R: Graphics

This handout will provide an introduction to creating graphics in R. Here, we will cover only the more basic, traditional graphics. You should be aware, however, that more advanced users can create extremely complex and interesting graphical summaries of data using R.

First, to see some examples of graphs that can be created in R, enter the following command at the prompt.

> demo(graphics)

Next, we will discuss the construction of some basic graphs in R.

**HISTOGRAMS AND DENSITY SMOOTHERS**

Read the NutritionData.txt file into R. Once this data set has been attached, the names in this data frame are as follows.

> names(NutritionData)

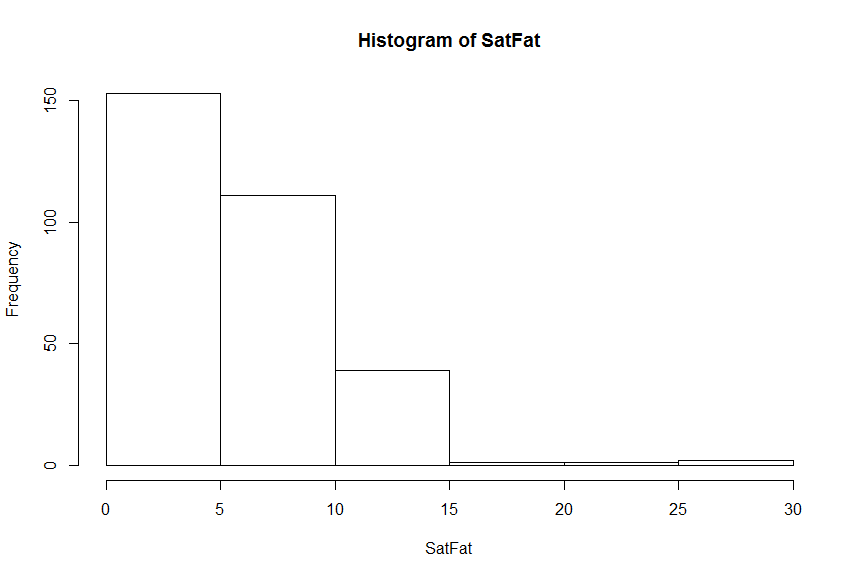
[1] "Location" "ItemName" "Type" "Calories" "TotalFat" "SatFat" "Cholesterol"

[8] "Sodium" "Carbohydrates" "Fiber"

**Creating a Histogram in R**

The most basic form of the hist() function is employed below.

> hist(SatFat)



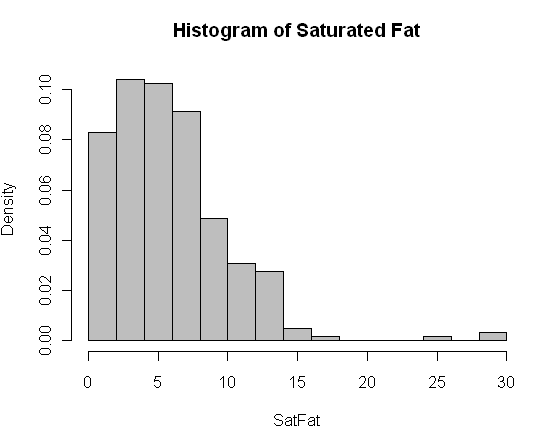
As shown in the following documentation, several optional arguments exist that can be used to modify the resulting plot.

|  |
| --- |
| Usage hist(x, ...)  ## Default S3 method:  hist(x, breaks = "Sturges",  freq = NULL, probability = !freq,  include.lowest = TRUE, right = TRUE,  density = NULL, angle = 45, col = NULL, border = NULL,  main = paste("Histogram of" , xname),  xlim = range(breaks), ylim = NULL,  xlab = xname, ylab,  axes = TRUE, plot = TRUE, labels = FALSE,  nclass = NULL, ...) |

For example, enter the following command at the prompt.

> hist(SatFat,breaks=20,freq=F,main="Histogram of Saturated Fat",col='gray')

R returns the following:



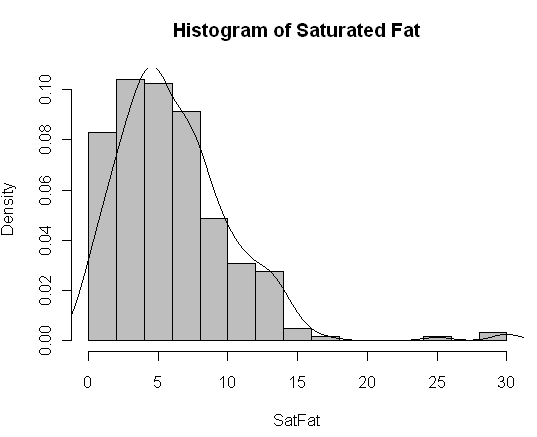
Tasks:

1. Change the freq= option to TRUE. What changes?
2. Change the breaks= option so that there are breakpoints at 0, 10, 20, and 30.

**Adding a Density Smoother to a Histogram in R**

The following command will add a “trend” to the histogram. This trend line is called a density smoother.

> lines(density(SatFat))



Once again, several optional arguments exist that can be used to modify the resulting density smoother.

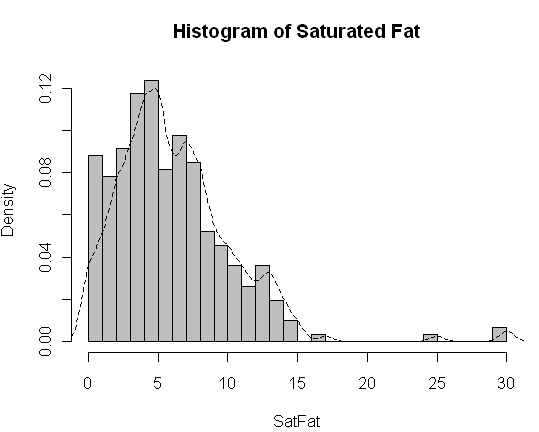
|  |
| --- |
| Usage density(x, ...)  ## Default S3 method:  density(x, bw = "nrd0", adjust = 1,  kernel = c("gaussian", "epanechnikov", "rectangular",  "triangular", "biweight",  "cosine", "optcosine"),  weights = NULL, window = kernel, width,  give.Rkern = FALSE,  n = 512, from, to, cut = 3, na.rm = FALSE, ...) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Usage lines(x, ...)  ## Default S3 method:  lines(x, y = NULL, type = "l", ...) Arguments  |  |  | | --- | --- | | x, y | coordinate vectors of points to join. | | type | character indicating the type of plotting; actually any of the types as in [plot.default](http://127.0.0.1:15653/help/library/graphics/help/plot.default). | | ... | Further graphical parameters (see [par](http://127.0.0.1:15653/help/library/graphics/help/par)) may also be supplied as arguments, particularly, line type, lty, line width, lwd, color, col and for type = "b", pch. Also the line characteristics lend, ljoin and lmitre. | |

For example, we can modify the appearance of the histogram/density smoother as follows:

> hist(SatFat,breaks=40,freq=F,main="Histogram of Saturated Fat",   
 col='gray')

> lines(density(SatFat,adjust=0.50),lty=2)



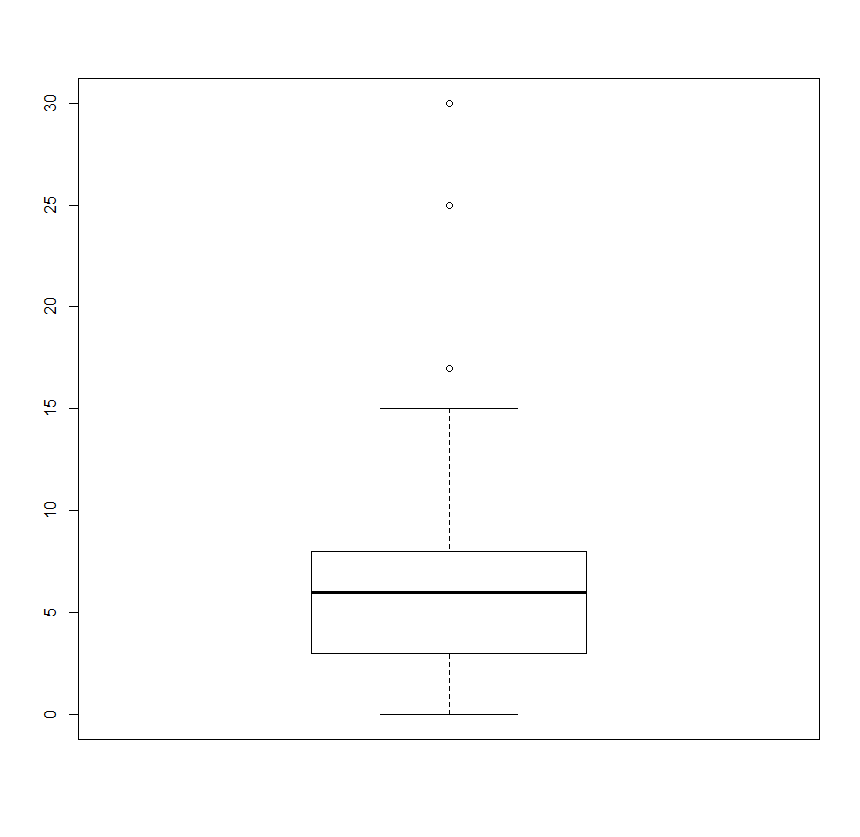
Tasks:

1. Change the adjust= option to a few different values. What changes?
2. Change the lty= option to 5 and then to “dotted”. What changes?

**BOXPLOTS**

The most basic form of the boxplot() function is employed below.

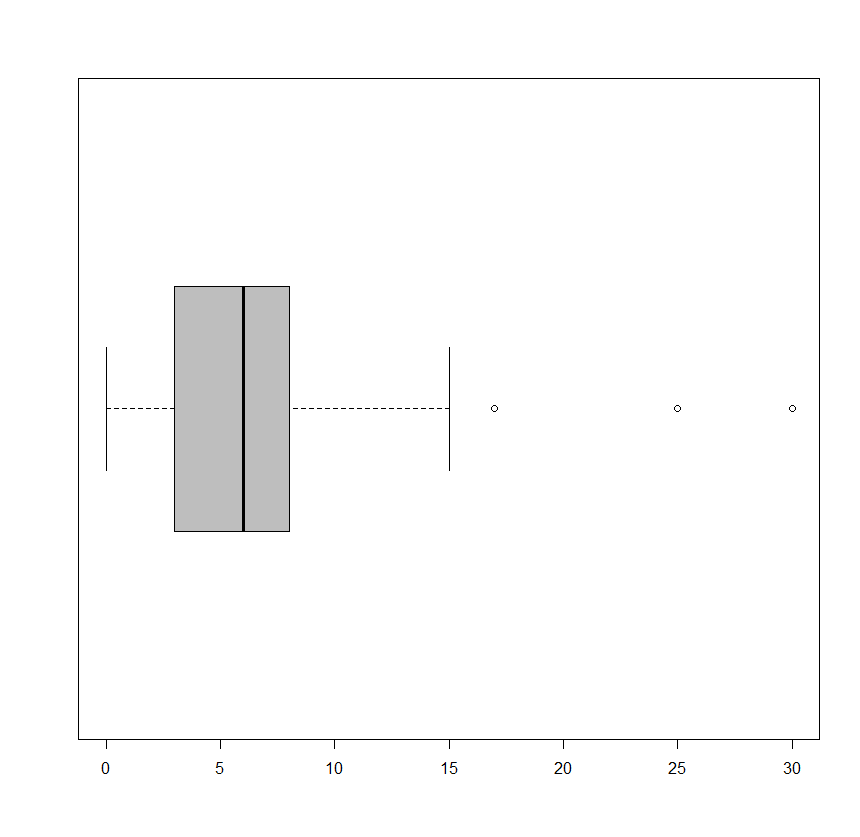
> boxplot(SatFat)



You can learn more about the optional arguments from the help documentation.

|  |
| --- |
| Usage boxplot(x, ...)  ## S3 method for class 'formula'  boxplot(formula, data = NULL, ..., subset, na.action = NULL)  ## Default S3 method:  boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,  notch = FALSE, outline = TRUE, names, plot = TRUE,  border = par("fg"), col = NULL, log = "",  pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),  horizontal = FALSE, add = FALSE, at = NULL) |

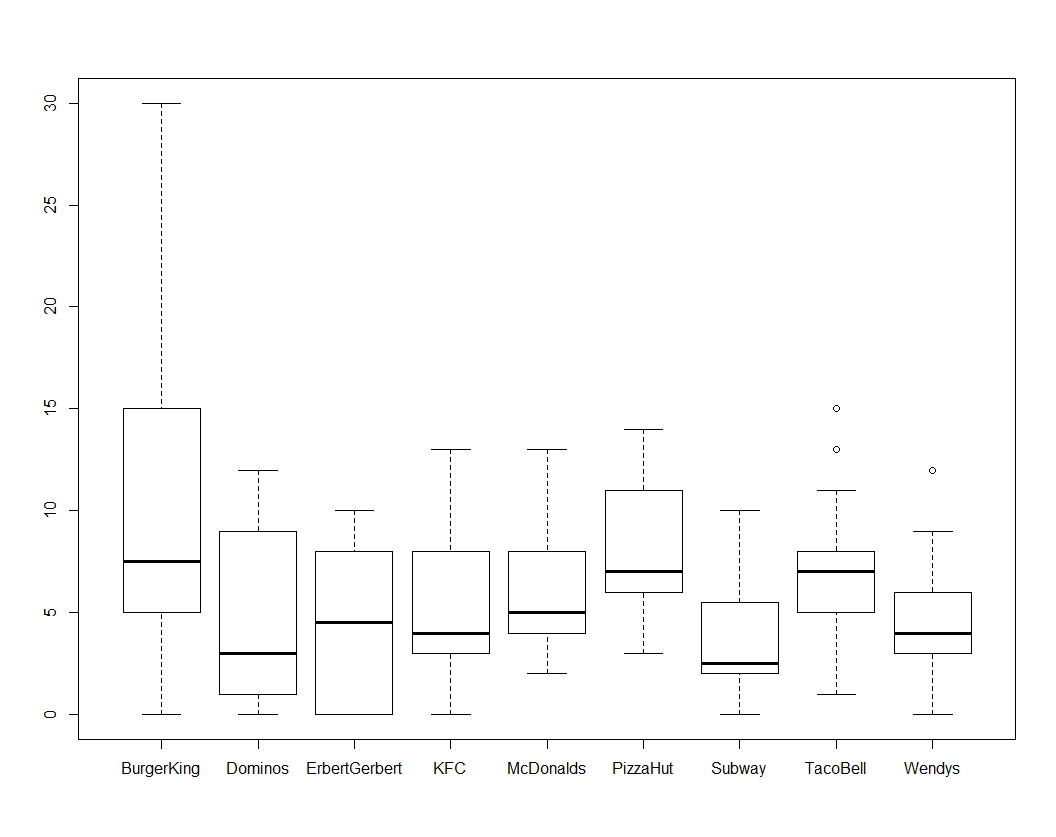
For example, you can change the orientation and color of the boxplot.  
  
> boxplot(SatFat, horizontal=T, col='gray')



You can also use the boxplot() function to obtain side-by-side boxplots.

Option 1: > boxplot(SatFat ~ Location, NutritionData)

Option 2: > boxplot(NutritionData$SatFat ~ NutritionData$Location)



Finally, you can change the width of the boxes to reflect the sample size as shown below.

Recall that the table() function returns a vector containing the counts for each group.

> table(Location)

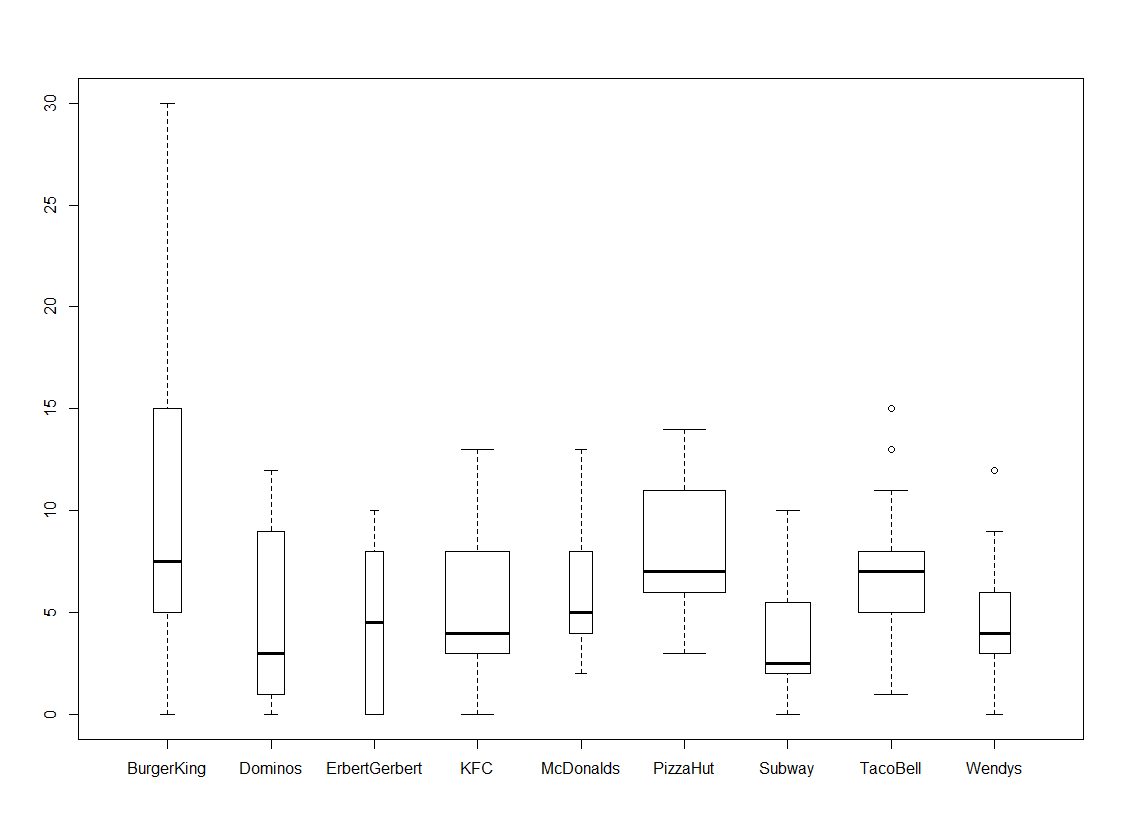
Location

BurgerKing Dominos ErbertGerbert KFC McDonalds PizzaHut Subway TacoBell Wendys

22 22 14 51 18 66 36 53 25

These counts can subsequently be used in the boxplot() function to change the width of each location’s boxplot to reflect the sample size from that location.

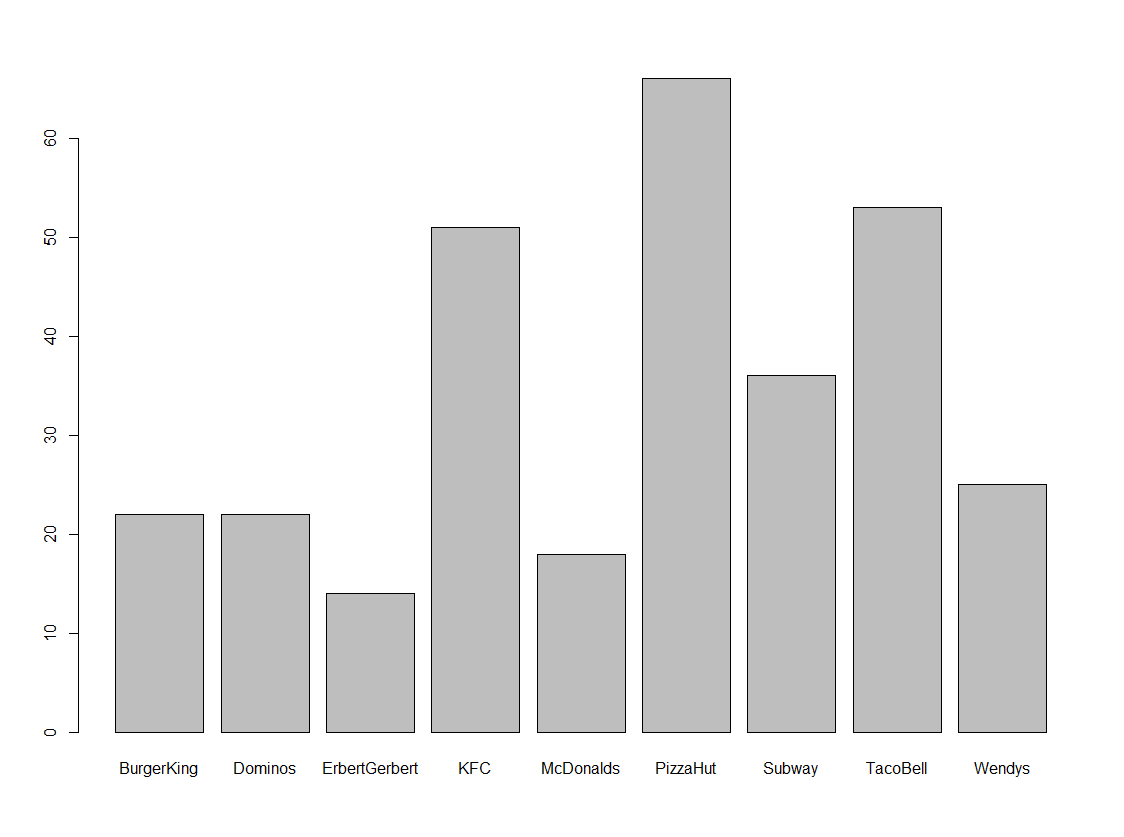
> boxplot(SatFat ~ Location, NutritionData, width=table(Location))



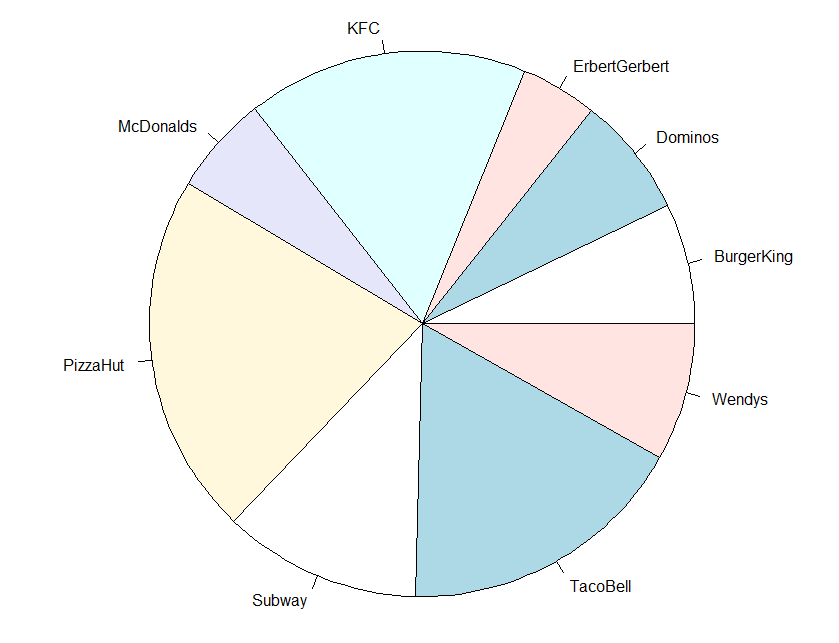
**BAR CHARTS AND PIE CHARTS**

You can obtain bar charts and/or pie charts using the following functions in R.

> barplot(table(Location))



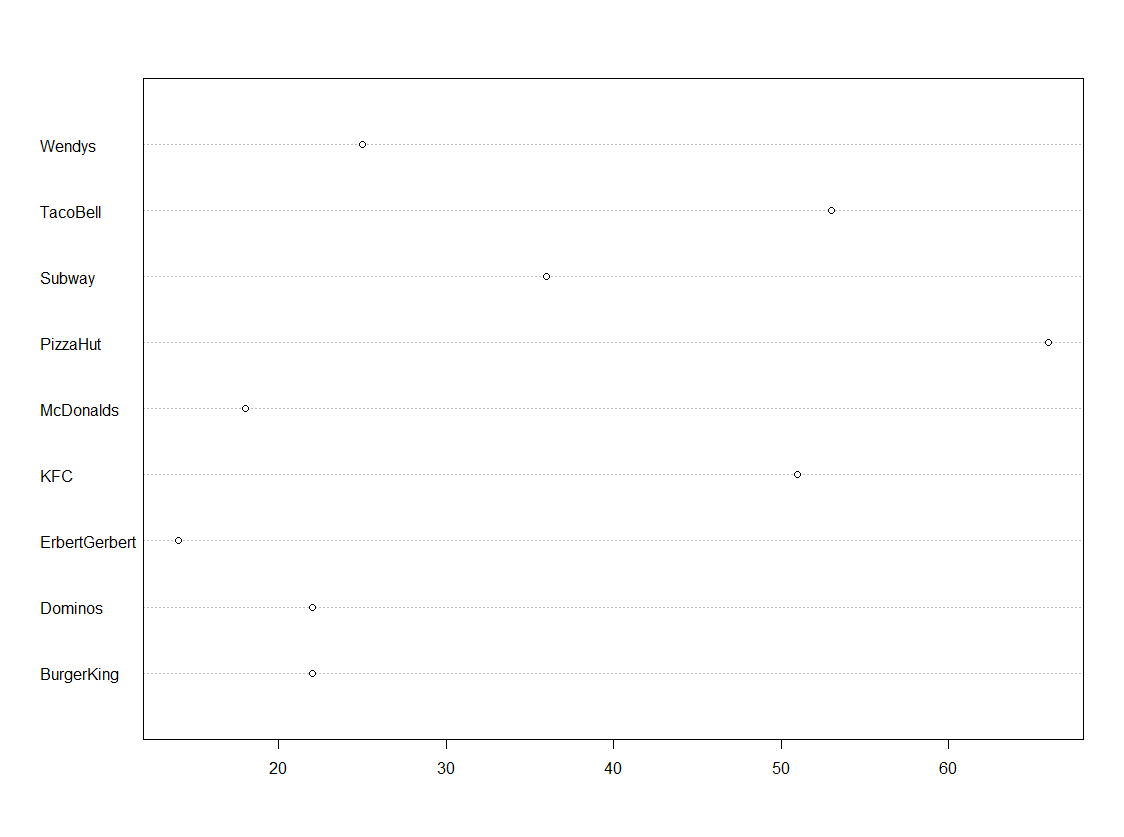
> pie(table(Location))



Note the following comment from the R documentation regarding pie charts:

|  |
| --- |
| NotePie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data. Cleveland (1985), page 264: “Data that can be shown by pie charts always can be shown by a dot chart. This means that judgements of position along a common scale can be made instead of the less accurate angle judgements.” This statement is based on the empirical investigations of Cleveland and McGill as well as investigations by perceptual psychologists. |

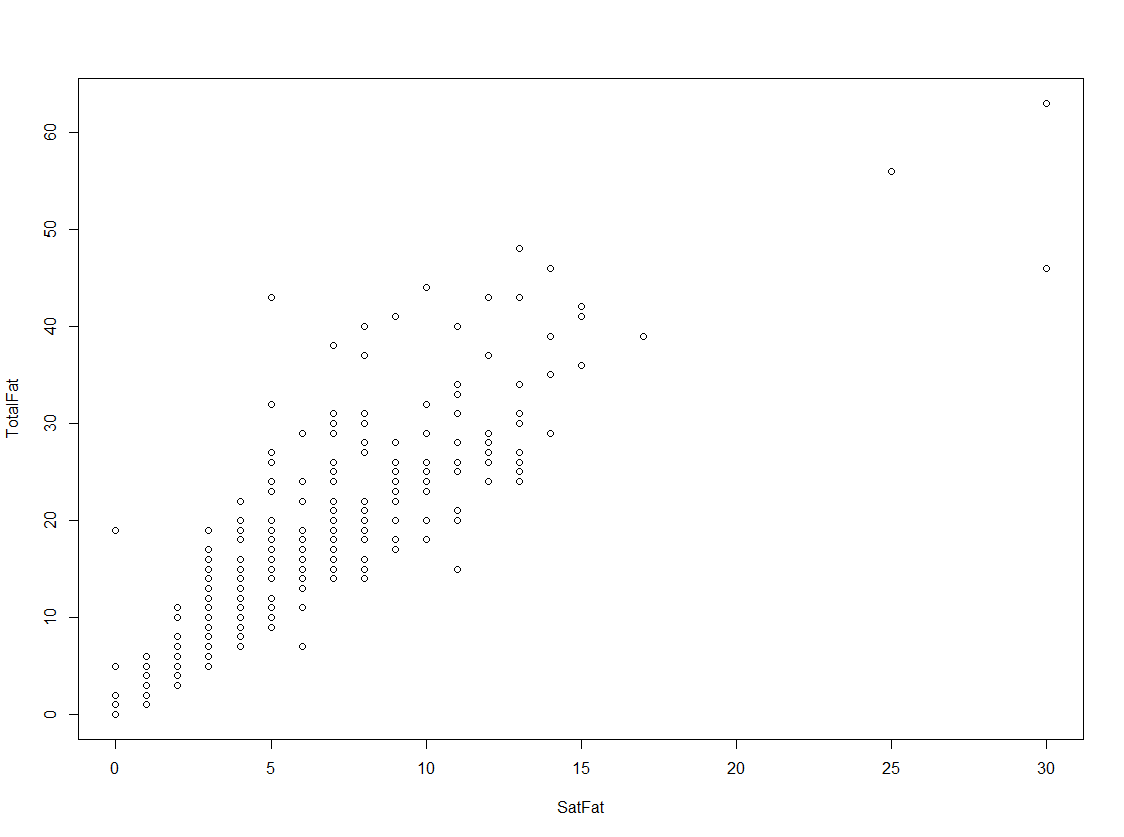
> dotchart(table(Location))



**SCATTERPLOTS AND SMOOTHERS**

Scatterplots are simple to create in R using the plot() function. For example, we could examine the relationship between *Saturated Fat* and *Total Fat* by creating the following plot.

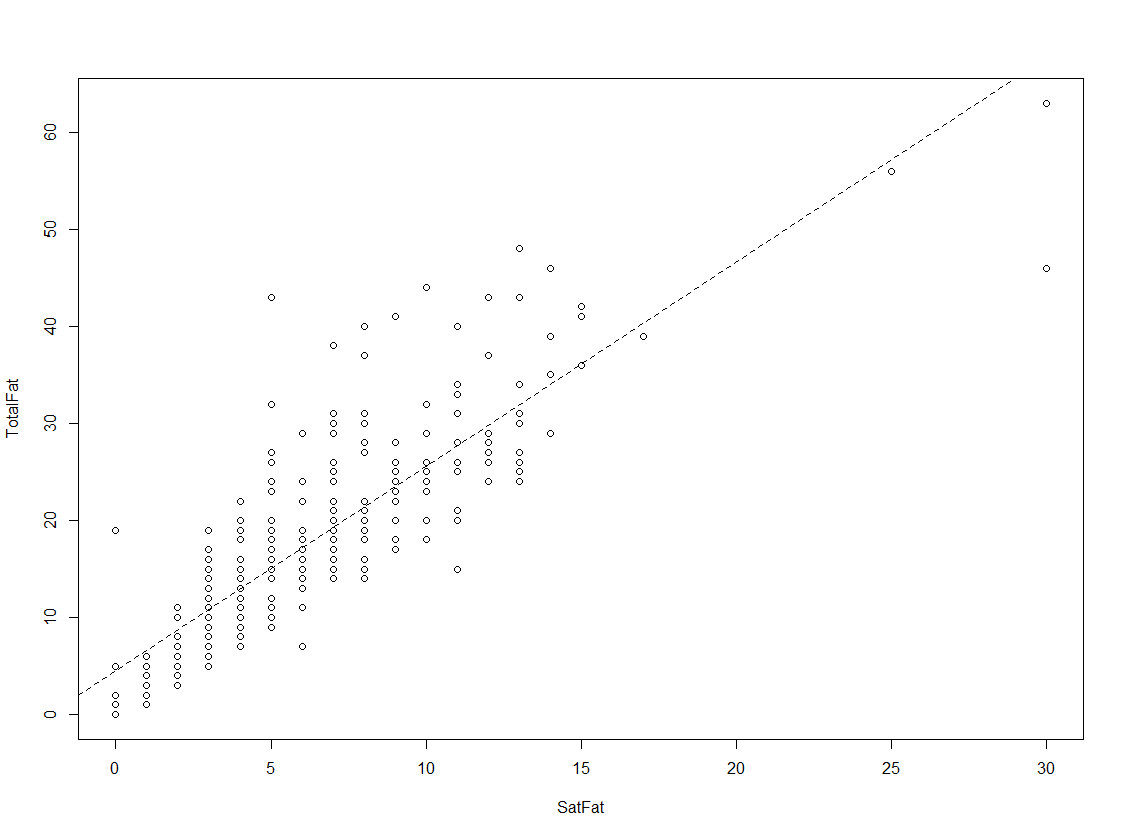
> plot(SatFat,TotalFat)



To add a trend line (i.e., the regression line) to this plot, you can use the abline() function.

> plot(SatFat,TotalFat)

> abline(lm(TotalFat~SatFat),lty=2)

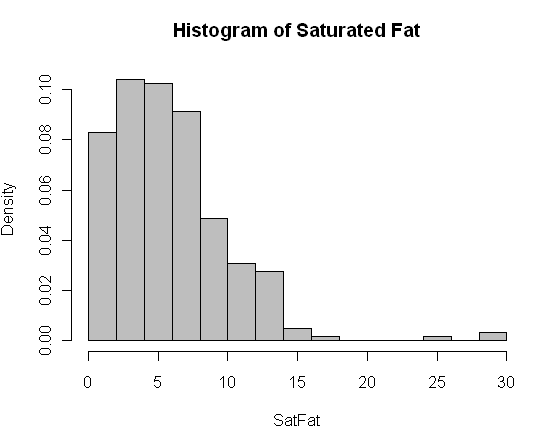


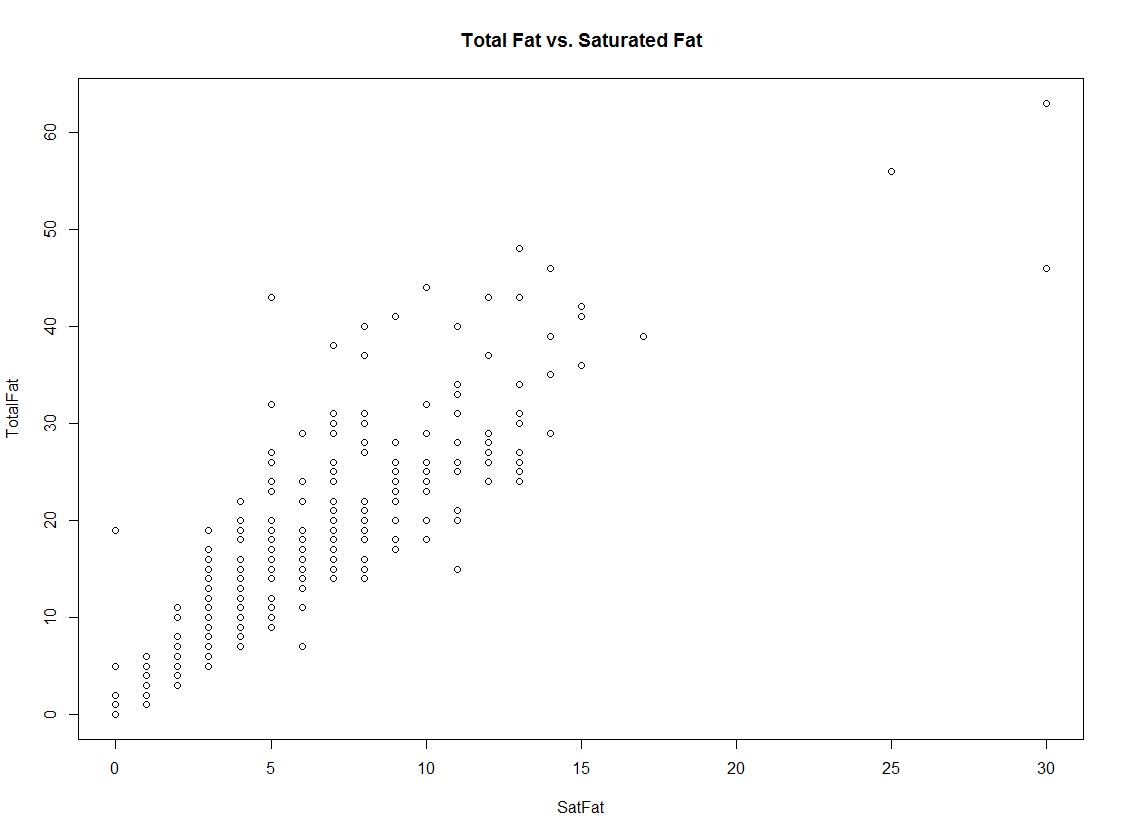
**CHANGING GRAPH PARAMETERS**

The basic graphing functions we have discussed so far all have modifiable parameters,   
some of which we have observed (e.g., changing the breaks in a histogram). The   
following examples highlight modifications that are commonly made to graphs created in R.  
  
**Adding a Main Title to a Graph**

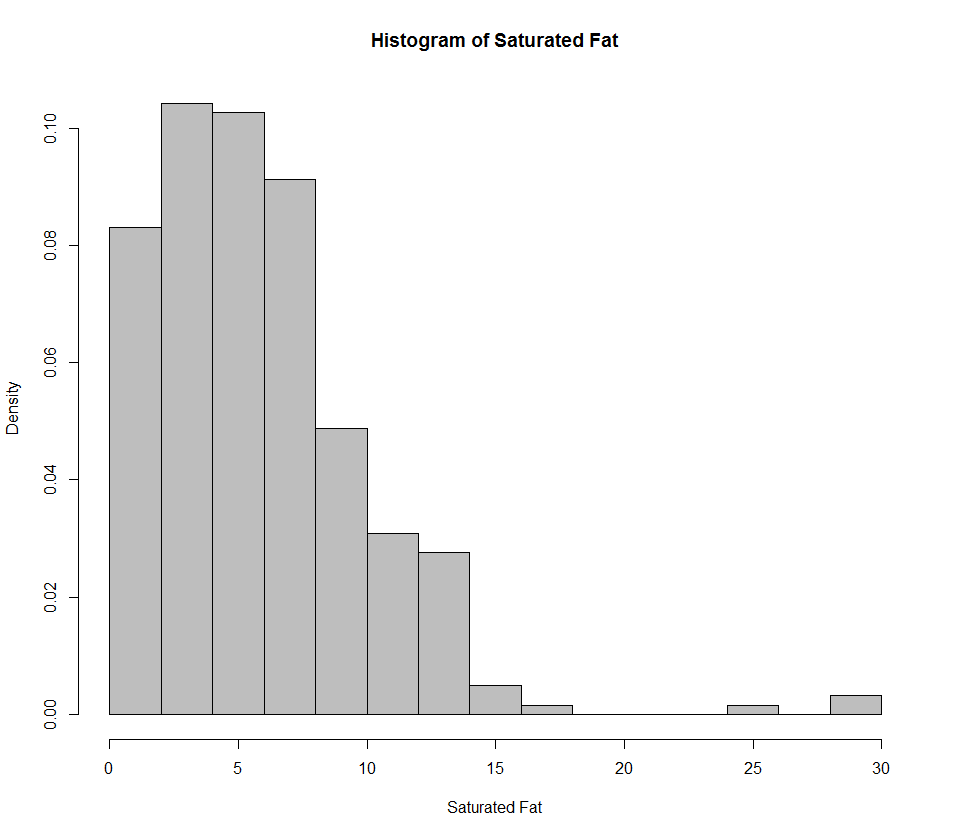
Earlier in this handout, we added a title to the histogram of *Saturated Fat* values.

> hist(SatFat,breaks=20,freq=F,main="Histogram of Saturated Fat",col='gray')

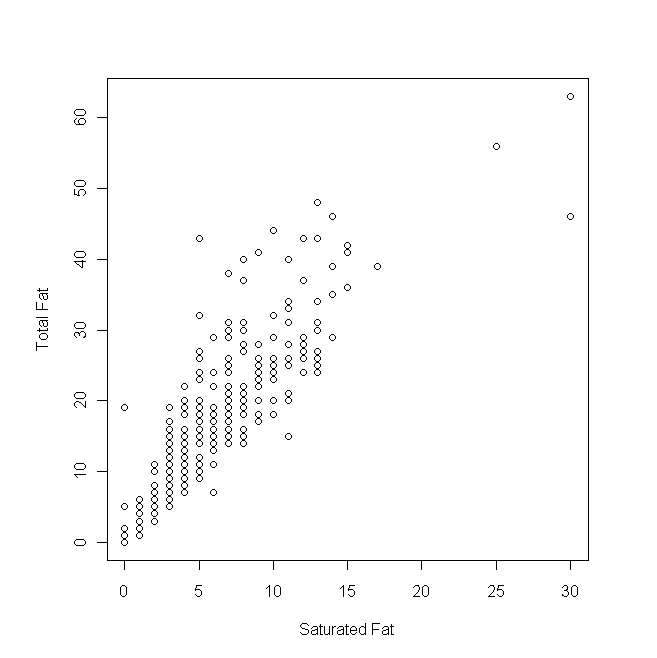


Next, consider the scatterplot of *Total Fat* vs. *Saturated Fat*. We could similarly add a title to this plot using the following command.  
  
> plot(SatFat,TotalFat, main="Total Fat vs. Saturated Fat")  
  


**Changing Axis Labels**We can change the x-axis label on the histogram of *Saturated Fat* as follows.  
  
> hist(SatFat,breaks=20,freq=F,main="Histogram of Saturated Fat",  
 col='gray', xlab="Saturated Fat")

  
  
Similarly, we can change the axis labels on the scatterplot of *Total Fat* vs. *Saturated Fat*.

> plot(SatFat,TotalFat,xlab="Saturated Fat", ylab="Total Fat")

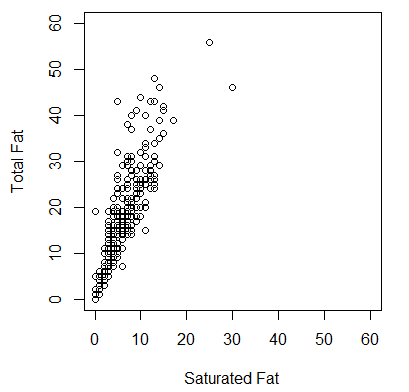


**Changing the Range on the X- and/or Y-Axes**

The following command will change the range on the x- and y-axis from the defaults. Note that the purpose of the par() function is used to make the plot region square.

> par(pty="square")

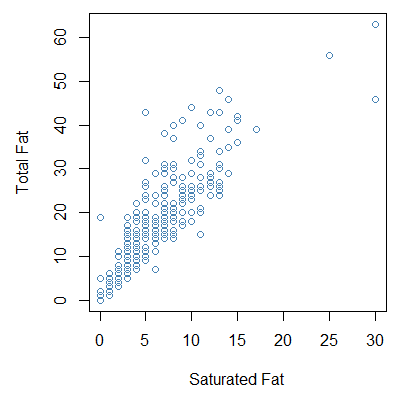
> plot(SatFat,TotalFat,xlab="Saturated Fat", ylab="Total Fat",   
 xlim=c(0,60),ylim=c(0,60))



**Changing the Color of Plotting Symbols**

To see all of the colors available in R, type colors() at the prompt. You can then change the color of a symbol as follows.

> plot(SatFat,TotalFat,xlab="Saturated Fat", ylab="Total Fat",   
 col="steelblue")



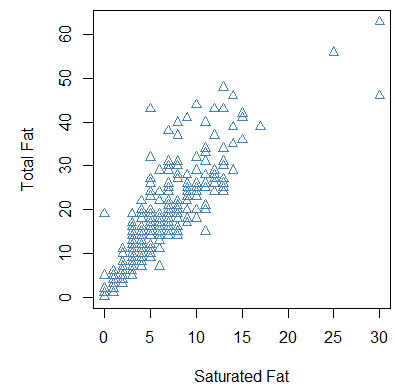
**Changing the Plotting Symbols**

To see a list of the most common plotting symbols, type ?points at the prompt.



To make the plotting symbol open triangles instead of open circles, you could use the following command.

plot(Nutritional$SatFat, Nutritional$TotalFat,xlab="Saturated Fat", ylab="Total Fat", col="steelblue", pch=2)



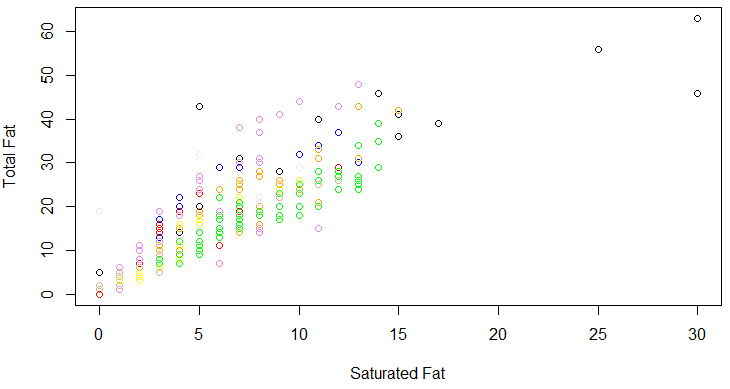
Tasks:

1. Create the above graph using a red open square as the plotting symbol.
2. Create the above graph using a red square filled with another color of your choice as the plotting symbol.

**Changing the Color of the Plotting Symbols Based on Levels Another Variable**

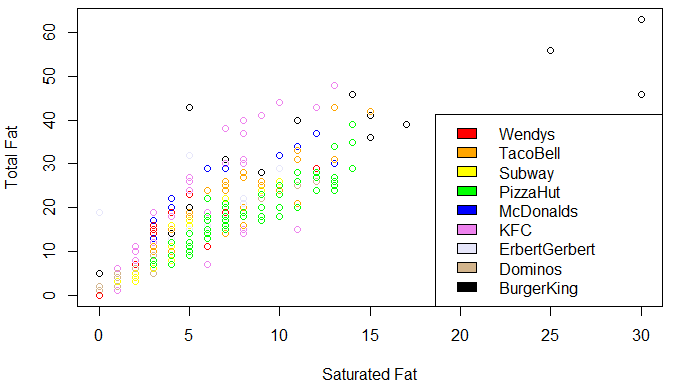
Finally, note that we could also color the symbols in the scatterplot according to *Location*.

> plot(Nutritional$SatFat, Nutritional$TotalFat,xlab="Saturated Fat", ylab="Total Fat",   
 col=c('red','orange','yellow','green','blue','violet','lavender',  
 'tan','black' ) [match(Nutritional$Location,c("Wendys","TacoBell",  
 "Subway","PizzaHut","McDonalds","KFC","ErbertGerbert","Dominos",  
 "BurgerKing"))])



You can add a legend to the graph as follows.

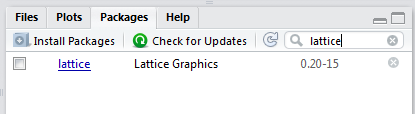
>legend("bottomright",legend=c("Wendys","TacoBell","Subway","PizzaHut",  
 "McDonalds","KFC","ErbertGerbert","Dominos","BurgerKing"),fill=  
 c('red', 'orange', 'yellow', 'green', 'blue', 'violet', 'lavender',   
 'tan', 'black' ))



**LATTICE GRAPHICS**

The lattice package is a very powerful add-on package that implements Trellis graphics in R.

To load this (or any other) package in R, go to the lower right-hand window of the R Studio window. You can search for the package of interest.



Here, you can check the box next to “lattice” and note that R automatically runs the following command.

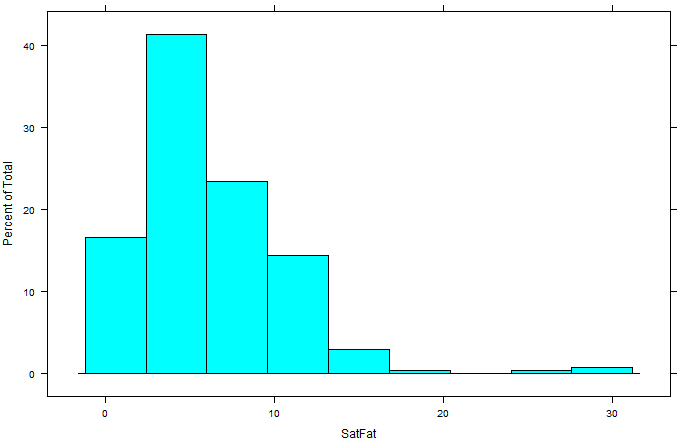
> library("lattice", lib.loc="C:/Program Files/R/R-3.2.3/library")

Once this command has been entered, you can use the package. If the package is not listed on your local machine, you can select “Install Packages.” There are literally hundreds of available packages for R; some are great and others aren’t, so be careful.

**Obtaining a Histogram Using the Lattice Package**

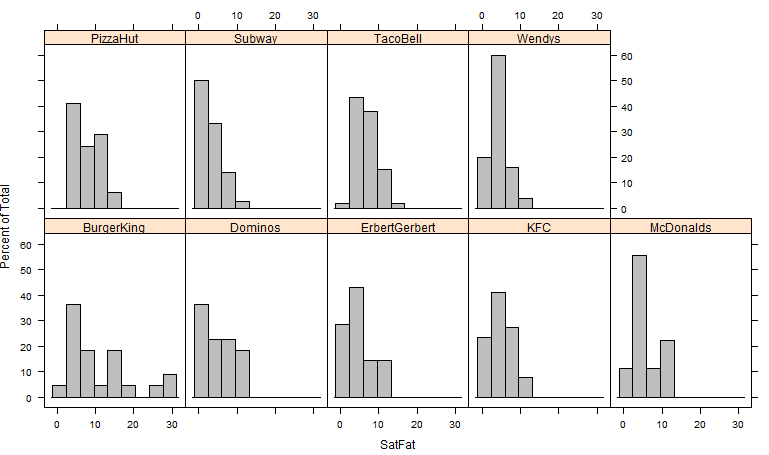
You can use the histogram() function once the lattice package has been installed:

> histogram(~SatFat, data=Nutritional)



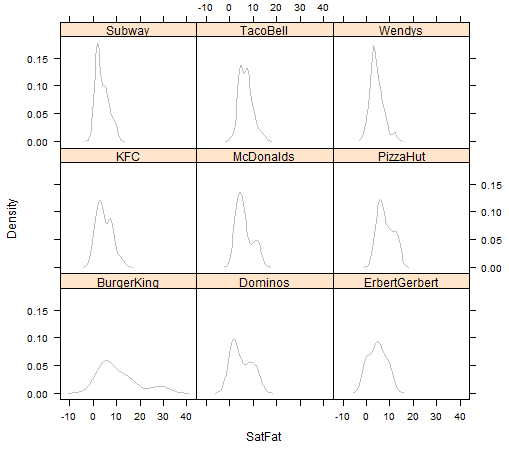
Note that a more interesting display would compare the distribution of Saturated Fat across Location. This is easily implemented with lattice graphics.

> histogram(~SatFat| Location,data=Nutritional,col="gray")



**Obtaining a Density Plot Using the Lattice Package**

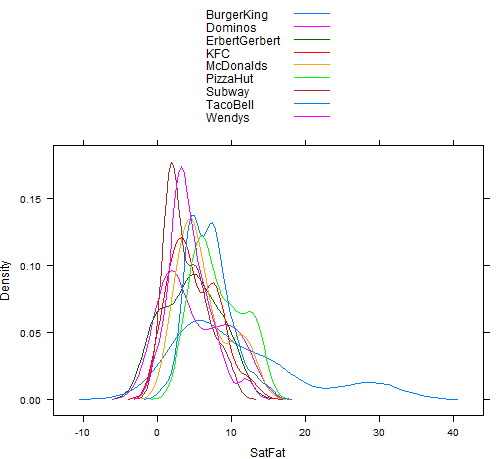
> densityplot(~SatFat|Location,data= Nutritional,col="gray",  
 plot.points=FALSE)



Task: Re-submit the above command with the plot.points argument omitted. What happens?

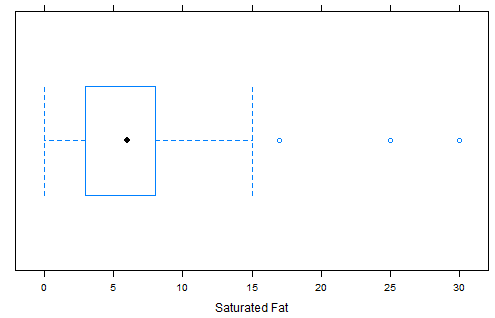
Note that instead of displaying the density plots in a separate panel for each location, you could alternatively overlay the density plots as follows.

> densityplot(~SatFat,data= Nutritional,groups=Location,  
 plot.points=FALSE,auto.key=TRUE)

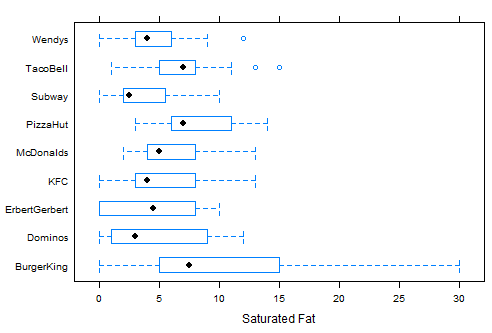


**Obtaining Boxplots Using the Lattice Package**A boxplot for Saturated Fat can be obtained with the bwplot() function.

> bwplot(~SatFat, data=Nutritional xlab="Saturated Fat")

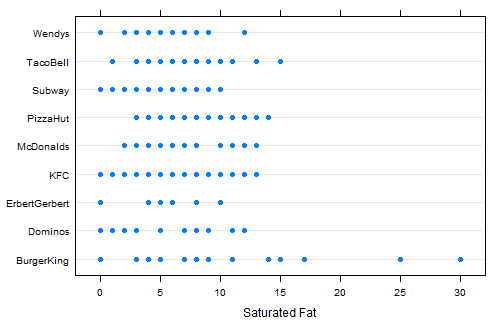


Comparative boxplots can be obtained as follows:  
  
> bwplot(Location ~ SatFat, data=Nutritional, xlab="Saturated Fat")



**Obtaining Dotplots Using the Lattice Package**

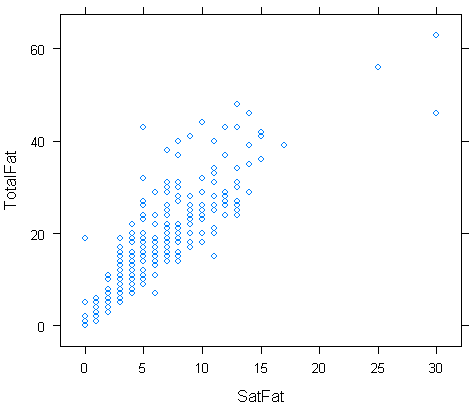
> dotplot(Location ~ SatFat,data=Nutritional, xlab="Saturated Fat")



Task: Enter the following command at the prompt. Compare this to the code and resulting graph from page 8.   
  
> dotplot(~Location, data=Nutritional)

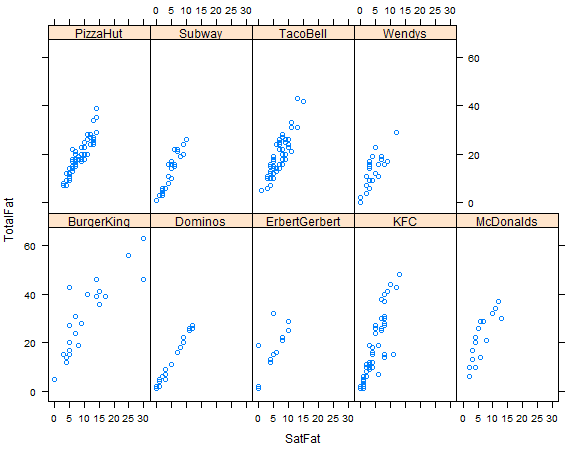
**Obtaining a Scatterplot Using the Lattice Package**

> xyplot(TotalFat~SatFat, data=Nutritional)



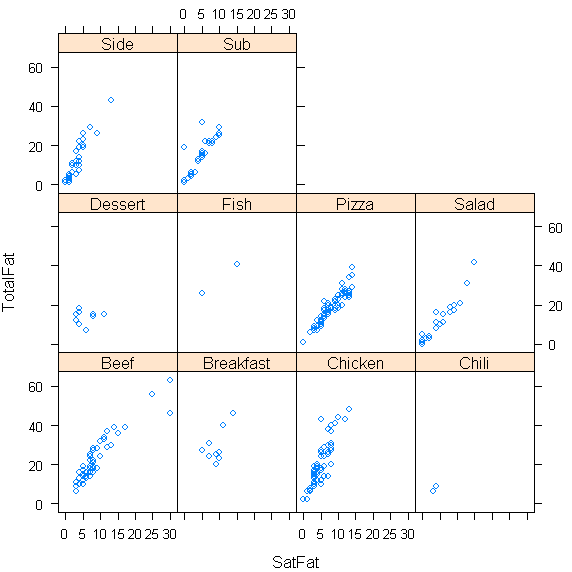
Next, note that you can also obtain the scatterplot above for each location fairly easily using the built-in conditioning functionality provided by the lattice package.

> xyplot(TotalFat~SatFat | Location, data=Nutritional)



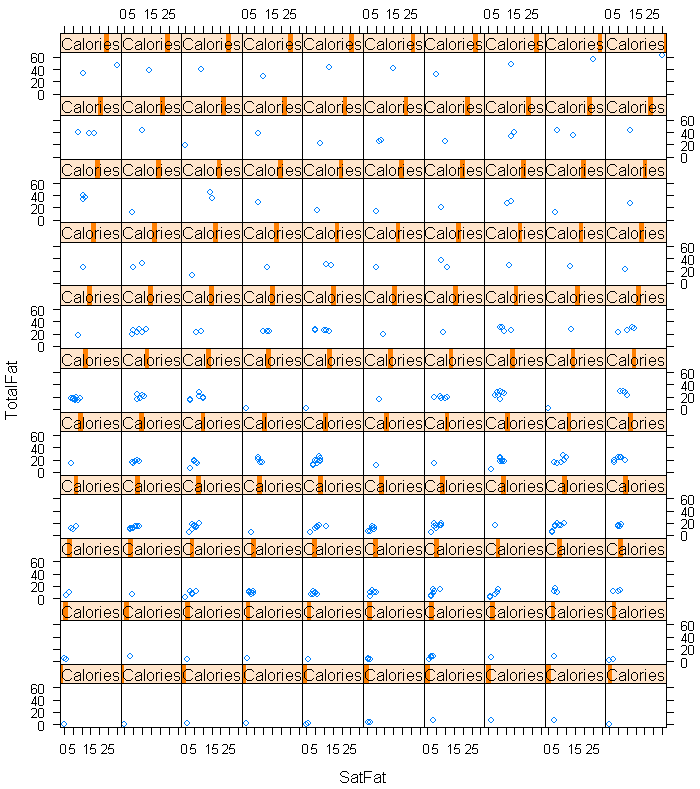
We can also get a scatterplot for each Type.

> xyplot(TotalFat~SatFat | Type, data=Nutritional)



Note what happens if the conditioning variable is *continuous*.

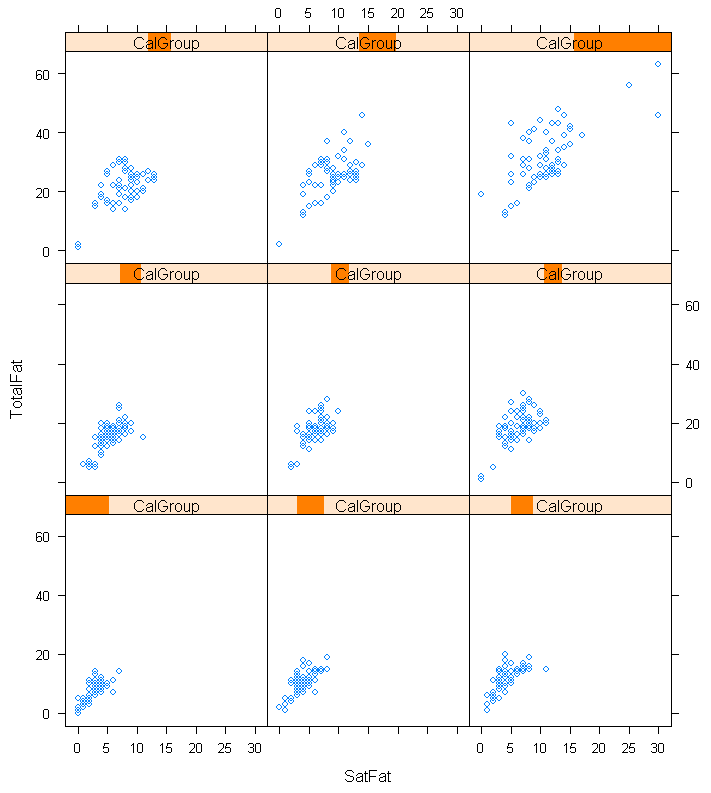
> xyplot(TotalFat~SatFat | Calories, data=Nutritional)



This can be modified by specifying groupings for the Calorie variable using the equal.count() function. The following command specifies that the values of *Calorie* be divided into nine groups, each with about the same number of observations.

> CalGroup = equal.count(Nutritional$Calories,number=9)

> xyplot(TotalFat~SatFat | CalGroup, data=Nutritional)



**Obtaining Scatterplot Matrices Using the Lattice Package**

The splom() function can be used to create a scatterplot matrix for the numerical variables in this data set.

> splom(Nutritional[,4:10])

