Data Import and Manipulations

Andrew McAdam

2019-01-15

Contents

Multiple Vectors and Data Frames

So far we have been dealing mostly with variables (e.g. "x" and "answer") and vectors (e.g., "one.to.100"). R can also handle matrices just fine and these are important for many calculations but we won't be dealing with them too much here. Most often you will be dealing with data in data frames. These are a list of vectors (variables) of the same length that are linked together across rows (replicates). That is, the data in the first row of each vector is logically linked to the first row of all other vectors because all of these data were collected from the same forest stand, plant, survey respondent or sediment core. Some of this terminology might be new to you but this is the way you are used to seeing data presented (hopefully!) in spreadsheet programs like Excel.

We can create a new empty data frame using

```
temp<-data.frame()
temp
```

```
## data frame with 0 columns and 0 rows
```

This is a start but it is pretty boring for now since we don't have any data in this data frame. Let's make something a little fancier, which is a data frame with three variables (response, pred1 and pred2). For now we will simply define response to be the numbers 1:10, pred1 will be 2:11 and pred2 will be 3:12. Note that all three variables MUST be the same length

```
temp2 <- data.frame(response=1:10,pred1=2:11,pred2=3:12)
temp2
```

```
##    response pred1 pred2
## 1         1     2     3
## 2         2     3     4
## 3         3     4     5
## 4         4     5     6
## 5          5     6     7
## 6          6     7     8
## 7          7     8     9
## 8          8     9    10
## 9          9    10    11
## 10        10    11    12
```

Note that I have specified the three columns separated by commas and each column is a vector that is defined to be a sequence of 10 numbers.

Referencing Elements in a Data Frame

Note that just like with vectors we can refer to any particular cell in a data frame by referencing its row and column. We can do this to ask R what the value of a cell is or to change a value of a cell. So for example we can ask R what the value in our new data frame called "temp2" is in the 4th row and the first column (i.e. "response")

```
temp2[4,1]
```

```
## [1] 4
```

The notation is [row, column] We can change specific values in a data frame using the assignment action:

```
temp2[4,1]<-45
temp2
```

```
##    response pred1 pred2
## 1         1     2     3
## 2         2     3     4
## 3         3     4     5
## 4        45     5     6
## 5         5     6     7
## 6         6     7     8
## 7         7     8     9
## 8         8     9    10
## 9         9    10    11
## 10       10    11    12
```

We can also ask R for more than one cell at a time

```
temp2[1:5, 1]
```

```
## [1]  1  2  3 45  5
```

This can be read as… "R please return the values in rows one through 5 and the first column only".

If we don't specify a row or column we get them all

```
temp2[1:5, ]
```

```
##   response pred1 pred2
## 1        1     2     3
## 2        2     3     4
## 3        3     4     5
## 4       45     5     6
## 5        5     6     7
```

Remember that we still need to include the column component, but we just leave it blank. We get an error if we use… temp2[1:5]

Error in `[.data.frame`(temp2, 1:5) : undefined columns selected

This would work for a vector though because it only has one dimension

```
one.to.ten <- 1:10
one.to.ten[1:5]
```

```
## [1] 1 2 3 4 5
```

So you can do a lot with data manipulation using R that we wont get into in this class, but you are unlikely to enter data in R. Instead you will need to import a raw data file. R isn't much fun if you can't get your data imported!

Importing Data from Excel

We will start by importing a csv file that is located in the data folder in CourseLink. Open this file and save it to your working directory. If you have done this you do not need to specify the entire path name. I would recommend that you keep your folders organized and include a subfolder called 'data' in your working directory.

```
part.temp<-read.table("data/PartD2.csv", sep=",", header=T)
```

Note that there was no error message!!!

Note that I have included the exact file name (case sensitive) with the proper extension.

I have indicated that the file has a header (header=T). This is because I have variable names stored in the first row of the datafile. R needs to know if these are variable names or actual data!

Finally I have indicated that the file is a comma separated file (sep="‘,"). This is what I always use.

Note that there is a similar function that is not generic to all data types, but is instead specific to .csv files. It is called read.csv. This function will tend to have fewer errors when importing data. In this case you don't need to specify that the separator is ",".

```
part.temp<-read.csv("data/PartD2.csv", header=T)
```

As before we can view a data file by simply typing its name

```
part.temp[1:5, ]
```

```
##   GRID BR   YR AGE   BY LN Dam FOOD LSIZE Julian conestm1
## 1   FL  1 1989   4 1985  1   3    0     2    106 3.160903
## 2   FL  3 1989   2 1987  1   5    0     3    123 3.160903
## 3   LL  3 1989   3 1986  1   6    0     2    119 3.160903
## 4   FL  4 1989   3 1986  1  12    0     2    116 3.160903
## 5   FL  1 1989   6 1983  1  14    0     2     98 3.160903
```

The "head" command lists the first few rows of a datafile.

```
head(part.temp)
```

```
##   GRID BR   YR AGE   BY LN Dam FOOD LSIZE Julian conestm1
## 1   FL  1 1989   4 1985  1   3    0     2    106 3.160903
## 2   FL  3 1989   2 1987  1   5    0     3    123 3.160903
## 3   LL  3 1989   3 1986  1   6    0     2    119 3.160903
## 4   FL  4 1989   3 1986  1  12    0     2    116 3.160903
## 5   FL  1 1989   6 1983  1  14    0     2     98 3.160903
## 6   SU  1 1989   4 1985  1  21    0     3    107 3.160903
```

IMPORTANT: I have provided you with a copy of the PartD2.csv file so that you can follow along and do exactly what I do in class. As a general rule these are data (sometimes unpublished) that belong to me or my colleagues. You are free to play around with the data and to try things out with these data in R. You are not free to publish results with these data without talking to me first and you are not free to share these data with anyone outside this class. If someone else wants a copy of the data have them contact me directly.

There are a few other functions that are important for summarizing data. First we can recall all of the variable names in the data file using

```r
names (part.temp)
```

```
##  [1] "GRID"     "BR"       "YR"       "AGE"      "BY"       "LN"
##  [7] "Dam"      "FOOD"     "LSIZE"    "Julian"   "conestm1"
```

We can get the number of variables in the data file using

```r
length(part.temp)
```

```
## [1] 11
```

Note this IS NOT the number of observations (or rows) in the data file. It is the number of variables within the data frame. If we want the number of rows we need to refer not to the whole dataframe but to a specific variable within the dataframe using the $ symbol.

```r
length(part.temp$GRID)
```

```
## [1] 1422
```

Or we can get both using

```r
dim(part.temp)
```

```
## [1] 1422   11
```

So there are 1422 rows and 11 columns in the data file

An extremely useful function summarizes the data in the data file

```r
summary (part.temp)
```

```
##  GRID           BR              YR             AGE              BY
##  AG: 66   Min.   :0.000   Min.   :1989   Min.   :1.000   Min.   :1983
##  EN: 44   1st Qu.:1.000   1st Qu.:1994   1st Qu.:2.000   1st Qu.:1991
##  FL: 33   Median :1.000   Median :1997   Median :3.000   Median :1994
##  KL:509   Mean   :1.291   Mean   :1997   Mean   :2.943   Mean   :1994
##  LL:291   3rd Qu.:1.000   3rd Qu.:2000   3rd Qu.:4.000   3rd Qu.:1997
##  SU:477   Max.   :7.000   Max.   :2004   Max.   :8.000   Max.   :2003
##  SX:  2                                  NA's   :42
##        LN               Dam              FOOD             LSIZE
##  Min.   :0.0000   Min.   :  3.0   Min.   : 0.000   Min.   :0.000
##  1st Qu.:1.0000   1st Qu.:221.0   1st Qu.: 0.000   1st Qu.:2.000
##  Median :1.0000   Median :392.0   Median : 0.000   Median :3.000
##  Mean   :0.9937   Mean   :392.2   Mean   : 0.447   Mean   :2.734
##  3rd Qu.:1.0000   3rd Qu.:557.8   3rd Qu.: 0.000   3rd Qu.:3.000
##  Max.   :5.0000   Max.   :782.0   Max.   :10.000   Max.   :7.000
##                                   NA's   :176      NA's   :11
##      Julian          conestm1
##  Min.   : 57.00   Min.   :0.000
##  1st Qu.: 99.25   1st Qu.:1.402
##  Median :116.00   Median :2.759
##  Mean   :117.96   Mean   :2.677
##  3rd Qu.:134.00   3rd Qu.:3.748
##  Max.   :203.00   Max.   :5.327
##
```

You will use this one a lot!!!

Note that there is different notation used to summarize different variables. This is because when we imported the data R automatically assumed that any text variable (i.e. GRID) was a factor and that numerical variables (e.g., Julian) is a continuous variable. Sometimes this is correct and sometimes I have just used numbers to refer to factors when I could have just as easily used a letter or word (e.g., BR, LN, FOOD). We can change these into factors using…

```
part.temp$LN<-factor(part.temp$LN)
part.temp$FOOD<-factor(part.temp$FOOD)
part.temp$Dam<-factor(part.temp$Dam)
part.temp$BR<-factor(part.temp$BR)
```

See how things look quite different now.

```
summary (part.temp)
```

```
##  GRID           BR             YR            AGE              BY
##  AG: 66   1      :1211   Min.   :1989   Min.   :1.000   Min.   :1983
##  EN: 44   2      :  85   1st Qu.:1994   1st Qu.:2.000   1st Qu.:1991
##  FL: 33   4      :  75   Median :1997   Median :3.000   Median :1994
##  KL:509   3      :  27   Mean   :1997   Mean   :2.943   Mean   :1994
##  LL:291   0      :  12   3rd Qu.:2000   3rd Qu.:4.000   3rd Qu.:1997
##  SU:477   6      :   6   Max.   :2004   Max.   :8.000   Max.   :2003
##  SX:  2   (Other):   6                  NA's   :42
##  LN            Dam            FOOD           LSIZE           Julian
##  0:  13   166    :   8   0      :1080   Min.   :0.000   Min.   : 57.00
##  1:1408   167    :   7   2      : 102   1st Qu.:2.000   1st Qu.: 99.25
##  5:   1   485    :   7   3      :  19   Median :3.000   Median :116.00
##           9      :   6   4      :  19   Mean   :2.734   Mean   :117.96
##           201    :   6   10     :  16   3rd Qu.:3.000   3rd Qu.:134.00
##           218    :   6   (Other):  10   Max.   :7.000   Max.   :203.00
##           (Other):1382   NA's   : 176   NA's   :11
##     conestm1
##  Min.   :0.000
##  1st Qu.:1.402
##  Median :2.759
##  Mean   :2.677
##  3rd Qu.:3.748
##  Max.   :5.327
##
```

For some variables it isn't entirely clear whether they should be considered continuous or as a factor (e.g., YR, AGE). So we might want to create a new variable that is a factor but which leaves the original variable intact.

```
part.temp$YRF<-factor(part.temp$YR)
part.temp$AGEF<-factor(part.temp$AGE)
summary (part.temp)
```

```
##  GRID           BR             YR            AGE              BY
##  AG: 66   1      :1211   Min.   :1989   Min.   :1.000   Min.   :1983
##  EN: 44   2      :  85   1st Qu.:1994   1st Qu.:2.000   1st Qu.:1991
##  FL: 33   4      :  75   Median :1997   Median :3.000   Median :1994
##  KL:509   3      :  27   Mean   :1997   Mean   :2.943   Mean   :1994
##  LL:291   0      :  12   3rd Qu.:2000   3rd Qu.:4.000   3rd Qu.:1997
##  SU:477   6      :   6   Max.   :2004   Max.   :8.000   Max.   :2003
##  SX:  2   (Other):   6                  NA's   :42
##  LN            Dam            FOOD          LSIZE           Julian
##  0:  13   166    :   8   0      :1080   Min.   :0.000   Min.   : 57.00
##  1:1408   167    :   7   2      : 102   1st Qu.:2.000   1st Qu.: 99.25
##  5:   1   485    :   7   3      :  19   Median :3.000   Median :116.00
##           9      :   6   4      :  19   Mean   :2.734   Mean   :117.96
##           201    :   6   10     :  16   3rd Qu.:3.000   3rd Qu.:134.00
##           218    :   6   (Other):  10   Max.   :7.000   Max.   :203.00
##           (Other):1382   NA's   : 176   NA's   :11
##     conestm1          YRF           AGEF
##  Min.   :0.000   1999   :181   2      :371
##  1st Qu.:1.402   1996   :129   3      :337
##  Median :2.759   1994   :115   1      :233
##  Mean   :2.677   1995   :111   4      :231
##  3rd Qu.:3.748   1998   : 98   5      :121
##  Max.   :5.327   1992   : 97   (Other): 87
##                  (Other):691   NA's   : 42
```

Note that R interprets missing data as NA

We can ask R about missing data using

```
is.na(part.temp[561, 4])
```

```
## [1] FALSE
```

```
is.na(part.temp[562, 4])
```

```
## [1] TRUE
```

We can also specify the negative to ask whether the value is NOT NA

```
!is.na(part.temp[562, 4])
```

```
## [1] FALSE
```

Note that missing data (NA) can cause some functions to not work unless we make some special consideration for them

```
std<-function(x){
#This function takes a vector x and standardizes the values within x to a mean of
#zero and a sd of one
 (x-mean(x))/sd(x)
 }

 summary(std(part.temp$AGE))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      NA      NA      NA     NaN      NA      NA    1422
```

We can improve our function so that it doesn't return NA when any of the data are missing. Instead we can ask it to simply omit those observations from the calculation

```
 std<-function(x){
#This function takes a vector x and standardizes the values within x to a mean of
   #zero and a sd of
 (x-mean(x, na.rm=T))/sd(x, na.rm=T)
 }

summary(std(part.temp$AGE))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -1.32260 -0.64207  0.03846  0.00000  0.71900  3.44113       42
```

Subsets

We often want to deal with only a subset of a dataset. For example we might want to look at only the parturition dates before March 1 (Julian date of 60)

```
part.temp$Julian[part.temp$Julian<60]
```

```
## [1] 57
```

Wow there is only one! What if we wanted to know what year this was in?

```
part.temp$YR[part.temp$Julian<60]
```

```
## [1] 1999
```

Neat! We can also get sophisticated and specify more that one criterion at once

```
part.temp$Julian[part.temp$Julian<90&part.temp$Julian>80]
```

```
##   [1] 81 87 81 87 84 85 83 87 89 88 89 87 82 81 89 88 84 85 86 85 85 86 85
##  [24] 83 89 84 87 84 88 89 86 81 83 83 86 84 86 87 89 87 83 87 84 84 85 83
##  [47] 84 83 87 82 88 88 87 88 86 85 89 86 89 86 85 86 87 88 87 88 82 81 89
##  [70] 87 87 88 86 86 86 84 81 88 86 87 84 89 89 89 85 89 87 85 85 81 87 88
##  [93] 86 88 89 86 85 84 84 86 86 88 88 87 89 89 89 87 84 86 84 87 88 87
```

Some of the other useful logical terms are , <, ==, <=, & (and), ! (not), != (not equal to), | (or)

Here we have used relational operators within indexing to subset our data, but we can also use the subset command

```
summary (subset(part.temp, part.temp$YR==1993))
```

```
##   GRID          BR           YR            AGE              BY       LN
##   AG: 0   1      :54   Min.   :1993   Min.   :1.000   Min.   :1986   0: 0
##   EN: 8   4      : 9   1st Qu.:1993   1st Qu.:1.000   1st Qu.:1989   1:73
##   FL: 0   2      : 7   Median :1993   Median :2.000   Median :1991   5: 0
##   KL:20   3      : 3   Mean   :1993   Mean   :2.658   Mean   :1990
##   LL:21   0      : 0   3rd Qu.:1993   3rd Qu.:4.000   3rd Qu.:1992
##   SU:24   5      : 0   Max.   :1993   Max.   :7.000   Max.   :1992
##   SX: 0   (Other): 0
##        Dam          FOOD        LSIZE           Julian         conestm1
##   28     : 1   0      :73   Min.   :0.000   Min.   : 87.0   Min.   :3.454
##   47     : 1   2      : 0   1st Qu.:2.000   1st Qu.: 99.0   1st Qu.:3.454
##   65     : 1   3      : 0   Median :3.000   Median :105.0   Median :3.454
##   72     : 1   4      : 0   Mean   :2.849   Mean   :115.5   Mean   :3.454
##   81     : 1   5      : 0   3rd Qu.:3.000   3rd Qu.:122.0   3rd Qu.:3.454
##  100    : 1   6      : 0   Max.   :5.000   Max.   :193.0   Max.   :3.454
##  (Other):67   (Other): 0
##       YRF          AGEF
##  1993   :73   2      :26
##  1989   : 0   1      :20
##  1990   : 0   5      : 8
##  1991   : 0   3      : 7
##  1992   : 0   4      : 7
##  1994   : 0   7      : 3
##  (Other): 0   (Other): 2
```

So this a summary of the part.temp data file, but only for those values that were in year=1993.

It is important to note that when you refer to values of a factor and not a continuous variable you need to specify the text code in quotes. This tells R that you are referring to a particular text string value for that variable and not some other object. So for example, if I wanted to restrict the summary to only the GRID called SU, I need to put SU in quotes.

If I type… summary (subset(part.temp, part.temp$GRID==SU))

I will get an error… Error in eval(expr, envir, enclos) : object 'SU' not found

```
summary (subset(part.temp, part.temp$GRID=="SU"))
```

```
##  GRID           BR            YR             AGE              BY
##  AG:  0   1      :390   Min.   :1989   Min.   :1.000   Min.   :1985
##  EN:  0   4      : 37   1st Qu.:1994   1st Qu.:2.000   1st Qu.:1991
##  FL:  0   2      : 25   Median :1998   Median :3.000   Median :1995
##  KL:  0   3      : 13   Mean   :1997   Mean   :3.052   Mean   :1994
##  LL:  0   0      :  8   3rd Qu.:2000   3rd Qu.:4.000   3rd Qu.:1997
##  SU:477   6      :  2   Max.   :2004   Max.   :8.000   Max.   :2003
##  SX:  0   (Other):  2
##  LN           Dam            FOOD          LSIZE           Julian
##  0:  9   9      :  6   0      :351   Min.   :0.000   Min.   : 57.0
##  1:468   353    :  6   2      : 25   1st Qu.:2.000   1st Qu.:100.0
##  5:  0   366    :  6   3      :  4   Median :3.000   Median :119.0
##          683    :  6   4      :  0   Mean   :2.809   Mean   :119.5
##          47     :  5   5      :  0   3rd Qu.:3.000   3rd Qu.:135.0
##          128    :  5   (Other):  0   Max.   :6.000   Max.   :198.0
##          (Other):443   NA's   : 97   NA's   :7
##     conestm1          YRF           AGEF
##  Min.   :0.000   1999   : 63   3      :110
##  1st Qu.:1.402   1998   : 45   2      :107
##  Median :2.337   2001   : 41   1      : 89
##  Mean   :2.564   1992   : 34   4      : 82
##  3rd Qu.:3.748   1994   : 30   5      : 50
##  Max.   :5.327   1997   : 30   6      : 29
##                  (Other):234   (Other): 10
```

This is because we could include a variable in there rather than a text string

```
variable<-"SU"
summary (subset(part.temp, part.temp$GRID==variable))
```

```
##  GRID           BR            YR             AGE              BY
##  AG:  0   1      :390   Min.   :1989   Min.   :1.000   Min.   :1985
##  EN:  0   4      : 37   1st Qu.:1994   1st Qu.:2.000   1st Qu.:1991
##  FL:  0   2      : 25   Median :1998   Median :3.000   Median :1995
##  KL:  0   3      : 13   Mean   :1997   Mean   :3.052   Mean   :1994
##  LL:  0   0      :  8   3rd Qu.:2000   3rd Qu.:4.000   3rd Qu.:1997
##  SU:477   6      :  2   Max.   :2004   Max.   :8.000   Max.   :2003
##  SX:  0   (Other):  2
##  LN          Dam           FOOD         LSIZE           Julian
##  0:  9   9      :  6   0      :351   Min.   :0.000   Min.   : 57.0
##  1:468   353    :  6   2      : 25   1st Qu.:2.000   1st Qu.:100.0
##  5:  0   366    :  6   3      :  4   Median :3.000   Median :119.0
##          683    :  6   4      :  0   Mean   :2.809   Mean   :119.5
##          47     :  5   5      :  0   3rd Qu.:3.000   3rd Qu.:135.0
##          128    :  5   (Other):  0   Max.   :6.000   Max.   :198.0
##          (Other):443   NA's   : 97   NA's   :7
##     conestm1          YRF           AGEF
##  Min.   :0.000   1999   : 63   3      :110
##  1st Qu.:1.402   1998   : 45   2      :107
##  Median :2.337   2001   : 41   1      : 89
##  Mean   :2.564   1992   : 34   4      : 82
##  3rd Qu.:3.748   1994   : 30   5      : 50
##  Max.   :5.327   1997   : 30   6      : 29
##                  (Other):234   (Other): 10
```

I hope you can see that the reason you need the quotes is because if we had a variable in the workspace called SU and a value for the factor GRID that was SU then it could get quite confusing if we didn't specify which one we were referring to.

We can also get quite fancy in our subsetting

```
subset1<-part.temp$GRID=="SU"|part.temp$GRID=="KL"&part.temp$YR==1993
summary (subset(part.temp$Julian, subset1))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    57.0   100.0   118.0   119.5   135.0   198.0
```

Note that this is different from

```
subset2<-(part.temp$GRID=="SU"|part.temp$GRID=="KL")&part.temp$YR==1993
summary (subset(part.temp$Julian,subset2))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    88.0    98.0   105.5   116.4   121.2   193.0
```

Note how I have used the brackets to control the subsetting. In the first example I was saying that I want either records of (SU) OR records that are both (KL and 1993). In the second example I am asking for records that are (either SU or KL) and (1993). You need to think very carefully about what you are doing with these complex subsets!

Summarizing By Groups

We can also summarize data by levels of a factor using the tapply command. For example we might be interested in the average breeding date of females in each year

```
tapply(part.temp$Julian, part.temp$YRF, mean)
```

```
##      1989      1990      1991      1992      1993      1994      1995
## 103.30667 137.42500 128.08333 145.23711 115.46575  95.74783 152.09910
##      1996      1997      1998      1999      2000      2001      2002
##  98.62016 117.82105 120.51020  95.19890 120.06667 125.89873 125.71186
##      2003      2004
## 125.21569 128.74074
```

Or by age

```
tapply(part.temp$Julian, part.temp$AGEF, mean)
```

```
##        1        2        3        4        5        6        7        8
## 124.3176 119.6469 116.0801 115.2511 120.2314 113.0154 117.5263 123.0000
```

Sourcing Code

Remember that when we imported the PartD2.csv file we had to convert some of the variables to factors and then create some new variables before we were ready to do interesting things with the data. If you save your workspace then you don't need to keep importing your data into R each time you restart the program. But if you do end up importing your data file a lot it would become a bit of a pain to have to retype all that code. So instead you can write a simple source file that you can simply refer to when you need this to be done. With this approach, you do not rely on saving your workspace every time you close your R session. You would then start each session with a clean working directory, and call your script file with the "source" command.

For an exercise, extract all the data manipulation commands from this document that we needed to reach our final part.temp object, put these in one script file, name this file for instance "part.temp.import.R", close R without saving your workspace, open R again without loading a workspace. Check that you have no objects in your workspace with

ls()

Then load your just created script file with:

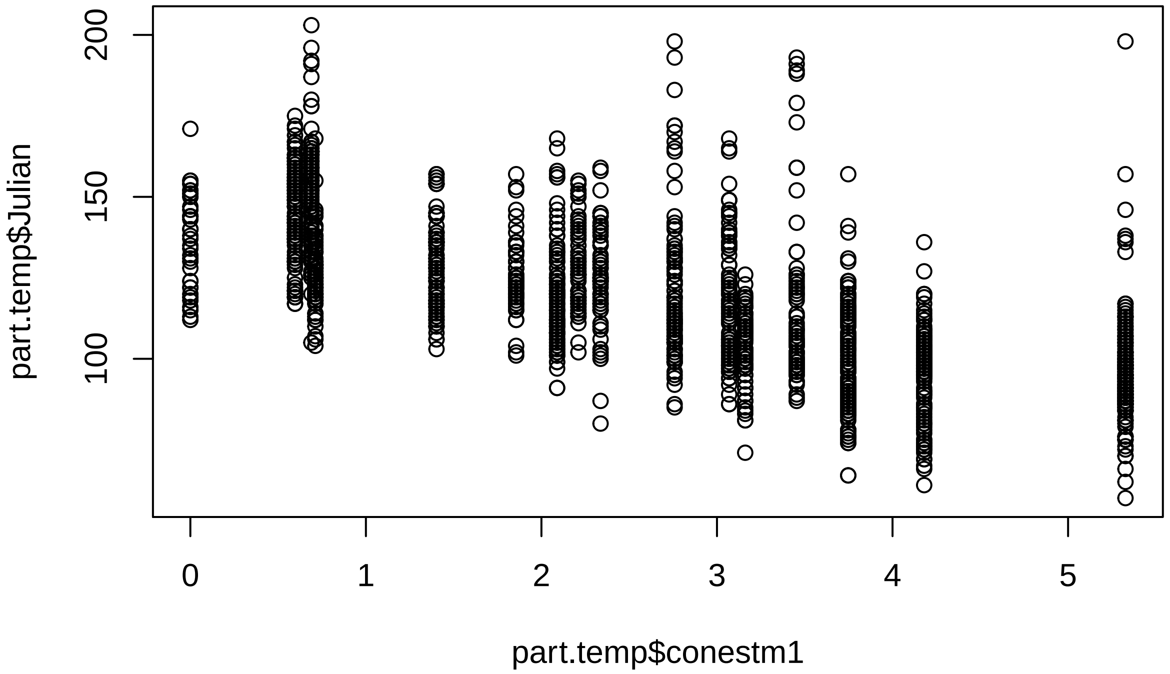
source("part.temp.import.R")

Basic Graphics

The final thing that I want to do is to provide a brief overview of graphics. I will try and return to graphics using R as we work our way through the course. The graphics in R are very powerful but they can be a little daunting at first because you have the ability to change just about every aspect of the graphics.

Basics graphics in R come from the plot function. To start with we will plot the relationship between food abundance (conestm1) and parturition date (Julian) by individual female red squirrels. The notation for the plot command is to first provide the variable for the X axis and then the variable for the Y axis.

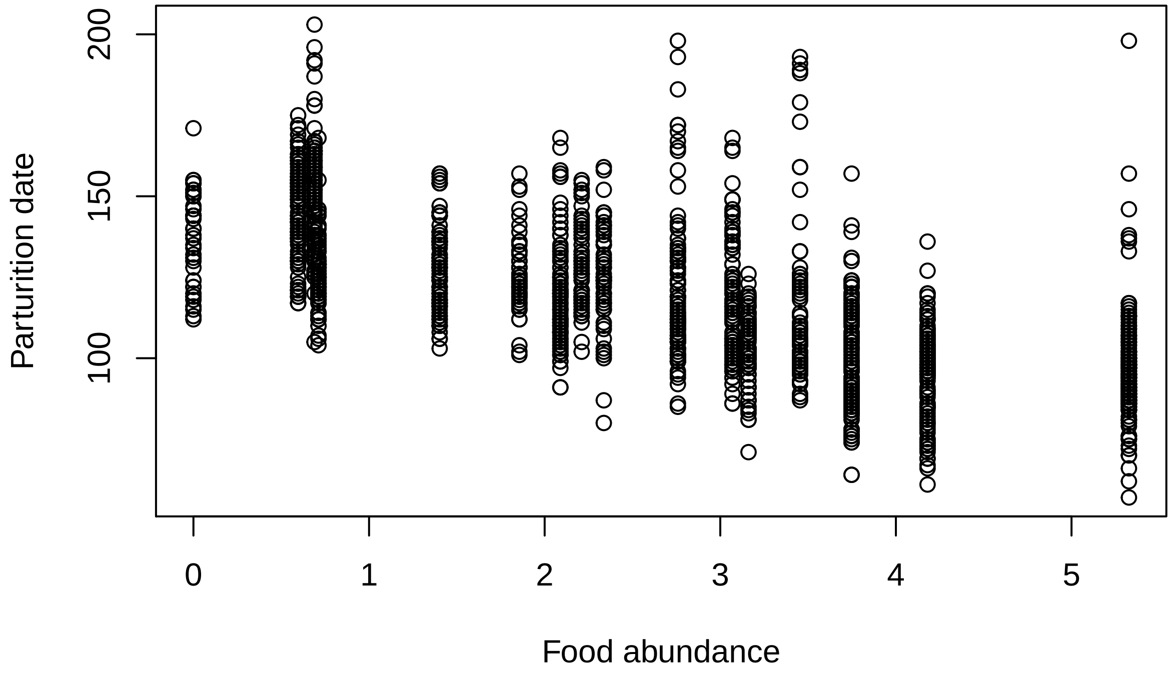
```
plot(part.temp$conestm1, part.temp$Julian)
```



This is the simplest type of plot, but we can pretty it up a bit by specifying some of the parameters rather than accepting the defaults

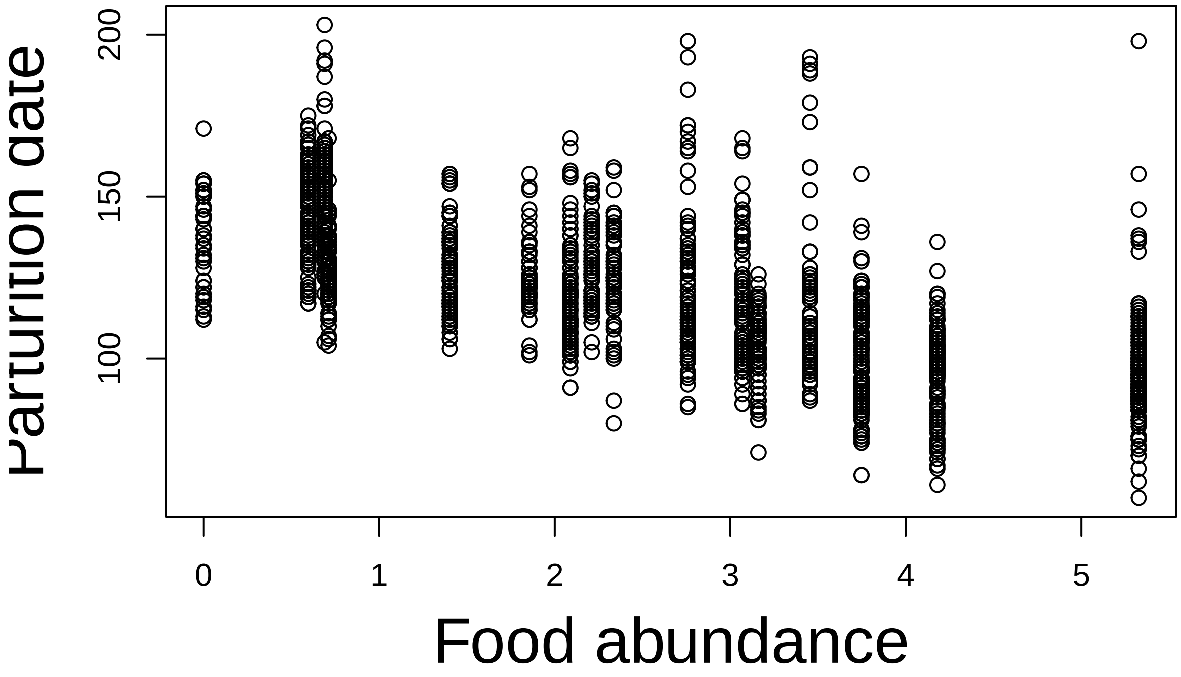
First of all we probably want to rename our axis labels since they don't look very good right now

```
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date")
```



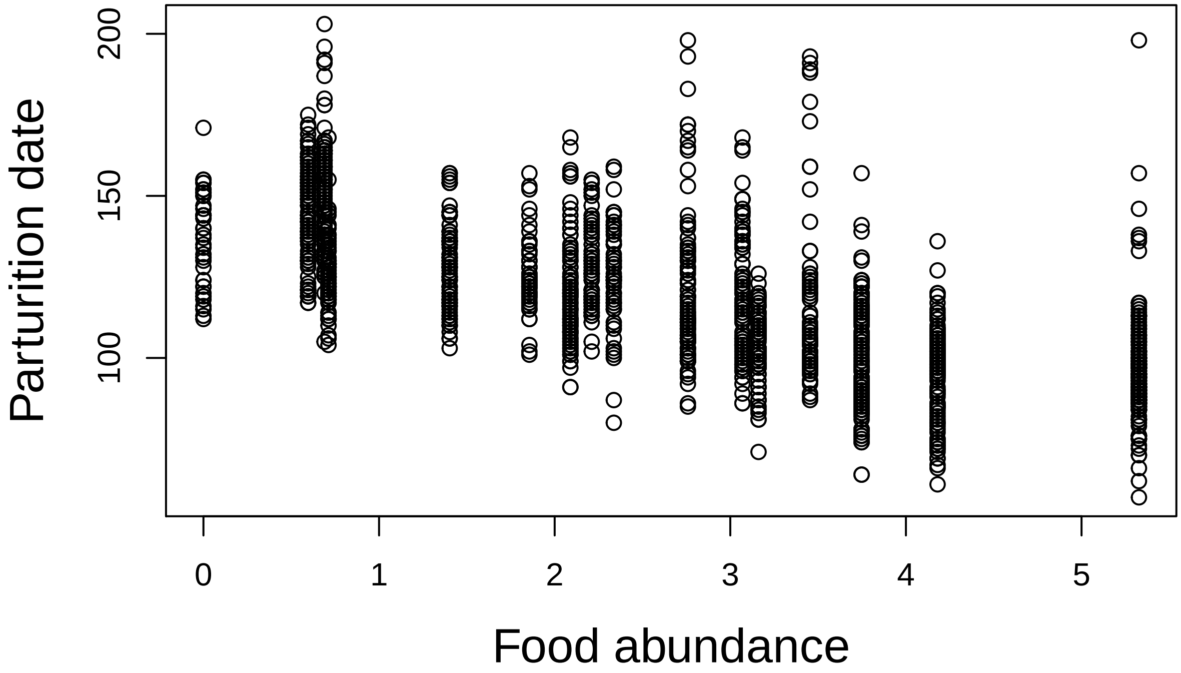
Those labels are also a bit small so we can make then bigger

```
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date", cex.lab=2)
```



Maybe that's a bit too big

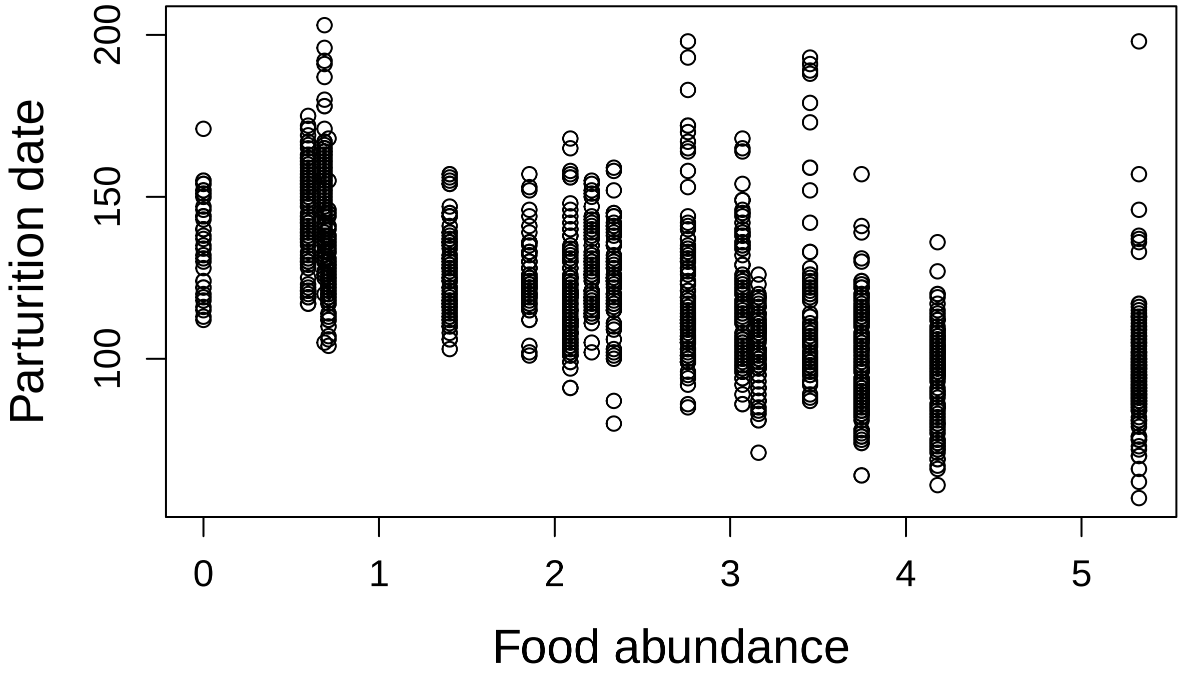
```r
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date", cex.lab=1.5)
```



Note that the cex stands for character expansion and 2 means 2x as big as the default

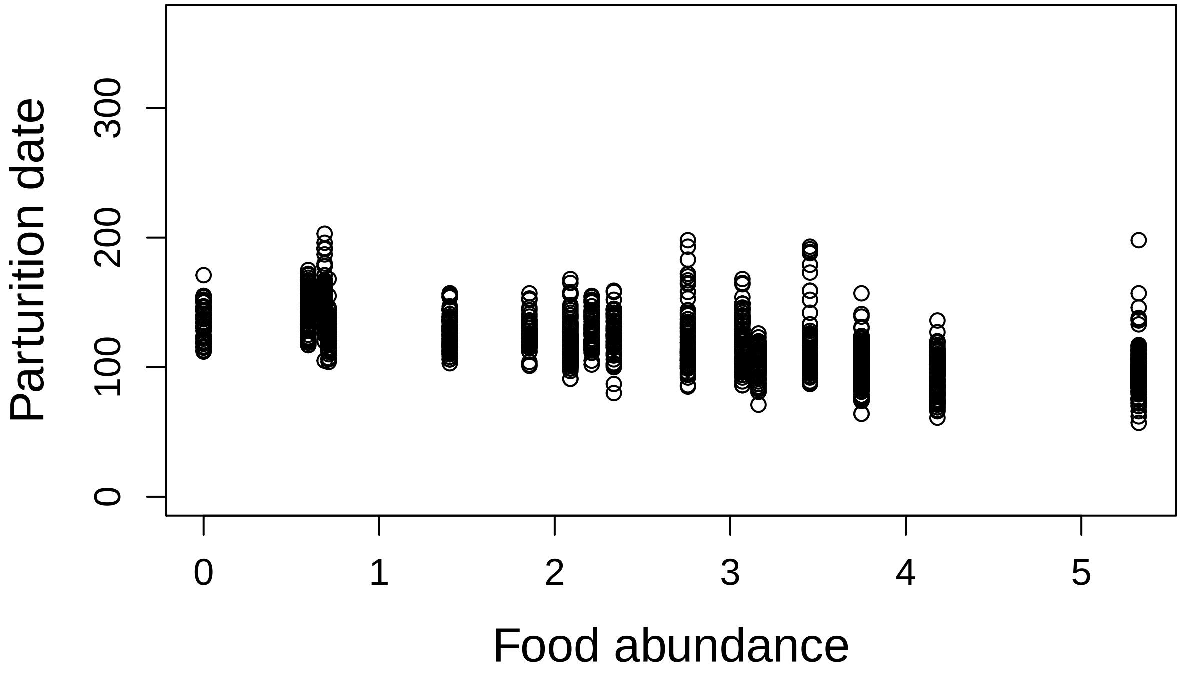
We can also change the size of the axes

```r
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date", cex.lab=1.5, cex.axis=1.2)
```



We might want to change the scale of the y axis

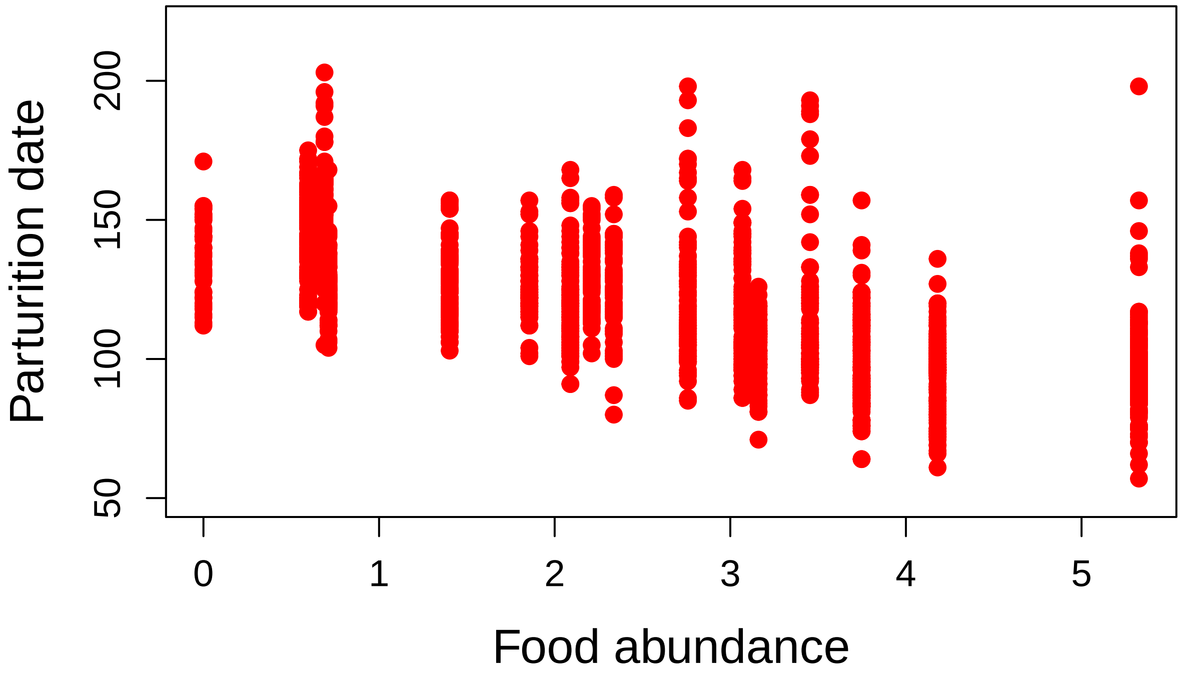
```
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date", cex.lab=1.5, cex.axis=1.2, ylim=c(0, 365))
```



Here the first number is the minimum value whereas the last number is the maximum value. This doesn't look very good so I don't think we will use this.

Finally we can change the plot character both their type, size and color

```
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abundance",
ylab="Parturition date", cex.lab=1.5, cex.axis=1.2, ylim=c(50, 220),
pch=19, cex=1.1, col="red")
```



We can also add a trend line to the plot. This is done in a slightly different way. Instead of providing the command within the plot command we specify it afterward and it is applied to the graph that is open. You can do this in three ways

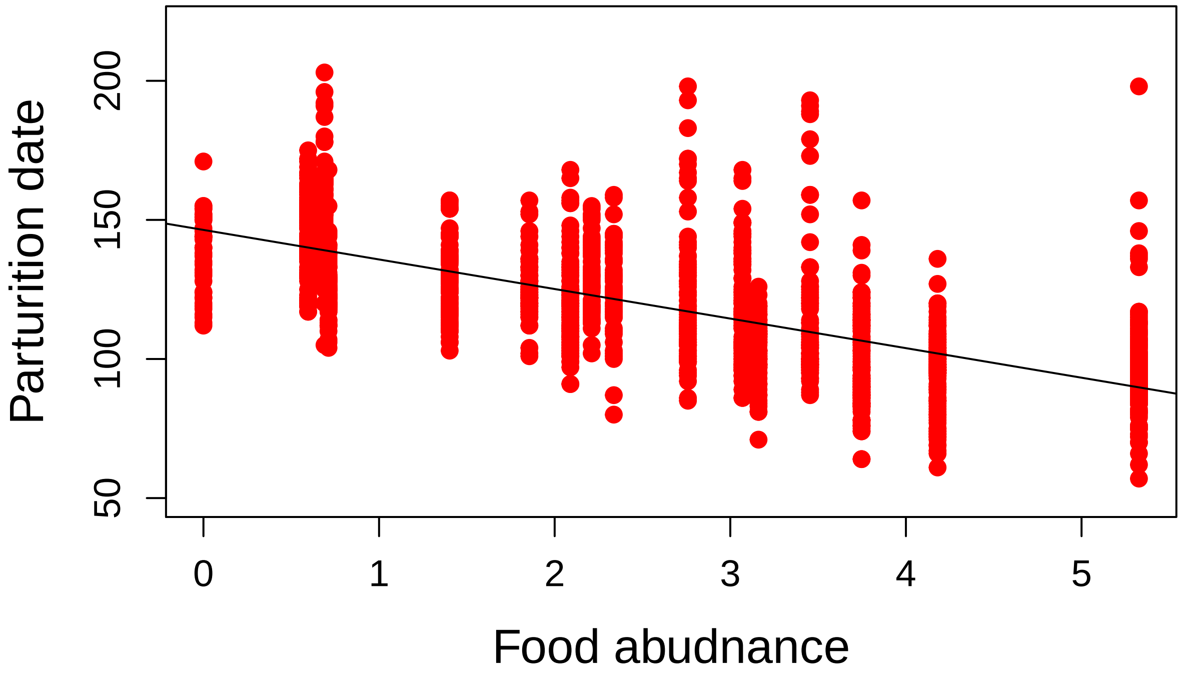
abline(h=100) abline(v=2) abline(150, -10) where the first value represents the intercept and the second represents the slope.

or abline(lm(part.temp*Julian part.temp*conestm1)) where R fits the line from the linear model of the effects of cones on breeding date.

Note that in the above notation when you are entering these commands into the console while you have an open ‘Quartz’ plot window then these lines simply get added to the existing open plot.

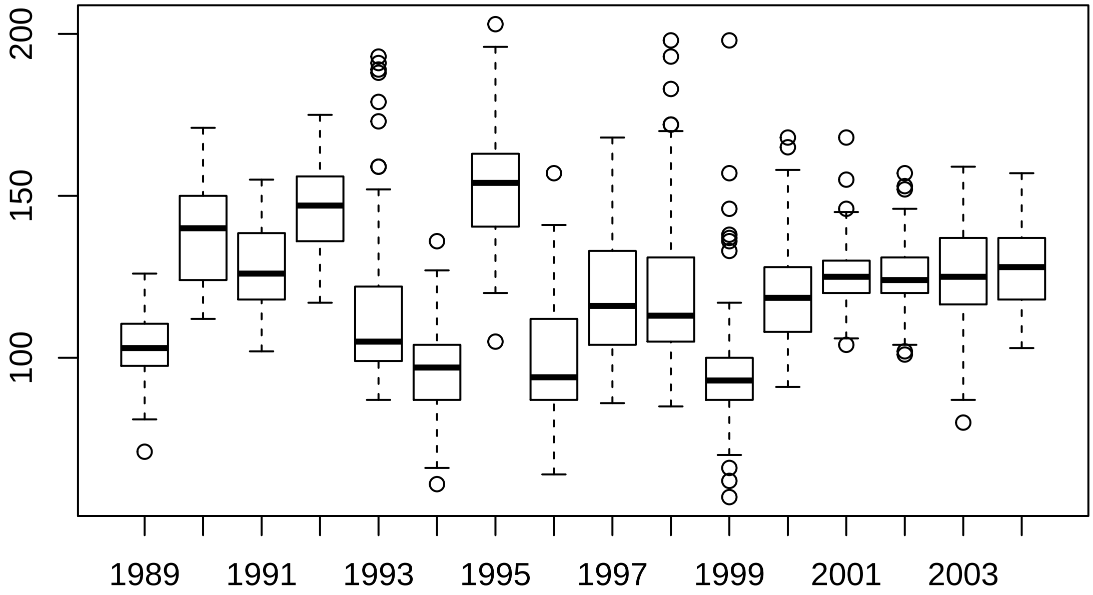
To bring it all together we can use:

```
plot(part.temp$conestm1, part.temp$Julian, xlab="Food abudnance",
ylab="Parturition date", cex.lab=1.5, cex.axis=1.2, ylim=c(50, 220),
pch=19, cex=1.1, col="red")
abline(lm(part.temp$Julian~part.temp$conestm1))
```



We can create a series of boxplots in a similar way if the variable on the x axis is a factor

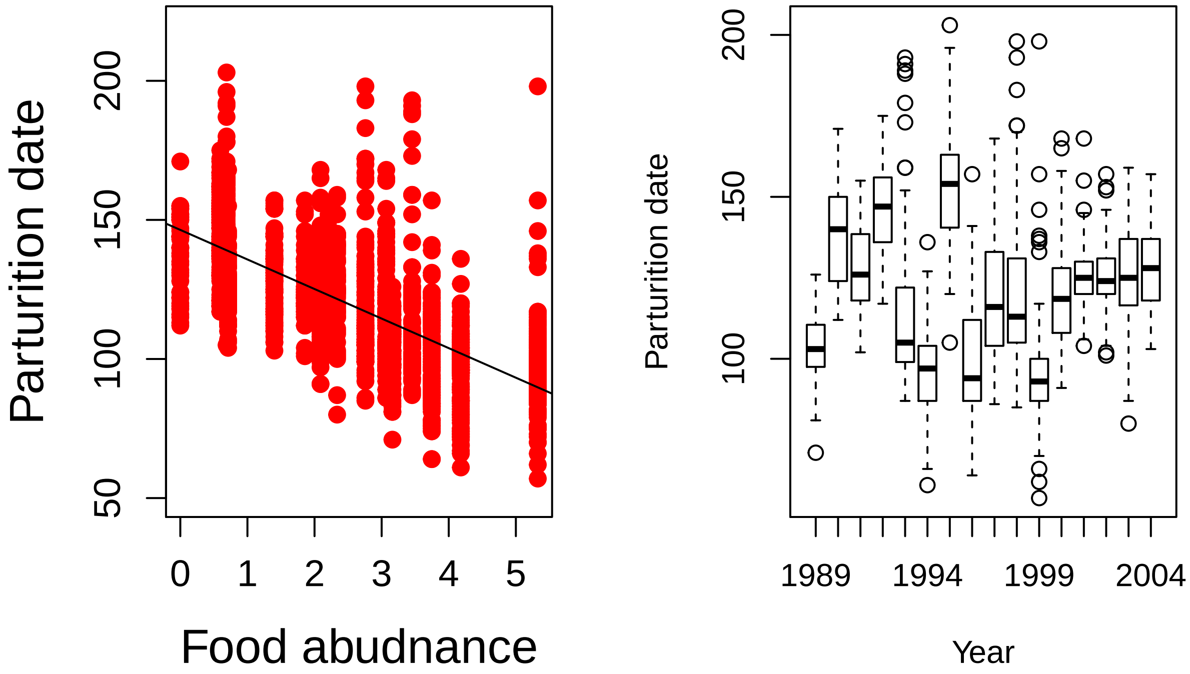
```
plot(part.temp$YRF, part.temp$Julian)
```



Here the middle line represents the median, the box is the inter-quartile range, the whiskers extend to the range of the data or 1.5 times the box size from the nearest hinge - whichever is less. Points beyond this are noted.

We can plot figures side by side on the same page by changing the parameter mfrow. We will indicate how many rows and columns of figures we want. This one has one row and two columns.

```
par(mfrow=c(1,2))
 plot(part.temp$conestm1, part.temp$Julian, xlab="Food abudnance", ylab="Parturition date", cex.lab=1.5
abline(lm(part.temp$Julian~part.temp$conestm1))
 plot(part.temp$YRF, part.temp$Julian, xlab="Year", ylab="Parturition date")
```



We can either save these graphics once they are produced or we can specify a name and format for them in advance. To do this we turn on the graphics program, create the graphs, then turn it off.

```
pdf(file="Sept23Fig.pdf")
 par(mfrow=c(1,2))
 plot(part.temp$conestm1, part.temp$Julian, xlab="Food abudnance", ylab="Parturition date", cex.lab=1.5
 abline(lm(part.temp$Julian~part.temp$conestm1))
 plot(part.temp$YRF, part.temp$Julian, xlab="Year", ylab="Parturition date")
 dev.off()
```

```
## pdf
##   2
```

Note that no new graphics window has been created now. Instead This pdf file has been stored in your working directory (not your workspace) and is ready to be submitted for publication!!!!

ggplot

All of the figures above were made using base R. This is a bit of an old-fashioned way to make figures because there is a new package called ggplot that is extremely flexible and powerful for making figures. The notation is quite different from what I have described above for base R, but it is worth learning if you want to make all yoru figures in R. For the purpose of this course base R will work fine.