Making Sense of Data

Kadambari Devarajan

Post Graduate Program in Wildlife Biology and Conservation,
National Center for Biological Sciences,
Wildlife Conservation Society - India Program,
Bangalore.

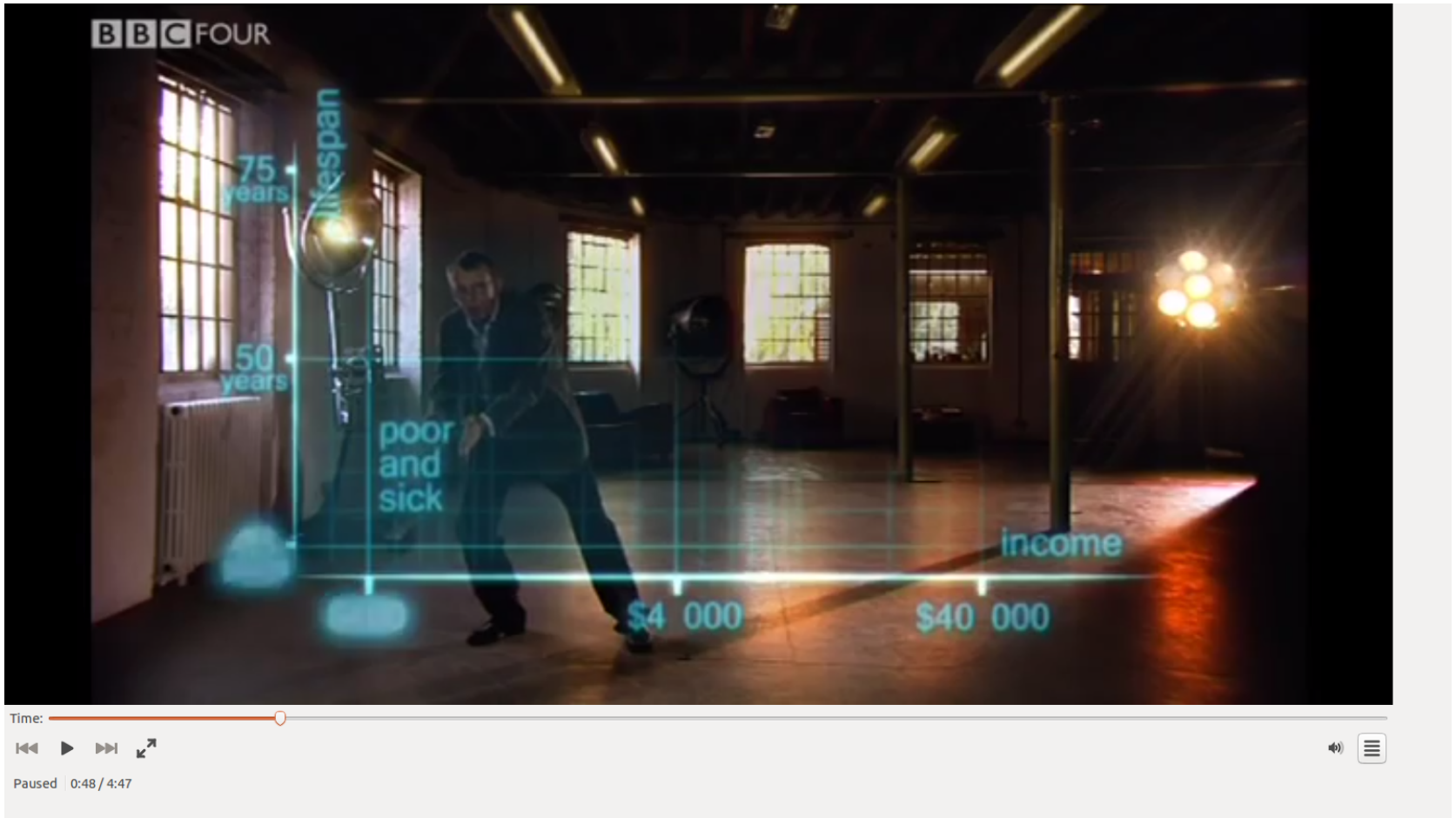
http://kadambarid.in
kadambari.devarajan@gmail.com

R Workshop for M.Phil and PhD Students at IIT Bombay
Elective: Planning and Development, Module: Statistics

Outline

- Data
- Statistics
- R + RStudio
- Getting to Know the UI
- Rolling with R: Basic Usage
- Plotting
- Exercises
- Advanced Capabilities

A World of Data



Hans Rosling's 200 Countries, 200 Years

A World of Data

Hans Rosling's River of Myths

Lies, Damned Lies, and Statistics

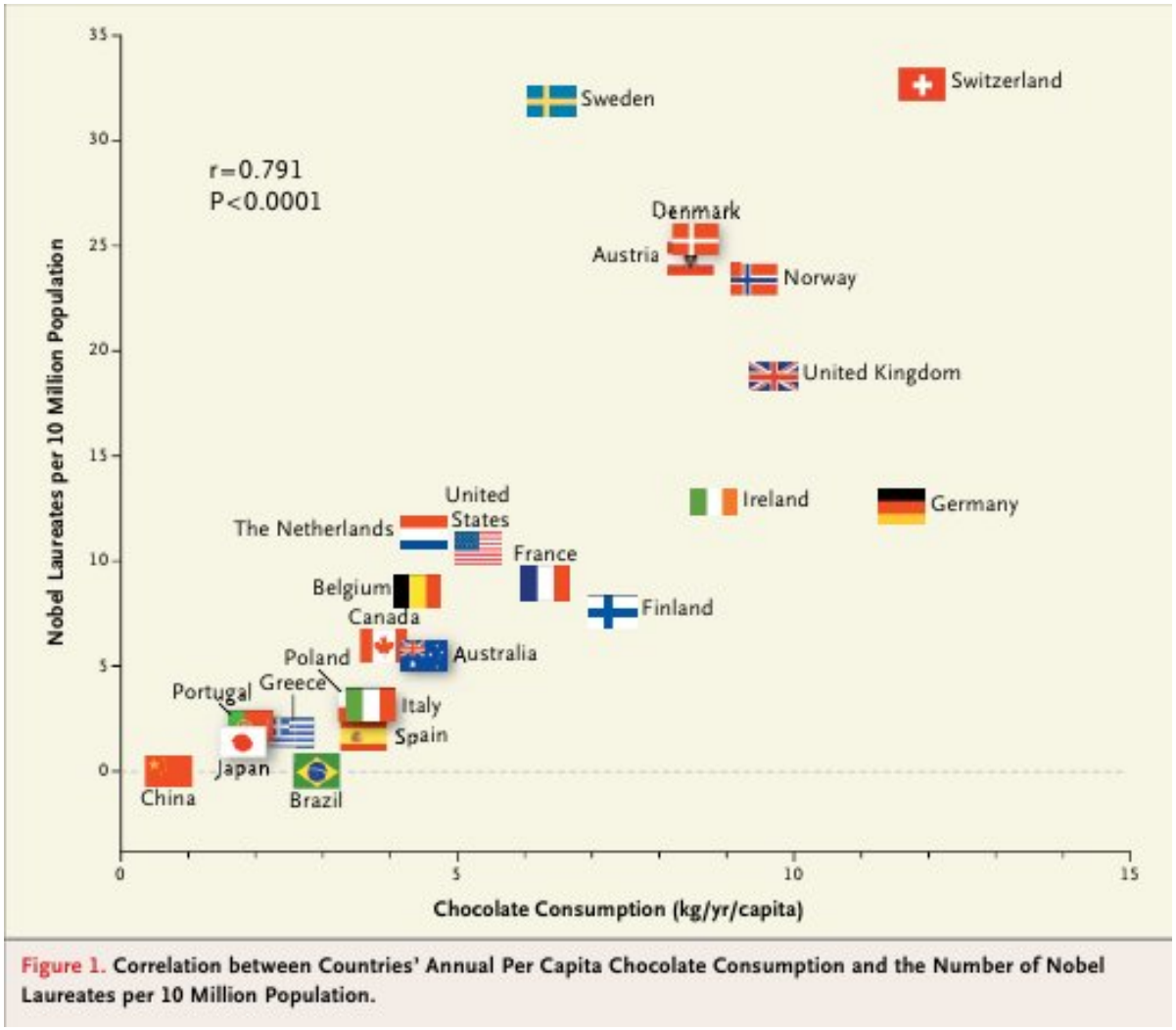


Figure 1. Correlation between Countries' Annual Per Capita Chocolate Consumption and the Number of Nobel Laureates per 10 Million Population.

Image Source: `http://www.nejm.org/doi/full/10.1056/NEJMon1211064`

Lies, Damned Lies, and Statistics

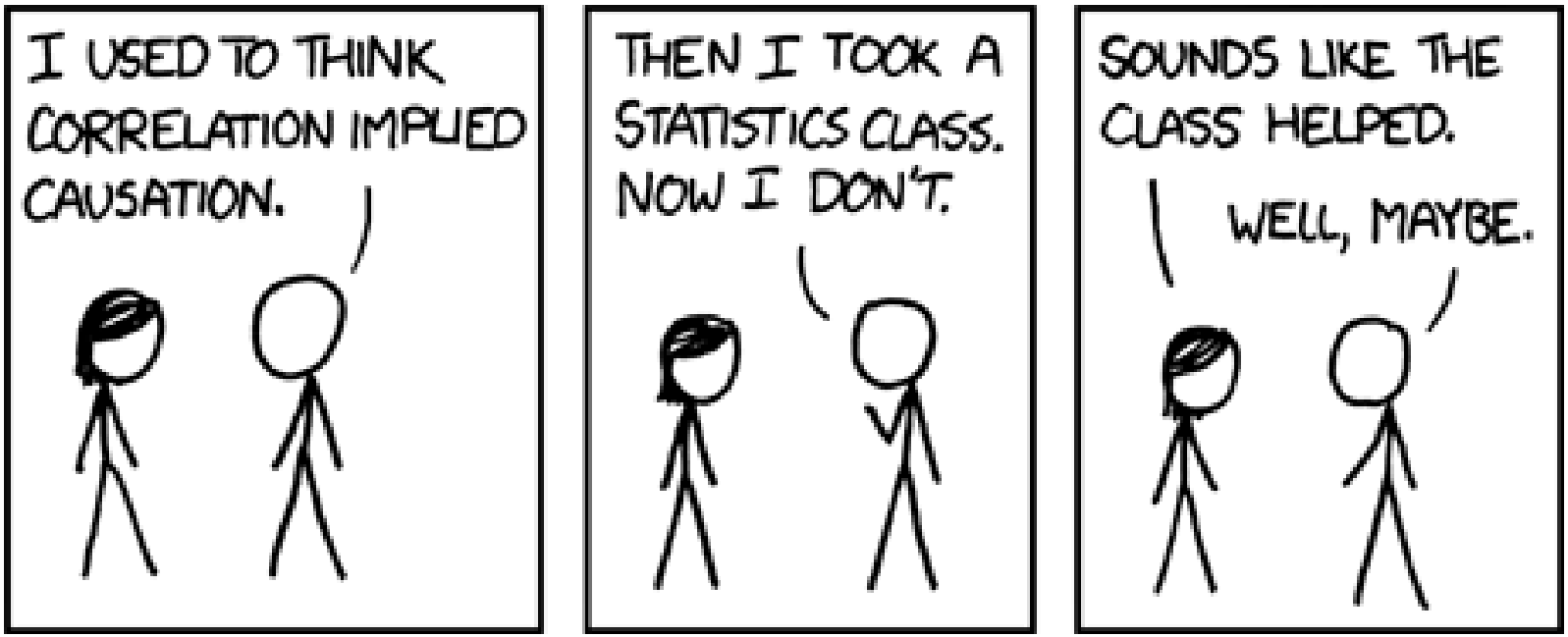


Image Source: `http://imgs.xkcd.com/comics/correlation.png`

20000 feet view of R

- Programming language - not just a statistics package!
- Object-oriented
  - data/information stored as objects
  - operations on objects
- Flexible and powerful

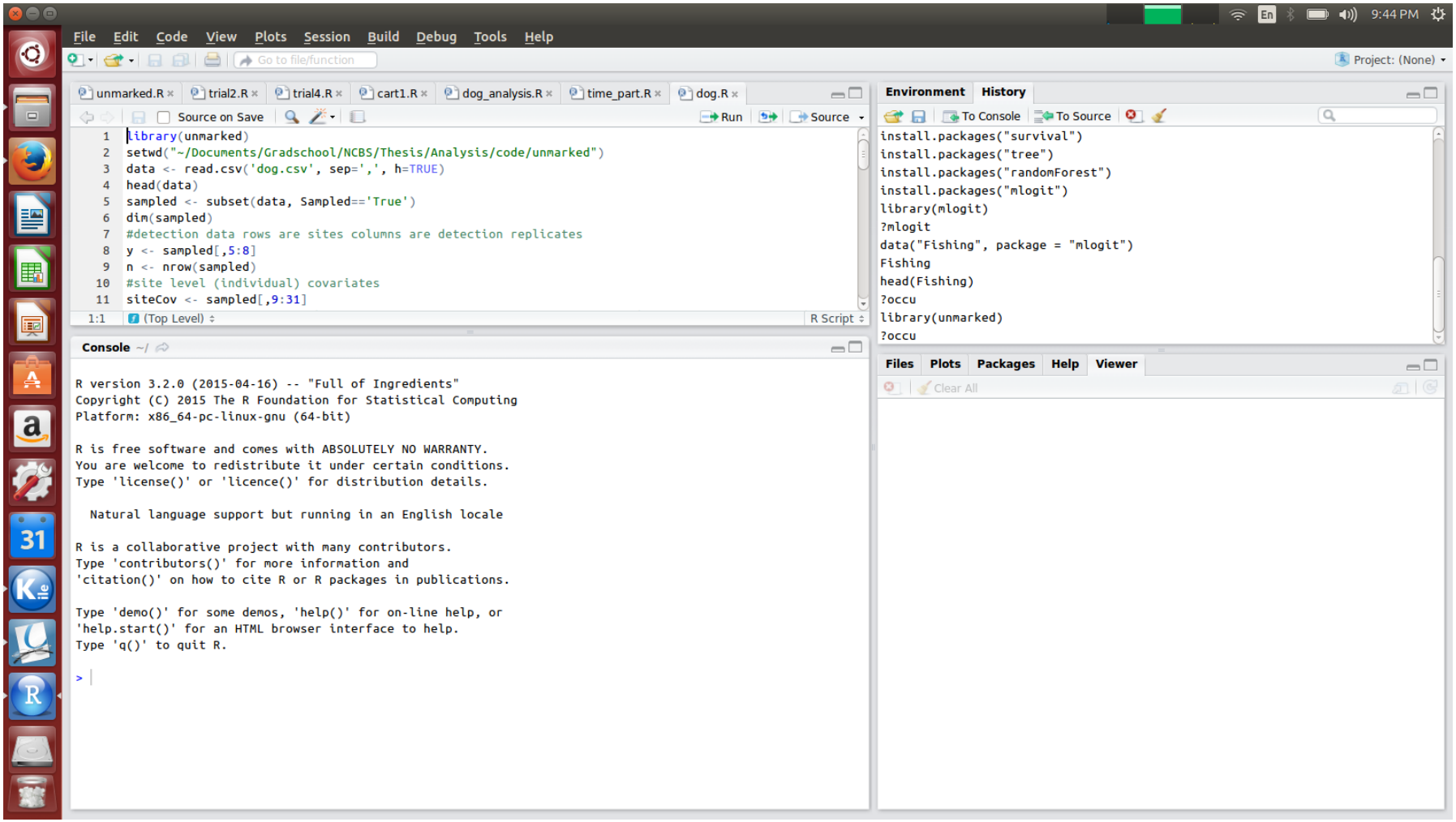
Why R Rocks

- One of the most powerful environments for statistics, currently
  - Interactive
  - Data structures
  - Functions as objects
  - Missing data
- Command-line == Clarity!
- Avoiding the dangers of button-clicking

Why R Rocks

- Safety with scripts
- Pretty pictures - graphics and visualization
- Free (as in “free beer” AND “freedom”)
  - Packages
  - Community

RStudio


File Edit Code View Plots Session Build Debug Tools Help
unmarked.R trial2.R trial4.R cart1.R dog_analysis.R time_part.R dog.R
Source on Save
Run Source
1 library(unmarked)
2 setwd("~/Documents/Gradschool/NCBS/Thesis/Analysis/code/unmarked")
3 data <- read.csv('dog.csv', sep=',', h=TRUE)
4 head(data)
5 sampled <- subset(data, Sampled=='True')
6 dim(sampled)
7 #detection data rows are sites columns are detection replicates
8 y <- sampled[,5:8]
9 n <- nrow(sampled)
10 #site level (individual) covariates
11 siteCov <- sampled[,9:31]
1:1 (Top Level) R Script
Console ~/
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment History
To Console To Source
install.packages("survival")
install.packages("tree")
install.packages("randomForest")
install.packages("mlogit")
library(mlogit)
?mlogit
data("Fishing", package = "mlogit")
Fishing
head(Fishing)
?occu
library(unmarked)
?occu
Files Plots Packages Help Viewer
Clear All

Getting to Know the UI

- Console
- Help
- File editing
- File browser
- Plots
- Menus

RStudio Basics

- Files - Open, Save
- Executing

If typing directly in the console, just pressing ‘Enter’ will suffice for the command to be executed. However, if typing in the source (recommended), ‘Ctrl-Enter’ will do the trick.

- Executing a block of commands
- Multiline commands and the ‘+’ symbol

Some Ground Rules

- Everyone must type along
- Any text following the command prompt (“>”) has to be typed into your Source or Console
- The output has not been given in the slides

Some Tips

- Navigation on console
  - Arrow keys
  - Tab completion
- Help
- Help using functions

```
> ?sin
> ?mean
> ??mean
```

Let's Get Rolling with R

Basic Arithmetic

```
> 23 + 79 # Evaluates expression
# and prints the result
```

- The '>' symbol is called the 'prompt'
- Anything following the '#' symbol is a 'comment'

Let's Get Rolling with R

Expressions

```
> 12/4 + 2 # Operator precedence
```

- 12/(4 + 2) is different from (12/4) + 2
- Use parantheses

R as a Calculator

Try these:

```
> 17 + 24
> 1.23456*42
> 47/6
> 4.567^54
> 2/4 + 1
> 2/(4 + 1)
```

R as a Scientific Calculator

Try these:

```
> sqrt(3)
> sin(pi/2)
> asin(0.5)
> asin(0.5)*180/pi
> log(2)
> log10(2)
```

Assignment

Assignment binds information to an object
"<-" is the assignment symbol
"=" can also be used

```
> x <- 5
# Assign 5 to the variable x
> y <- 4
> x
> x + y
> wt = 50
> val1 <- x - y
```

Assignment

```
> x + 4
> val1*30
> x^3
```

Objects

- Everything is an object
- Objects can be of different types
- Objects contain data
- We can perform operations on objects

Naming Objects

- Names must always start with a character (never a numeral)
- Names are case sensitive (wt is different from WT and wT)
- Names can be separated by an underscore (eg. female_wt) or a period (eg. female.wt)
  - Never use a space for separating compound names (eg. female wt is invalid)

Object Types

```
> wt <- 60.3
# Object type - Numeric
> x <- "hello"
# Object type - String
#(or Character)
> z <- TRUE
# Object type - Logical
#(TRUE or FALSE)
# Double precision float
```

Object Types

- Vectors
  - Simplest
  - Series of elements of a single data type
  - Similar to a column of values in a spreadsheet
- Matrices
- Data frames

Object Types: Vectors

Create using 'concatenate' - c(<comma separated list>)

```
> data <- c(1,4,3,2,1)
# c() stands for concatenate
# Values put into the same vector
> data*2
# Simultaneous operations - Useful
# functionality of vectors
# Operations on a vector are
# carried out one element at a time
> alphabet <- c("a","b","c","d")
# Vector of type 'character'
```

Object Types: Vectors

Exercise:

```
> x <- 5
> x <- x + 1
> a <- c("x", "y", "z")
> a <- c(1, 2, "c")
> x <- c(1, 2, 3, 4)
> a[3]
> x[2]
> x[c(1, 3, 4)]
```

Find the type of an object:

```
> typeof(x)
```

Accessing Elements of a Vector

```
> b <- c("a", "b", "c", "d")
# Vector b of type 'character'
> b
> b[1]
# Value of the first element of b
> b[c(2,4)]
# Value of 2nd and 4th
# elements of b
> d <- b[-1]
# Assign all of vector b
# except the first element to d
```

Relational Operations

Operators - <, <=, >, >=, ==, !=
Try these:

```
> x <- 2
> x > 4
> x < 5
> a <- c(1, 2, 3, 4)
> a != 3
```

Logical Vectors

```
> a <- c(1,3,4,5)
> a[a<3]
```

This is the same as:

```
> a[c(TRUE, FALSE, FALSE, FALSE)]
```

Now try:

```
> which(a<3)
```

Exercise

- Create a vector ‘vec’ containing the values 10 through 60 in increments of 10
- Display the elements of ‘vec’
- Increase every element of the vector by 5 and assign these values to a new vector ‘vec1’

Exercise

- Display the elements of ‘vec1’
  - In how many ways can the 3rd and 5th elements of ‘vec1’ be displayed?
  - Display the values of the 3rd and 5th (of ‘vec’) using the methods discussed
- Display all elements of ‘vec1’ that are less than 35
  - less than and equal to 35

Logical Operations

Operators - !, &, |

```
> x <- c(1, 2, 3, 4, 5, 6)
> x > 3 & x < 6
```

- Exercise: Display elements of 'vec1' greater than 20 and less than and including 65

Do not confuse

- The relational operator $>$ and the command prompt $>$
- $<-$, =, and ==
  - Assign - Assignment Operator - $<-$ and =
  - Check/Verify - Relational Operator - ==

Do not confuse

- The different brackets used:
  - Parantheses - () - eg. functions
  - Square brackets - [] - eg. vector operations
  - Curly braces - {} - eg. expressions (*enclosing an expression that already uses parantheses*)
    Note - () and {} can be use interchangeably for most part

Object Types: Functions

- Functions have a name and a variable number of arguments
- Built-in functions
- User defined functions

```
> ?log
> log(x=100, base=10)
# The arguments x and base are
# passed to the function log()
> log(100,10)
# Arguments can be passed in right
# order without naming them
```

Object Types: Functions

Generating a sequence of numbers:

```
> seq(from=2, to=20, by=2)
# function to generate regular
# sequence of nos.
```

Generating random numbers:

```
> runif(n=10)
# Default - random numbers
# from 0 to 1
> runif(n=100, min=0, max=100)
```

Summarizing Data

```
> a <- runif(n=100)
> b <- a[a<0.5]
> length(b)
# Count of no. of elements in b
> sum(b)
# Sum of the elements in vector b
> mean(b)
> sd(b)
> median(b)
> summary(b)
```

Plotting Data

```
> b_seq <- 1:length(b)
> plot(x=b_seq, y=b)
```

Matrices

```
> x <- matrix(c(5,7,9,6,3,4),nrow=3)
> y <- matrix(c(5,7,9,6), ncol=2)
> dim(x)
> x[1,1]
> x[2,]
> x[,2]
> x[2]?
> x%*%y
> t(x)
> solve(y)
```

Readily Available Data

R comes bundled with ready datasets that one can play around with.

```
> data()
> trees
# Also try ?trees
> summary(trees)
> head(trees)
> tail(trees)
```

This is a dataframe!

Accessing Elements of a Dataframe

```
> trees[1:5,]
> trees[,2]

> names(trees)
> trees$Girth
> trees$Volume
```

Accessing Elements: 'attach' function

Attach to a dataset:

```
> attach(trees)
> mean(Height)
> mean(Girth)

> detach(trees)  # When finished
```

Accessing Elements: 'with' function

with can be conveniently used instead of **attach**

```
   > plot(iris$Petal.Length ~
+   iris$Species)
   > with(iris, plot(Petal.Length ~
+   Species)) # Same thing!
```

Notice, no need to **detach**

I recommend using 'with' instead of 'attach'!

Plotting Data

```
# Histogram
> hist(trees$Height)

# Boxplot
> attach(trees)
> boxplot(Height)

# Scatterplot
> plot(Height, Girth)
> detach(trees)
```

Exercise: Rewrite these to use ‘with’ instead of ‘attach’

Plotting Data

Multiple plots:

```
> attach(trees)

> par(mfrow=c(2,2))
> hist(Height); boxplot(Height)
> hist(Volume); boxplot(Volume)

> detach(trees)
```

Exercise: Rewrite these to use ‘with’ instead of ‘attach’

Plotting Data

Other plots:

```
> barplot(1:10)
> ?pie
```

Categorical Data

Explore the **iris** dataset (as was shown for **trees**).

```
> iris$Species
> pie(iris$Species)

# Using 'table'
> table(iris$Species)
> pie(table(iris$Species))
```

Plotting Data: Using Formulae

- Very convenient with categorical data
- Use ~ to create a formula

```
> boxplot(iris$Petal.Length ~
+ iris$Species)
> plot(iris$Petal.Length ~
+ iris$Species) # Same thing!

> plot(iris$Petal.Length ~
+ iris$Species, col=c("red", "blue",
+ "green"))
```

Subsetting

Subsets of vectors/data frames

```
> subset(iris,
+ iris$Species=="setosa")
> subset(iris, Species=="setosa")
# Also works!
> subset(iris, Species="setosa")
# Wrong!
> subset(iris, select=
+ c(Petal.Width, Petal.Length))
# Check docs for more options
```

Creating Your Own Dataframes

```
> x <- 1:20
> y <- x*x
> z <- y + 10
> df <- data.frame(x=x, y=y, z=z)
> df
> df <- data.frame(a=x, b=y, c=z)
# Changes name of column
> names(df)
```

Add/Remove Columns

```
> df$total <- df$x + df$y
# Adds a column called 'total'
> names(df)
# To remove this column:
> df$total <- NULL
> names(df)
```

Add/Remove Rows

```
> rbind(df, c(-1, -2, -3))
> df
# Also check the cbind function
```

Using ifelse

```
> x <- 1:10
> ifelse(x < 6, "blue", "green")
> ifelse(x < 4, "blue",
+ ifelse(x < 7, "green", "red"))
# Color values in scatterplot
> with(iris, plot(
+     Petal.Length, Petal.Width,
+     col=ifelse(
+         Species=="setosa", "red",
+     ifelse(
+             Species=="virginica",
+     "blue", "green"))))
```

Missing Values

- Indicated by NA
- Typically automatically handled
- Use **is.na** to find the NA values

```
> x <- 1:5; y <- x*x
> plot(x, y)
> x[3] <- NA
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> plot(x, y)
```

Type-along Exercise

```
> wt <- c(69, 73, 70, 69, 90, 48,48)
> mean(wt)
> summary(wt)
> plot(wt)
> hist(wt*20)
> hist(wt, breaks=4)
> hist(wt*20, breaks=7,
+   xlim=c(500,2500), col="blue")
# Vertical line on existing graph
> abline(v=930)
```

R Scripts

- Use RStudio to create the code
- Save it to filename.R
- Run it directly using menus/shortcut
- Run it on console using:

```
> source("file.R")
```

Class Exercise

- Collect height and weight data in class
- Create a spreadsheet
- Save as a .csv file

Class Exercise

Set working directory

- If R can't find relative files

```
> getwd()
> setwd()
> setwd("~/Documents/Gradschool
+ /Analysis/code/unmarked")
```

- Using the UI
  - Menu: Session $\Rightarrow$ Set Working Directory
  - Shortcut: Ctrl+Shift+H

Class Exercise

Reading CSV data

```
> read.csv("file.csv")
> read.csv("http://www.ats.ucla.
edu/stat/data/hsb2.csv")
> data <- read.csv('pop.csv',
sep=',', h=TRUE)
```

Explore the data, summarize and visualize.

Installing Packages

- Through command-line
- Through UI

Some social science packages: demography, survey, sampling

Plots Great and Small

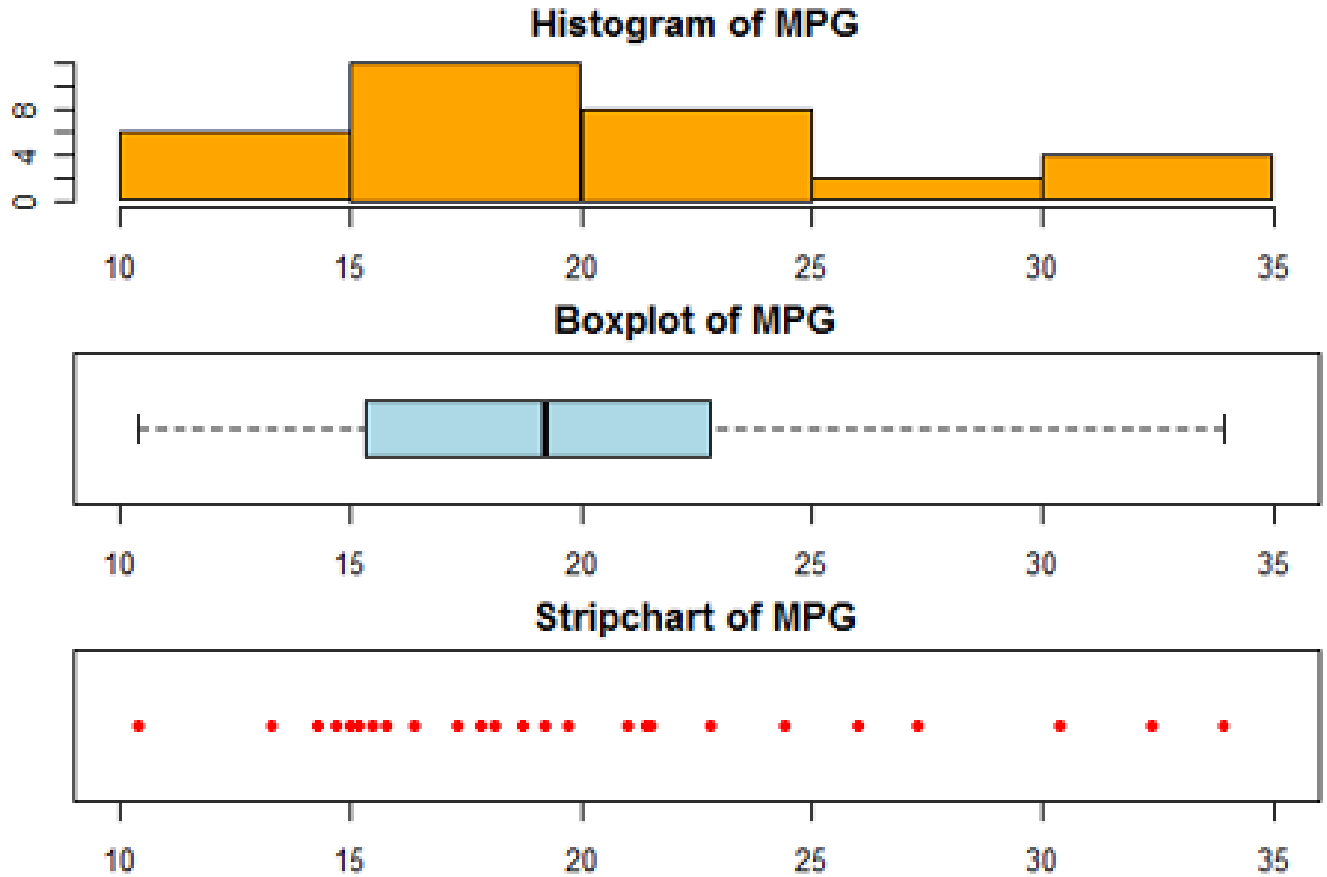


Image Source: `http://www.statmethods.net/graphs/`

Advanced Plots

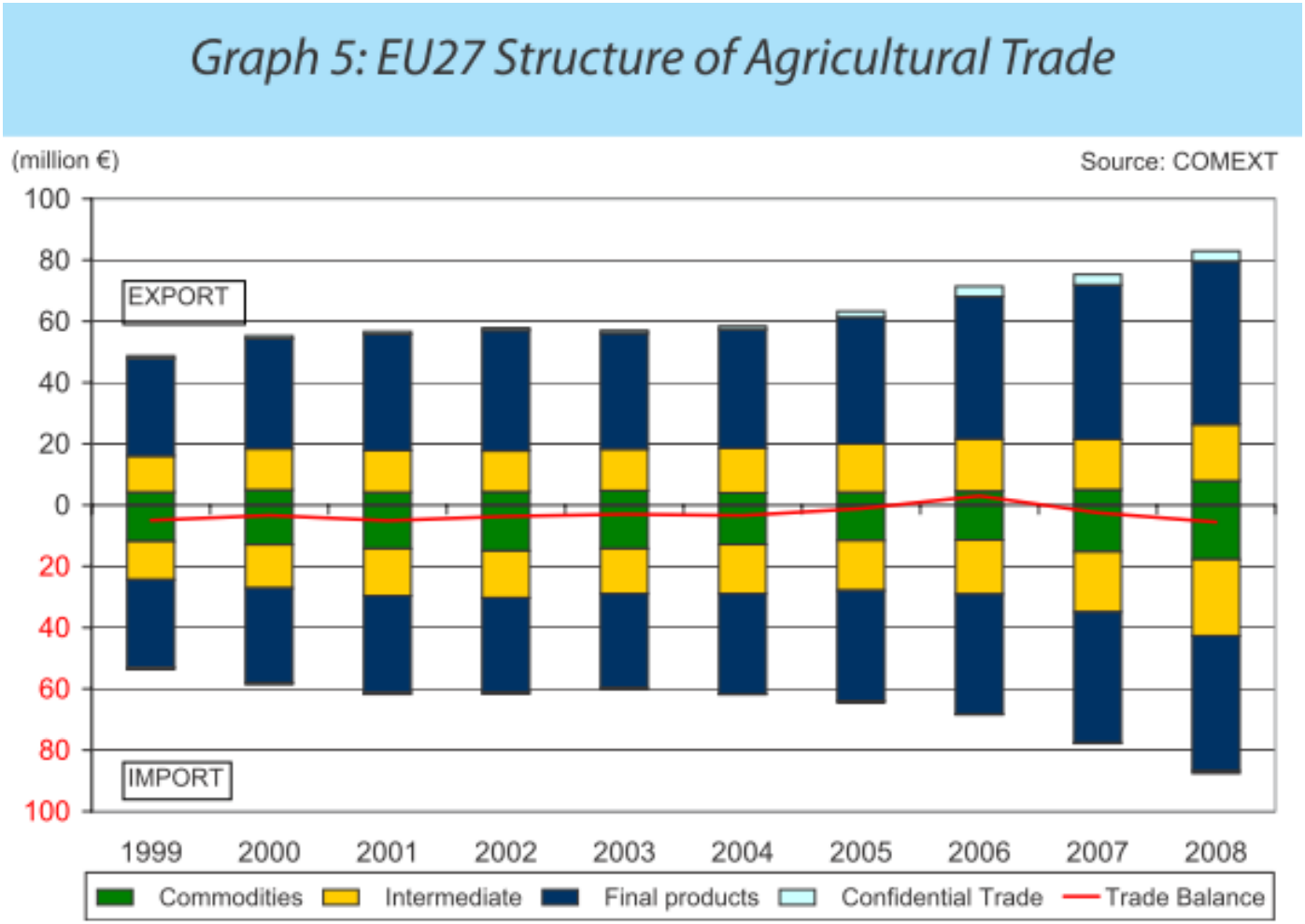
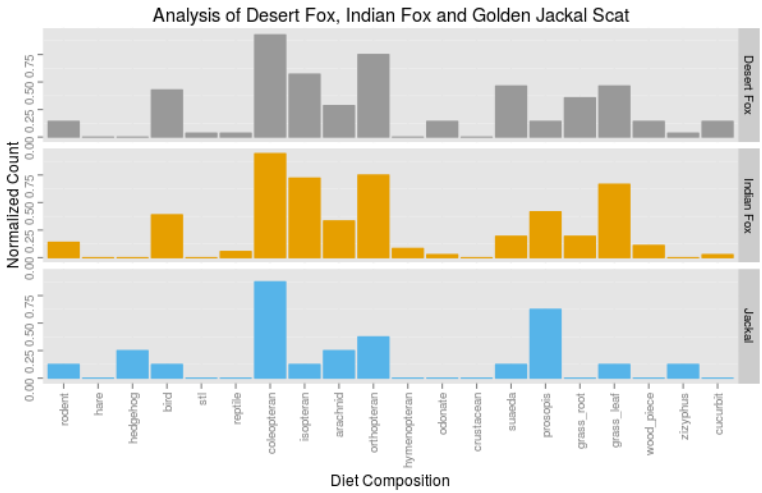
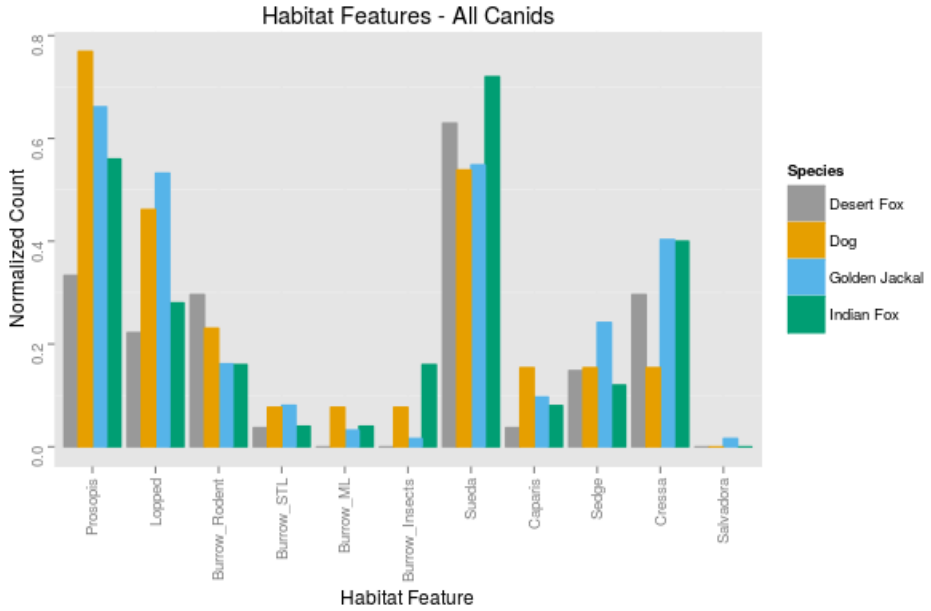


Image Source: `https://learnr.wordpress.com/`

Advanced Plots

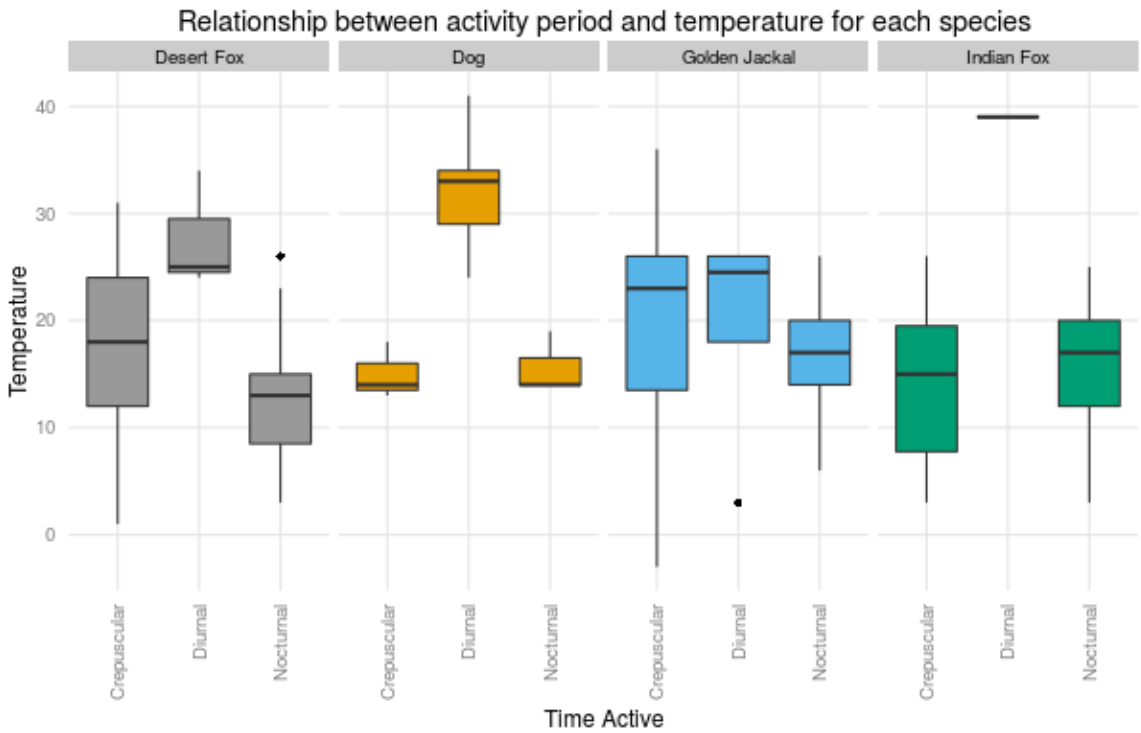
ggplot2 - Grammar of Graphics





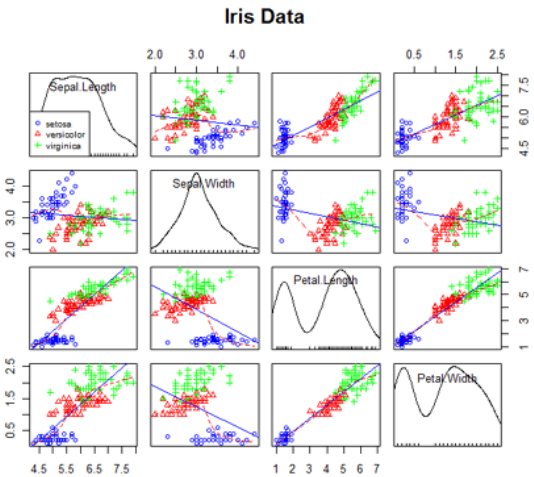
Advanced Plots

ggplot2 - Grammar of Graphics



Advanced Plots

- 3D plots
- Interactive graphs



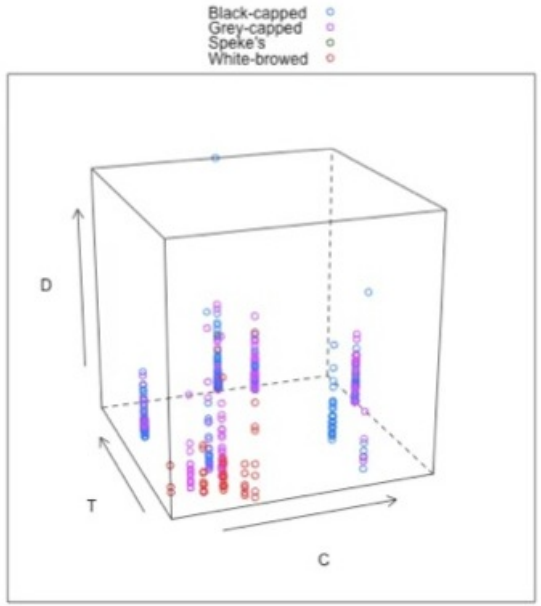
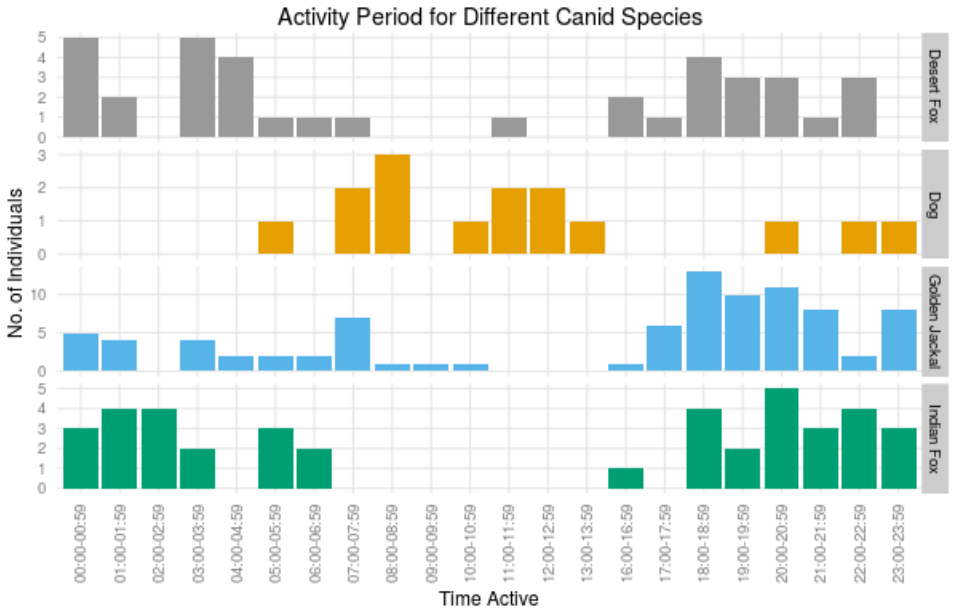
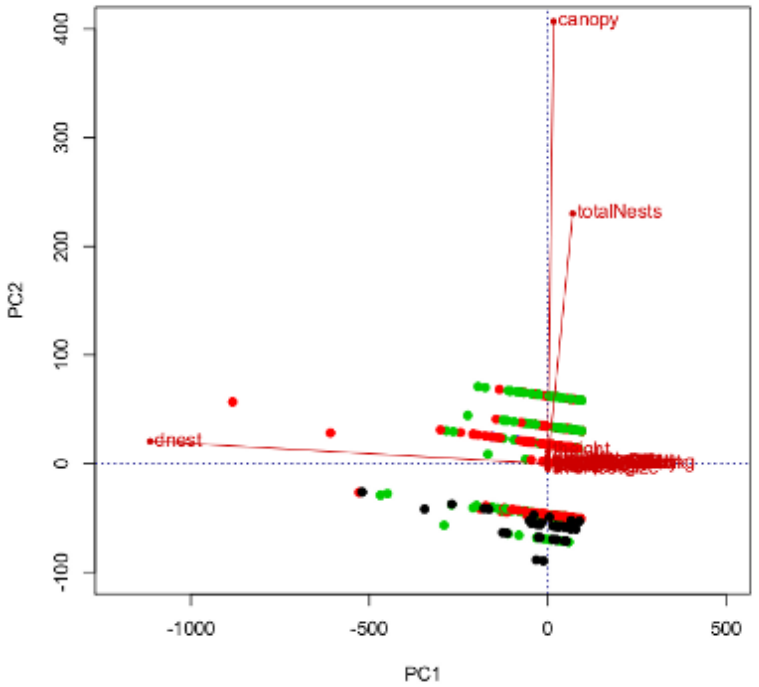


Image Source 1: `http://www.statmethods.net/advgraphs/index.html`
Image Source 3: `https://learnr.wordpress.com/2010/08/16/consultants-chart-in-ggplot2/`

Advanced Analysis





Acknowledgements

- Dr. Prabhu Ramachandran (Aerospace Engineering, IIT Bombay) for sharing his “Data Analysis and Interpretation” course lecture notes and allowing me to adapt it for the purposes of this presentation.
- Dr. Suhel Quader (NCBS and NCF) for his excellent “Basic Statistics” course at NCBS and letting me adapt elements of his course.
- Dr. Sarbani Banerjee (IIT Bombay) for her invitation to run this workshop.