R Course

Data Object Handling

Overview

1. How to examine and summarise data objects (vectors, matrices, scalars or data frame)
2. Identify the class of whole objects or individual columns
3. Identify missing data points and cope with them
4. Performing simple statistics
5. Perform simple visual summaries

Examining Data Objects

```
> dim(pizza.frame)
      [1] 8 4
> is(pizza.frame)
      [1] "data.frame" "oldClass“
> str(pizza.frame)
      `data.frame':   8 obs. of  4 variables:
      $ weight     : num  50 60 70 80 90 100 110 120
      $ pizza.week : num  5 6 6 5 7 8 10 10
      $ metabolism : num  100 90 80 70 70 70 70 50
      $ north.south: Factor w/ 2 levels "0","1": 1 1 2 1 2 2 2 2
```

Are the variables of the class you assume they are ?

```
> colnames(pizza.frame)
      [1] "weight"      "pizza/week"  "metabolism"  "north/south"
> rownames(pizza.frame)
      [1] "1" "2" "3" "4" "5" "6" "7" "8“
> rownames(pizza.frame) <- c(“Cathy”, ”Elke”, ”Heena”, ”Mike”, ”Jose”, ”Lin”, ”Matt”, ”Leo”)
> rownames(pizza.frame)
      [1] "Cathy" "Elke"  "Heena" "Mike"  "Jose"  "Lin"   "Matt"  "Leo"
```

Examining Data Objects

```
> pizza.frame[,1:2]
        weight pizza/week
Cathy     50          5
Elke      60          6
Heena     70          6
Mike      80          5
Jose      90          7
Lin      100          8
Matt     110         10
Leo      120         10
```

This is a simple method, but becomes unwieldy as the matrix becomes huge

```
> fix(pizza.frame)
> pizza.frame <- fix(pizza.frame)
```

Both these methods bring up a simple data editor – be warned it is very simple and there is no undo function

As a habit keep a backup of you master data object somewhere (either in active level or saved onto directory)

Missing Data: Where and What

Missing data can occur for lots of reasons
- Actual missing data (random or structured)
- Data handling error

NA is R's missing code (stands for *not available*)

NaN is produced by certain computational results – e.g. 0/0 or Inf/Inf both give NaN (*Not a Number*)

```
> whoops <- c("10","20","20a")
> whoops
[1] "10"  "20"  "20a"
> as.numeric(whoops)
[1] 10 20 NA
Warning message:
NAs introduced by coercion
> table(pizza.frame[1:2])
     pizza.week
weight 5 6 7 8 10
   50  1 0 0 0  0
   60  0 1 0 0  0
   70  0 1 0 0  0
   80  1 0 0 0  0
   90  0 0 1 0  0
  100 0 0 0 1  0
  110 0 0 0 0  1
  120 0 0 0 0  1
```

```
> table(pizza.frame[1:2])

     pizza.week
weight 5 6 7 8 10
  60  0 1 0 0  0
  70  0 1 0 0  0
  80  1 0 0 0  0
  90  0 0 1 0  0
  100 0 0 0 1  0
  110 0 0 0 0  1
  120 0 0 0 0  1

> table(pizza.frame[1:2],exclude=c())
     pizza.week
weight 5 6 7 8 10
  60   0 1 0 0  0
  70   0 1 0 0  0
  80   1 0 0 0  0
  90   0 0 1 0  0
  100  0 0 0 1  0
  110  0 0 0 0  1
  120  0 0 0 0  1
  <NA> 1 0 0 0  0
```

Missing Data: Where and What

Most methods have ‘ways’ of coping with missing data

Nearly all of them default to removing the rows with missing data.

Generally you want fill in that data by various methods (e.g. mean filling, random, prediction by linear prediction or using EM algorithm)

To do this you need to replace the missing data, how?

```
> pizza.frame[is.na(pizza.frame)]
[1] NA

> pizza.frame[is.na(pizza.frame)] <- 50
> pizza.frame[,1]
[1]  50  60  70  80  90 100 110 120
```

A lot of R is about indexing so it’s good thoroughly understand how it works

The is.na() function doesn’t identify where the missing data point is though

```
> which(is.na(pizza.frame[,1]))
[1] 1
```

You might be tempted to do the following but it doesn’t do what you think it should

```
> pizza.frame==NA
```

	weight	pizza.week	metabolism	north.south
Cathy	NA	NA	NA	NA
Elke	NA	NA	NA	NA
Heena	NA	NA	NA	NA
Mike	NA	NA	NA	NA
Jose	NA	NA	NA	NA
Lin	NA	NA	NA	NA
Matt	NA	NA	NA	NA
Leo	NA	NA	NA	NA

Have to use the is.na() function

Summaries Statistics - Slow

```
>summary(pizza.frame[,"pizza.week"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.000   5.750   6.500   7.125   8.500  10.000

> range(pizza.frame[,1])
   [1]  50 120

> min(pizza.frame[,1])
   [1] 50
> max(pizza.frame[,1])
   [1] 120

> which.max(pizza.frame[,1])
   [1] 8
> which.min(pizza.frame[,1])
   [1] 1

> median(pizza.frame[,1])
   [1] 85
> mean(pizza.frame[,1])
   [1] 85

> var(pizza.frame[,1])
   [1] 600
 > sd(pizza.frame[,1])
   [1] 24.49490
```

```
> cov(pizza.frame[,1],pizza.frame[,2])
   [1] 45

> cor(pizza.frame[,1],pizza.frame[,2])
   [1] 0.904534

> cor.test(y=pizza.frame[,1],x=pizza.frame[,2])

        Pearson's product-moment correlation

data:  pizza.frame[, 2] and pizza.frame[, 1]
t = 5.1962, df = 6, p-value = 0.002022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5511899 0.9827814
sample estimates:
     cor
0.904534

> quantile(pizza.frame[,1],probs=c(seq(0,1,0.1)))
   0%  10%  20%  30%  40%  50%  60%  70%  80%  90%100%
   50   57   64   71   78   85   92   99  106  113  120
```

Summaries Statistics - Fast

In large datasets, we're going to get bored very quick doing all of this data checking using this approach

This is where the benefits of R come into play

You can easily write your own functions / programs to perform whatever standard data check you prefer

I won't go too deeply into this now but will show you the apply functions

apply(), tapply(), lapply() and sapply()

These are highly powerful functions that can be used to make highly repetitive tasks be done easily

```
> sapply(pizza.frame[,1:3], mean)
      weight pizza.week metabolism
      85.000      7.125     75.000

> tapply(pizza.frame[,1], pizza.frame[,4], mean)
      North South
      77.5  92.5

> apply( pizza.frame[,1:3], 2, function (x) { sd(x) / sqrt( length(x) ) } )
      weight pizza.week metabolism
      8.6602540  0.7180703  5.3452248
```

Visual Summaries

Personally all analysis should begin with visual analysis of the data

Visualisation "*retains the information in the data*"
- W. Edwards Deming

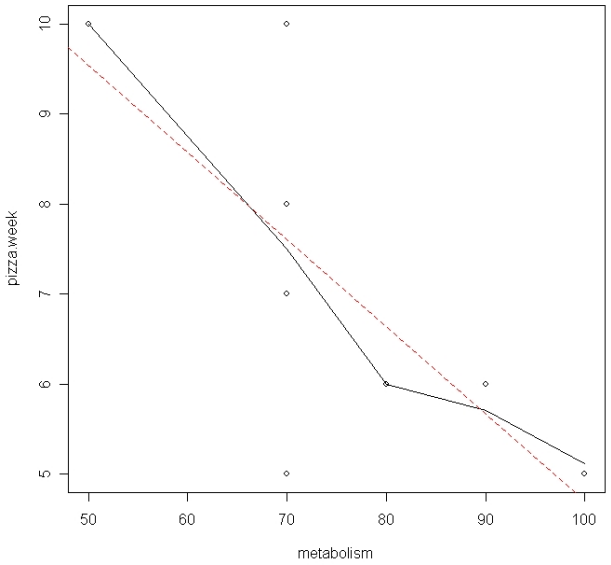
One of R's many strong point is it superb and highly flexible visual and graphing facilities

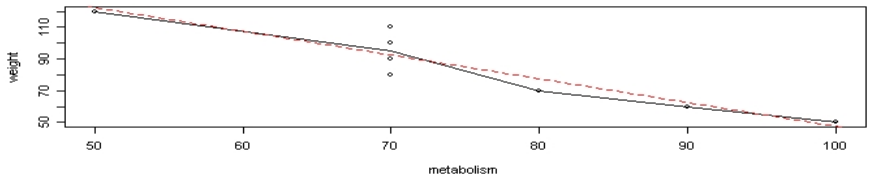
Two main plotting and graphics systems: Standard plots (barcharts, piecharts etc) and trellis plots (taken from Clevelands' Elements of Graphing data)

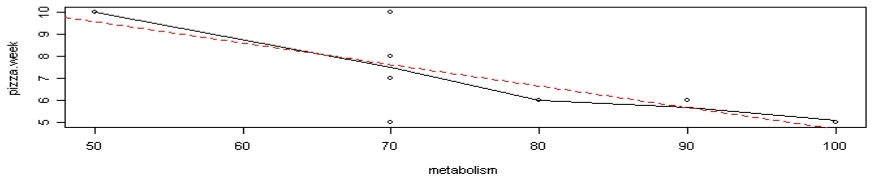
```
plot(pizza.frame[,2:1])
lines(lowess(y=pizza.frame[,1],x=pizza.frame[,2]))
abline(lm(pizza.frame[,1]~pizza.frame[,2]),lty=2,col=2
)

par(mfrow=c(3,1))
```

This breaks the plotting window into a 3 row by 1 column plotting matrix allowing me to plot multiply plots







Visual Summaries

List of useful visual tools:

boxplot()
boxplot(weight~north.south,data=pizza.frame)

image()
demo(image)

identify() – allows you to interact with the graph to identify the matrix co-co-ordinates of an observation (useful for outliers)

qqplot(), qqnorm() and qqline() – quantile plots, similar to a histogram but all of the data can be visulalised

```
par(mfrow=c(2,1))
q <- runif(100)
qqnorm(q)
qqline(q,col=2)
w <- rnorm(100,0,1)
qqnorm(w)
qqline(w,col=2)
```