LOADING LIBRARIES

```
library(plyr)
library(dplyr)
library(naniar)
library(lattice)
```

LOADING AND CLEANING DATA

```
load("census.RData")
dim(census)
```

```
## [1] 74020 31
```

COMMENTS:There are 74020 rows and 31 columns in the census data

1. How many states are represented among the 74020 census tracts?

```
n_distinct(census$State_name)
```

```
## [1] 52
```

```
n_distinct(census$County_name)
```

How many counties?

```
## [1] 1955
```

COMMENTS: There 52 distinct states and 1955 distinct counties.

2. Checking for classes

###Class of column 8

```
class(census$Med_HHD_Inc_ACS_09_13)
```

```
## [1] "factor"
```

class of column 9

```
class(census$Med_House_value_ACS_09_13)
```

```
## [1] "factor"
```

checking for the number of missing values in column 8

```
length(census$Med_HHD_Inc_ACS_09_13 [census$Med_HHD_Inc_ACS_09_13==""])
```

```
## [1] 1020
```

checking for the number of missing values in column 9

```
length(census$Med_House_value_ACS_09_13[census$Med_House_value_ACS_09_13==""])
```

```
## [1] 1804
```

COMMENTS: There are 1020 missing values in the median household income variable and 1804 in median house value variable.

3 Converting the 2 columns in From Factors to Numbers

```
dee = as.numeric(gsub("[\\$,]", "", census$ Med_HHD_Inc_ACS_09_13))
han = as.numeric(gsub("[\\$,]", "", census$ Med_House_value_ACS_09_13))
summary(dee)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     2499   37216   50341   56297   68777  250001    1020
```

```
summary(han)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    10300  104500  162500  218393  272900 1000001    1804
```

checking for missing values after variable conversion

```
n_miss(dee)
```

```
## [1] 1020
```

```
n_miss(han)
```

```
## [1] 1804
```

COMMENTS : The missing vlaues are the same as the previous question.

4 counting the number of missing values in each row

```
num.na.row = apply(census, 1, function(x) sum(is.na(x)))
```

obtaining the indexes of rows containing missing values

```
contains.na=num.na.row[num.na.row>0]
## finding the mean value of the number of missing values in each row
mean(contains.na)
```

```
## [1] 6.872415
```

COMMENTS: the mean number of missing values is 8.107081

5 states with no missing values

```
miss.row=unique(census[rowSums(is.na(census))>0,2])
no.miss.row=unique(census[,2])
setdiff(no.miss.row,miss.row)
```

```
## [1] "West Virginia"
```

COMMENTS: West Virginia has no missing values

6 remove all rows with missing values

```
new_census = census[complete.cases(census), ]
nrow(new_census)
```

```
## [1] 70877
```

```
attach(new_census)
```

COMMENTS : There are 70877 rows in the new census data

How many states are in the newdataframe?

```
n_distinct(State_name)
```

```
## [1] 51
```

COMMENTS: there are 51 states in the new dataframe

How many counties?

```
n_distinct(County_name)
```

```
## [1] 1861
```

COMMENTS: there are 1861 counties in the new dataframe

State that has been taken off

```
setdiff(unique(census$State_name),unique(State_name))
```

```
## [1] "Puerto Rico"
```

COMMENTS : Puerto Rico has been taken off

PART TWO : EXPLORATORY STATISTICS

1. the percentages of the population that are less than 5 years old

```
pct_Pop_0_4_ACS_09_13 =100-rowSums(new_census[,12:16])
Tot_Pop_0_4 =Tot_Population_ACS_09_13*(pct_Pop_0_4_ACS_09_13/100)
pop.state_0_4= tapply(Tot_Pop_0_4,State_name, sum)
```

determining Which state has the highest number of 0-4 years

```
sort(pop.state_0_4)[length(pop.state_0_4)]
```

```
## California
##      2492629
```

COMMENTS: California has the highest with a value of 2492629

2. Percentage of 0-4 year old

```
kat <- tapply(Tot_Population_ACS_09_13, State_name, sum)
faz<- tapply(Tot_Pop_0_4, State_name, sum)
pct.all.states <- (faz/kat)*100
sort(pct.all.states)[length(pct.all.states)]
```

```
##        Utah
## 9.252903
```

COMMENTS: Utah has the highest percentage of 9.2529%

3 Calculating the correlation between each of the numeric variables

```
corr=cor(new_census[,8:31])
```

Highest correlation variables

```
## finding the variables with the highest correlation value
max.cor<- which(cor==max(cor[which(cor!=1 , arr.ind = T)]), arr.ind = T)
names(new_census[0,8:31])[max.cor]
```

```
## [ 1] "pct_College_ACS_09_13" "Med_HHD_Inc_ACS_09_13"
## [3] "Med_HHD_Inc_ACS_09_13" "pct_College_ACS_09_13"
```

COMMENT: pct_College_ACS_09_13 and Med_HHD_Inc_ACS_09_13 has the highest correlation

Least correlation variables

```
min.cor<- which(cor==min(cor), arr.ind = T)
names(new_census[,8:31])[min.cor]
```

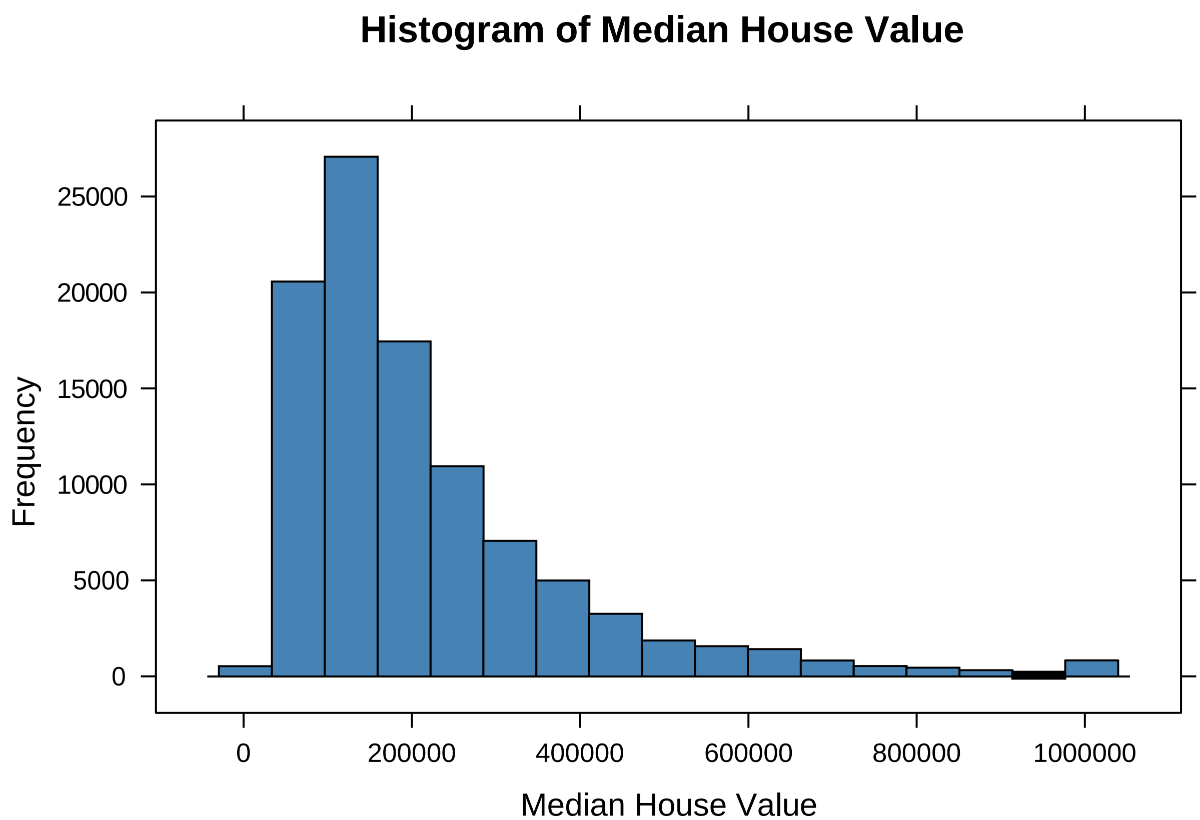
```
## [1] "pct_Single_Unit_ACS_09_13"     "pct_Renter_Occp_HU_ACS_09_13"
## [3] "pct_Renter_Occp_HU_ACS_09_13" "pct_Single_Unit_ACS_09_13"
```

COMMENT : pct_Single_Unit_ACS_09_13 andpct_Renter_Occp_HU_ACS_09_13 has the lowest correlation.

pct_College_ACS_09_13 and Med_HHD_Inc_ACS_09_13 has the highest correlation

4. Plot a histogram of Med_House_value_ACS_09_13, and label the axes appropriately.

```
hist(Med_House_value_ACS_09_13, main='Histogram of
        Median House Value', xlab='Median house
        value',
        col='steelblue')
```



5 Applying my.test() Function to variables in columns 10 through 31 of the census data frame

```
my.test = function(var) {
group = census$Med_House_value_ACS_09_13 == 1000001
p.val = t.test(var[group], var[!group])$p.value return(p.val)
 }
```

```
sort(apply(new_census[,10:31], 2, my.test)[1:2])
```

finding the 2 smallest p values

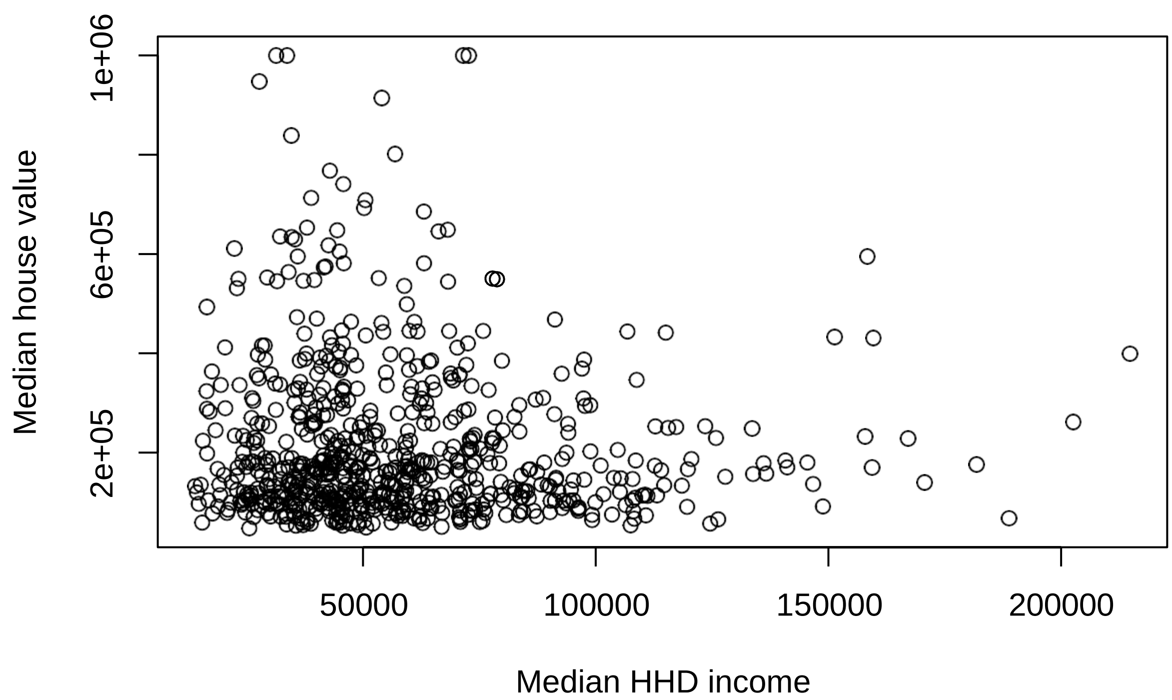
```
##     Mail_Return_Rate_CEN_2010        pct_Males_ACS_09_13
##                  6.259086e-13               8.373656e-01
```

COMMENT : Mail_Return_Rate_CEN_2010 and pct_Males_ACS_09_13 are the variables with the smallest values. The P-values are 6.259086e-13 for Mail_Return_Rate_CEN_2010 and 8.373656e-01 for pct_Males_ACS_09_13.

SAMPLING AND PLOTTING

1 Writing a Function plot.sample()

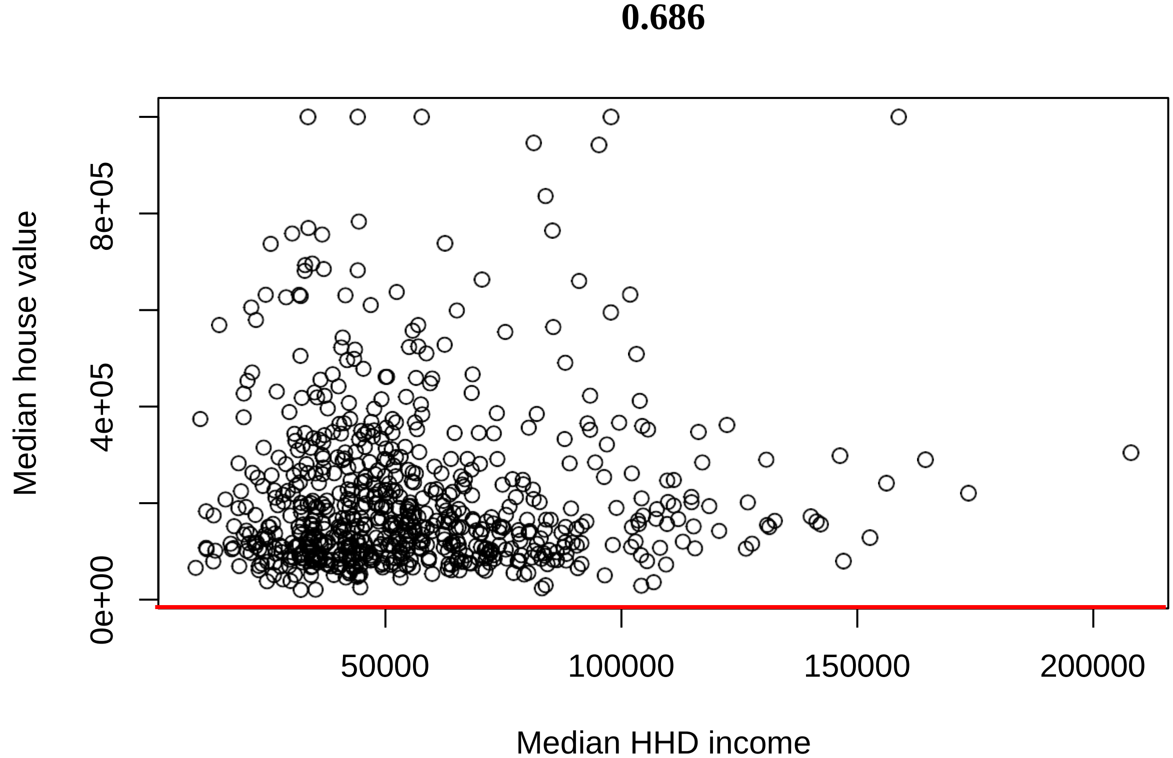
```
plot.sample=function(x,y,nsample,xlab,ylab){
   x=sample(x,500)
   y=sample(y,500)
   if(length(x)==length(y)){
      plot(x,y,xlab = xlab,ylab = ylab)
   }else{stop("the lenght of x and y are unequal")}
}

plot.sample(Med_HHD_Inc_ACS_09_13, Med_House_value_ACS_09_13, xlab="Median HHD
                  income", ylab="Median house value")
```



2. adding a trend line

```
add.lineartrend.info=function(x,y){
   abline(lm(y~x),lwd=2,col="red")
   title (signif(cor(x,y),3))
}
```

```
plot.sample(Med_HHD_Inc_ACS_09_13, Med_House_value_ACS_09_13, xlab="Median HHD
                  income", ylab="Median house value")
add.lineartrend.info(Med_HHD_Inc_ACS_09_13,Med_House_value_ACS_09_13)
```



3. plot all function

```
plot.all = function(dataset, nsample=500){
  p = ncol(dataset)
  orig.par = par()
  # Set the margins, and plotting grid
  par(mar=c(2,2,2,2))
  par(mfrow=c(p,p))
  # TODO: your plotting code goes here
  for (x in 1:p){
    for (y in 1:p){
      if(x==y){ plot(c(0,10),c(0,10),type = "n")
        text(5,5,labels =
               paste(names(dataset)[x]))
      }else{
        plot.sample(dataset[,x],dataset[,y],
                    xlab = names(dataset)[x],ylab = names(dataset)[y])
        add.lineartrend.info(dataset[,x],dataset[,y])
      }
```

```
      }
    }
    par(mar=orig.par$mar)
    par(mfrow=orig.par$mfrow)
}


mydat = new_census[,c("Med_HHD_Inc_ACS_09_13",
"Med_House_value_ACS_09_13","pct_College_ACS_09_13", "Mail_Return_Rate_CEN_2010")]
plot.all(mydat)
```