Data_Frames 2_3

HDS, Mike Kozlowski

June 24, 2018

Data Frames, Section 2.3

This discussion of Data Frames is mean to accompany vand der Loo and de Jonge, section 2.3 on Data frames

The data frame in R is the R approach to handling a 2 dimensional data table, where each row is an observation (or event or individual or specimen) and each column is a variable, a measurement of some property of the observation.

This corresponds to a table in an SQL style database, or a worksheet in an Excel file.

Each column may be a different type of variable, but all entries in a given column are the same type of data. Mising data is allowed and is denoted “NA”.

Just as a reminder:

We can create a data frame easily enough

```
mydata=data.frame(x=c(1,2,3,4,5),name=c("Al","Bob","Cathy","Don","Eric"),
                  sex=as.factor(c("M","M","F","M","M")))
summary(mydata)
```

```
##        x         name           sex
##  Min.   :1   Length:5           F:1
##  1st Qu.:2   Class :character   M:4
##  Median :3   Mode  :character
##  Mean   :3
##  3rd Qu.:4
##  Max.   :5
```

```
str(mydata)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ x   : num  1 2 3 4 5
##  $ name: chr  "Al" "Bob" "Cathy" "Don" ...
##  $ sex : Factor w/ 2 levels "F","M": 2 2 1 2 2
```

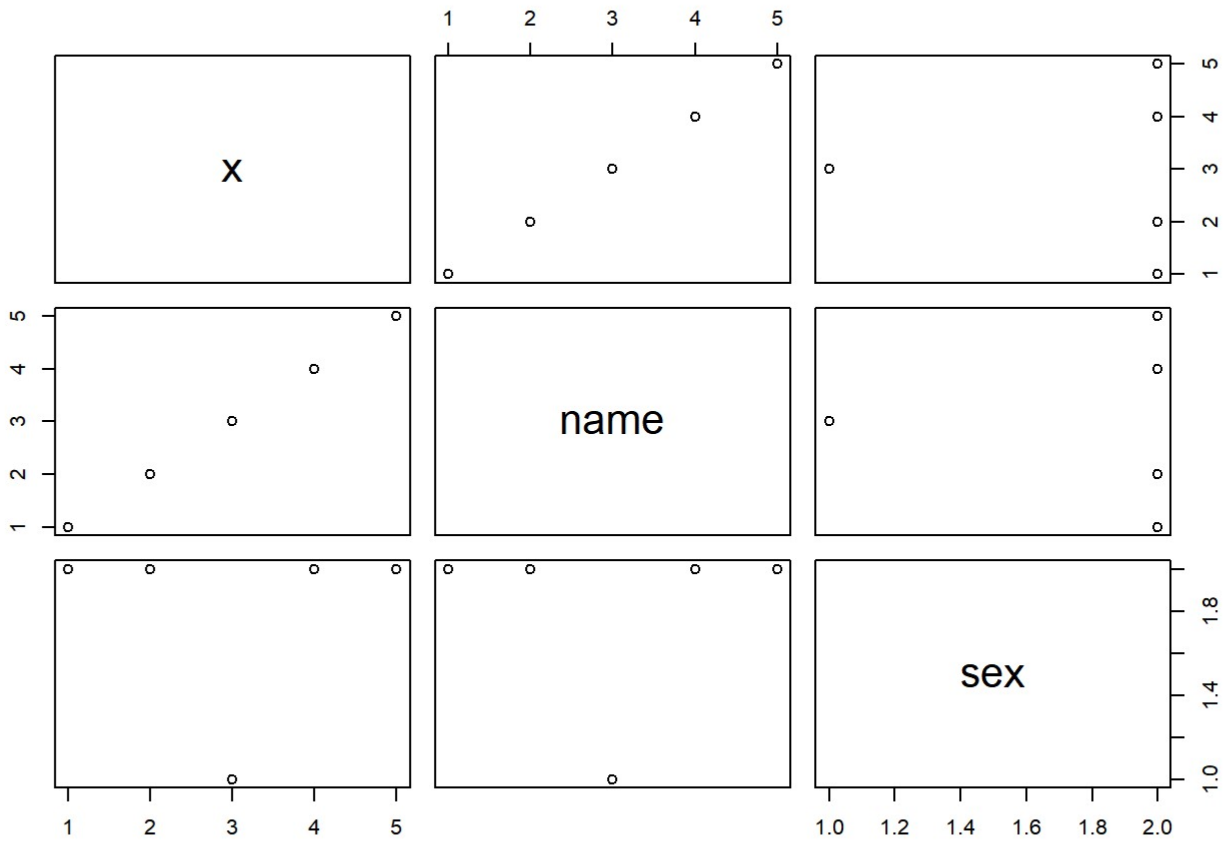
We can assign row names or labels as well, Note that the definition of the data frame included column names

```
rownames(mydata)<-c("P1","P2","P3","P4","P5")
print(mydata)
```

```
##    x  name sex
## P1 1    Al   M
## P2 2   Bob   M
## P3 3 Cathy   F
## P4 4   Don   M
## P5 5  Eric   M
```

There is a also a default plot function

```
plot(mydata)
```



We can also create a data frame from a set of vectors that are already defined

```
x=1:10
y=x^2+rnorm(10,mean=2,sd=0.3)
z=as.factor(rep(c("A","B"),each=5))
newdata=data.frame(A=x,B=y,C=z)
str(newdata)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ A: int  1 2 3 4 5 6 7 8 9 10
##  $ B: num  3.04 6.26 11.04 18.26 27.18 ...
##  $ C: Factor w/ 2 levels "A","B": 1 1 1 1 1 2 2 2 2 2
```

###The Iris data set

It is common for packages to include example data sets, so that you can try the functions in the package on a sample data set.

There is a classic data set, the “iris” data set, that is commonly discussed. A lot of these data sets are the “lab rats” of the big data world, everybody has seen them and knows them. They are endlessly reused as examples for different methods.

The Iris flower data set or Fisher’s Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis*. It is sometimes called Anderson’s Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species. Two of the three species were collected in the Gaspé Peninsula “all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus”.

RA Fisher was a British Applied Statistician who worked mostly between the world wars- he invented a lot of multivariate statistical methods, as well as some early permutation methods you will encounter shortly.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

We can load it, and look at the structure

```
data(iris)
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 ...
```

We can refer to columns within a data frame using the name and the column name

```
mean(iris$Sepal.Length)
```

```
## [1] 5.843333
```

Or we can use the number of the column inside a double bracket. Sepal.length is the first column

```
mean(iris[[1]])
```

```
## [1] 5.843333
```

We can access elements within the data frame much the same way we access the contents of a matrix, by using row and column indices. If we want the first 10 rows of the Petal.Width column (column 4 above), we would use iris[1:10,4] - notice how this follows pretty much the same row,column formalism used in linear algebra

```
iris[1:10,4]
```

```
##  [1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1
```

We can slice the iris data abit

The mean sepal.length was 5.83. Let’s see how many flowers from the first species, setosa, had a petal.length greater than the average- we need to slice out Species==“setosa” and Sepal.Length>5.8

```
A=iris[(iris$Species=="setosa")&(iris$Sepal.Length>5.83),]
str(A)
```

```
## 'data.frame':    0 obs. of  5 variables:
##  $ Sepal.Length: num 
##  $ Sepal.Width : num 
##  $ Petal.Length: num 
##  $ Petal.Width : num 
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..:
```

Interesting- is this a mistake, or are none of the Sepal.Lengths of seposa specimens above the average (where the average is over all species?)

```
summary(iris[(iris$Species=="setosa"),])
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100  
##  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200  
##  Median :5.000   Median :3.400   Median :1.500   Median :0.200  
##  Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246  
##  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300  
##  Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600  
##        Species  
##  setosa    :50  
##  versicolor: 0  
##  virginica : 0  
##                 
##                 
## 
```

Okay, so here is the slicing of the iris data structure to simply extract the setosa specimens, the maximum Sepal Length is 5.8 for all setosa -they all have relatively low Sepal lengths

Now- Notice the format of the slice

iris[(iris$Species==“setosa”),]

iris[(test condition),] slices out all the rows that meet the test condition (species=“setosa”) and all the columns of the data

We could select specific columns of the data, to select only columns 1 to 3, we could use

iris[(test condition), 1:3]

for example.

###The mpg data set

The mpg data set lists a number of characteristics of cars from 1999 and 2008, with a range of models and characteristics of vehicles. It is included as an example file withe the ggplot2 package

We can use either library() or require() to load the ggplot2 package.

```
require("ggplot2")
```

```
## Loading required package: ggplot2
```

```
data(mpg)
```

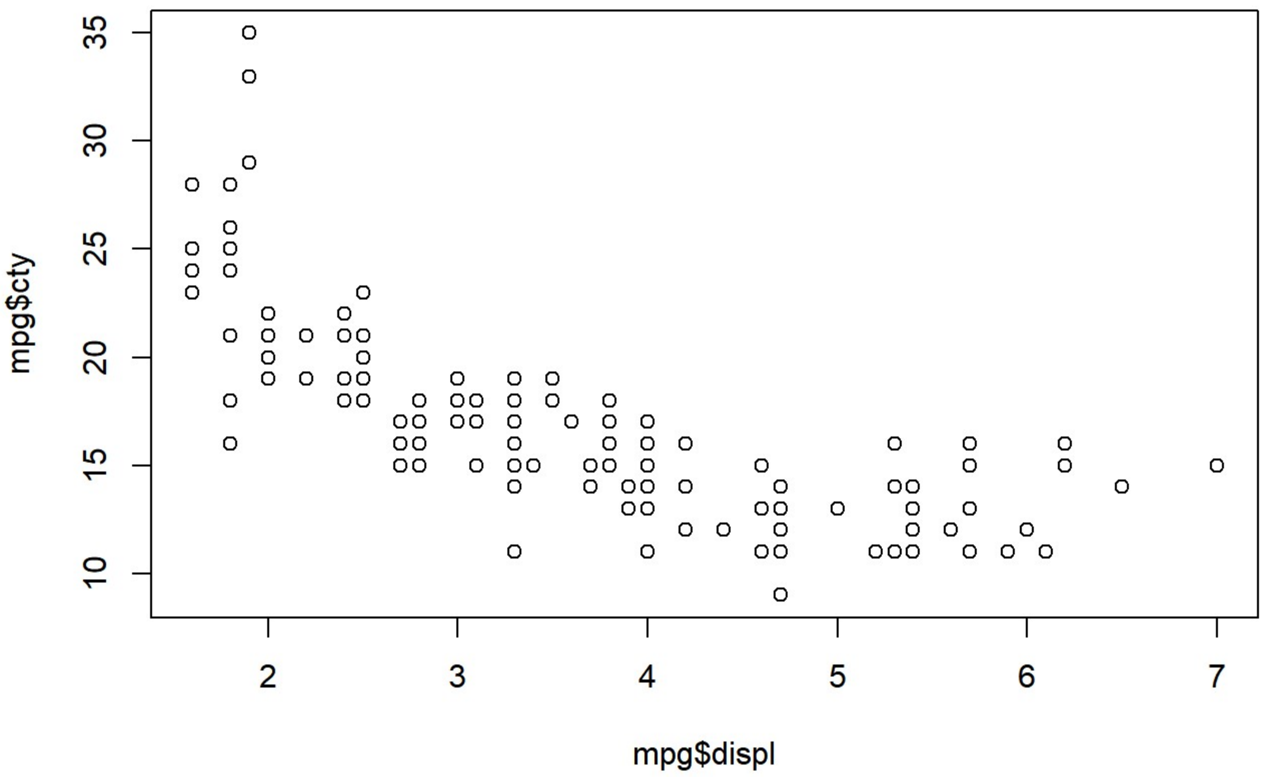
Questions to answer

What variables are present?

A, AData, iris, mydata, newdata, mpg, myfile, x, y, z

Can you plot the city mpg rating as a function of engine displacment (displ, a measure of engine size)?

```
plot(mpg$displ, mpg$cty)
```



How many models made by “ford” are in this data frame?

```
sum(mpg$manufacturer == "ford")
```

```
## [1] 25
```

###Loading Data from Excel Files

Exel (like many other programs) is capable of loading and saving data files in a range of different formats.

One of the easiest formats to work with is the CSV (comma separated value) format.

When you save a single Excel worksheet as a comma separated data file, there is often a *header* rown that contains the names of the variables in the column. Each column is separatd by a comma, with a linefeed at the end of each line.

There are several ways to import such a file into R and load it into a data frame, the read.table and read.csv options are the most straight-forward “canned” options to use.

R does have a file I/O system, which we will discuss a bit later.

To load a file, one needs the name of the file, and the location of the file in the directory structure of the computer you are using, or a URL.

To select a file interactively, you can use the file.choose() function to open a file selection window so you can navigate to the file.

I went to the Open Data Buffalo site run by the city of Buffalo as an open, public data repository which is located at:

https://data.buffalony.gov/ (https://data.buffalony.gov/)

and I downloaded (exported) a CSV file of the Assessment Roll of Property values for the City of Buffalo, a file called 2017-2018_Assessment_Roll.csv

You can download this from Open Data Buffalo, or from D2L, and save it on your machine, then use file.choose and read.csv as below to load the file

```
myfile=file.choose()
AData=read.csv(myfile,header=TRUE)
```

Questions

What data is included in this file?

Sets of numbers and strings corresponding to different keys, such as LAND.VALUE with several values following it, and ADDRESS with addresses like “144 PEACH”.

What is the median total.value of homes between 1000 and 2000 squarefeet?

```
median(AData$TOTAL.VALUE[(AData$TOTAL.LIVING.AREA > 1000) & (AData$TOTAL.LIVING.AREA < 200
0)])
```

```
## [1] 64000
```

What is the average size of the lots in Buffalo?

```
mean(AData$FRONT * AData$DEPTH)
```

```
## [1] 10913.52
```

What is the highest valued property in the Delaware council district?

```
max(AData$TOTAL.VALUE[AData$COUNCIL.DISTRICT == "DELAWARE"])
```

```
## [1] 23445000
```