Last Lecture

- Introduction to R Programming
- Controlling Loops
- Defining your own functions
Today

- Introduction to Graphics in R
- Examples of commonly used graphics functions
- Common options for customizing graphs
Computer Graphics

- Graphics are important for conveying important features of the data

- They can be used to examine
  - Marginal distributions
  - Relationships between variables
  - Summary of very large data

- Important complement to many statistical and computational techniques
Example Data

The examples in this lecture will be based on a dataset with six variables:

- Height (in cm)
- Weight (in kg)
- Waist Circumference (in cm)
- Hip Circumference (in cm)
- Systolic Blood Pressure
- Diastolic Blood Pressure
<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Waist</th>
<th>Hip</th>
<th>bp.sys</th>
<th>bp.dia</th>
</tr>
</thead>
<tbody>
<tr>
<td>172</td>
<td>72</td>
<td>87</td>
<td>94</td>
<td>127.5</td>
<td>80</td>
</tr>
<tr>
<td>166</td>
<td>91</td>
<td>109</td>
<td>107</td>
<td>172.5</td>
<td>100</td>
</tr>
<tr>
<td>174</td>
<td>80</td>
<td>95</td>
<td>101</td>
<td>123</td>
<td>64</td>
</tr>
<tr>
<td>176</td>
<td>79</td>
<td>93</td>
<td>100</td>
<td>117</td>
<td>76</td>
</tr>
<tr>
<td>166</td>
<td>55</td>
<td>70</td>
<td>94</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>163</td>
<td>76</td>
<td>96</td>
<td>99</td>
<td>160</td>
<td>87.5</td>
</tr>
<tr>
<td>154</td>
<td>84</td>
<td>98</td>
<td>118</td>
<td>130</td>
<td>80</td>
</tr>
<tr>
<td>165</td>
<td>90</td>
<td>108</td>
<td>101</td>
<td>139</td>
<td>80</td>
</tr>
<tr>
<td>155</td>
<td>66</td>
<td>80</td>
<td>96</td>
<td>120</td>
<td>70</td>
</tr>
<tr>
<td>146</td>
<td>59</td>
<td>77</td>
<td>96</td>
<td>112.5</td>
<td>75</td>
</tr>
<tr>
<td>164</td>
<td>62</td>
<td>76</td>
<td>93</td>
<td>130</td>
<td>47.5</td>
</tr>
<tr>
<td>159</td>
<td>59</td>
<td>76</td>
<td>96</td>
<td>109</td>
<td>69</td>
</tr>
<tr>
<td>163</td>
<td>69</td>
<td>96</td>
<td>99</td>
<td>155</td>
<td>100</td>
</tr>
<tr>
<td>143</td>
<td>73</td>
<td>97</td>
<td>117</td>
<td>137.5</td>
<td>85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Reading in the Data

```r
> dataset <- read.table("815data.txt", header = T)
> summary(dataset)
```

<table>
<thead>
<tr>
<th></th>
<th>Height</th>
<th>Weight</th>
<th>Waist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>131.0</td>
<td>0.00</td>
<td>0.0</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>153.0</td>
<td>55.00</td>
<td>74.0</td>
</tr>
<tr>
<td>Median</td>
<td>159.0</td>
<td>63.00</td>
<td>84.0</td>
</tr>
<tr>
<td>Mean</td>
<td>159.6</td>
<td>64.78</td>
<td>84.6</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>166.0</td>
<td>74.00</td>
<td>94.0</td>
</tr>
<tr>
<td>Max.</td>
<td>196.0</td>
<td>135.00</td>
<td>134.0</td>
</tr>
</tbody>
</table>

...
Graphics in R

- `plot()` is the main graphing function
- Automatically produces simple plots for vectors, functions or data frames
- Many useful customization options…
Plotting a Vector

- `plot(v)` will print the elements of the vector `v` according to their index

```r
# Plot height for each observation
> plot(dataset$Height)
# Plot values against their ranks
> plot(sort(dataset$Height))
```
Plotting a Vector

```r
plot(dataset$Height)  
plot(sort(dataset$Height))
```
Common Parameters for plot()

- Specifying labels:
  - `main` – provides a title
  - `xlab` – label for the x axis
  - `ylab` – label for the y axis

- Specifying range limits
  - `ylim` – 2-element vector gives range for x axis
  - `xlim` – 2-element vector gives range for y axis
A Properly Labeled Plot

```
plot(sort(dataset$Height), ylim = c(120,200),
     ylab = "Height (in cm)", xlab = "Rank", main = "Distribution of Heights")
```
Plotting Two Vectors

- `plot()` can pair elements from 2 vectors to produce x-y coordinates

- `plot()` and `pairs()` can also produce composite plots that pair all the variables in a data frame.
Plotting Two Vectors

plot(dataset$Hip, dataset$Waist, 
     xlab = "Hip", ylab = "Waist", 
     main = "Circumference (in cm)", pch = 2, col = "blue")
Plotting Two Vectors

```
plot(dataset$Hip, dataset$Waist,
     xlab = "Hip", ylab = "Waist",
     main = "Circumference (in cm)", pch = 2, col = "blue")
```
Plotting Two Vectors

```
plot(dataset$Hip, dataset$Waist,
     xlab = "Hip", ylab = "Waist",
     main = "Circumference (in cm)", pch = 2, col = "blue"
```

Possible Outlier

These options set
the plotting symbol
(pch) and line color
(col)
plot(dataset[-c(4,5,6)])
Plotting Contents of a Dataset

Weight and Waist Circumference Appear Strongly Correlated

You could check this with the `cor()` function.

```
plot(dataset[-c(4,5,6)])
```
Histograms

- Generated by the `hist()` function

- The parameter `breaks` is key
  - Specifies the number of categories to plot
  - Specifies the breakpoints for each category

- The `xlab`, `ylab`, `xlim`, `ylim` options work as expected
```r
hist(dataset$bp.sys, col = "lightblue",
     xlab = "Systolic Blood Pressure", main = "Blood Pressure")
```
Can you explain the peculiar pattern? Graphical representations of data are useful at identifying these sorts of artifacts…

```r
hist(dataset$bp.sys, col = "lightblue", breaks = seq(80,220,by=2),
     xlab = "Systolic Blood Pressure", main = "Blood Pressure")
```
Boxplots

• Generated by the boxplot() function

• Draws plot summarizing
  • Median
  • Quartiles (Q1, Q3)
  • Outliers – by default, observations more than 1.5 * (Q1 – Q3) distant from nearest quartile
A Simple Boxplot

boxplot(dataset, col = rainbow(6), ylab = "Appropriate Units")
Adding Individual Observations

- `rug()` can add a tick for each observation to the side of a `boxplot()` and other plots.

- The `side` parameter specifies where tickmarks are drawn.

```r
> boxplot(dataset$Weight, 
            main = "Weight (in kg)",
            col = "red")
> rug(dataset$Weight, side = 2)
```
Customizing Plots

- R provides a series of functions for adding text, lines and points to a plot

- We will illustrate some useful ones, but look at `demo(graphics)` for more examples
Drawing on a plot

- To add additional data use
  - `points(x,y)`
  - `lines(x,y)`

- For freehand drawing use
  - `polygon()`
  - `rect()`
Text Drawing

- Two commonly used functions:
  - `text()` – writes inside the plot region, could be used to label datapoints
  - `mtext()` – writes on the margins, can be used to add multiline legends

- These two functions can print mathematical expressions created with `expression()`
Plotting Two Data Series

```r
> x <- seq(0,2*pi, by = 0.1)
> y <- sin(x)
> y1 <- cos(x)
> plot(x,y, col = "green", type = "l", lwd = 3)
> lines(x,y1, col = "red", lwd = 3)
> mtext("Sine and Cosine Plot", side = 3, line = 1)
```
Printing on Margins, Using Symbolic Expressions

```r
> f <- function(x) x * (x + 1) / 2
> x <- 1:20
> y <- f(x)
> plot(x, y, xlab = "", ylab = "")
> mtext("Plotting the expression", side = 3, line = 2.5)
> mtext(expression(y == sum(1, x, i)), side = 3, line = 0)
> mtext("The first variable", side = 1, line = 3)
> mtext("The second variable", side = 2, line = 3)
```

![Plotting the expression](plot.png)

The first variable
The second variable
Adding a Label Inside a Plot

Who will develop obesity?

```r
> hist(dataset$Weight, xlab = "Weight",
       main = "Who will develop obesity?", col = "blue")
> rect(90, 0, 120, 1000, border = "red", lwd = 4)
> text(105, 1100, "At Risk", col = "red", cex = 1.25)
```
## Symbolic Math
### Example from `demo(plotmath)`

<table>
<thead>
<tr>
<th>Big Operators</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sum(x[i], i = 1, n)</td>
<td>( \sum_{i=1}^{n} x_i )</td>
</tr>
<tr>
<td>prod(plain(P)(X == x), x)</td>
<td>( \prod_{x} P(X = x) )</td>
</tr>
<tr>
<td>integral(f(x) * dx, a, b)</td>
<td>( \int_{a}^{b} f(x)dx )</td>
</tr>
<tr>
<td>union(A[i], i == 1, n)</td>
<td>( \bigcup_{i=1}^{n} A_i )</td>
</tr>
<tr>
<td>intersect(A[i], i == 1, n)</td>
<td>( \bigcap_{i=1}^{n} A_i )</td>
</tr>
<tr>
<td>lim(f(x), x %&gt;% 0)</td>
<td>( \lim_{x \to 0} f(x) )</td>
</tr>
<tr>
<td>min(g(x), x &gt;= 0)</td>
<td>( \min_{x \geq 0} g(x) )</td>
</tr>
<tr>
<td>inf(S)</td>
<td>( \inf S )</td>
</tr>
<tr>
<td>sup(S)</td>
<td>( \sup S )</td>
</tr>
</tbody>
</table>
Further Customization

- The `par()` function can change defaults for graphics parameters, affecting subsequent calls to `plot()` and friends.

- Parameters include:
  - `cex, mex` – text character and margin size
  - `pch` – plotting character
  - `xlog, ylog` – to select logarithmic axis scaling
Multiple Plots on A Page

- Set the `mfrow` or `mfcol` options
  - Take 2 dimensional vector as an argument
  - The first value specifies the number of rows
  - The second specifies the number of columns

- The 2 options differ in the order individual plots are printed
Multiple Plots

> par(mfcol = c(3,1))
> hist(dataset$Height,
  breaks = 10,
  main = "Height (in cm)",
  xlab = "Height")
> hist(dataset$Height * 10,
  breaks = 10,
  main = "Height (in mm)",
  xlab = "Height")
> hist(dataset$Height / 2.54,
  breaks = 10,
  main = "Height (in inches)",
  xlab = "Height")
Outputting R Plots

- R usually generates output to the screen.

- In Windows and the Mac, you can point and click on a graph to copy it to the clipboard.

- However, R can also save its graphics output in a file that you can distribute or include in a document prepared with Word or LATEX.
Selecting a Graphics Device

- To redirect graphics output, first select a device:
  - `pdf()` – high quality, portable format
  - `postscript()` – high quality format
  - `png()` – low quality, but suitable for the web

- After you generate your graphics, simply close the device
  - `dev.off()`
Example of Output Redirection

> x <- runif(100)
> y <- runif(100) * 0.5 + x * 0.5

# This graph is plotted on the screen
> plot(x, y, ylab = "This is a simple graph")

# This graph is plotted to the PDF file
> pdf("my_graph.pdf")
> plot(x, y, ylab = "This is a simple graph")
> dev.close()

# Where does this one go?
> plot(x, y, ylab = "This is a simple graph")
Today

- Introduction to Graphics in R
- Examples of commonly used graphics functions
- Common options for customizing graphs