Lab 2

1 Data frames in R

Data frames are some of the most used objects in R. Data frames can be thought of as a generalization of a matrix containing a list of vectors and/or factors of the same length. Data frames have a rectangular structure and data in the same row number across columns come from the same experimental unit. Consider the following example from the lecture notes.

Four students (A-D) each perform an analysis in which exactly 10.00 ml of exactly 0.1 M sodium hydroxide is titrated with exactly 0.1 M hydrochloric acid. Each student performs five replicate titrations, with the results shown in $Table1\_1$.

A	10.08	10.11	10.09	10.10	10.12
B	9.88	10.14	10.02	9.80	10.21
C	10.19	9.79	9.69	10.05	9.78
D	10.04	9.98	10.02	9.97	10.04

It is highly recommended that you will place the data files into a directory that is easy to access. For example you can use "$E:\Chemometrics\Labs$" if you are using an external drive and we will assume for the rest of the semester that this is the location of your R files.

The measurements are stored in the $.txt$ file called $Table1\_1.txt$. To import the file first set the working directory for R as the folder in which you stored the $.txt$ data files. For example if the $Table1\_1.txt$ file is in the folder $E:\Chemometrics\Labs$ then set the working directory of R to be this folder using the *setwd* command:

$> setwd("E:/Chemometrics/Labs")$

NB: Notice that in the command above \ has been replaced by /.

To import the 5 replicate titrations done by each student (A-D) stored in the $Table1\_1.txt$ file use

the *read.table* command

$> Titration <- read.table("Table1_1.txt")$

Using *read.table* is an easy way to read a data file that is laid out as a matrix and may have headers (column descriptions). In this particular example the 5 replicate measurements done by each student are saved as a table with one row for each student and a name in the first column to label each student/row. To inform R that the file contains row names use the $row.names = 1$ option in the *read.table* command

$> Titration <- read.table("Table1_1.txt", row.names = 1)$

Another way of reading data is by specifying the complete path for the folder.

$> Titration <- read.table("E:/Chemometrics/Labs/Table1_1.txt", row.names = 1)$

Setting the current working directory in the *Labs* folder with the *setwd* command is NOT necessary if the full path is specified. To print the contents of the *Titration* data frame, type its name and press *Enter* in the prompt line.

$> Titration$

Accessing values in a data frame is similar to displaying values in a matrix. To access the first row of the *Titration* data frame use

$> Titration[1,]$

or the third column

$> Titration[, 3]$

Exercise: Calculate the mean and standard deviation for each student using the *mean*() and *sd*() functions and apply these functions to each row using the *apply* command.

$> apply(Titration, 1, mean)$ displays the means for each student

A	B	C	D
10.10	10.01	9.90	10.01

In the parameters of the *apply* function, number 1 represents the fact that the mean function is

applied to rows. If number 2 is used instead, the mean function is applied by columns. Similarly $> apply(Titration,\ 1\ , sd)$ displays the standard deviations for each student

A B C D

0.01581139 0.17175564 0.21047565 0.03316625

2 The empirical distribution of repeated measurements

Although the standard deviation gives a measure of the spread the data around the mean value, it does not indicate the shape of the distribution. Consider the nitrate ion concentration example from the lecture notes. The 50 replicate determinations of the nitrate ion concentration in a particular water specimen are contained in $Table2\_1$.

0.51	0.51	0.51	0.50	0.51	0.49	0.52	0.53	0.50	0.47
0.51	0.52	0.53	0.48	0.49	0.50	0.52	0.49	0.49	0.50
0.49	0.48	0.46	0.49	0.49	0.48	0.49	0.49	0.51	0.47
0.51	0.51	0.51	0.48	0.50	0.47	0.50	0.51	0.49	0.48
0.51	0.50	0.50	0.53	0.52	0.52	0.50	0.50	0.51	0.51

The measurements are stored in the $.txt$ file called $Table2\_1.txt$. To import the file first assume that the working directory for R is already set as $E : \backslash Chemometrics \backslash Labs$. The data in file $Table2\_1.txt$ is organized in 5 rows and 10 columns but this structuring is only done to match the table presented in the lecture notes and it is not meaningful for the analysis of the data. Hence we are not reading the file using the $read.table$ command which would produce a $data.frame$ structure with 5 rows and 10 column, and use instead the $scan$ command which will produce a vector with 50 elements.

$> Nitrate\ <-\ scan("Table2\_1.txt")$

To calculate the mean and the standard deviations of the numeric vector that stores the 50 repeated

measurements, use the $mean()$ and $sd()$ commands

$> \ mean(Nitrate)$

[1] 0.4998

$> \ sd(Nitrate)$

[1] 0.01647385

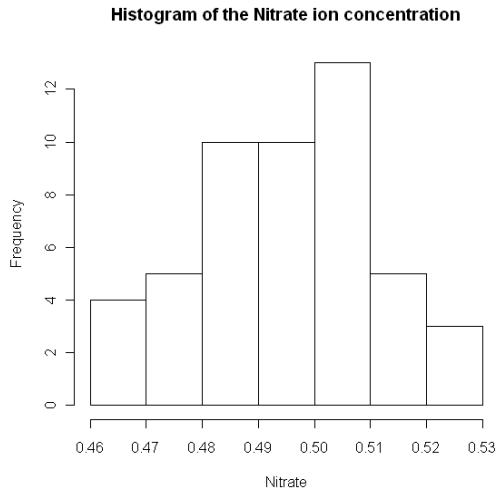
To appreciate the distribution of the results draw the histogram of the Nitrate data.

$> \ hist(Nitrate)$

Modify the title of the histogram obtained by R using the $main =$ "" option

$> \ hist(Nitrate, main = "Histogram \ of \ the \ Nitrate \ ion \ concentration")$

The histogram shows that the measurements are symmetrical around the mean with most of the



measurements clustered around the centre. This histogram can be easily approximated by a Normal density function.

3 The normal distribution

The density function of a Normal distribution with mean μ and standard deviation σ is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp - \frac{(x-\mu)^2}{2\sigma^2} \tag{1}$$

The density function of the Normal **standard** distribution with mean μ=0 and standard deviation σ=1 is a simplified version of equation (1)

$$f(x) = \frac{1}{\sqrt{2\pi}} exp - \frac{x^2}{2} \tag{2}$$

In R there are several function that allow us access to the Normal distribution. We can calculate the density, the distribution function, the quantile function and generate random numbers from the normal distribution with mean equal to *mean* and standard deviation equal to *sd*.

For the Normal standard distribution N(0,1) these functions are

dnorm(x, mean = 0, sd = 1)

pnorm(q, mean = 0, sd = 1)

qnorm(p, mean = 0, sd = 1)

rnorm(n, mean = 0, sd = 1)

3.1 dnorm

dnorm(x, mean = 0, sd = 1) evaluates the density function described in equation (2). To derive the value of the density function $f(x) = \frac{1}{\sqrt{2\pi}} exp - \frac{x^2}{2}$ at $x = -1$ use the **dnorm** function

```
> dnorm(-1, mean = 0, sd = 1)
[1] 0.2419707
```

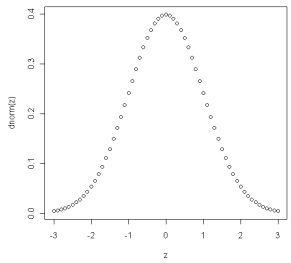
Similarly evaluate the density curve for $x = 0$, $x = 1$, $x = 1.5$ and $x = 2$. Observe the values of the dnorm function for $x = -1$ and $x = 1$. They are both equal because the density curve is symmetric about the mean value of 0. What is value of the density function at $x = -2$?

We can use **dnorm** to draw the curve described by equation (2) by calculating the density function for a sufficient large number of x values. First generate a sequence of equally distanced values for x within the (-3,+3) range using the *seq* command. The distance between points is 0.1, the first value in the sequence is -3 and the last value in the increasing sequence of x values is -3.

$> z <- seq(-3, 3, by = 0.1)$

$> plot(z, dnorm(z))$

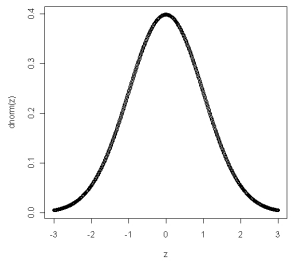
Plotting each of the elements of z in this sequence against their corresponding *dnorm* values results in a rough approximation of the Normal standard density function.



An improved approximation of the curve can be obtained if we calculate the density function for a more dense set of x values which we obtain by setting a smaller distance between the x points.

$> z <- seq(-3, 3, by = 0.01)$

$> plot(z, dnorm(z))$



3.2 pnorm

Given a number q, **pnorm** computes the probability that a normally distributed random value will be less than that number. This function calculates the cumulative distribution function.

The probability of a normal standard variable to take values **less** than -1.96 is calculated with:

$> \ pnorm(-1.96, mean = 0, sd = 1)$

> [1] 0.02499790

The probability of a normal standard variable to take values **greater** than +1.96 is calculated with:

$> \ 1 - pnorm(1.96, mean = 0, sd = 1)$

> [1] 0.02499790

The probability of a normal standard variable to take values less than 0 equals the probability of taking values greater than 0 since the area under the normal standard curve to the left hand side of 0 equals the area under the curve to the right hand side of 0. Check that both are equal to 0.5 .

3.3 qnorm

The next function we look at is **qnorm** which is the inverse of pnorm. The idea behind qnorm is that you give it a probability, and it returns the number whose cumulative distribution matches the probability. For example, if you have a normally distributed random variable with mean zero and standard deviation one, then if you give the function a probability it returns the associated Z-score. The Z-score for which the area to the left under the Normal standard variable equals 0.5 is 0.

$> \ qnorm(0.5, mean = 0, sd = 1)$

> [1] 0

The Z-score for which the area to the left under the Normal standard variable equals 0.025 is -1.96 .

$> \ qnorm(0.025, mean = 0, sd = 1)$

> [1] 0

3.4 rnorm

The **rnorm** function generates random numbers whose distribution is normal. The argument that you give it is n the number of random numbers that you want to generate, and it has optional arguments to specify the mean and standard deviation. For example $> \ rnorm(50, mean = 0, sd = 1)$ will generate 50 numbers from the normal standard deviation.

$> \ rnorm(3, mean = 0, sd = 1)$

$> \ [1] \ -0.5815100 \ 0.7661802 \ -1.0734987$

4 The *t* distribution

Just like for the Normal distribution we can calculate the density, the distribution function, the quantile function and generate random numbers from the t distribution with $n-1$ degrees of freedom. The functions can be used just like the commands for the normal distribution. One difference is that the commands assume that the values are normalized to mean zero and standard deviation one, so we have to use a little algebra to use these functions in practice. The other difference is that we have to specify the number of degrees of freedom. The commands follow the same kind of naming convention, and the names of the commands are **dt**, **pt**, **qt**, and **rt**. A few examples are given below to show how to use the different commands.

dt(x, df)

pt(q, df)

qt(p, df)

rt(n, df)

4.1 dt

We can plot the density function for the t distribution with 20 degrees of freedom, df=20 using the dt command.

$> \ x <- seq(-5, 5, by = .1)$

$> \ plot(x, dt(x, df = 20))$

4.2 pt

The cumulative probability distribution function pt can calculate the probability that a t distributed random value with 15 degrees of freedom will be less than -1.

$> \ pt(-1, \ df = 15)$

> [1] 0.1665851

Also use pt to calculate the probability that a t distributed random value with 15 degrees of freedom will be less than +1.

$> \ pt(1, \ df = 15)$

> [1] 0.833415

Or calculate the probability that a t distributed random value with 15 degrees of freedom is **greater** than +1.

$> \ 1 \ - pt(1, \ df = 15)$

> [1] 0.1665851

4.3 qt

The inverse cumulative probability distribution function **qt** calculates the quantile for which the probability that a t distributed random value with 10 degrees of freedom is 0.025.

$> \ qt(0.025, \ df = 10)$

> [1] −2.228139

4.4 rt

Random numbers can be generated according to the t distribution with 20 degrees of freedom using the **rt** command.

> $rt(3,\ df = 20)$

> [1] 0.7548628 0.4053671 −0.5883879

5 Confidence intervals in R

Finding a 95% confidence interval for the mean for the nitrate ion concentrations in Table 2.1.

First read the data

> $Nitrate <- scan("Table2\_1.txt")$

then compute the length of the sample computing confidence interval.

> $n = length(Nitrate)$

Derive the sample mean and store it in a variable "sample.mean".

> $sample.mean < -mean(Nitrate)$

Derive the sample standard deviation and store it in a variable "sample.sd".

> $sample.sd < -sd(Nitrate)$

Derive the standard error of the mean as $Z_{value} * \frac{sample.sd}{\sqrt{n}}$.

> $error < -qnorm(0.975) * sample.sd/sqrt(n)$

Calculate the lower and upper limits of the confidence intervals.

> $CI.lower <- sample.mean - error$

> $CI.upper <- sample.mean + error$