

How I Learned to Stop Worrying and Love the R Console

Irfan Kanat
Department of Information Systems
Arizona State University

November 3, 2015

Outline

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

Who am I?

Irfan Kanat, PhD Candidate

R user since 2006

Open Source Evangelist

Before We Begin

Got R & R Studio Installed?

Get your workshop documents:

<https://github.com/iekanat/rworkshop>

What is this about?

A brief introduction to R.

- R Console
- Importing Data
- Packages
- Sample analyses
- Basic visualization
- Where to get help?



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 step to start analysis.

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 step 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 memory¹
- R is single core¹

¹Except 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

- 1 Introduction
- 2 Familiar Examples**
- 3 R Console
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

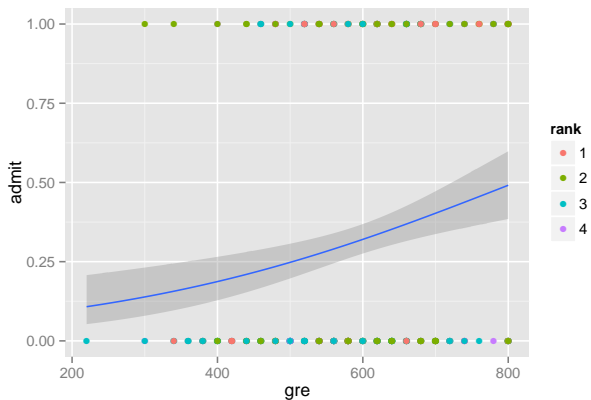
Logistic Regression

```
# 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 *
## rank3       -1.340204   0.345306  -3.881  0.000104 ***
## rank4       -1.551464   0.417832  -3.713  0.000205 ***
## ---
```

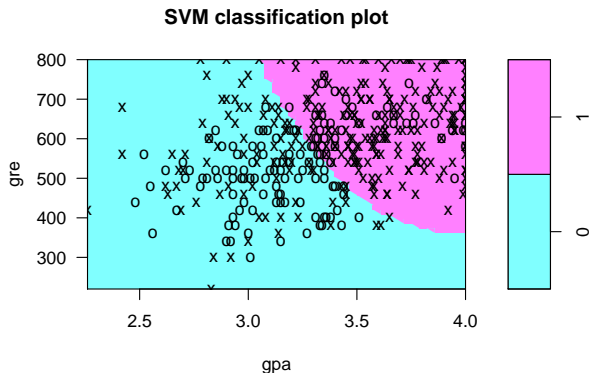
Logistic Regression

```
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

- 1 Introduction
- 2 Familiar Examples
- 3 R Console**
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

Command Driven Interface

Command line may be intimidating

Power over Convenience

Consider the number of

- Functions
- Parameters
- Data sources
- Variables
- Replications

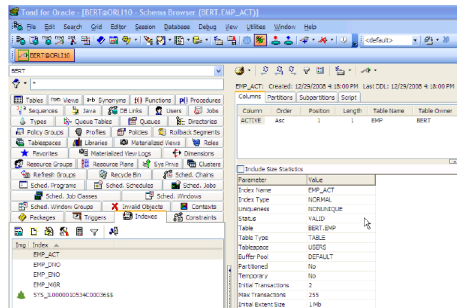
Command Driven Interface

Command line may be intimidating

Power over Convenience

Consider the number of

- Functions
- Parameters
- Data sources
- Variables
- Replications



R Studio

The screenshot displays the R Studio interface with four main panels:

- Script Editor (Top Left):** Contains an R script with comments and code. A red box highlights the script content, and a large red letter 'A' is overlaid on it.
- Environment (Top Right):** Shows the current environment with variables like 'admtData', 'csv', 'd', 'xLsx', and 'Values'. A blue box highlights this panel, and a large blue letter 'C' is overlaid on it.
- Console (Bottom Left):** Displays the output of the R script, including deviance values and the number of Fisher Scoring iterations. A green box highlights this panel, and a large green letter 'B' is overlaid on it.
- Plots (Bottom Right):** Shows an 'SVM classification plot' with 'gpa' on the x-axis and 'pre' on the y-axis. The plot displays two classes of data points (circles and crosses) separated by a decision boundary. A yellow box highlights a specific region in the plot, and a large yellow letter 'D' is overlaid on it.

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

```
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
```

Variables II

```
C <- c(1, 3, 7, 9)  # A list can be in a variable
```

```
C
```

```
## [1] 1 3 7 9
```

```
C + A
```

```
## [1] 3 5 9 11
```

```
C * A
```

```
## [1] 2 6 14 18
```

```
C < 5
```

```
## [1] TRUE TRUE FALSE FALSE
```


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 23  
## [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
```

Indexes and Data Frames IV

```
Countries[, "names"]
```

```
## [1] US TR DE  
## Levels: DE TR US
```

```
Countries$names
```

```
## [1] US TR DE  
## Levels: DE TR US
```

```
Countries$those
```

```
## [1] TRUE FALSE 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"          "lmvreg_0"   "logit_0"
## [11] "logit_1"    "logit_2"    "mtcars"     "mvreg_0"    "mvreg_0_fit"
## [16] "mvreg_0_res" "pima2"      "Pima.tr"    "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 = FALSE,  
##   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",  
##         sQuote(name)), domain = NA)  
##       pos <- name
```

Commonly Used Functions I

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

```
## [1] "A"          "admitData"  "B"          "c"          "C"
## [6] "Countries"  "HelloWorld" "i"          "lmvreg_0"   "logit_0"
## [11] "logit_1"    "logit_2"    "mtcars"     "mvreg_0"    "mvreg_0_fit"
## [16] "mvreg_0_res" "pima2"      "Pima.tr"    "svm_0"
```

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

```
## [1] "lmvreg_0"    "logit_0"    "mvreg_0"    "mvreg_0_fit" "mvreg_0_res"
## [6] "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

```
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 V

```
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
```

Questions



Outline

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data**
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

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 = "127.0.0.1",
  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

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data
- 5 Packages**
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

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.

Most of these can be achieved through the R Studio GUI. (Area D, packages pane)

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] "MASS"      "e1071"      "ggplot2"    "knitr"      "stats"
## [6] "graphics"  "grDevices"  "utils"      "datasets"   "methods"
## [11] "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"
## [13] "colorspace"  "curl"        "data.table"  "DBI"
```

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: Stats package, `lm()` (loaded by default)

Generalized Linear Models: Stats package, `glm()`

Traditional Econometric Models: `plm` package

Mixed Modeling: `nlme` and `lme4` packages


Commonly Used Packages: Machine Learning

Classifiers: caret², and class packages

Support Vector Machines: kernlab, e1071 packages

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

Neural Networks: neuralnet package.

²Caret package is a wrapper for a host of classification and regression model training functions. It eases visualizations, data manipulation, and analytics among others. 

Questions



Outline

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations**
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?

Motor Trends Dataset

We will use 1974 Motor Trend dataset. It has 32 observations and 11 variables.

- mpg: Miles per gallon
- cyl: Number of cylinders
- disp: Displacement
- hp: Horse Power
- drat: Rear axle ratio
- wt: Weight
- qsec: quarter mile time
- vs: V - S
- am: 0 automatic, 1 manual
- gear: Gears
- carb: Number of carburetors

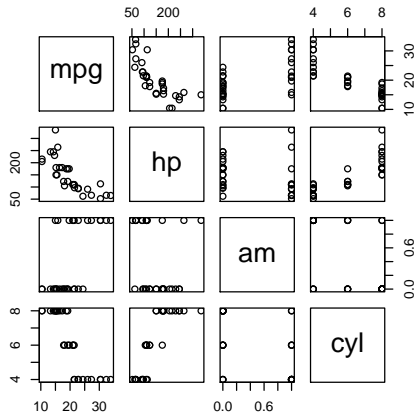
Motor Trends Dataset I

```
summary(mtcars)
```

```
##           mpg           cyl           disp           hp
##  Min.    :10.40   Min.    :4.000   Min.    : 71.1   Min.    : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##
##      drat           wt           qsec           vs
##  Min.    :2.760   Min.    :1.513   Min.    :14.50   Min.    :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##
##      am           gear           carb
##  Min.    :0.0000   Min.    :3.000   Min.    :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Motor Trends Dataset II

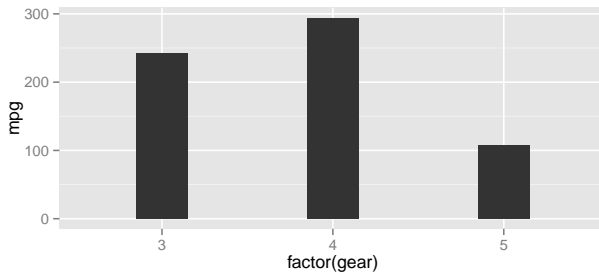
```
pairs(mtcars[, c("mpg", "hp", "am", "cyl")]) # Visualize Correlations
```



Bar Charts I

```
# Load the Dataset
data(mtcars)
## ggplot2 package is already loaded, else library(ggplot2)

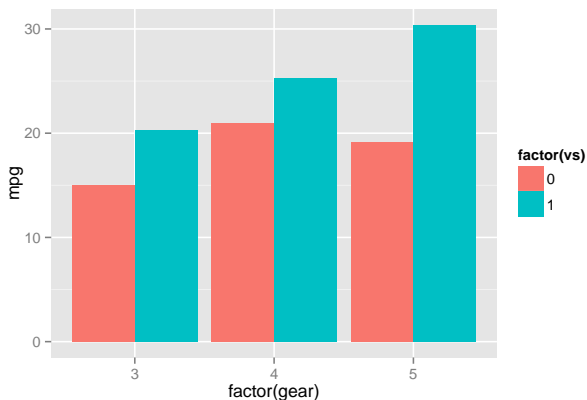
## Simple plot of just the means Initialize the plot with variables of
## interest
ggplot(mtcars, aes(x = factor(gear), y = mpg)) + # Instruct ggplot to plot bars of
geom_bar(stat = "identity", width = 0.3)
```



Bar Charts II

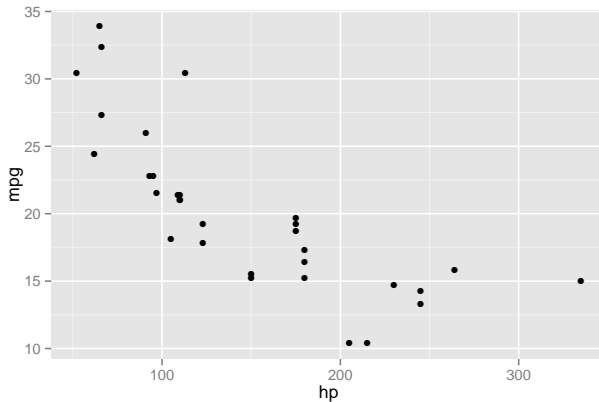
If we are interested in a third categorical variable vs:

```
ggplot(mtcars, aes(x = factor(gear), y = mpg, fill = factor(vs)), color = factor(vs))  
  stat_summary(fun.y = mean, position = position_dodge(), geom = "bar")
```



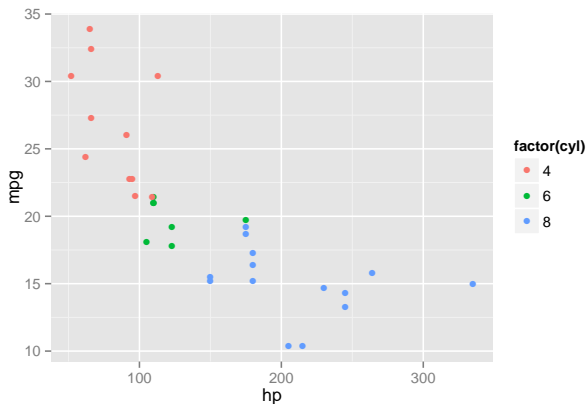
Scatter Plots I

```
# Initialize Plot  
ggplot(mtcars, aes(x = hp, y = mpg)) + # Instruct points to be plotted  
geom_point()
```



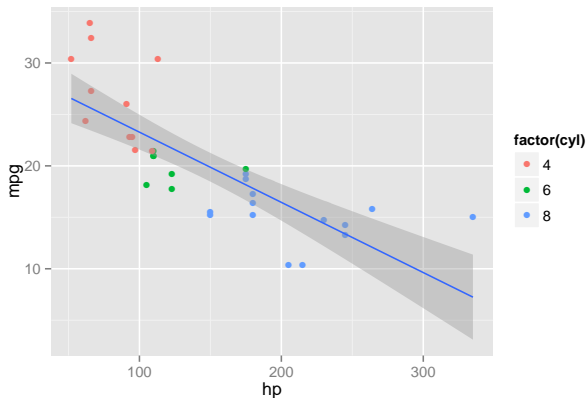
Scatter Plots II

```
ggplot(mtcars, aes(x = hp, y = mpg)) + # Put some color in those points  
geom_point(aes(color = factor(cyl)))
```



Scatter Plots III

```
ggplot(mtcars, aes(x = hp, y = mpg)) + geom_point(aes(color = factor(cyl))) +  
  # Add a regression line  
  geom_smooth(method = lm)
```



Multivariate Regression

We will keep using motor trends data set.

Pay close attention to how we specify the model.

Formula

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

R Model

$$Y \sim x_1 + x_2$$

This basic structure will remain constant across many R packages.

MV Regression I

```
# Let us estimate body weight index
mvreg_0 <- lm(mpg ~ hp + cyl + am, mtcars)
summary(mvreg_0)

##
## Call:
## lm(formula = mpg ~ hp + cyl + am, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.864 -1.811 -0.158  1.492  6.013
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.88834    2.78422   11.094 9.27e-12 ***
## hp          -0.03688    0.01452   -2.540  0.01693 *
## cyl         -1.12721    0.63417   -1.777  0.08636 .
## am           3.90428    1.29659    3.011  0.00546 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.807 on 28 degrees of freedom
## Multiple R-squared:  0.8041, Adjusted R-squared:  0.7831
## F-statistic: 38.32 on 3 and 28 DF,  p-value: 4.791e-10
```

MV Regression II

```
# Access Fitted Values View first 3 predictions  
mvreg_0$fitted.values[1:3]
```

```
##      Mazda RX4 Mazda RX4 Wag      Datsun 710  
##      23.97302      23.97302      26.85433
```

```
# Bonus: Are the residuals normally distributed  
shapiro.test(mvreg_0$residuals)
```

```
##  
##      Shapiro-Wilk normality test  
##  
## data:  mvreg_0$residuals  
## W = 0.98366, p-value = 0.8961
```

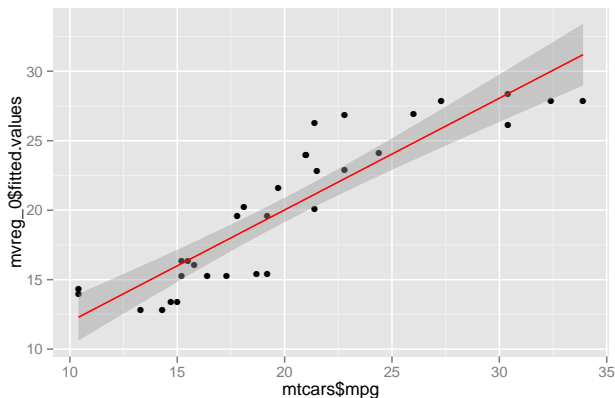
```
# AIC of the model  
AIC(mvreg_0)
```

```
## [1] 162.5849
```

MV Regression III

```
# Plot the fitted values against real values
```

```
ggplot(mtcars, aes(x = mtcars$mpg, y = mvreg_0$fitted.values)) + geom_point() +  
  stat_smooth(method = "lm", col = "red")
```



Logistic Regression

We will use a well established dataset on Diabetes among Pima Indians During Pregnancy.

Variables are:

- npreg: Number of times pregnant
- glu: Plasma glucose concentration
- bp: Diastolic blood pressure (mm Hg)
- skin: Triceps skin fold thickness (mm)
- bmi: Body mass index (weight in kg/(height in m)²)
- ped: Diabetes pedigree function
- age: Age (years)
- type: Diabetic (yes, no)

Logistic Regression

Dependent variable will be type (binary).

It is basically a regression with a binomial link function.

Formula

$$\log \left(\frac{p}{1-p} \right) = \beta_0 + \beta_1 x_1 + \epsilon$$

Logit I

```
library(MASS) # data file is part of MASS package
```

```
data(Pima.tr) # Load data file
```

```
??Pima.tr # Help on data file
```

```
head(Pima.tr, 3) # Top 3 rows
```

```
##      npreg glu bp skin  bmi    ped age type
## 1         5  86 68   28 30.2 0.364  24   No
## 2         7 195 70   33 25.1 0.163  55  Yes
## 3         5  77 82   41 35.8 0.156  35   No
```

Logit II

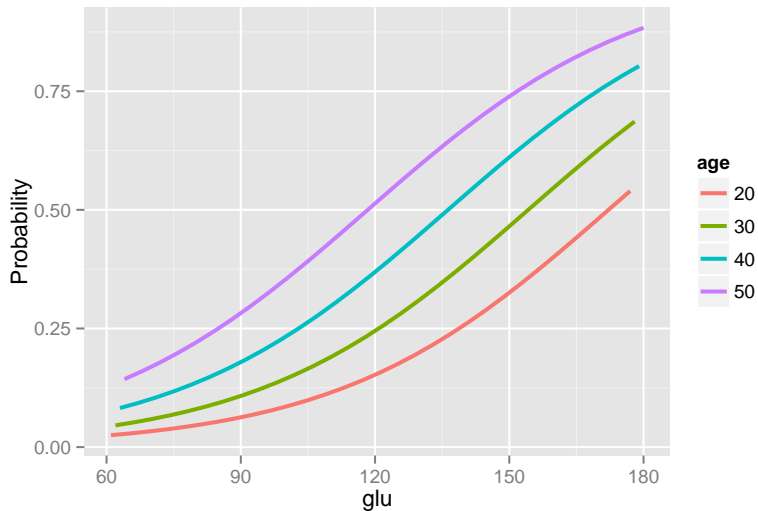
```
logit_2 <- glm(type ~ glu + ped + age, Pima.tr, family = "binomial")
summary(logit_2)

##
## Call:
## glm(formula = type ~ glu + ped + age, family = "binomial", data = Pima.tr)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.0755  -0.7169  -0.4236   0.7078   2.3526
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.706001    1.082177  -7.121 1.07e-12 ***
## glu          0.032903    0.006617   4.973 6.60e-07 ***
## ped          1.869117    0.626096   2.985 0.00283 **
## age          0.059003    0.017505   3.371 0.00075 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 187.10  on 196  degrees of freedom
## AIC: 195.1
##
## Number of Fisher Scoring iterations: 4
```

Logit III

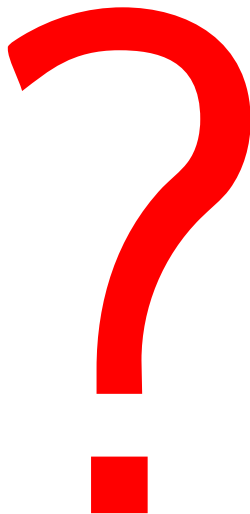
```
# Let us create a simulated dataset to plot out categorical age
pima2 <- data.frame(glu = 61:180, ped = mean(Pima.tr$ped), age = rep(c(20, 30,
  40, 50), 30))
# Get the predicted probabilities to plot
pima2$pred <- predict(logit_2, newdata = pima2, type = "response")
# Declare age as categorical
pima2$age <- as.factor(pima2$age)
## Plot the results, with categorical age as lines o. Initialize the plot
## with probability on y axis and glucose on x
ggplot(pima2, aes(x = glu, y = (pred))) + # Colored lines for each age category
geom_line(aes(colour = age), size = 1) + # Rename the y axis label
ylab("Probability")
```

Logit IV



Caret Package

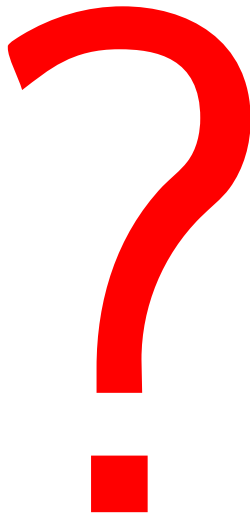
Questions



Outline

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting**
- 8 Where to Go Next?

Questions



Outline

- 1 Introduction
- 2 Familiar Examples
- 3 R Console
- 4 Importing Data
- 5 Packages
- 6 Sample Analysis and Visualizations
 - Descriptive Visualizations
 - Statistics
 - Machine Learning
- 7 Reporting
- 8 Where to Go Next?**