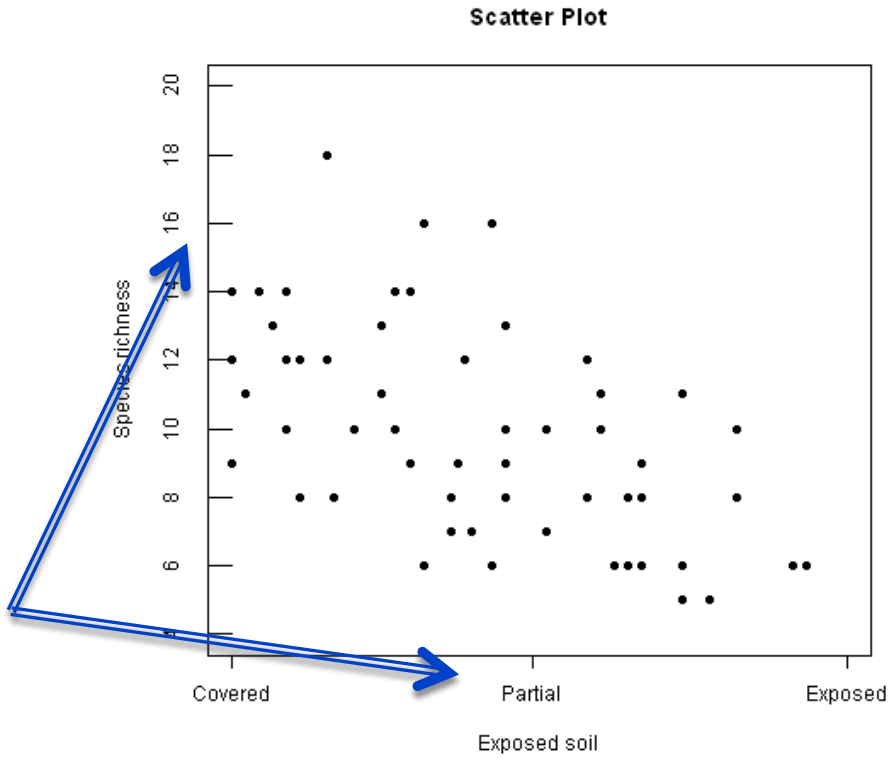


Note: The numeric axes are still "there", R is just NOT DISPLAYING them!

Plotting in R

- Step 2: Re-build axes using the axis() function.

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16,
     axes=FALSE)
axis(2, at = seq(4,20,by=2, tcl = 1)
axis(1, at = c(0,22,45),
     labels = c("Covered",
                "Partial","Exposed"))
box()
```



Note: The tcl option allows you to change the direction and length of the tick marks.

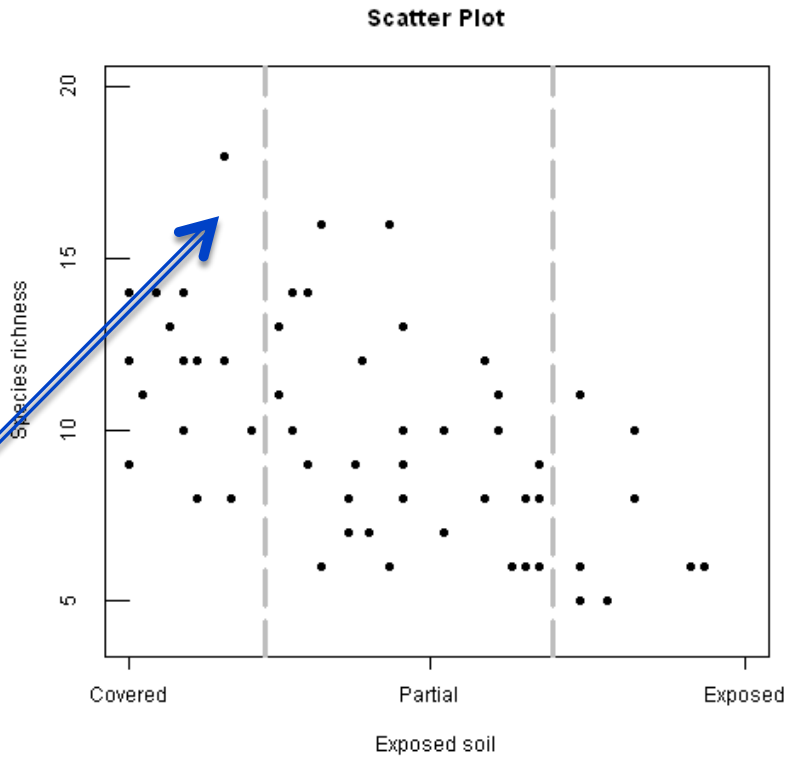
- The default is -0.5 as seen on x-axis; y-axis marks face inward and are twice as long.
- The default is to use labels given for at option (y-axis); but can overwrite using label option (y-axis).

Plotting in R

- Maybe useful to add vertical (or horizontal) reference lines)

```
plot(x = data$BARESOIL,
     y = data$R,
     xlab = "Exposed soil",
     ylab = "Species richness",
     main = "Scatter plot",
     ylim = c(4, 20),
     xlim = c(0, 45),pch=16, axes=FALSE)

axis(2, at = seq(4,20,by=2, tcl = 1)
axis(1, at = c(0,22,45),
     labels = c("Covered",
                "Partial","Exposed"))
box()
```

```
abline(v=10,lty=5,col='gray',lwd=3)
abline(v=31,lty=5,col='gray',lwd=3)
```

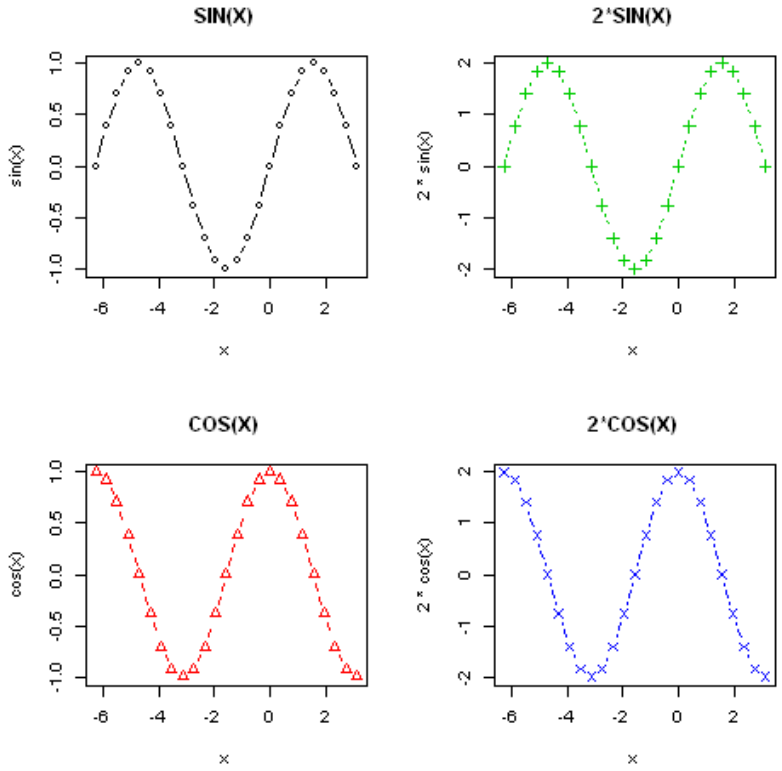


To create horizontal reference lines, use the h option in the abline() function.

Plotting in R

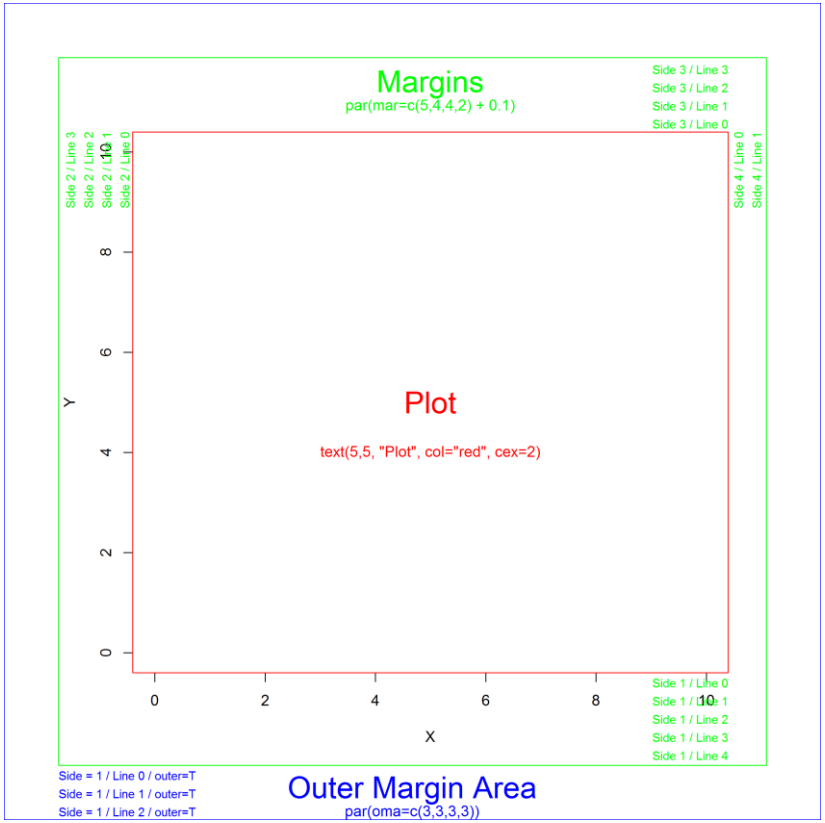
- When plotting multiple figures in the same window, may want to …
  - Add overall title
  - Add over axes labels
  - Add overall legend

- Can do this in R, but requires that you place text in the 'Outer Margin Area' (OMA)
  - It is another par() option
  - Like mfcol and mfrow
  - Changes the global plotting parameters in R session



Plotting in R

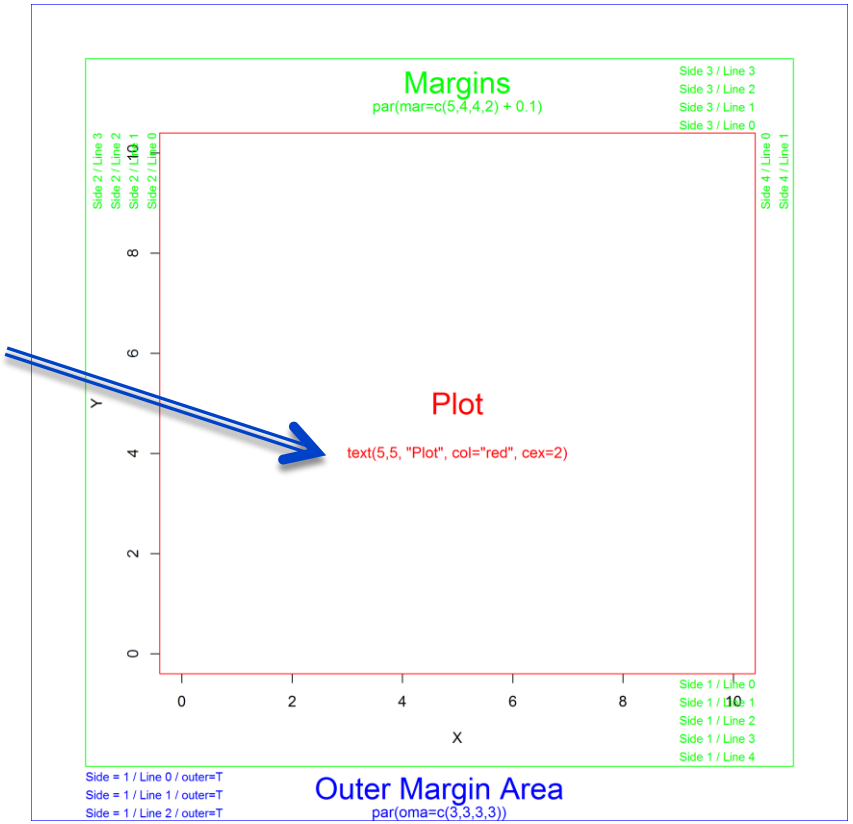
- R figures have 3 working plotting areas:
  - The plotting window
    - Add text using text() function
  - The potting margins
    - Add text using mtext() function
  - The outer plotting margins
    - Add text using mtext() function with outer=TRUE option
- Same rules apply for figures with multiple plots



Plotting in R

- R figures have 3 working plotting areas:
  - The plotting window
    - Add text using text() function

To add text in the plotting window, use the text() function. It is a low-level command.
- Specify the x and y coordinate of the text to add to the plot
- Specify the text to add to the plot
- Can change the size, color, and font

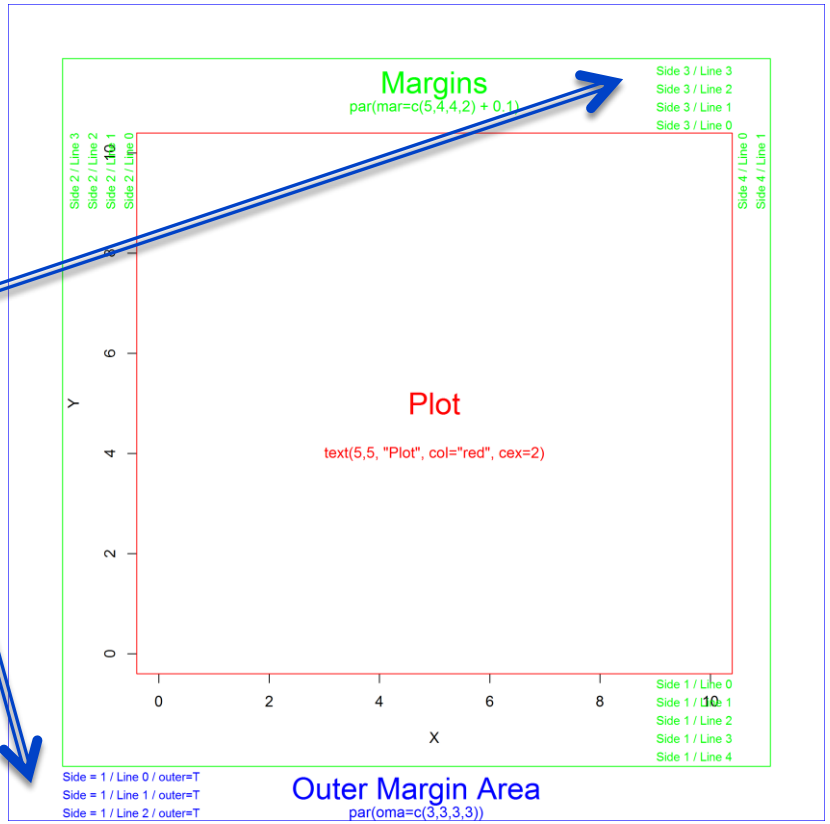


- Same rules apply for figures with multiple plots.

Plotting in R

To add text in the plotting margins use the mtext() function. It is a low-level command.

- Specify the side (bottom/left/top/right)
- Specify the line (depends on the mar and oma options)
- Specify the text to add
- Can specify the adjustment



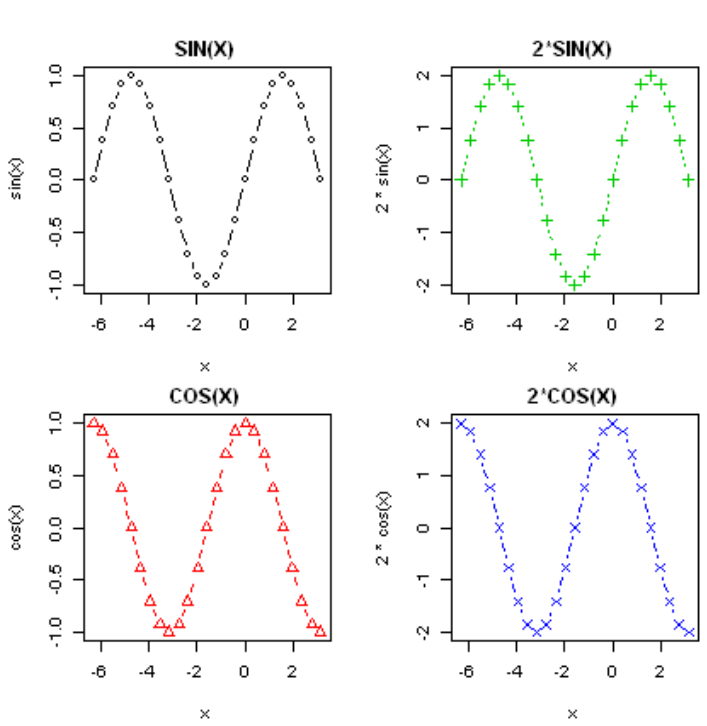
- R figures have 3 working plotting areas:
  - The potting margins
    - Add text using mtext() function
  - The outer plotting margins
    - Add test using mtext() function with outer=TRUE option

Plotting in R

- Set margin and outer margin area sizes so that we have room for titles and labels:
  - Default number of lines per margin area are shown on previous slide! Increase number lines from default for more space.

```
par(mfcol=c(2,2),
    mar=c(4,4,2,2),
    oma=c(3,3,3,1))
```

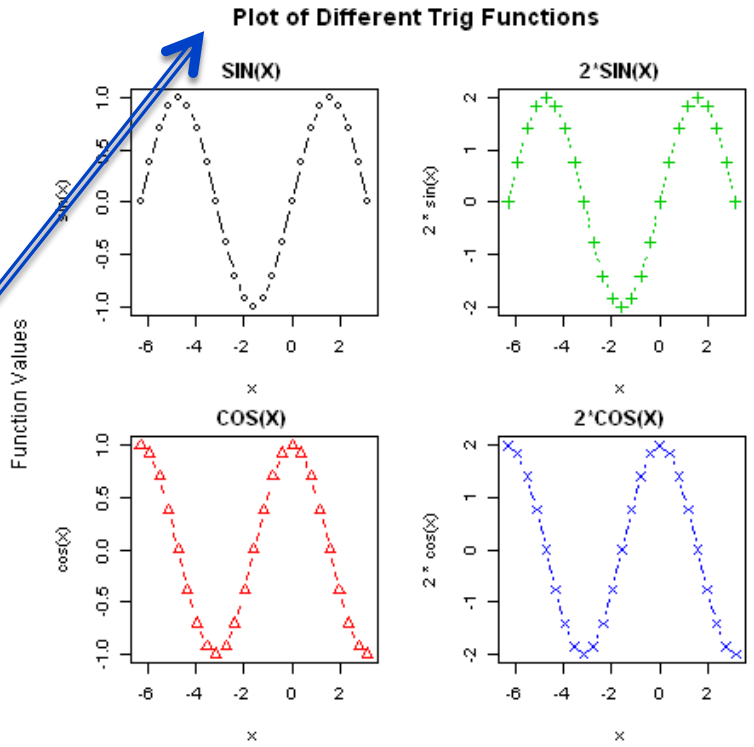
```
x <- seq(-2*pi,pi,by=pi/8)
plot(x,sin(x),type="b",main="SIN(X)",
     pch=1,col=1,lty=1)
plot(x,cos(x),type="b",main="COS(X)",
     pch=2,col=2,lty=2)
plot(x,2*sin(x),type="b",main="2*SIN(X)",
     pch=3,col=3,lty=3)
plot(x,2*cos(x),type="b",main="2*COS(X)",
      pch=4,col=4,lty=4)
```



Plotting in R

- Add overall title and y-axis label using the mtext() function.

```
par(mfcol=c(2,2),
    mar=c(4,4,2,2),
    oma=c(3,3,3,1))

x <- seq(-2*pi,pi,by=pi/8)
plot(x,sin(x),type="b",main="SIN(X)",
     pch=1,col=1,lty=1)
plot(x,cos(x),type="b",main="COS(X)",
     pch=2,col=2,lty=2)
plot(x,2*sin(x),type="b",main="2*SIN(X)",
     pch=3,col=3,lty=3)
plot(x,2*cos(x),type="b",main="2*COS(X)",
      pch=4,col=4,lty=4)

mtext('Plot of Different Trig Functions',
      side=3,outer=TRUE,line=1,cex=1.2,font=2)
mtext('Function Values',
      side=2,outer=TRUE,line=1)
```

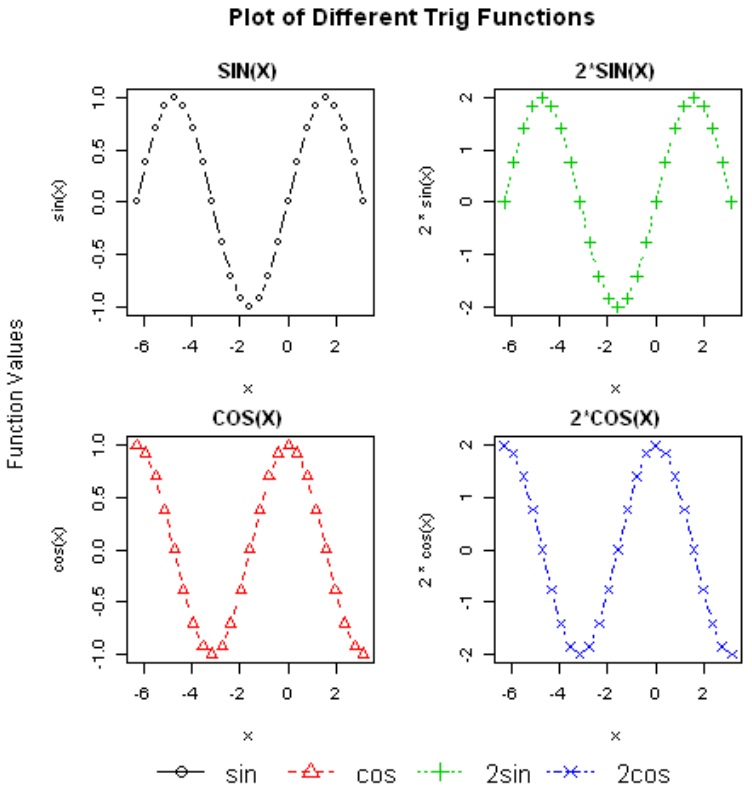


Plotting in R

- Add overall title and y-axis label using the mtext() function.

```
par(mfcol=c(2,2),
    mar=c(4,4,2,2),
    oma=c(3,3,3,1))

x <- seq(-2*pi,pi,by=pi/8)
plot(x,sin(x),type="b",main="SIN(X)",
     pch=1,col=1,lty=1)
plot(x,cos(x),type="b",main="COS(X)",
     pch=2,col=2,lty=2)
plot(x,2*sin(x),type="b",main="2*SIN(X)",
     pch=3,col=3,lty=3)
plot(x,2*cos(x),type="b",main="2*COS(X)",
      pch=4,col=4,lty=4)

mtext('Plot of Different Trig Functions',
      side=3,outer=TRUE,line=1,cex=1.2,font=2)
mtext('Function Values',
      side=2,outer=TRUE,line=1)
```

```
par(usr=c(0,1,0,1),   # Reset the coordinates
    xpd=NA)           # Allow plotting outside the plot region
legend(-1.4,-0.35,legend=c('sin','cos','2sin','2cos'),
       pch=1:4,lty=1:4,col=1:4,bty='n',horiz=TRUE,cex=1.5)
```



Plotting in R

- We have just covered the very basics of plotting in R.
  - We have only covered three high level plotting commands and several low level plotting commands.
  - However, these few commands are very flexible and can be used to make many useful plots.
    - They are the good foundation for making R graphics.

- Can do much, much more using R graphics!
  - `ggplot2()` and `lattice()` are packages that are very popular for making publication ready graphics.
    - These packages attempt to make it easier to make more informative plots.
    - However, there is a HUGE learning curve associated with these packages!
  - References:
    - *ggplot2: Elegant Graphics for Data Analysis* by Hadley Wickham
      - http://www.amazon.com/ggplot2-Elegant-Graphics-Data-Analysis/dp/0387981403
    - Lattice: Multivariate Data Visualization with R by Deepayan Sarkar
      - http://www.amazon.com/Lattice-Multivariate-Data-Visualization-Use/dp/0387759689