Data mining & Machine Learning

CS 373
Purdue University

Dan Goldwasser
dgoldwas@purdue.edu

Today's Lecture

DATA is a big word

- *What does it actually mean? What can we expect to find we collect data?*
- *In the age of big-data, how can we quickly "summarize" it?*
  - ***Find patterns, identify noise, etc.***

How can we answer questions using data?

Data quality

- Examples of data quality problems:
  - Noise
  - Outliers
  - Missing values
  - Duplicate data

Other data preprocessing methods

- Sampling
- Attribute transformations
  - Discretization, distance calculations
  - **Feature construction**
- Dimensionality reduction and feature selection
- Recent trend: ***Representation Learning***

Data exploration and visualization

Exploratory data analysis

- Data analysis approach that employs a number of (mostly graphical) techniques to:
  - Maximize insight into data
  - Uncover underlying structure
  - Identify important variables
  - Detect outliers and anomalies
  - Test underlying modeling assumptions
  - Develop parsimonious models
  - **Generate hypotheses from data**

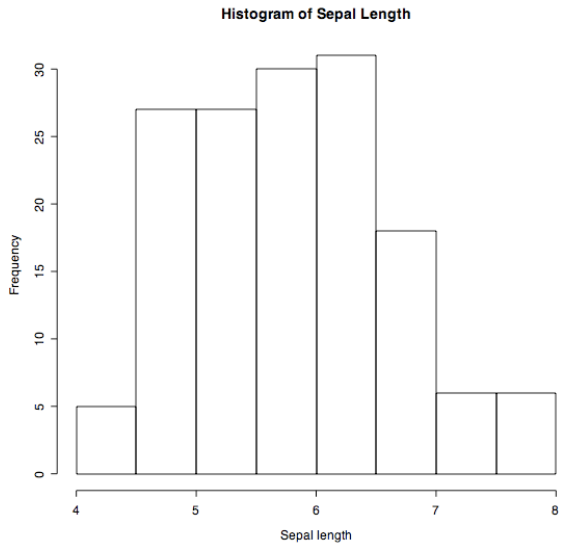
Visualizing/summarizing data

- **Low-dimensional data**
  - Summarizing data with simple statistics
  - Plotting raw data (1D, 2D, 3D)

- **Higher-dimensional data**
  - Principal component analysis
  - Multidimensional scaling

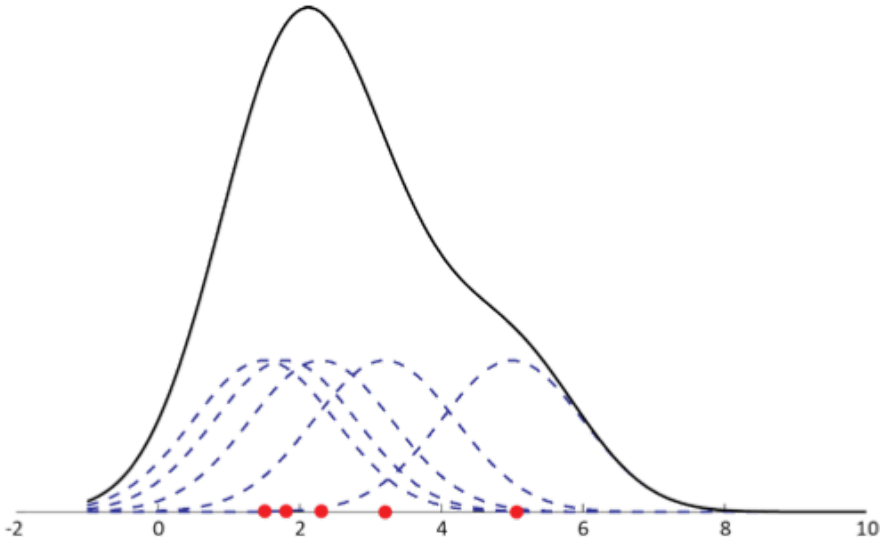
Histograms (1D)

- Most common plot for univariate data
- Split data range into equal-sized bins, count number of data points that fall into each bin
- **Graphically shows:**
  - Center (location)
  - Spread (scale)
  - Skew
  - Outliers
  - Multiple modes



Histogram limitations

- Histograms can be misleading for small datasets
  - Slight changes in the data or binning approach can result in different histograms
- **Solution**: *smoothed density plots*
  - Use kernel function to estimate density at each point x, pools information from neighboring points

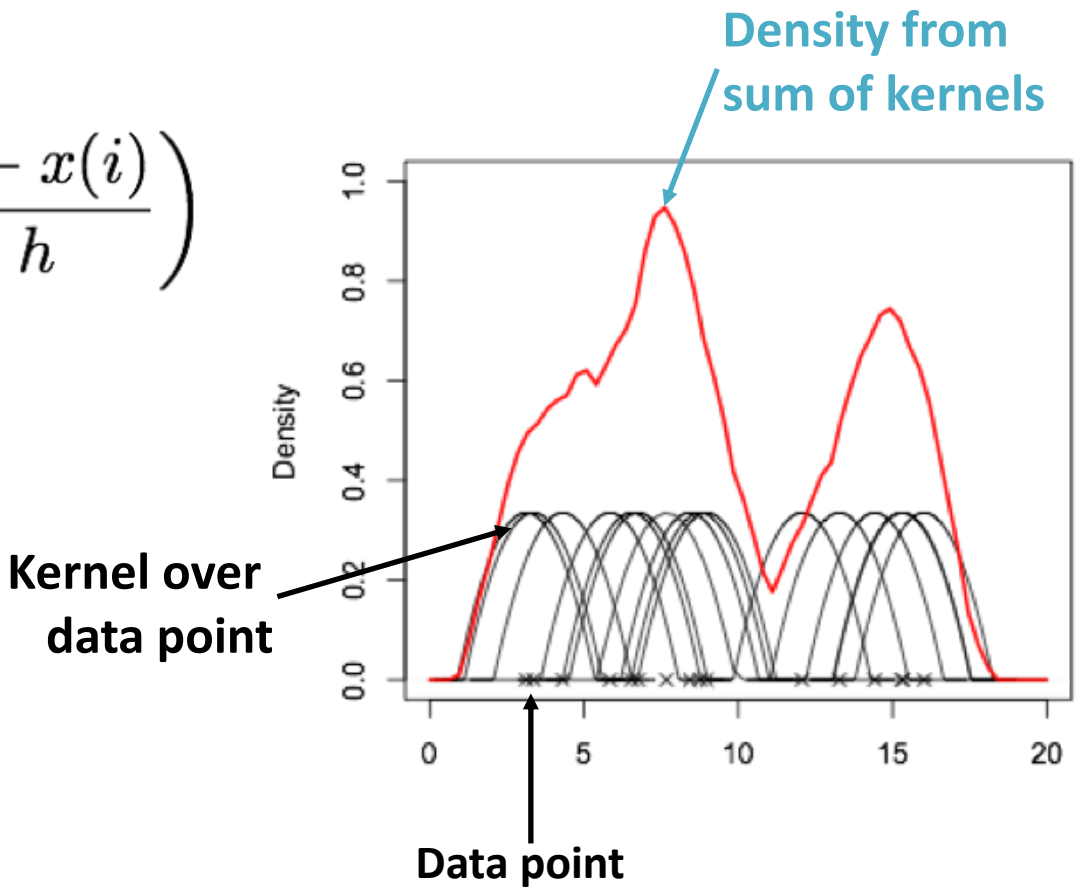


Density plots

- Estimated density is:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x(i)}{h}\right)$$

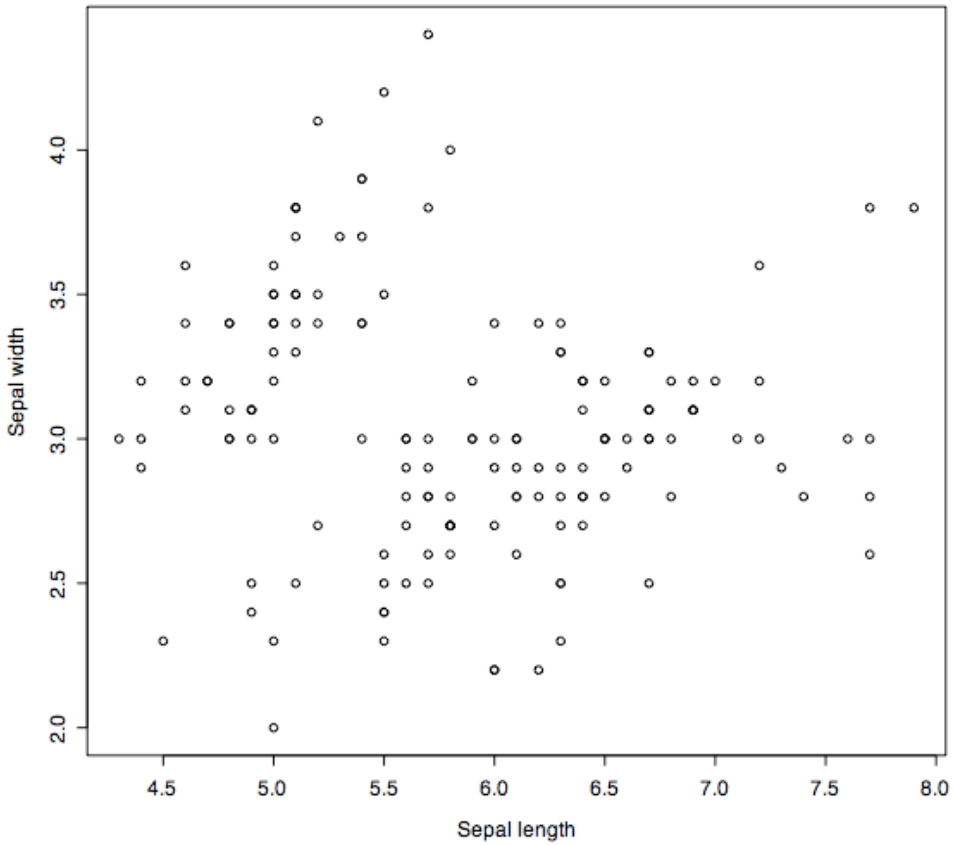
- **Two parameters:**
  - Kernel function K (e.g., Gaussian, Epanechnikov)
  - Bandwidth h



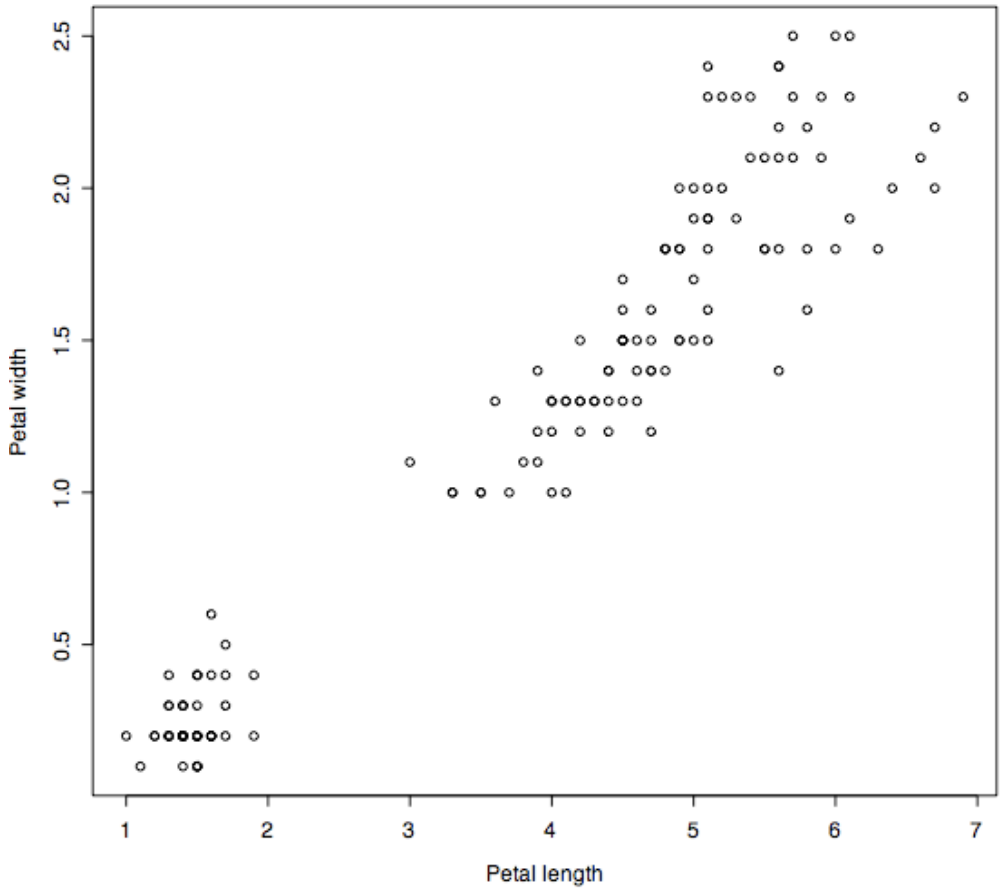
Scatter plot (2D)

- Most common plot for bivariate data
  - Horizontal X axis: the suspected **independent** variable
  - Vertical Y axis: the suspected **dependent** variable
- **Graphically shows:**
  - If X and Y are related
  - Linear or non-linear relationship
  - If the variation in Y depends on X
  - Outliers

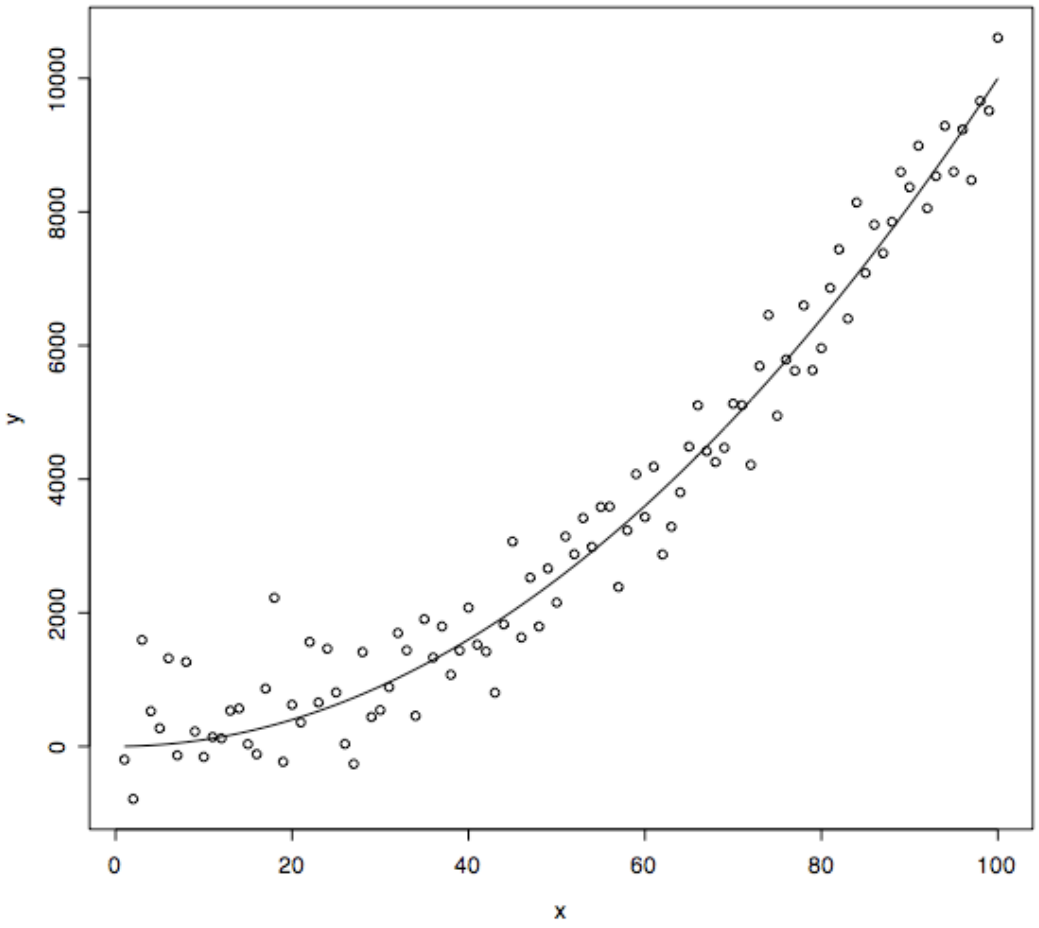
No relationship



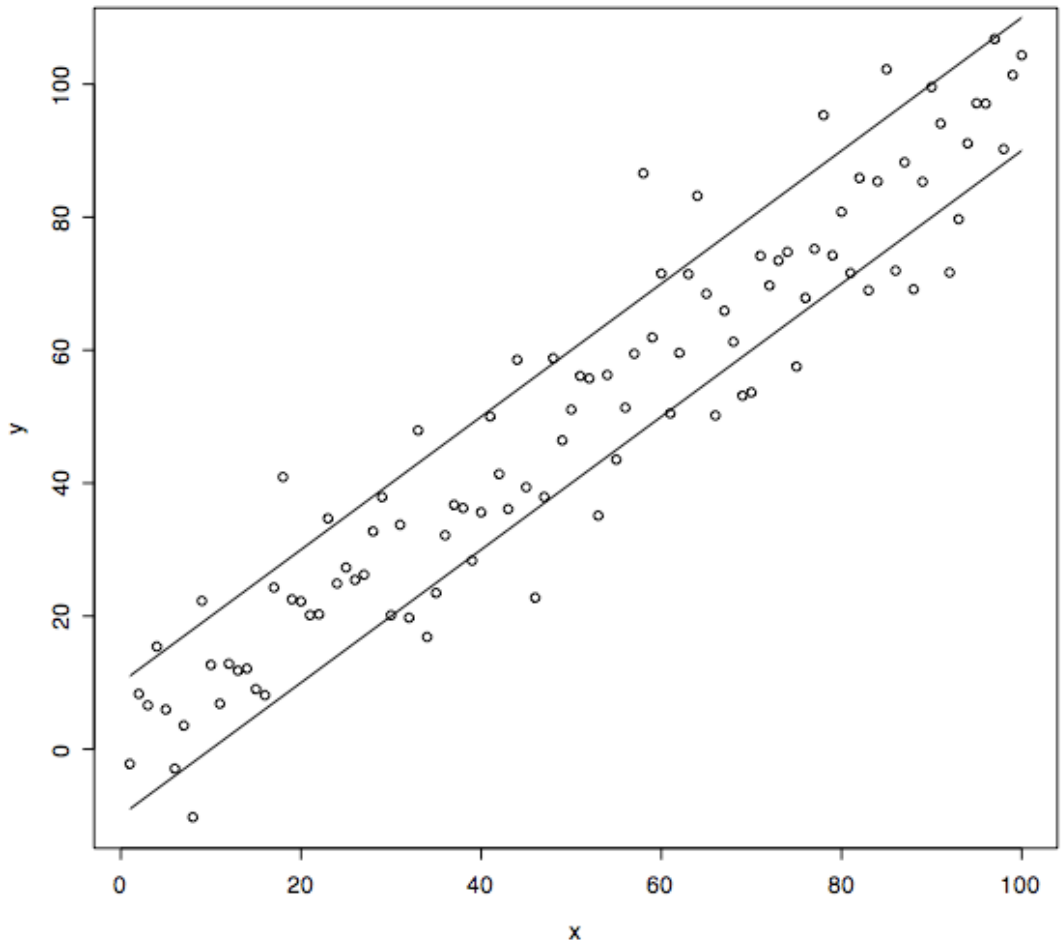
Linear relationship



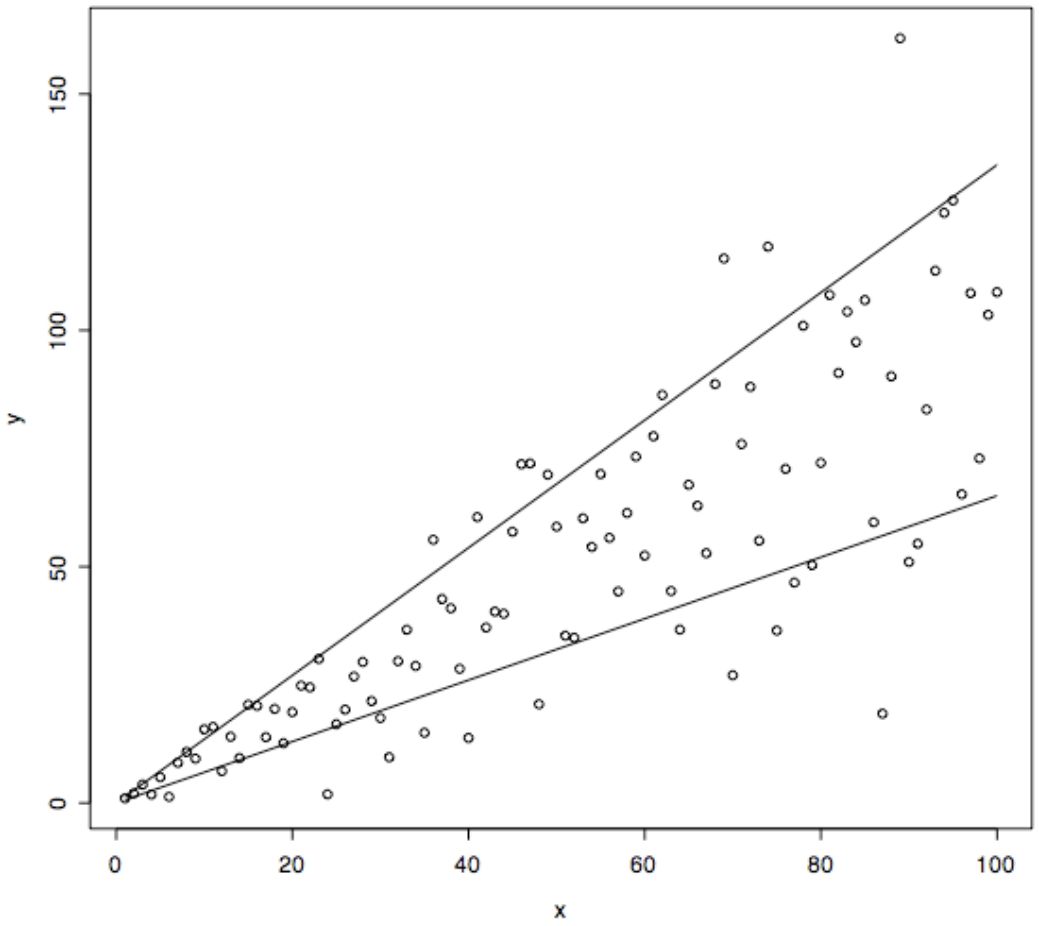
Non-linear relationship



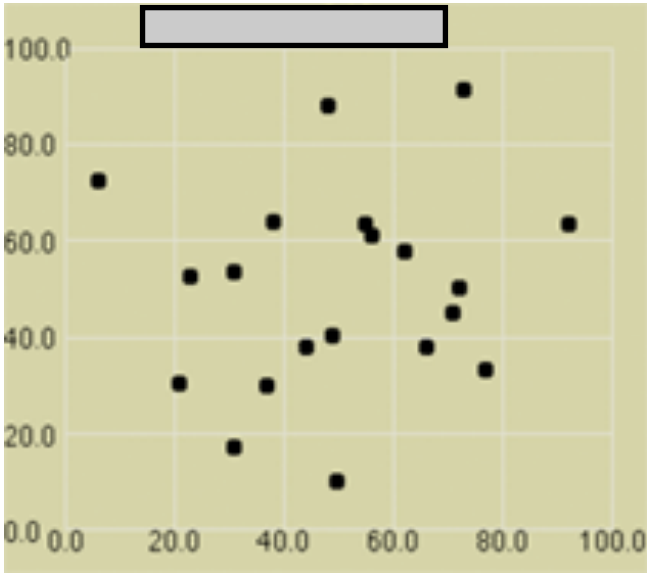
Homoskedastic

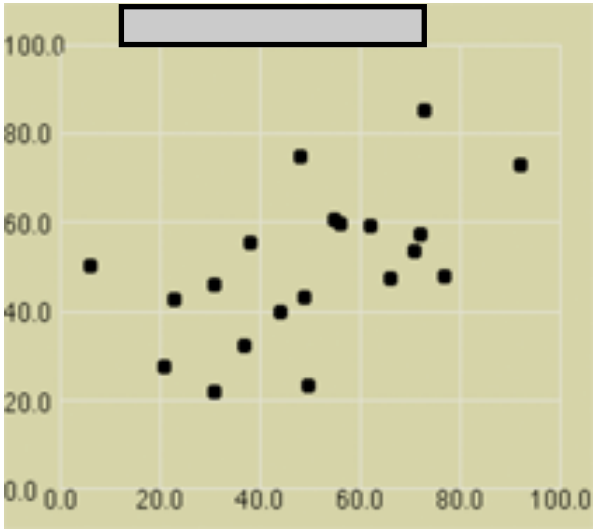


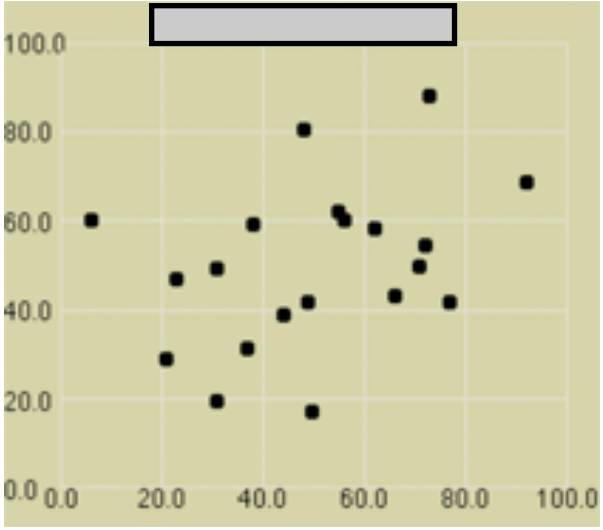
Heteroskedastic



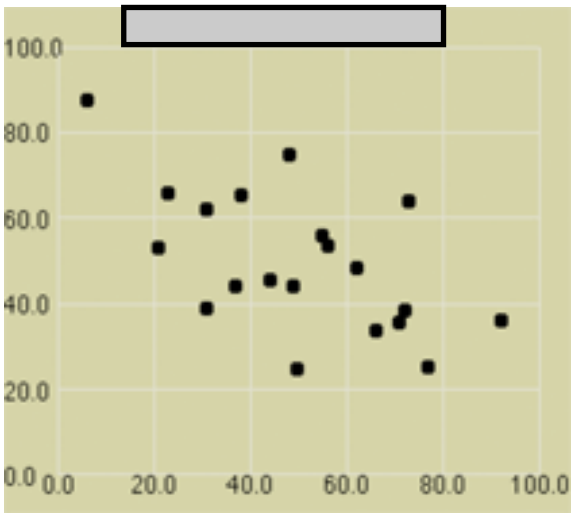
Which one of the plots shows the strongest/ weakest correlation?

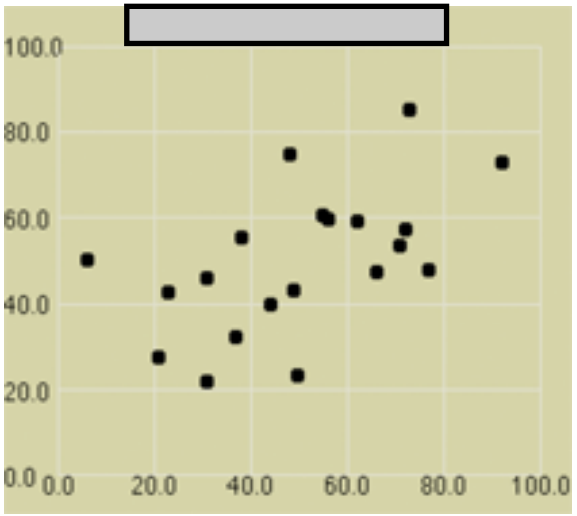


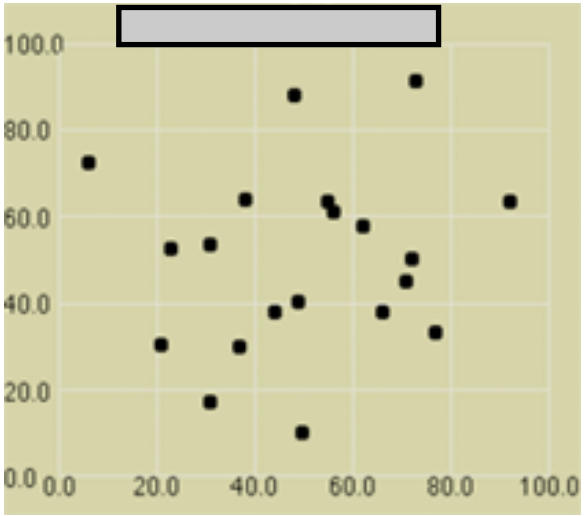




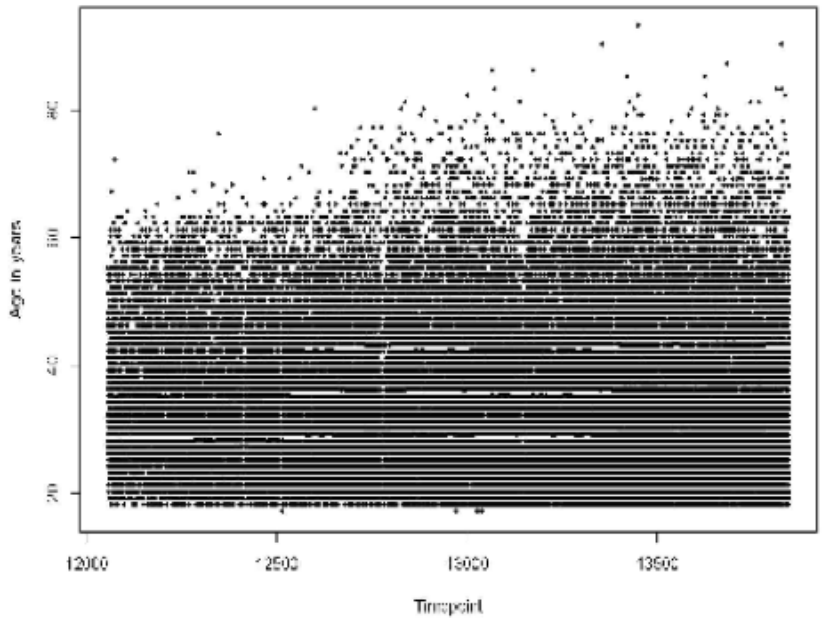
Which one of the plots describes a positive correlation?



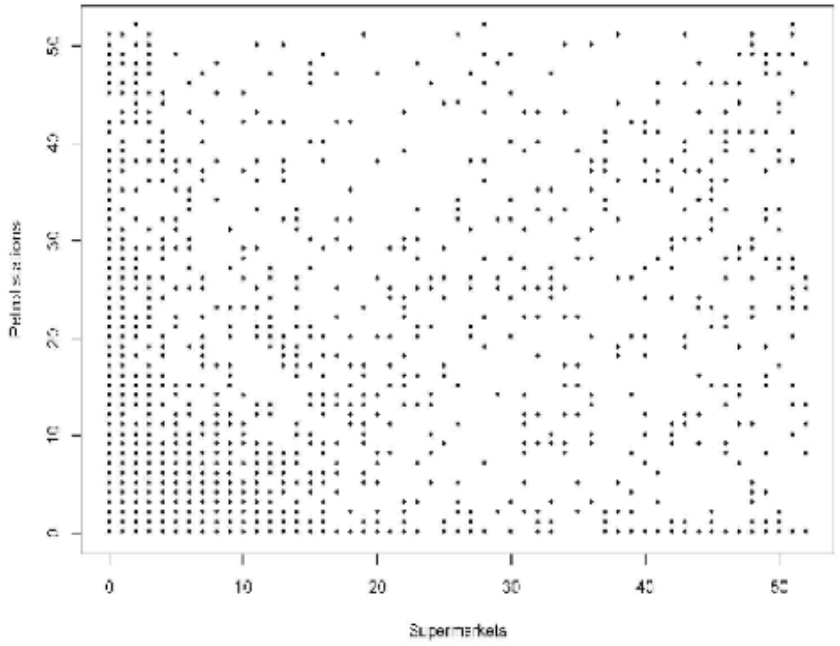




Scatterplot limitations

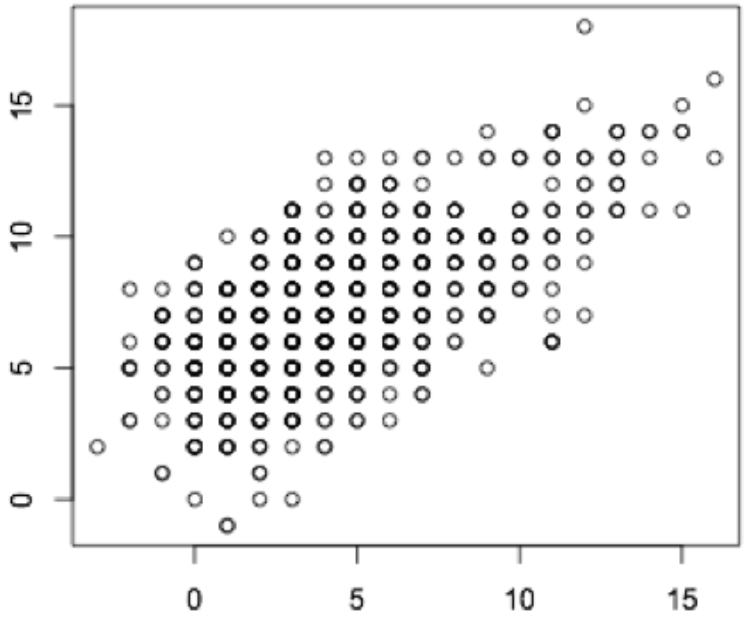


Too much data

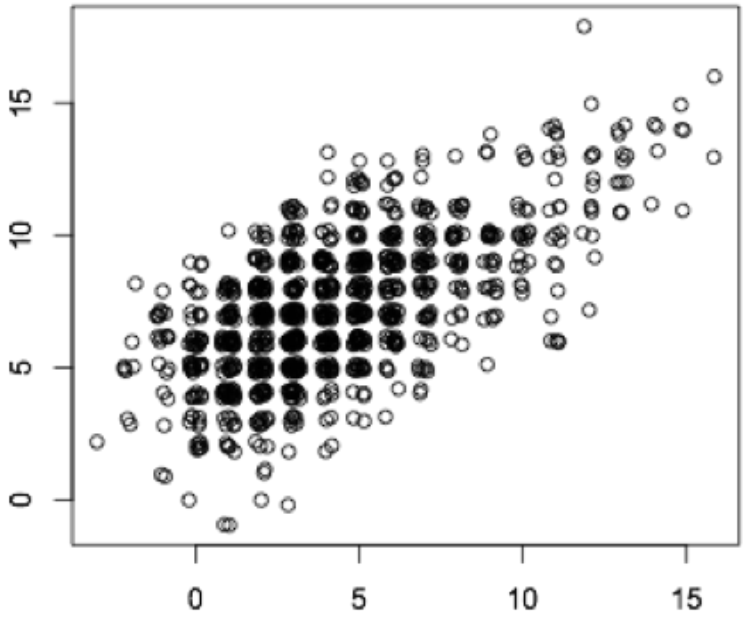


Overprinting

Scatterplot limitations

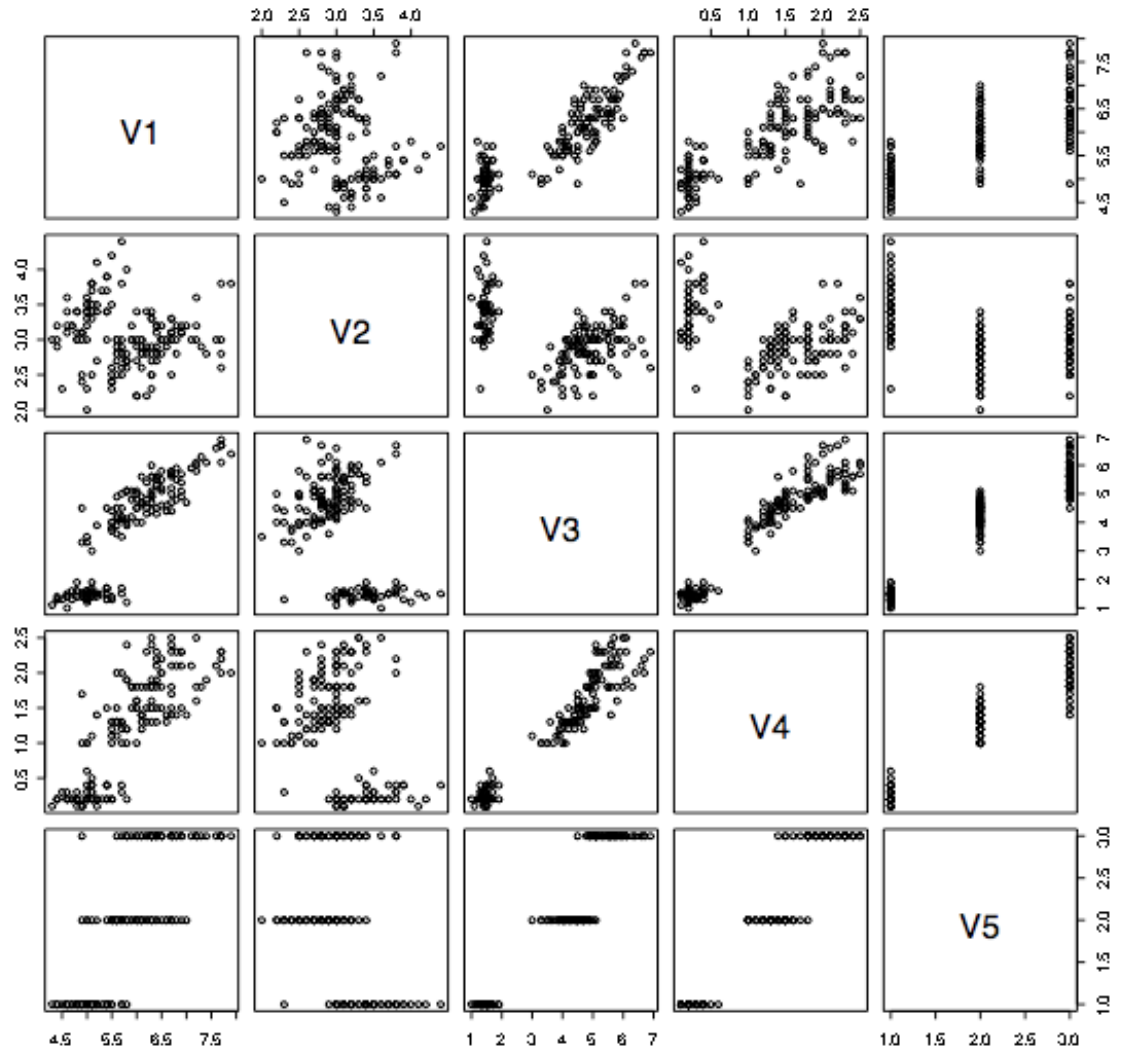


Overprinting



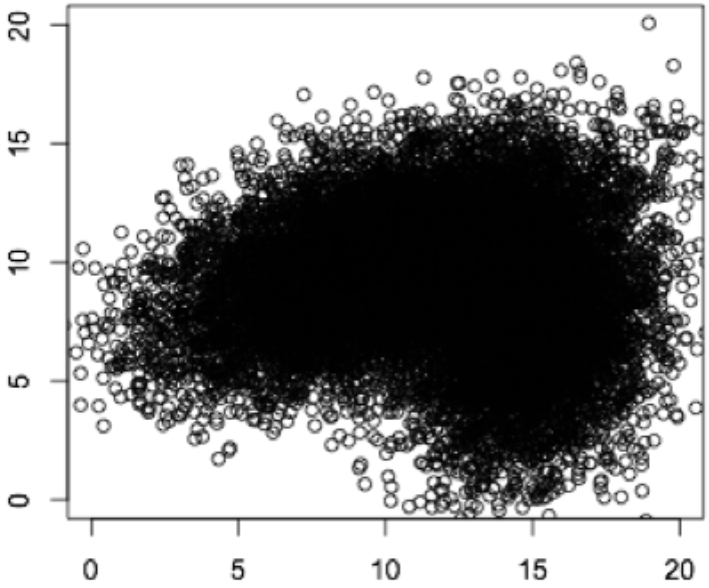
Solution: Jitter points

Scatterplot matrix

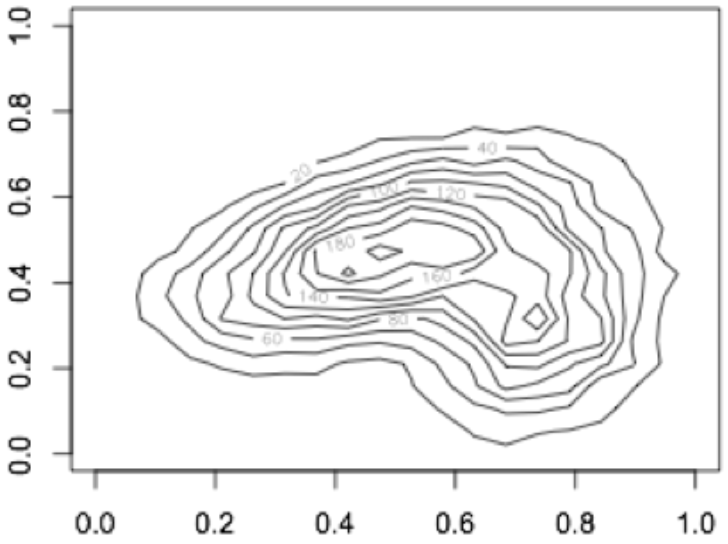


Contour plot (3D)

- Limitations of 2D scatterplot (e.g., when there is too much data to discern relationship)



- Solution: represent a 3D surface by plotting constant z slices (contours) in a 2D format



Introduction to R

- See: http://cran.r-project.org/

```
You can use R interactively in the console.
Just start typing commands.

x = 5     ### assign x the value 5
x            ### print x
[1] 5
print(x) ### another way to print x
[1] 5
x <- 5       ### you can also use <- for assignments
y = "Hello there"
y = sqrt(10)
z = x + y
```

Scalars are treated by R as vectors of length 1. They print with a leading "[1]" to indicate they are the first element of a vector.

Vectors can be created using the c() command.
c() stands for concatenate.

Square brackets are used to get subsets of a vector. The colon is used for sequences.

```
x = 1:5                    ### the vector (1,2,3,4,5)
x = seq(1,5,length=5)      ### same thing
x[1] = 17
x[3:5] = 0
x
[1] 17  2  0  0  0
x[c(1,4,5)]
[1] 17  0  0
```

```
y = 1:10
x = c(5,4,3,2,1,5,4,3,2,1)
z = x + y    ### R carries out operations on vectors,
                        element by element
z
[1]  6  6  6  6  6 11 11 11 11 11
x == 2           ### This is a logical vector.
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
FALSE

x = c(5,4,3,2,1,5,4,3,2,1)
sort(x)      ### sorts the elements in the vector
[1] 1 1 2 2 3 3 4 4 5 5
rank(x)      ### returns the ranks of each element
[1] 9.5 7.5 5.5 3.5 1.5 9.5 7.5 5.5 3.5 1.5
```

```
junk = c(1, 2, 3, 4, 5, 0.5, 2, 6, 0, 1, 1, 0)
m = matrix(junk,ncol=3)
m
     [,1] [,2] [,3]
[1,]    1  5.0    0
[2,]    2  0.5    1
[3,]    3  2.0    1
[4,]    4  6.0    0
dim(m)       ### matrix dimensions
[1] 4 3
y = m[,1]    ### y is column 1 of m
x = m[2,]    ### x is row 2 of m
z = m[1,2]   ### z is entry [1,2]
t(m)            ### transpose of m
     [,1] [,2] [,3] [,4]
[1,]    1  2.0    3    4
[2,]    5  0.5    2    6
[3,]    0  1.0    1    0
```

If you have data in a file, you can read it into R using the read.table command.

Suppose file.txt looks like this:

```
2 4 17.2
3 8 12
3 3.4 19
2 52 101.2
1 1 3
```

```
a = read.table("file.txt")
```

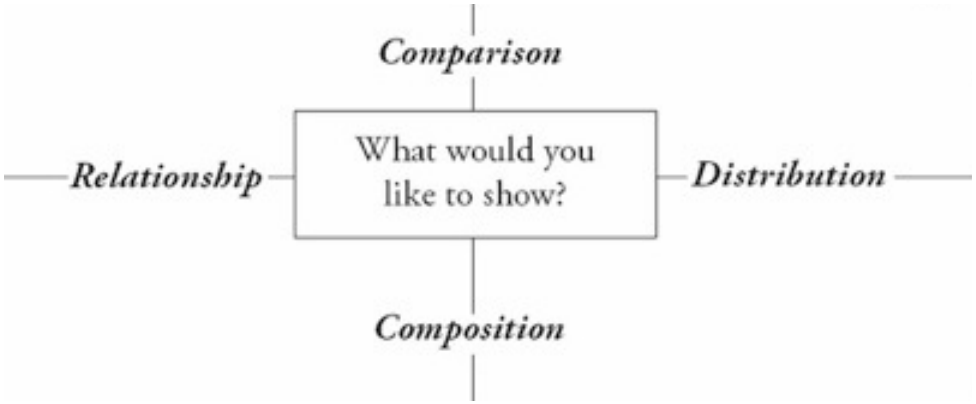
This places the data into a data frame, which is like a matrix but is more general. Each column can be a different type (character, numeric etc.)
Read the help file on data.frame and read.table for more information.
Note: Use help(xxxx) to get help on command xxxx.

```
x = runif(100,0,1)   ### generate 100 uniform random
                          numbers in range [0,1]
y = rnorm(10,0,1) ### generate 10 random Normals,
                          with mean 0, sd 1
mean(y)
[1] -0.4525719
median(y)
[1] -0.7323456
range(y)
[1] -1.979648  1.059751
max(y)
[1] 1.059751
min(y)
[1] -1.979648
sqrt(var(y))
[1] 1.051884
summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.9800 -1.0170 -0.7323 -0.4526  0.4933  1.0600
```

There are many options related to plotting. You control them with the par command, which stands for “plotting parameters.” Type help(par).

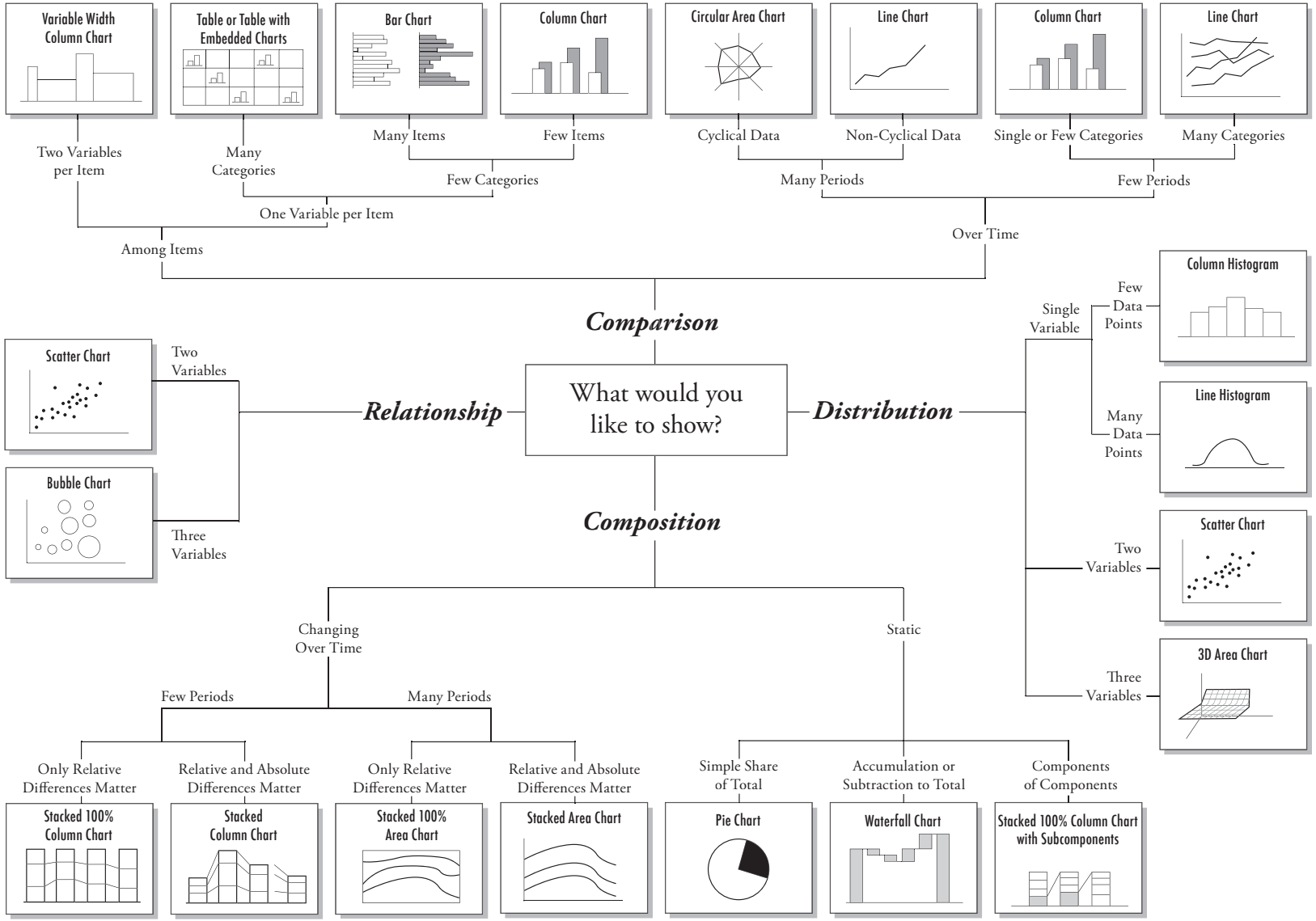
```
d <- read.table(file='iris.dat',sep=',')
pdf(file='example-plots.pdf',height=6,width=12)
                ### opens pdf file
par(mfrow=c(1,2),mar=c(3,3,1,1),cex=1.4,mgp=c(2,0.5,0),tc
k=-0.015)
                ### sets plot parameters
plot(density(d[,3]),main='petal length’)
                ### first plot
plot(d[,1],d[,2],xlab='sepal length',ylab='sepal width’)
                ### second plot
dev.off()   ### close pdf file
```

Note: You can also use R in Batch mode. To do this, store your R commands in a file, say, file.r.
In R type: source("file.r") to execute the commands.



http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html

Chart Suggestions—A Thought-Starter



http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html

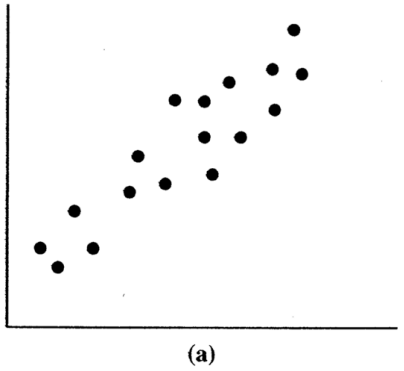
Covariance and correlation

Covariance

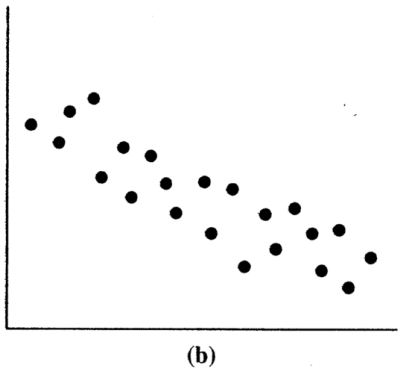
- *Measures how variables X and Y vary together*

$$COV(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})$$

  - Positive if large values of X are associated with large values of Y
  - Negative if large values of X are associated with small values of Y



(a)



(b)

Measures **linear** relationship

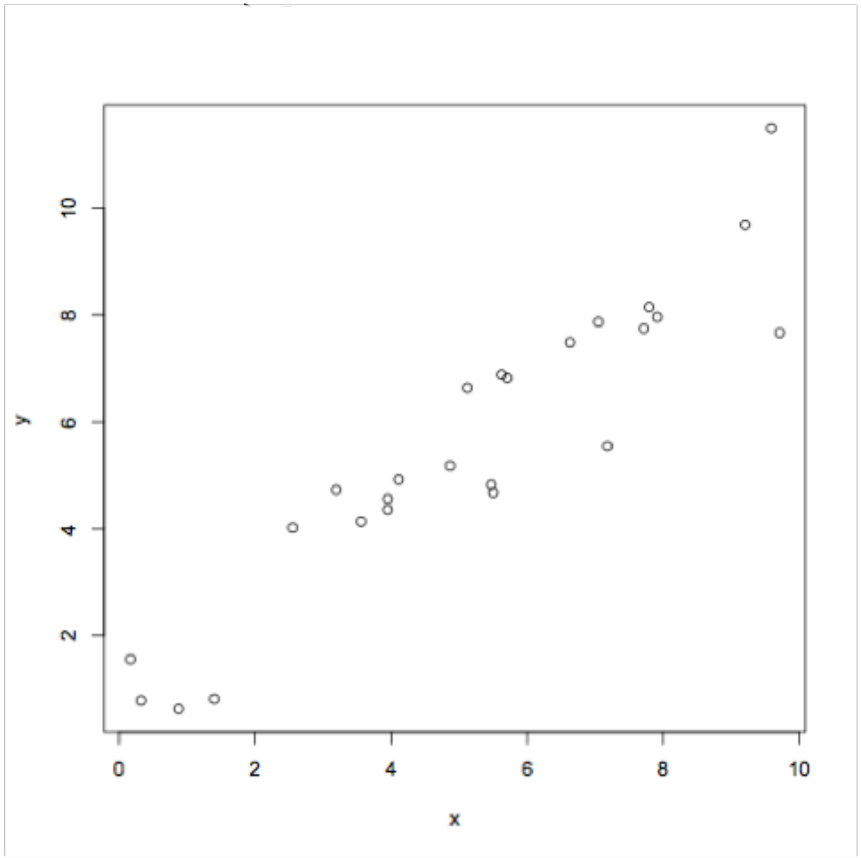
- **Covariance matrix (Σ)**
  - Symmetric matrix of covariances for p variables

Example

```
> ###create data
> x<-runif(25,0,10)
> y<-x+rnorm(25,0,1)
> z <- cbind(x,y)

> ###scatterplot
> plot(x,y)

> ###compute covariance
> cov(z)
          x         y
x 7.772214 7.197420
y 7.197420 7.586875
```



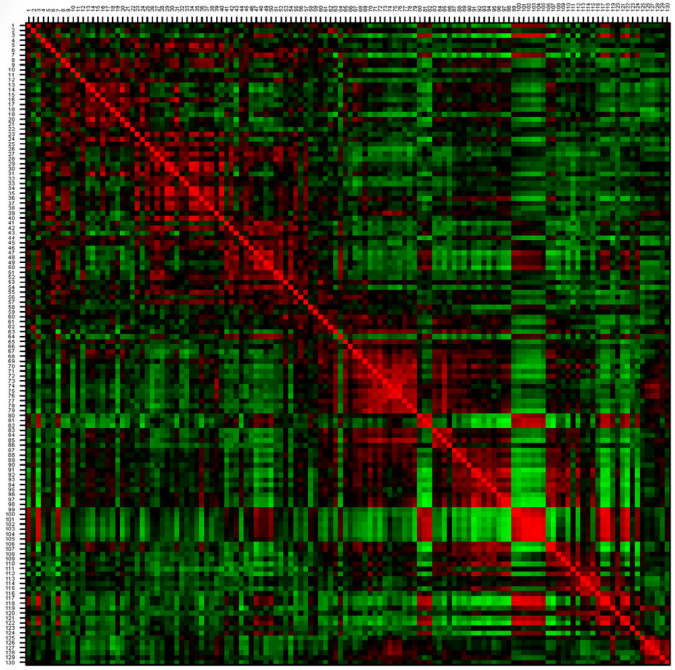

```
[24,] 3.9462735  4.5606035
[25,] 1.3967384  0.8176381
```

Correlation coefficient

- Covariance depends on ranges of X and Y
- *Correlation standardizes covariance by dividing through standard deviations*

$$\rho(x,y) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})}{\sigma_x \sigma_y}$$

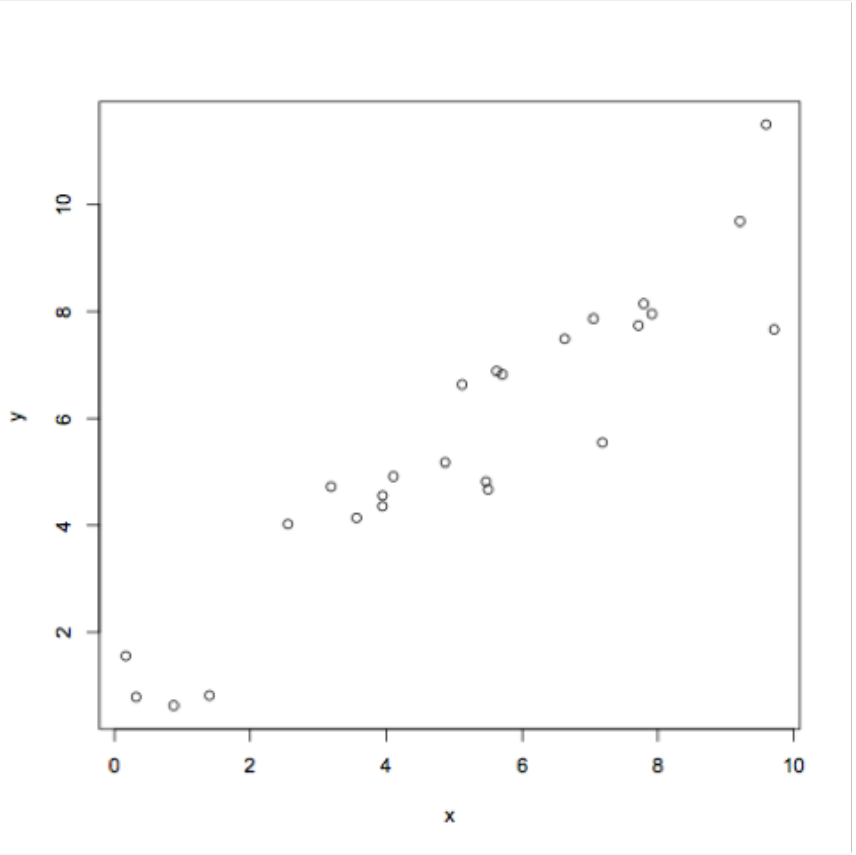
- **Correlation matrix**
  - Symmetric matrix of correlations for p variables
  - What values are on the diagonal?



Example (cont)

```
>###compute covariance
> cov(z)
          x         y
x 7.772214 7.197420
y 7.197420 7.586875

>###compute correlation
> cor(z)
           x         y
x 1.0000000 0.9372879
y 0.9372879 1.0000000
```



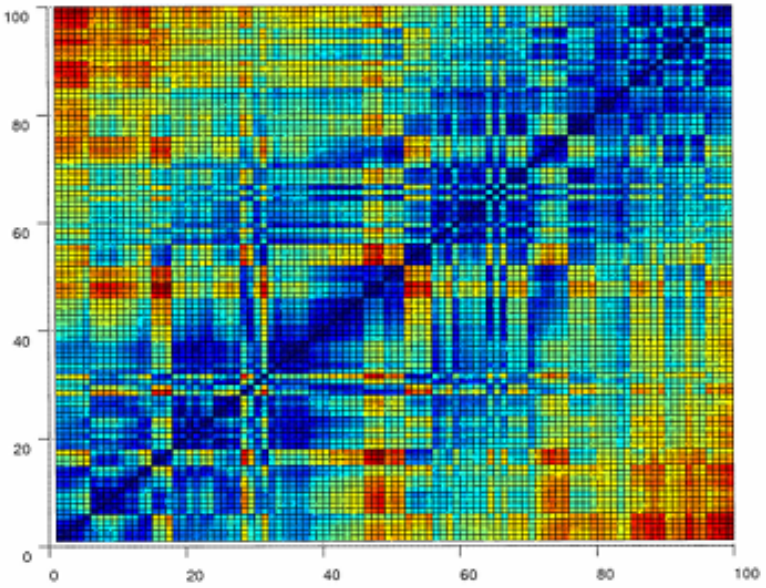
Distance measures

Distance measures

- If data objects have the same fixed set of numeric attributes, then the **data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute**

- Many data mining techniques then **use similarity/dissimilarity measures to characterize relationships between the instances**, e.g.,
  - Nearest-neighbor classification
  - Cluster analysis

- **Proximity**: general term to indicate similarity and dissimilarity
- **Distance**: dissimilarity only



Metric properties

- A **metric** *d(i,j)* is a dissimilarity measure that satisfies the following properties:
  - $d(i,j) \geq 0$ *for all i,j and* $d(i,j)=0$ *iff* $i=j$ **Positivity**
  - $d(i,j) = d(j,i)$ *for all i,j* **Symmetry**
  - $d(i,j) \leq d(i,k)+d(k,j)$ *for all i,j,k* **Triangle inequality**

Distance metrics

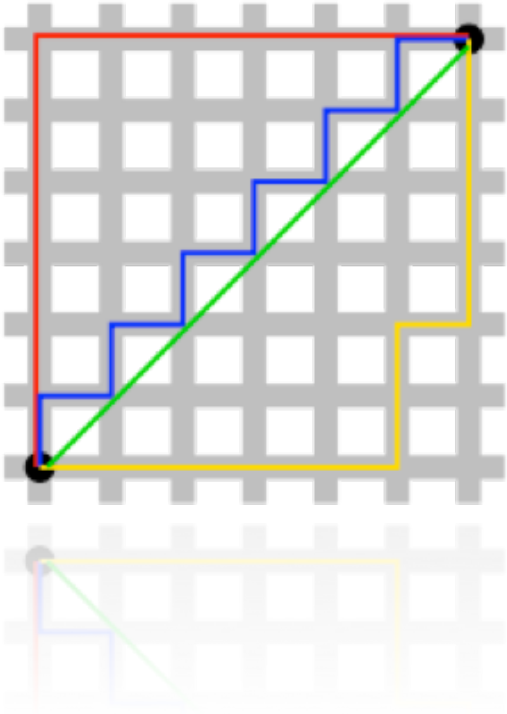
- **Manhattan distance (L1)**

$$d_M(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

- **Euclidean distance (L2)**

$$d_E(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

  - Most common metric
  - Assumes variables are commensurate



- **Weighted** Euclidean distance

$$d_{WE}(x, y) = \sqrt{\sum_{i=1}^{p} w_i (x_i - y_i)^2}$$

  - Can weight variables by relative importance

Standardization

- **Normalization**
  - Removes effect of scale
  - Divide each variable by its standard deviation
  - Weights all variables equally

$$x'_k = \frac{x_k - \bar{x}_k}{\hat{\sigma}_k}$$

subtract mean

divide by stdev

$$d'_E(x, y) = \sqrt{\sum_{i=1}^{p} (x'_i - y'_i)^2}$$

Correlation among variables

- Variables contribute independently to additive measure of distance
- May not be appropriate if variables are highly correlated
- Can standardize variables in a way that accounts for covariance

Diameter,
$Height_1$,
$Height_2$,
…,
$Height_{100}$

Mahalanobis distance

$$d_{MH}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$

pxp covariance matrix

- Automatically accounts for scaling
- Corrects for correlation between attributes
- Tradeoff:
  - Covariance matrix can be hard to estimate accurately
  - Memory and time complexity is quadratic rather than linear

Distance measures for binary data

- d(x,y) when items x and y are p-dimensional binary vectors
- Let n_{11} be the number of attributes where both items have value 1, etc.

$$n_{11} = \sum_{i}^{p} \mathbb{I}(x_i + y_i = 2)$$

	y=1	y=0
x=1	n_{11}	n_{01}
x=0	n_{10}	n_{00}

- **Hamming Distance:** $n_{10}+n_{01}$
- **Matching coefficient**
  - Similarity normalized by number of bits

$$d_{MC}(x,y) = \frac{n_{11}+n_{00}}{n_{11}+n_{00}+n_{10}+n_{01}}$$

- **Jaccard coefficient**
  - If we don't care about matches on zeros

$$d_{JC}(x,y) = \frac{n_{11}}{n_{11}+n_{00}+n_{10}+n_{01}}$$