FIT5149: Applied Data Analysis
Exploratory Data Analysis

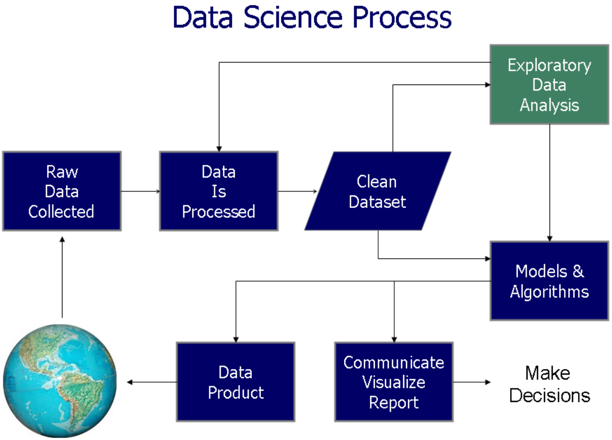
Dr. Lan Du

Faculty of Information Technology, Monash University, Australia

Week 2

1. Summary Statistics

2. Basic graphs



∗ from wikipedia

A variable is any characteristics, number, or quantity that can be measured or counted. There are different types of variables based on the ways they are studied, measured and represented:

- Quantitative variable
  Have values that describe a measurable quantity as a number, like 'how many' or 'how much'. It is meaningful to do arithmetic.
  - **Continuous**
    Observations can take any value between two specified values.
  - **Discrete**
    Observations can take a value based on a count from a set of distinct values
- Qualitative variable
  Have values that describe a 'quality' or 'characteristic' of a data unit, like 'what type' or 'which category'
  - **Ordinal**
    Observations can take a value that can be logically ordered or ranked
  - **Nominal** or Categorical
    Observations can take a value that is not able to be organised in a logical sequence

Data Matrix, Observations and Variables

- **Variables:**
  Age, sex, business income and expenses, country of birth, capital expenditure, class grades, eye colour, vehicle type
- **Continuous** (how much)
  Height, time, age, and temperature
- **Discrete** (how many)
  Number of registered cars, number of business locations, and number of children in a family
- **Ordinal** (has order)
  Academic grades (i.e. A, B, C), clothing size (i.e. small, medium, large, extra large), attitudes (i.e. strongly agree, agree, disagree, strongly disagree), dates
- **Nominal** (no order)
  Gender, business type, eye colour, religion and brand

Variables - Examples

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
2	7129300520	20141013T000000	221900	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
3	6414100192	20141209T000000	538000	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	47.721	-122.319	1690	7639
4	5631500400	20150225T000000	180000	2	1	770	10000	1	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062
5	2487200875	20141209T000000	604000	4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000
6	1954400510	20150218T000000	510000	3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503
7	7237550310	20140512T000000	1225000	4	4.5	5420	101930	1	0	0	3	11	3890	1530	2001	0	98053	47.6561	-122.005	4760	101930
8	1321400060	20140627T000000	257500	3	2.25	1715	6819	2	0	0	3	7	1715	0	1995	0	98003	47.3097	-122.327	2238	6819
9	2008000270	20150115T000000	291850	3	1.5	1060	9711	1	0	0	3	7	1060	0	1963	0	98198	47.4095	-122.315	1650	9711
10	2414600126	20150415T000000	229500	3	1	1780	7470	1	0	0	3	7	1050	730	1960	0	98146	47.5123	-122.337	1780	8113
11	3793500160	20150312T000000	323000	3	2.5	1890	6560	2	0	0	3	7	1890	0	2003	0	98038	47.3684	-122.031	2390	7570
12	1736800520	20150403T000000	662500	3	2.5	3560	9796	1	0	0	3	8	1860	1700	1965	0	98007	47.6007	-122.145	2210	8925
13	9212900260	20140527T000000	468000	2	1	1160	6000	1	0	0	4	7	860	300	1942	0	98115	47.69	-122.292	1330	6000
14	114101516	20140528T000000	310000	3	1	1430	19901	1.5	0	0	4	7	1430	0	1927	0	98028	47.7558	-122.229	1780	12697
15	6054650070	20141007T000000	400000	3	1.75	1370	9680	1	0	0	4	7	1370	0	1977	0	98074	47.6127	-122.045	1370	10208
16	1175000570	20150312T000000	530000	5	2	1810	4850	1.5	0	0	3	7	1810	0	1900	0	98107	47.67	-122.394	1360	4850

```
> sapply(home_data,class)
         id          date       price    bedrooms   bathrooms  sqft_living    sqft_lot
  "numeric"      "factor"   "integer"   "integer"   "numeric"    "integer"   "integer"
     floors    waterfront        view   condition       grade   sqft_above    basement
  "numeric"     "integer"   "integer"   "integer"   "integer"    "integer"   "integer"
   yr_built  yr_renovated     zipcode         lat        long sqft_living15  sqft_lot15
  "integer"     "integer"   "integer"   "numeric"   "numeric"    "integer"   "integer"
```

Variables - Examples

```
> str(home_data)
'data.frame':       21613 obs. of  21 variables:
 $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
 $ date         : Factor w/ 372 levels "20140502T000000",..: 165 221 291...
 $ price        : int  221900 538000 180000 604000 510000 1225000 257500...
 $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
 $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
 $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
 $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
 $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
 $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
 $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
 $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
 $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
 $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
 $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
 $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 ...
 $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
 $ long         : num  -122 -122 -122 -122 -122 ...
```

Have a Look at iris Data

- The iris dataset has been used for classification
- Consists of 50 samples from each of three classes of iris flowers
- Use str() to compactly display the structure of an arbitrary R object

```
> str(iris)
'data.frame':       150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> dim(iris)
[1] 150 5
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

Have a Look at iris Data

- dim() for the dimension of the data,
- names() to obtain names of data

```
> dim(iris)
[1] 150 5
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

Have a Look at iris Data

- `attributes()` returns the attributes of data

```
> attributes(iris)
$names
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
"Species"
$row.names
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
[20] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
[39] 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
[58] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
[77] 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
[96] 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
[115] 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
[134] 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
$class
[1] "data.frame"
```

Have a Look at `iris` Data

- The first or last rows of data can be retrieved with `head()` or `tail()`
- We can also retrieve the values of a single column

```
> head(iris)
> tail(iris)
> iris[1:5,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3.0          1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
4          4.6         3.1          1.5         0.2 setosa
5          5.0         3.6          1.4         0.2 setosa
> iris[1:10, "Sepal.Length"]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
> iris$Sepal.Length[1:10]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

Summary Statistics

Statistics on variables

- Measure of centre
  - Mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

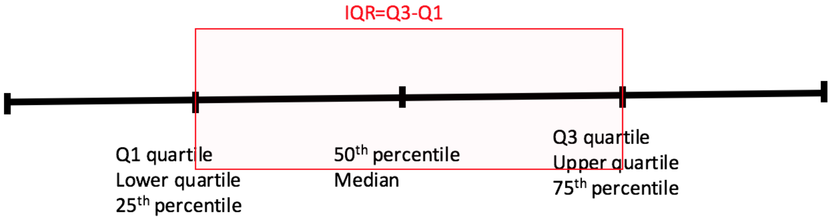
  - Median
    - Order the data
    - Find the mid point or average of two mid points
- Measure of spread
  - Variance

$$\text{var} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

  - Standard deviation $sd = \sqrt{\text{variance}}$
  - Range $= \textbf{max} - \textbf{min}$
  - IQR $= Q_3 - Q_1$
- Robust statistics: extreme observations have little effect
  - Median is more robust than mean
  - IQR more robust than range, variance and std
  - they are better for skewed

Quartiles

- For a sorted data
- Quartiles are 3 points that divide into 4 equal groups
- Each group is a quarter of data
- Lower hinge $= Q1 - 1.5 \times \text{IQR}$
- Upper hinge $= Q3 + 1.5 \times \text{IQR}$



```
> x <- c(0:10, 50)
> xm <- mean(x)
> xm
[1] 8.75
> c(xm, mean(x, trim = 0.10)) #trimmed mean
[1] 8.75 5.50

> median(1:4)
[1] 2.5
> median(1:5)
[1] 3
> median(c(1:3, 100, 1000))
[1] 3
> c(median(1:4), mean(1:4))
[1] 2.5 2.5
> c(median(1:5), mean(1:5))
[1] 3 3
> c(median(c(1:3, 100, 1000)), mean(c(1:3, 100, 1000)))
[1]   3.0 221.2
> var(1:20)
[1] 35
> sd(1:20)
[1] 5.91608
> sqrt(var(1:20))==sd(1:20)
[1] TRUE
> range(3:10)
[1]  3 10
> diff(range(3:10))
[1] 7
```

```
> IQR(c(3:10, 100, 1000))
[1] 4.5
> c(IQR(c(3:10, 100, 1000)), diff(range(c(3:10, 100, 1000))))
[1]   4.5 997.0
> boxplot(c(3:100, 150, 200))
```

Summary of Variables

MONASH University

- Distribution of every variable can be checked with function `summary()`
- It returns the minimum, maximum, mean, median, and the first (25
- For factors (or categorical variables), it shows the frequency of every level.

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> summary(iris$Sepal.Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900
> summary(iris)
  Sepal.Length    Sepal.Width    Petal.Length    Petal.Width        Species
 Min.   :4.300 Min.   :2.000 Min.   :1.000 Min.   :0.100 setosa    :50
 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300 versicolor:50
 Median :5.800 Median :3.000 Median :4.350 Median :1.300 virginica :50
 Mean   :5.843 Mean   :3.057 Mean   :3.758 Mean   :1.199
 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800
 Max.   :7.900 Max.   :4.400 Max.   :6.900 Max.   :2.500
```

- The mean, median and range can also be obtained with functions with `mean()`, `median()` and `range()`

Summary of Variables

- Quartiles and percentiles are supported by function `quantile()`
- Use `var()` to check variance

```
> quantile(iris$Sepal.Length)
0% 25% 50% 75% 100%
4.3 5.1 5.8 6.4 7.9
> quantile(iris$Sepal.Length, c(.1, .3, .65))
10% 30% 65%
4.80 5.27 6.20
> var(iris$Sepal.Length)
[1] 0.6856935
```

Further Summary of Variables

```
> summary(cars)
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
> fivenum(cars$speed) # min, lower hinge, median, upper-hinge, max.
[1]  4 12 15 19 25
> boxplot.stats(cars$speed) # Boxplot stats: hinges, n, CI of the median, outliers
$stats
[1]  4 12 15 19 25

$n
[1] 50

$conf
[1] 13.43588 16.56412

$out
numeric(0)
```

Basic graphs

Graphical Representations

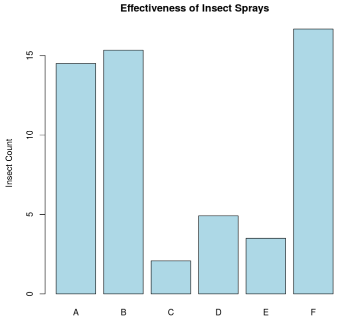
- To understand the properties of data
- To find possible pattern in data
- To guide us in choosing better and more suitable models
- To communicate the outcome with others

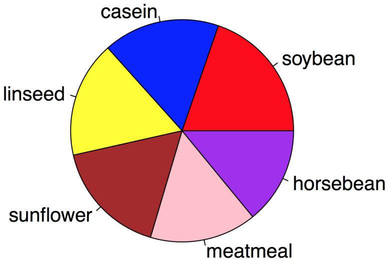
	Single variable	Two variables	Categorical
Numerical	Histogram Box plot	Scatter plot	Side-by-side box plot
Categorical	Pie chart Bar plot	segmented bar plot Mosaic plot	

Table: Data exploration choices

Single Categorical: Bar Chart and Pie Charts

- Visual presentation of categorical data
- Bar charts show a quantitative value related to a categorical variable.
- The length of the bar is proportional to the value they represent
- The bars could be rearranged in any order
- Pie charts show the relative contribution.
- Slices to illustrate numerical proportion

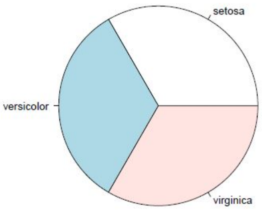


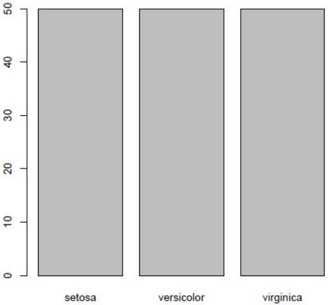


Frequency

- The frequency of factors can be calculated with function `table()`,
- Plotted as a pie chart with `pie()` or a bar chart with `barplot()`.

```
> table(iris$Species)
setosa versicolor virginica
    50         50        50
> pie(table(iris$Species))
> barplot(table(iris$Species))
```





Pie Chart

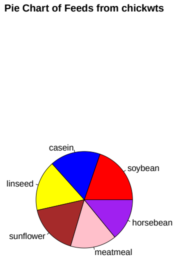
```
> str(chickwts)
'data.frame':        71 obs. of  2 variables:
 $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
 $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 ...
> feeds <- table(chickwts$feed)
> pie(feeds[order(feeds, decreasing = TRUE)],
+     col = c("red", "blue", "yellow", "brown", "pink", "purple"),
+     main = "Pie Chart of Feeds from chickwts")
> feeds

   casein horsebean   linseed  meatmeal   soybean sunflower
       12        10        12        11        14        12
> prop.table(feeds)

   casein horsebean   linseed  meatmeal   soybean sunflower
0.1690141 0.1408451 0.1690141 0.1549296 0.1971831 0.1690141
> round(prop.table(feeds), 2)

   casein horsebean   linseed  meatmeal   soybean sunflower
     0.17      0.14      0.17      0.15      0.20      0.17
> round(prop.table(feeds), 2) * 100

   casein horsebean   linseed  meatmeal   soybean sunflower
       17        14        17        15        20        17
```



Single Numerical: Histogram

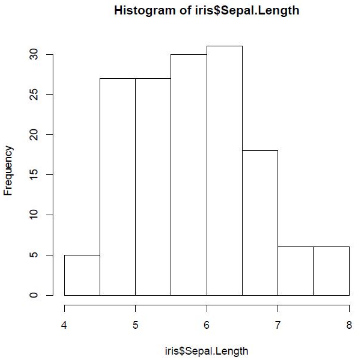
- A histogram is a graphical representation of the distribution of numerical data
- Shape of the distribution of a single numeric variable
  - Bins, height representing the frequency or percentage frequency (relative)
  - First step: bin the values (divide the entire range of values into a series of intervals)
  - the bins represent ranges of data,
  - The length of the bar is proportional to the frequency of each bin
- You cannot rearrange bins!
- Normalised histogram for relative frequencies
- A bar chart is a plot of categorical variables; it is better to put gaps between bars in a bar chart
  - show the frequency of data values for a single variable

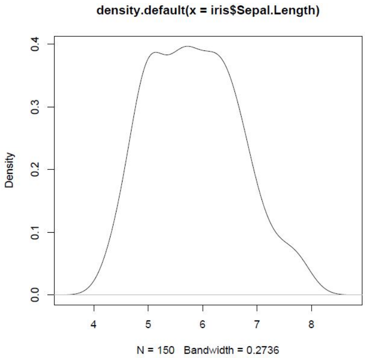
Explore Individual Variables

MONASH University

- Check variable's distribution with histogram and density using functions `hist()` and `density()`

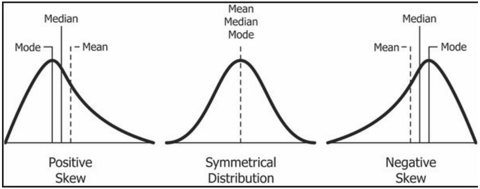
```
> hist(iris$Sepal.Length)
> plot(density(iris$Sepal.Length))
```

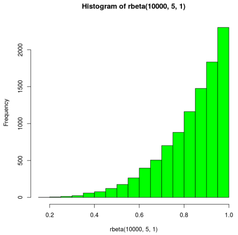


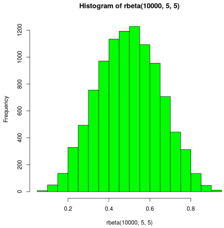


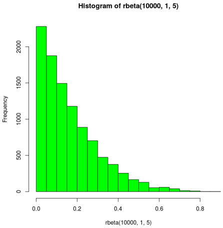
Visual Attributes of Distributions

- Shape of skewness
  - Symmetric
  - Left skewed
  - Right Skewed



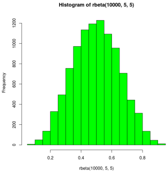


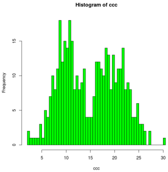


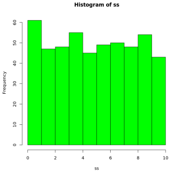


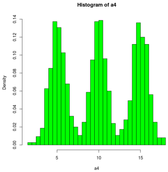
Visual Attributes of Distributions

- Modality
  - Unimodal
  - Bimodal
  - Uniform
  - Multimodal



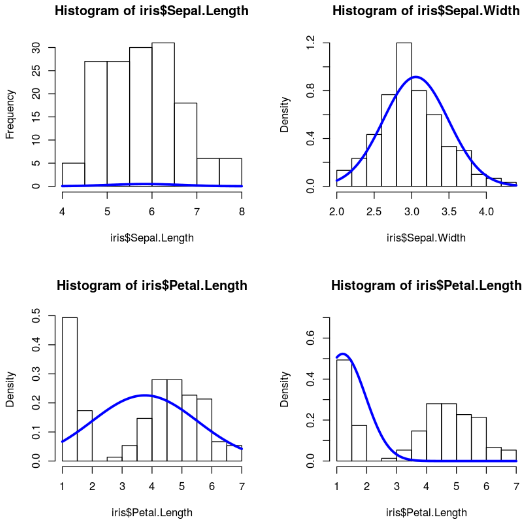






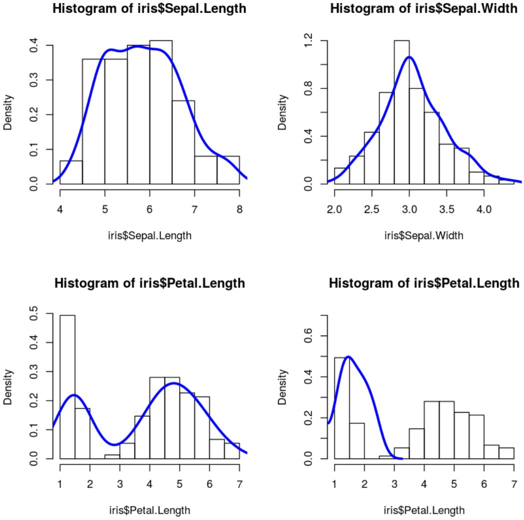
Histogram and Density

curve(dnorm(x, mean = mean(iris$Petal.Width), sd = sd(iris$Petal.Width)), add = TRUE)



Histogram and Density

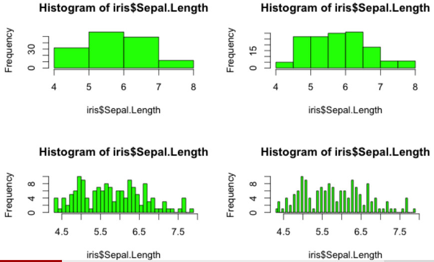
lines(density(iris$Petal.Width), col = "blue", lwd = 3)



Visual Attributes of Distributions

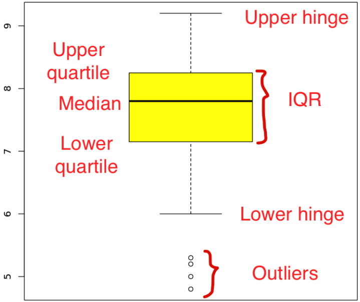
- The issue of bin size
  - Wider bin size: lose some interesting information
  - Narrower: difficult to see overall picture
  - Right size: it depends on data

```
> oldpar <- par()
> par(mfrow=c(2,2))
> hist(iris$Sepal.Length, breaks=5, col="green")
> hist(iris$Sepal.Length,breaks=10, col="green")
> hist(iris$Sepal.Length,breaks=50, col="green")
> hist(iris$Sepal.Length, breaks=100, col="green")
> par(oldpar)
```



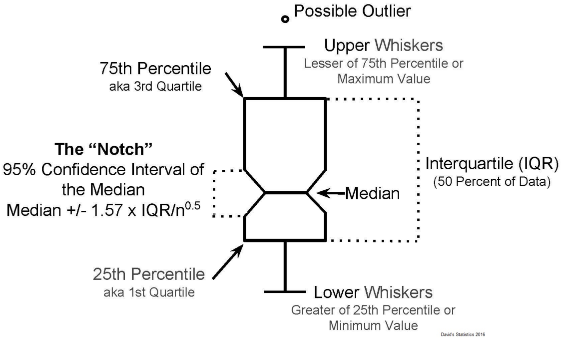
Boxplot

- Represent numerical data through their quartiles
- Whiskers indicating variability outside the upper and lower quartiles
- Highlights outliers
- Shows median and IQR $= Q_3 - Q_1$
- Lower hinge $= Q1 - 1.5 \times \text{IQR}$
- Upper hinge $= Q3 + 1.5 \times \text{IQR}$



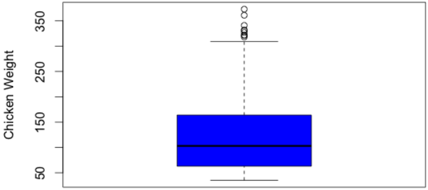
Boxplot

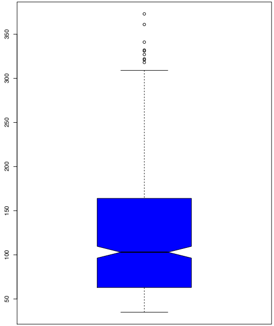
- Represent numerical data through their quartiles
- Whiskers indicating variability outside the upper and lower quartiles
- Highlights outliers
- Shows median and IQR $= Q_3 - Q_1$
- Lower hinge $= Q1 - 1.5 \times \text{IQR}$
- Upper hinge $= Q3 + 1.5 \times \text{IQR}$



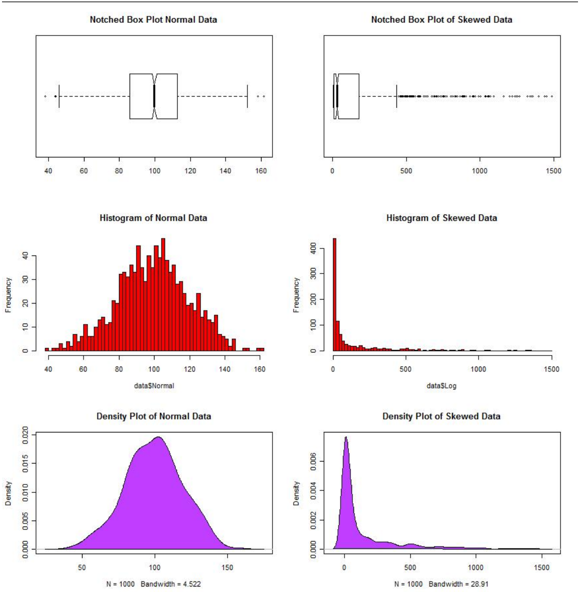
Boxplot

```
> summary(ChickWeight$weight)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   35.0    63.0   103.0   121.8   163.8   373.0
> boxplot.stats(ChickWeight$weight)
$stats  [1]  35  63 103 164 309
$n  [1] 578
$conf  [1]  96.36235 109.63765
$out  [1] 318 331 327 341 332 361 373 321 322
> data("ChickWeight")
> boxplot(ChickWeight$weight, col="blue")
```

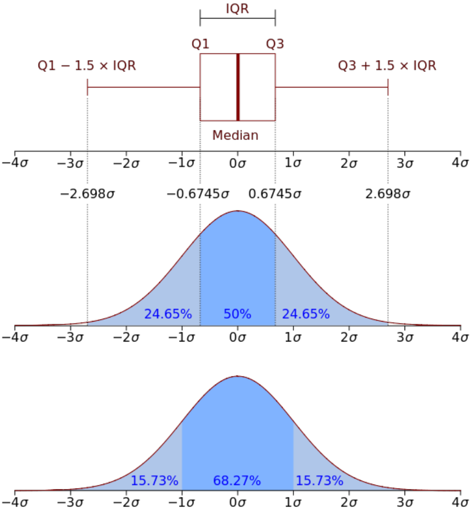




Boxplot and Density



Boxplot and Normal Density



Graphical Representations

	Single variable	Two variables	Categorical
Numerical	Histogram Box plot	Scatter plot	Side-by-side box plot
Categorical	Pie chart Bar plot	segmented bar plot Mosaic plot	

Table: Data exploration choices

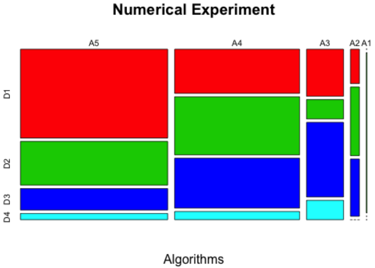
Two Categorical Variable

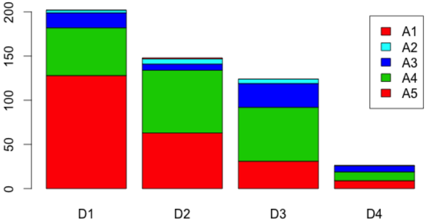
- There 500 problems with different levels of difficulties D_1, D_2, D_3, D_4
- We have 5 algorithms A_5, A_4, A_3, A_2, A_1
- The first column of the table shows
  - We have 202 problem of difficulty level 1
  - Algorithm $A5$ could solve 128
  - Algorithm $A1$ could not solve any of them
- First row shows
  - Algorithm $A1$ could solve 128 of problems of difficulty level D_1

	D1	D2	D3	D4	row sum
A5	128	63	31	9	231
A4	54	71	61	10	196
A3	17	7	27	7	58
A2	3	6	5	0	14
A1	0	1	0	0	1
column sum	202	148	124	26	500

Table: Frequency matrix for algorithms applied on problems

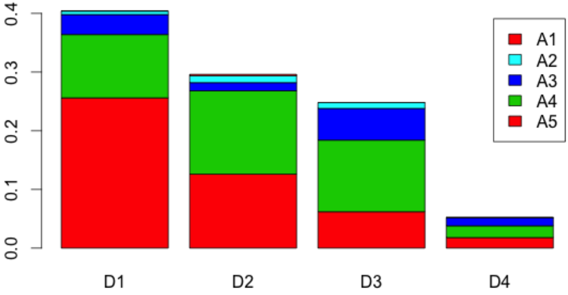
```
> freqt <- matrix(c(128,63,31,9,54,71,61,10,17,7,27,7,3,6,5,0,0,1,0,0), 5,4, byrow=TRUE)
> freqt
     [,1] [,2] [,3] [,4]
[1,]  128   63   31    9
[2,]   54   71   61   10
[3,]   17    7   27    7
[4,]    3    6    5    0
[5,]    0    1    0    0
> dimnames(freqt) = list( c("A5", "A4", "A3", "A2", "A1"), c("D1", "D2", "D3", "D4"))
> freqt
    D1 D2 D3 D4
A5 128 63 31  9
A4  54 71 61 10
A3  17  7 27  7
A2   3  6  5  0
A1   0  1  0  0
> rowSums(freqt); sum(rowSums(freqt))
 A5  A4  A3  A2  A1
231 196  58  14   1
[1] 500
> colSums(freqt); sum(colSums(freqt))
 D1  D2  D3  D4
202 148 124  26
[1] 500
> mosaicplot(freqt,main="Numerical Experiment",sub="Algorithms",col=c(2,3,4,5))
> barplot(freqt, col=c(2,3,4,5), legend=TRUE)
```





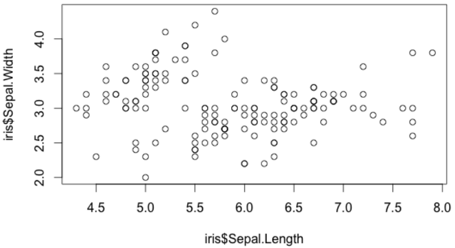
(http://www.pmean.com/definitions/mosaic.htm)

```
> pfreqt<- prop.table(freqt)
> pfreqt
      D1    D2    D3    D4
A5 0.256 0.126 0.062 0.018
A4 0.108 0.142 0.122 0.020
A3 0.034 0.014 0.054 0.014
A2 0.006 0.012 0.010 0.000
A1 0.000 0.002 0.000 0.000
> barplot(pfreqt,  col=c(2,3,4,5), legend=TRUE)
```



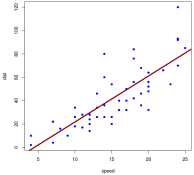
Two Numerical: Scatter Plots

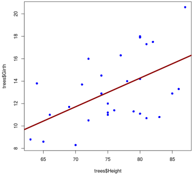
- **Scatter plots** are one of the most widely used statistical graphics.
  - They show two numerical variables
  - can reveal linear or nonlinear relationships between the variables,
  - shows correlations between the variables and
  - the presence of extreme values (outliers).
- Explanatory variable (*x*-axis) and response variable (*y*-axis)
- To see the possible relationship between two numerical variables
- Visualising a line or a curve through the cloud of points

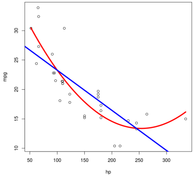


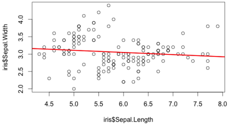
Relationship Between Two Numerical Variable

- Direction of relationship
  - Increasing
  - Decreasing
- Shape
  - Linear
  - Nonlinear
- Strength
  - Strong
  - Weak





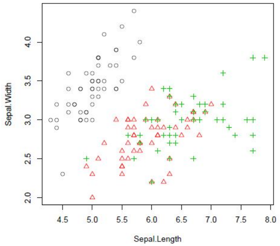


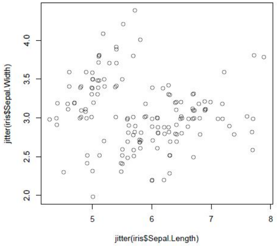


Exploring two numerical variables

- A scatter plot can be drawn for two numeric variables with `plot()`
- We can use `jitter()` to add a small amount of noise to the data before plotting, when there are many overlapping points

```
> with(iris, plot(Sepal.Length, Sepal.Width, col=Species, pch=as.numeric(Species)))
> plot(jitter(iris$Sepal.Length), jitter(iris$Sepal.Width))
```





Explore Multiple Variables

- Investigate the relationships between two variables
- Covariance and correlation between variables with `cov()` and `cor()`

```
> cov(iris$Sepal.Length, iris$Petal.Length)
[1] 1.274315
> cov(iris[,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
> cor(iris$Sepal.Length, iris$Petal.Length)
[1] 0.8717538
> cor(iris[,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Correlation, Variance and Covariance (Matrices)

- `cor()` for correlation, `cov()` for covariance
- `var`, `cov` and `cor` compute the variance of x and the covariance or correlation of x and y if these are vectors.
- If x and y are matrices then the covariances (or correlations) between the columns of x and the columns of y are computed.
- Covariance shows how much two random variables change together

$$cov(X, Y) = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

$$cov(X, X) = var(X)$$

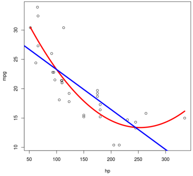
- Correlation shows the dependency between two random variable

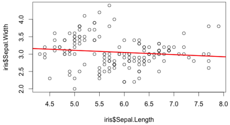
$$\rho_{X,Y} = corr(X, Y) = \frac{cov(X, Y)}{\sigma_x \sigma_y}$$

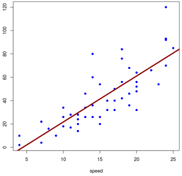
Covariance and Correlation

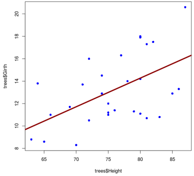
MONASH University

- Covariance of two random variables shows how they are related
- Positive covariance, then they are positively related
- Negative covariance, then they are negatively related
- The correlation coefficient of two random variable is covariance divided by the product of their standard deviations
- it shows how the two random variable are linearly related
- If the correlation is close to 1, then they are positively linearly related
- If the correlation is close to -1, then they are negatively linearly related
- If the correlation is close to 0, then they are weakly related





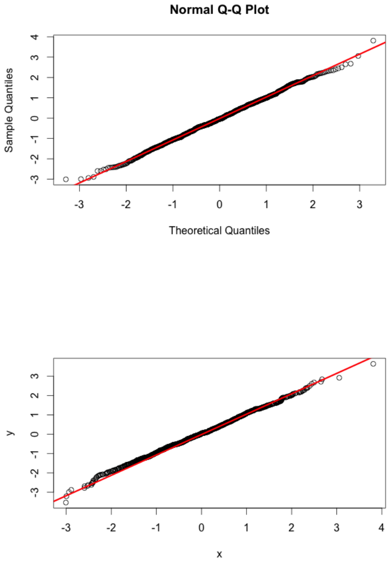




Checking Distribution Similarity: qqplot

- quantile-quantile (q-q) plot is for determining if two data sets have similar populations with a common distribution
- plot of the quantiles of the first data set against the quantiles of the second data set
- the fraction (or percent) of points below the given value
- the 0.4 (or 40%) quantile is the point at which 40% percent of the data fall below and 60% fall above that value

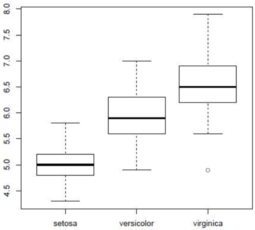
```
> x<- rnorm(1000)
> y<- rnorm(2000)
> z<- runif(500)
> qqnorm(x) ; qqline(x, col="red", lwd=3)
> qqnorm(z) ; qqline(x, col="red", lwd=3)
> qqplot(x, y, plot.it = TRUE); qqline(x, col="red", lwd=3)
> qqplot(x, z, plot.it = TRUE); qqline(x, col="red", lwd=3)
```


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
y
x
z
x

Exploring categorical against numerical variables

- Compute the stats of Sepal.Length of every Species with `aggregate()`

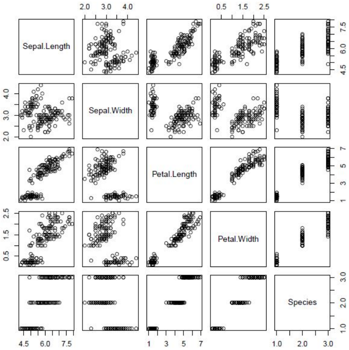
```
> aggregate(Sepal.Length ~ Species, summary, data=iris)
     Species Sepal.Length.Min. Sepal.Length.1st Qu. Sepal.Length.Median
1     setosa             4.300                4.800               5.000
2 versicolor 4.900 5.600 5.900
3  virginica 4.900 6.225 6.500
  Sepal.Length.Mean Sepal.Length.3rd Qu. Sepal.Length.Max.
1             5.006                5.200             5.800
2             5.936                6.300             7.000
3             6.588                6.900             7.900
> boxplot(Sepal.Length~Species, data=iris)
```



Multivariate Scatter Plots

- A matrix of scatter plots can be produced with function `pairs()`

```
> pairs(iris)
```



Data Exploration Recipie

1. Find variables and decide if they are numerical or categorical. `str()`, `attributes()`
   - Numerical: Continuous and Discrete
   - Categorical: Ordinal and Nominal
2. Find statistics of each variable
   - Quantitative:Find `summary()`, `fivenum()`, `boxplot.stats()`
   - Qualitative: Find frequencies. `table()` or `prob.table()`
3. Then perform pictorial representation of each single variable
   - Quantitative: Histograms or box plots. `hist()`, `boxplot()`
   - Qualitative: Bar chart or pie charts. `plot()`, `barplot()`, `pie()`
4. Be aware of outliers and robust statistics
5. Association between variables
   - `scatterplot()` to compare two numerical variables
   - Side-by-side boxplots for categorical and numerical variables
   - `pairs()` for a matrix of scatter plots of all variables
   - `cor()`, `cov()` for correlation and covariance between variables

Base Graphics

- `plot(x,y)` or `hist(x)` will launch a graphics device
- par has all the parameters to change the output
- some important parameters:
  - `xlab` for the x-axis label `xlab="weight"`
  - `ylab` for the y-axis label `ylab="error"`
  - `pch`: the plotting symbol (default is open circle)
  - `lty`: the line type to be dashed, dotted, etc., (default is solid line)
  - `lwd`: the line width, lwd=3
  - `col`: the plotting color, col="green"
  - `main`: for the main title `main="the plot of weight and height"`
  - `bg` changes the background color
  - `mar` changes the margin size
  - `mfrow` is the number of plots in each (row, column)
    par(mfrow=c(2,3)) 2 rows, and 3 coluns in each row. The plots are filled row-wise
  - `mfcol` is the number of plots per (row, column)

Plotting Functions

- `plot` creates a scatterplot, or other type depending on the data
- `lines` adds line to a plot
- `points` add points to an exisiting plot
- `title` adds a title
- `legend` to add legends
  `legend("topleft", col = c("green", "yellow"), pch = 5, legend = c("2012", "Before"))`
- `abline`: This function adds one or more straight lines through the current plot
- it is better to save the defult parametrs before any change

```
> par() shows current settings
> oldpar <- par() makes a copy of current settings
> par(oldpar)   brings back original settings, neglect possible warnings!
```

Save Charts into Files

- You can save generated charts as PDF and PS with `pdf()` and `postscript()`
- Picture files of BMP, JPEG, PNG and TIFF formats can be generated respectively with `bmp()`, `jpeg()`, `png()` and `tiff()`
- the files (or graphics devices) need be closed with `graphics.off()` or `dev.off()` after plotting

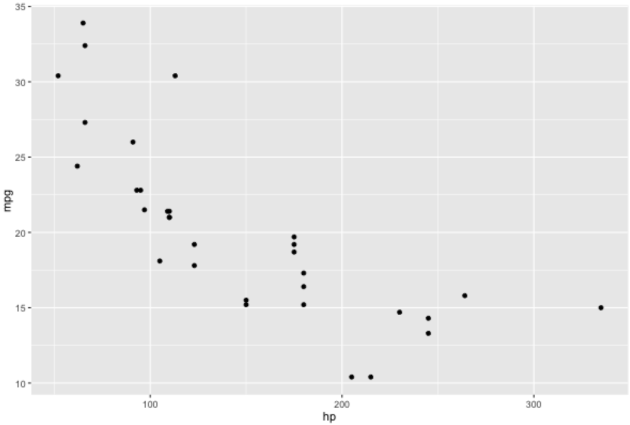
```
> #save as a PDF file
> pdf("myPlot.pdf")
> x <- 1:50
> plot(x, log(x))
> graphics.off()
>
>  #save as a postscript file
> postscript("myPlot2.ps")
> x <- -20:20
> plot(x, x^2)
> graphics.off()
```

Plotting Systems

- Base plotting system
- It has different layers, and you can add one layer over another
  - What we had so far
- Latice plotting system
  - You need to use the package lattice: require("lattice")
  - You need to insert many information to the function
- ggplot2 system
  - You need to use the package ggplot2: require("ggplot2")
  - Between base and lattice systems

Example

```
> require(ggplot2)
Loading required package: ggplot2
Warning message:
package 'ggplot2' was built under R version 3.2.4
> qplot(hp, mpg, data=mtcars)
```

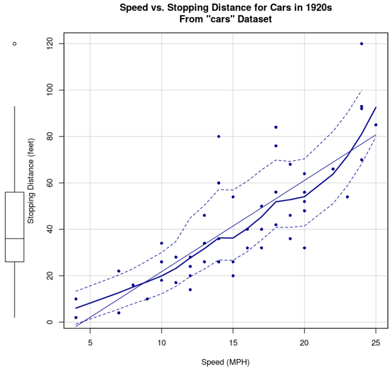


Final Word

- Using graphical devices for making charts is a huge topic
- However, you know enough to do a proper explorations
- If you like to learn more check the following websites:
  - The R Graph Gallery
    `http://www.r-graph-gallery.com/`
  - R Bloggers
    `https://www.r-bloggers.com/`
- Some complicated graphs are here

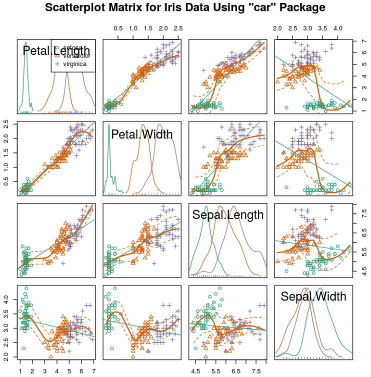
Example 1

```
require(car)
# "scatterplot" has marginal boxplots, smoothers, and quantile regression intervals
scatterplot(cars$dist ~ cars$speed,
            pch = 16,
            col = "darkblue",
            main = "Speed vs. Stopping Distance for Cars in 1920s From čars¨ Dataset",
            xlab = "Speed (MPH)",
            ylab = "Stopping Distance (feet)")
```

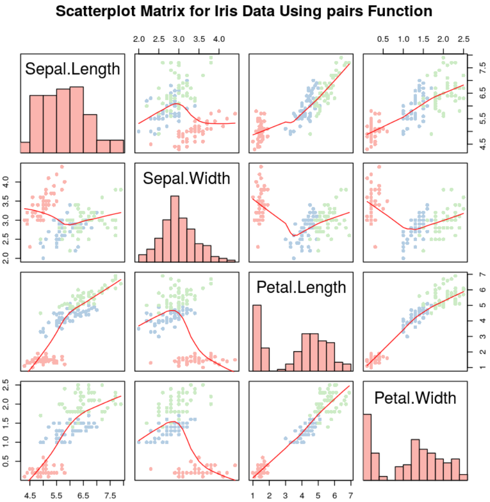


Example 2

```
# Gives kernel density and run plot for each variable
#library(car)
scatterplotMatrix(~Petal.Length + Petal.Width + Sepal.Length + Sepal.Width | Species,
                  data = iris,
                  col = brewer.pal(3, "Dark2"),
                  main="Scatterplot Matrix for Iris Data Using ¨car¨ Package")
```



Example 3


Scatterplot Matrix for Iris Data Using pairs Function
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 5.5 6.5 7.5
1 2 3 4 5 6 7

The R code

```
> require("RColorBrewer")
Loading required package: RColorBrewer
> display.brewer.pal(3, "Pastel1")
> panel.hist <- function(x, ...){
usr <- par("usr"); on.exit(par(usr))
par(usr = c(usr[1:2], 0, 1.5) )
h <- hist(x, plot = FALSE)
breaks <- h$breaks; nB <- length(breaks)
y <- h$counts; y <- y/max(y)
 rect(breaks[-nB], 0, breaks[-1], y,  ...)
# Removed "col = "cyan" from code block; original below
 # rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
 }
> pairs(iris[1:4],
       panel = panel.smooth,  # Optional smoother
       main = "Scatterplot Matrix for Iris Data Using pairs Function",
       diag.panel = panel.hist,
       pch = 16,
       col = brewer.pal(3, "Pastel1")[unclass(iris$Species)])
```