Graphics in R

Biostatistics 615/815

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

The Data File

Height	Weight	Waist	Hip	bp.sys	bp.dia
172	72	87	94	127.5	80
166	91	109	107	172.5	100
174	80	95	101	123	64
176	79	93	100	117	76
166	55	70	94	100	60
163	76	96	99	160	87.5
154	84	98	118	130	80
165	90	108	101	139	80
155	66	80	96	120	70
146	59	77	96	112.5	75
164	62	76	93	130	47.5
159	59	76	96	109	69
163	69	96	99	155	100
143	73	97	117	137.5	85

. . .

Reading in the Data

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

     Height           Weight            Waist
 Min.   :131.0   Min.   :  0.00   Min.   :  0.0
 1st Qu.:153.0   1st Qu.: 55.00   1st Qu.: 74.0
 Median :159.0   Median : 63.00   Median : 84.0
 Mean   :159.6   Mean   : 64.78   Mean   : 84.6
 3rd Qu.:166.0   3rd Qu.: 74.00   3rd Qu.: 94.0
 Max.   :196.0   Max.   :135.00   Max.   :134.0

. . .
```

Graphics in R

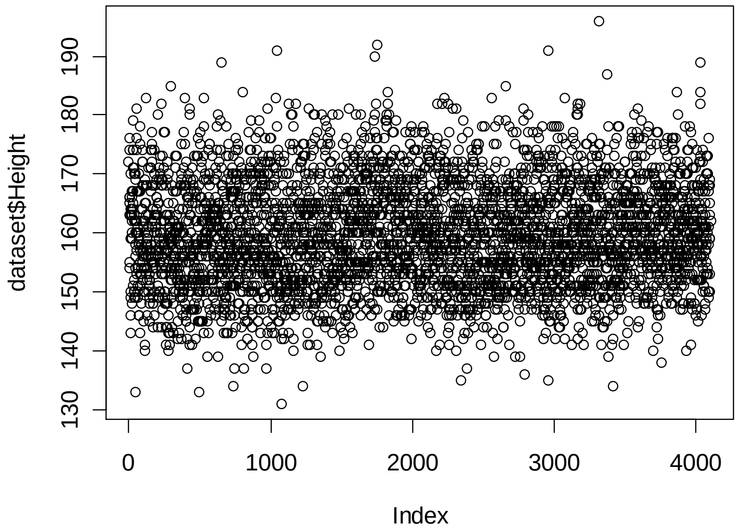
- `plot()` is the main graphing function
- Automatically produces simple plots for vectors, functions or data frames
- Many useful customization options…

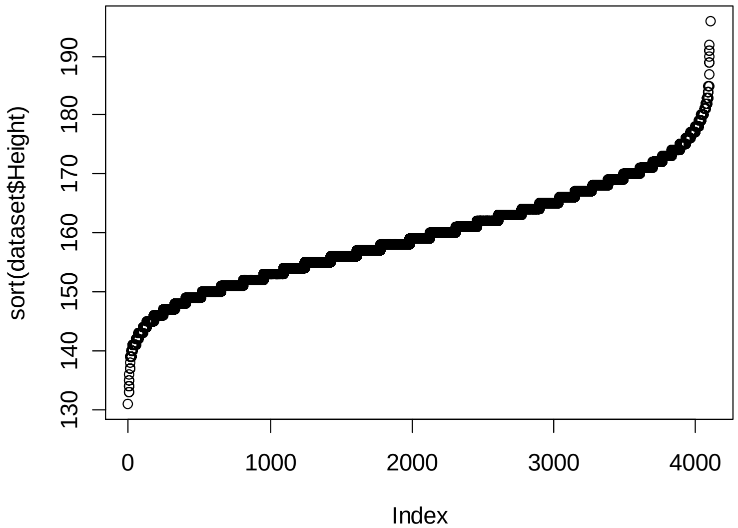
Plotting a Vector

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

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

Plotting a Vector





`plot(dataset$Height)`

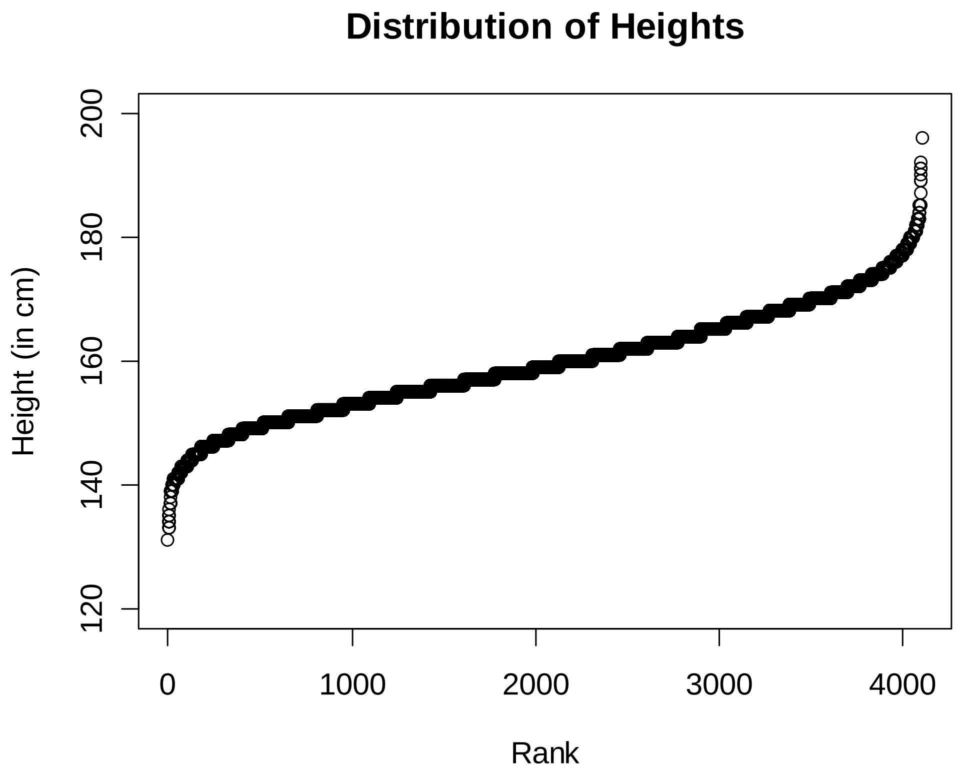
`plot(sort(dataset$Height))`

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

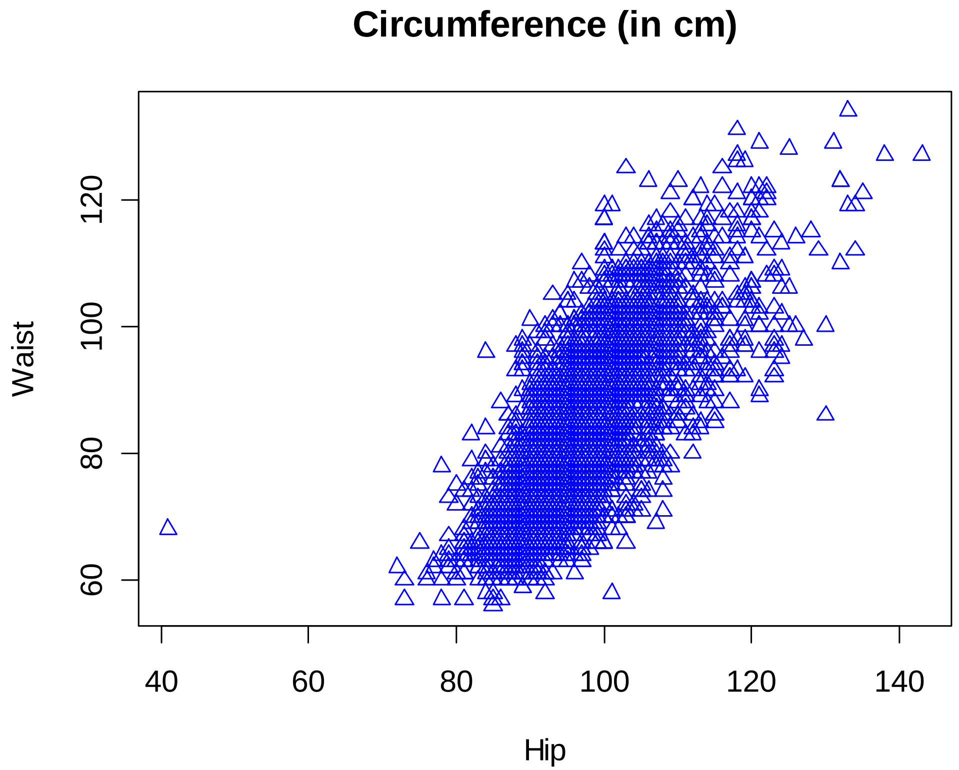



```
plot(sort(dataset$Height), ylim = c(120,200),
ylab = "Height (in cm)", xlab = "Rank", main = "Distribution of Heights")
```

Plotting Two Vectors

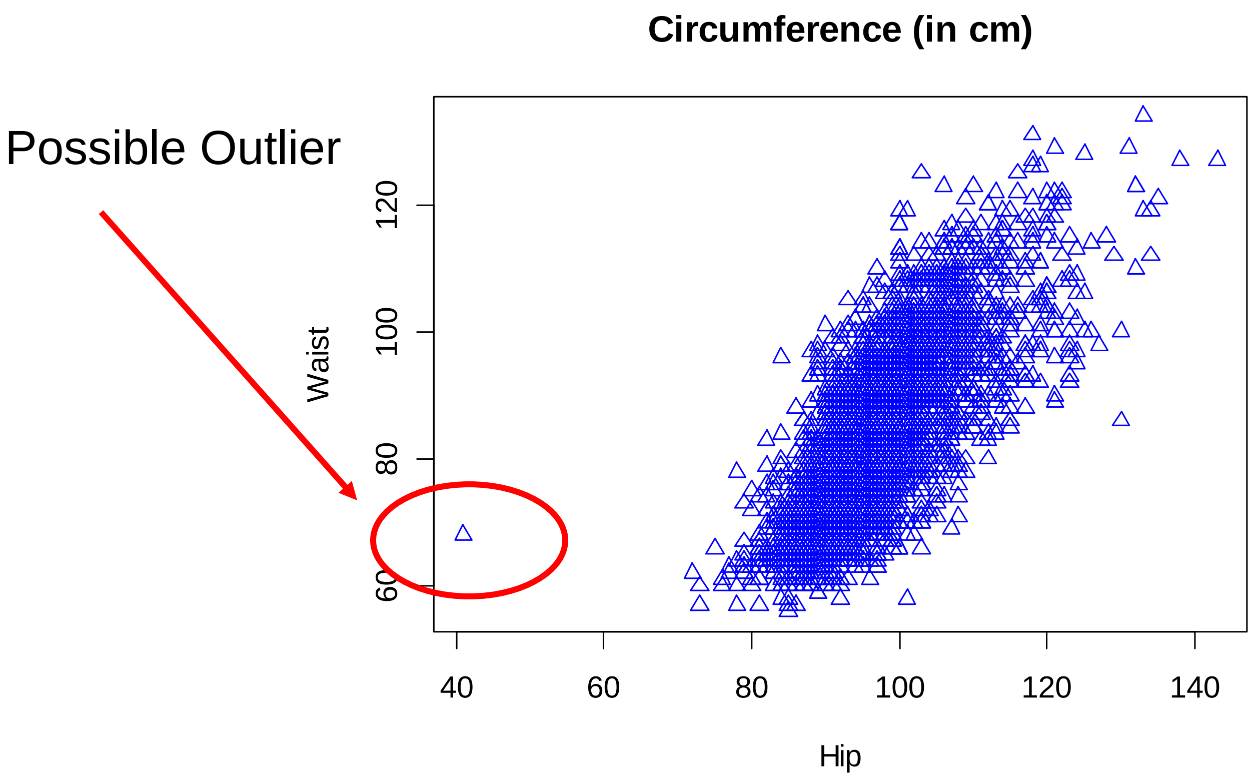
- `plot()` can pair elements from 2 vectors to produce x-y coordinates
- `plot()` and `pairs()` can also produce composite plots that pair all the variables in a data frame.

Plotting Two Vectors




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

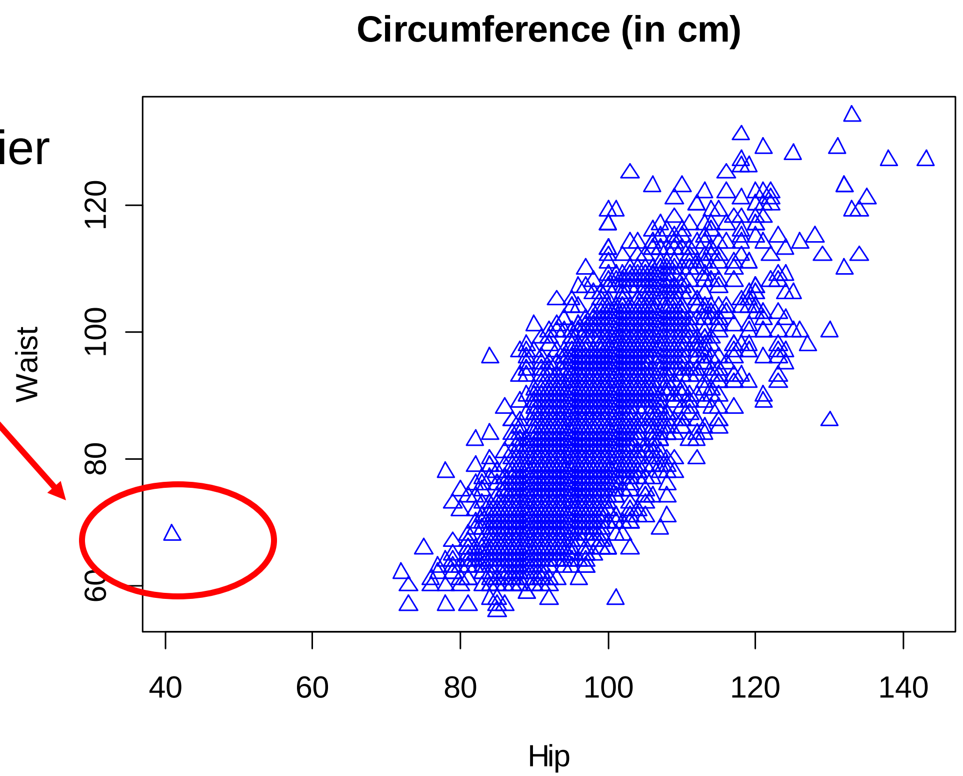
Plotting Two Vectors




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

Plotting Two Vectors

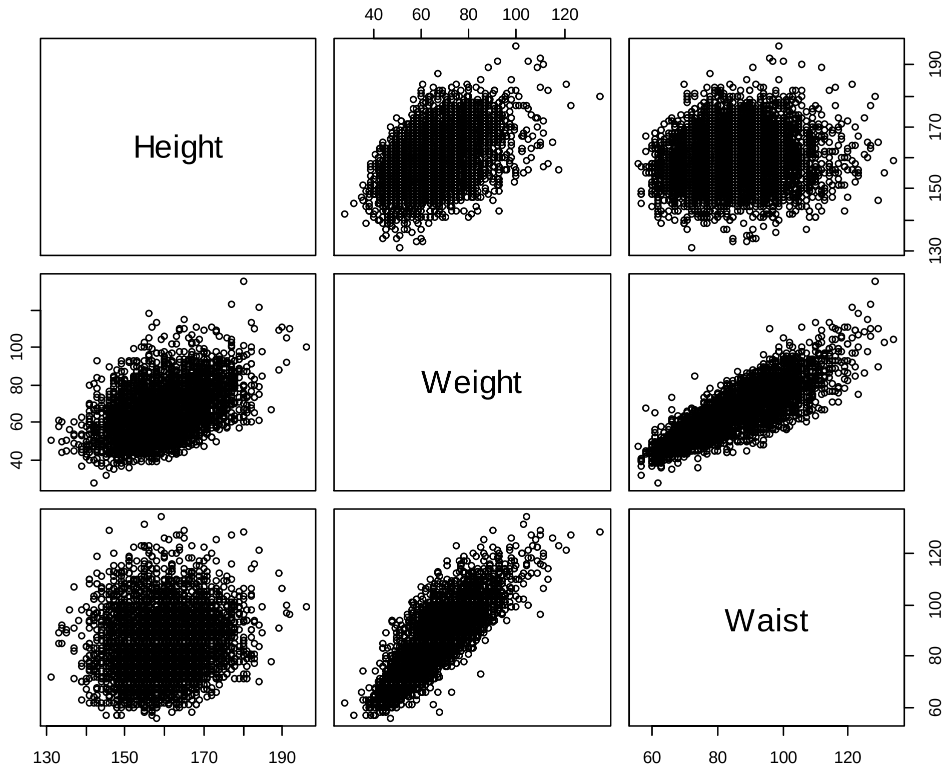
Possible Outlier



These options set the plotting symbol (pch) and line color (col)

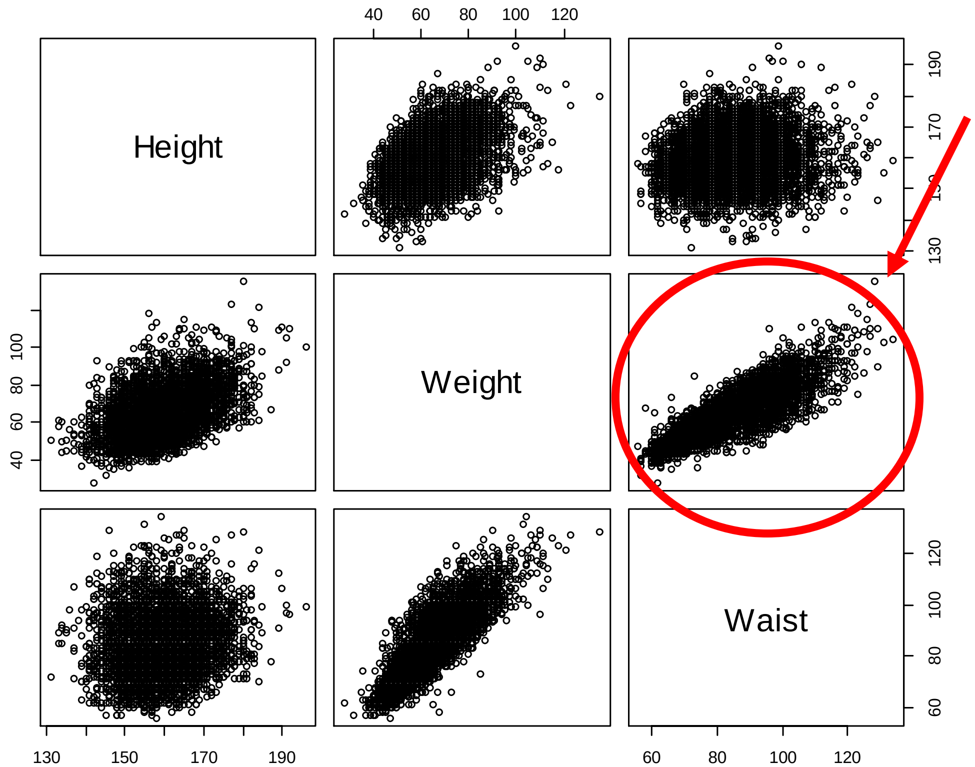
```
plot(dataset$Hip, dataset$Waist,
  xlab = "Hip", ylab = "Waist",
main = "Circumference (in cm)", pch = 2, col = "blue")
```

Plotting Contents of a Dataset




```
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

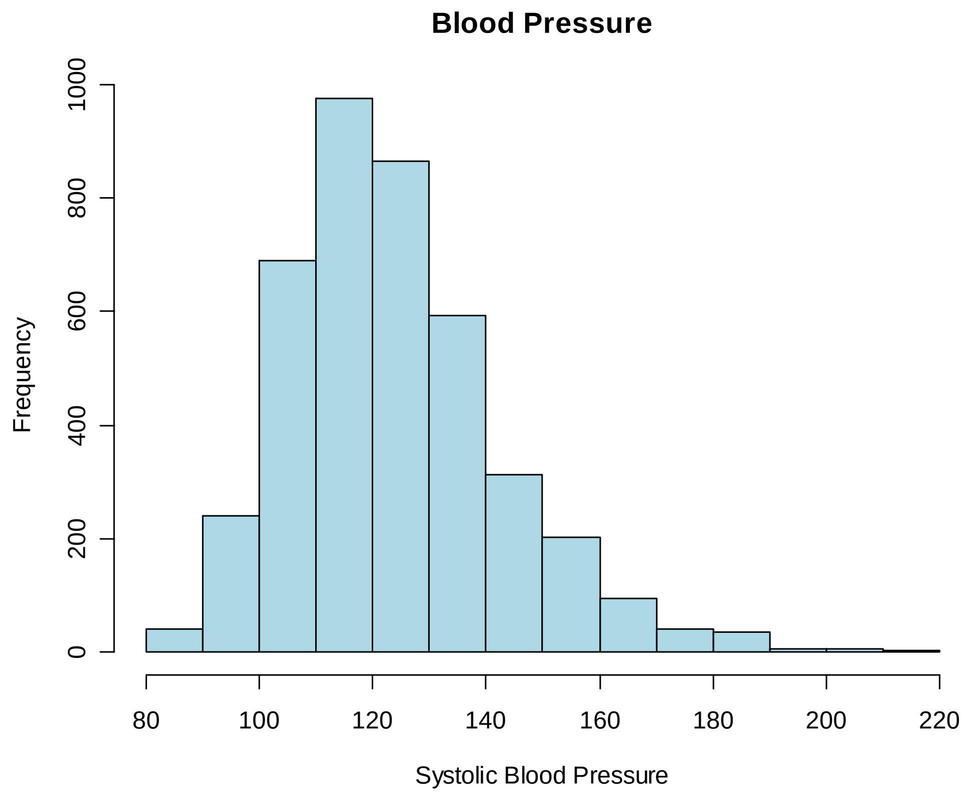
- Generated by the `hist()` function

- The parameter `breaks` is key
  - Specifies the number of categories to plot

  or
  - Specifies the breakpoints for each category

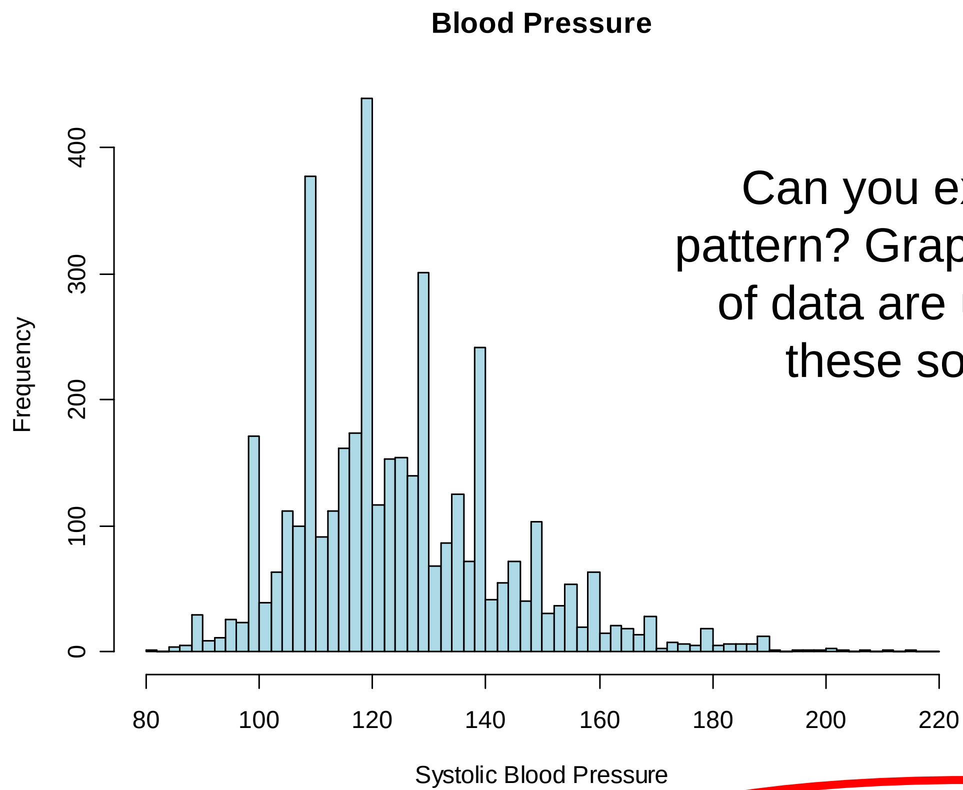
- The `xlab, ylab, xlim, ylim` options work as expected

Histogram




```
hist(dataset$bp.sys, col = "lightblue",
xlab = "Systolic Blood Pressure", main = "Blood Pressure")
```

Histogram , Changed breaks



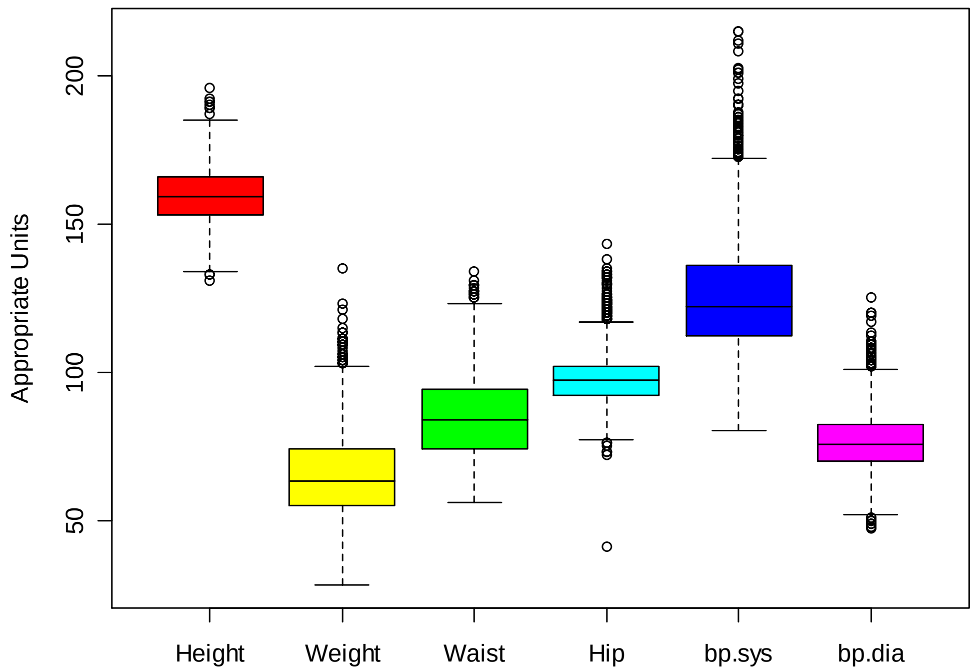
Can you explain the peculiar pattern? Graphical representations of data are useful at identifying these sorts of artifacts…

```
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

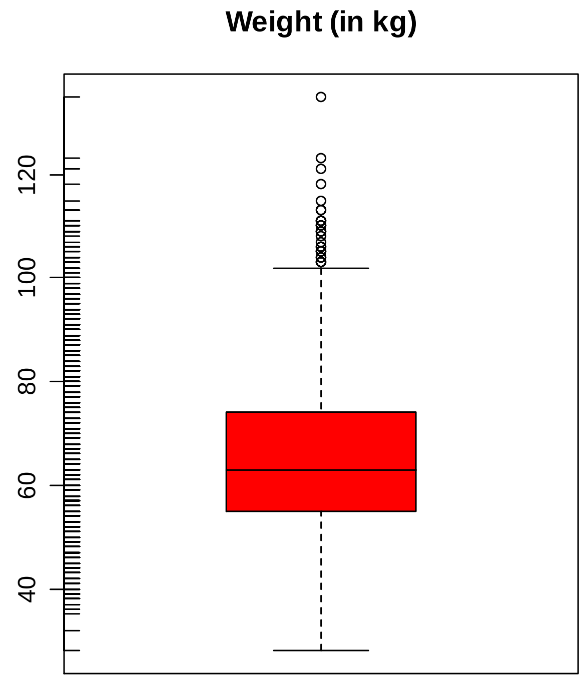
A Simple Boxplot




```
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




```
> 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)`

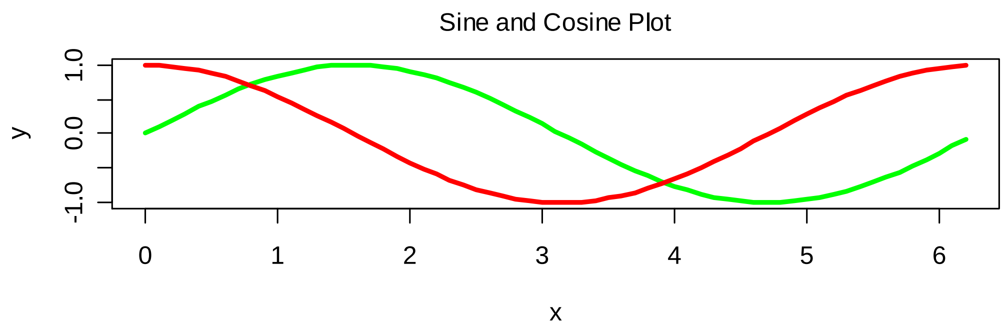
- For freehand drawing use
  - `polygon()`
  - `rect()`

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()`

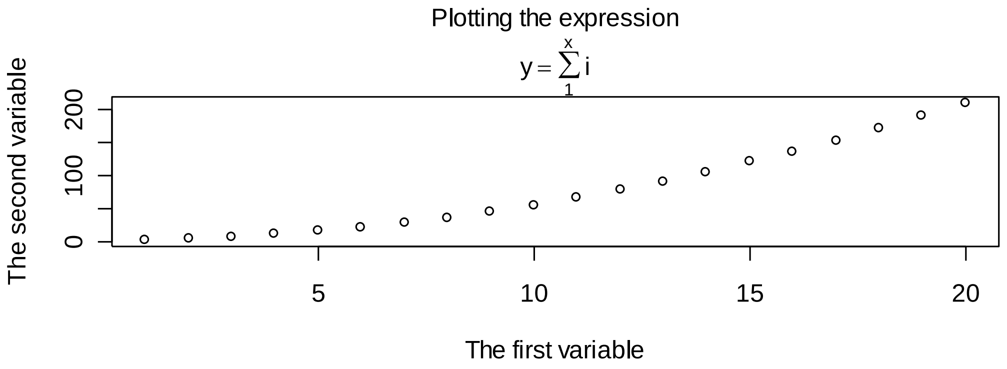
Plotting Two Data Series

```
> 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)
```

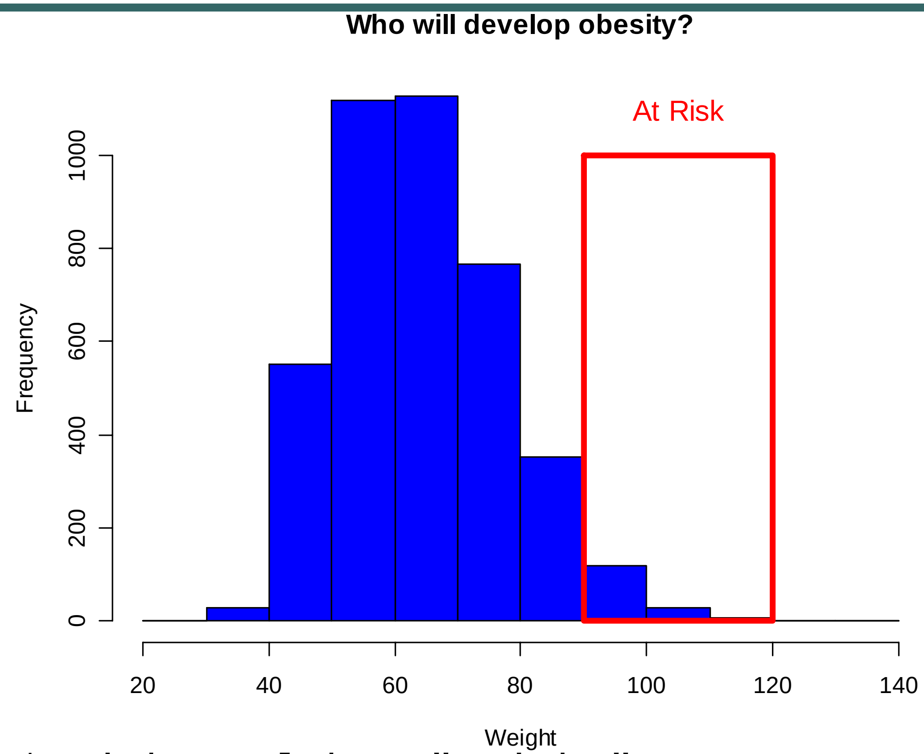


Printing on Margins, Using Symbolic Expressions

```
> 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(i,1,x,i)), side = 3, line = 0)
> mtext("The first variable", side = 1, line = 3)
> mtext("The second variable", side = 2, line = 3)
```



Adding a Label Inside a Plot




```
> 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)

Big Operators	
sum(x[i], i = 1, n)	$\sum_{1}^{n} x_i$
prod(plain(P)(X == x), x)	$\prod_{x} \mathrm{P}(X = x)$
integral(f(x) * dx, a, b)	$\int_a^b f(x)dx$
union(A[i], i == 1, n)	$\bigcup_{i=1}^{n} A_i$
intersect(A[i], i == 1, n)	$\bigcap_{i=1}^{n} A_i$
lim(f(x), x %->% 0)	$\lim_{x \to 0} f(x)$
min(g(x), x >= 0)	$\min_{x \geq 0} g(x)$
inf(S)	$\inf S$
sup(S)	$\sup S$

Further Customization

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

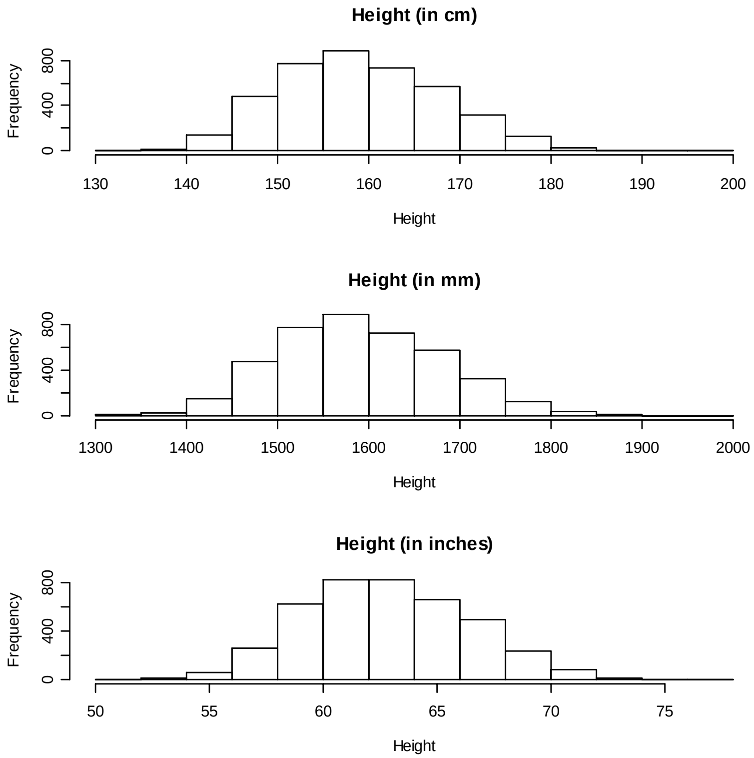
- Parameters include:
  - `cex, mex` – text character and margin size
  - `pch` – plotting character
  - `xlog, ylog` – to select logarithmic axis scaling

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