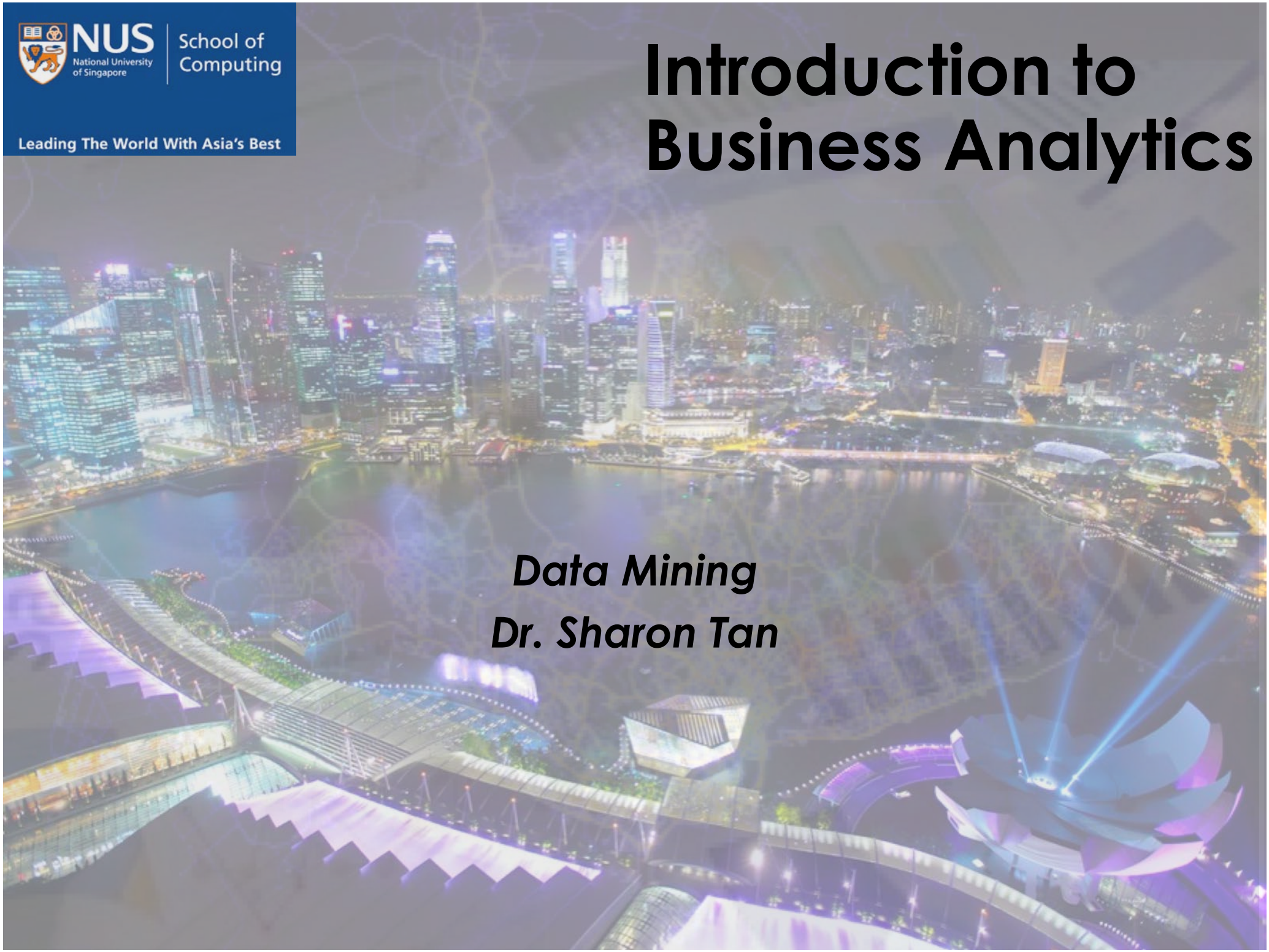
NUS National University of Singapore | School of Computing

Leading The World With Asia's Best

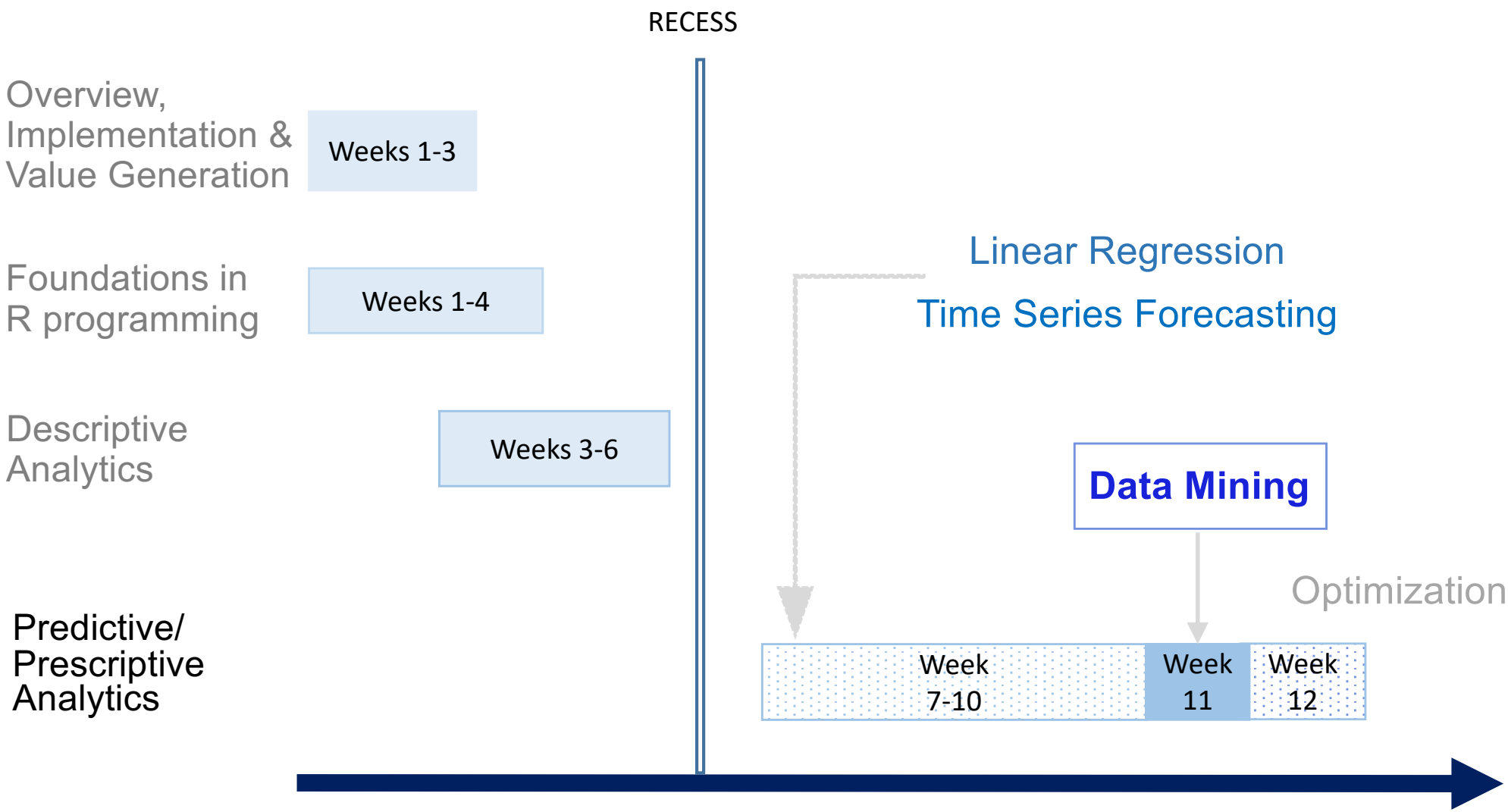
Introduction to Business Analytics



Data Mining

Dr. Sharon Tan

Course Map


RECESS
Overview, Implementation & Value Generation
Weeks 1-3
Foundations in R programming
Weeks 1-4
Descriptive Analytics
Weeks 3-6
Predictive/ Prescriptive Analytics
Linear Regression
Time Series Forecasting
Data Mining
Optimization
Week 7-10
Week 11
Week 12

Learning outcomes

- Understand the concept of data dimensionality reduction; Be able to apply principal component analysis to summarize information in a large dataset.
- Understand the similarity and difference between model selection and data dimensionality reduction.
- Understand basic unsupervised clustering technique such as k-mean and supervised classification such as logistic regression and be able to assess the quality of classifier using classification matrix.
- Be able to visualize (plot) the output of various techniques mentioned above.

Recall in Regression (2)

Steps for Model Building

Step 1) Write down your hypotheses.

The selected independent variables should make sense in attempting to explain the dependent variable.
Use: logic / theory / your experience / your intuition.

Step 2) Check data, relationships, and assumptions

Plot all your variables. Also make scatterplots between pairs of variables.
Check correlations for linear relationship and possible multicollinearity
Check distribution of variables (Normally distributed? Bimodal?)
Check amount of missing data

Step 3) Use a systematic approach to building your model

Use an analysis plan. E.g., write down and test your hypotheses or plan and do stepwise regression, or a series of ANOVAs.

Step 4) Evaluate and Interpret your model

Correlation != Causation (especially in a "predictive" model)
Principle of parsimony: All things being equal, simpler models are usually better.

Recall in Regression (2)

Pairwise Model Selection

"Full model": $Y \sim X_1 + X_2$

"Restricted model": $Y \sim X_1$

```
m_full <- lm (y ~ x1*x2, df1)
m_restricted <- lm(y~x1, df1)
# model comparison,
anova(m_restricted, m_full)
```

H_0: b_2 (coefficient on X_2) =0

H_1: b_2 (coefficient on X_2) !=0

- anova function with two lm objects conducts a test to see if the explanatory power of the full model is significantly better than the explanatory power of the restricted model i.e., "Is the full model significantly better?"
- to use an anova, restricted model must be a nested model within the full model (ie. it must be a "subset" of the full model)
- Eg: $Y \sim X_1 + X_2$ [restricted: model without interactions]

  $Y \sim X_1 + X_2 + (X_1 * X_2)$ [full: model with interactions]

Data Mining

- Data analysis uses data to test and validate theories, models and hypotheses (with prior theory)
- Data mining uncovers hidden patterns and extract information/knowledge from large data set (without prior theory)
- Data mining with machine learning algorithms is very popular in today's world of big data.
- BUT we need to be cautious of problems of generalization and overfitting.

Data Mining Techniques

Data Exploration

Data Dimensionality Reduction

Data Classification

Data Exploration through Visualizations

We have learnt different types of data visualizations that allow us to study the distribution of the variables and the relationships between pairs of variables, identify outlier issues, missing data, etc.

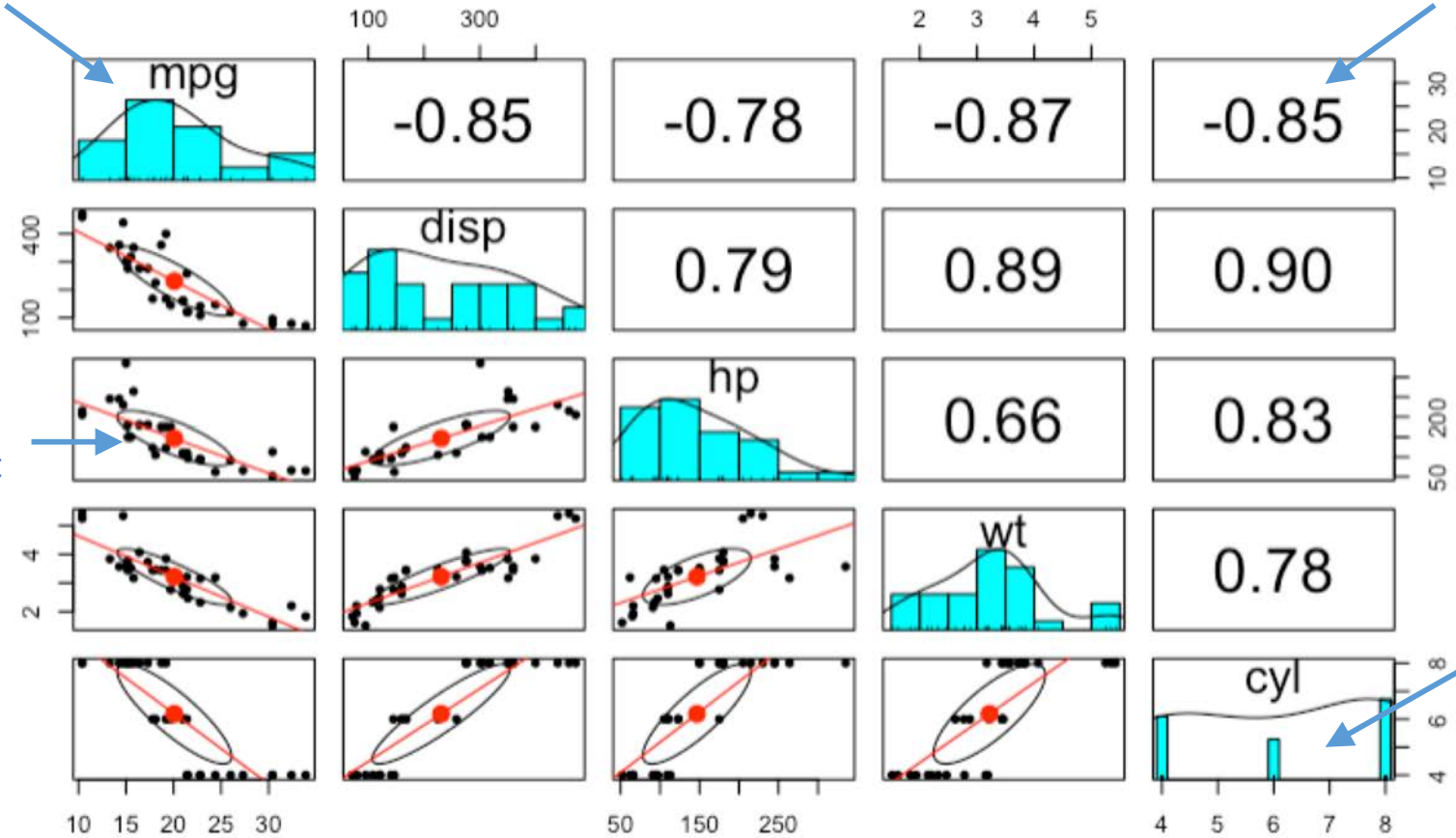
```
# a quick way to show all relationships among variables in a dataset using
'pairs.panels()' in 'psych' package.
mt5<- mtcars %>% select(mpg,disp, hp,wt,cyl)
pairs.panels(mt5, lm=TRUE)
```

Univariate Histogram

Pearson Correlation

Bivariate scatterplots with best fit lines

Discrete variable



Data Dimensionality Reduction

- In linear regression, we have explored how variation of our predictors explain the observed variance for the response variable.

Ecommerce platform data:

CustID	Mon.Sale	Tue.Sale	Wed.Sale	Thu.Sale	Fri.Sale	Sat.Sale	Sun.Sale	……
1	13.50	54.99	45.66	45.00	50.00	0	0	
2	67.90	38.50	50.60	55.50	60.70	0	0	
3	0	0	0	0	0	99.12	56.00	
4	0	0	0	0	0	89.12	120.22	

items purchased, customer demographics, location, mode of payment, browsing behavior, etc.

- With big data, we potentially have hundreds of predictors (i.e. high-dimensionality). Using all predictors have consequences such as over-fitting, low test power, multicollinearity, etc.
- Model selection techniques (like stepwise regression) allow for dimensionality reduction through feature elimination by removing predictors with low explanatory power. Benefits are simplicity and ease of understanding. However we lose information from predictors dropped.

Data Dimensionality Reduction

Question: How could we extract most amount of information (variations) from data and at the same time not include all predictors (dimensions/features) in the model?

Ecommerce platform data: DV

CustID	Mon.Sale	Tue.Sale	Wed.Sale	Thu.Sale	Fri.Sale	Sat.Sale	Sun.Sale	……
1	13.50	54.99	45.66	45.00	50.00	0	0	
2	67.90	38.50	50.60	55.50	60.70	0	0	
3	0	0	0	0	0	99.12	56.00	
4	0	0	0	0	0	89.12	120.22	

items purchased, customer demographics, location, mode of payment, browsing behavior, etc.

- Say, we could construct "new" predictors that combine several predictors.

  Weekday.Sales = 0.2*Mon.Sale + 0.2* Tue.Sale… + 0.2* Fri.Sale

  Weekend.Sales = 0.5*Sat.Sale + 0.5*Sun.Sale

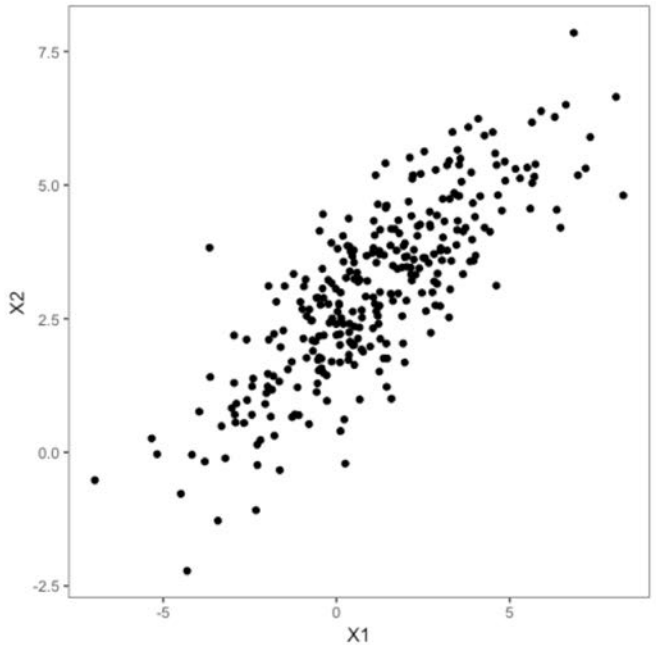
- Instead of using 7 predictors, we reduced it to 2 predictors which summarize information in all 7 predictors.
- This approach of dimensionality reduction involves finding a smaller set of principal predictors which extract and summarize information of high-dimensional data. It is known as feature extraction.

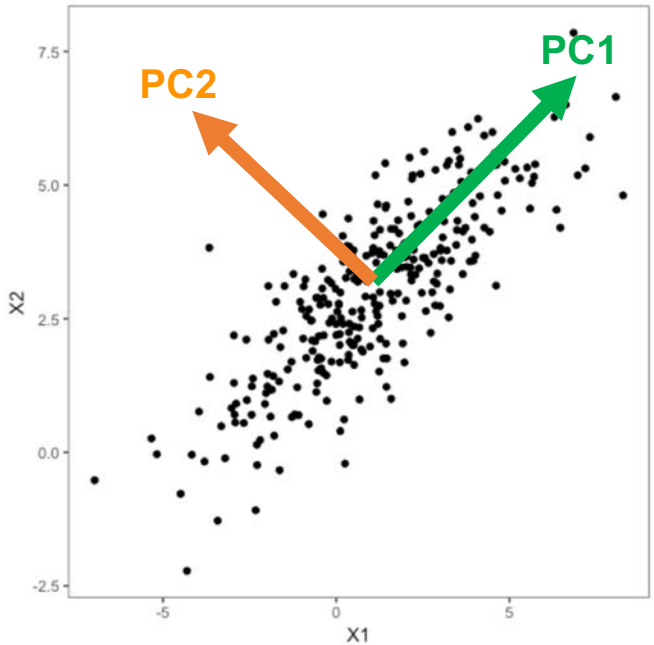
Principal Component Analysis

- A common approach to feature extraction approach of dimensionality reduction is Principal Component Analysis (PCA)

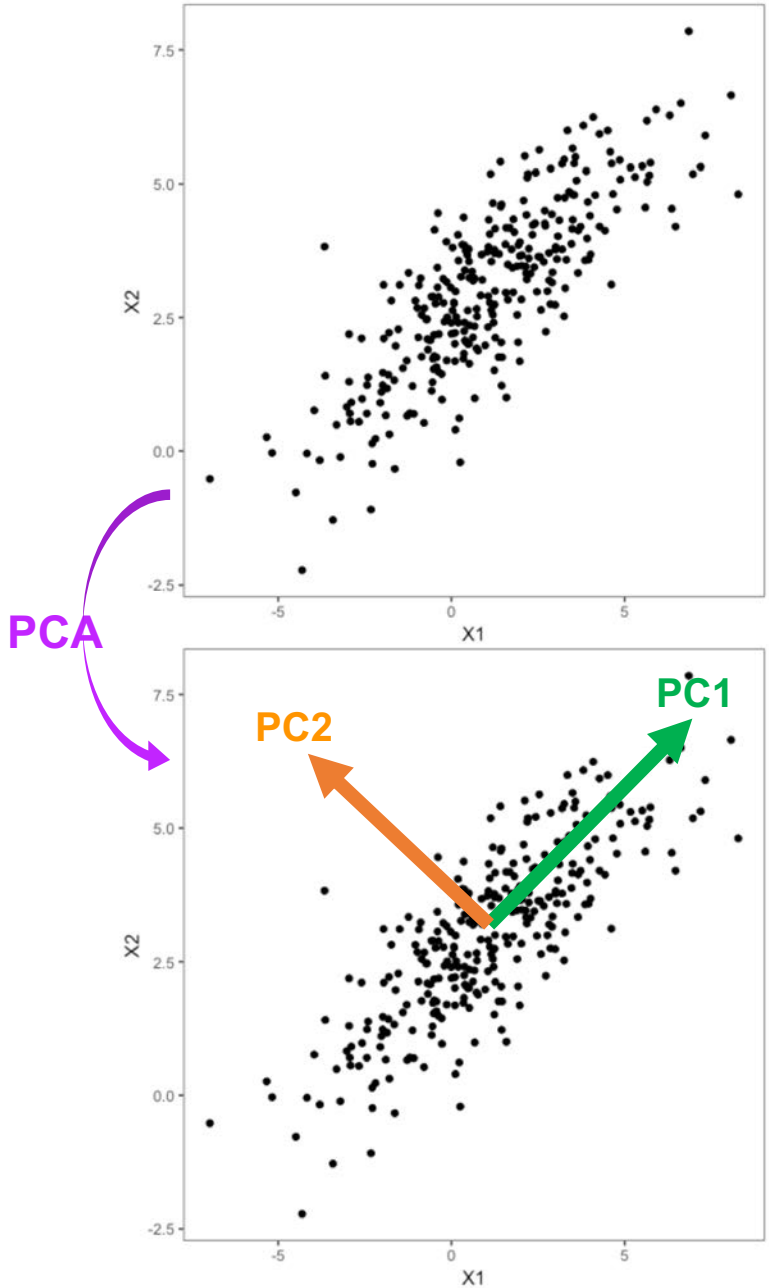
Cust ID	X1	X2	X3	..	Xk	PC1	PC2	.....	PC3	PCk
1	…	…	…	…	…	…	…		…	…

- PCA tries to find a set of "Principal Components" (PCs) that explain the most variance in the data, where PC1 explains the most variation of all X; PC2 is orthogonal to PC1 and explains second most variation and so forth.





Principal Component Analysis



Imagine we have this “data cloud” formed by each data point (along X1, X2) in a 2D Euclidean space

PC1 chosen to be along the “longest” dimension of the data cloud:

- minimise the "distances" of all points projected onto PC1

Given PC1 is fixed, PC2 is chosen to be along the second “longest” dimension of the data cloud and orthogonal to PC1; and so on (if there are more PC’s or X’s).

Principal Component Analysis

- Given k predictors (X1, X2, … Xk), PCA will return k PCs, all mutually orthogonal.

- PCA does a coordinate transformation from X1-Xk to PC1-PCk. The PCs are "ranked" by the % of variance explained. We can then choose a subset of the PCs, e.g. the first k PCs, say based on the cumulative variance explained. (e.g., the first 3 PCs may account for 90% of the variance...)

- Each principal component is a normalized linear combination of all predictors X,

where $PC1 = \phi_{11}X_1 + \phi_{12}X_2 + \ldots + \phi_{1k}X_k$, s.t. $\sum_i \phi_{1i}^2 = 1$

where for the p-th principal component, p = 1; 2; : : : ; k :

$$PCp = \phi_{p1}X_1 + \phi_{p2}X_2 + \ldots + \phi_{pk}X_k \text{ , s.t. } \sum_i^k \phi_{pi}^2 = 1$$

- The linear weights of predictors on PC[p] (ϕ_{p1}; ϕ_{p2}; ; ϕ_{pk}) are called loadings of PC's. It describes how PC[p] is composed of all k predictors.

- One issue is that these PCs are hard to interpret.

Example of Running PCA in R

```
# using 'prcomp()' function on mtcars data (renamed mt), removing binary vars
# "vs" and "am" and DV "mpg"
pca_mt <- prcomp(formula = ~ cyl + disp + hp + drat + wt + qsec + gear + carb ,
data = mt, center = TRUE, scale = TRUE)
# display the output of PCA on ‘mt'
summary(pca_mt)
```

center = TRUE is needed to center the variables

scale = TRUE is to standardize all predictors, making them comparable.

OUTPUT:

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     2.1921 1.4423 0.69434 0.51480 0.41808 0.33026 0.24457 0.15428
Proportion of Variance 0.6007 0.2600 0.06026 0.03313 0.02185 0.01363 0.00748 0.00298
Cumulative Proportion  0.6007 0.8607 0.92094 0.95407 0.97591 0.98955 0.99702 1.00000
```

First two PCs makes up more than 80% of the variance explained.

Example of Running PCA in R

```
# The loadings of the 8 PCs are stored in pca_mt$rotation.
# To view them use the print function
print(pca_mt$rotation)
```

```
              PC1          PC2         PC3         PC4         PC5         PC6         PC7         PC8
cyl    0.4397107 -0.006493485  0.21389399 -0.04117715  0.18531236 -0.02585384  0.83363458 -0.17091062
disp   0.4329689  0.097947929  0.02325701 -0.34154106 -0.44542861 -0.14459253 -0.01593520  0.68277769
hp     0.4018186 -0.261415110 -0.02755330 -0.06992909 -0.23581312  0.78493300 -0.18347022 -0.24324000
drat  -0.3374188 -0.348884539 -0.11277805 -0.84480258  0.13928091  0.03059658  0.12126786 -0.05455856
wt     0.3998604  0.179105944 -0.50702904 -0.19356866 -0.12225944 -0.42501264 -0.18423726 -0.53287799
qsec  -0.2521760  0.478355318 -0.63254001  0.02871245 -0.08864198  0.37604432  0.36411288  0.15750318
gear  -0.2235402 -0.555702724 -0.19470252  0.28156232 -0.62706739 -0.19478214  0.29552606 -0.07448889
carb   0.2653893 -0.480205676 -0.49511212  0.21331221  0.52557339 -0.03719490 -0.05074471  0.35797632
```

Example of

```
# To view just the loadings of PC1 and PC2:
> rbind(pca_mt$rotation[,1],pca_mt$rotation[,2])

              cyl       disp         hp       drat        wt       qsec       gear       carb
[1,]  0.439710686 0.43296889  0.4018186 -0.3374188 0.3998604 -0.2521760 -0.2235402  0.2653893
[2,] -0.006493485 0.09794793 -0.2614151 -0.3488845 0.1791059  0.4783553 -0.5557027 -0.4802057
```

```
# Recall that PC is the normalized linear combination of ALL predictors, ie:
# sum of squares of those coefficients add up to one. We can check this:
> sum(pca_mt$rotation[,1]^2)
[1] 1
```

Applying PCA in linear regression

```
# Let's run a linear regression on mpg with the top 2 PCs which explain 86%
of all predictors
# We will use the data with rotated variables (e.g. pca_mt$x here)
```

data with original variables

```
> mtsub<-subset(mtcars, select=-c(vs,am,mpg))
> mtsub
                   cyl  disp  hp drat    wt  qsec gear carb
Mazda RX4            6 160.0 110 3.90 2.620 16.46    4    4
Mazda RX4 Wag        6 160.0 110 3.90 2.875 17.02    4    4
Datsun 710           4 108.0  93 3.85 2.320 18.61    4    1
Hornet 4 Drive       6 258.0 110 3.08 3.215 19.44    3    1
Hornet Sportabout    8 360.0 175 3.15 3.440 17.02    3    2
Valiant              6 225.0 105 2.76 3.460 20.22    3    1
Duster 360           8 360.0 245 3.21 3.570 15.84    3    4
Merc 240D            4 146.7  62 3.69 3.190 20.00    4    2
Merc 230             4 140.8  95 3.92 3.150 22.90    4    2
Merc 280             6 167.6 123 3.92 3.440 18.30    4    4
Merc 280C            6 167.6 123 3.92 3.440 18.90    4    4
Merc 450SE           8 275.8 180 3.07 4.070 17.40    3    3
Merc 450SL           8 275.8 180 3.07 3.730 17.60    3    3
Merc 450SLC          8 275.8 180 3.07 3.780 18.00    3    3
Cadillac Fleetwood   8 472.0 205 2.93 5.250 17.98    3    4
Lincoln Continental  8 460.0 215 3.00 5.424 17.82    3    4
```

data with rotated variables (in terms of PCs)

```
> pca_mt$x
                        PC1        PC2        PC3        PC4         PC5        PC6         PC7        PC8
Mazda RX4        -0.64738354 -1.1828309  0.26961666  0.12911933  0.704263314 -0.46009396  0.005912376  0.16202803
Mazda RX4 Wag    -0.62220221 -0.9862442 -0.06075048  0.08767061  0.644621749 -0.45301193  0.072004783  0.07251144
Datsun 710       -2.30846879  0.2933300  0.35170586  0.11256845 -0.316275460  0.08388222 -0.297983964 -0.17963196
Hornet 4 Drive   -0.15488166  1.9814200  0.28127869  0.30702848 -0.209984621  0.08039856 -0.003786184  0.16027080
Hornet Sportabout  1.62839918  0.8573121  0.93256842 -0.14839587 -0.157042413  0.05057134  0.191712384  0.17883872
Valiant          -0.10747735  2.4368571 -0.05846838  0.87273767 -0.226851141  0.10106679  0.054062009 -0.03581427
Duster 360        2.54904056 -0.3354249  0.62904594 -0.09513895  0.310906300  0.50454721 -0.309747655  0.19283141
Merc 240D        -1.93029468  0.7805570 -0.84421667  0.27263007 -0.242686562 -0.43374070 -0.168296230  0.03020595
Merc 230         -2.32825091  1.2689874 -1.91290945 -0.05366293 -0.413925312  0.59175103  0.394765850  0.13455350
Merc 280         -0.48182626 -0.5967823 -0.81463865 -0.06933960  0.443713547 -0.28796106  0.195207235 -0.12866518
Merc 280C        -0.56649913 -0.4361654 -1.02702593 -0.05969885  0.413950324 -0.16169707  0.317465024 -0.07578051
Merc 450SE        1.78218192  0.7436474  0.16412709  0.21847628  0.335354029 -0.01524720  0.098402133 -0.38257509
Merc 450SL        1.61501184  0.7349496  0.26951669  0.28895221  0.367916364  0.17452663  0.203174546 -0.17977937
Merc 450SLC       1.57899647  0.8511800  0.10201556  0.28548786  0.341826655  0.23698412  0.275265060 -0.17175344
Cadillac Fleetwood  3.26713410  0.9686922 -0.90288066 -0.21854764 -0.343052500 -0.37947478 -0.160896060  0.25399449
Lincoln Continental 3.33333123  0.8644244 -0.95744485 -0.34327289 -0.329888919 -0.35623644 -0.235601399  0.03640205
```

Applying PCA in linear regression

```
# Let's run a linear regression on mpg with the top 2 PCs which explain 86% of
all predictors
> mt_pca = mt
> mt_pca$pc1 = pca_mt$x[,"PC1"]
> mt_pca$pc2 = pca_mt$x[,"PC2"]
# run a linear regression of `mpg ~ pc1 + pc2`
> pcafit = lm(mpg ~ pc1 + pc2, mt_pca)
> summary(pcafit)
```

```
Call:
lm(formula = mpg ~ pc1 + pc2, data = mt_pca)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.7923 -1.4296 -0.7709  1.3392  5.8644 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  20.0906     0.4594  43.728  < 2e-16 ***
pc1          -2.4953     0.2129 -11.718 1.61e-12 ***
pc2          -0.2019     0.3237  -0.624    0.538    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.599 on 29 degrees of freedom
Multiple R-squared:  0.826,	Adjusted R-squared:  0.814 
F-statistic: 68.85 on 2 and 29 DF,  p-value: 9.7e-12
```

pc1 is strongly significant.
Coefficient for pc1 is -2.4953 – hard to interpret given that pc1 is linear combination of all predictors.

R^2 = 0.826 very high and together with large F-stat, indicate a powerful explanatory linear model.

Visualizing PCA

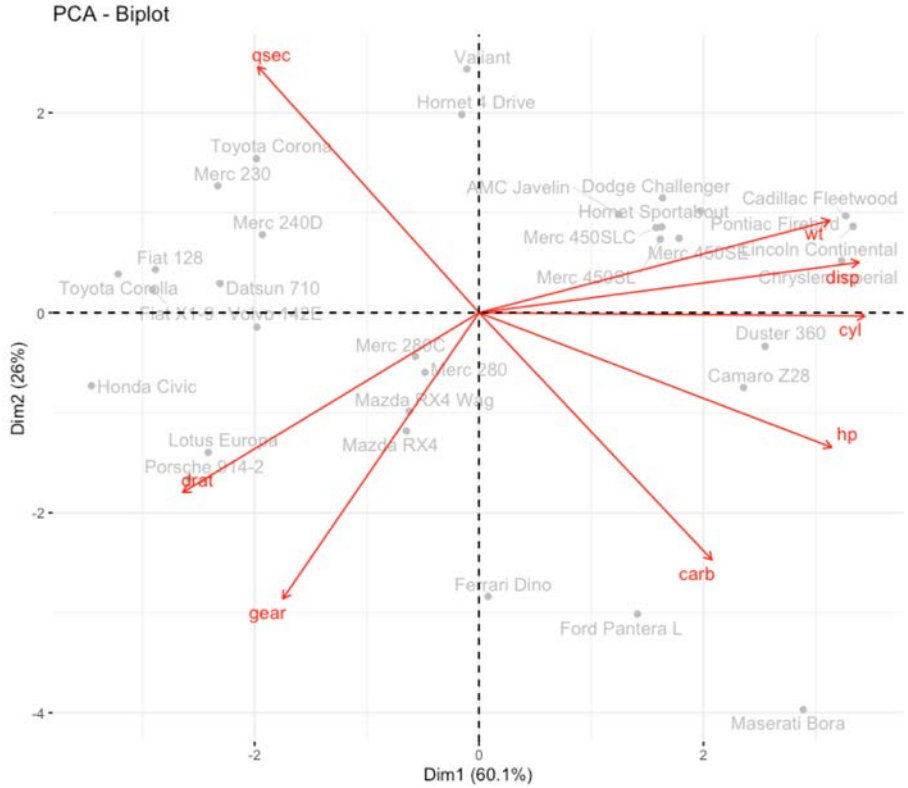
```
> rbind(pca_mt$rotation[,1],pca_mt$rotation[,2])
              cyl       disp         hp       drat        wt       qsec       gear       carb
[1,]  0.439710686 0.43296889  0.4018186 -0.3374188 0.3998604 -0.2521760 -0.2235402  0.2653893
[2,] -0.006493485 0.09794793 -0.2614151 -0.3488845 0.1791059  0.4783553 -0.5557027 -0.4802057
```

```
# using factoextra package for PCA visualization
library(factoextra)
```

avoid overlapping text

```
fviz_pca_biplot(pca_mt, repel = TRUE,
    col.var = “red”, # Variables color
    col.ind = “grey”  # Individuals color
)
```

We can use biplot to show compositions of top 2 PC’s with respect to all predictors.



You can see how data points (grey) look like in the “plane” formed by pc1 and pc2 (where most variation should be).

The red arrows in biplot are pointing in the direction of the original predictors, as projected into the 2D plane of the biplot.

For instance, the arrow direction of cyl is given by (0.440, -0.00649) in the coordinate set up by pc1 and pc2. Similar to other arrows.

Clustering Analysis

- Dimensionality reduction is about “compressing” the features of the data
- In clustering analysis, we draw insights about the observations

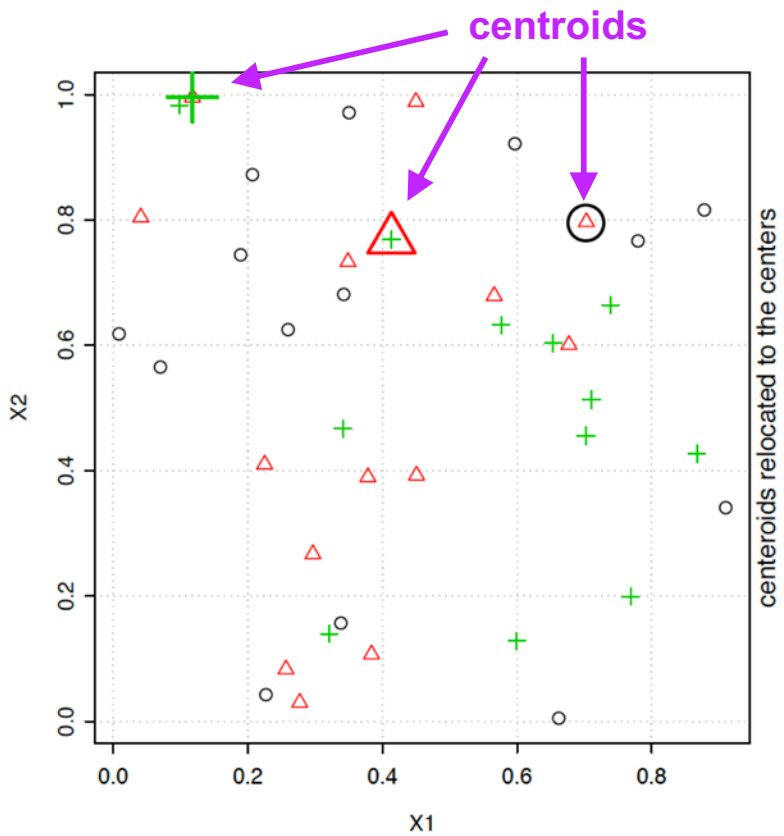
Ecommerce platform data:

Cust type	Tran	Mon.Sale	Tue.Sale	Wed.Sale	Thu.Sale	Fri.Sale	Sat.Sale	Sun.Sale	……
1	1	13.50	54.99	45.66	45.00	50.00	0	0	
1	2	67.90	38.50	50.60	55.50	60.70	0	0	
2	3	0	39.10	0	0	0	99.12	56.00	
3	4	0	50.60	0	90.20	0	88.12	86.00	
3	5	0	10.12	0	50.60	0	45.00	46.00	
4	6	0	79.20	0	0	50.60	89.12	120.22	

Can we identify different types of customers, with different online shopping profiles? Can we identify clusters of observations ("rows"), based on the similarity?

K-Means Clustering

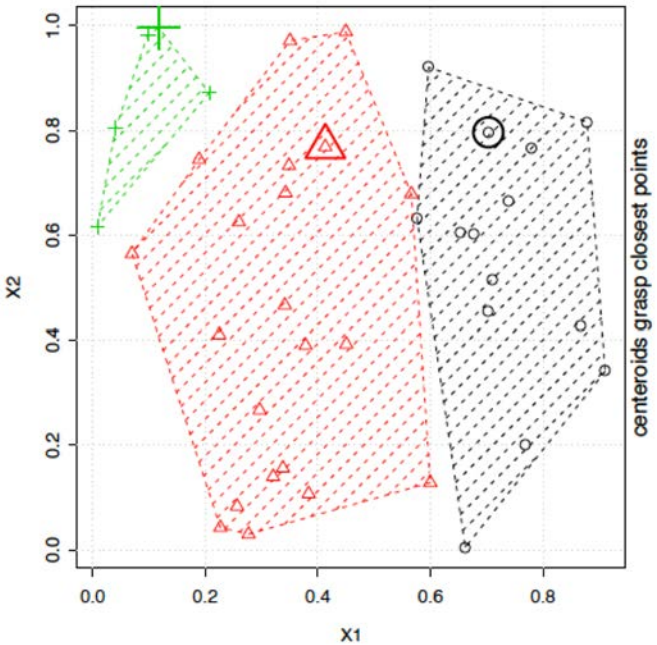
- K-means clustering is a widely-used algorithm to partition a given dataset of observations into k different groups (or clusters).
- The idea is to assign each observation to the nearest centroid (mean) with within-cluster distance minimized.
- It consists of two steps, which are repeated over and over until convergence



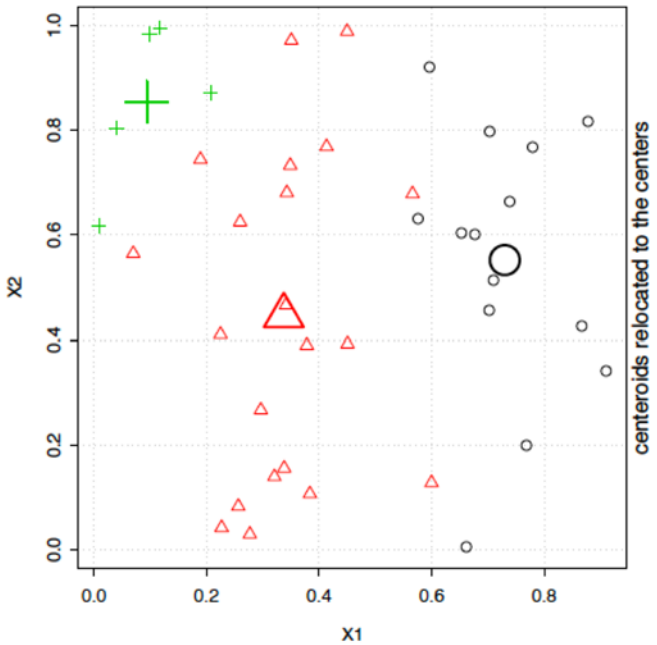
Before clustering, we need to decide on k – the number of clusters to fit the data to.

1. Initialisation Step: Place the centroids of k clusters on k randomly chosen datapoints. (here k=3)

K-Means Clustering

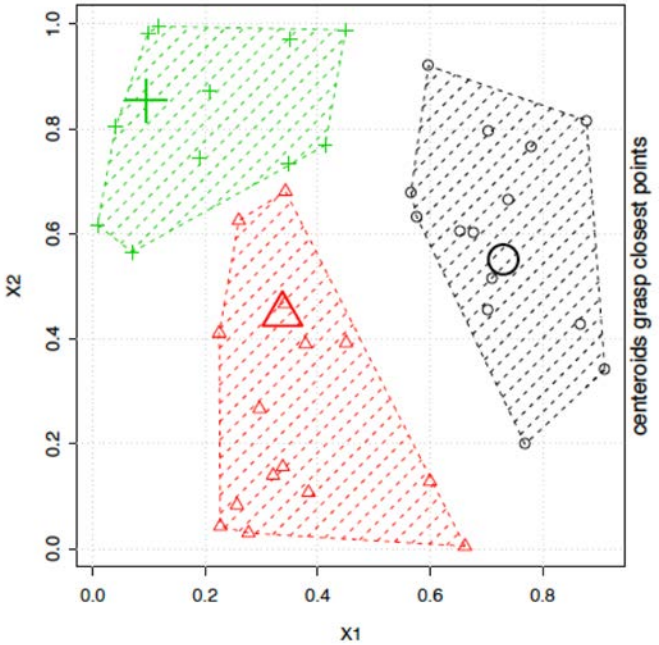


2. Assignment Step: Distance from each datapoint to all centroids are computed such that datapoints are “assigned” to the cluster with the closest centroid.



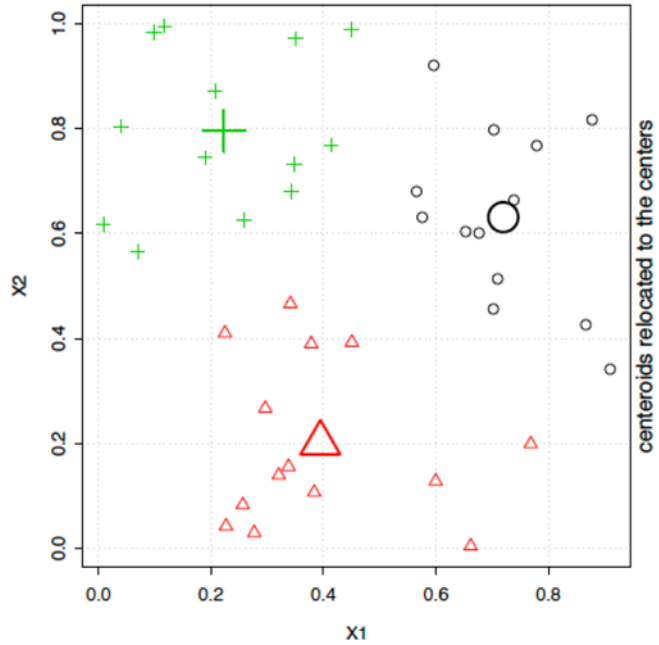
3. Update Step: Update the centroid position to be the mean of all points assigned to that cluster.

K-Means Clustering



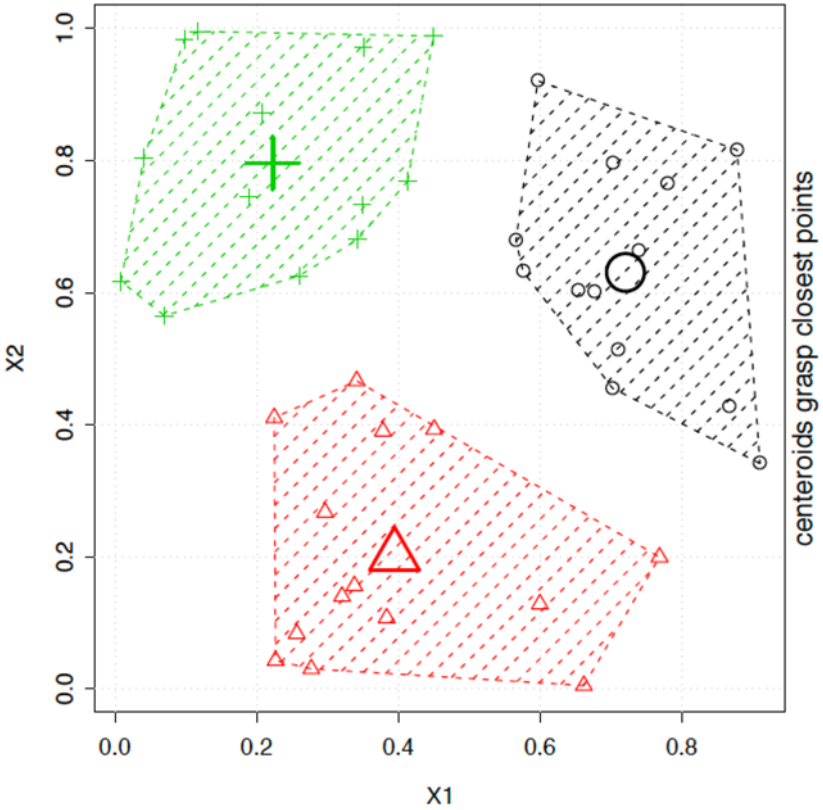
Repeat

2. Assignment Step: Distance from each datapoint to all centroids are computed such that datapoints are “assigned” to the cluster with the closest centroid.



3. Update Step: Update the centroid position to be the mean of all points assigned to that cluster.

K-Means Clustering



k-means algorithm stops when there are no more changes of centroid's positions or pre-specified maximum number of iterations is reached.

Note: There is no guarantee that k-means will reach the global optimum, i.e. each run of k-means algorithm may depend on the random initialization.

```
# In R, we run the k-means algorithm using `kmeans()` function
km_dmatrix = kmeans(dmatrix, centers = 3, nstart = 10)
```

numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).

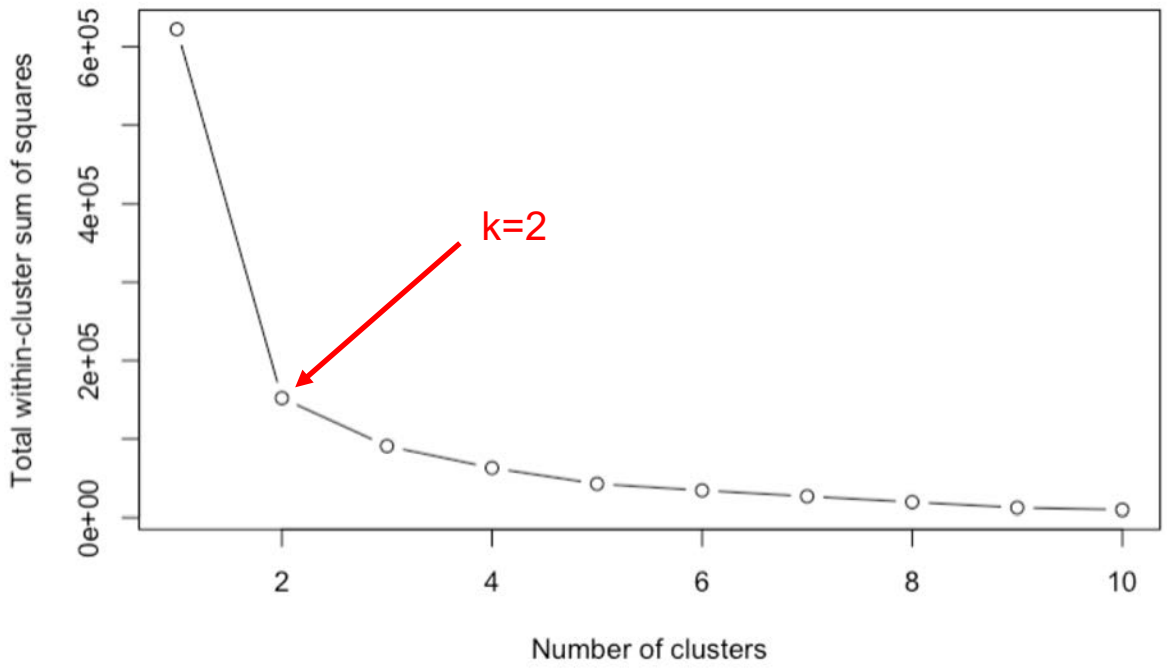
nstart argument: number of different initializations the kmeans algorithm will repeat with, and then report the best one (lowest within-cluster sum-of-squared distance)

K-Means Clustering

How to determine k?

Plot the "within-cluster sum of squared errors" as a function of the number of clusters.

```
mtsub = mtcars %>% select("cyl", "disp", "hp", "drat", "wt", "qsec", "gear", "carb")
wss <- rep(NA, 10)
for(k in c(1:10)) {
  wss[k] = kmeans(mtsub, k, nstart=10)$tot.withinss
}
plot(wss, type="b", xlab="Number of clusters", ylab="Total within-cluster sum of squares")
```

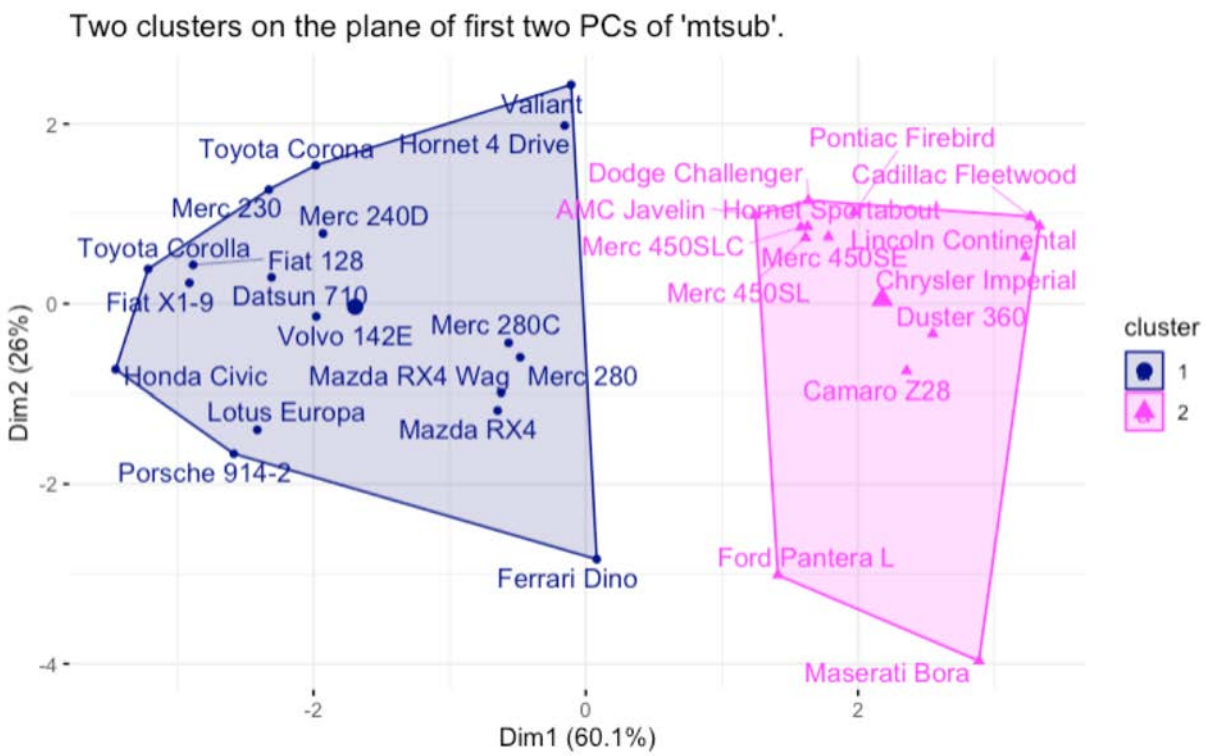


When choosing the best number of clusters, look for the “elbow”!

In this case, k = 2.

K-Means Clustering

```
# k-means plot using 'fviz_cluster' in 'factoextra' package
fviz_cluster(km_mt, data = mtsub, repel = TRUE,
             palette = c("navy","magenta"),
             ggtheme = theme_minimal(),
             main = "Two clusters on the plane of first two PCs of ‘mtsub'.")
```



fviz_cluster() applies PCA first and plots the k-means clustering of observations that are projected onto the “plane” of top two PC’s.

Classification

K-means clustering is a non-supervised learning algorithm – no human labels are provided and clusters (or categories) are derived from data.

Classification, one of most used machine learning technique, identifies which category a new observation belongs to, based on a trained classifier on a training set of data in which membership of observations are already labelled.

Classification is a form of supervised learning technique since such classifier has to be trained on pre-labelled data by human.

Customer	Wkday Spending	Wkend Spending	Marital Status	#Browsing Time/day	Purchased @ 11/11
Aimee	$300	$50	Single	200	**Yes**
Lorraine	$0	$20	Single	50	No
Tesla	$50	$100	Married	20	No

There are many classifiers: naive Bayes, k-nearest neighbor, various neural networks (which you can learn in future courses) and logistic regression (which you already know!)

$$\text{logit}(p) = \log \frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$$

p is probability of a label for an event being positive.

Here, logistic regression is used to learn how to label the events, classify a label (/group) on existing, labelled data, and also to predict labels on new or unlabelled data

Evaluating Performance of Classifiers

Classification (or Confusion) Matrix

	Predict = Yes	Predict = No
Actual = Yes	True Positive (TP)	False Negative (FN)
Actual = No	False Positive (FP)	True Negative (TN)

Classification Accuracy	(TP+TN)/ (TP+TN+FP+FN)	Overall, how often is the classifier correct?
Recall /Sensitivity(or hit/true positive rate)	TP/ (TP+ FN)	When it's actually yes, how often does it predict yes?
Precision	TP/ (TP + FP)	When it predicts yes, how often is it actually yes?
Specificity (true negative rate)	TN/ (TN+ FP)	When it's actually no, how often does it predict no?
F-Score/ F1-Score	2*Precision*Recall/ (Precision + Recall)	Weighted average of Recall and Precision

What we want:
High sensitivity: improve detecting true positives and reduce mislabelling them as negative
High precision: increase relevance of predicted positives
High specificity: improve detecting true negatives and reduce false alarms

Evaluating Performance of Classifiers

Example: Using Logistic Regression to predict probability of <Disease X> given medical history, travel history, exercise records, dietary records, etc.

	Predict = Yes	Predict = No
Actual = Yes	80 (True Positive)	20 (False Negative)
Actual = No	40 (False Positive)	60 (True Negative)

Recall/Sensitivity
= 80/ (80 + 20)
= 80%

High recall is desired because we want to maximise detecting the true disease cases. (Undetected cases go untreated)

Precision = 80/ (80 + 40)
= 66.67%

High precision is desired because we want to minimize misclassifying healthy individuals (where treatments may have unnecessary adverse effects)

Specificity = 60/ (60 + 40)
= 60%

High specificity is desired because we want to minimize false positives and misclassifying healthy individuals (imagine telling someone they have cancer when they do not)

Accuracy = (80 + 60)/ (80 + 20 + 40 + 60)
= 70%

F-score = 2*(0.6667)(0.8)/0.6667+0.8)
= 72.7%

Classification Matrix in Hypothesis Testing

Classification matrix in hypothesis testing give us type I and type II errors :

- Type I error (α): probability of rejecting the null, given the null is true,

i.e. reject a true null, or Pr(reject H0|H0 is true) = α .

- Type II error (β): probability that we fail to reject the null, given the null is false,

i.e. fail to reject a false null, or Pr(don't reject H0|H0 is false) = β .

	Predict = Yes **Reject H0**	Predict = No **Fail to reject H0**
Actual = Yes **H0 is False**	True Positive (TP) Correct rejection = 1 - β	False Negative (FN) Type II error = β
Actual = No **H0 is True**	False Positive (FP) Type I error = α	True Negative (TN) Correct no rejection = $1 - \alpha$

In hypothesis testing, type I error is controlled via our specified "significance level", e.g. α = 0.05.

β is the probability of type II error and 1- β is called the power of test, i.e. the probability that we (successfully) reject the null, given that the null is false.

Logit Classifier and Classification Matrix in R

Example: apply logit classifier to a subset of titanic.csv to classify survival of each passenger. confusionMatrix() in caret package is then used to produce classification matrix.

```
# use 'glm()' with specified parameter 'family = binomial' for logistic regression
fit_surv = glm(survived ~ sex + age + sibsp + parch + fare + embarked, family =
binomial, data = titanic, control = list(maxit = 50))
# predict the survival probability using fitted logistic regression
predprob_surv = predict(fit_surv, type = 'response')
# define survived = 1 when predicted probability >= 0.5; 0 otherwise
pred_surv = ifelse(predprob_surv >= 0.5, 1, 0)
# using 'confusionMatrix()' in 'caret' package
cm = confusionMatrix(pred_surv, titanic$survived, positive = 'Survived')
```

Predicted data

Actual data

Survived is specified as "positive" event.

```
Confusion Matrix and Statistics

                Reference
Prediction      Not Survived Survived
  Not Survived           474      112
  Survived                75      228

               Accuracy : 0.7897
                 Sensitivity : 0.6706
                 Specificity : 0.8634
              Pos Pred Value : 0.7525
              Neg Pred Value : 0.8089
```

= (474 + 228)/(474 + 75 + 112 + 228)

or recall = 228/(112 + 228)

= 474/(474 + 75)

or precision = 228/(228 + 75)

= 474 /(474 + 112)

Summary

- As data analysts, we may encounter datasets with many variables (or dimensions/features/columns), as well as many observations ("rows").
- Data Mining is a set of techniques to obtain insights from such large datasets without a priori theory. In this lecture, we have discussed several data mining techniques including data visualisation, dimensionality reduction, and data classification
- Dimensionality reduction can be performed through feature elimination or feature extraction. PCA is a common method for feature extraction. It involves identifying a smaller set of principal predictors that transform and summarize all variables. Principal components in PCA are ranked by proportion of variations explained in data.
- Clustering seeks to group similar observations based on some similarity. k-means algorithm (an unsupervised learning technique) can be used to find k clusters of data points based on distances of their features.
- Classification logistic regression gives us a commonly used logit classifier to label binary outcomes (hence it’s a supervised learning technique) and classification matrix is used to evaluate classifiers.