Nicholas Boulanger

Beginner tutorial

1. (mtcars) is a data frame, according to the “class” function.
2. (precip) is a vector, because it has no dimensions but has length.
3. Using the “as” function, a data frame with only one data category can be converted to a matrix.
4. Precip[14]= Atlanta
5. <- Data.frame
6. Numeric. The “typeof” function reports “double”, which is numeric for my intents and purposes.
7. Mtcars[2,7] Mtcars [“Mazda RX4 Wag”,7] Mtcars [2,”qsec”] mtcars [“Mazda RX4 Wag”,”qsec”]
8. Precip[“Sacramento”]<-23, etc.
9. There are no trees with those traits. I used trees[1-3] to compare the data side-by-side.
10. 411.348

Intermediate tutorial- section 1

1. The “replace” function allows for a random grid to be produced with more slots than there are options. If replace=FALSE, the random grid cannot have 96 slots; it can only have one TRUE and one FALSE.
2. Using MyMatrix[which(MyMatrix)]<-1 produces numeric true/false.
3. Using the “apply” function, it is possible to select the row of interest, and determine whether its elements are true or false.

Intermediate tutorial- section 2

1. 16 times
2. Use the “sum” function in conjunction with MyMatrix and the column in question.
3. As before, but with the “prod” function instead of “sum”.
4. Use the “which” function to find all instances of 10, and use <- to turn them into twelves.
5. 33
6. First, you must convert MyMatrix into a data.frame. From there, you can use MyMatrix[12,] to manually convert the data into characters.
7. Rows 2 and 3 have a sum of greater than 70.

Advanced tutorial

1. Greetings<-function(Argument){print(Argument)}
2. Step one: divide the data.frame into subsets: > Setosa<-iris[which(iris[,"Species"]=="setosa"),]

> Virginica<-iris[which(iris[,"Species"]=="virginica"),]

> Versicolor<-iris[which(iris[,"Species"]=="versicolor"),]

Step two: write the function:

> IrisResults<-list(data=Setosa,Virginica,Versicolor)

>IrisSpecies<-function(iris){IrisResults}

3. All the code:

SepalLength<-iris[,1]

SepalWidth<-iris[,2]

PetalLength<-iris[,3]

PetalWidth<-iris[,4]

rows<-array(data=c(1:150))

Answer<-array(data=NA,dim=150)

IrisPetals<-function(iris){

for (Index in rows)

if (SepalWidth[Index]>=3.1)

Answer[Index]<-(sum((PetalLength[Index]),(SepalLength[Index])))

else if (SepalWidth[Index]<3.1)

Answer[Index]<-((SepalLength[Index])-(PetalLength[Index]))

return(Answer)

}

4. All the code: v4<-mtcars[which(mtcars[,"cyl"]==4),]

v6<-mtcars[which(mtcars[,"cyl"]==6),]

v8<-mtcars[which(mtcars[,"cyl"]==8),]

meanv4<-mean(v4[,1])

meanv6<-mean(v6[,1])

meanv8<-mean(v8[,1])

MPG<-function(number){

if (number==4){meanv4}

else if (number==6) {meanv6}

else if (number==8) {meanv8}}

1. Simply create a data.frame using six sampling vertices:

UrNotLucky<-function(go){one<-sample(c(1:69),size=1000000,replace=TRUE)

two<-sample(c(1:69),size=1000000,replace=TRUE)

three<-sample(c(1:69),size=1000000,replace=TRUE)

four<-sample(c(1:69),size=1000000,replace=TRUE)

five<-sample(c(1:69),size=1000000,replace=TRUE)

six<-sample(c(1:26),size=1000000,replace=TRUE)

data.frame(one,two,three,four,five,six)}

1. MyMatrix<-matrix(NA,nrow=1000000,ncol=6)

MyMatrix[ , 6] <- sample(c(1:26),1000000,replace=TRUE)

for (index in 1:1000000) {

MyMatrix[index,c(1,2,3,4,5)] <- sample(c(1:69),5,replace=FALSE)

}

MyNumbers<-c(1,2,3,4,5,6)

didiwin<-function(MyNumbers){

for (index in 1:dim(MyMatrix)[1]) {

GameOutcome[index]<-MyMatrix[index,]==MyNumbers

}}

GameOutcome<-array(data=NA,dim=1000000)

which(GameOutcome)

Expert Problem Set

1. > mean(precip)

[1] 34.88571

> median(precip)

[1] 36.6

> sd(precip)

[1] 13.70665

1. (precip) is best viewed as a histogram to better see the central precipitation tendencies.
2. Length(precip)=70

RandomNormal<-(rnorm(70,mean=34.88571,sd=13.70665))

1. compareprecip<-function(RandomNormal,precip,Iterations=100){

+ Barrel<-c(RandomNormal,precip)

+ ReplicatedMeans<-array(data=NA,dim=Iterations)

+ for (Counter in 1:Iterations) {

+ NewRandomNormal<-sample(Barrel,length(RandomNormal),replace=TRUE)

+ Newprecip<-sample(Barrel,length(precip),replace=TRUE)

+ RandomNormalMean<-mean(NewRandomNormal)

+ precipMean<-mean(Newprecip)

+ ReplicatedMeans[Counter]<-RandomNormalMean-precipMean}

+ return(ReplicatedMeans)}

> ReplicationResults<-compareprecip(RandomNormal,precip,100)

> length(which(ReplicationResults>(0.1946061)))

49%

1. The function I developed in question 4 is a relatively good test of whether or not the two distributions are identical, because they really are quite different.