L(1) + 5 pages in R

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
% * %	لضرب الصفوفات
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2
pi	π

gamma(x) ← Γ(x)

$\leftarrow \left[\frac{5}{2}\right]$ صحيح

factorial(x) x!

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if x is TRUE

Numeric Functions

Function	Description
abs(*x*)	absolute value
sqrt(*x*)	square root
ceiling(*x*)	ceiling(3.475) is 4
floor(*x*)	floor(3.475) is 3
trunc(*x*)	trunc(5.99) is 5
round(*x*, digits=*n*)	round(3.475, digits=2) is 3.48
signif(*x*, digits=*n*)	signif(3.475, digits=2) is 3.5
cos(*x*), sin(*x*), tan(*x*)	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x*) =ln (x)	natural logarithm
log10(*x*)	common logarithm
exp(*x*)	e^x $= e^x$
log(x, base)	log x to any base..

Function	Description

dnorm(*x*)	normal density function (by default m=0 sd=1)
	# plot standard normal curve
	x <- pretty(c(-3,3), 30)
	y <- dnorm(x)
	plot(x, y, type='l', xlab="Normal Deviate", ylab="Density", yaxs="i")

pnorm(*q*)	cumulative normal probability for q (area under the normal curve to the right of q) pnorm(1.96) is 0.975
qnorm(*p*)	normal quantile. value at the p percentile of normal distribution qnorm(.9) is 1.28 # 90th percentile
rnorm(*n*, m=0,sd=1)	n random normal deviates with mean m and standard deviation sd. #50 random normal variates with mean=50, sd=10 x <- rnorm(50, m=50, sd=10)

dbinom(*x*, *size*, *prob*)
pbinom(*q*, *size*, *prob*)
qbinom(*p*, *size*, *prob*)
rbinom(*n*, *size*, *prob*)
	binomial distribution where size is the sample size
	and prob is the probability of a heads (pi)
	# prob of 0 to 5 heads of fair coin out of 10 flips
	dbinom(0:5, 10, .5)
	# prob of 5 or less heads of fair coin out of 10 flips
	pbinom(5, 10, .5)

dpois(*x*, *lamda*)
ppois(*q*, *lamda*)
qpois(*p*, *lamda*)
rpois(*n*, *lamda*)
	poisson distribution with m=std=lamda
	#probability of 0,1, or 2 events with lamda=4
	dpois(0:2, 4)
	# probability of at least 3 events with lamda=4
	1- ppois(2,4)

dunif(*x*, min=0, max=1) **punif(*q*, min=0, max=1)** **qunif(*p*, min=0, max=1)** **runif(*n*, min=0, max=1)**	uniform distribution, follows the same pattern as the normal distribution above. #10 uniform random variates x <- runif(10)

Function	Description
mean(*x*, trim=0, na.rm=FALSE)	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)
sd(*x*)	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(*x*)	median
quantile(*x*, *probs*)	quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
range(*x*)	range
sum(*x*)	sum
diff(*x*, lag=*1*)	lagged differences, with lag indicating which lag to use
min(*x*)	minimum
max(*x*)	maximum
scale(*x*, center=TRUE, scale=TRUE)	column center or standardize a matrix.

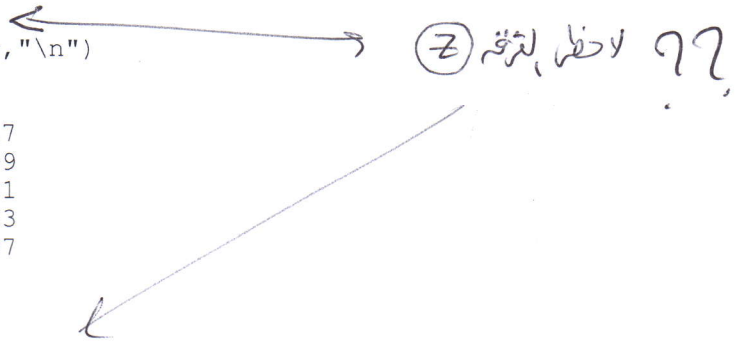
Other Useful Functions

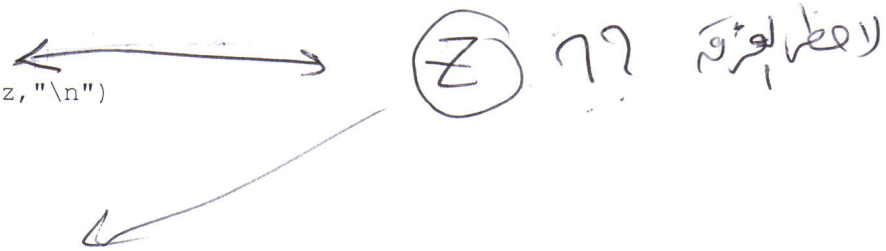
Function	Description
seq(*from* , *to*, *by*)	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)
rep(*x*, *ntimes*)	repeat *x n* times y <- rep(1:3, 2) # y is c(1, 2, 3, 1, 2, 3)
cut(*x*, *n*)	divide continuous variable in factor with *n* levels y <- cut(x, 5)

```
> # 1) Mathematics   as given in the powe point
> x=2
> y=9
> z=x*y
> z
[1] 18
> print(z)
[1] 18
> K=c(1,2,5,3,10)

> K
[1]  1  2  5  3 10

> # print used for printing one variable
> # cat is used for printing more than one variable
> cat(x,y,z,K)
2 9 18 1 2 5 3 10>
> cat(x,y,z,K,"\n")
2 9 18 1 2 5 3 10

> # descriptive statistics

> #mean(x)
> # var(x)
> # sd(x)
> # median(x)
> #range(x)
> # Summary(x)   gives minimum  Q1     median   mean  Q3   max
> # coeffeient of variation CV= sd(x)/mean(x)

> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

> cv=sd(K)/mean(K)
> print(cv)
[1] 0.8485014

> # الاسهم تعطينا الاوامر السابقة او التالية
> # لفتح نافذة جديدة  file ---New script---تظهر لنا نافذة R Editor----نكتب بها الاوامر ثم نحدد
الاوامر حسابها ونضغط Ctrl+R
> # نجدها تم نقلها لنافذة R console وتم الحصول على النتائج

> x=2
> y=3
> z=x*y
> print(z)
[1] 6

> # 3) Loop  هو استخدام الاقواس { }  وبينهما الامر المطلوب تكراره
>
> for(i in 1:10)
+ {
+ print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
> #  لحفظ الملف نقف بالمؤشر داخل الصفحة ومن file----save as
> # لفتح ملف قديم file-----open script
> for(i in 0:10)
+ {
+ cat(i,i^2,i*3)
+ }
0 0 01 1 32 4 63 9 94 16 125 25 156 36 187 49 218 64 249 81 2710 100 30> for(i in 0:10)
+ {
1 cat(i,i^2,i*3)
```

```
+ cat(i,i^2,i*3,"\n")
+ }
0 0 0
1 1 3
2 4 6
3 9 9
4 16 12
5 25 15
6 36 18
7 49 21
8 64 24
9 81 27
10 100 30
> for(i in 0:10)
+ {
+ cat(i,i^2,i*3,"\n")
+ }
0 0 0
1 1 3
2 4 6
3 9 9
4 16 12
5 25 15
6 36 18
7 49 21
8 64 24
9 81 27
10 100 30
> #  "\n"  معناها ضع النتائج بنفس السطر
> for(i in 0:10)
+ {
+ x1=i+2
+ x2=sqrt(i)
+ x3=i^3+2*i
+ cat(i," ",x1," ",x2," ",x3,"\n")
+ }
0   2   0   0
1   3   1   3
2   4   1.414214   12
3   5   1.732051   33
4   6   2   72
5   7   2.236068   135
6   8   2.44949   228
7   9   2.645751   357
8   10   2.828427   528
9   11   3   747
10   12   3.162278   1020
> #  " "  لوضع مسافة بين الاعمدة لتكون متباعدة
>
> # Distributions
>
> ## dnorm------calculate normal f(x)
> ## Pnorm------          normal F(x) CDF
> ## qnorm------          normal inverse (find the value of x)
> ## rnorm------          normal random sample
>

> ## Examples
>
> # if X has N(2,9). find f(x=1.5), F(0.75), p(x> 1.5)
> # f(1.5)=dnorm(1.5,2,sqrt(9))

> dnorm(1.5,2,sqrt(9))
[1] 0.1311466

> # F(0.75)=pnorm(0.75,2,3)
> pnorm(0.75,2,3)
[1] 0.3384611

> ## p(x>1.5)=1-pnorm(1.5,2,3)
> 1-pnorm(1.5,2,3)
[1] 0.5661838

> ##  if x has standard normal N(0,1), we need  not to put the values 0, 1 in the command as
follows:
```

```
> ## let x has N(0,1) , find p(x<2) and find a if p(x<a)=0.975
> # P(x<2)=pnorm(2)  and a= qnorm(0.975)

> pnorm(2)
[1] 0.9772499

> qnorm(0.975)
[1] 1.959964

> ### find random sample from normal (1,4)has size 10
> rnorm(10,1,2)
 [1] -1.0219220 -2.2359595 -0.9670488  1.7617904  2.6505148  0.2426578 -0.8236448  2.3430551
4.9988127 -0.2156232



> rnorm(10)
 [1] -1.3424810 -1.1931046 -0.4511279 -3.6477612  0.0843024 -0.9119739  1.3767376 -0.7590830
1.4619257  0.0967894
> ####   to construct standard normal table, we have to use loop

> for(z in 0:34)
+ {
+ Z=z/10
+ pz=pnorm(z)
+ cat(Z," ",pz,"\n")
+ }
0   0.5
0.1   0.8413447
0.2   0.9772499
0.3   0.9986501
0.4   0.9999683
0.5   0.9999997
0.6   1
0.7   1
0.8   1
0.9   1
1   1
1.1   1
1.2   1
1.3   1
1.4   1
1.5   1
1.6   1
1.7   1
1.8   1
1.9   1
2   1
2.1   1
2.2   1
2.3   1
2.4   1
2.5   1
2.6   1
2.7   1
2.8   1
2.9   1
3   1
3.1   1
3.2   1
3.3   1
3.4   1

> options(digits=4)
>
> for(z in 0:34)
+ {
+ Z=z/10
+ pz=pnorm(Z)
+ cat(Z," ",pz,"\n")
+ }
0   0.5
0.1   0.5398
0.2   0.5793
0.3   0.6179
```





```
0.4   0.6554
0.5   0.6915
0.6   0.7257
0.7   0.758
0.8   0.7881
0.9   0.8159
1   0.8413
1.1   0.8643
1.2   0.8849
1.3   0.9032
1.4   0.9192
1.5   0.9332
1.6   0.9452
1.7   0.9554
1.8   0.9641
1.9   0.9713
2   0.9772
2.1   0.9821
2.2   0.9861
2.3   0.9893
2.4   0.9918
2.5   0.9938
2.6   0.9953
2.7   0.9965
2.8   0.9974
2.9   0.9981
3   0.9987
3.1   0.999
3.2   0.9993
3.3   0.9995
3.4   0.9997


> for(z in 0:34)
+ {
+ Z=z/10
+ p=pnorm(Z)
+ p1=pnorm(Z+0.01)
+ p2=pnorm(Z+0.02)
+ p3=pnorm(Z+0.03)
+ cat(Z,"  ",p," ",p1," ",p2,"  ",p3,"\n")
+ }
0    0.5   0.504   0.508    0.512
0.1    0.5398   0.5438   0.5478    0.5517
0.2    0.5793   0.5832   0.5871    0.591
0.3    0.6179   0.6217   0.6255    0.6293
0.4    0.6554   0.6591   0.6628    0.6664
0.5    0.6915   0.695   0.6985    0.7019
0.6    0.7257   0.7291   0.7324    0.7357
0.7    0.758   0.7611   0.7642    0.7673
0.8    0.7881   0.791   0.7939    0.7967
0.9    0.8159   0.8186   0.8212    0.8238
1    0.8413   0.8438   0.8461    0.8485
1.1    0.8643   0.8665   0.8686    0.8708
1.2    0.8849   0.8869   0.8888    0.8907
1.3    0.9032   0.9049   0.9066    0.9082
1.4    0.9192   0.9207   0.9222    0.9236
1.5    0.9332   0.9345   0.9357    0.937
1.6    0.9452   0.9463   0.9474    0.9484
1.7    0.9554   0.9564   0.9573    0.9582
1.8    0.9641   0.9649   0.9656    0.9664
1.9    0.9713   0.9719   0.9726    0.9732
2    0.9772   0.9778   0.9783    0.9788
2.1    0.9821   0.9826   0.983    0.9834
2.2    0.9861   0.9864   0.9868    0.9871
2.3    0.9893   0.9896   0.9898    0.9901
2.4    0.9918   0.992   0.9922    0.9925
2.5    0.9938   0.994   0.9941    0.9943
2.6    0.9953   0.9955   0.9956    0.9957
2.7    0.9965   0.9966   0.9967    0.9968
2.8    0.9974   0.9975   0.9976    0.9977
2.9    0.9981   0.9982   0.9982    0.9983
3    0.9987   0.9987   0.9987    0.9988
3.1    0.999   0.9991   0.9991    0.9991
3.2    0.9993   0.9993   0.9994    0.9994
```

```
3.3    0.9995    0.9995    0.9995    0.9996
3.4    0.9997    0.9997    0.9997    0.9997
>
```

> ### another method(code 2) → تعتبر الأفضل من الطريقة السابقة

```
> i=seq(0,3.49,by=0.01)
> p=pnorm(i)
> m=matrix(p,ncol=10,byrow=TRUE)
> rownames(m)=seq(0,3.4,by=0.1)
> colnames(m)=seq(0,0.09,by=0.01)
> options(digits=4)
> m
```

(الصفوف 0.1 — الأعمدة 0.01 — 4 أرقام بعد العلامة العشرية)

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998