R Packages: Table of Contents

cluster

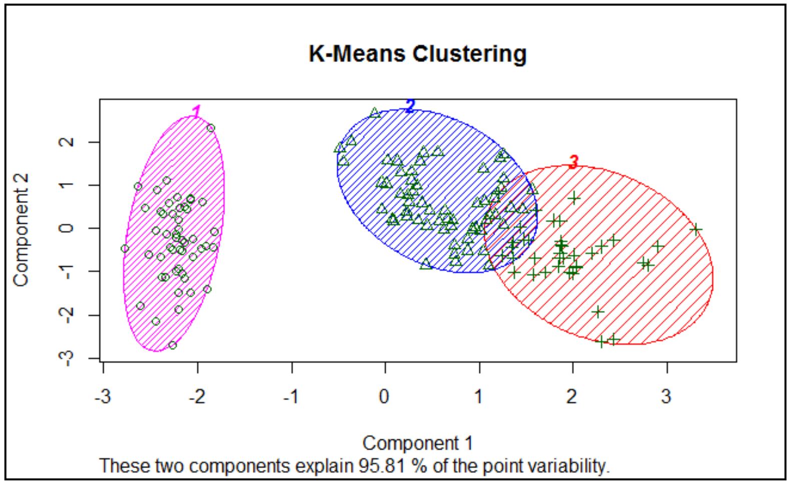
Draws a 2-Dimentional clustering plot

```
Sample Code input:

library(cluster)
data(iris)
dat <- iris[, -5] #without class label
# Kmeans clustre analysis
clus <- kmeans(dat, centers=3)
clusplot(dat, clus$cluster, color=TRUE,
shade=TRUE, labels=4, lines=0)
```

```
Code Output: (given code above what's
executed when it's run?)

> dat <- iris[, -5]
> clus <- kmeans(dat, centers=3)
> clusplot(dat, clus$cluster, color=TRUE,
shade=TRUE, labels=4, lines=0)
```



Used for:

- Cluster analysis
- Data mining

Application:

- Visualizing clustering result
- Used with k-means clustering

Important Functions:

- `kmeans()`
- Perform k-Means clustering
- `clusplot()`
- Plotting and visualizing clusters

e1071

Contains several statistical and machine learning functions

```
Sample Code input:

data(iris)
train <- subset(iris, select = -Species)
actual <- iris$Species
model <- svm(train, actual)
predicted <- predict(model, train)
table(predicted, actual)
```

Used for:

- Data analysis
- Machine learning
- Predictive modeling

Application:

- Classification problems
- Support Vector Machines (SVM), Naïve Bayes algorithms

Important Functions:

- `svm(traindata,class)`
- Train SVM model on given dataset
- `predict(model,dataset)`
- Predict the test data using trained model

```
Code Output: (given code above what's executed when it's run?)

> table(predicted, actual)
            actual
predicted    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
```

ggmap

Creates customized geographical location maps

Used for:

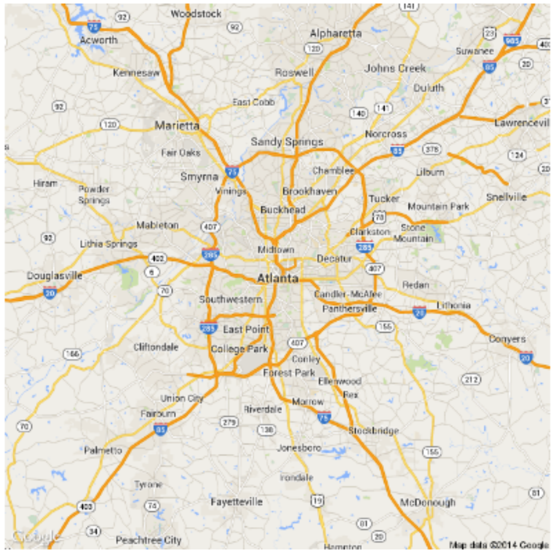
- Generating maps

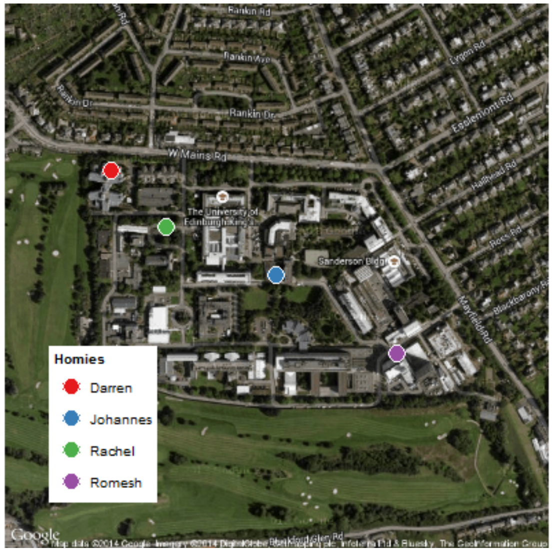
Application:

- Location based analytics
- Object tracking
- Realtime traffic updates

Important Functions:

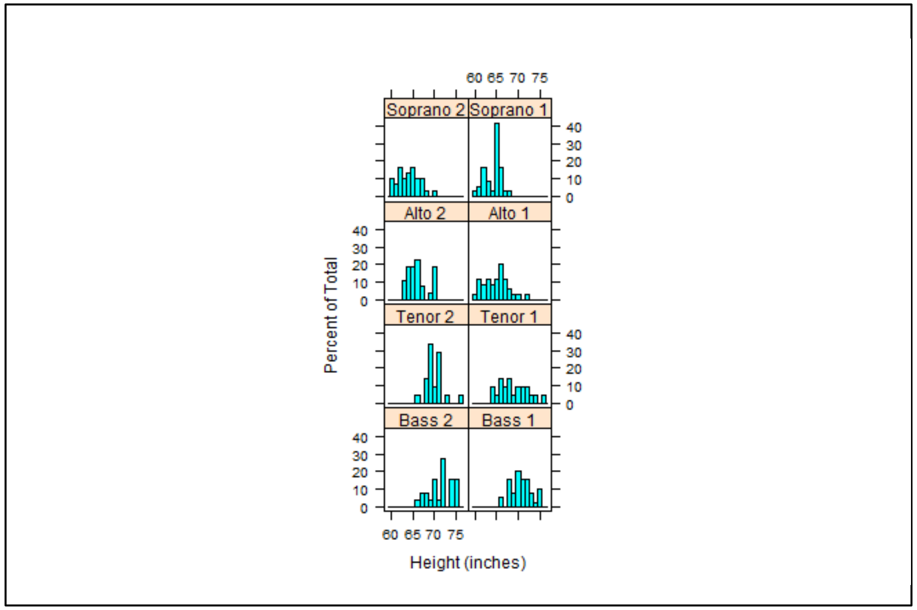
- `get_map()`
- Gets the map layout for given latitude and longitude data
- `ggmap()`
- Generates the customized map





lattice

A powerful and elegant high-level data visualization package

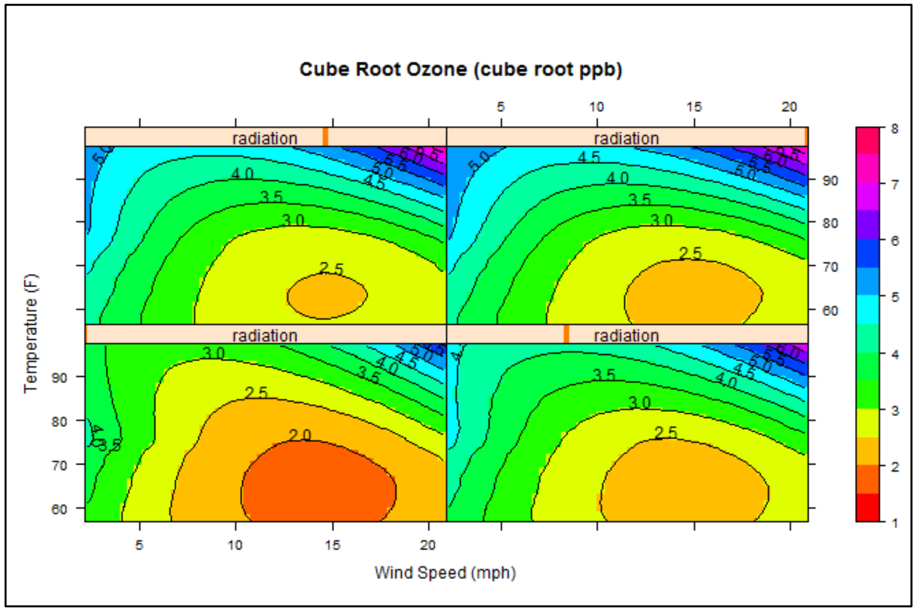


Used for:

- Lattice package is used to create complex graphs in a simple way

Application:

- Multiple histograms
- Contour graphs
- 3-Dimension plots



Important Functions:

- `histogram()`
- Plots multiple histograms in one frame
- `contourplot()`
- Generates contour graph with color intensity

RCurl

Working with HTTP

Used for:

- Working with the Web

Important Functions:

- `getURL()`
- Gets the contents of the URL

```
Sample Code input:

library(RCurl)
contents = getURL("http://www.google.com")
contents
```

```
Code Output: (given code above what's executed when it's run?)

> contents
[1] "<!doctype html><html itemscope=\"\" itemtype=\"http://schema.org/WebPage\"
lang=\"en\"><head><meta content=\"Search the world's information, including
webpages, images, videos and more. Google has many special features to help you find
exactly what you're looking for.\" name=\"description\"><meta content=\"noodp\"
name=\"robots\"><meta content=\"/images/google_favicon_128.png\" itemprop=\"image
\"><title>Google</title><script>
…
```

rjson

Working with JSON

Used for:

- Working with the Web

Important Functions:

- `fromJSON()`
- Converts JSON object into R object
- `toJSON()`
- Converts R object into JSON

```
Sample Code input:

library(rjson)
data = list(Age=c(22,33), Name=c("Mary", "John"),
Salary=c("40K", "30K"))
dataj = toJSON(data); dataj
fromJSON(dataj)
```

```
Code Output: (given code above what’s executed when it’s run?)

> dataj
[1] "{\"Age\":[22,33],\"Name\":[\"Mary\",\"John\"],\"Salary\":[\"40K\",\"30K\"]}”

> fromJSON(dataj)
$Age
[1] 22 33

$Name
[1] "Mary" "John"

$Salary
[1] "40K" "30K"
```

RODBC

Creates connection to multiple databases

```
Sample Code input:

library(RODBC)
myconn <- odbcConnect
('mydatasrc',uid='abc', pwd='xyz')
cust_data <- sqlQuery(myconn, 'select *
from CUSTOMER')
close(myconn)
```

Used for:

- Creating connection to several databases such as MySQL, Oracle, Postgres and so on.
- Fetching data at run-time directly from R

Application:

- Query data from table
- Create and delete tables directly from R

Important Functions:

- odbcConnect()
- Perform authentication and create connection to database
- sqlQuery()
- Executes query directly from R

```
Code Output: (given code above what's executed when it's run?)

> myconn <- odbcConnect ('mydatasrc',uid='abc', pwd='xyz')
> cust_data <- sqlQuery(myconn, 'select * from CUSTOMER')
> close(myconn)
```

rpart

Creates tree based models for regression and classification

```
Sample Code input:

library(rpart)
# create tree
fit <- rpart(Kyphosis ~ Age + Number +
Start, method="class", data=kyphosis)
# plot tree
plot(fit, uniform=TRUE,
main="Classification Tree for Kyphosis")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```

```
Code Output: (given code above what's
executed when it's run?)

> library(rpart)
> fit <- rpart(Kyphosis ~ Age + Number +
Start, method="class", data=kyphosis)
> plot(fit, uniform=TRUE,
main="Classification Tree for Kyphosis")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
> printcp(fit)
```

Used for:

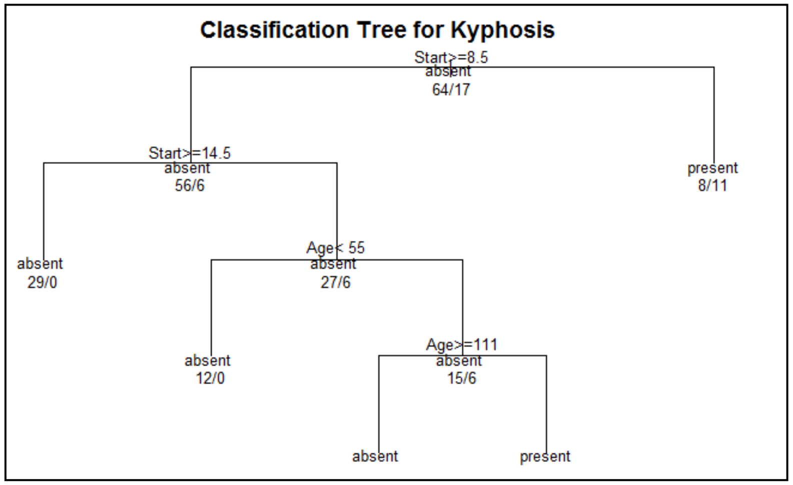
- Data analysis
- Data mining
- Machine learning

Application:

- Solving classification problems
- Solving regression problems

Important Functions:

- `fit(formula, method, data)`
- Creates a decision tree model for given formula
- `plot(model)`
- Visualizes the decision tree



wordcloud

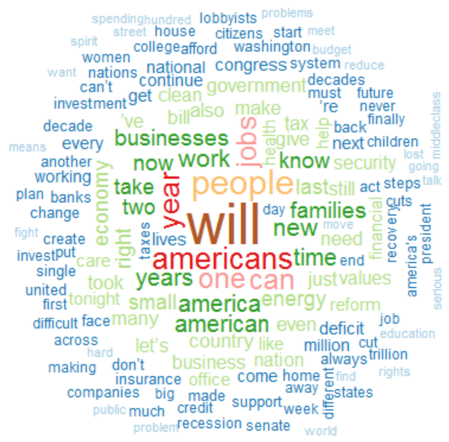
Creates a word cloud based on frequency/counts

```
Sample Code input:

data(SOTU)
text <- tm_map(text, function(x)
removeWords(tolower(x),stopwords()))
wordcloud(text[[1]], colors =
brewer.pal(12,"Paired"),random.order=FALSE
,max.words = 150, scale=c(3,0.5))
```

```
Code Output: (given code above what's
executed when it's run?)

> wordcloud(text[[1]], colors =
brewer.pal(12,"Paired"),random.order=FALSE
,max.words = 150, scale=c(3,0.5))
```



Used for:

- Text pining
- Text parsing
- Sentence segmentation

Application:

- Text visualization
- Natural Language Processing (NLP)

Important Functions:

- `tm_map()`
- Filter stop words, punctuations etc. from text
- `Wordcloud()`
- Creates visualization based on word frequency