```
## 462    20.2 14.65 17.7         0
## 463    20.2 13.99 19.5         0
## 464    20.2 10.29 20.2         0
## 465    20.2 13.22 21.4         0
## 466    20.2 14.13 19.9         0
## 467    20.2 17.15 19.0         0
## 468    20.2 21.32 19.1         0
## 469    20.2 18.13 19.1         0
## 470    20.2 14.76 20.1         0
## 471    20.2 16.29 19.9         0
## 472    20.2 12.87 19.6         0
## 473    20.2 14.36 23.2         0
## 474    20.2 11.66 29.8         0
## 475    20.2 18.14 13.8         0
## 476    20.2 24.10 13.3         0
## 477    20.2 18.68 16.7         0
## 478    20.2 24.91 12.0         0
## 479    20.2 18.03 14.6         0
## 480    20.2 13.11 21.4         0
## 481    20.2 10.74 23.0         0
## 482    20.2  7.74 23.7         0
## 483    20.2  7.01 25.0         0
## 484    20.2 10.42 21.8         0
## 485    20.2 13.34 20.6         0
## 486    20.2 10.58 21.2         0
## 487    20.2 14.98 19.1         0
## 488    20.2 11.45 20.6         0
## 489    20.1 18.06 15.2         0
## 490    20.1 23.97  7.0         0
## 491    20.1 29.68  8.1         0
## 492    20.1 18.07 13.6         0
## 493    20.1 13.35 20.1         0
## 494    19.2 12.01 21.8         0
## 495    19.2 13.59 24.5         0
## 496    19.2 17.60 23.1         0
## 497    19.2 21.14 19.7         0
## 498    19.2 14.10 18.3         0
## 499    19.2 12.92 21.2         0
## 500    19.2 15.10 17.5         0
## 501    19.2 14.33 16.8         0
## 502    21.0  9.67 22.4         0
## 503    21.0  9.08 20.6         0
## 504    21.0  5.64 23.9         0
## 505    21.0  6.48 22.0         0
## 506    21.0  7.88 11.9         0
```

The following chunk will help you in creating Training/Validation/Testing Partitons in your Data.

```r
set.seed(1)
train.rows <- sample(rownames(housing), dim(housing)[1]*0.6)
# The code above means -- sample(data,selecting column using dim() and then multiplying it by 0.6 which
train.data <- housing[train.rows, ] #This command is basically superposing the entire housing data on t

valid.rows <- setdiff(rownames(housing), train.rows) #The setdiff() command is a SETS command which cho
```

```
valid.data <- housing[valid.rows, ]

# alternative code for ^^validation^^ (works only when row names are numeric): # collect all the column

# Just in case if you also want to add a TEST PARTITION in your data you would execute the following co
#   test.rows <- setdiff(rownames(housing), union(train.rows, valid.rows))
#   test.data <- housing[test.rows, ]
```

#Chapter 3:- Data Visualization

1.Histograms 2.Box Plots 3.Bar Graphs 4.HeatMaps

"{r} ## simple heatmap of correlations (without values) heatmap(cor(housing), Rowv = NA, Colv = NA)

install.packages("reshape") library(ggplot2) library(reshape) # to generate input for the plot cor.mat <- round(cor(housing),2) # rounded correlation matrix melted.cor.mat <- melt(cor.mat) ggplot(melted.cor.mat, aes(x = X1, y = X2, fill = value)) + geom_tile() + geom_text(aes(x = X1, y = X2, label = value))

#The command below gives you a heatmap of missing data heatmap(1 * is.na(housing), Rowv = NA, Colv = NA)

"

#Full of Errors......

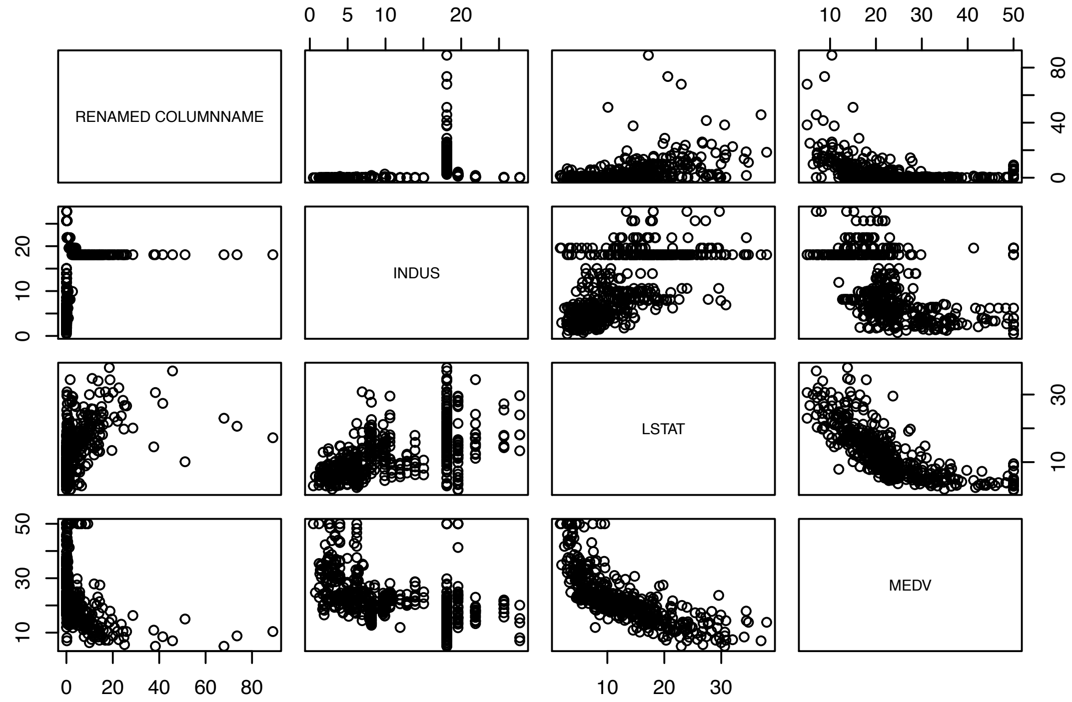
`{r} ## color plot par(xpd=TRUE) # allow legend to be displayed outside of plot area plot(housing$NOX ~ housing$LSTAT, ylab = "NOX", xlab = "LSTAT",col = ifelse(housing$CAT..MEDV == 1, "black", "gray")) # add legend outside of plotting area # In legend() use argument inset = to control the location of the legend relative # to the plot. legend("topleft", inset=c(0, -0.2),legend = c("CAT.MEDV = 1", "CAT.MEDV = 0"), col = c("black", "gray"),pch = 1, cex = 0.5) # alternative plot with ggplot library(ggplot2) ggplot(housing, aes(y = NOX, x = LSTAT, colour= CAT..MEDV)) + geom_point(alpha = 0.6) ## panel plots # compute mean MEDV per RAD and CHAS # In aggregate() use argument drop = FALSE to include all combinations # (exiting and missing) of RAD X CHAS. data.for.plot <- aggregate(housing$MEDV, by = list(housing$RAD, housing$CHAS),FUN = mean, drop = FALSE) names(data.for.plot) <- c("RAD", "CHAS", "meanMEDV") # plot the data par(mfcol = c(2,1)) barplot(height = data.for.plot$meanMEDV[data.for.plot$CHAS == 0], names.arg = data.for.plot$RAD[data.for.plot$CHAS == 0], xlab = "RAD", ylab = "Avg. MEDV", main = "CHAS = 0") barplot(height = data.for.plot$meanMEDV[data == 1], names.arg = data.for.plot$RAD[data.for.plot$CHAS == 1], xlab = "RAD", ylab = "Avg. MEDV", main = "CHAS = 1") # alternative plot with ggplot ggplot(data.for.plot) + geom_bar(aes(x = as.factor(RAD), y = \`meanMEDV\`), stat = "identity") + xlab("RAD") + facet_grid(CHAS ~ .)   NOX0.4 0.5 0.6 0.7 0.8`

#Simple Plot

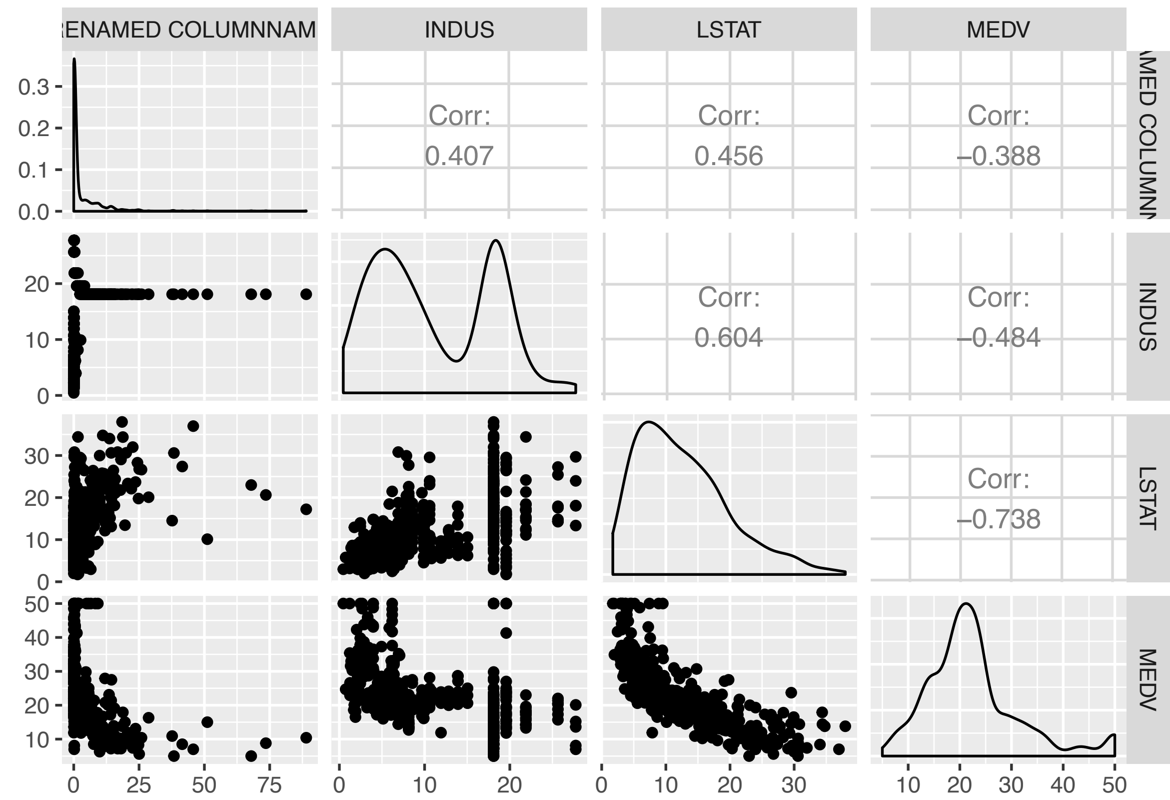
```
# use plot() to generate a matrix of 4X4 panels with variable name on the diagonal,
# and scatter plots in the remaining panels.

plot(housing[, c(1, 3, 12, 13)])


# ALTERNATIVE, nicer plot (displayed)

#install.packages("GGally")
library(GGally)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```




```r
ggpairs(housing[, c(1, 3, 12, 13)])
```



#RESCALING to view the visualization in a better way.

"{r}

scatter plot: regular and log scale

plot(housing$MEDV\ housing$CRIM, xlab = "CRIM", ylab = "MEDV") # to use logarithmic scale set argument log = to either 'x', 'y', or 'xy'. plot(housing$MEDV\ housing$CRIM,xlab = "CRIM", ylab = "MEDV", log = 'xy')

ALTERNATIVE log-scale plot with ggplot

library(ggplot2) ggplot(housing) + geom_point(aes(x = CRIM, y = MEDV)) + scale_x_log10(breaks = 10^(-2:2), labels = format(10^(-2:2), scientific = FALSE, drop0trailing = TRUE)) + scale_y_log10(breaks = c(5, 10, 20, 40)) "

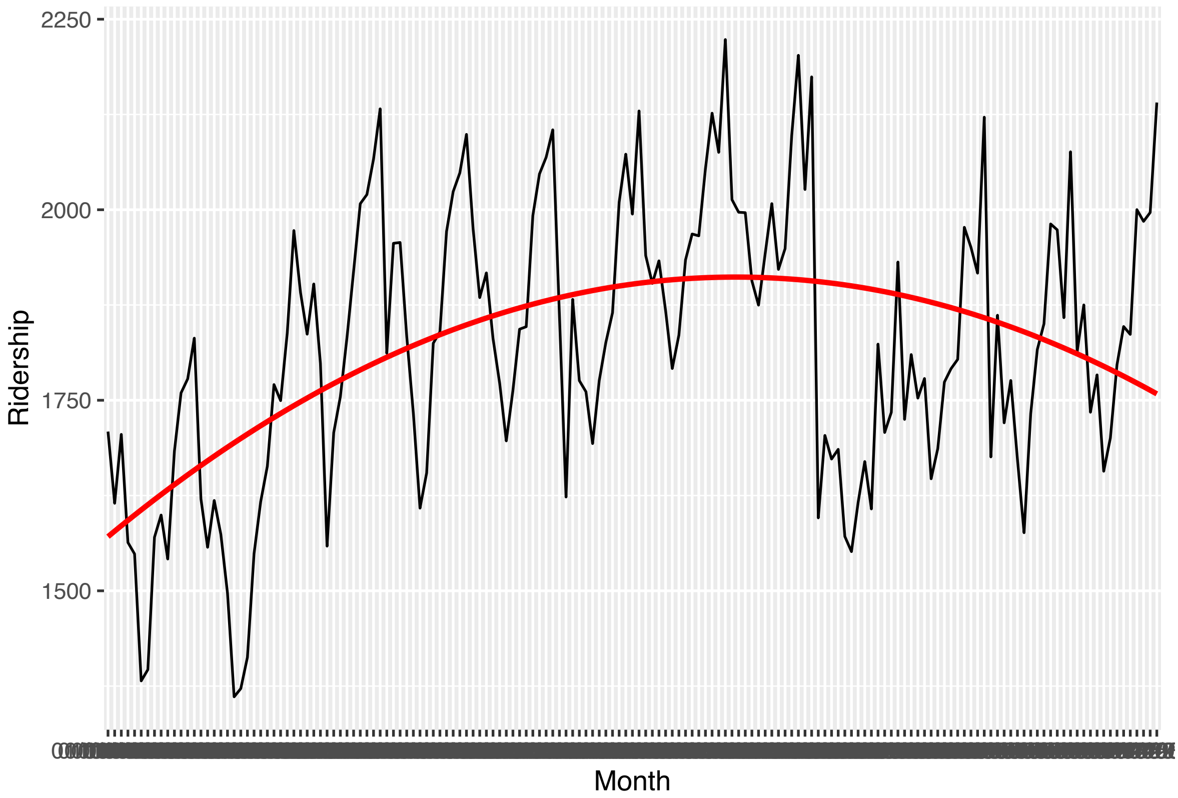
#See and learn how to draw this graph

Read about:- 1. aes 2. geom_line() 3. geaom_smooth()

#GGPLOT IS CERTAINLY ONE OF THE MOST IMPORTANT LIBRARIES IN R.

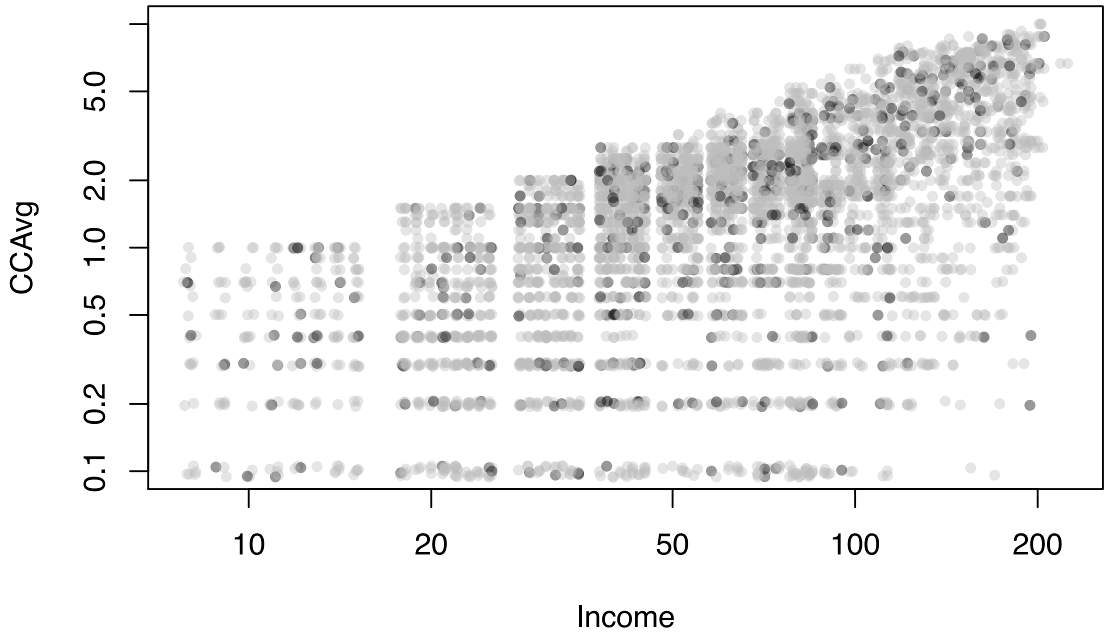
```
# alternative plot with ggplot
library(ggplot2)
Amtrak.df <- read.csv("Amtrak.csv")
```

```r
ggplot(Amtrak.df, aes(y = Ridership, x = Month, group = 12)) +
geom_line() + geom_smooth(formula = y ~ poly(x, 2), method= "lm", colour = "red", se = FALSE, na.rm = TI
```



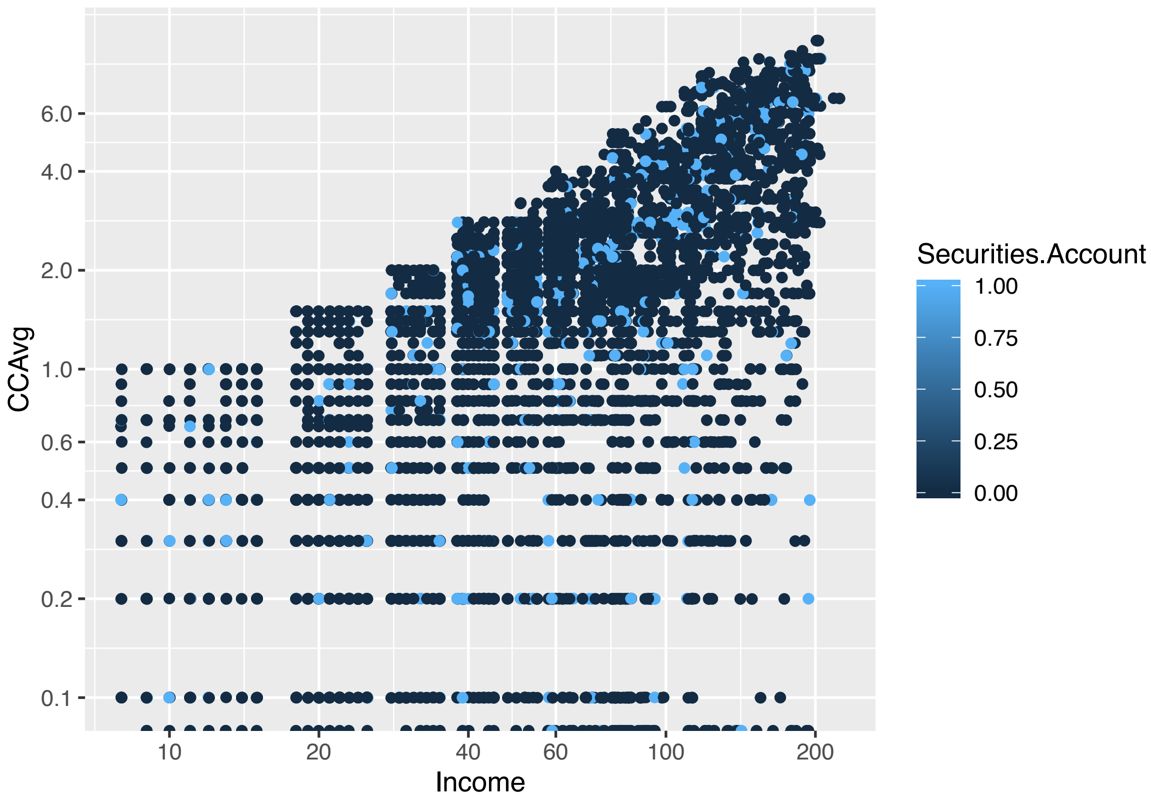

```r
#install.packages("scales")
library(scales)
universal.df <- read.csv("UniversalBank.csv")
plot(jitter(universal.df$CCAvg, 1) ~ jitter(universal.df$Income, 1),
col = alpha(ifelse(universal.df$Securities.Account == 0, "gray", "black"), 0.4), pch = 20, log = 'xy', y
xlab = "Income", ylab = "CCAvg")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 53 y values <= 0 omitted
## from logarithmic plot
```




```
# ALTERNATIVE with ggplot

library(ggplot2)
ggplot(universal.df) +
geom_jitter(aes(x = Income, y = CCAvg, colour = Securities.Account)) + scale_x_log10(breaks = c(10, 20,
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



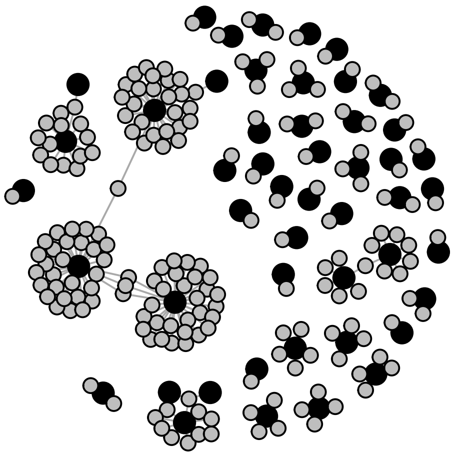
#NETWORK GRAPH

Understand the Graph properly EACH and every line of code

```
#install.packages("igraph")
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
ebay.df <- read.csv("eBayNetwork.csv")
# transform node ids to factors
ebay.df[,1] <- as.factor(ebay.df[,1])
ebay.df[,2] <- as.factor(ebay.df[,2])
graph.edges <- as.matrix(ebay.df[,1:2])
g <- graph.edgelist(graph.edges, directed = FALSE)
isBuyer <- V(g)$name %in% graph.edges[,2]
plot(g, vertex.label = NA, vertex.color = ifelse(isBuyer, "gray", "black"), vertex.size = ifelse(isBuye
```



#TreeMaps

{r} install.packages("treemap") library(treemap) tree.df <- read.csv("EbayTreemap.csv") # add column for negative feedback tree.df$negative.feedback <- 1* (tree.df$Seller.Feedback < 0) # draw treemap treemap(tree.df, index = c("Category","Sub.Category", "Brand"), vSize = "High.Bid", vColor = "negative.feedback", fun.aggregate = "mean", align.labels = list(c("left", "top"), c("right", "bottom"), c("center", "center")), palette = rev(gray.colors(3)), type = "manual", title = "")

"{r} library(ggmap) SCstudents <- read.csv("SC-US-students-GPS-data-2016.csv")

#In your GCP Console the API Key that you need to use for Google Maps in R is "Maps Static API"….Make sure you disable it once done using because it is NOT for FREE.

register_google(key = "AIzaSyDDLm-37lvXKv2ItpvPJDMm5tETM9tkOPo", write = TRUE)
Map <- get_map("Oklahoma City,OK", zoom = 4) ggmap(Map) + geom_point(aes(x = longitude, y = latitude), data = SCstudents,alpha = 0.5, colour = "red", size = 0.8) "

```
library(mosaic)
```

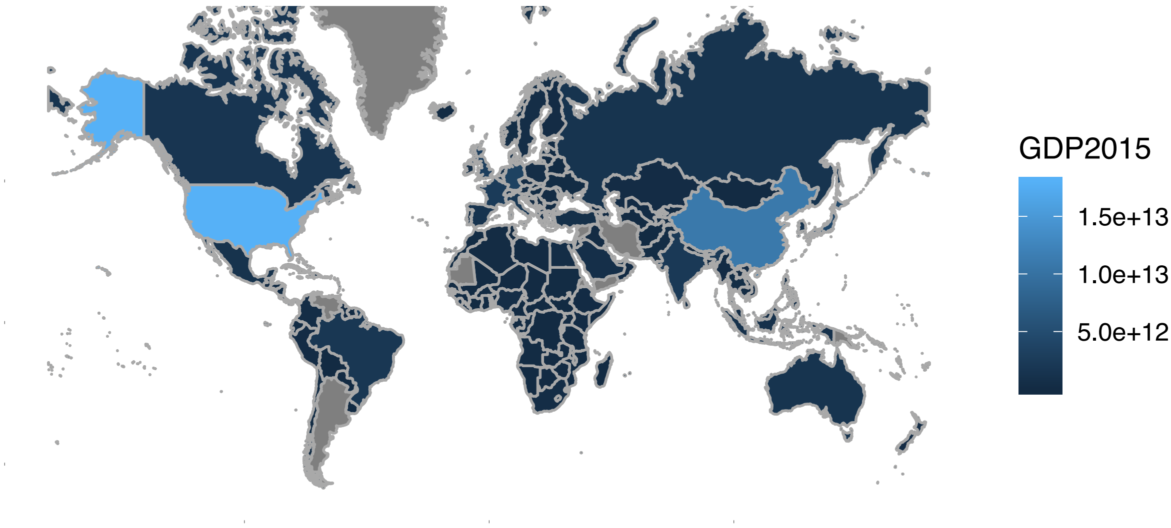
```
## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union
```

```
## The following object is masked from 'package:GGally':
##
##     nasa

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:scales':
##
##     rescale

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median,
##     prop.test, quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```r
#install.packages("lattice")
#install.packages("ggformula")
#install.packages("ggstance")
#install.packages("mapproj")

gdp.df <- read.csv("gdp.csv", skip = 4, stringsAsFactors = FALSE)
names(gdp.df)[5] <- "GDP2015"
happiness.df <- read.csv("Veerhoven.csv")
# gdp map
mWorldMap(gdp.df, key = "Country.Name", fill = "GDP2015") + coord_map()
```

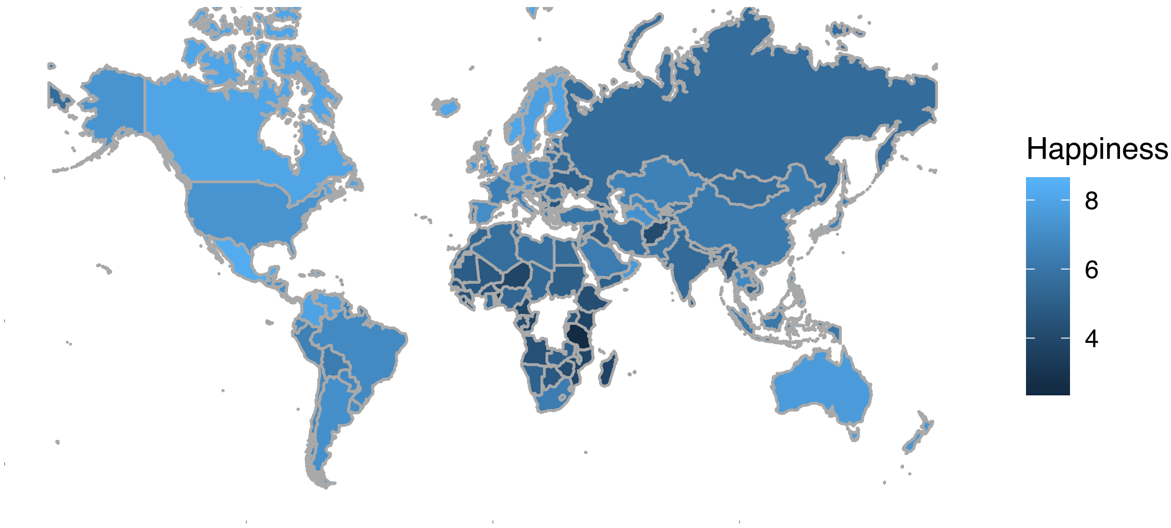
```
## Mapping API still under development and may change in future releases.

## Warning in standardName(x, countryAlternatives, ignore.case =
## ignore.case, : 52 items were not translated
```




```r
# eell-being map
mWorldMap(happiness.df, key = "Nation", fill = "Score") + coord_map() +
scale_fill_continuous(name = "Happiness")
```

```
## Mapping API still under development and may change in future releases.
```

```
## Warning in standardName(x, countryAlternatives, ignore.case =
## ignore.case, : 9 items were not translated
```



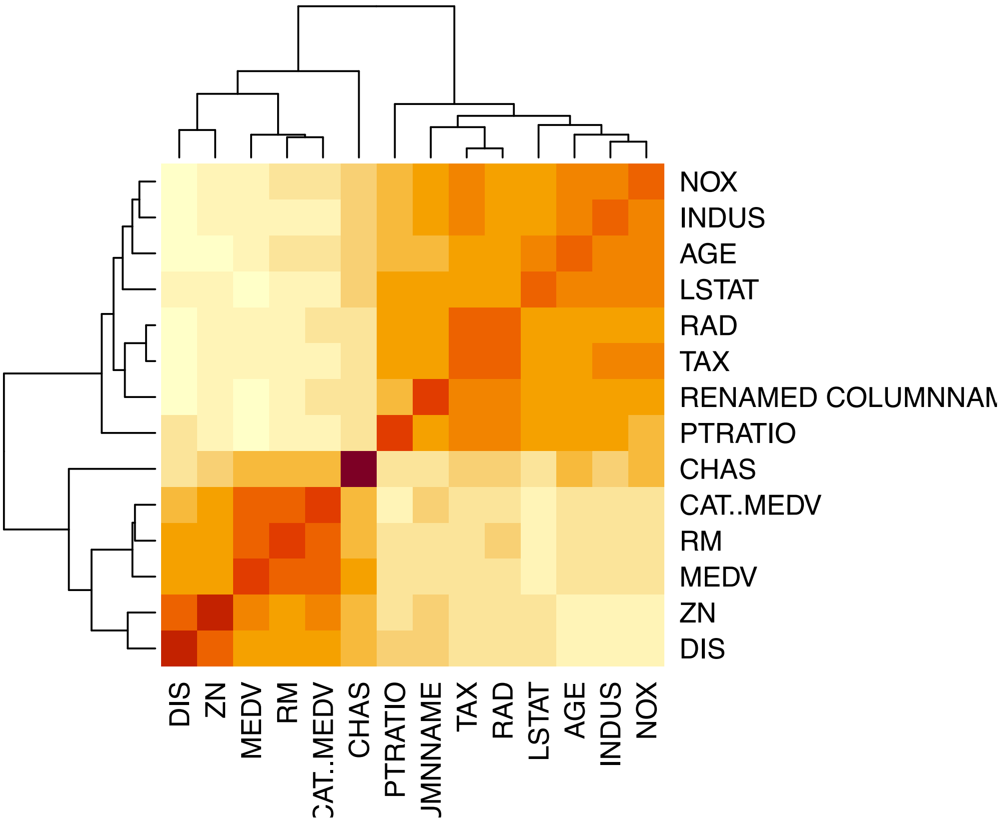
Prediction • Plot outcome on the y-axis of boxplots, bar charts, and scatter plots. • Study relation of outcome to categorical predictors via side-by-side box- plots, bar charts, and multiple panels. • Study relation of outcome to numerical predictors via scatter plots. • Use distribution plots (boxplot, histogram) for determining needed trans- formations of the outcome variable (and/or numerical predictors). • Examine scatter plots with added color/panels/size to determine the need for interaction terms. • Use various aggregation levels and zooming to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

Classification • Study relation of outcome to categorical predictors using bar charts with the outcome on the y-axis. • Studyrelationofoutcometopairsofnumericalpredictorsviacolor-coded scatter plots (color denotes the outcome). • Study relation of outcome to numerical predictors via side-by-side box- plots: Plot boxplots of a numerical variable by outcome. Create similar displays for each numerical predictor. The most separable boxes indicate potentially useful predictors. • Use color to represent the outcome variable on a parallel coordinate plot. • Use distribution plots (boxplot, histogram) for determining needed trans- formations of numerical predictor variables. • Examine scatter plots with added color/panels/size to determine the need for interaction terms. • Use various aggregation levels and zooming to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

Time Series Forecasting • Create line graphs at different temporal aggregations to determine types of patterns. • Use zooming and panning to examine various shorter periods of the series to determine areas of the data with different behavior. • Use various aggregation levels to identify global and local patterns. • Identify missing values in the series (that will require handling). • Overlay trend lines of different types to determine adequate modeling choices.

Unsupervised Learning • Create scatter plot matrices to identify pairwise relationships and cluster- ing of observations. • Use heatmaps to examine the correlation table. • Use various aggregation levels and zooming to determine areas of the data with different behavior. • Generate a parallel coordinates plot to identify clusters of observations.

```r
heatmap(cor(housing))
```



#Principal Component Analysis

```r
cereals.df <- read.csv("Cereals.csv")
# compute PCs on two dimensions
pcs <- prcomp(data.frame(cereals.df$calories, cereals.df$rating))
summary(pcs)
```

```
## Importance of components:
##                            PC1    PC2
## Standard deviation     22.3165 8.8844
## Proportion of Variance  0.8632 0.1368
## Cumulative Proportion   0.8632 1.0000
```

```r
pcs$rot
```

```
##                            PC1       PC2
## cereals.df.calories  0.8470535 0.5315077
## cereals.df.rating   -0.5315077 0.8470535
```

```r
scores <- pcs$x
head(scores, 5)
```

```
##              PC1        PC2
## [1,] -44.921528  2.1971833
## [2,]  15.725265 -0.3824165
## [3,] -40.149935 -5.4072123
## [4,] -75.310772 12.9991256
## [5,]   7.041508 -5.3576857
```

#Evaluating Predictive Performance In this chapter we will evaluate how data mining methods can be assessed on the basis of the following Prediction Accuracy Measures:- 1. Avreage Error 2. MAPE 3. RMSE 4. MPE 5. RMSE

Shown below is how to do Performance Evaluation:-

```r
# package forecast is required to evaluate performance
library(forecast)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```r
# load file
toyota.corolla.df <- read.csv("ToyotaCorolla.csv")

# randomly generate training and validation sets
training <- sample(toyota.corolla.df$Id, 600)
validation <- sample(setdiff(toyota.corolla.df$Id, training), 400)

# run linear regression model
reg <- lm(Price~., data=toyota.corolla.df[,-c(1,2,8,11)], subset=training,na.action=na.exclude)
pred_t <- predict(reg, na.action=na.pass)
pred_v <- predict(reg, newdata=toyota.corolla.df[validation,-c(1,2,8,11)],na.action=na.pass)
```

```
## Warning in predict.lm(reg, newdata = toyota.corolla.df[validation, -c(1, :
## prediction from a rank-deficient fit may be misleading
```

```r
# Performance Evaluation

# training
accuracy(pred_t, toyota.corolla.df[training,]$Price)
```

```
##                     ME     RMSE      MAE       MPE    MAPE
## Test set -1.177212e-10 1051.666 797.3029 -1.006025 8.15141
```

```r
# validation
accuracy(pred_v, toyota.corolla.df[validation,]$Price)
```

```
##                ME    RMSE      MAE        MPE     MAPE
## Test set 100.5568 2646.66 944.7535 -0.1726984 8.758258
```