CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 02
2020 Spring Data-622
Review of Statistics and Probability with R

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Refresher

In the next two weeks we will set a goal to achieve:
working proficiency in R and
introduce essential concepts from Statistical/Probability/Linear Algebra

R is a comprehensive mature language, 35 years in the running.
We will introduce capabilities limited to what is required for this course.

If R is an ocean, [Statistics/Probability/Linear Algebra] are an universe.
Therefore, the next two lectures, are limited review of topics, on need-only basis

Loading datasets into data.frames
Partitioning datasets
Apply/lapply vector operations

EDA:descriptive/summary statistics
EDA:correlation,Cov,std,var

Normalizing/scaling

Loading datasets into data.frames
Partitioning datasets
Apply/lapply vector operations

EDA:descriptive/summary statistics
EDA:correlation,Cov,std,var

Normalizing/scaling

Loading data

Source:
- preloaded,
- text,
- Files on disk:
  - Tab
  - comma seperated values,
- From the internet:
  - dataset using URLs

Visualizing
- Pairs

Data Preparation
- Coding
- Scaling
- sampling
- randomizing
- Training
- Test

For R, you can use Rstudio or Rgui or any other tool you prefer.

Working Environment: R on IBM Cloud

rkannan@F4Linux1: ~

```
rkannan@F4Linux1:~$ R

R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: powerpc64le-unknown-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.

> alldsets<-data()
> str(alldsets)
List of 4
 $ title  : chr "Data sets"
 $ header : NULL
 $ results: chr [1:105, 1:4] "datasets" "datasets" "datasets" "
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "Package" "LibPath" "Item" "Title"
 $ footer : chr "Use `data(package = .packages(all.available =
vailable* packages."
```

preloaded data
str function

For R, you can use Rstudio or Rgui. Or on IBM Cloud as shown above. On IBM, enter R to start R.

Working Environment: R on IBM Cloud

```
> head(alldsets$results)
     Package    LibPath               Item
[1,] "datasets" "/usr/lib/R/library" "AirPassengers"
[2,] "datasets" "/usr/lib/R/library" "BJsales"
[3,] "datasets" "/usr/lib/R/library" "BJsales.lead (BJsales)"
[4,] "datasets" "/usr/lib/R/library" "BOD"
[5,] "datasets" "/usr/lib/R/library" "CO2"
[6,] "datasets" "/usr/lib/R/library" "ChickWeight"
     Title
[1,] "Monthly Airline Passenger Numbers 1949-1960"
[2,] "Sales Data with Leading Indicator"
[3,] "Sales Data with Leading Indicator"
[4,] "Biochemical Oxygen Demand"
[5,] "Carbon Dioxide Uptake in Grass Plants"
[6,] "Weight versus age of chicks on different diets"
> dim(alldsets$results)
[1] 105   4
```

str function reveals the inner structure of any R Object
head – is like Unix *head* cmd reveals 6/8 lines
We find alldsets$results is a data.frame of 105 rows
each with 4 columns, using *dim* function

For R, you can use Rstudio or Rgui. Or on IBM Cloud as shown above.

R comes with many datasets

```
7   6  6  6 | WWWusage            Internet Usage per Minute
8   4  4  4 | WorldPhones         The World's Telephones
9  12 12 12 | ability.cov         Ability and Intelligence Tests
10  7  7  7 | airmiles            Passenger Miles on Commercial US Airlines, 1937-1960
11  5  5  5 | airquality          New York Air Quality Measurements
> data()    | anscombe            Anscombe's Quartet of 'Identical' Simple Linear Regressions
>           | attenu              The Joyner-Boore Attenuation Data
            | attitude            The Chatterjee-Price Attitude Data
            | austres             Quarterly Time Series of the Number of Australian Residents
            | beaver1 (beavers)   Body Temperature Series of Two Beavers
```

```
> anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89
```

Anscombe, is one of the 105 pre-loaded datasets
Anscombe is an instructive dataset in that it reminds us the importance of visual review of data and relationships.
Quantitative review alone could be misleading – a simple chart reveals lot more
John Snow stopped Cholera in London merely With visualization.

Anscombe quartets in R

```
> summ1<-lm(anscombe$y1~anscombe$x1)
> summ2<-lm(anscombe$y2~anscombe$x2)
> summ3<-lm(anscombe$y3~anscombe$x3)
> summ4<-lm(anscombe$y4~anscombe$x4)
> summ1

Call:
lm(formula = anscombe$y1 ~ anscombe$x1)

Coefficients:
(Intercept)  anscombe$x1
     3.0001       0.5001

> summ2

Call:
lm(formula = anscombe$y2 ~ anscombe$x2)

Coefficients:
(Intercept)  anscombe$x2
      3.001        0.500

> summ3

Call:
lm(formula = anscombe$y3 ~ anscombe$x3)

Coefficients:
(Intercept)  anscombe$x3
     3.0025       0.4997

> summ4

Call:
lm(formula = anscombe$y4 ~ anscombe$x4)

Coefficients:
(Intercept)  anscombe$x4
     3.0017       0.4999
```

Note that

```
> apply(anscombe,2,sd)
      x1       x2       x3       x4       y1       y2       y3       y4
3.316625 3.316625 3.316625 3.316625 2.031568 2.031657 2.030424 2.030579
> apply(anscombe,2,mean)
      x1       x2       x3       x4       y1       y2       y3       y4
9.000000 9.000000 9.000000 9.000000 7.500909 7.500909 7.500000 7.500909
> 
```

Continuing, on with Anscombe,the Central tendencies and dispersion are Identical. A regression yields over them near identical results. We can iterate over the columns of a df using *apply* func.

So are they identical?

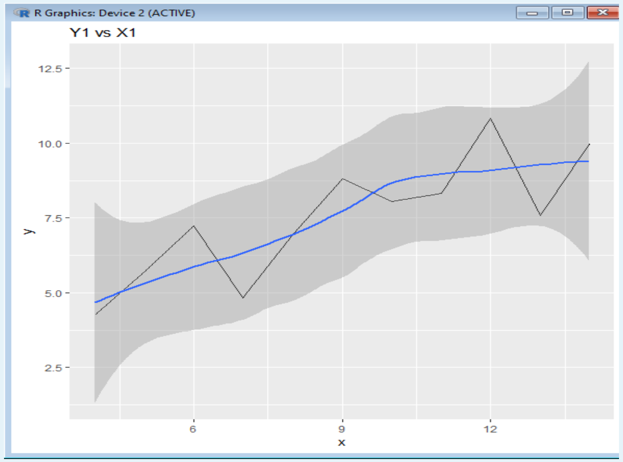
Let us seek some guidance with charts.

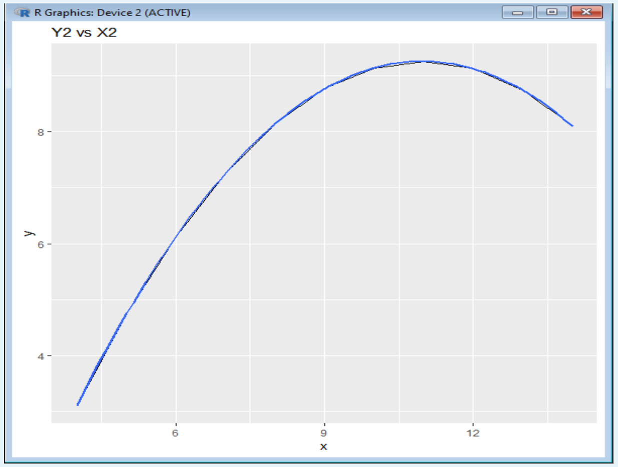
Anscombe Charts!

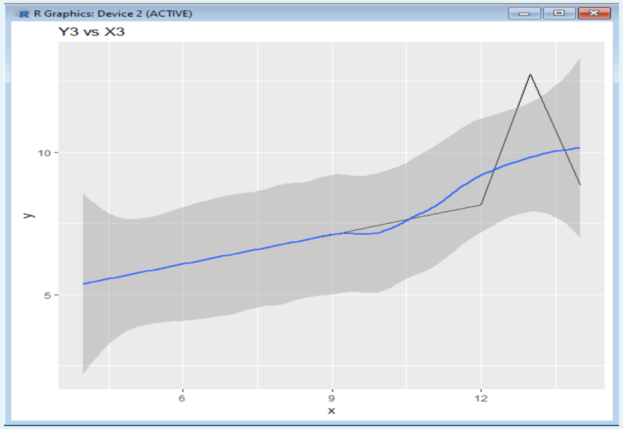
Wow! Anscombe vectors are anything but same.

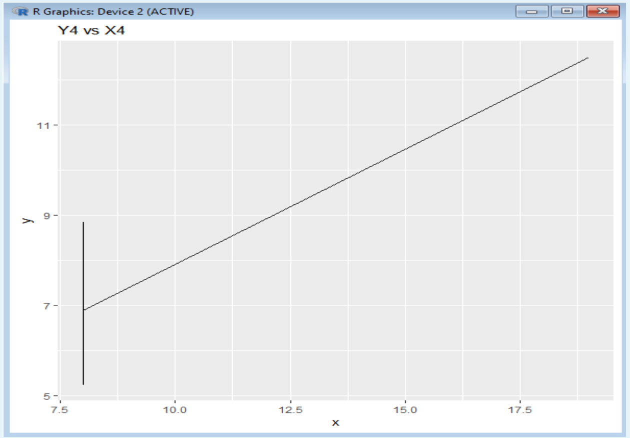
A picture is Worth a Thousand Words.

EDA is important to get to know your data.







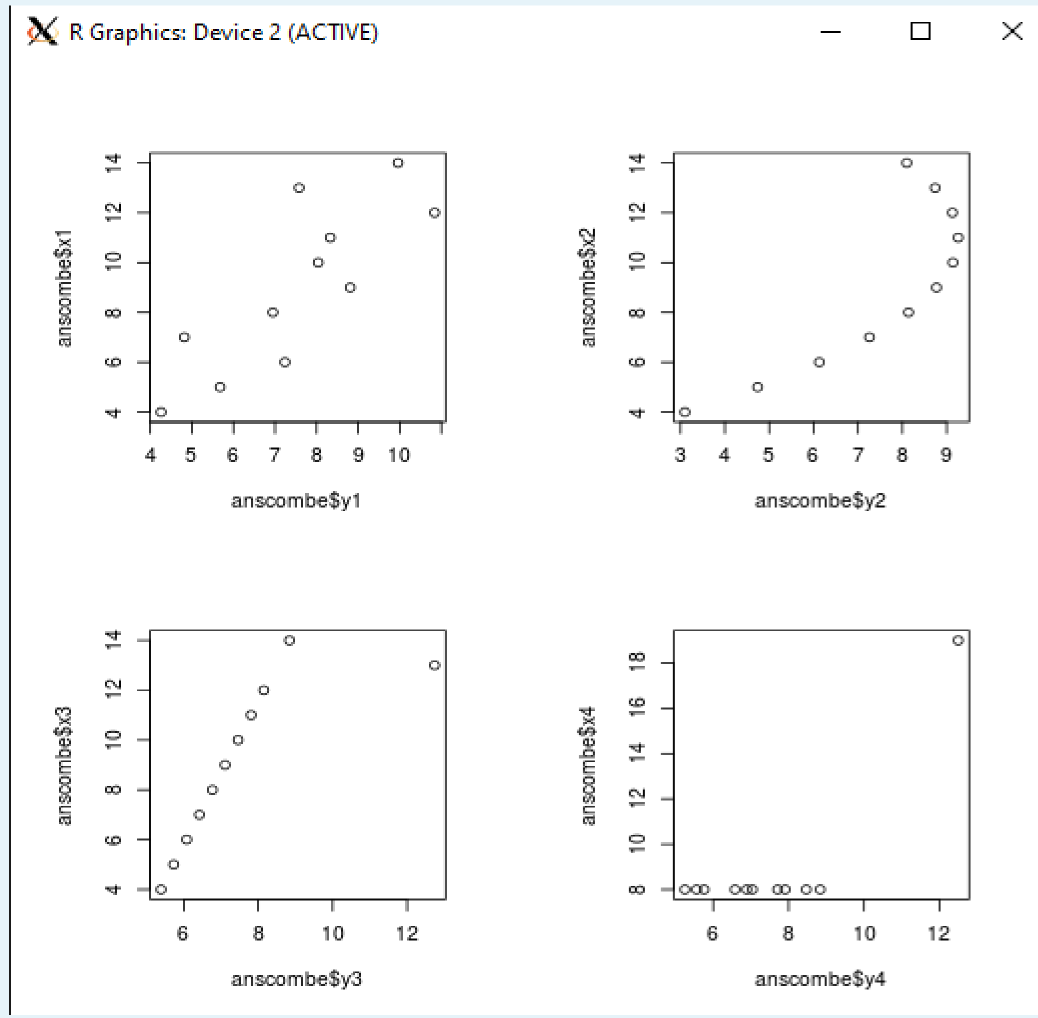



```
ggplot(data.frame(x=anscombe$x2,y=anscombe$y2),aes(x=x,y=y))+ggtitle("Y2 vs X2")+geom_line()+geom_smooth()
ggplot(data.frame(x=anscombe$x1,y=anscombe$y1),aes(x=x,y=y))+ggtitle("Y1 vs X1")+geom_line()+geom_smooth()
ggplot(data.frame(x=anscombe$x3,y=anscombe$y3),aes(x=x,y=y))+ggtitle("Y3 vs X3")+geom_line()+geom_smooth()
ggplot(data.frame(x=anscombe$x4,y=anscombe$y4),aes(x=x,y=y))+ggtitle("Y4 vs X4")+geom_line()+geom_smooth()
```

Base R comes with plot

Plot anscombe data side by side

old<-par(mfrow=c(2,2),pty="s")
plot(anscombe$y1,anscombe$x1)
plot(anscombe$y2,anscombe$x2)
plot(anscombe$y3,anscombe$x3)
plot(anscombe$y4,anscombe$x4)



Vector processing with R

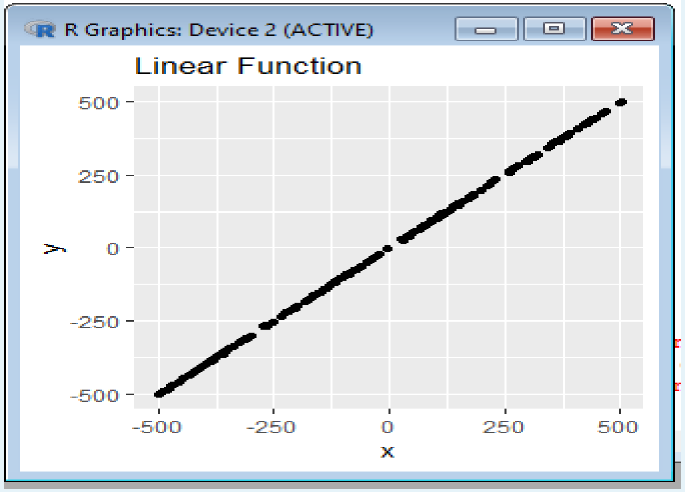
set.seed(13); # so it can be repeated and reproduced
base_x<-sample(-500:500,200,replace=T);
 # base_x is a vector of 200 numbers between -500, 500
delta_x<-rnorm(200) # vector of 200 random normal !
 # numbers with 0 mean and a variance of 1
delta_2_13<-rnorm(200,2,13)# 200 randorm normal
 # numbers with mean 2 and variance 13
expt_x<-base_x+delta_x ; # expt is vector of perturbations
*# # y=x+2*sin(1.5*x)+N(0,0.2)*
*Y<-rnorm(200,0,0.2)+expt_x+2*sin(1.5*expt_x)*
dfyx<-data.frame(y=Y,x=expt_x)# linear rel

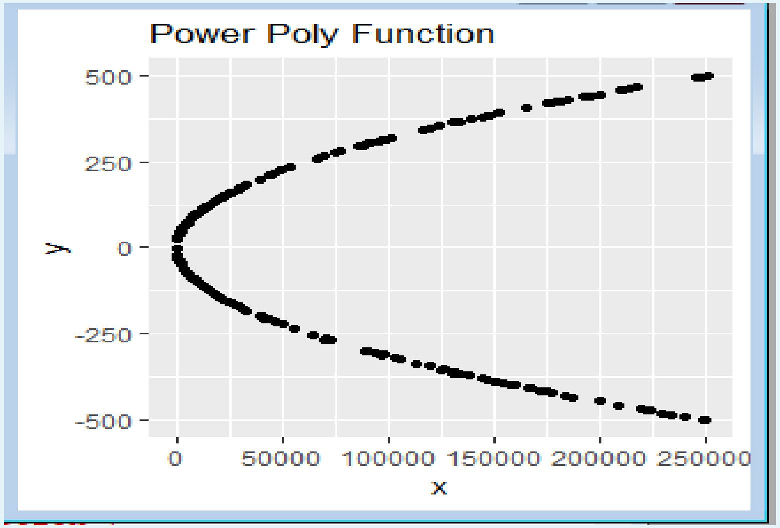
*dfyxsq<-data.frame(y=Y,x=expt_x*expt_x)#power fun, poly*

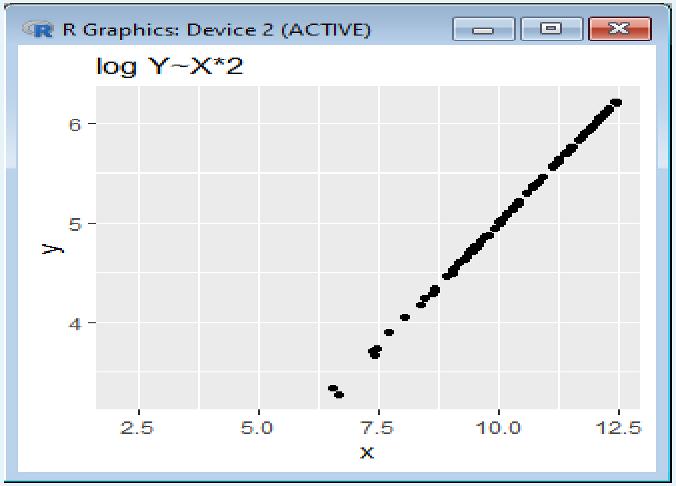
logdfyxsq<-log(dfyxsq)#log transformation

Trivial plots with ggplot

if (!require(ggplot2)) require(ggplot2)
ggplot(dfyx,aes(y=y,x=x))+geom_point()
+ggtitle("Linear Function")
ggplot(dfyxsq,aes(y=y,x=x))+geom_point()
+ggtitle("Power Poly Function")







ggplot(log(dfyxsq),aes(y=y,x=x))+geom_point()+ggtitle("log Y~X*2")

Let us take stock

We have examined anscombe one of many preloaded datasets in R
We used str, dim and head – certain helper functions to examine data.frames.
We used apply to compute sd and mean columnwise
We plotted some trivial charts
We generated data using the formula

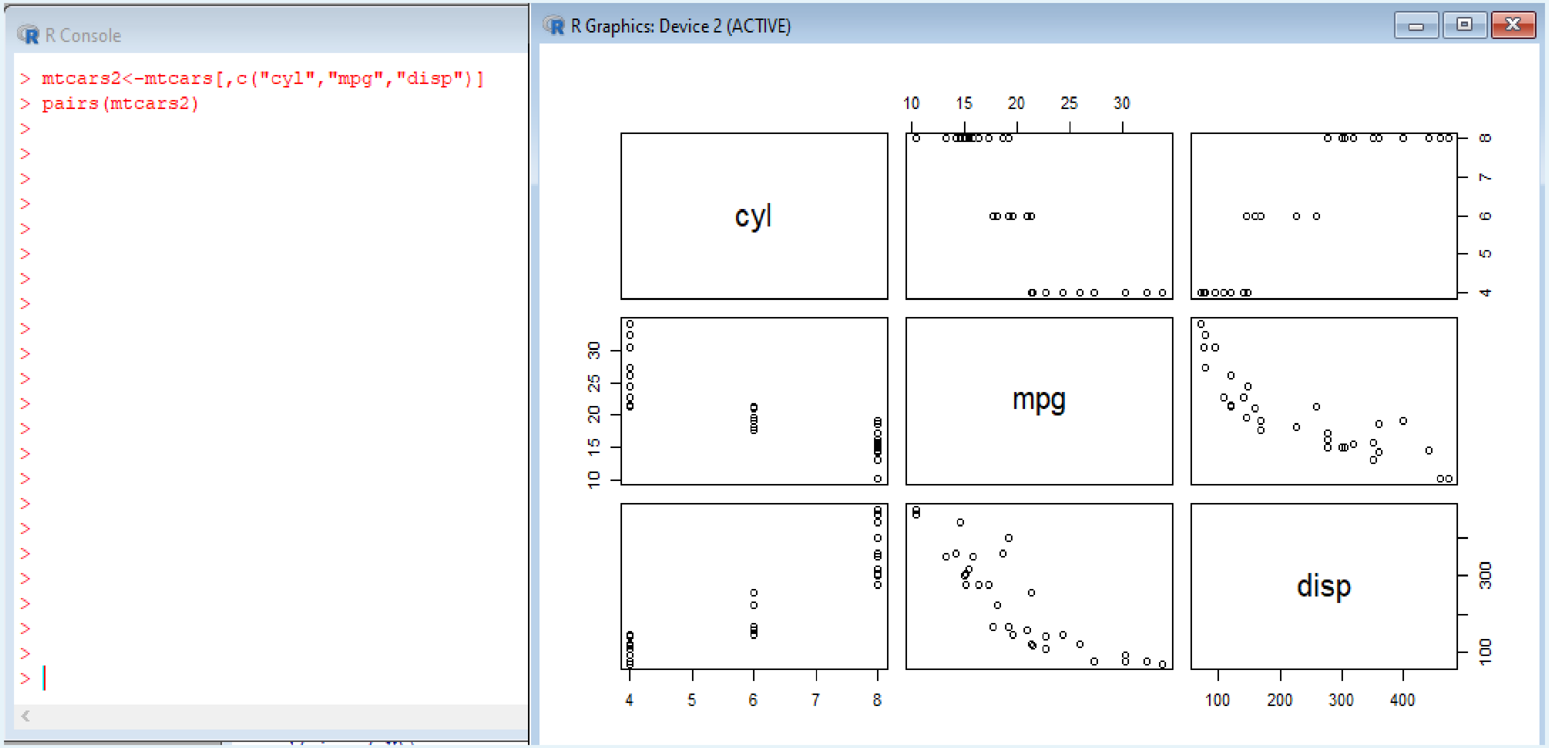
$y=x+2*sin(1.5*x)+N(0,0.2)$

Squared x to make it a polynomial
And then took the log to make a linear equation
We plotted each case

Now we move on to loading data from URLs, external files
Along the way we will learn to use preliminary data cleaning Tasks

EDA:pairs

mtcars2<-mtcars[,c("cyl","mpg","disp")]
pairs(mtcars2)



Sourcing data from the net

wine<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",sep=",",head=F)
head(wine)

```
> wine<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/
wine.data",sep=",",head=F)
> head(wine)
  V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
1  1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
2  1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
3  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
4  1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
5  1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
6  1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
```

plot(wine$V4, wine$V5)
text(wine$V4, wine$V5,wine$V1)
dfx<-data.frame(avg=unlist(apply(wine,2,mean)),sd=unlist(apply(wine,2,sd)), xx=names(wine))

dfx2<-
cbind(dfx,high=unlist(apply(wine,2,max)),low=unlist(apply(wine,2,min)))

As we can see the features are not comparable...

More operation on data.frames twh

TheWorkHorse of data science in R

```
> dfx2<-cbind(dfx,high=unlist(apply(wine,2,max)),low=unlist(apply(wine,2,min)))
> dfx2
            avg          sd   x    high    low
V1    1.9382022   0.7750350  V1    3.00   1.00
V2   13.0006180   0.8118265  V2   14.83  11.03
V3    2.3363483   1.1171461  V3    5.80   0.74
V4    2.3665169   0.2743440  V4    3.23   1.36
V5   19.4949438   3.3395638  V5   30.00  10.60
V6   99.7415730  14.2824835  V6  162.00  70.00
V7    2.2951124   0.6258510  V7    3.88   0.98
V8    2.0292697   0.9988587  V8    5.08   0.34
V9    0.3618539   0.1244533  V9    0.66   0.13
V10   1.5908989   0.5723589 V10    3.58   0.41
V11   5.0580899   2.3182859 V11   13.00   1.28
V12   0.9574494   0.2285716 V12    1.71   0.48
V13   2.6116854   0.7099904 V13    4.00   1.27
V14 746.8932584 314.9074743 V14 1680.00 278.00
```

sd is the standard deviation and V2,V5,V6,V11,V14 are much different than the others. Range has defined by high/low is wildly varying.

The effect due to changes in V1 is very likely to be masked by changes in these variables. We must normalize them – aka scaling.

scaling

```
> scaled<-apply(wine,2,FUN=function(v){(v-mean(v))/sd(v)})
> dim(scaled)==dim(wine)
```

scaleddfx<-data.frame(max=unlist(apply(scaled,2,max)), min=unlist(apply(scaled,2,min)))

```
> scaleddfx
         max       min
V1  1.370000 -1.210529
V2  2.253415 -2.427388
V3  3.100446 -1.428952
V4  3.147447 -3.668813
V5  3.145637 -2.663505
V6  4.359076 -2.082381
V7  2.532372 -2.101318
V8  3.054216 -1.691200
V9  2.395645 -1.862979
V10 3.475269 -2.063214
V11 3.425768 -1.629691
V12 3.292407 -2.088840
V13 1.955399 -1.889723
V14 2.963114 -1.488987
```

```
> dfx2[,c("high","low")]
       high    low
V1     3.00   1.00
V2    14.83  11.03
V3     5.80   0.74
V4     3.23   1.36
V5    30.00  10.60
V6   162.00  70.00
V7     3.88   0.98
V8     5.08   0.34
V9     0.66   0.13
V10    3.58   0.41
V11   13.00   1.28
V12    1.71   0.48
V13    4.00   1.27
V14 1680.00 278.00
```

Range is comparable.

Effect of Scaling

scaleddfx<-data.frame(max=unlist(apply(scaled,2,max)),
min=unlist(apply(scaled,2,min)))
scaledsd<-apply(scaled,2,sd)
(scaledavg<-apply(scaled,2,mean))
round((scaledavg<-apply(scaled,2,mean)),1)

```
> scaledsd<-apply(scaled,2,sd)
> scaledsd
 V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1 
> (scaledavg<-apply(scaled,2,mean))
           V1            V2            V3            V4            V5 
 8.194132e-17 -8.594093e-16 -6.734236e-17  8.046486e-16 -7.683704e-17 
           V6            V7            V8            V9           V10 
-4.095117e-17 -1.391677e-17  6.947239e-17 -1.041614e-16 -1.287594e-16 
          V11           V12           V13           V14 
 3.675080e-17  2.100477e-16  3.009648e-16 -1.037131e-16 
> round((scaledavg<-apply(scaled,2,mean)),1)
 V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 
  0   0   0   0   0   0   0   0   0   0   0   0   0   0 
```

Every variable has equal influence

Sourcing data from console

```
df <- read.table(header=TRUE, text='
 cond yval
        A 2
        B 2.5
        C 1.6
        ')

# Three variables
df2 <- read.table(header=TRUE, text='
        cond1 cond2 yval
        A     I 2
        A     J 2.5
        A     K 1.6
        B     I 2.2
        B     J 2.4
        B     K 1.2
        C     I 1.7
        C     J 2.3
        C     K 1.9
        ')
```

Preparing data for Analysis

Iris – another preloaded dataset

```
> iris[20:30,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
20          5.1         3.8          1.5         0.3  setosa
21          5.4         3.4          1.7         0.2  setosa
22          5.1         3.7          1.5         0.4  setosa
23          4.6         3.6          1.0         0.2  setosa
24          5.1         3.3          1.7         0.5  setosa
25          4.8         3.4          1.9         0.2  setosa
26          5.0         3.0          1.6         0.2  setosa
27          5.0         3.4          1.6         0.4  setosa
28          5.2         3.5          1.5         0.2  setosa
29          5.2         3.4          1.4         0.2  setosa
30          4.7         3.2          1.6         0.2  setosa
> dim(iris)
[1] 150   5
> iris[50:55,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
50          5.0         3.3          1.4         0.2     setosa
51          7.0         3.2          4.7         1.4 versicolor
52          6.4         3.2          4.5         1.5 versicolor
53          6.9         3.1          4.9         1.5 versicolor
54          5.5         2.3          4.0         1.3 versicolor
55          6.5         2.8          4.6         1.5 versicolor
> 
```

```
> table(iris$Species)

    setosa versicolor  virginica 
        50         50         50 
```

Generating representative sample

```
> iris[20:30,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
20          5.1         3.8          1.5         0.3  setosa
21          5.4         3.4          1.7         0.2  setosa
22          5.1         3.7          1.5         0.4  setosa
23          4.6         3.6          1.0         0.2  setosa
24          5.1         3.3          1.7         0.5  setosa
25          4.8         3.4          1.9         0.2  setosa
```

Is not representative

iris[sample(1:nrow(iris),5,replace=F),]

```
> iris[sample(1:nrow(iris),5,replace=F),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
65           5.6         2.9          3.6         1.3 versicolor
123          7.7         2.8          6.7         2.0  virginica
110          7.2         3.6          6.1         2.5  virginica
80           5.7         2.6          3.5         1.0 versicolor
71           5.9         3.2          4.8         1.8 versicolor
> tridx<-sample(nrow(iris),110,replace=F)
> tridx
  [1]  81  79 140 106 116  45 103  78  71 100  25  23  26 136  27 119 130  11  51   8  87
 [22]  68 149  93 113  19 145  53 111   1  62 142  34  59 132  92 108  24  35  98  74  77
 [43]  70  99  95  44  56   9  66  75  30  28 144 148  67 128  76  21 146  64 141  43 125
 [64]  32  52  20  69  96 124  60  82  61 150   4  88   2 139  73  14 122  16 126   3  49
 [85]  29 115 114 137 134 133 138  38  46  37  58  84  22  15  97 135  13 121 118 104  50
[106] 131  83 101 105  63
> trainset<-iris[tridx,]
> testset<-iris[-tridx,]
```

Iris into train/test set

```
> trainset<-iris[tridx,]
> testset<-iris[-tridx,]
> dim(testset)
[1] 40  5
> dim(trainset)
[1] 110   5
> head(trainset)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
81           5.5         2.4          3.8         1.1 versicolor
79           6.0         2.9          4.5         1.5 versicolor
140          6.9         3.1          5.4         2.1  virginica
106          7.6         3.0          6.6         2.1  virginica
116          6.4         3.2          5.3         2.3  virginica
45           5.1         3.8          1.9         0.4     setosa
> head(testset)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
10          4.9         3.1          1.5         0.1  setosa
12          4.8         3.4          1.6         0.2  setosa
17          5.4         3.9          1.3         0.4  setosa
```

We can now split datasets

```
> tridx<-sample(nrow(iris),110,replace=F)
> tridx
  [1]  81  79 140 106 116  45 103  78  71 100  25  23  26 136  27 119 130  11  51   8  87
 [22]  68 149  93 113  19 145  53 111   1  62 142  34  59 132  92 108  24  35  98  74  77
 [43]  70  99  95  44  56   9  66  75  30  28 144 148  67 128  76  21 146  64 141  43 125
 [64]  32  52  20  69  96 124  60  82  61 150   4  88   2 139  73  14 122  16 126   3  49
 [85]  29 115 114 137 134 133 138  38  46  37  58  84  22  15  97 135  13 121 118 104  50
[106] 131  83 101 105  63
> trainset<-iris[tridx,]
> testset<-iris[-tridx,]
> dim(testset)
[1] 40  5
> dim(trainset)
[1] 110   5
> head(trainset)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
81           5.5         2.4          3.8         1.1 versicolor
79           6.0         2.9          4.5         1.5 versicolor
140          6.9         3.1          5.4         2.1  virginica
106          7.6         3.0          6.6         2.1  virginica
116          6.4         3.2          5.3         2.3  virginica
45           5.1         3.8          1.9         0.4     setosa
> head(testset)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
10          4.9         3.1          1.5         0.1  setosa
12          4.8         3.4          1.6         0.2  setosa
17          5.4         3.9          1.3         0.4  setosa
> 
```

Spliting into 10 Folds

```
> seq(1,nrow(iris),15)
 [1]   1  16  31  46  61  76  91 106 121 136
> starts<-seq(1,nrow(iris),15)
> ends<-seq(15,nrow(iris),15)
> ends
 [1]  15  30  45  60  75  90 105 120 135 150
> sample3<-iris[starts[3]:ends[3],]
> sample3
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
31          4.8         3.1          1.6         0.2  setosa
32          5.4         3.4          1.5         0.4  setosa
33          5.2         4.1          1.5         0.1  setosa
34          5.5         4.2          1.4         0.2  setosa
35          4.9         3.1          1.5         0.2  setosa
36          5.0         3.2          1.2         0.2  setosa
37          5.5         3.5          1.3         0.2  setosa
38          4.9         3.6          1.4         0.1  setosa
39          4.4         3.0          1.3         0.2  setosa
40          5.1         3.4          1.5         0.2  setosa
41          5.0         3.5          1.3         0.3  setosa
42          4.5         2.3          1.3         0.3  setosa
43          4.4         3.2          1.3         0.2  setosa
44          5.0         3.5          1.6         0.6  setosa
45          5.1         3.8          1.9         0.4  setosa
```

But this is not random

3rd Fold

```
> iris2<-sample(nrow(iris),nrow(iris),replace=F)
> iris2
  [1]  63  57  27  30 136  43  76  75  94   3 114  91  46  89  93 141  36  77  64 106  90
 [22]  29 132  12   1  81  21  17  72  37 124 120  50  74 109 134  13 118 148  73  24  62
 [43]  42  40  69 110  60 116  19  11  49   2  97  38  65  18 102   7  95  15  80 140  34
 [64]  84  68 103  33  41 135 127   9  99  39  96  20 101  44 126 147  82 131 139  53 115
 [85] 137  31  92 125 112  59  66  16 108   5  56  23  32  78 111  28 129   4 149 123  88
[106]  87  52 105  83  45 128 146  71  86 133  14  26   6 100  98  67  10  55   8  35 144
[127] 122 117 107  47 145  61 121 130  70  51 143  79 138 104 150 113  48  54  58  85 142
[148] 119  22  25
```

```
> iris2
  [1]  63  57  27  30 136  43  76  75  94   3 114  91  46  89  93 141  36  77  64 106  90
 [22]  29 132  12   1  81  21  17  72  37 124 120  50  74 109 134  13 118 148  73  24  62
 [43]  42  40  69 110  60 116  19  11  49   2  97  38  65  18 102   7  95  15  80 140  34
 [64]  84  68 103  33  41 135 127   9  99  39  96  20 101  44 126 147  82 131 139  53 115
 [85] 137  31  92 125 112  59  66  16 108   5  56  23  32  78 111  28 129   4 149 123  88
[106]  87  52 105  83  45 128 146  71  86 133  14  26   6 100  98  67  10  55   8  35 144
[127] 122 117 107  47 145  61 121 130  70  51 143  79 138 104 150 113  48  54  58  85 142
[148] 119  22  25
> random_sample3<-iris2[starts[3]:ends[3]]
> iris_random_sample3<-iris[random_sample3,]
> iris_random_sample3
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
124          6.3         2.7          4.9         1.8  virginica
120          6.0         2.2          5.0         1.5  virginica
50           5.0         3.3          1.4         0.2     setosa
74           6.1         2.8          4.7         1.2 versicolor
109          6.7         2.5          5.8         1.8  virginica
134          6.3         2.8          5.1         1.5  virginica
13           4.8         3.0          1.4         0.1     setosa
118          7.7         3.8          6.7         2.2  virginica
148          6.5         3.0          5.2         2.0  virginica
73           6.3         2.5          4.9         1.5 versicolor
24           5.1         3.3          1.7         0.5     setosa
62           5.9         3.0          4.2         1.5 versicolor
42           4.5         2.3          1.3         0.3     setosa
40           5.1         3.4          1.5         0.2     setosa
69           6.2         2.2          4.5         1.5 versicolor
```

What have we learned?

We can load data
- from network
- from file
- from console

We can plot
We can scale
We can split dataset

http://www.endmemo.com/program/R/
http://www.r-tutor.com/
http://www.r-bloggers.com/
https://stackoverflow.com/questions