How I Learned to Stop Worrying and Love the R Console

Irfan Kanat
Department of Information Systems
Arizona State University

November 2, 2015

Outline

Who am I?

Irfan Kanat, PhD Candidate

R user since 2006

Open Source Evangelist

What is this about?

A brief introduction to R.

- R Console
- Importing Data
- Packages
- Sample analyses
- Basic visualization
- Where to get help?
  - Documentation
  - Stack Exchange
  - R community

What is R?

From R project web site:
R is a language *and* an environment for statistical computing and graphics.

- Language
- Environment
- Statistics and Visualization

What is R?

All this means R is very flexible, which played a huge role in its success.

My take: Low cost, high quality, open source solution for your analysis needs.

When to Use R?

R is very strong for your classical machine learning and statistical analysis. Thousands of packages address almost all analysis needs. It is a logical first stop to start analysis.

[1] Except when it is not. There are packages to overcome these issues.

When to Use R?

R is very strong for your classical machine learning and statistical analysis. Thousands of packages address almost all analysis needs. It is a logical first stop to start analysis.

Yet it's core design is starting to show its age. There are certain down sides to traditional R:

- Everything is stored in memory[1]
- R is single core[1]

[1] Except when it is not. There are packages to overcome these issues.

Best Part of R

Packages CRAN houses over 7K packages. Providing functionality way beyond what is available in commercial packages.

Community Millions of users mean, all your questions are either already answered or will be in hours.

Performance While memory and core restrictions are real, for the cost of a single user license of a commercial package, you can buy better hardware to run R. Furthermore, with the packages providing multicore and flatfile functionality, R performance is on par or better than commercial packages

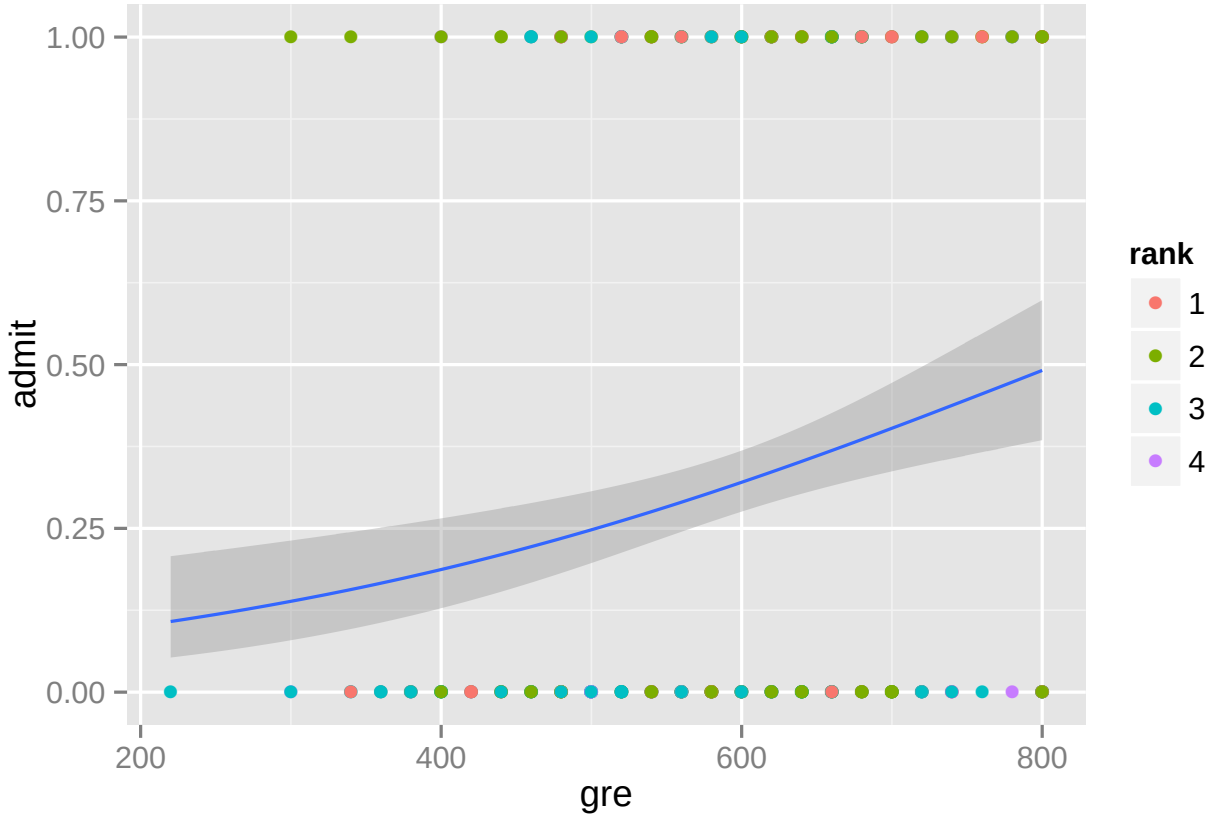
Outline

Logistic Regression

```
# Fit the model
logit_0 <- glm(admit ~ ., admitData, family = "binomial")
# Display fitted model
summary(logit_0)

##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = admitData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
```

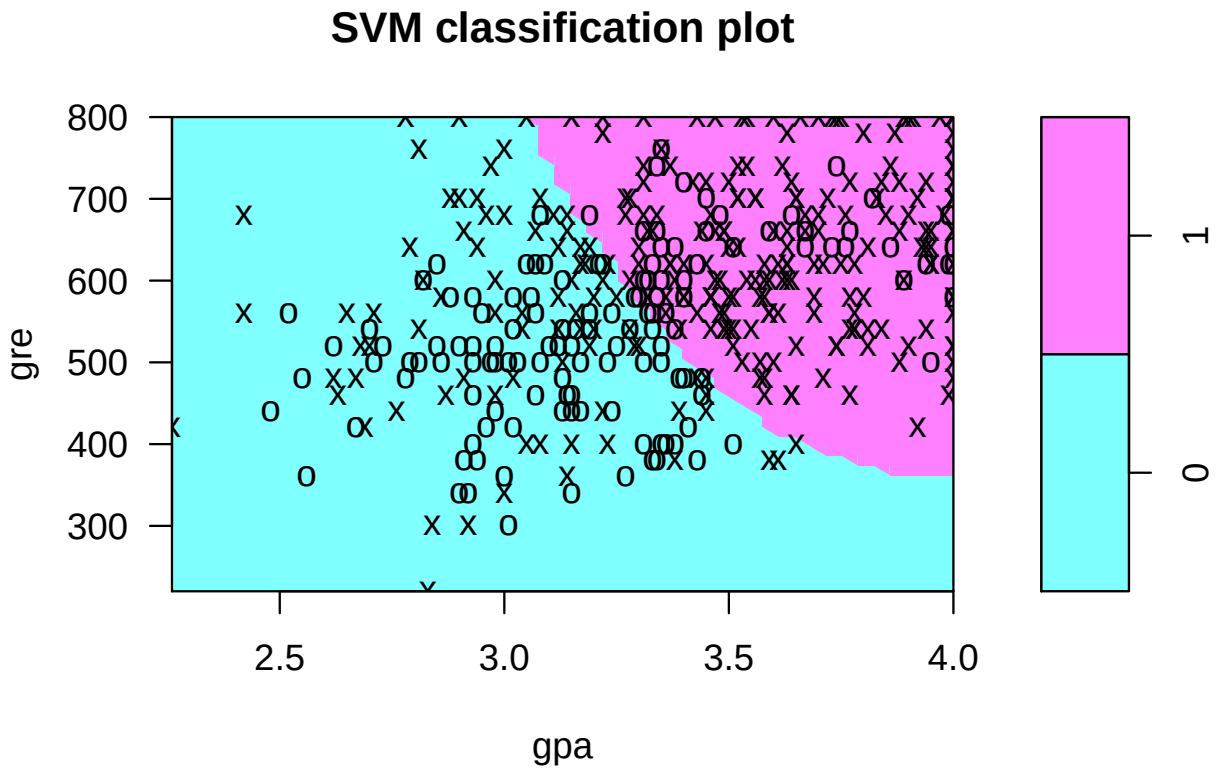
Logistic Regression

```
ggplot(admitData, aes(x = gre, y = admit)) + geom_point(aes(colour = rank))
    stat_smooth(method = "glm", family = "binomial", , se = T)
```

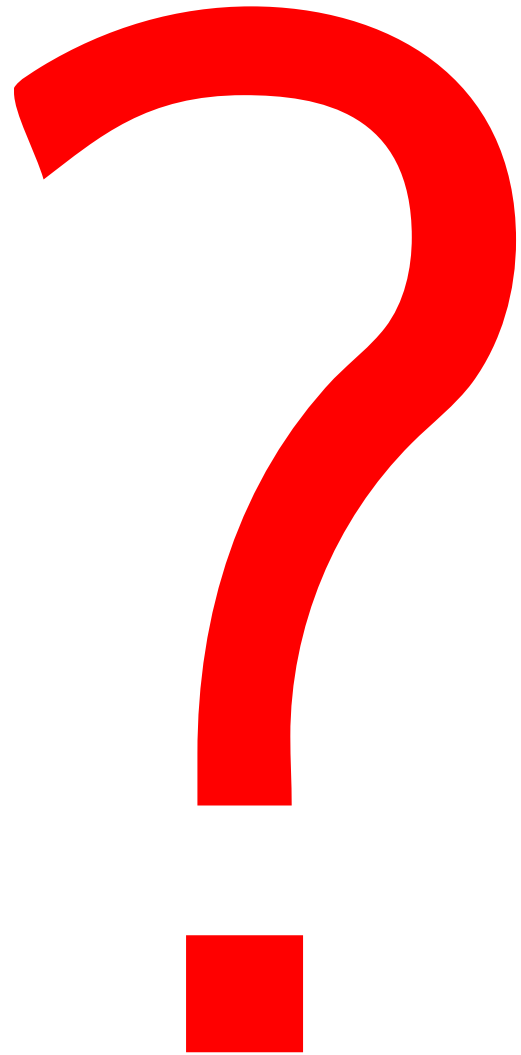


Support Vector Machine

```
# Fit the model
svm_0 <- svm(admit ~ ., data = admitData, type = "C-classification")
# Plot the results
plot(svm_0, admitData, gre ~ gpa)  # Let us plot the results
```



Questions



Outline

Command Driven Interface

Command line may be intimidating

Power or Convenience

Consider the number of

- Functions
- Parameters
- Data sources
- Variables
- Replications

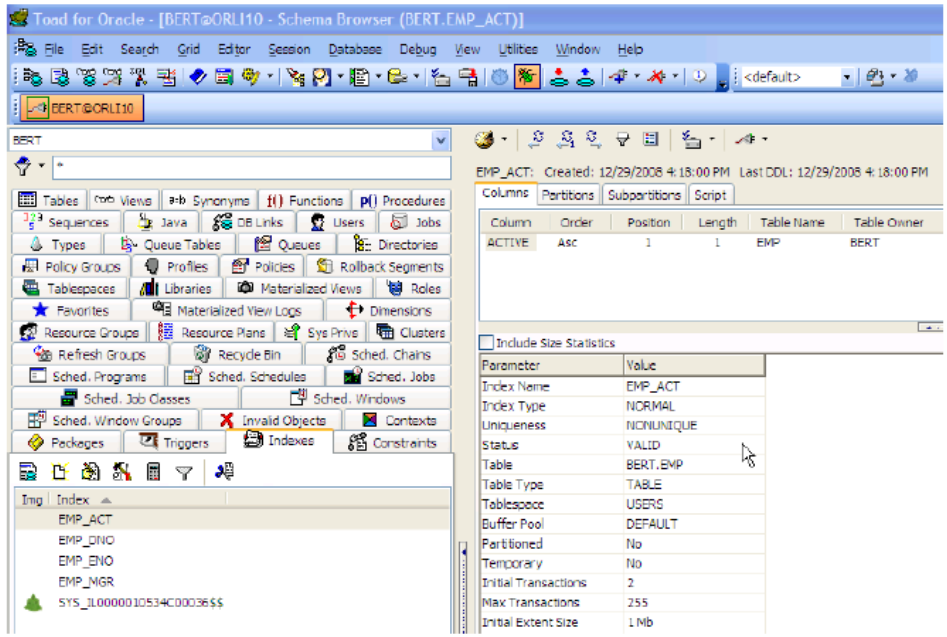
Command Driven Interface

Command line may be intimidating

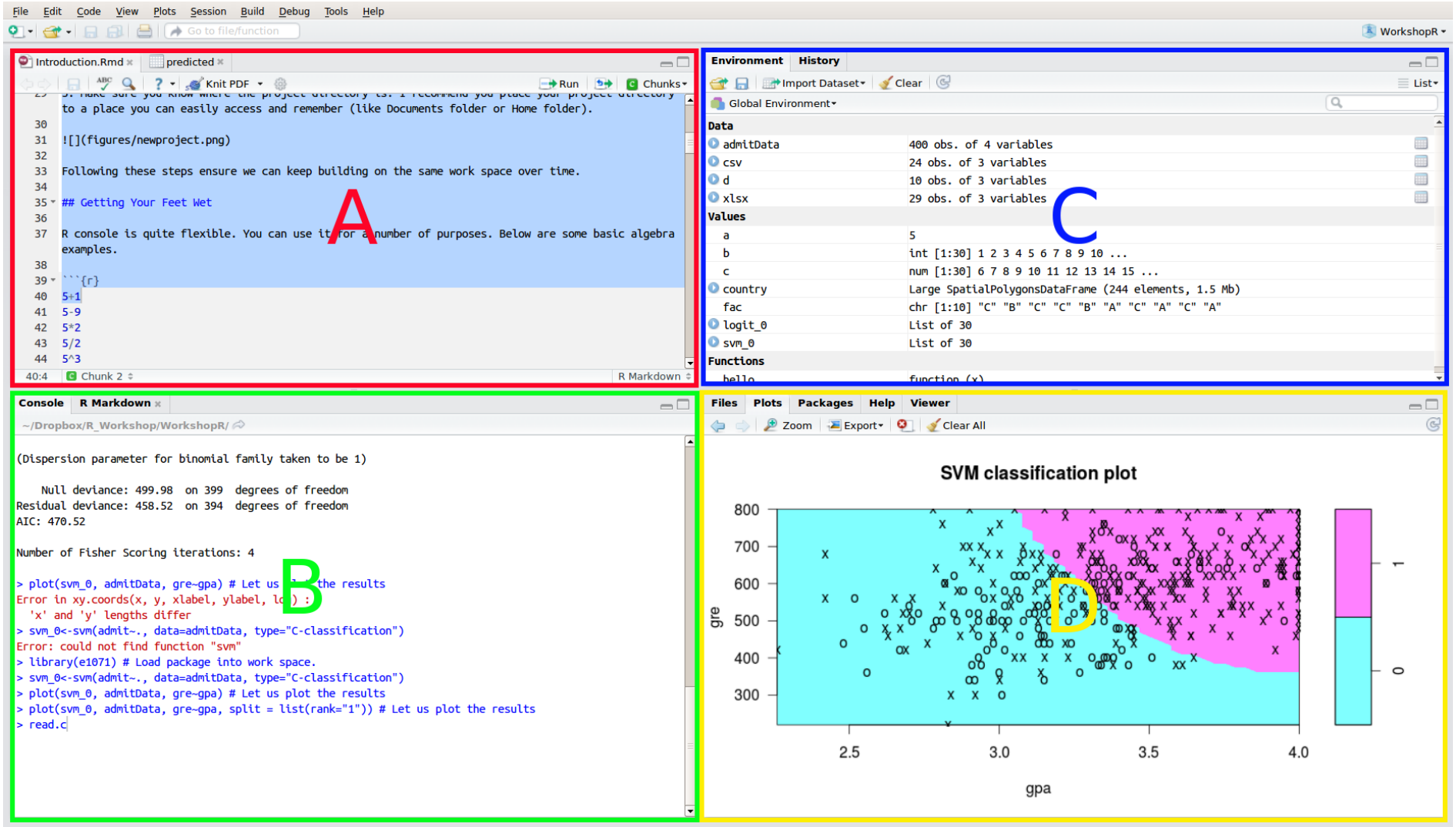
Power or Convenience

Consider the number of

- Functions
- Parameters
- Data sources
- Variables
- Replications



R Studio


File Edit Code View Plots Session Build Debug Tools Help
Introduction.Rmd
predicted
Knit PDF
Run
Chunks
to a place you can easily access and remember (like Documents folder or Home folder).

Following these steps ensure we can keep building on the same work space over time.
Getting Your Feet Wet
R console is quite flexible. You can use it for a number of purposes. Below are some basic algebra examples.
```{r}
5+1
5-9
5*2
5/2
5^3
A
Environment
History
Import Dataset
Clear
List
Global Environment
Data
admitData 400 obs. of 4 variables
csv 24 obs. of 3 variables
d 10 obs. of 3 variables
xlsx 29 obs. of 3 variables
Values
a 5
b int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
c num [1:30] 6 7 8 9 10 11 12 13 14 15 ...
country Large SpatialPolygonsDataFrame (244 elements, 1.5 Mb)
fac chr [1:10] "C" "B" "C" "C" "B" "A" "C" "A" "C" "A"
logit_0 List of 30
svm_0 List of 30
Functions
C
Console
R Markdown
~/Dropbox/R_Workshop/WorkshopR/
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 499.98 on 399 degrees of freedom
Residual deviance: 458.52 on 394 degrees of freedom
AIC: 470.52
Number of Fisher Scoring iterations: 4
> plot(svm_0, admitData, gre~gpa) # Let us the results
Error in xy.coords(x, y, xlabel, ylabel, lo) :
'x' and 'y' lengths differ
> svm_0<-svm(admit~., data=admitData, type="C-classification")
Error: could not find function "svm"
> library(e1071) # Load package into work space.
> svm_0<-svm(admit~., data=admitData, type="C-classification")
> plot(svm_0, admitData, gre~gpa) # Let us plot the results
> plot(svm_0, admitData, gre~gpa, split = list(rank="1")) # Let us plot the results
> read.c
B
Files
Plots
Packages
Help
Viewer
Zoom
Export
Clear All
SVM classification plot
gre
gpa
800
700
600
500
400
300
2.5
3.0
3.5
4.0
1
0
D
```

New Project

File > New Project

Empty Directory > Empty Project > Directory Name: Workshop

R as a Calculator I

```
# Arithmetics
2 + 2

## [1] 4

2 * 3

## [1] 6

2^3

## [1] 8

log(100, 10)

## [1] 2
```

R as a Calculator II

```
# Logic
1 == 2

## [1] FALSE

1 != 2

## [1] TRUE

2 < 3

## [1] TRUE
```

Variables I

Variables II

```
A <- 2

A

## [1] 2

a   # Case sensitive

## Error in eval(expr, envir, enclos):  object 'a' not found

"A" != "a"   # Explanation

## [1] TRUE

B <- 7

A + B

## [1] 9
```

Indexes and Data Frames I

```
1:30
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 2
## [24] 24 25 26 27 28 29 30
```

```
C[3]
```

```
## [1] 7
```

```
C[c(2, 3)]
```

```
## [1] 3 7
```

```
C[1:3]
```

```
## [1] 1 3 7
```

Indexes and Data Frames II

```
Countries <- data.frame(names=c("US","TR","DE"), supply=c(10, 8, 7),
                        those=c(TRUE, FALSE, FALSE))

Countries

##   names supply those
## 1    US     10  TRUE
## 2    TR      8 FALSE
## 3    DE      7 FALSE
```

Indexes and Data Frames III

```
Countries[2, ]

##   names supply those
## 2    TR      8 FALSE

Countries[, 3]

## [1]  TRUE FALSE FALSE

Countries[2, 3]

## [1] FALSE

Countries[1:2, ]

##   names supply those
## 1    US     10  TRUE
## 2    TR      8 FALSE
```

Loops in R

CAUTION!

R is notoriously inefficient with your classic loops

- Structure of the Data Frame
- Memory Management

Try to use an apply function instead.

Vectorize your operations.

For Loop in R

```
for (i in 1:3) print(i)

## [1] 1
## [1] 2
## [1] 3

# Iterating through a data frame
for (i in 1:nrow(Countries)) {
    print(Countries[i, ])
}

##   names supply those
## 1    US     10  TRUE
##   names supply those
## 2    TR      8 FALSE
##   names supply those
## 3    DE      7 FALSE
```

Functions I

```
ls()  # List the contents of the environment
```

```
##  [1] "A"          "admitData"  "B"          "c"          "C"
##  [6] "Countries"  "HelloWorld" "i"          "logit_0"    "logit_1"
## [11] "svm_0"
```

```
mean(C)  # Takes parameters
```

```
## [1] 5
```

```
mean(C, trim = 0.1, na.rm = T)  # Takes multiple parameters
```

```
## [1] 5
```

```
log(sum(C)/length(C))  # Can be combined
```

```
## [1] 1.609438
```

Functions II

```
HelloWorld <- function(x, y = 1) {
    for (i in 1:y) {
        print(paste("Hello", x))
    }
}

HelloWorld("MSBA")

## [1] "Hello MSBA"

HelloWorld("MSBA", 2)

## [1] "Hello MSBA"
## [1] "Hello MSBA"
```

Functions III

```
HelloWorld  # Review the source code

## function(x, y = 1) {
##     for (i in 1:y) {
##         print(paste("Hello", x))
##     }
## }

ls

## function (name, pos = -1L, envir = as.environment(pos), all.names = FALS
##     pattern, sorted = TRUE)
## {
##     if (!missing(name)) {
##         pos <- tryCatch(name, error = function(e) e)
##         if (inherits(pos, "error")) {
##             name <- substitute(name)
##             if (!is.character(name))
##                 name <- deparse(name)
##             warning(gettextf("%s converted to character string"
```

Commonly Used Functions I

```
ls()  # Get a list of objects in the workspace

##  [1] "A"          "admitData"  "B"          "c"          "C"
##  [6] "Countries"  "HelloWorld" "i"          "logit_0"    "logit_1"
## [11] "svm_0"

ls(pattern = "*_0")  # partial match on object search

## [1] "logit_0" "svm_0"

rm("svm_0")  # Remove an object from the workspace

# rm(list=ls(pattern=ls())) # This would remove everything if ran
```

Commonly Used Functions II

```
mean(A)   # Mean

## [1] 2

sd(admitData[, "gre"])   # Standard Deviation

## [1] 115.5165

AIC(logit_0)

## [1] 470.5175
```

Commonly Used Functions III

```
str(Countries)  # Look at the structure of objects
```

```
## 'data.frame': 3 obs. of  3 variables:
##  $ names : Factor w/ 3 levels "DE","TR","US": 3 2 1
##  $ supply: num  10 8 7
##  $ those : logi  TRUE FALSE FALSE
```

```
summary(Countries)  # Get summary of data
```

```
##  names      supply         those
##  DE:1   Min.   : 7.000   Mode :logical
##  TR:1   1st Qu.: 7.500   FALSE:2
##  US:1   Median : 8.000   TRUE :1
##         Mean   : 8.333   NA's :0
##         3rd Qu.: 9.000
##         Max.   :10.000
```

Commonly Used Functions IV

```
psych::describe(admitData)  # Better version of summary stats in psych pack

##        vars   n   mean     sd median trimmed    mad    min max  range  sk
## admit     1 400   0.32   0.47    0.0    0.27   0.00   0.00   1   1.00  0.
## gre       2 400 587.70 115.52  580.0  589.06 118.61 220.00 800 580.00 -0.
## gpa       3 400   3.39   0.38    3.4    3.40   0.40   2.26   4   1.74 -0.
## rank*     4 400   2.48   0.94    2.0    2.48   1.48   1.00   4   3.00  0.
##        kurtosis   se
## admit     -1.39 0.02
## gre       -0.36 5.78
## gpa       -0.60 0.02
## rank*     -0.91 0.05
```

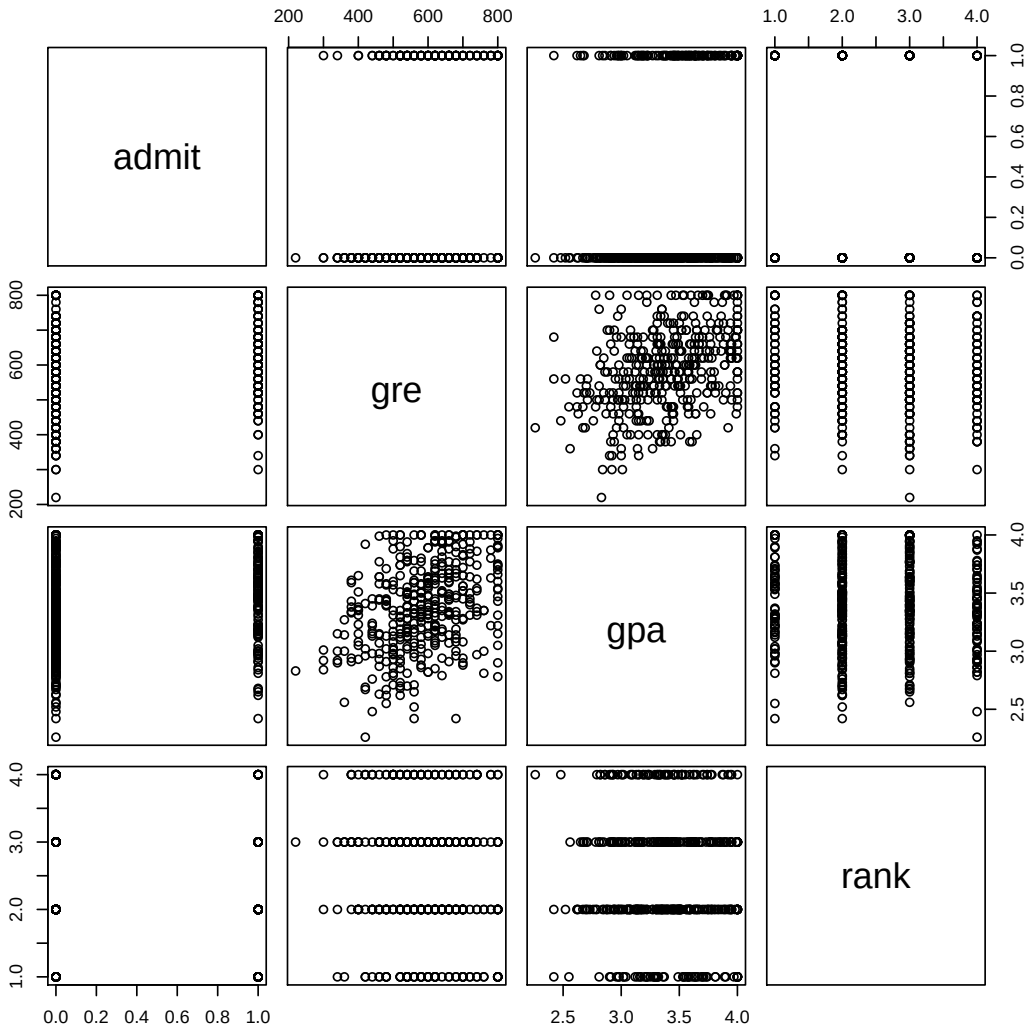
Commonly Used Functions V

```
summary(logit_1)  # Get summary of model


##
## Call:
## glm(formula = admit ~ gre, family = "binomial", data = admitData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1623  -0.9052  -0.7547   1.3486   1.9879
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.901344   0.606038  -4.787 1.69e-06 ***
## gre          0.003582   0.000986   3.633  0.00028 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 486.06  on 398  degrees of freedom
## AIC: 490.06
##
## Number of Fisher Scoring iterations: 4
```

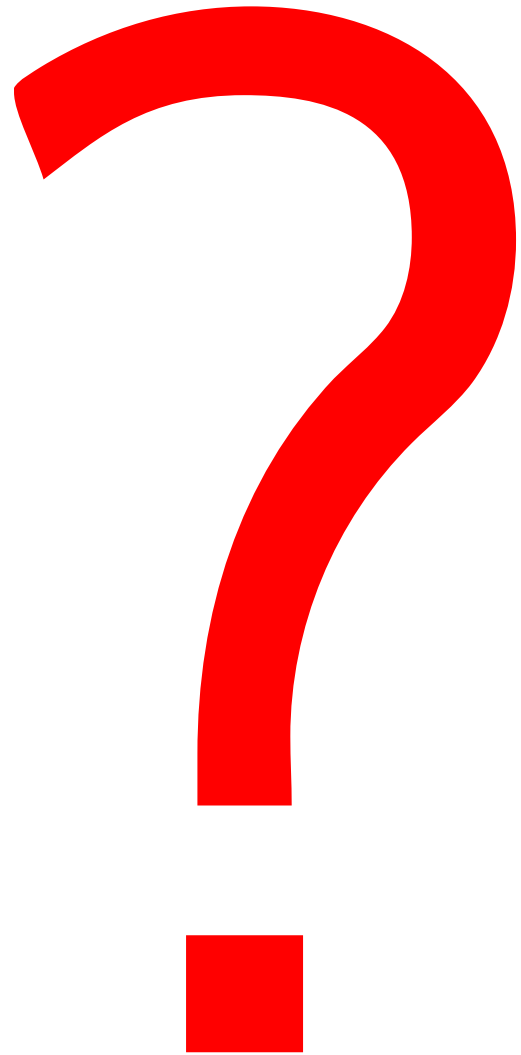
Commonly Used Functions VI

```
cor(admitData[, 1:3])  # Get correlation matrix

##              admit       gre       gpa
## admit 1.0000000 0.1844343 0.1782123
## gre   0.1844343 1.0000000 0.3842659
## gpa   0.1782123 0.3842659 1.0000000

pairs(admitData)  # Visualize correlation table
```

Commonly Used Functions VII



Questions



Outline

Importing Data

R allows importing data from a wide variety of sources.

- Comma Separated Values (CSV)
- Databases
- Flat files
- Lesser statistical packages
- and more

Importing CSV Files

CSV has certain advantages that make it popular.

- Compatibility
- Flexibility
- Simplicity

Sample

"iso2", "Supply", "Those"
"AU", 20, 0
"TR", 80, 1
"US", 100, 0
"GB", 50, 0
"DE", 70, 0

We use read.csv() or read.csv2() commands to import the csv files.

```
saveData <- read.csv("PathToCSV", header = TRUE, sep = ",", quote = "\"", )
```

Working with Excel Files

Much like CSV, except it lacks the simplicity, flexibility, and compatibility of CSV.

```
# Load the necessary library
library(xlsx)
# Read in the data from excel file
xlsx <- read.xlsx("country.xlsx", sheetIndex = 1)
```

Working with Databases

No speed advantage.

Data larger than memory.

Working with databases:

- Work in the database.
- Import data from database.

Working with Databases

```
# Load the necessary library
library(RMySQL)
# Establish connection to the database.
channel <- dbConnect(MySQL(), user = "uname", password = "pwd", host = "127
    dbname = "exampledata")
# Send query and save results in R workspace
sql <- dbGetQuery(channel, "SELECT * FROM table;")
```

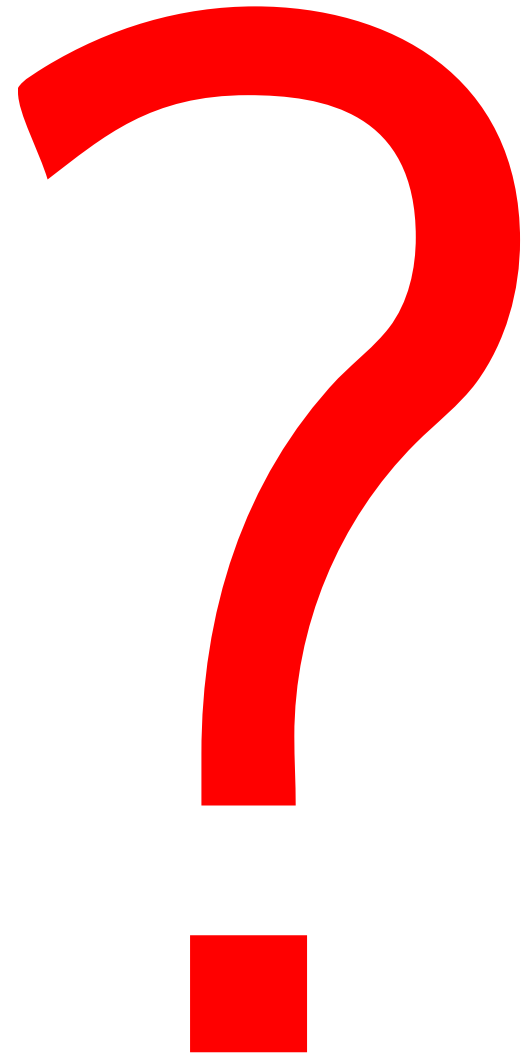
Lesser Statistical Packages :P

Foreign Package

Newer file formats

- sas7bdat
- readstata13

Questions



Outline

Packages: Source of R's Power

Encountered already

Make R extendible

Like libraries

Collection of:

- functions
- documentation
- data files

Gifts from the Community

Currently over 7000 packages

for

- Statistical Modeling
- Machine Learning
- Data Manipulation
- Visualization
- ...

from

- Economics
- Computer Science
- Statistics
- Medicine
- ...

Great but Where are My Gifts?

Comprehensive R Archive Network (CRAN)

A Group of FTP and HTML servers hosting R packages.

R has built in package management facilities.

Package Management

```
# Installing a package
install.packages("e1071")  # Notice the quotes around package name
# Loading package into memory
library(e1071)  # Notice the lack of quotes
# Unload package
detach("package:e1071", unload = TRUE)  # Notice the package: prepended
```

```
# Get the list of packages loaded
(.packages())
```

```
##  [1] "e1071"     "ggplot2"   "knitr"     "stats"     "graphics"
##  [6] "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
# Get list of all installed packages (part of output omitted)
.packages(all.available = T)
```

```
##   [1] "acepack"      "assertthat"   "bdsmatrix"    "betareg"
##   [5] "BH"           "biglm"        "bit"          "bitops"
##   [9] "brew"         "car"          "caTools"      "chron"
```

How to Find Packages

If you want to search a certain word in installed packages' documentation, you can always use ?? or help.search()

```
??mixed
help.search("mixed model")
```

Internet searches are a bit problematic as R can be a bit ambiguous until Google learns you are interested in the statistical computing environment.

Comprehensive R Archive Network (CRAN)

R Forge

R site search also available with command RSiteSearch()

R seek

Commonly Used Packages: Data Manipulation

data.tables Replaces traditional data.frame.

- Faster access/write
- Improved selection
- Improved subsetting
- Improved aggregation

Not a drop-in replacement as it breaks compatibility in some cases.

ddplyr

Additional functionality for:

- selection
- filtering
- aggregation

Provides efficient back-end data structures to speed things up.
Works with databases as well.

Commonly Used Packages: Statistics

Multivariate Regression: Base package
Generalized Linear Models: glm package

Traditional Econometric Models: plm package

Mixed Modeling: nlme and lme4 packages

Commonly Used Packages: Machine Learning

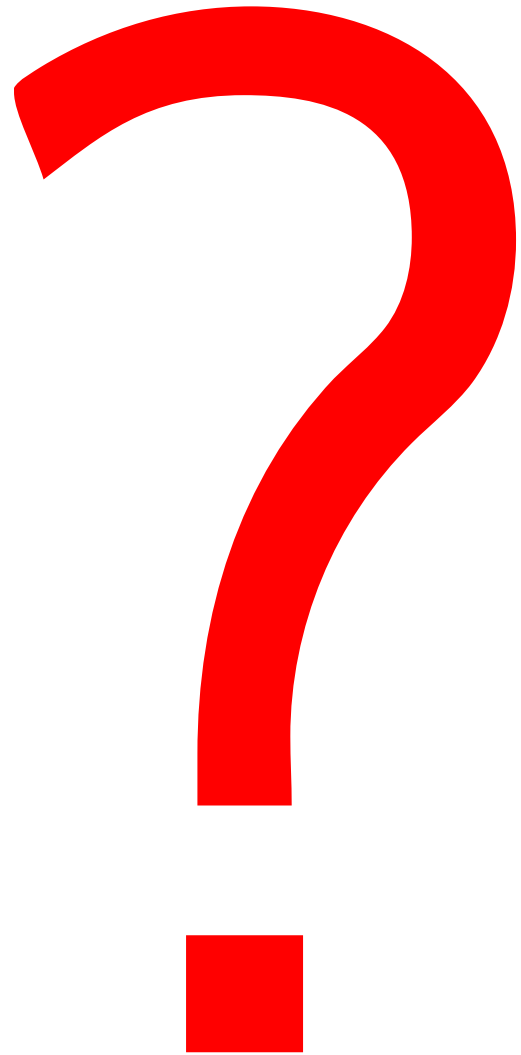
Classifiers (KNN, LVQ): class package

Support Vector Machines: kernlab, e1071 packages

Clustering: Base package (kmeans(), hclust()), mclust package

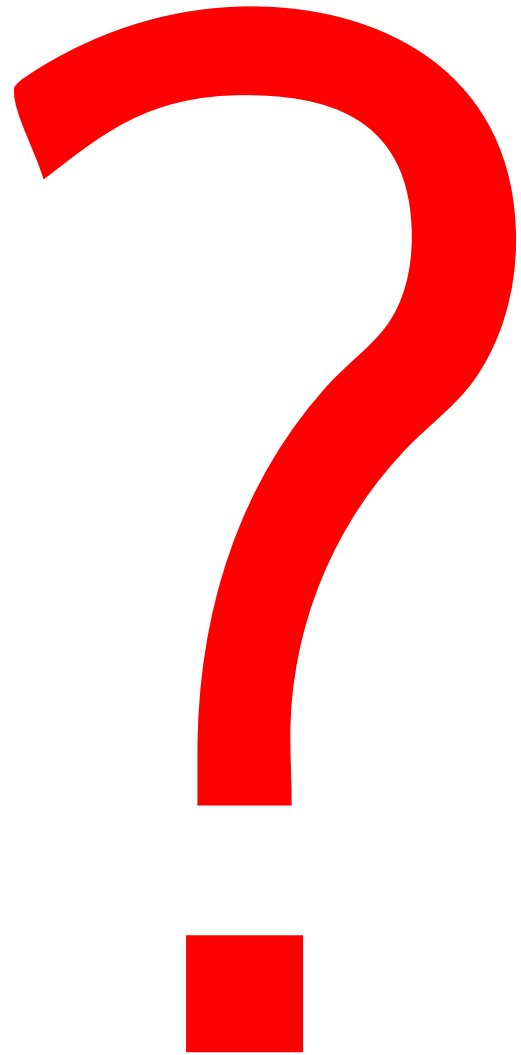
Neural Networks: neuralnet package.

Questions



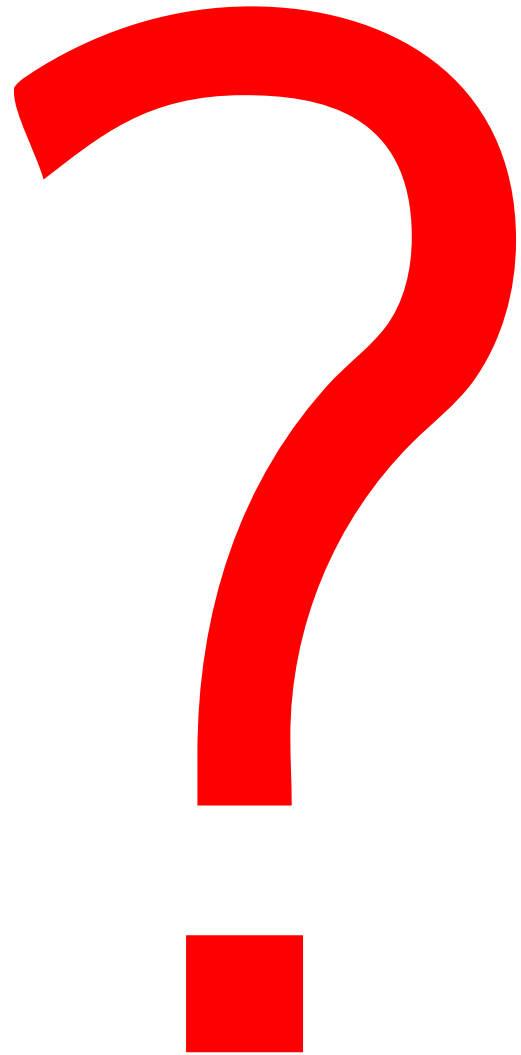
Outline

Questions



Outline

Questions



Outline