Data preprocessing

By Saba Akram

2/16/2021

cars2 <- read.csv("D:/M.Sc in Banking and Financial Analytics/Sem 3/Data Analytic and Machine learning/Data Mining and Predictive Analysis/Data sets/cars2.txt", header=TRUE, stringsAsFactors=TRUE)  
  
## Missing data  
  
## We look at four variables from cars  
cars.4var<-cars2[,c(1,3,4,8)]  
  
head(cars.4var)

## mpg cubicinches hp brand  
## 1 14.0 350 165 US  
## 2 31.9 89 71 Europe  
## 3 17.0 302 140 US  
## 4 15.0 400 150 US  
## 5 30.5 98 63 US  
## 6 23.0 350 125 US

## We make certain entries missing  
cars.4var[2,2]<-cars.4var[4,4]<-NA  
head(cars.4var)

## mpg cubicinches hp brand  
## 1 14.0 350 165 US  
## 2 31.9 NA 71 Europe  
## 3 17.0 302 140 US  
## 4 15.0 400 150 <NA>  
## 5 30.5 98 63 US  
## 6 23.0 350 125 US

## Now we try to replace the missing values with constant  
cars.4var[4,4]<-"Missing"

## Warning in `[<-.factor`(`\*tmp\*`, iseq, value = "Missing"): invalid factor level,  
## NA generated

cars.4var[2,2]<-0  
head(cars.4var)

## mpg cubicinches hp brand  
## 1 14.0 350 165 US  
## 2 31.9 0 71 Europe  
## 3 17.0 302 140 US  
## 4 15.0 400 150 <NA>  
## 5 30.5 98 63 US  
## 6 23.0 350 125 US

## We can also replace the values with mean  
  
##cars.4var[2,2]<-mean(na.omit(cars.4var$Cubicinches))  
cars.4var[2,2]<-201   
  
## Now we will replace the missing data with the mode  
  
our\_table<-table(cars.4var$brand)  
our\_table

##   
## Europe Japan US   
## 48 51 163

## Calculating the mode from our\_table  
  
our\_mode<-names(our\_table)[our\_table==max(our\_table)]  
cars.4var[4,4]<-our\_mode  
head(cars.4var)

## mpg cubicinches hp brand  
## 1 14.0 350 165 US  
## 2 31.9 201 71 Europe  
## 3 17.0 302 140 US  
## 4 15.0 400 150 US  
## 5 30.5 98 63 US  
## 6 23.0 350 125 US

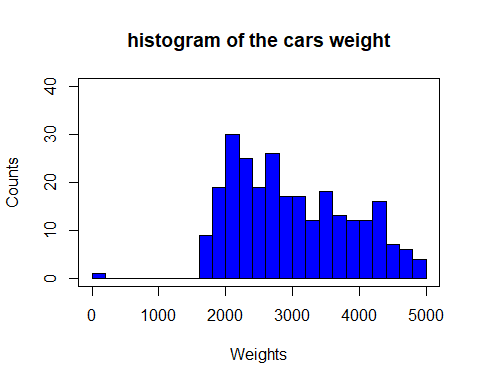
## Now we will try to generrate random variables.  
  
obs\_brand<-sample(na.omit(cars.4var$brand),1)  
#obs\_cubicinches<-sample(na.omit(cars.4var$Cubicinches),1)  
obs\_brand

## [1] US  
## Levels: Europe Japan US

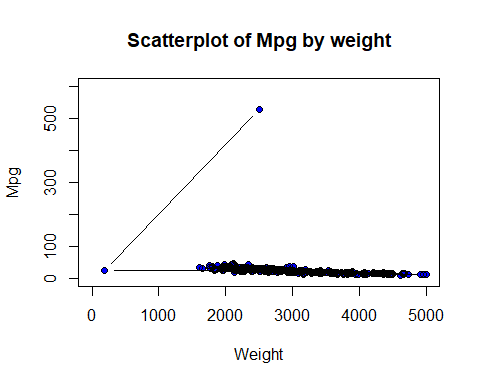
obs\_cubicinches<-86  
cars.4var[2,2]<-obs\_cubicinches  
cars.4var[4,4]<-obs\_brand  
head(cars.4var)

## mpg cubicinches hp brand  
## 1 14.0 350 165 US  
## 2 31.9 86 71 Europe  
## 3 17.0 302 140 US  
## 4 15.0 400 150 US  
## 5 30.5 98 63 US  
## 6 23.0 350 125 US

## Next we try to create a histogram  
  
## Setting up the plot area  
  
par(mfrow=c(1,1))  
  
## Create the histogram bars  
  
hist(cars2$weightlbs,breaks = 30,xlim = c(0,5000),col = "blue",border = "black",ylim = c(0,40),xlab = "Weights",ylab = "Counts",main = "histogram of the cars weight")  
  
## We can also make a box around the plot  
  
box(which = "plot",lty = "solid",col="black")



## Now we will try to create the scatter plot  
  
plot(cars2$weightlbs,cars2$mpg,xlim = c(0,5000),ylim = c(0,600),xlab = "Weight",ylab = "Mpg",main = "Scatterplot of Mpg by weight",type = "p",pch=16,col="blue",)  
  
## Adding open black circles  
  
points(cars2$weightlbs,cars2$mpg,type = "b",col="black")



## Next we will try descriptive statistics  
  
mean(cars2$weightlbs)

## [1] 2992.873

median(cars2$weightlbs)

## [1] 2830

length(cars2$weightlbs)

## [1] 263

sd(cars2$weightlbs)

## [1] 867.4516

summary(cars2$weightlbs)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 192.5 2245.5 2830.0 2992.9 3654.5 4997.0

## Nw we will try to transform the data  
  
## Min-Max Normalization  
summary(cars2$weightlbs)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 192.5 2245.5 2830.0 2992.9 3654.5 4997.0

mi<-min(cars2$weightlbs)  
ma<-max(cars2$weightlbs)  
  
minimax.weight<-(cars2$weightlbs-mi)/(ma-mi)  
minimax.weight

## [1] 0.8359871 0.3605994 0.6778021 0.7427412 0.3868249 0.7716724 0.8680404  
## [8] 0.8574253 0.6946613 0.3866167 0.4272037 0.4153398 0.8218337 0.7225518  
## [15] 0.6636487 0.8674160 0.3803726 0.6744718 0.4340722 0.4657092 0.3801644  
## [22] 0.5010927 0.5868457 0.8037257 0.3695494 0.3335415 0.3289624 0.9292330  
## [29] 0.7038193 0.4553023 0.4032678 0.6422104 0.4815277 0.5110834 0.3730877  
## [36] 0.6707254 0.3418670 0.5260693 0.7831200 0.6395046 0.7710480 0.9252784  
## [43] 0.7185971 0.6925799 0.6022479 0.5083776 0.7414924 0.4584244 0.3637215  
## [50] 0.5968363 0.8890623 0.6238943 0.7111042 0.4090956 0.8715787 0.8503486  
## [57] 0.5697783 0.3918202 0.3630971 0.7552295 0.5729004 0.4550942 0.5656156  
## [64] 0.3991050 0.8418150 0.6936206 0.3658029 0.3689250 0.3397856 0.7675096  
## [71] 0.6055781 0.5094183 0.4361536 0.4896451 0.4217921 0.5874701 0.6020398  
## [78] 0.4834010 0.4313664 0.4490582 0.8601311 0.9906338 0.6728067 0.5206577  
## [85] 0.4220002 0.2956603 0.5373088 0.6811323 0.4022271 0.5260693 0.3710063  
## [92] 0.4386513 0.8133000 0.6528255 0.4292850 0.7731294 0.8963472 0.6132792  
## [99] 0.7571027 0.7469039 1.0000000 0.9810594 0.4024352 0.4178374 0.4313664  
## [106] 0.5083776 0.4459361 0.3897388 0.5158705 0.6892497 0.4342804 0.5704027  
## [113] 0.8622125 0.3335415 0.4032678 0.7477365 0.5302321 0.5924654 0.3855760  
## [120] 0.8634613 0.4182537 0.3685087 0.9315225 0.5312728 0.3416589 0.4938079  
## [127] 0.9286086 0.4240816 0.4973462 0.3345822 0.4386513 0.5173275 0.9904256  
## [134] 0.7242169 0.8840670 0.8320325 0.3668436 0.6195234 0.5729004 0.5799771  
## [141] 0.5219065 0.4118014 0.5146217 0.3262566 0.7102716 0.5753981 0.6592778  
## [148] 0.5291914 0.7285878 0.6270163 0.4890207 0.4927672 0.8809449 0.6172338  
## [155] 0.4727859 0.6472057 0.4020189 0.4043085 0.5081694 0.6696847 0.5849724  
## [162] 0.8091373 0.5810178 0.5643667 0.4290769 0.6603185 0.6757207 0.4979707  
## [169] 0.5441773 0.4022271 0.4365699 0.5489645 0.3485274 0.4958893 0.4232490  
## [176] 0.4155479 0.8726194 0.7242169 0.7029868 0.3855760 0.5822666 0.4063898  
## [183] 0.8372359 0.6750963 0.4584244 0.5989177 0.6798834 0.7799979 0.8193360  
## [190] 0.9454678 0.5052555 0.5156624 0.8803205 0.8095535 0.8085129 0.4061817  
## [197] 0.4781975 0.3730877 0.3630971 0.3647622 0.4274118 0.5092101 0.6717661  
## [204] 0.7154751 0.4596732 0.7612655 0.3991050 0.5887189 0.8128838 0.5635342  
## [211] 0.5458424 0.6769695 0.3252159 0.3916120 0.6328442 0.5042148 0.3710063  
## [218] 0.8372359 0.5000520 0.8538870 0.6245187 0.5416797 0.8876054 0.7550213  
## [225] 0.8012280 0.3768342 0.9204912 0.3031533 0.4983869 0.5406390 0.7015298  
## [232] 0.5062962 0.4994276 0.8961390 0.6155687 0.4032678 0.8124675 0.5770632  
## [239] 0.3501925 0.4220002 0.8039338 0.6095327 0.8382766 0.5160787 0.8661671  
## [246] 0.6673952 0.7373296 0.3939016 0.4084712 0.4854824 0.4011864 0.5135810  
## [253] 0.5739411 0.7899886 0.5271100 0.4536372 0.7591841 0.3345822 0.5500052  
## [260] 0.6442918 0.6578208 0.0000000 0.4802789

## Now we will try decimal scaling  
  
max(abs(cars2$weightlbs)) ## Maximum absolute value comes to be 4997 (It is of 4 digits)

## [1] 4997

d.weight<-cars2$weightlbs/(10^4)  
d.weight

## [1] 0.42090 0.19250 0.34490 0.37610 0.20510 0.39000 0.43630 0.43120 0.35300  
## [10] 0.20500 0.22450 0.21880 0.41410 0.36640 0.33810 0.43600 0.20200 0.34330  
## [19] 0.22780 0.24300 0.20190 0.26000 0.30120 0.40540 0.19680 0.17950 0.17730  
## [28] 0.46570 0.35740 0.23800 0.21300 0.32780 0.25060 0.26480 0.19850 0.34150  
## [37] 0.18350 0.27200 0.39550 0.32650 0.38970 0.46380 0.36450 0.35200 0.30860  
## [46] 0.26350 0.37550 0.23950 0.19400 0.30600 0.44640 0.31900 0.36090 0.21580  
## [55] 0.43800 0.42780 0.29300 0.20750 0.19370 0.38210 0.29450 0.23790 0.29100  
## [64] 0.21100 0.42370 0.35250 0.19500 0.19650 0.18250 0.38800 0.31020 0.26400  
## [73] 0.22880 0.25450 0.22190 0.30150 0.30850 0.25150 0.22650 0.23500 0.43250  
## [82] 0.49520 0.34250 0.26940 0.22200 0.16130 0.27740 0.34650 0.21250 0.27200  
## [91] 0.19750 0.23000 0.41000 0.33290 0.22550 0.39070 0.44990 0.31390 0.38300  
## [100] 0.37810 0.49970 0.49060 0.21260 0.22000 0.22650 0.26350 0.23350 0.20650  
## [109] 0.26710 0.35040 0.22790 0.29330 0.43350 0.17950 0.21300 0.37850 0.27400  
## [118] 0.30390 0.20450 0.43410 0.22020 0.19630 0.46680 0.27450 0.18340 0.25650  
## [127] 0.46540 0.22300 0.25820 0.18000 0.23000 0.26780 0.49510 0.36720 0.44400  
## [136] 0.41900 0.19550 0.31690 0.29450 0.29790 0.27000 0.21710 0.26650 0.17600  
## [145] 0.36050 0.29570 0.33600 0.27350 0.36930 0.32050 0.25420 0.25600 0.44250  
## [154] 0.31580 0.24640 0.33020 0.21240 0.21350 0.26340 0.34100 0.30030 0.40800  
## [163] 0.29840 0.29040 0.22540 0.33650 0.34390 0.25850 0.28070 0.21250 0.22900  
## [172] 0.28300 0.18670 0.25750 0.22260 0.21890 0.43850 0.36720 0.35700 0.20450  
## [181] 0.29900 0.21450 0.42150 0.34360 0.23950 0.30700 0.34590 0.39400 0.41290  
## [190] 0.47350 0.26200 0.26700 0.44220 0.40820 0.40770 0.21440 0.24900 0.19850  
## [199] 0.19370 0.19450 0.22460 0.26390 0.34200 0.36300 0.24010 0.38500 0.21100  
## [208] 0.30210 0.40980 0.29000 0.28150 0.34450 0.17550 0.20740 0.32330 0.26150  
## [217] 0.19750 0.42150 0.25950 0.42950 0.31930 0.27950 0.44570 0.38200 0.40420  
## [226] 0.20030 0.46150 0.16490 0.25870 0.27900 0.35630 0.26250 0.25920 0.44980  
## [235] 0.31500 0.21300 0.40960 0.29650 0.18750 0.22200 0.40550 0.31210 0.42200  
## [244] 0.26720 0.43540 0.33990 0.37350 0.20850 0.21550 0.25250 0.21200 0.26600  
## [253] 0.29500 0.39880 0.27250 0.23720 0.38400 0.18000 0.28350 0.32880 0.33530  
## [262] 0.01925 0.25000

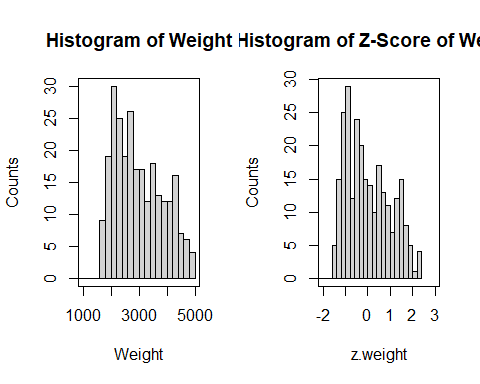
## Now we will try Z-Score Standardization  
  
z.weight<-(cars2$weightlbs-mean(cars2$weightlbs))/sd(cars2$weightlbs)  
z.weight

## [1] 1.401954157 -1.231045771 0.525824584 0.885498829 -1.085792710  
## [6] 1.045738317 1.579485675 1.520692770 0.619201551 -1.086945512  
## [11] -0.862149108 -0.927858826 1.323563616 0.773677029 0.447434043  
## [16] 1.576027269 -1.121529574 0.507379750 -0.824106640 -0.648880725  
## [21] -1.122682376 -0.452904374 0.022050079 1.223269836 -1.181475282  
## [26] -1.380910040 -1.406271685 1.918409484 0.669924842 -0.706520829  
## [31] -0.994721346 0.328695430 -0.561267768 -0.397569874 -1.161877646  
## [36] 0.486629313 -1.334797957 -0.314568125 1.109142431 0.313709003  
## [41] 1.042279911 1.896506244 0.751773789 0.607673531 0.107357432  
## [46] -0.412556301 0.878582017 -0.689228798 -1.213753740 0.077384578  
## [51] 1.695918684 0.227248848 0.710272915 -0.962442888 1.599083310  
## [56] 1.481497499 -0.072479691 -1.058125460 -1.217212146 0.954666954  
## [61] -0.055187660 -0.707673631 -0.095535732 -1.017777388 1.434232615  
## [66] 0.613437541 -1.202225719 -1.184933688 -1.346325978 1.022682276  
## [71] 0.125802265 -0.406792291 -0.812578619 -0.516308487 -0.892121962  
## [76] 0.025508485 0.106204630 -0.550892550 -0.839093067 -0.741104891  
## [81] 1.535679197 2.258486094 0.498157334 -0.344540979 -0.890969160  
## [86] -1.590720016 -0.252316813 0.544269417 -1.000485357 -0.314568125  
## [91] -1.173405667 -0.798744995 1.276298731 0.387488335 -0.850621088  
## [96] 1.053807932 1.736266757 0.168455942 0.965042172 0.908554871  
## [101] 2.310362188 2.205457199 -0.999332555 -0.914025201 -0.839093067  
## [106] -0.412556301 -0.758396922 -1.069653481 -0.371055427 0.589228697  
## [111] -0.822953838 -0.069021284 1.547207217 -1.380910040 -0.994721346  
## [116] 0.913166079 -0.291512084 0.053175735 -1.092709522 1.554124030  
## [121] -0.911719597 -1.187239292 1.931090307 -0.285748073 -1.335950759  
## [126] -0.493252446 1.914951078 -0.879441139 -0.473654811 -1.375146029  
## [131] -0.798744995 -0.362985812 2.257333292 0.782899445 1.668251435  
## [136] 1.380050917 -1.196461709 0.203040004 -0.055187660 -0.015992389  
## [141] -0.337624167 -0.947456461 -0.377972239 -1.421258112 0.705661706  
## [146] -0.041354035 0.423225199 -0.297276094 0.807108289 0.244540879  
## [151] -0.519766894 -0.499016456 1.650959404 0.190359181 -0.609685455  
## [156] 0.356362679 -1.001638159 -0.988957336 -0.413709103 0.480865303  
## [161] 0.011674861 1.253242690 -0.010228379 -0.102452544 -0.851773890  
## [166] 0.428989210 0.514296563 -0.470196405 -0.214274345 -1.000485357  
## [171] -0.810273015 -0.187759898 -1.297908291 -0.481724425 -0.884052348  
## [176] -0.926706024 1.604847321 0.782899445 0.665313634 -1.092709522  
## [181] -0.003311566 -0.977429315 1.408870969 0.510838157 -0.689228798  
## [186] 0.088912599 0.537352604 1.091850400 1.309729991 2.008328045  
## [191] -0.429848332 -0.372208229 1.647500997 1.255548294 1.249784283  
## [196] -0.978582117 -0.579712601 -1.161877646 -1.217212146 -1.207989729  
## [201] -0.860996306 -0.407945093 0.492393324 0.734481758 -0.682311985  
## [206] 0.988098214 -1.017777388 0.032425298 1.273993127 -0.107063753  
## [211] -0.205051929 0.521213375 -1.427022122 -1.059278262 0.276819337  
## [216] -0.435612343 -1.173405667 1.408870969 -0.458668384 1.501095135  
## [221] 0.230707254 -0.228107970 1.687849070 0.953514151 1.209436211  
## [226] -1.141127209 1.869991797 -1.549219142 -0.467890800 -0.233871980  
## [231] 0.657244020 -0.424084322 -0.462126790 1.735113955 0.181136765  
## [236] -0.994721346 1.271687523 -0.032131618 -1.288685874 -0.890969160  
## [241] 1.224422638 0.147705505 1.414634979 -0.369902625 1.569110457  
## [246] 0.468184480 0.855525976 -1.046597439 -0.965901295 -0.539364529  
## [251] -1.006249367 -0.383736249 -0.049423649 1.147184899 -0.308804115  
## [256] -0.715743245 0.976570193 -1.375146029 -0.181995887 0.340223450  
## [261] 0.415155585 -3.228275356 -0.568184581

length(cars2$weightlbs)

## [1] 263

## Now we will try to create two histograms side by side  
  
par(mfrow=c(1,2)) ## 1 row 2 columns.  
  
hist(cars2$weightlbs,breaks = 20,xlim = c(1000,5000),main = "Histogram of Weight",xlab = "Weight",ylab = "Counts")  
box(which = "plot",lty = "solid",col="black")  
  
hist(z.weight,breaks = 20,xlim = c(-2,3),main = "Histogram of Z-Score of Weight",ylab = "Counts")  
box(which = "plot",lty = "solid",col="black")



## Now we check for the skewness  
(3\*(mean(cars2$weightlbs)-median(cars2$weightlbs)))/sd(cars2$weightlbs)

## [1] 0.5632797

(3\*(mean(z.weight)-median(z.weight)))/sd(z.weight)

## [1] 0.5632797

## Now we will transform the data to normality using various methods.  
  
## Square root method  
  
sqrt.weight<-sqrt(cars2$weightlbs)  
sqrt.weight\_Skew<-(3\*(mean(sqrt.weight)-median(sqrt.weight)))/sd(sqrt.weight)  
sqrt.weight\_Skew

## [1] 0.3387833

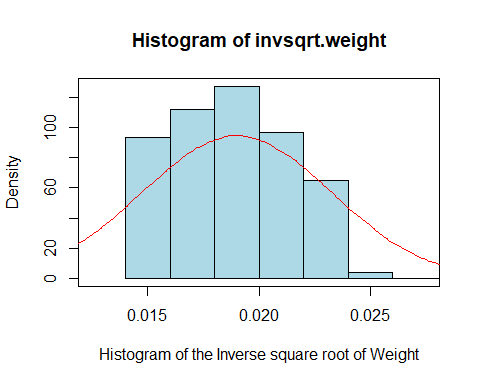
## Natral Log Method  
  
ln.weight<-log(cars2$weightlbs)  
ln.weight\_skew<-(3\*(mean(ln.weight)-median(ln.weight)))/sd(ln.weight)  
ln.weight\_skew

## [1] 0.08594381

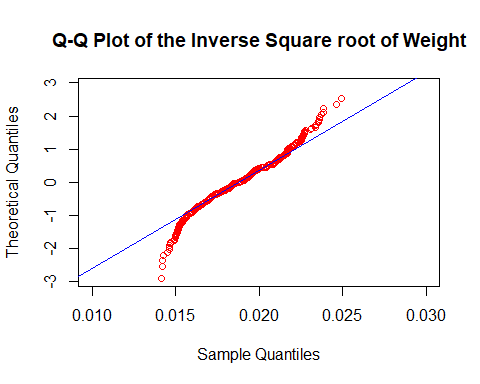
## Inverse Sqare root method  
  
invsqrt.weight<-(1/sqrt(cars2$weightlbs))  
invsqrt.weight\_skew<-(3\*(mean(invsqrt.weight)-median(invsqrt.weight)))/sd(invsqrt.weight)  
invsqrt.weight\_skew

## [1] 0.1453245

par(mfrow=c(1,1))  
x<-rnorm(1000000,mean = mean(invsqrt.weight),sd=sd(invsqrt.weight))  
hist(invsqrt.weight,breaks = 30,xlim = c(0.0125,0.0275),col = "lightblue",prob=TRUE,border = "black",xlab = "Histogram of the Inverse square root of Weight")  
box(which = "plot",lty = "solid",col="black")  
## Overlay with normal density  
lines(density(x),col="red")



## Normal Q-Q Plot  
qqnorm(invsqrt.weight,datax = TRUE,col="red",ylim = c(0.01,0.03),main = "Q-Q Plot of the Inverse Square root of Weight")  
qqline(invsqrt.weight,col="blue",datax = TRUE)



## De-Transform Data  
## Transform x using y=1/sqrt(x)  
x<-cars2$weightlbs[1]  
x

## [1] 4209

y<-1/sqrt(x)  
y

## [1] 0.01541383

## Now de-transforming the x using x=1/(y)^2  
detransformedx<-1/(y)^2  
detransformedx

## [1] 4209

## Now we will try Creating indicator variables  
  
north\_flag<-east\_flag<-south\_flag<-c(rep(NA,10)) # Repeating NA 10 times.  
north\_flag

## [1] NA NA NA NA NA NA NA NA NA NA

east\_flag

## [1] NA NA NA NA NA NA NA NA NA NA

south\_flag

## [1] NA NA NA NA NA NA NA NA NA NA

region<-c(rep(c("North","South","East","West"),2),"North","South")  
region

## [1] "North" "South" "East" "West" "North" "South" "East" "West" "North"  
## [10] "South"

## Now we will try change region variables to indictors  
  
for (i in 1:length(region)) {  
 if(region[i]=="North")north\_flag[i]=1  
 else north\_flag[i]=0  
 if(region[i]=="East")east\_flag[i]=1  
 else(east\_flag[i]=0)  
 if(region[i]=="South")south\_flag[i]=1  
 else(south\_flag[i]=0)  
   
}  
  
north\_flag

## [1] 1 0 0 0 1 0 0 0 1 0

east\_flag

## [1] 0 0 1 0 0 0 1 0 0 0

south\_flag

## [1] 0 1 0 0 0 1 0 0 0 1

## Now we will see Index fields  
  
## Data frames have an index field; the leftmost columns of the cars2 data set  
  
cars2[order(cars2$mpg),] ## cars2 data frame arranged in order of increasing mpg.

## mpg cylinders cubicinches hp weightlbs time.to.60 year brand  
## 227 10.0 8 360 215 4615.0 14 1971 US  
## 14 11.0 8 350 180 3664.0 11 1974 US  
## 101 11.0 8 400 150 4997.0 14 1974 US  
## 82 12.0 8 429 198 4952.0 12 1974 US  
## 97 12.0 8 350 180 4499.0 13 1974 US  
## 102 12.0 8 400 167 4906.0 13 1974 US  
## 133 12.0 8 455 225 4951.0 11 1974 US  
## 7 13.0 8 351 158 4363.0 13 1974 US  
## 47 13.0 8 318 150 3755.0 14 1977 US  
## 51 13.0 8 400 150 4464.0 12 1974 US  
## 60 13.0 8 360 175 3821.0 11 1974 US  
## 93 13.0 8 350 175 4100.0 13 1974 US  
## 127 13.0 8 360 170 4654.0 13 1974 US  
## 138 13.0 8 302 129 3169.0 12 1976 US  
## 188 13.0 8 318 150 3940.0 13 1977 US  
## 190 13.0 8 440 215 4735.0 11 1974 US  
## 193 13.0 8 400 190 4422.0 13 1973 US  
## 209 13.0 8 307 130 4098.0 14 1973 US  
## 241 13.0 8 350 145 4055.0 12 1977 US  
## 254 13.0 8 350 145 3988.0 13 1974 US  
## 1 14.0 8 350 165 4209.0 12 1972 US  
## 8 14.0 8 440 215 4312.0 9 1971 US  
## 28 14.0 8 351 148 4657.0 14 1976 US  
## 42 14.0 8 302 140 4638.0 16 1975 US  
## 45 14.0 8 455 225 3086.0 10 1971 US  
## 53 14.0 8 340 160 3609.0 8 1971 US  
## 65 14.0 8 318 150 4237.0 15 1974 US  
## 153 14.0 8 455 225 4425.0 10 1971 US  
## 177 14.0 8 400 175 4385.0 12 1973 US  
## 178 14.0 8 304 150 3672.0 12 1974 US  
## 189 14.0 8 351 153 4129.0 13 1973 US  
## 195 14.0 8 318 150 4077.0 14 1973 US  
## 223 14.0 8 318 150 4457.0 14 1975 US  
## 225 14.0 8 302 137 4042.0 15 1974 US  
## 237 14.0 8 318 150 4096.0 13 1972 US  
## 245 14.0 8 454 220 4354.0 9 1971 US  
## 218 14.5 8 351 152 4215.0 13 1977 US  
## 4 15.0 8 400 150 3761.0 10 1971 US  
## 120 15.0 8 429 198 4341.0 10 1971 US  
## 135 15.0 8 350 145 4440.0 14 1976 US  
## 149 15.0 8 350 165 3693.0 12 1971 US  
## 154 15.0 6 250 72 3158.0 20 1976 US  
## 194 15.0 8 350 145 4082.0 13 1974 US  
## 206 15.0 8 390 190 3850.0 9 1971 US  
## 220 15.0 8 302 130 4295.0 15 1978 US  
## 231 15.0 8 383 170 3563.0 10 1971 US  
## 246 15.0 8 318 150 3399.0 11 1974 US  
## 24 15.5 8 351 142 4054.0 14 1980 US  
## 81 15.5 8 400 190 4325.0 12 1978 US  
## 13 16.0 8 302 140 4141.0 14 1975 US  
## 18 16.0 8 304 150 3433.0 12 1971 US  
## 32 16.0 6 250 100 3278.0 18 1974 US  
## 41 16.0 6 250 105 3897.0 19 1976 US  
## 56 16.0 8 400 230 4278.0 10 1974 US  
## 100 16.0 6 250 100 3781.0 17 1975 US  
## 113 16.0 8 351 149 4335.0 15 1978 US  
## 123 16.0 8 400 170 4668.0 12 1976 US  
## 136 16.0 8 318 150 4190.0 13 1977 US  
## 167 16.0 6 225 105 3439.0 16 1972 US  
## 234 16.0 8 318 150 4498.0 15 1976 US  
## 243 16.0 8 400 180 4220.0 11 1978 US  
## 160 16.2 6 163 133 3410.0 16 1979 Europe  
## 39 16.5 8 351 138 3955.0 13 1980 US  
## 55 16.5 8 350 180 4380.0 12 1977 US  
## 224 16.5 6 168 120 3820.0 17 1977 Europe  
## 16 16.9 8 350 155 4360.0 15 1980 US  
## 3 17.0 8 302 140 3449.0 11 1971 US  
## 94 17.0 6 250 100 3329.0 16 1972 US  
## 96 17.0 6 231 110 3907.0 21 1976 US  
## 134 17.0 8 304 150 3672.0 12 1973 US  
## 257 17.0 8 305 130 3840.0 15 1980 US  
## 44 17.5 6 250 110 3520.0 16 1978 US  
## 70 17.5 8 305 145 3880.0 13 1978 US  
## 162 17.5 8 318 140 4080.0 14 1979 US  
## 183 17.5 8 305 140 4215.0 13 1977 US  
## 221 17.5 6 258 95 3193.0 18 1977 US  
## 88 17.6 6 225 85 3465.0 17 1982 US  
## 212 17.7 6 231 165 3445.0 13 1979 US  
## 29 18.0 6 250 78 3574.0 21 1977 US  
## 61 18.0 6 232 100 2945.0 16 1974 US  
## 87 18.0 6 199 97 2774.0 16 1971 US  
## 98 18.0 6 250 88 3139.0 15 1972 US  
## 110 18.0 8 307 130 3504.0 12 1971 US  
## 112 18.0 4 121 112 2933.0 15 1973 Europe  
## 116 18.0 6 225 95 3785.0 19 1976 US  
## 157 18.0 3 70 90 2124.0 14 1974 Japan  
## 163 18.0 6 171 97 2984.0 15 1976 US  
## 184 18.0 8 318 150 3436.0 11 1971 US  
## 187 18.0 6 250 105 3459.0 16 1976 US  
## 208 18.0 6 250 88 3021.0 17 1974 US  
## 242 18.0 6 225 105 3121.0 17 1974 US  
## 260 18.0 6 232 100 3288.0 16 1972 US  
## 150 18.1 8 302 139 3205.0 11 1979 US  
## 99 18.2 8 318 135 3830.0 15 1980 US  
## 43 18.5 6 250 110 3645.0 16 1977 US  
## 66 18.5 6 250 98 3525.0 19 1978 US  
## 57 19.0 6 156 108 2930.0 16 1977 Japan  
## 156 19.0 6 250 88 3302.0 16 1972 US  
## 159 19.0 6 232 100 2634.0 13 1972 US  
## 204 19.0 6 225 100 3630.0 18 1978 US  
## 15 19.1 6 225 90 3381.0 19 1981 US  
## 83 19.2 8 305 145 3425.0 13 1979 US  
## 145 19.2 8 267 125 3605.0 15 1980 US  
## 247 19.4 8 318 140 3735.0 13 1979 US  
## 181 19.8 6 200 85 2990.0 18 1980 US  
## 166 19.9 8 260 110 3365.0 16 1979 US  
## 71 20.0 6 198 95 3102.0 17 1975 US  
## 111 20.0 4 97 88 2279.0 19 1974 Japan  
## 129 20.0 4 114 91 2582.0 14 1974 Europe  
## 169 20.0 6 156 122 2807.0 14 1974 Japan  
## 235 20.0 4 130 102 3150.0 16 1977 Europe  
## 40 20.2 6 232 90 3265.0 18 1980 US  
## 50 20.2 6 200 88 3060.0 17 1982 US  
## 179 20.2 8 302 139 3570.0 13 1979 US  
## 238 20.2 6 200 85 2965.0 16 1979 US  
## 172 20.3 5 131 103 2830.0 16 1979 Europe  
## 147 20.6 6 225 110 3360.0 17 1980 US  
## 186 20.8 6 200 85 3070.0 17 1979 US  
## 34 21.0 6 199 90 2648.0 15 1971 US  
## 118 21.0 6 231 110 3039.0 15 1976 US  
## 140 21.0 4 120 87 2979.0 20 1973 Europe  
## 175 21.0 4 122 86 2226.0 17 1973 US  
## 205 21.0 4 140 72 2401.0 20 1974 US  
## 229 21.0 6 200 85 2587.0 16 1971 US  
## 78 21.1 4 134 95 2515.0 15 1979 Japan  
## 22 21.5 4 121 110 2600.0 13 1978 Europe  
## 38 21.5 3 80 110 2720.0 14 1978 Japan  
## 222 21.6 4 121 115 2795.0 16 1979 Europe  
## 48 22.0 4 122 86 2395.0 16 1973 US  
## 62 22.0 4 108 94 2379.0 17 1974 Japan  
## 139 22.0 4 121 98 2945.0 15 1976 Europe  
## 211 22.0 6 146 97 2815.0 15 1978 Japan  
## 215 22.0 6 225 100 3233.0 15 1977 US  
## 259 22.0 6 232 112 2835.0 15 1983 US  
## 261 22.0 6 250 105 3353.0 15 1977 US  
## 36 22.4 6 231 110 3415.0 16 1982 US  
## 77 22.5 6 232 90 3085.0 18 1977 US  
## 6 23.0 8 350 125 3900.0 17 1980 US  
## 33 23.0 4 120 97 2506.0 15 1973 Japan  
## 84 23.0 4 115 95 2694.0 15 1976 Europe  
## 146 23.0 4 120 88 2957.0 17 1976 Europe  
## 164 23.0 6 198 95 2904.0 16 1974 US  
## 165 23.0 4 97 54 2254.0 24 1973 Europe  
## 202 23.0 4 140 83 2639.0 17 1976 US  
## 233 23.0 4 140 78 2592.0 19 1976 US  
## 240 23.0 4 122 86 2220.0 14 1972 US  
## 124 23.2 4 156 105 2745.0 17 1979 US  
## 255 23.5 6 173 110 2725.0 13 1982 US  
## 203 23.9 8 260 90 3420.0 22 1980 US  
## 19 24.0 4 113 95 2278.0 16 1973 Japan  
## 20 24.0 4 107 90 2430.0 15 1971 Europe  
## 23 24.0 6 200 81 3012.0 18 1977 US  
## 54 24.0 4 116 75 2158.0 16 1974 Europe  
## 74 24.0 4 119 97 2545.0 17 1976 Japan  
## 252 24.0 4 121 110 2660.0 14 1974 Europe  
## 256 24.0 4 113 95 2372.0 15 1971 Japan  
## 161 24.3 4 151 90 3003.0 20 1981 US  
## 117 24.5 4 151 88 2740.0 16 1978 US  
## 85 25.0 4 116 81 2220.0 17 1977 Europe  
## 103 25.0 4 98 80 2126.0 17 1973 US  
## 109 25.0 4 121 115 2671.0 14 1976 Europe  
## 244 25.0 4 110 87 2672.0 18 1971 Europe  
## 262 25.0 6 250 105 192.5 15 1977 US  
## 9 25.4 5 183 77 3530.0 20 1980 Europe  
## 210 25.4 6 168 116 2900.0 13 1982 Japan  
## 191 25.8 4 156 92 2620.0 14 1982 US  
## 37 26.0 4 97 46 1835.0 21 1971 Europe  
## 67 26.0 4 97 46 1950.0 21 1974 Europe  
## 79 26.0 4 98 90 2265.0 16 1974 Europe  
## 95 26.0 4 98 79 2255.0 18 1977 US  
## 122 26.0 4 79 67 1963.0 16 1975 Europe  
## 131 26.0 4 97 78 2300.0 15 1975 Europe  
## 137 26.0 4 91 70 1955.0 21 1972 US  
## 168 26.0 4 156 92 2585.0 15 1983 US  
## 176 26.0 4 96 69 2189.0 18 1973 Europe  
## 201 26.0 4 116 75 2246.0 14 1975 Europe  
## 126 26.5 4 140 72 2565.0 14 1977 US  
## 106 26.6 4 151 84 2635.0 16 1982 US  
## 141 26.8 6 173 115 2700.0 13 1980 US  
## 31 27.0 4 97 88 2130.0 15 1971 Japan  
## 72 27.0 4 112 88 2640.0 19 1983 US  
## 121 27.0 4 101 83 2202.0 15 1977 Europe  
## 125 27.0 4 97 60 1834.0 19 1972 Europe  
## 148 27.0 4 151 90 2735.0 18 1983 US  
## 230 27.0 4 140 86 2790.0 16 1983 US  
## 52 27.2 4 141 71 3190.0 25 1980 Europe  
## 92 27.2 4 119 97 2300.0 15 1979 Japan  
## 197 27.2 4 135 84 2490.0 16 1982 US  
## 152 27.5 4 134 95 2560.0 14 1979 Japan  
## 73 28.0 4 97 92 2288.0 17 1973 Japan  
## 89 28.0 4 90 75 2125.0 15 1975 US  
## 132 28.0 4 151 90 2678.0 17 1981 US  
## 155 28.0 4 107 86 2464.0 16 1977 Europe  
## 232 28.0 4 120 79 2625.0 19 1983 US  
## 249 28.0 4 97 75 2155.0 16 1977 Japan  
## 192 28.4 4 151 90 2670.0 16 1980 US  
## 219 28.8 6 173 115 2595.0 11 1980 US  
## 49 29.0 4 97 78 1940.0 15 1978 Europe  
## 59 29.0 4 90 70 1937.0 14 1977 Europe  
## 75 29.0 4 98 83 2219.0 17 1975 Europe  
## 142 29.0 4 97 75 2171.0 16 1976 Japan  
## 173 29.0 4 68 49 1867.0 20 1974 Europe  
## 199 29.0 4 90 70 1937.0 14 1976 Europe  
## 250 29.0 4 135 84 2525.0 16 1983 US  
## 69 29.5 4 97 71 1825.0 12 1977 Europe  
## 158 29.5 4 98 68 2135.0 17 1979 Japan  
## 30 29.9 4 98 65 2380.0 21 1982 US  
## 35 30.0 4 97 67 1985.0 16 1978 Japan  
## 108 30.0 4 88 76 2065.0 15 1972 Europe  
## 214 30.0 4 79 70 2074.0 20 1972 Europe  
## 5 30.5 4 98 63 2051.0 17 1978 US  
## 128 30.9 4 105 75 2230.0 15 1979 US  
## 27 31.0 4 71 65 1773.0 19 1972 Japan  
## 90 31.0 4 119 82 2720.0 19 1983 US  
## 174 31.0 4 112 85 2575.0 16 1983 US  
## 228 31.0 4 76 52 1649.0 17 1975 Japan  
## 151 31.3 4 120 75 2542.0 18 1981 Japan  
## 180 31.5 4 98 68 2045.0 19 1978 Japan  
## 46 31.6 4 120 74 2635.0 18 1982 Japan  
## 17 31.8 4 85 65 2020.0 19 1980 Japan  
## 2 31.9 4 89 71 1925.0 14 1980 Europe  
## 68 32.0 4 91 67 1965.0 16 1983 Japan  
## 143 32.0 4 144 96 2665.0 14 1983 Japan  
## 226 32.0 4 83 61 2003.0 19 1975 Japan  
## 251 32.1 4 98 70 2120.0 16 1981 US  
## 105 32.2 4 108 75 2265.0 15 1981 Japan  
## 80 32.4 4 108 75 2350.0 17 1982 Japan  
## 171 32.4 4 107 72 2290.0 17 1981 Japan  
## 63 32.7 6 168 132 2910.0 11 1981 Japan  
## 216 32.9 4 119 100 2615.0 15 1982 Japan  
## 26 33.0 4 91 53 1795.0 17 1977 Japan  
## 114 33.0 4 91 53 1795.0 18 1976 Japan  
## 58 33.5 4 98 83 2075.0 16 1978 US  
## 200 33.5 4 85 70 1945.0 17 1978 Japan  
## 182 33.8 4 97 67 2145.0 18 1981 Japan  
## 11 34.0 4 108 70 2245.0 17 1983 Japan  
## 185 34.0 4 112 88 2395.0 18 1983 US  
## 91 34.1 4 86 65 1975.0 15 1980 Japan  
## 104 34.2 4 105 70 2200.0 13 1980 US  
## 12 34.3 4 97 78 2188.0 16 1981 Europe  
## 119 34.4 4 98 65 2045.0 16 1982 US  
## 86 35.0 4 72 69 1613.0 18 1972 Japan  
## 144 35.1 4 81 60 1760.0 16 1982 Japan  
## 130 36.1 4 98 66 1800.0 14 1979 US  
## 258 36.1 4 91 60 1800.0 16 1979 Japan  
## 253 36.4 5 121 67 2950.0 20 1981 Europe  
## 217 37.0 4 85 65 1975.0 19 1982 Japan  
## 21 37.2 4 86 65 2019.0 16 1981 Japan  
## 115 37.3 4 91 69 2130.0 15 1980 Europe  
## 10 37.7 4 89 62 2050.0 17 1982 Japan  
## 76 38.0 6 262 85 3015.0 17 1983 US  
## 170 38.0 4 105 63 2125.0 15 1983 US  
## 25 38.1 4 89 60 1968.0 19 1981 Japan  
## 239 39.0 4 86 64 1875.0 16 1982 US  
## 213 39.1 4 79 58 1755.0 17 1982 Japan  
## 207 40.8 4 85 65 2110.0 19 1981 Japan  
## 196 41.5 4 98 76 2144.0 15 1981 Europe  
## 198 43.1 4 90 48 1985.0 22 1979 Europe  
## 107 43.4 4 90 48 2335.0 24 1981 Europe  
## 236 44.0 4 97 52 2130.0 25 1983 Europe  
## 248 44.3 4 90 48 2085.0 22 1981 Europe  
## 64 46.6 4 86 65 2110.0 18 1981 Japan  
## 263 527.0 6 250 105 2500.0 15 1977 US

## For vector matrices, we add a column to act as an index field  
  
x<-c(1,1,3:4,3)  
x

## [1] 1 1 3 4 3

y<-c(9,9:1)  
y

## [1] 9 9 8 7 6 5 4 3 2 1

z<-c(2,1:9)  
z

## [1] 2 1 2 3 4 5 6 7 8 9

marix<-t(rbind(x,y,z))  
matrix

## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>

indexed\_m<-cbind(c(1:length(x)),matrix)  
indexed\_m

## matrix  
## [1,] 1 ?   
## [2,] 2 ?   
## [3,] 3 ?   
## [4,] 4 ?   
## [5,] 5 ?

indexed\_m[order(z)]

## [[1]]  
## [1] 2  
##   
## [[2]]  
## [1] 1  
##   
## [[3]]  
## [1] 3  
##   
## [[4]]  
## [1] 4  
##   
## [[5]]  
## [1] 5  
##   
## [[6]]  
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>  
##   
## [[7]]  
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>  
##   
## [[8]]  
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>  
##   
## [[9]]  
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>  
##   
## [[10]]  
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)   
## {  
## if (is.object(data) || !is.atomic(data))   
## data <- as.vector(data)  
## .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),   
## missing(ncol)))  
## }  
## <bytecode: 0x0000000012e8e870>  
## <environment: namespace:base>

## Fr any dublicated records we use "anyDublicated"  
anyDuplicated(cars2)

## [1] 0

## To examine each record, use "Duplicated"  
duplicated(cars2) ## It will show TRUE(if duplicate founds) or False(if not found).

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

## Let's duplicate the first record.  
new.cars<-rbind(cars2,cars2[1,])  
## Now checking for duplicates  
anyDuplicated(new.cars) # The 264th record id duplicate

## [1] 264

duplicated(new.cars)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE