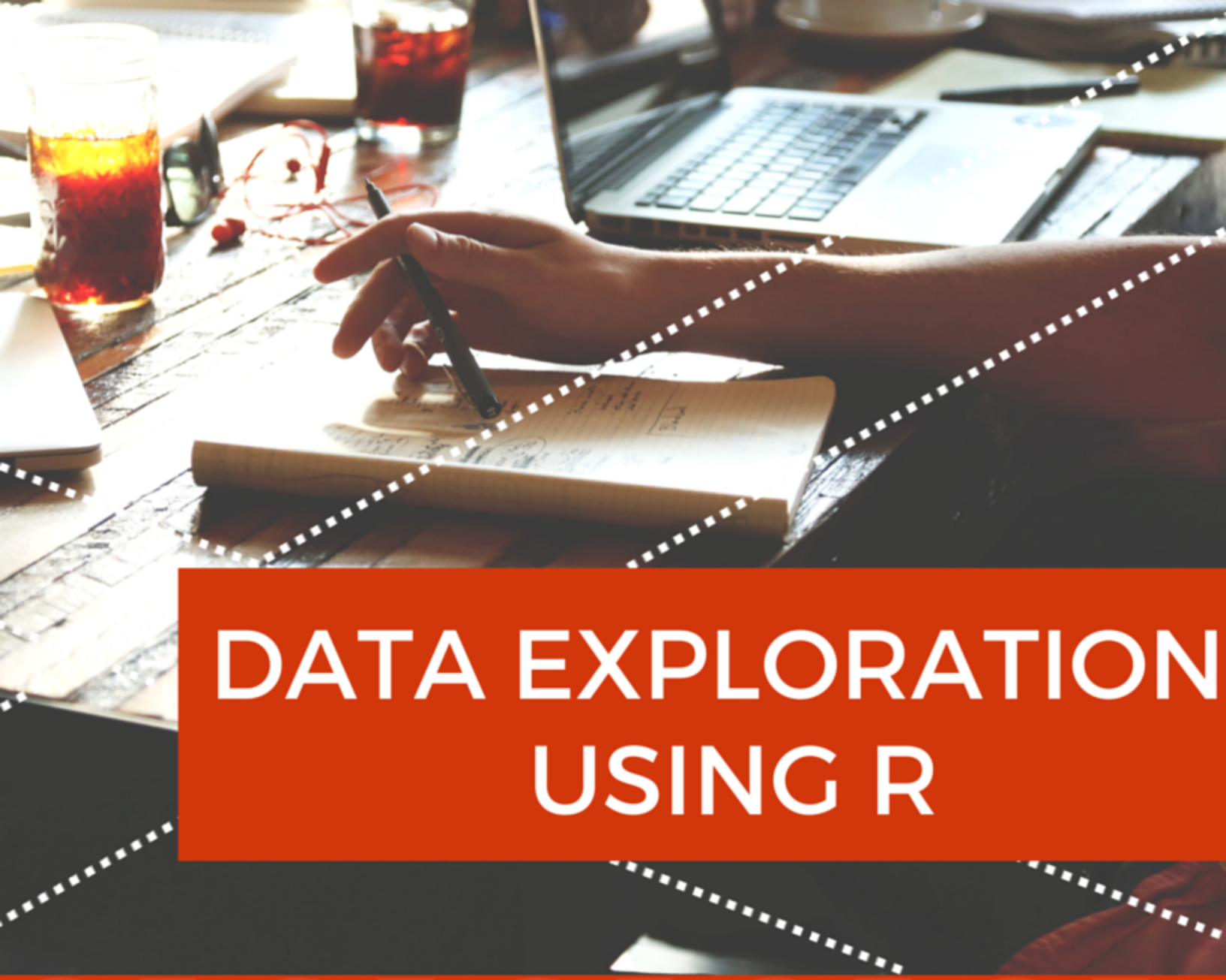

DATA EXPLORATION USING R

datasciencedojo
unleash the data scientist in you

DATA SCIENCE DOJO

2205 152ND AVE NE, REDMOND, WA 98052

The Library of Congress has catalogued this edition as follows:
Data Science Dojo LLC
Data Science and Data Engineering Handbook : 1.
Registration Number: TX0008030133
2015-05-22

Printed in the United States of America

Contents

1. Appendix B: Data Exploration Using R

1.1 Overview

1.2 Getting the Data

- Download the data set "Titanic_train.csv" at `https://github.com/datasciencedojo/bootcamp/tree/master/Datasets/Titanic_train.csv`
- Set working directory of R to the directory of the file using setwd()

```
# Reads in the Titanic dataset and sets it to a variable
# called 'Titanic'.
titanic = read.csv("C:\\myFolder\\titanic.csv");

colnames(titanic)
# "PassengerId" "Survived"  "Pclass"  "Name"
# "Sex"         "Age"       "SibSp"   "Parch"
# "Ticket"      "Fare"      "Cabin"   "Embarked"

# revise the name of a column
colnames(titanic)[5] <- "Gender"
```

1.3 Data Exploration

Look at the first few rows.

```
head(titanic[,1:5], 4)
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
##                                                 Name
   Gender
## 1                             Braund, Mr. Owen Harris
     male
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
   female
## 3                              Heikkinen, Miss. Laina
   female
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel)
   female
```

Let's look at a more specific sample of the data.

```
# View the last six columns of the data frame, and the
# first 4 rows in those columns.
head(titanic[,6:ncol(titanic)], 4)

#Age SibSp Parch           Ticket    Fare Cabin Embarked
#1  22     1     0        A/5 21171  7.2500              S
#2  38     1     0         PC 17599 71.2833   C85        C
#3  26     0     0 STON/O2. 3101282  7.9250              S
#4  35     1     0           113803 53.1000  C123        S
```

Summary of the data frame using str() function.

```
str(titanic)

## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr.
   Anthony",..: 109 191 35..
##  $ Gender     : Factor w/ 2 levels "female","male": 2 1
    1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels
   "110152","110413",..: 524 597 670 50..
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..:
    1 83 1 57 1 1 131..
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2
   4 4 4 3 4 4 ...
```

1.4 Categorical Preprocessing

There are some values that must be casted into categorical features so that our visualizations do not treat them as numerical values.

```
# Converts survived into a factor
titanic$Survived = factor(titanic$Survived)

# Converts embarked into a factor
titanic$Embarked = factor(titanic$Embarked)

# Summarizes the newly casted data frame.
str(titanic$Survived)
##  Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
str(titanic$Embarked)
##  Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2
    ...
```

To remove confusion, rename the categorical values themselves from numeric zeros and ones into their abstracted values.

```
levels(titanic$Survived) <- c("died", "survived")
levels(titanic$Embarked) <- c("Unknown", "Cherbourg", "
   Queenstown", "Southampton")
str(titanic$Survived)
##  Factor w/ 2 levels "died","survived": 1 2 2 2 1 1 1 1
   2 2 ...
str(titanic$Embarked)
##  Factor w/ 4 levels "Unknown","Cherbourg",..: 4 2 4 4 4
    3 4 4 4 2 ...
```

1.5 Data Exploration Through Visualization

1.5.1 Pie Charts

What is the rate of survival overall? Let's visualize the distribution of survival in a pie chart. (Figure: 1.1)

```
survivedTable = table(titanic$Survived); survivedTable
##
##     died survived
##      549      342
# mar, and oma mean margin and outer margin of the plot,
# cex gives the size of texts
par(mar = c(0, 0, 0, 0), oma = c(0, 0, 0, 0), cex=1.5)
pie(survivedTable,labels=c("Died","Survived"))
```

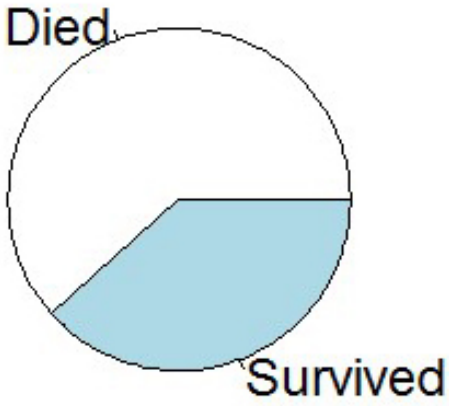

Figure 1.1

Was gender a good predictor? Let's visualize pie chart distributions on survival rate based upon gender. (Figure: 1.2)

```
par(mfrow = c(1, 2), mar = c(0, 0, 2, 0), oma = c(0, 1, 0,
    1), cex=1.5)
pie(table(titanic[titanic$Gender=="male", "Survived"]),
    labels=c("Dead","Survived"),  main="Survival Portion
       of Men")
pie(table(titanic[titanic$Gender=="female", "Survived"]),
    labels=c("Dead","Survived"), main="Survival Portion of
        Women")
```

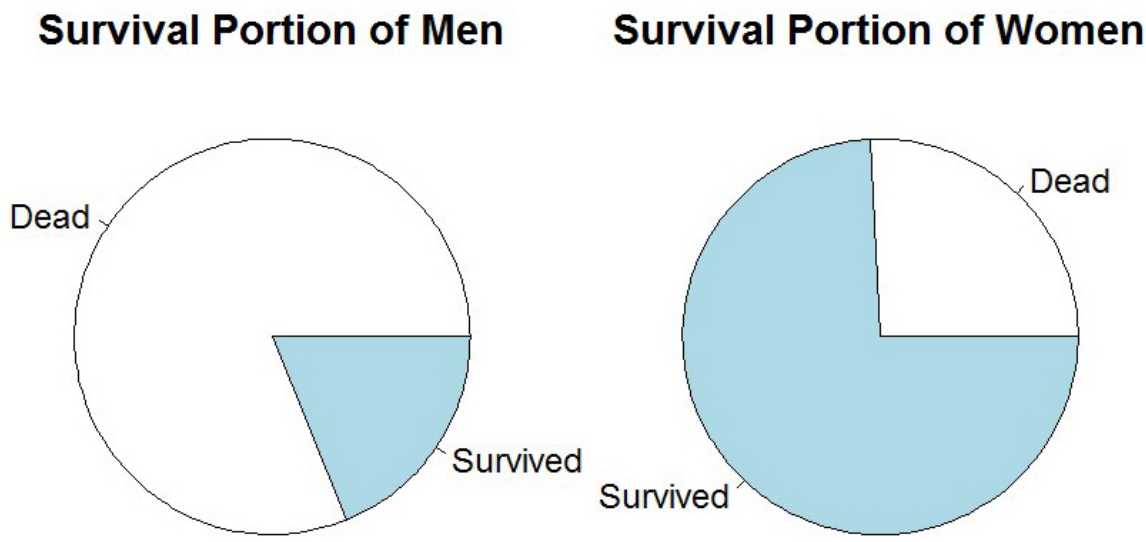

Figure 1.2

What is the distribution of age among titanic passengers?

```
# isolate the Age column
Age <- titanic$Age

#summary stats for the age coulmn
summary(Age)
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.42   20.12   28.00   29.70   38.00   80.00     177
```

What was the age statistics for those who survived versus those that died?

```
summary(titanic[titanic$Survived=="died", "Age"])
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.00   21.00   28.00   30.63   39.00   74.00     125

summary(titanic[titanic$Survived=="survived", "Age"])
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.42   19.00   28.00   28.34   36.00   80.00       52
```

1.5.2 Box Plot Distributions

When you want to compare the distribution or find patterns between two attributes where one is a numeric and the other is a categorical, a comparison box plot is a good option.

```
#Comparing age (numeric), with gender (categorical)
boxplot(
  titanic$Age~titanic$Sex, #comparison attributes
  data=titanic, #data source
  main="Age Distribution by Gender", #title
  xlab="Gender", #x-axis label
  ylab="Age", #y-axis label
  col=c("red", "green") #color
)
```

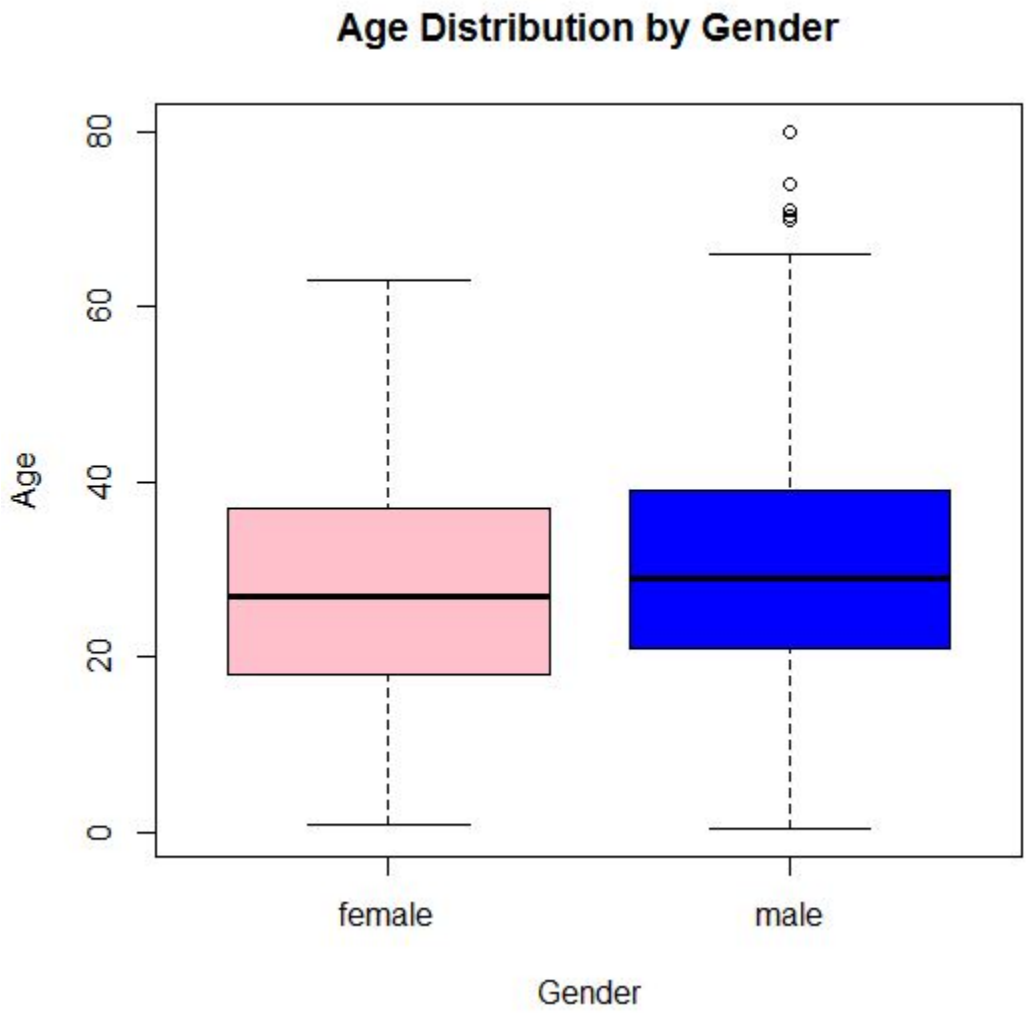

Figure 1.3: Comparison box plot between age and gender

```
# Comparing age (numeric) with survived (categorical)
boxplot(
  titanic$Age~titanic$Survived, #comparison attributes
  data=titanic, #data source
  main="Age Distribution by Survival", #title
  xlab="Survived", #x-axis label
  ylab="Age", #y-axis label
  col=c("blue", "red") #color
)
```

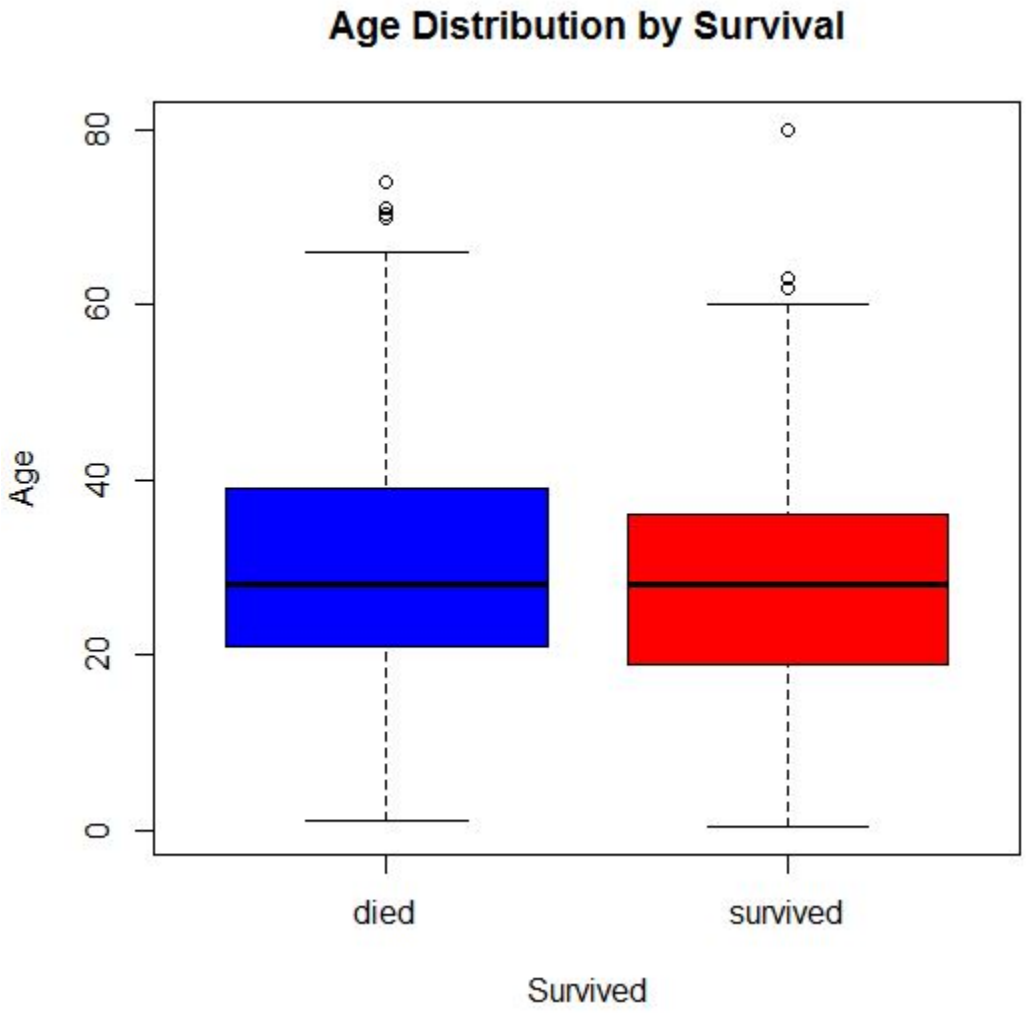

Figure 1.4: Comparison box plot between age and survived

1.5.3 Histogram

Let's visualize the distribution of age in a histogram. (Figure: 1.5)

```
hist(Age, col="blue", xlab="Age", ylab="Frequency",
     main = "Distribution of Passenger Ages on Titanic",
     cex.lab=1.6, cex.axis=1.4, cex.main=1.6)
```

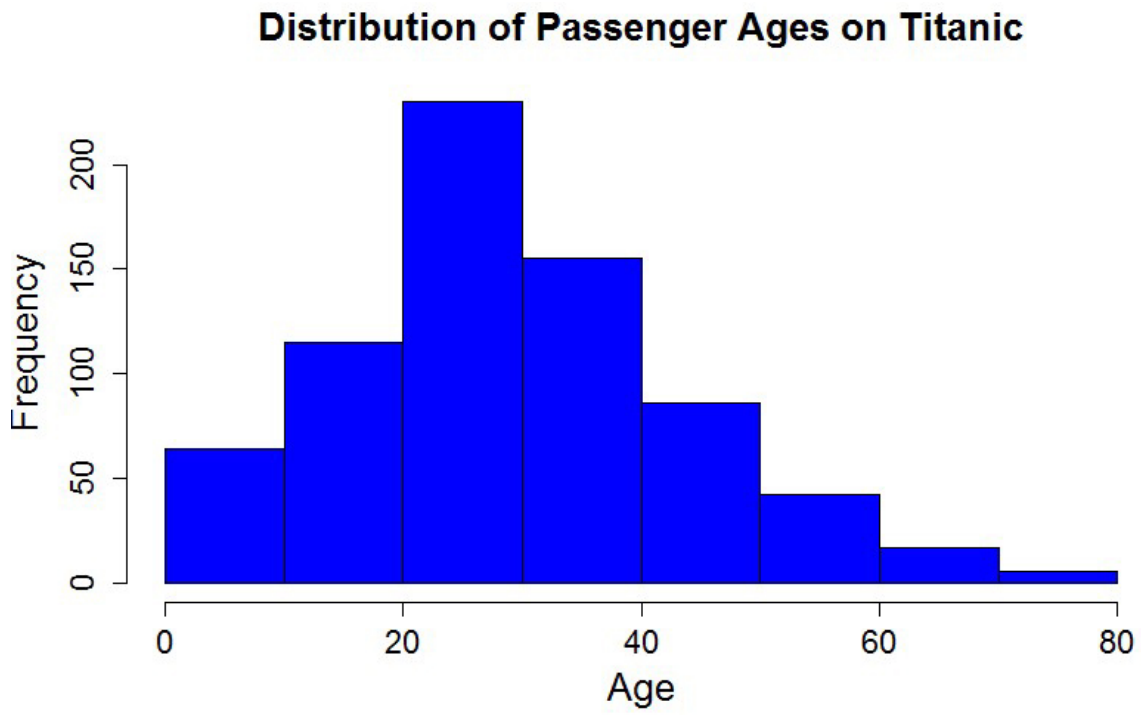

Figure 1.5

1.5.4 Kernel Density Plot

Let's smoothen the histogram into a kernel density plot. (Figure: 1.6)

```
# density() requires all NAs to be removed
d = density(na.omit(Age))
plot(d, main = "kernel density of Ages of Passengers",
     xlab="Age", cex.lab=1.6, cex.axis=1.4)
polygon(d, col="red", border="blue")
```

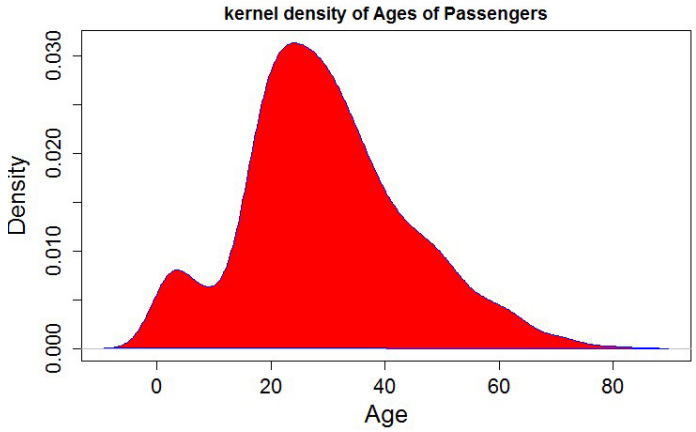

Figure 1.6

Comparison of density plots of Age with different Gender (Figure: 1.7)

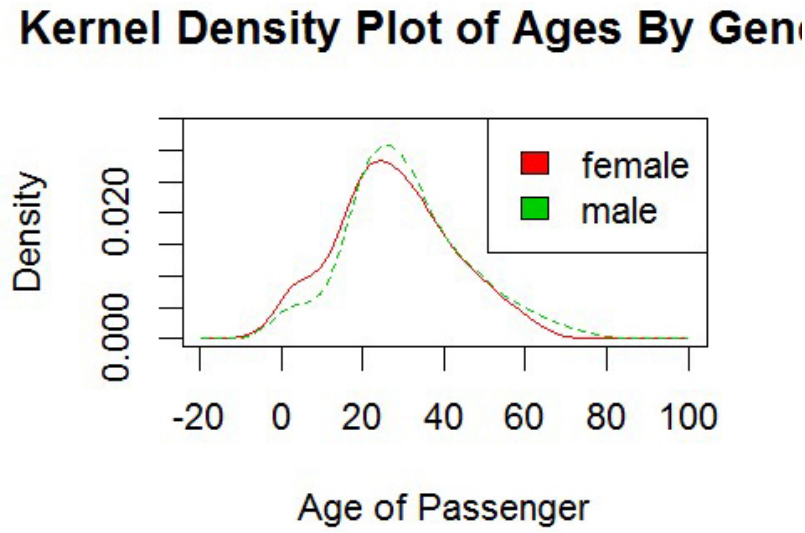

Figure 1.7

Did age have an impact on survival? (Figure: 1.8)

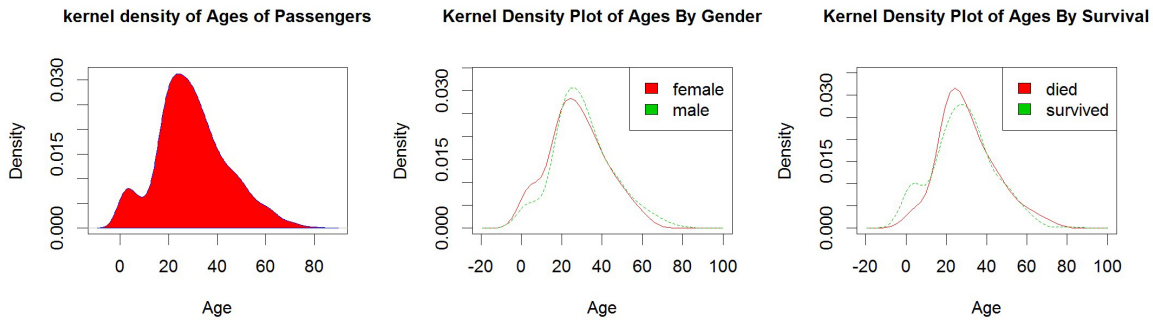

Figure 1.8

1.6 Feature Engineering

Let's add more features to our data. In this specific exercise we will derive from age, whether or not a person is a child or an adult. For the purpose of this example we will define a child as someone who is below 13 years of age.

```
## Multi dimensional comparison
Child <- titanic$Age # Isolating age.
## Now we need to create categories: NA = Unknown, 1 =
   Child, 2 = Adult
## Every age below 13 (exclusive) is classified into age
   group 1
Child[Child<13] <- 1
## Every child 13 or above is classified into age group 2
Child[Child>=13] <- 2
## Use labels instead of 0's and 1's
Child[Child==1] <- "Child"
Child[Child==2] <- "Adult"
```

Merging the new age column back into the dataset.

```
# Appends the new column to the titanic dataset
titanic_with_child_column <- cbind(titanic, Child)
# Removes rows where age is NA
titanic_with_child_column <- titanic_with_child_column[!is
   .na(titanic_with_child_column$Child),]
## Show part of the data frame with new feature: Child
head(titanic_with_child_column[,c(1:3, 13)], 4)
##   PassengerId Survived Pclass Child
## 1           1     died      3 Adult
## 2           2 survived      1 Adult
## 3           3 survived      3 Adult
## 4           4 survived      1 Adult
```

Now let's look at the survival rate of our newly created column. Did children have higher survival rates over adults? (Figure: 1.9)

```
par(mfrow = c(1, 2), mar = c(0, 1, 2, 1), oma = c(0, 1, 0, 1), cex=1.5)
pie(table(titanic_with_child_column[titanic_with_child_column$Child=="Child", "Survived"]),
    labels=c("Dead","Survived"), main="Survival Portion of Child")
pie(table(titanic_with_child_column[titanic_with_child_column$Child=="Adult", "Survived"]),
    labels=c("Dead","Survived"), main="Survival Portion of Adult")
```

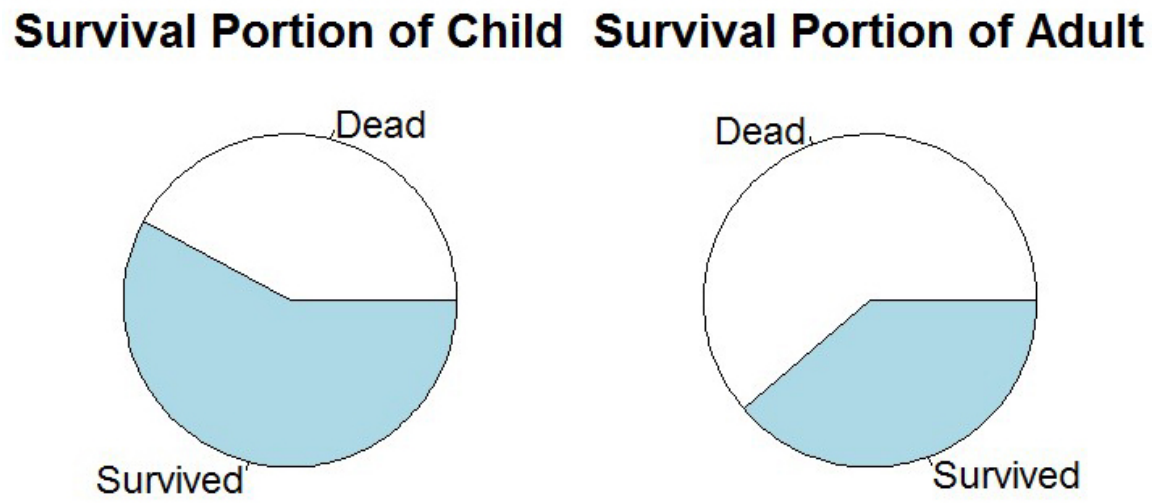

Figure 1.9

Child vs Gender (Figure: 1.10)

```
child.gender <- table(titanic_with_child_column$Gender,
   titanic_with_child_column$Child)
gender.survived <- table(titanic_with_child_column$
   Survived, titanic_with_child_column$Gender)
child.survived <- table(titanic_with_child_column$Survived
   , titanic_with_child_column$Child)
barplot(child.gender, main="Gender Proportion by Child or
   Adult",
        col=c("red","blue"), legend = rownames(child.
           gender))
barplot(gender.survived, main="Survived Proportion by
   Gender",
        col=c("red","blue"), legend = rownames(gender.
           survived))
barplot(child.survived, main="Survived Proportion by Child
    or Adult",
        col=c("red","blue"), legend = rownames(child.
           survived))
```

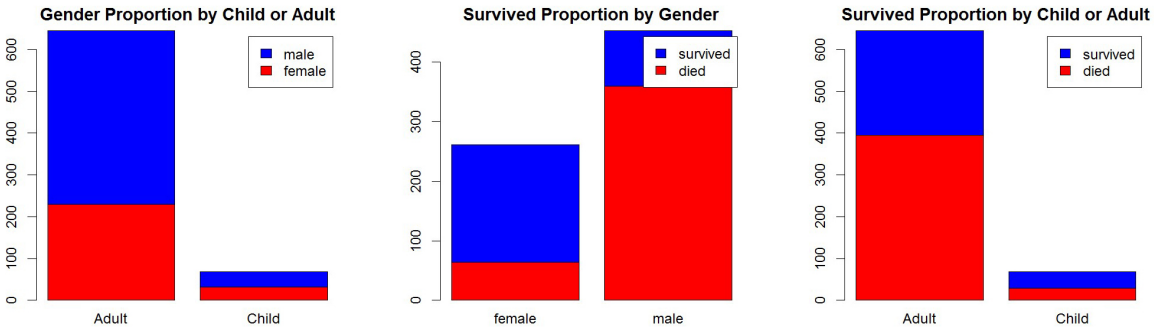

Figure 1.10

1.7 Advanced Visualizations in R

- **Base graphics:** constructed piecemeal. Conceptually simpler and allows plotting to mirror the thought process.
- **Lattice graphics:** entire plots created in a simple function call.
- **ggplot2 graphics:** an implementation of the Grammar of Graphics by Leland Wikinson. Combines concepts from both base and lattice graphics. (Need to install ggplot2 library)

A list of interactive visualization in R can be found at:
`http://ouzor.github.io/blog/2014/11/21/interactive-visualizations.html`

Base plotting system with airquality date. Note that air quality data ships natively with R. (Figure: 1.11)

```
par(cex=1.5) ## increase the size of texts
## scatter plot
plot(x = airquality$Temp, y = airquality$Ozone)
```

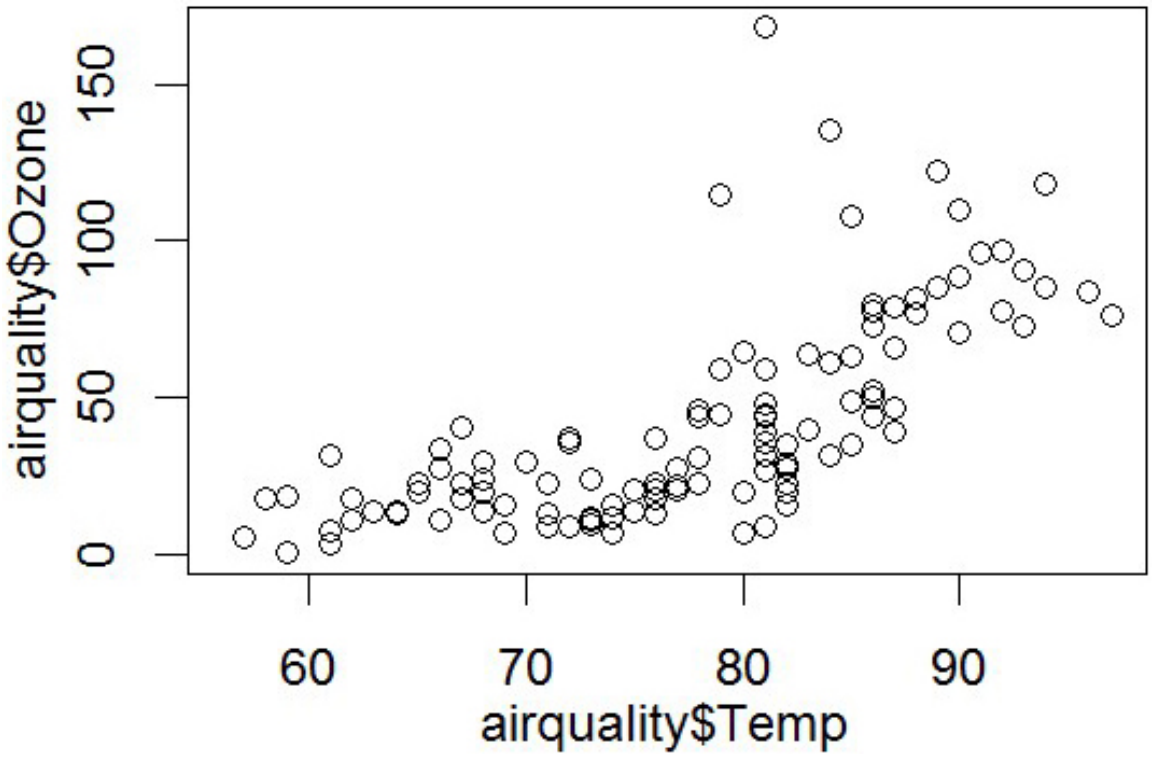

Figure 1.11

Base plotting system. (Figure: 1.12)

```
# par() function is used to specify global graphics
# parameters that affect all plots in an R session.
# Type ?par to see all parameters
par(mfrow=c(1, 2), mar=c(4, 4, 2, 1), oma=c(0, 0, 2, 0),
   cex=1.5)
# mfrow=c(1, 2): the figures will be drawn in an 1x2
# array by row. mar/oma: A numerical vector of the
# form (bottom, left, top, right), which gives the
# number of lines of margin to be specified on the
# four sides of the plot for mar, and size of the outer
# margins in lines of text for oma.
with(airquality,
    {
      plot(Wind, Ozone, main="Ozone and Wind")
      plot(Temp, Ozone, main="Ozone and Temperature")
      mtext(
        "Ozone and Weather in New York City",
        outer=TRUE
      )
    }
)
```

For the difference and relation of mar and oma, check out:
`http://research.stowers-institute.org/mcm/efg/R/Graphics/Basics/mar-oma/index.htm`

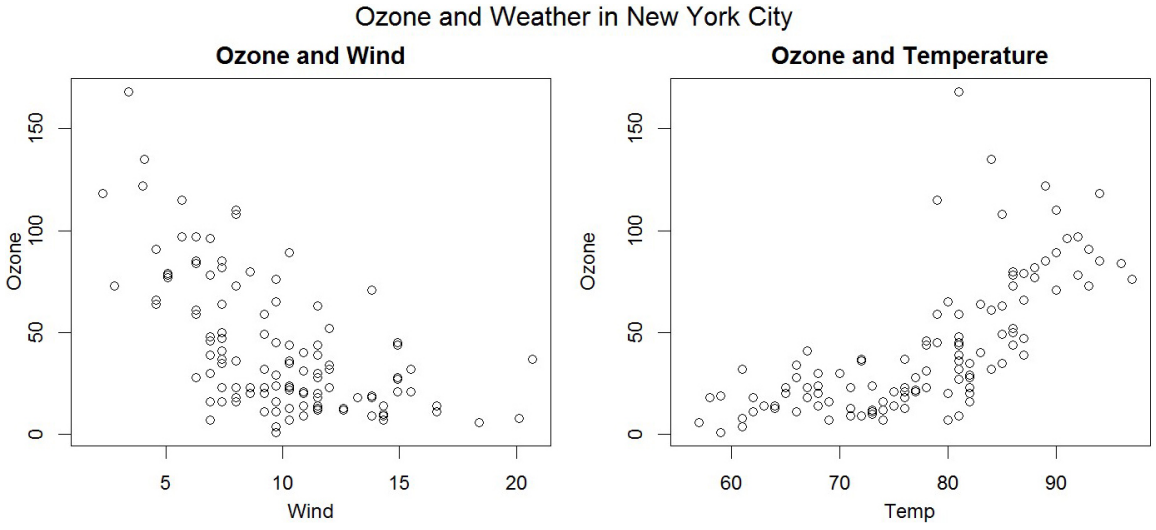

Figure 1.12

Plotting functions (high level) **PHASE ONE: Mount a canvas panel on the easel, and draw the draft (Initialize a plot).**

- **boxplot()**: a boxplot show the distribution of a vector. It is very useful to example the distribution of different variables.
- **plot()**: one of the most frequently used plotting functions in R.
- **barplot()**: create a bar plot with vertical or horizontal bars.
- **hist()**: compute a histogram of the given data values.
- **pie()**: draw a pie chart.

Remember to use **?plot** or **str(plot)**, etc. to check the arguments when you want to make more personalized plots. A **tutorial** of base plotting system with more details: `http://bcb.dfci.harvard.edu/~aedin/courses/BiocDec2011/2.Plotting.pdf`

Plotting functions (low level) **PHASE TWO: Add more details on your canvas, and make an artwork (Add more on an existing plot)**

- **lines**: adds liens to a plot, given a vector of x values and corresponding vector of y values
- **points**: adds a point to the plot
- **text**: add text labels to a plot using specified x,y coordinates
- **title**: add annotations to x,y axis labels, title, subtitles, outer margin
- **mtext**: add arbitrary text to margins (inner or outer) of plot
- **axis**: specify axis ticks

Save your artwork R can generate graphics (of varying levels of quality) on almost any type of display or printing device. Like:

- **postscript()**: for printing on PostScript printers, or creating PostScript graphics files.
- **pdf()**: produces a PDF file, which can also be included into PDF files.
- **jpeg()**: produces a bitmap JPEG file, best used for image plots. **help(Devices)** for a list of them all. Simple example:

```
## png(filename = 'plot1.png', width = 480, height = 480,
   units = 'px')
## plot(x, y)
## dev.off()
```

Example: boxplot and histogram

```
## the layout
par(mfrow = c(2, 1), mar = c(2, 0, 2, 0), oma = c(0, 0, 0,
    0))
## histogram at the top
hist(airquality$Ozone, breaks=12, main = "Histogram of
   Ozone")
## box plot below for comparison
boxplot(airquality$Ozone, horizontal=TRUE, main = "Box
   plot of Ozone")
```

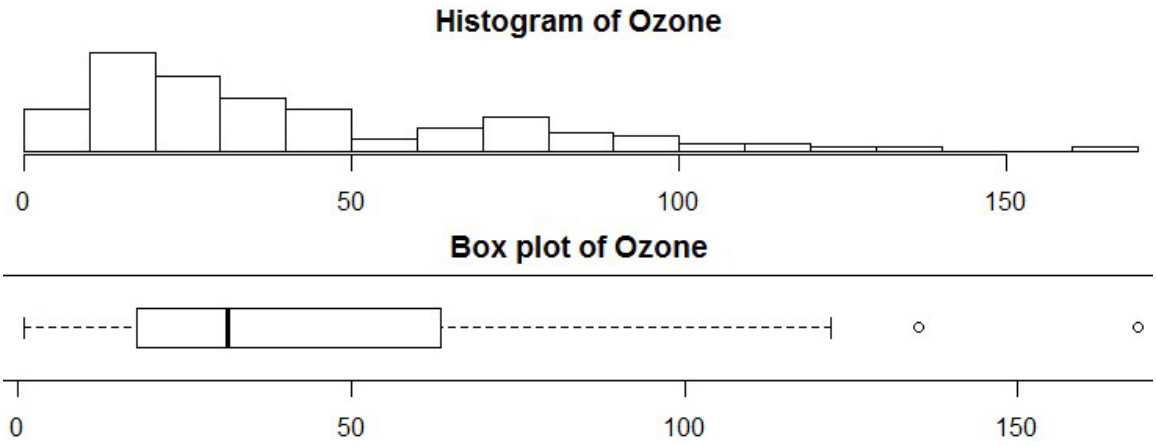

Figure 1.13

Lattice plotting system

```
## install.packages("lattice") # install the lattice
   library if you haven't
library(lattice) # load the library
## Warning: package 'lattice' was built under R version
   3.1.2
## set the seed so our plots are the same
set.seed(10)
## use rnorm(): generate 100 random numbers
x <- rnorm(100)
## use rep() function to generate a vector, with first 25
   elements are 1,
## second 25 elements are 2, ...
f <- rep(1:4, each = 25)
y <- x + f - f * x + rnorm(100, sd = 0.5)
# name the levels of the factor: first 25 elements are in
   Group 1,
## second 25 elements are in Group 2, ...
f <- factor(f, labels = c("Group 1", "Group 2", "Group 3",
    "Group 4"))
xyplot(y ~ x | f)
```

(Figure: 1.14)

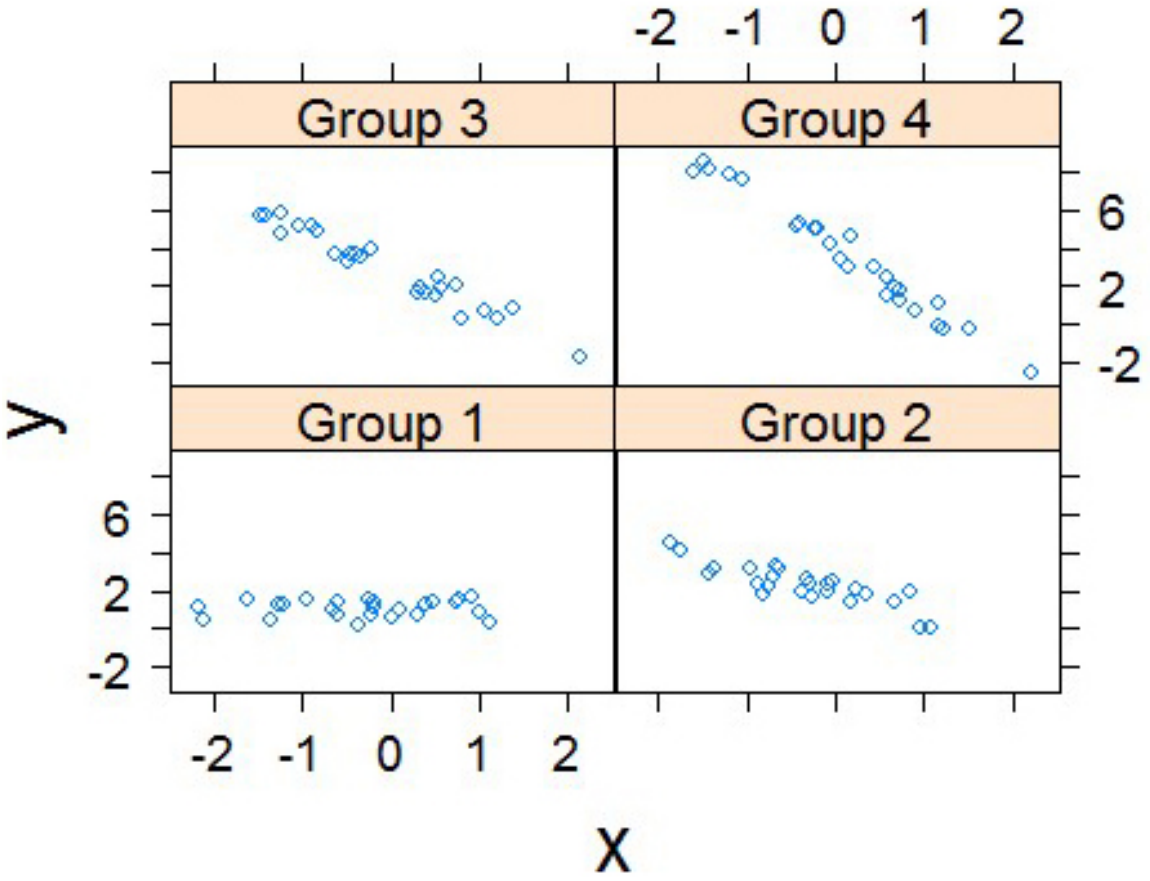

Figure 1.14

Want more on the plot? Customize the panel function:

```
xyplot(y ~ x | f, panel = function(x, y, ...) {
    # call the default panel function for xyplot
    panel.xyplot(x, y, ...)
    # adds a horizontal line at the median
    panel.abline(h = median(y), lty = 2)
    # overlays a simple linear regression line
    panel.lmline(x, y, col = 2)
})
```

(Figure: 1.15)

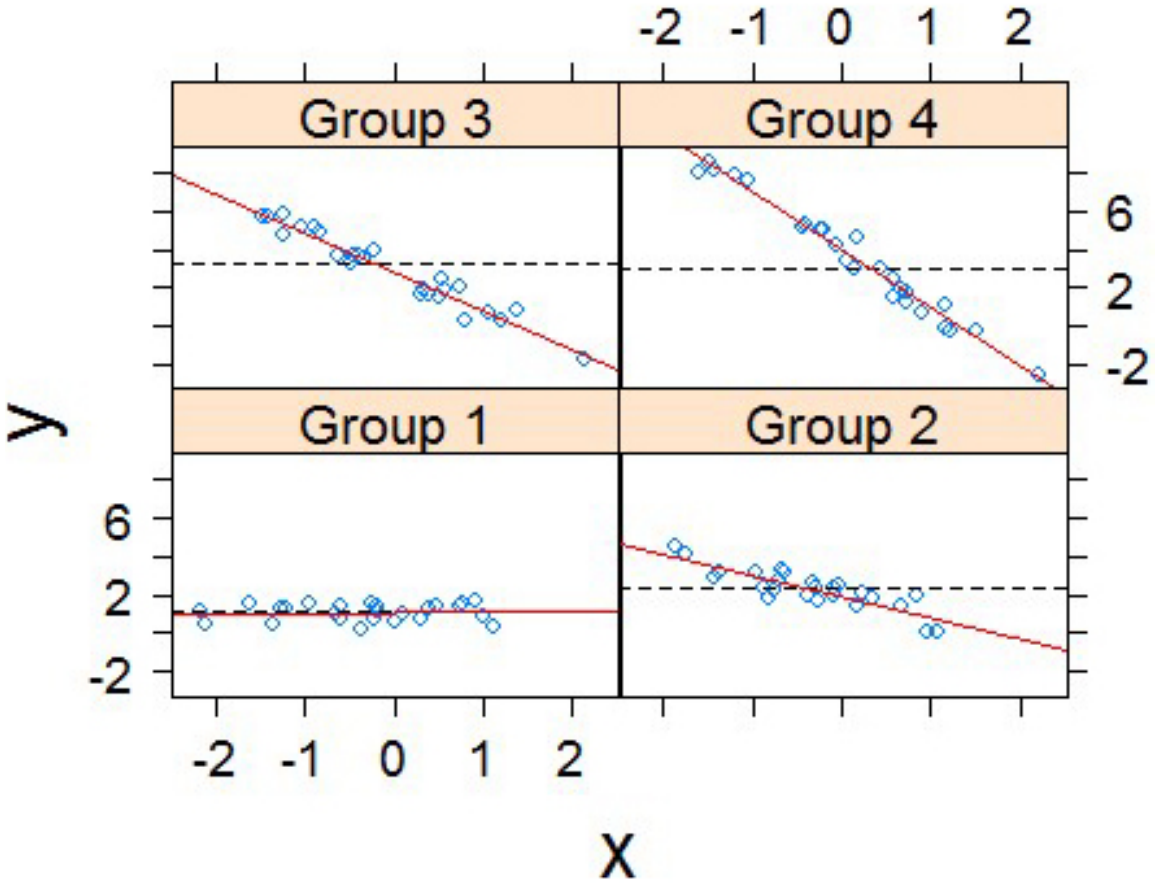

Figure 1.15

Plotting functions

- **xyplot()**: main function for creating scatterplots
- **bwplot()**: box and whiskers plots (box plots)
- **histogram()**: histograms
- **stripplot()**: box plot with actual points
- **dotplot()**: plot dots on "violin strings"
- **splom()**: scatterplot matrix (like pairs() in base plotting system)
- **levelplot()/contourplot()**: plotting image data

Very useful when we want a lot...

```
1 pairs(iris) ## iris is a data set in R
```

(Figure: 1.16)

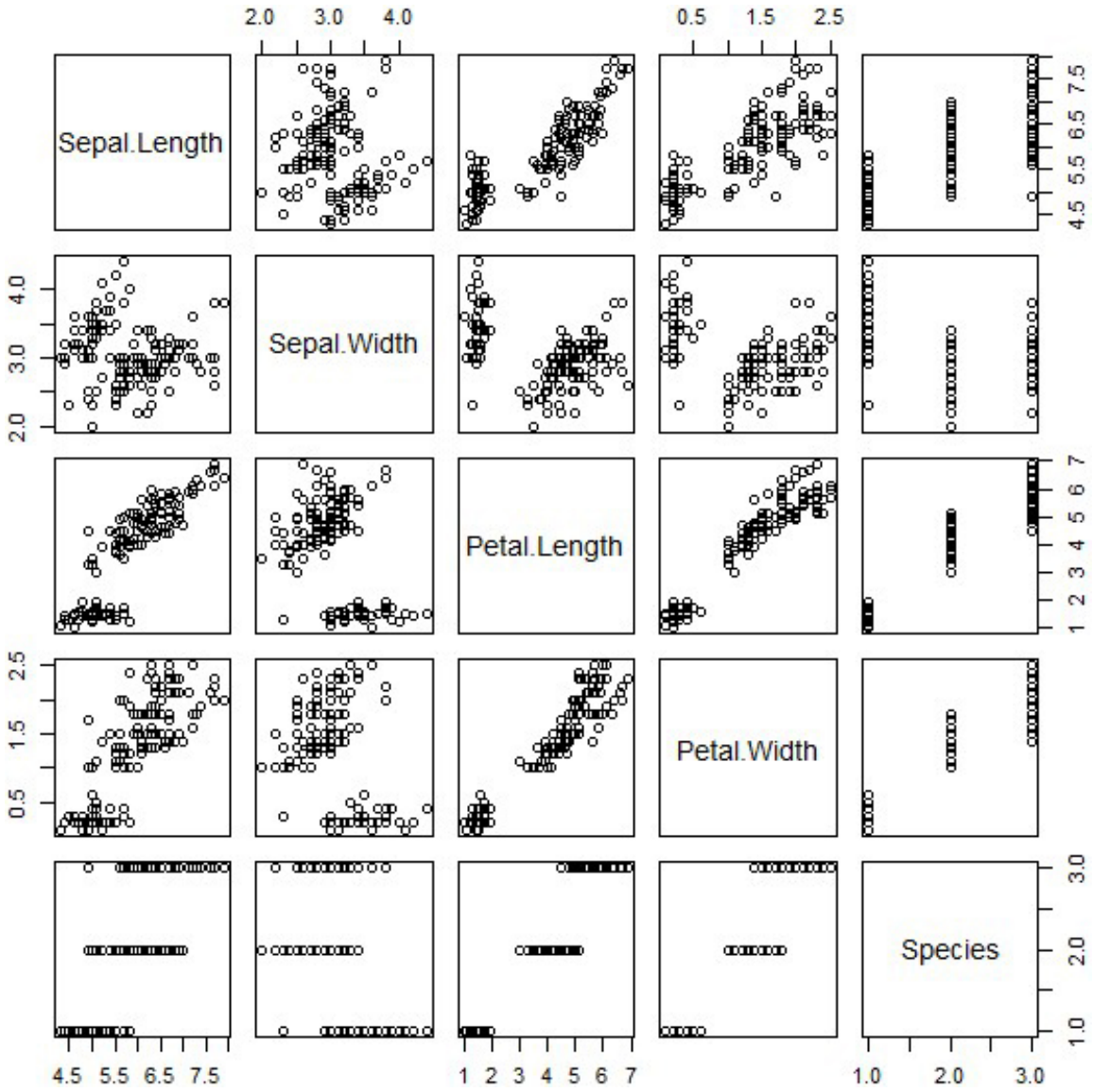

Figure 1.16

ggplot2

- An implementation of the **Grammar of Graphics by Leland Wikinson**
- Written by Hadley Wickham (while he was a graduate student as Iowa State)
- A **"third"** graphics system for R (along with base and lattice) Available from CRAN via install.packages() web site: `http://ggplot2.org(betterdocumentation)`
- Grammar of graphics represents the abstraction of graphics ideas/objects Think **"verb"**, **"noun"**, **"adjective"** for graphics "Shorten" the distance from mind to page
- Two main functions: **qplot()** hides what goes on underneath, which is okay for most operations **ggplot()** is the core function and very flexible for doing this qplot() cannot do

qplot function The qplot() function is the analog to plot() but with many build-in features Syntax somewhere in between base/lattice Difficult to be customized (don't bother, use full ggplot2 power

in that case)

```
## install.packages("ggplot2") # need to install ggplot2 first
library(ggplot2) # load the library
levels(mpg$drv) <- c("4wd", "front-wheel drv", "rear-wheel drv") # rename the levels
theme_set(theme_gray(base_size = 20)) # change text size
qplot(displ, hwy, data = mpg, facets = .~drv)
```

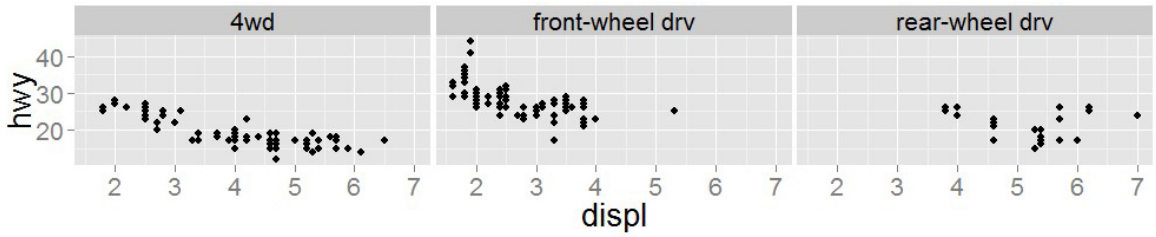

Figure 1.17

ggplot function When building plots in ggplot2 (ggplot, rather than using qplot) The **"artist's palette"** model may be the closest analogy Plots are built up in layers * **Step I: Input the data noun**: the data

```
library(ggplot2) ## need to install and load this library
g <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) ## this would not show you add plot
```

Step II: Add layers adjective: describe the type of plot you will produce.

```
g <- g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ Species) +
  theme(text = element_text(size = 20))
g

#geom_smooth(method=''lm''): add linear regression line;
#geom\_grid(. \~ Species): lay out panels in a grid by Species;
#theme(): revision of the appearance (It is actually the step III)
```

(Figure: 1.18)

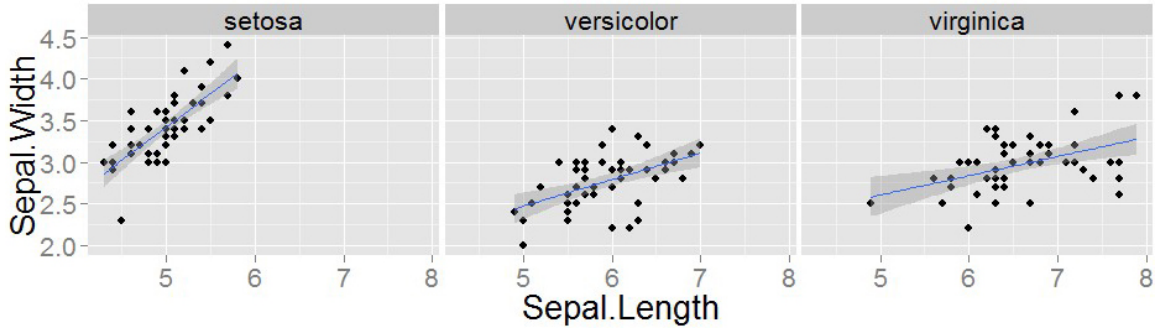

Figure 1.18

Step III: Add metadata and annotation adjective: control the mapping between data and aesthetics.

```
g + ggtitle("Sepal length vs. width for different species"
    ) + theme_bw() +
      theme(text = element_text(size = 18)) # add "verb"

#ggtitle(): add the title
#theme_bw: choose the classic dark-on-light ggplot2 theme.
```

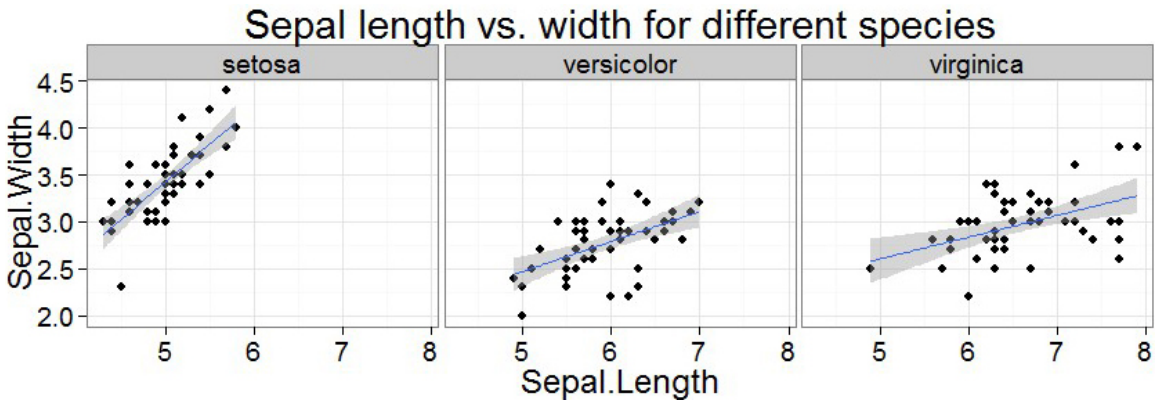

Figure 1.19

Great documentation of ggplot with all functions in step II and III and demos: `http://docs.ggplot2.org/current/`

Overview of the diamond data

```
data(diamonds) # loading diamonds data set
head(diamonds, 9) # first few rows of diamond data set
#   carat       cut color clarity depth table price    x
      y    z
# 1  0.23     Ideal     E     SI2  61.5    55   326 3.95
   3.98 2.43
# 2  0.21   Premium     E     SI1  59.8    61   326 3.89
   3.84 2.31
# 3  0.23      Good     E     VS1  56.9    65   327 4.05
   4.07 2.31
# 4  0.29   Premium     I     VS2  62.4    58   334 4.20
   4.23 2.63
# 5  0.31      Good     J     SI2  63.3    58   335 4.34
   4.35 2.75
# 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94
   3.96 2.48
# 7  0.24 Very Good     I    VVS1  62.3    57   336 3.95
   3.98 2.47
# 8  0.26 Very Good     H     SI1  61.9    55   337 4.07
   4.11 2.53
# 9  0.22      Fair     E     VS2  65.1    61   337 3.87
   3.78 2.49
```

Histogram of carat

```
library(ggplot2)
ggplot(data=diamonds) + geom_histogram(aes(x=carat),
   binwidth=1/5) +
    theme(text = element_text(size = 25))
```

(Figure: 1.20)

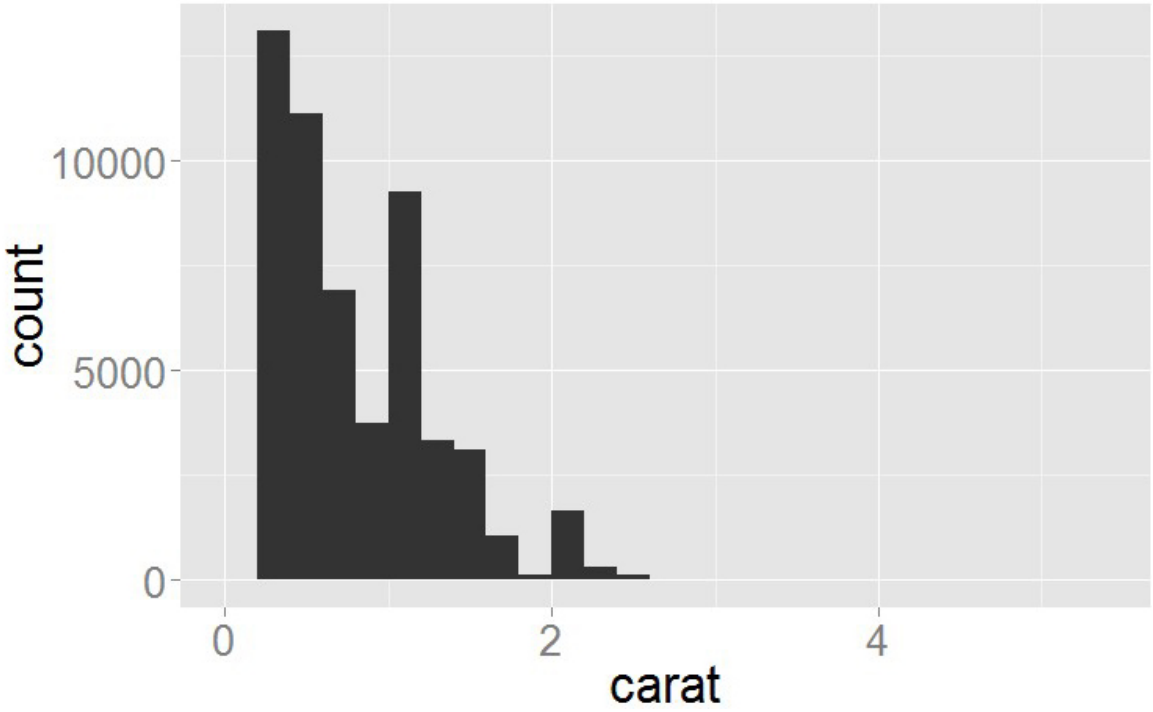

Figure 1.20

Density plot of carat

```
ggplot(data=diamonds) +
geom_density(aes(x=carat),fill="gray50")+
    theme(text = element_text(size = 25))
```

(Figure: 1.21)

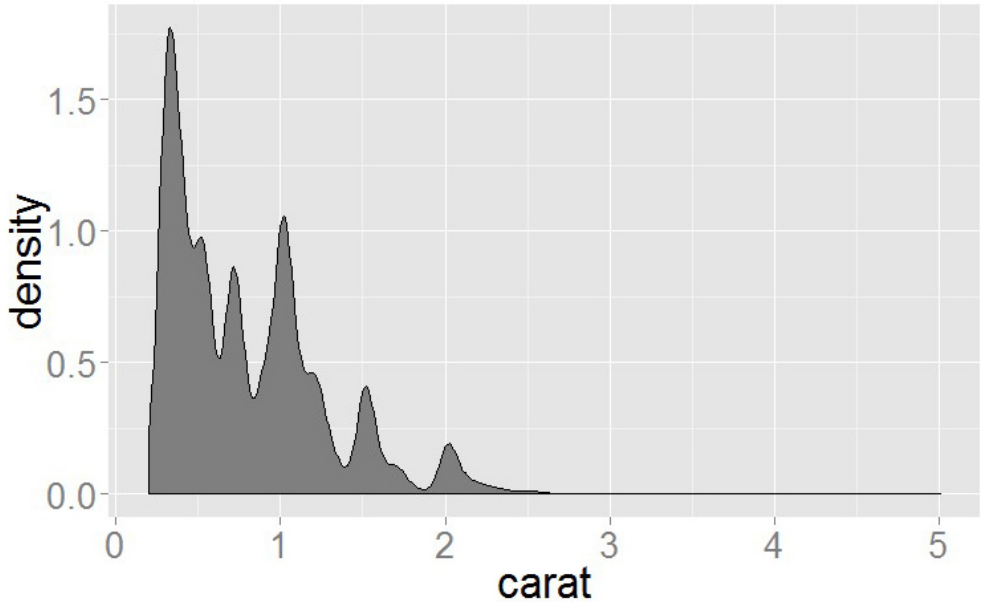

Figure 1.21

Scatter plots (carat vs. price)

```
ggplot(diamonds, aes(x=carat,y=price)) + geom_point() +
    theme(text = element_text(size = 25))
```

(Figure: 1.22)

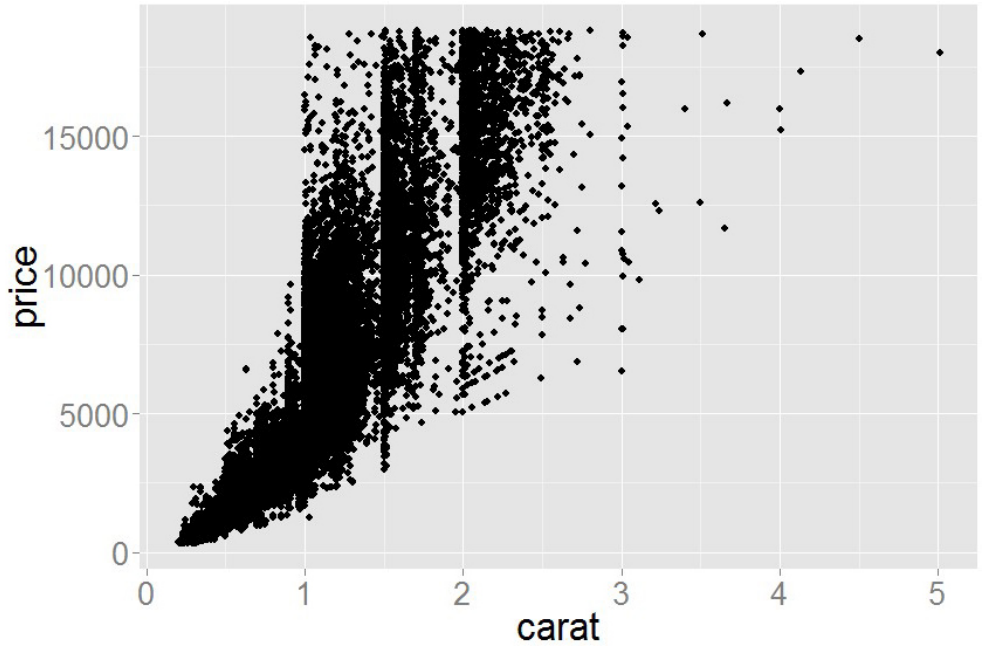

Figure 1.22

Carat with colors

```
g = ggplot(diamonds, aes(x=carat, y=price)) # saving first
    layer as variable
# rendering first layer and adding another layer
g + geom_point(aes(color=color)) + theme(text = element_
   text(size = 25))
```

(Figure: 1.23)

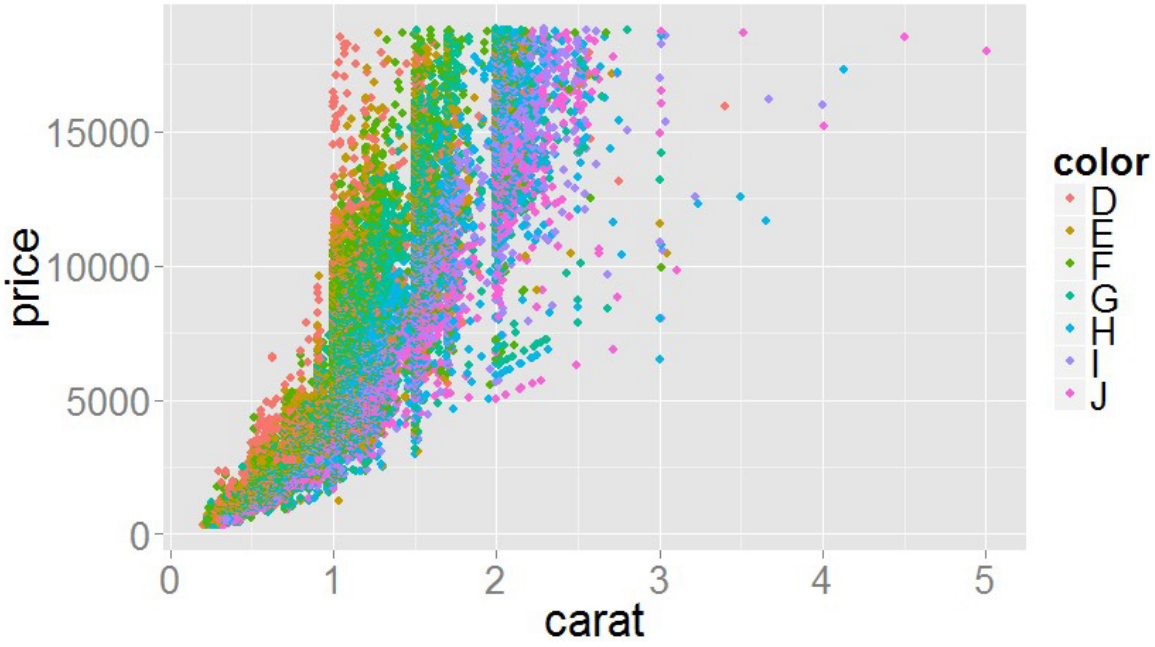

Figure 1.23

Carat with colors (more details)

```
g + geom_point(aes(color=color)) + facet_wrap(~color) +
    theme(text = element_text(size = 20))
```

(Figure: 1.24)

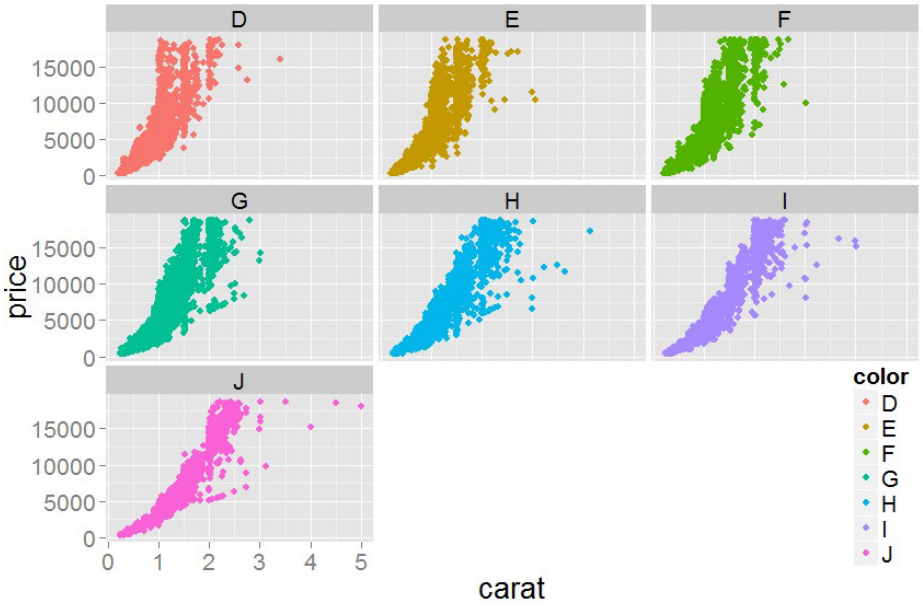

Figure 1.24

Let's consider cut and clarity (Figure: 1.25)

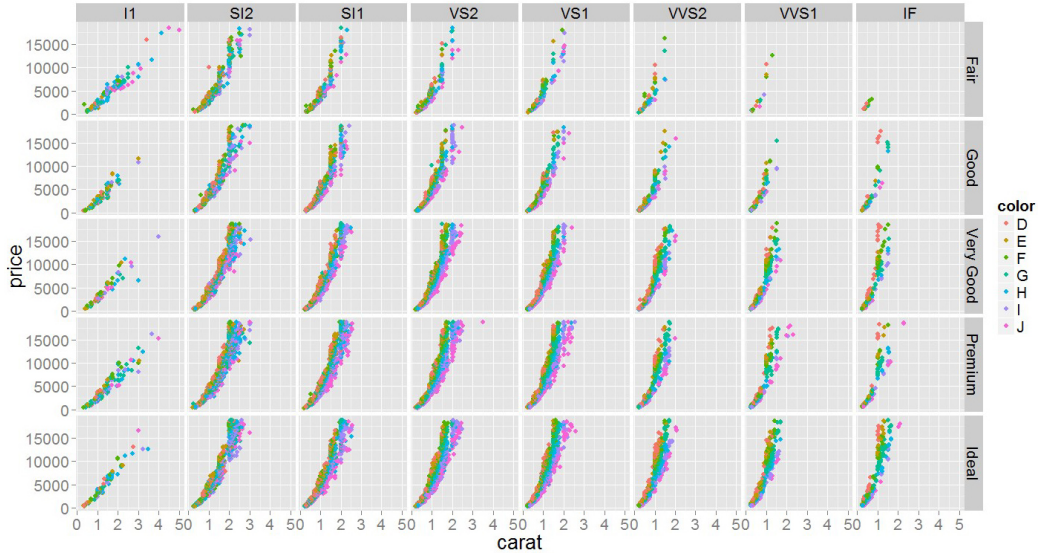

Figure 1.25

Your turn! What is your knowledge of diamond's price after exploring this data?

Fare matters

```
library(ggplot2)
ggplot(titanic_with_child_column, aes(y=Fare, x=Survived))
    +
    geom_boxplot() + facet_grid(Gender~Child) +
    theme(text = element_text(size = 25))
```

(Figure: 1.26)

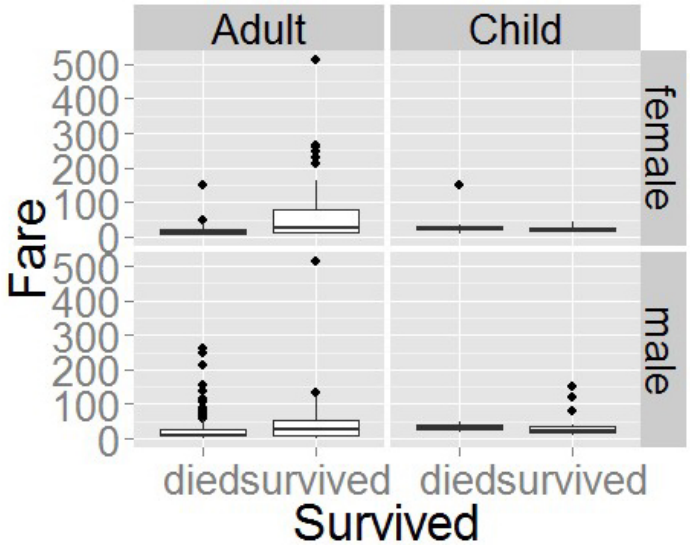

Figure 1.26

How about fare, ship class, port embarkation? (Figure: 1.27)

```
library(ggplot2)
titanic$Pclass = as.factor(titanic$Pclass)
ggplot(titanic, aes(y=Fare, x=Pclass)) + geom_boxplot() +
    facet_grid(~Embarked) + theme(text = element_text(size
        = 25))
```

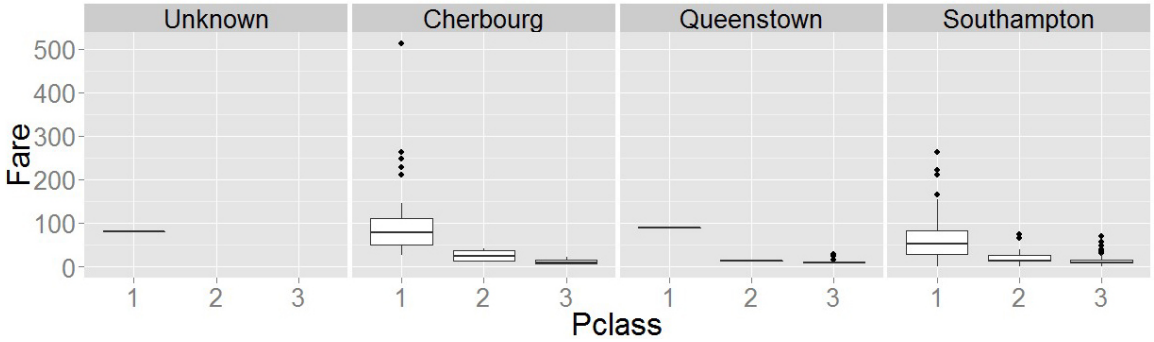

Figure 1.27

Data Science Dojo's Handbook is a collection of exercises that are designed to teach you the basics of data science and data engineering through hands-on experience. The book is accessibly written to guide anyone interested in learning more about data science.

Getting started should be easy. This Handbook will serve as an introduction to the fundamental concepts of data science and engineering, simplifying even the most complex topics to ground your knowledge in a strong foundation. Rather than focusing on the inaccessible math and programming of data science, that often deters most beginners, this book will give users the crash course into data mining theory.

Learn the best practices of the industry with the most up-to-date data science tools. Readers will be exposed to a variety of technologies such as Azure Machine Learning Studio, Cloud-based SQL databases, Hive, Hadoop, stream processors, and data ingestion hubs.

The hands-on approach in this book will cover both data science and data engineering; intertwining the two concepts to encourage future data scientists to cultivate a set of complementary skills.

COVERAGE INCLUDES

- Azure Machine Learning Studio
- Importing & writing datasets
- Data exploration & visualization
- Preprocessing data & missing values cleansing
- Data partitioning & cross validation
- Classification algorithms
- Regression algorithms
- Train, score, & model evaluation
- Deploying a predictive model as a web service
- Text entity extraction
- SQL databasing in the cloud (Azure SQL Database)
- Hadoop & HiveQL (HDInsight)
- Setting up big data pipelines in the cloud
- Analytics on streaming data
- Data ingestion (Azure Event Hubs)
- Stream processing (Azure Stream Analytics & StreamQL)
- Internet of Things

ABOUT US

Data Science Dojo is a Seattle company that offers bootcamps, corporate trainings, consulting services, and community events in all things data science. The company started in 2014, stemming from the realization that data science and data engineering were being treated as separate entities. Data Science Dojo instead treats the concepts holistically by encouraging a hands-on, accessible approach to learning. Data Science Dojo's carefully crafted curriculum and community of data scientists have enabled individuals from all around the world to unleash the data scientists within.

datasciencedojo

unleash the data scientist in you