R Programming

26-6-18

- No declaration for variables
- Assignment operators are = and -> eg n=10 or n<-10
- Case sensitive variables i.e. a and A are different
- t*n also *a<-print(t*n)*
- Concatenation [forms an array] eg. *data1<-c(1,2,3,4)* which gives 1 2 3 4 or data2<-2:5 which gives 2 3 4 5 or data3<-1:10-1 which gives 0 1 2 3 … or data4<-1:(5-1) which gives 1 2 3 4 or *data5 = seq(1,5,0.5)* which gives 1.0 1.5 2.0 …
- To scan values for an array, *z <- scan()*
- *gl(3,5)* to generate levels (number of levels from 1, number of replications) or gl(2,1,label=c("Male","Female")) or gl(3,5,length=30)
- Sorting numbers i.e. conc=c(204,34,65,3,90) and *sort(conc)* or *sort(conc, decreasing=TRUE)*

10-8-18

17-8-18

-Vector is one-dimensional matrix.
For homogenous elements, its called **atomic vector** and for heterogenous, its called **list**.

-Lists
list()
x<-list(“a”=2.5,”b”=True)
a and b are tags, not necessary
str(x) gives structure

-**Data frame** is a table or a two-dimensional array-like structure in which each column contain values of one variable and each row contains one set of values from each column.
>Column names should be non-empty
>Unique row names
>Data can be numeric, factor or char type
>Each column should have same number of items

data.frame() to create it and *str()* to get structure

For example:
emp.data<-
data.frame(emp_id=c(1:5),emp_name=c("Ram","Arun","Sam","Jerry","Peter"),salary=c(625,500,450,600,700),start_day=as.Date(c("2012-01-01","2013-09-23","2014-11-15","2014-05-11","2015-03-27")),stringsAsFactors=FALSE)

result<-data.frame(emp.data$emp_name,emp.data$salary) the two columns
result2<-emp.data[1:2,] first two rows n all columns
result3<-emp.data[c(3,5),c(2,4)] for 3rd n 5th row with 2nd and 4th column

emp.data$dept<-c(“IT”,”Operations”,”IT”,”HR”,”Finance”) to add column called dept

emp.newdata<-
data.frame(emp_id=c(6:8),emp_name=c("Rashmi","Pranab","Tushar"),salary=c(650,550,7
50),start_day=as.Date(c("2013-01-07","2010-09-27","2015-03-15")),dept=c("IT","Operatio
ns","Finance"),stringsAsFactors=FALSE) to add new rows
emp.finaldata<-rbind(emp.data,emp.newdata) to bind both the data sets

-**Importing data**
>Excel/D limiters/text file/SPSS
>For excel, .xls or .xlx
>Find directory path *getwd()*
>To set directory *setwd()*
>*"utils"* is the package already stored into R for importing .csv files
>Use *dir()* to list files in working directory
>*read.table() / read.csv() / read.csv2()* to import a .csv file
>*df<-read.table("Data.csv",header=TRUE)* so that first row isn't taken as data but a header
>*install.packages("readxl")* for importing Excel files..choose 0 Cloud
>*library("readxl")* to load
>*read_excel("Marks.xlsx")* to read it

31-7-18
-**Loading datasets and slicing**
>*library(datasets)* to load the datasets
>*library(help="datasets")* to see all the datasets available
>iris (type the dataset name to get data)
>*head(iris)* to see column name with first 6 data values
>*names(iris)* to see column names
>*tail(iris)* to see last 6 values
>names(iris)<-toupper(names(iris))
>*summary(iris)* for mean, standard deviation etc.
> *str(iris)* for structure
>sd(iris) for standard deviation of numeric data
>library(dplyr) to load for slicing
>dplyr is display layer by layer
>v <- *filter*(iris, species == "virginica")
>two <- filter(iris, species == "setosa",sepal.length>4) for two conditions
>n<-mutate(iris,new=sepal.width>0.5*sepal.length) and > filter(n,new=="TRUE") or Sum(n$new) to count
>sort<-arrange(iris,petal.length) to arrange

14-8-18
Graphics on R
-eg women and plot(women)
> #function
> x=seq(-2,2)
> y=x^2
> #edgy graph
> plot(x,y,type="l",xlab="X axis",ylab="Y axis",main="Parabola",col="red")
> #better
> sp<-spline(x,y) #spline interpolation of data pts
> lines(sp,col="blue")

```
> sp<-spline(x,y,n=20) #interpolation at n pts spanning[xmin,xmax]
> lines(sp,col="green")

>v<-c(7,12,28,3,41)
> t<-c(14,7,6,19,3)
> plot(v,type="o",col="red",xlab="Month",ylab="Rainfall",main="Rainfall chart")
> lines(t,type="o",col="blue")

> x<-c(1:5)
> y<-x #create some data
> par(pch=22,col="blue") #plotting symbol and color (partition n parching)
> par(mfrow=c(2,4)) #all plots on one page (2 rows n 4 columns)
> opts=c("p","l","o","b","c","s","S","h")
> for(i in 1:length(opts))
+ {
+ heading=paste("type=",opts[i])
+ plot(x,y,main=heading)
+ lines(x,y,type=opts[i]) (in same graph)
+ }

> #scatterplot
> attach(mtcars)
> head(mtcars)
> plot(wt,mpg,main="Weight/MPG graph",xlab="Car weight (lbs)",ylab="Miles per
gallon",pch=19)

> #Kernel density plot
> d<-density(mtcars$mpg) #Kernel density estimates
> plot(d)
> #Filled density plot
> d<-density(mtcars$mpg)
> plot(d,main="Kernel density of Miles per gallon")
> polygon(d,col="red",border="blue")
```

-Boxplot(X)
If X is a vector, the vector elements are the heights of the bars in the plot. If X is a matrix, the matrix columns are the heights of the bars in the plot stacked after the first bar. If argument beside=TRUE, then values are juxtaposed not stacked and horiz=TRUE creates horizontal barplot.

4-9-18
Mean,Median and Mode
-x<-c(1,2,3)
-mean(x) and *Median(x)*
-x[x<3] gives a part of vector
-For mode,*> xr=table(age)*
> mode=which(xr==max(xr))
-For freq distribution, *> x=c(0,1,2,3)*
> f=c(8,11,5,1)
> y=rep(x,f)
> mean(y)
OR

> *mean=sum(y)/length(y)* or
> *median(y)*
-For continuous distr, 145-150, 150-155,…
>*mid=seq(147.5,182.5,5)* will give midpoints of the intervals
> f=c(4,6,28,58,64,30,5,5) for frequencies
> *fr.distr=data.frame(mid,f)* to create data frame
> *fr.distr*
>*mean=sum(mid*f)/sum(f)* will give mean

> *cl=cumsum(f)* for cumulative
> *n=sum(f)*
> *ml=min(which(cl>=n/2))* to get position of median class so halfway
> *ml*
> *h=5* which is the class width
> *h*
> *freq=f[ml]* to get frequency of median class
> freq
> *c=cl[ml-1]* to get previous class' cumulative frequency
> c
> *l=mid[ml]-h/2*
> *l*
> *median=l+(((n/2)-c)/freq)*h* for formula
> *median*

> *modalclass=which(f==max(f))*
> *modalclass*
> *fm=f[modalclass]*
> *fm*
> *f1=f[modalclass-1]* for prev freq
> *f1*
> *f2=f[modalclass+1]* for next freq
> *f2*
> *l=midx[modalclass]-h/2*
> *l*
> *mode=l+((fm-f1)/(2*fm-f1-f2))*h*
> *mode*

QUESTION:
> #The distribution of age of males at the time of marriage was as follows:
> #Age 18-20 20-22 22-24 24-26 26-28 28-30
> #No. 5 18 28 37 24 22
> #Find at the time of marriage, average age, modal age and median age.
> mid=seq(19,29,2)
> mid
[1] 19 21 23 25 27 29
> f=c(5,18,28,37,24,22)
> f
[1] 5 18 28 37 24 22
> fr.distr=data.frame(mid,f)
> fr.distr
  mid f
1 19 5

```
2  21 18
3  23 28
4  25 37
5  27 24
6  29 22
> mean=sum(mid*f)/sum(f)
> mean
[1] 24.83582

> cl=cumsum(f)
> cl
[1]   5  23  51  88 112 134
> n=sum(f)
> n
[1] 134
> ml=min(which(cl>=n/2))
> ml
[1] 4
> h=2
> h
[1] 2
> freq=f[ml]
> freq
[1] 37
> c=cl[ml-1]
> c
[1] 51
> l=mid[ml]-h/2
> l
[1] 24
> median=l+(((n/2)-c)/freq)*h
> median
[1] 24.86486

> modalclass=which(f==max(f))
> modalclass
[1] 4
> fm=f[m]
Error: object 'm' not found
> fm=f[modalclass]
> fm
[1] 37
> f1=f[modalclass-1]
> f1
[1] 28
> f2=f[modalclass+1]
> f2
[1] 24
> l=mid[modalclass]-h/2
> l
[1] 24
> mode=l+((fm-f1)/(2*fm-f1-f2))*h
```

> mode
[1] 24.81818

11-9-18
Dispersion
Pdf uploaded

Correlation and Regression
-To plot graph,
> x<-c(15,25,35,45,55,65)
> y<-c(302,193,185,198,224,288)
> plot(x,y,main="Average age vs Time spent in the library",xlab="Age",ylab="Time spent in the library",col="red")
-To find correlation using formula
>r=var(x,y)/sqrt(var(x)*var(y))
OR directly
> cor(x,y)
-> cor.test(x,y,method="pearson") will test the Pearson's correlation where H0: There is no correlation and H1: There is correlation
p value
Reject if p< alpha and accept if p>alpha where alpha is fixed (say 0.05)

Check ppt for regression
> weight=c(15,26,27,25,25.5,27,32,18,22,20,26,24)
> bmi=c(13.35,16.12,16.74,16.00,13.59,15.73,15.65,13.85,16.07,12.8,13.65,14.42)
> cor(weight,bmi)
[1] 0.5790235
> model<-lm(bmi~weight)
> summary.lm(model)
-Multiple R-square is the coefficient of determination which gives how much independent variable influences dependent variable and Adjusted R-square gives standardized R square
H0:Model does not fit and H1: Model fits as per p-value

18-9-18
t.test()
-For two independent groups t.test(y1,y2)
For paired, t.test(y1,y2,paired=TRUE)
For popln comparison, t.test(y1,mu=2)
->x=c(….)
>t.test(x,alternative="greater",mu=0.3) for mentioning alternate hypothesis or *alt="less"* the mean in the output is sample mean
-qt(0.975,9) for table value at 97.5%
-t.test(alb, var.equal=FALSE,paired=FALSE) so df is in decimals as variances are not equal..by default variances are equal
-**F test**
>var.test(x,y)

19-9-19
Chi-square test
-Enter as matrix

> data<-matrix(c(35,42,61,48,51,68),ncol=3,byrow=T) Here, brow means data is entered row-wise, by default it is column wise

```
> data
     [,1] [,2] [,3]
[1,]   35   42   61
[2,]   48   51   68
> chisq.test(data)
```

-