

Advanced Forecasting Methods

Session 1

Jyotirmoy Bhattacharya

IIM Kozhikode

Winter 2010

Administrative Information

Quizzes & Assignments	30%
Project	30%
End-term	30%

Table: Evaluation pattern

Structure of the course

- Modeling, computation, evaluation
- Explanatory models
- Black-box models
- Not just time

The R Environment

Why R?

It is free, flexible, powerful and has a very rich set of third-party packages supporting a large variety of data analysis and graphics techniques.

Getting R

- Website: <http://www.r-project.org/>
- Download for Windows:
<http://cran.wustl.edu/bin/windows/base/release.htm>

R is a calculator

```
> 3 * (1 + 1)
```

```
[1] 6
```

```
> 4^2
```

```
[1] 16
```

```
> sqrt(5)
```

```
[1] 2.236068
```

```
> cos(pi)
```

```
[1] -1
```

Variables

```
> x <- 5  
> sqrt(x)
```

```
[1] 2.236068
```

```
> x <- 3 + 1  
> sqrt(x)
```

```
[1] 2
```

Vectors

- The function `c()` adds together its arguments to give us a vector.

```
> x <- c(2, 4, 16)
```

```
> y <- c(1, 2, 3)
```

- Arithmetic on vectors is component-wise

```
> x + y
```

```
[1] 3 6 19
```

```
> x * y
```

```
[1] 2 8 48
```

```
> sqrt(x)
```

```
[1] 1.414214 2.000000 4.000000
```

Recycling

- If two or more vectors of unequal length occur in an expression, the shorter vectors are repeated as often as required

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

```
> y <- c(5, 6)
```

```
> x + y
```

```
[1] 6 8 8 10 10
```

- Scalars behave like vectors of length 1

```
> x + 3
```

```
[1] 4 5 6 7 8
```

Logical operators

Logical operations also happen elementwise.

```
> x <- c(1, 2, 3)
```

```
> x >= 2
```

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

```
> x == 1
```

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

```
> x != 3
```

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

```
> y <- c(0, 3, 4)
```

```
> x > y
```

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

Compound logical operators

```
> x <- c(1, 2, 3)
```

- And

```
> x > 1 & x < 3
```

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

- Or

```
> x < 2 | x > 2
```

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

- Not

```
> !(x == 2)
```

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

Selecting elements from vectors

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

- By mentioning element numbers (subscripts)

```
> x[2]
```

```
[1] 4
```

```
> x[c(3, 4)]
```

```
[1] 1 2
```

```
> x[2:5]
```

```
[1] 4 1 2 6
```

- Negative numbers indicate items to exclude

```
> x[c(-2, -4)]
```

```
[1] 3 1 6 5
```

Selecting elements (contd.)

- Using logical conditions

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

```
> cond <- (x >= 3)
```

```
> cond
```

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

```
> x[cond]
```

```
[1] 3 4 6 5
```

or simply

```
> x[x >= 3]
```

```
[1] 3 4 6 5
```

R Packages

- R's functionality can be extended using third-part packages.
- CRAN (Comprehensive R Archive Network) is a set of sites across the world that maintain a collection of free R packages.
- You can install CRAN packages from within R.

Loading packages

- The packages `fma` provides all the data sets from our textbook and `forecast` supports many of the methods explained in it.
- You will need to download and install these packages once on your system.
- For every session, you will need to first load these packages:

```
> library(fma)
```

```
âĀĲtseriesâĀĴ version: 0.10-22
```

```
âĀĲtseriesâĀĴ is a package for time series analysis and  
finance.
```

```
See âĀĲlibrary(help="tseries")âĀĴ for details.
```

```
This is forecast 2.03
```

```
> library(forecast)
```

Loading data

- The dataset `auto` in the package `fma` contains the price, mileage, age and country of origin for 45 automobiles. It is discussed in Chapter 2.
- Loading the `fma` package automatically loads this dataset.
- R can also load external data in a variety of formats. Read the help for the functions `read.csv()` and `read.table()`.

Data frames

- R stores data sets internally in a type of objects known as a data frame.

```
> class(auto)
```

```
[1] "data.frame"
```

- A data frame has named columns for each variable and rows for each observation.

```
> dim(auto)
```

```
[1] 45  4
```

```
> names(auto)
```

```
[1] "Model"    "Country" "Mileage" "Price"
```

Summarizing data

```
> summary(auto)
```

Model	Country	Mileage	Price
Ford Probe: 1	Japan:19	Min. :18.00	Min. : 5866
Subaru XT 4: 1	USA :26	1st Qu.:20.00	1st Qu.:10565
Ford Tempo 4: 1		Median :22.00	Median :13150
Mazda 929 V6: 1		Mean :23.71	Mean :13132
Mazda MPV V6: 1		3rd Qu.:26.00	3rd Qu.:14944
Nissan Van 4: 1		Max. :35.00	Max. :24760
(Other)		:39	

Mean, median, quartiles

- Mean is the usual arithmetic average:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i$$

- The median is a number m such that at most half the observations are less than m and at most half of the observations are greater than m .
- The first quartile q_1 is a number such that at most one-fourth of the observations are less than q_1 and at most three-fourths of the observations are greater than q_1 .

Working with subsets of data

- An individual column of a data frame can be accessed by using the notation `dataframe$variable`.

```
> summary(auto$Mileage)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
18.00	20.00	22.00	23.71	26.00	35.00

- In general any subset of a data frame can be accessed by using the notation `dataframe[rows,cols]`.

```
> auto[2, 2:3]
```

	Country	Mileage
2	USA	18

- Vectors of names can be used for columns

```
> auto[2:3, c("Model", "Mileage")]
```

	Model	Mileage
2	Chevrolet Lumina APV V6	18
3	Dodge Grand Caravan V6	18

Working with subsets of data

- Keeping the row/column specification empty selects all the rows or columns:

```
> auto[2:3, ]
```

	Model	Country	Mileage	Price
2	Chevrolet Lumina APV	V6 USA	18	13995
3	Dodge Grand Caravan	V6 USA	18	15395

- For eg., to select all Japanese cars

```
> jpauto <- auto[auto$Country == "Japan", ]
```

Measures of dispersion

- Standard deviation

$$\sigma_X = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2}$$

```
> sd(jpauto$Mileage)
```

```
[1] 5.344287
```

Measures of dispersion(contd)

- Mean absolute deviation

$$\frac{1}{n} \sum_{i=1}^n |X_i - \bar{X}|$$

```
> mean(abs(jpauto$Mileage - mean(jpauto$Mileage)))
```

```
[1] 4.371191
```

- Median absolute deviation

```
> mad(jpauto$Mileage)
```

```
[1] 4.4478
```

Comparing countries

```
> by(auto$Mileage, auto$Country, summary)
```

```
auto$Country: Japan
```

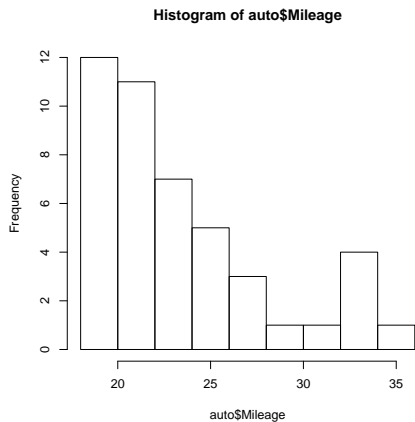
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
19.00	20.50	23.00	24.68	27.50	35.00

```
-----  
auto$Country: USA
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
18.00	20.25	22.00	23.00	24.75	33.00

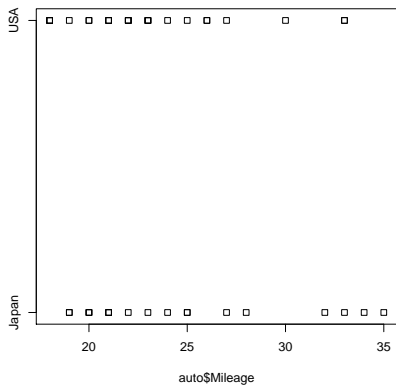
Histograms

```
> hist(auto$Mileage)
```



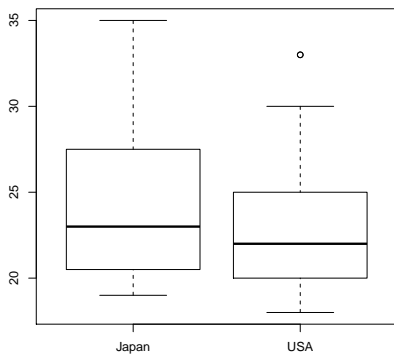
Stripchart

```
> stripchart(auto$Mileage ~ auto$Country)
```



Boxplot

```
> boxplot(auto$Mileage ~ auto$Country)
```



Interpreting the boxplot

- 1 The line in the middle is the median.
- 2 The two edges of the box are the first and the third quartile respectively.
- 3 All observations whose distance from the median is greater than 1.5 times the distance between the quartiles are considered as outliers and plotted separately.
- 4 The whiskers extend to cover the most distant observations which are not outliers.
- 5 (3) and (4) are R's default behavior which you can change.

Sorting

```
> order(jpauto$Price)

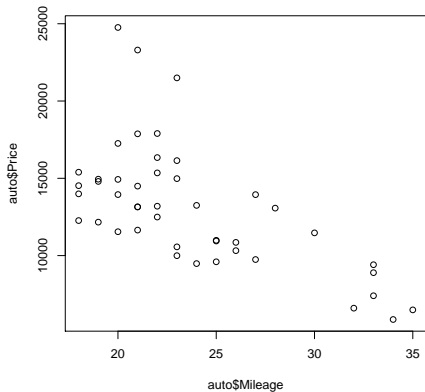
[1] 18 19 16 17 12 13 7 15 11 14 5 2 4 1 6 9 10 8 3

> jpauto[order(jpauto$Price), c("Model", "Price", "Mileage")]

      Model Price Mileage
44   Subaru Justy 3  5866    34
45   Toyota Tercel 4  6488    35
40   Mazda Protege 4  6599    32
43   Honda Civic CRX Si 4  9410    33
31   Subaru Loyale 4  9599    25
33   Mitsubishi Galant 4 10989    25
14   Nissan Stanza 4 11650    21
38   Subaru XT 4 13071    28
29   Nissan 240SX 4 13249    24
37   Honda Prelude Si 4WS 4 13945    27
12   Nissan Axxess 4 13949    20
7    Nissan Van 4 14799    19
11   Mitsubishi Wagon 4 14929    20
6    Mazda MPV V6 14944    19
13   Mitsubishi Sigma V6 17879    21
23   Nissan Maxima V6 17899    22
27   Toyota Cressida 6 21498    23
16   Mazda 929 V6 23300    21
9    Acura Legend V6 24760    20
```

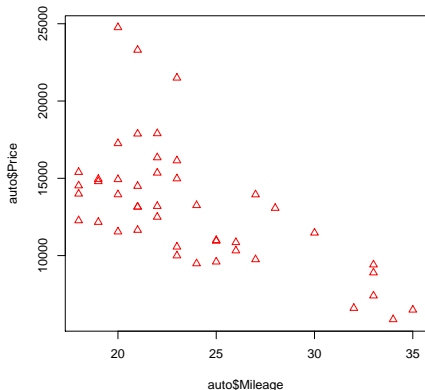
Scatterplots

```
> plot(auto$Mileage, auto$Price, type = "p")
```



Scatterplots(contd.)

```
> plot(auto$Mileage, auto$Price, type = "p", pch = 24, col = 'r')
```



Factors

- Categorical data is represented within R as objects of the class `factor`.

```
> class(auto$Country)
```

```
[1] "factor"
```

- Each potential value is given a numeric code and the code and the descriptive labels are stored separately.

```
> levels(auto$Country)
```

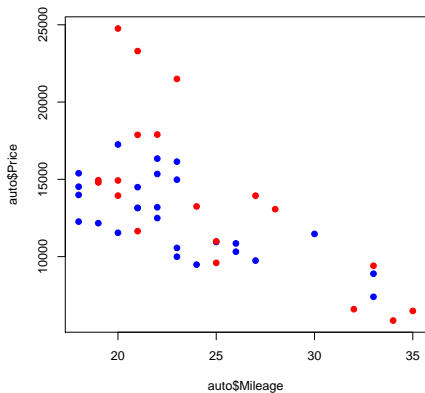
```
[1] "Japan" "USA"
```

```
> unclass(auto$Country)
```

```
[1] 2 2 2 2 2 1 1 2 1 2 1 1 1 1 2 1 2 2 2 2 2 2 1 2 2 2 1 2 1 2  
[39] 2 1 2 2 1 1 1  
attr(,"levels")  
[1] "Japan" "USA"
```

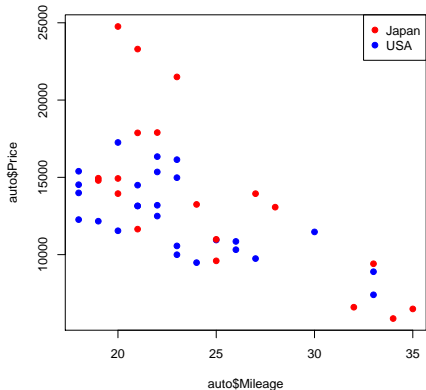
Scatterplots(contd.)

```
> clrs <- c("red", "blue")  
> clrsPerObs <- clrs[auto$Country]  
> plot(auto$Mileage, auto$Price, type = "p", pch = 19, col = clrsPerObs)
```



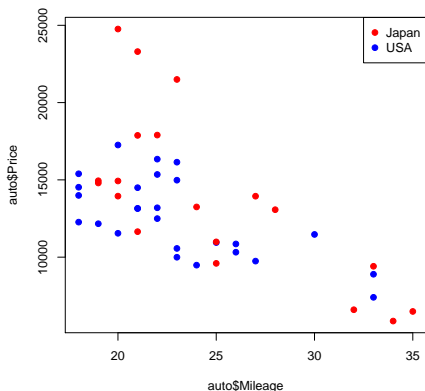
Scatterplots(contd.)

```
> clr = c("red", "blue")
> clrPerObs <- clr[auto$Country]
> plot(auto$Mileage, auto$Price, type = "p", pch = 19, col = clrPerObs)
> legend("topright", levels(auto$Country), pch = 19, col = c("red",
+ "blue"))
```



Interactive Graphics

```
> clr1 <- c("red", "blue")
> clrPerObs <- clr1[auto$Country]
> plot(auto$Mileage, auto$Price, type = "p", pch = 19, col = clrPerObs)
> legend("topright", levels(auto$Country), pch = 19, col = c("red",
+   "blue"))
```



```
> identify(auto$Mileage, auto$Price, auto$Model)
```

Time Series

- Time series data is represented by R classes that keep track of the times at which the data was observed.
- The dataset `beer` has observation on monthly Australian beer production in megalitres from Jan 1991 to Aug 1995.

```
> data(beer)
> class(beer)

[1] "ts"

> start(beer)

[1] 1991  1

> frequency(beer)

[1] 12

> end(beer)

[1] 1995  8

> summary(beer)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
119.0  135.5   145.5   149.3   156.2   192.0
```

Making your own time series

- The `ts()` function turns a numeric vector into a time series

```
> x <- c(1, 3, 9, 4, 6, 22, 0.5)
```

```
> class(x)
```

```
[1] "numeric"
```

```
> y <- ts(x, freq = 4, start = c(2010, 1))
```

```
> class(y)
```

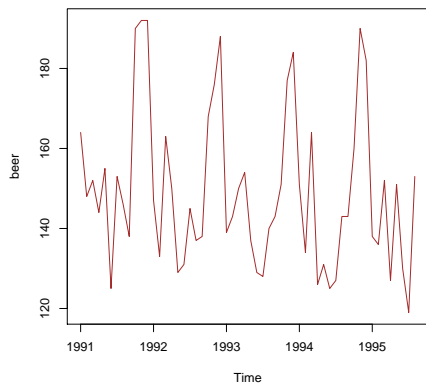
```
[1] "ts"
```

```
> y
```

```
      Qtr1 Qtr2 Qtr3 Qtr4
2010  1.0  3.0  9.0  4.0
2011  6.0 22.0  0.5
```

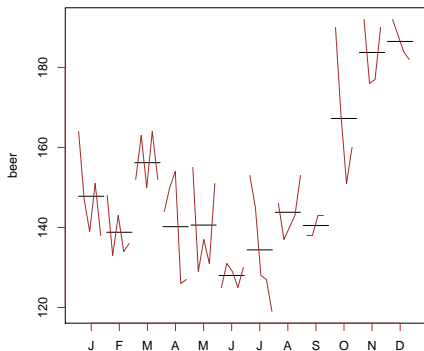
Plotting time series

```
> plot(beer, col = "brown")
```



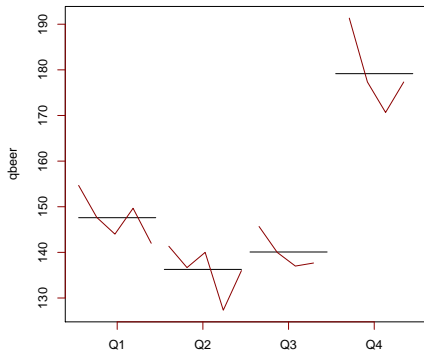
Seasonal plots

```
> monthplot(beer, col = "brown")
```



Seasonal plot (quarterly)

```
> qbeer <- aggregate(beer, 4, mean)
> monthplot(qbeer, col = "darkred")
```



What color names are available?

```
> length(colors())
```

```
[1] 657
```

```
> colors()
```

```
[1] "white"                "aliceblue"          "antiquewhite"
[4] "antiquewhite1"       "antiquewhite2"     "antiquewhite3"
[7] "antiquewhite4"       "aquamarine"        "aquamarine1"
[10] "aquamarine2"        "aquamarine3"       "aquamarine4"
[13] "azure"               "azure1"            "azure2"
[16] "azure3"             "azure4"            "beige"
[19] "bisque"             "bisque1"           "bisque2"
[22] "bisque3"           "bisque4"           "black"
[25] "blanchedalmond"     "blue"              "blue1"
[28] "blue2"             "blue3"             "blue4"
[31] "blueviolet"        "brown"             "brown1"
[34] "brown2"           "brown3"            "brown4"
[37] "burlywood"         "burlywood1"        "burlywood2"
[40] "burlywood3"        "burlywood4"        "cadetblue"
[43] "cadetblue1"        "cadetblue2"        "cadetblue3"
[46] "cadetblue4"        "chartreuse"        "chartreuse1"
[49] "chartreuse2"       "chartreuse3"       "chartreuse4"
[52] "chocolate"         "chocolate1"        "chocolate2"
[55] "chocolate3"       "chocolate4"        "coral"
[58] "coral1"           "coral2"            "coral3"
[61] "coral4"           "cornflowerblue"    "cornsilk"
[64] "cornsilk1"         "cornsilk2"         "cornsilk3"
[67] "cornsilk4"         "cyan"              "cyan1"
[70] "cyan2"            "cyan3"             "cyan4"
[73] "darkblue"          "darkcyan"          "darkgoldenrod"
[76] "darkgoldenrod1"    "darkgoldenrod2"    "darkgoldenrod3"
[79] "darkgoldenrod4"    "darkgray"          "darkgreen"
[82] "darkgrey"         "darkkhaki"         "darkslateblue"
```

Monthly averages

- `cycle()` gives us the sub-period for each time series observation.
- `aggregate()` for ordinary data can group data on the basis of a list of classifying variables.
- `unclass()` can convert time-series data into ordinary data.

```
> aggregate(unclass(beer), list(month = cycle(beer)), mean)
```

	month	x
1	1	147.80
2	2	138.80
3	3	156.20
4	4	140.20
5	5	140.60
6	6	128.00
7	7	134.40
8	8	143.80
9	9	140.50
10	10	167.25
11	11	183.75
12	12	186.50