IMS
INSTITUTE OF MATHEMATICAL SCIENCES

Claremont
GRADUATE UNIVERSITY

R Tutorial for Statistical Learning

Qidi Peng

1: Read, create and save data

Code	Comments	Results
a<-4; a=4; a	Generate a value. Define "a" by 4. The same. Output the value of a (by pressing "enter" key)	a [1] 4
c(1,2,3)	Generate a vector manually	[1] 1 2 3
1:5	Generate an increasing sequence of integers.	[1] 1 2 3 4 5
seq(from = 4.5, to = 2.5, by = -0.5)	Generate a decreasing sequence, starting from 4.5, ending at 2.5, with mesh -0.5 (by default, by=1).	[1] 4.5 4.0 3.5 3.0 2.5
rep(3,4)	Generate a constant sequence with length 4	[1] 3 3 3 3

matrix(1:8,nrow=2,ncol=4,byrow=TRUE)	Generate a 2*4 matrix with data 1 to 8, by putting the digits one by one in row (by default).	[,1] [,2] [,3] [,4] [1,] 1 2 3 4 [2,] 5 6 7 8
x=matrix(1:8,nrow=2,ncol=4,byrow=FALSE); x	Generate a 2*4 matrix with data 1 to 8, by putting the digits one by one in column.	x [,1] [,2] [,3] [,4] [1,] 1 3 5 7 [2,] 2 4 6 8
y=matrix(c("a", "b", 1, 2), nrow = 2); y	Generate matrix where the coefficients are not values. Notice that here R regards “1” and “2” as symbols, not values.	y [,1] [,2] [1,] "a" "1" [2,] "b" "2"
colnames(y) = c("char", "num"); y	One assigns the column’s names.	y char num [1,] "a" "1" [2,] "b" "2"
rownames(y) = c("Sept.15", "Sept.16"); y	One assigns the row’s names.	y char num Sept.15 "a" "1" Sept.16 "b" "2"
Age=c(23,21,22); Name=c("Jin","Kelly","Choi"); Gender=c("M","F","F"); Data=data.frame(Name,Age,Gender); Data=edit(Data)	data.frame() functions are used to construct data set by typing in the values of each variable and assigning each vector its corresponding variable name. Manually import and change data in the data set “Data”.	Name Age Gender 1 Jin 23 M 2 Kelly 21 F 3 Choi 22 M

M454_Grades =read.table("C:/Users/pengq/Desktop/M454.txt"); colnames(M454_Grades)=c("Name","Grades"); M454_Grades	Import data from a text file using read.table(directory/filename.txt). One has to create this .tex fle first. .	Names Grades 1 1 100 2 2 100 3 3 100 4 4 99 5 5 99 6 6 100 7 7 98 8 8 99 9 9 Absent 10 10 89
M454_Grades[1,]	Print the first row.	Name Grades 1 1 100
M454_Grades[,2]	Print the second column.	[1] 100 100 100 99 99 100 98 99
M454_Grades[9,2]	Print the element in Row 9, Column 2.	Absent 89 [1] Absent
read.table("website")	You can also read table from website.	

fpe =
read.table("http://data.princeton.edu/wws509/datasets/effort.dat");
fpe;
fpe=edit(fpe);
	Examples: read data from website
	
	Read the table.
	A workable excel.
		> fpe
		setting effort change
		Bolivia 46 0 1
		Brazil 74 0 10
		Chile 89 16 29
		Colombia 77 16 25
		CostaRica 84 21 29
		Cuba 89 15 40
		DominicanRep 68 14 21

		Ecuador 70 6 0 ElSalvador 60 13 13 Guatemala 55 9 4 Haiti 35 3 0
save(), sink(), write.table() save(fpe,file="data.Rda"); load("data.Rda")	Save data and variables Example: save the data.frame fpe. Load saved data.frame fpe.	
q()	Quit and save workspace (useful when too time consuming)	
help(write.table)	Use help to get the tutorial of functions	

2. Create functions

Codes	Comments	Results
Summation=function(x,y){x+y};	Create a function Summation(x,y)=x+y.	[1] 8

Summation(3,5)		
Product=function(a,b,c){z=a*b+c;z}; Product(2,5,7)	Create a function Product(a,b,c)=ab+c.	[1] 17
fix(arrangement); function(n,k){factorial(n)/factorial(n-k)}; arrangement(3,3)	Create a function arrangement(n,k)=n!/(n-k)!	[1] 6
arrangement=edit(arrangement); fix(arrangement);	Edit the existing function. The same.	
fix(Taylor_expansion); function(n,x) {a=1;for (i in 1:n){a=a+x^i/factorial(i)};a}; Taylor_expansion(100,3) exp(3)	Use “for loops” to create Taylor expansion of exponential of x, of order n: $1+x+x^2/2!+...+x^n/n!$ Approximate e^3 with n=100 True value of e^3	[1] 20.08554 [1] 20.08554
fix(parity); function(x){for(i in 1:length(x)){if (x[i]/2==floor(x[i]/2)){print(x[i]); print(" is even")} else {print(x[i]);print("is odd")};};}; parity(1:8)	Use “if else” to create function, which tells the parity of any real valued vector.	[1] 1 [1] "is odd" [1] 2 [1] " is even" [1] 3 [1] "is odd" [1] 4 [1] " is even" [1] 5 [1] "is odd" [1] 6 [1] " is even" [1] 7 [1] "is odd" [1] 8 [1] " is even"

fix(sum_square);
function(N){i=1;y=0;
while(i<=N){
    y=i*i+y;
    i=i+1
}};
 	
sum_square(5)
	Use “while loop” to generate the sum of squares: $1+2^2+3^2+...+n^2$.
	
	Determine $1+2^2+3^2+...+5^2$
		[1] 55

3+5, 4*8, 5*pi, exp(5), sin(pi), cos(pi), (c(1,2))^2, abs(-3), c(4,5)/3, complex(real=3,imaginary=5), c(1,2)%*%c(1,2)	Some useful functions.	

3. Plot functions

Codes	Comments	Results

fix(linear);
function(x,a,b){a*x+b};
x=seq(0,1,0.01);
plot(x, linear(x,2,-2),
col=”blue”,typ=”p”,xlab=”x”,ylab=”f(x)”);
 	
abline(h=-0.5,v=0.3,col=”red”);
 	
legend("bottomright",title="Linear function",
c("f","line"),col=c(26,33),lty=c(3,1))
	Illustrate the trajectory of linear function f(x)=ax+b.
	Create function.
	Input values of x.
	Plot the corresponding values of f(x)
	Color =(blue, red, green...), line charts type=
	(l,p,o,b,s,h,..), xlab(name of xline), ylab(name of yline)
	Add lines without erasing ex plots.
	
	Add legend (these are underlines commands helping to
	make footnotes of the plot).	

 windows() quartz() split.screen(c(2,2)) screen(1) plot(1:5,1:5) screen(2) plot(1:5,-(1:5)) screen(3) plot(1:5,2:6) screen(4) plot(1:5,-(2:6))	Open new windows for graphics. For windows system. For Mac system. Split screen into 2*2 small windows. Plot in the first window (topleft) Plot in the second window (topright) Plot in the third window (bottomleft) Plot in the fourth window (bottomright)	 [1] 1 2 3 4

pdf(“name of your plot”);
png();
jpeg();
	You can use “save” button to directly save your
	pictures or use these commands to save them.	

4: Generating random variables

Code	Comments	Results
runif(1);	Generate a uniform random variable over [0,1].	[1] 0.6197008
runif(4);	Generate a sequence of 4 independent uniform random variables over [0,1].	[1] 0.23239163 0.87730174 0.06837577 0.54853920
runif(1,3,4)	Generate a uniform random variable over [3,4].	[1] 3.892476

dunif(0.5)	The density function of the uniform distribution over [0,1], at 0.5.	[1] 1
punif(0.5)	The probability distribution function of the uniform distribution over [0,1], at 0.5.	[1] 0.5
floor(runif(1,1,7))	Generate a uniform random number over {1,2,3,4,5,6} (consider a die of 6 faces).	[1] 2
sample(0:1, 10, replace=T)	Toss a coin independently 10 times.	[1] 1 1 0 1 0 0 0 1 0 1
sample(1:6, 10, replace=T)	Roll a die independently 10 times.	[1] 3 1 3 5 4 1 4 5 2 4
sample(1:9, 3, replace=FALSE)	Randomly and equally likely pick 3 numbers from numbers 1 to 9, without replacement.	[1] 7 4 6
sample(1:3,1,replace=T,c(1/4,1/2,1/4))	Generate a non-uniform random number over {1,2,3}, with probability distribution (1/4,1/2,1/4).	[1] 2
sample(10)	Permute randomly and equally likely numbers 1 to 10.	[1] 5 3 4 6 7 1 2 9 8 10
rbinom(1,1,0.5)	Sample of Bernoulli(0.5).	[1] 1
rbinom(1,3,0.5)	Sample of Binomial distribution Binomial(3,0.5).	[1] 1
rexp(1,0.5)	Sample of exponential distribution Exponential(0.5).	[1] 2.365289
rnorm(1)	Sample of standard normal N(0,1).	[1] 0.9059011
qnorm(0.975)	Quantile of standard normal at level 0.975.	[1] 1.959964
qt(0.975,18)	Quantile of 18 degrees of freedom student t-distribution at level 0.975.	[1] 2.100922
qf(0.95,10,8)	Quantile of (10,8) degrees of freedom F-distribution at level 0.95.	[1] 3.347163
mean();	Average of vector;	
var();	Sample variance of a vector (unbiased)	
median();	Median of vector;	
sum();	Summation of all elements of a vector	
cumsum();	All partial sums of a sequence (this gives a vector of the same size as the sequence)	
length();	Cardinal (number of elements, dimension) of vector	[1] 1 2 3

X=1:6; X[X<=3];	Extract all elements of X (in order) which are less or equal to 3.	
var(), sd(), sort(), sort(1:5,decreasing=TRUE), quantile(), sqrt(),median()	Help yourself to find the meaning of these operators for vectors.	

5: Linear regression

Code	Comments	Results
install.packages("ISLR");	Install the data set package ISLR, which includes the data we use in this class..	
library(ISLR);	Load ISLR.	
library(MASS);	Load MASS, which contains some exotic statistical functions which are not contained in the base functions of R For more information on this package, search it on website.	
fix(Boston);	Open data.frame “Boston” in MASS.	
names(Boston);	Check variable names of the columns.	
?Boston;	Check all information about the data set Boston.	
lm.fit=lm(medv~lstat,data=Boston);	Simple linear regression with Y=medv, X=lstat.	
lm.fit;	Watch the estimates!	Call: lm(formula = medv ~ lstat, data = Boston) Coefficients: (Intercept) lstat 34.55 -0.95
names(lm.fit);	Statistical properties contained in the linear regression.	
coef(lm.fit);	Least squares coefficient estimates.	
confint(lm.fit,level=0.95);	0.95 confidence interval of the real parameters beta0 and beta1.	
summary(lm.fit);	More detailed information on this linear regression.	
predict(lm.fit,data.frame(lstat=c(5,10,15)),interval="prediction",level=0.95);	0.95 prediction interval of the response variable Y=fit, with predictor X=5,10,15.	
plot(Boston$lstat,Boston$medv,pch=”+”,col=”blue”);	Plot (X,Y)=(lstat,medv);	
abline(lm.fit,lwd=3,col="red");	Add the regression line without erasing the ex-plots.	

lm.fit2=lm(medv~lstat+age,data=Boston);
 	
summary(lm.fit2);
lm.fit3=lm(medv~.,data=Boston);
 	
summary(lm.fit3).
	Multiple linear regression (here are 2 predictors X=(lstat,age)).
	Statistical information on the regression.
	Linear regression, Y=medv, X contains all other variables contained in the data set Boston.	

lm.fit4=lm(medv~lstat+I(lstat^2),data=Boston);
 	
summary(lm.fit4);
	Non linear regression (here one assumes $Y=beta_0+beta_1X+beta_2X^2+epsilon$).	

6: Classification

Code	Comments	Results

# Logistic regression 	
summary(Smarket);
 	
cor(Smarket);
 	
cor(Smarket[,-9]);
Smarket$Direction;
 	
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5 +Volume,data=Smarket,family=binomial);
 	
coef(glm.fit);
glm.probs=predict(glm.fit,type="response");
 	
contrasts(Smarket$Direction);
plot(glm.fit);
 	
glm.pred=rep("down",1250);
	
	Consider data set Smarket (stock market data) in the package ISLR.
	Produce correlation matrix of all variables contained in Smarket. Error detected when meeting qualitative data.
	
	Decline the last 9 columns (variables) of Smarket.
	The variable “Direction” in Smarket is qualitative.
	
	Multiple logistic regression with response “Direction” and predictors Lag1 to Volume. Observe that here the p-values are relatively large, so there is no clear evidence of a real association between Direction and those variables.
	Estimates of the coefficients.
	Predict the probability P(Y=1|X) for each training data X. To see the meaning of event Y=1, use contrasts().
	Convert numbers to dummy variable.
	The glm function also provides figure representations. It’s slow. Please wait and watch.
	Create a vector of class prediction based on whether
		
		Error in cor(Smarket) : ‘x’ must be numeric.

glm.pred[glm.probas>0.5]=("up");
table(glm.pred,Smarket$Direction);
 	
mean(glm==Smarket$Direction)
	the predicted probability of a market increase is greater than 0.5.
	Produce a confusion matrix in order to determine how many observations were correctly or incorrectly classified. The diagonal elements of the confusion matrix indicate correct predictions, while the off-diagonals represent incorrect predictions.
	Compute the fraction of correct predictions.	

# Discriminant analysis 	
lda.fit=lda(Direction~Lag1+Lag2,data=Smarket,subset=Smarket$train);
 	
lda.fit;
 	
plot(lda.fit);
 	
lda.pred=predict(lda.fit,Smarket);
 	
lda.class=lda.pred$class;
qda.fit=qda(Direction~Lag1+Lag2,data=Smarket,subset=Smarket$train);
	Linear discriminant analysis function from package MASS.
	The result shows 49.2% of the training observations correspond to days during which the market went down.
	Illustration It produces the linear discriminant, obtained by computing beta1*Lag1+beta2*Lag2 for each of the training observations.
	The prediction function returns 3 variables: class, probability of predictor belonging to this class; and linear discriminant.
	Check the classes for each training observation.
	Quadratic discriminant analysis by using the same training data set.	

# KNN 	
library(class);
train=(Smarket$Year<2005);
train.X=cbind(Smarket$Lag1,Smarket$Lag2)[train,];
 	
test.X=cbind(Smarket$Lag1,Smarket$Lag2)[!train,];
 	
train.Direction=Smarket$Direction[train];
 	
set.seed(1);
knn.pred=knn(train.X,test.X,train.Direction,k=3);
	The K-nearest neighbors function is contained in the package “Class”.
	Consider the historical data before 2005.
	Input 1: a matrix of predictors X associated with the training data;
	Input 2: a matrix of predictors X associated with the data we want to make predictions;
	Input 3: a vector of class labels Y.
	Input 4: the value of K (here K=3).
	The result is random, we fix it by using set.seed.	