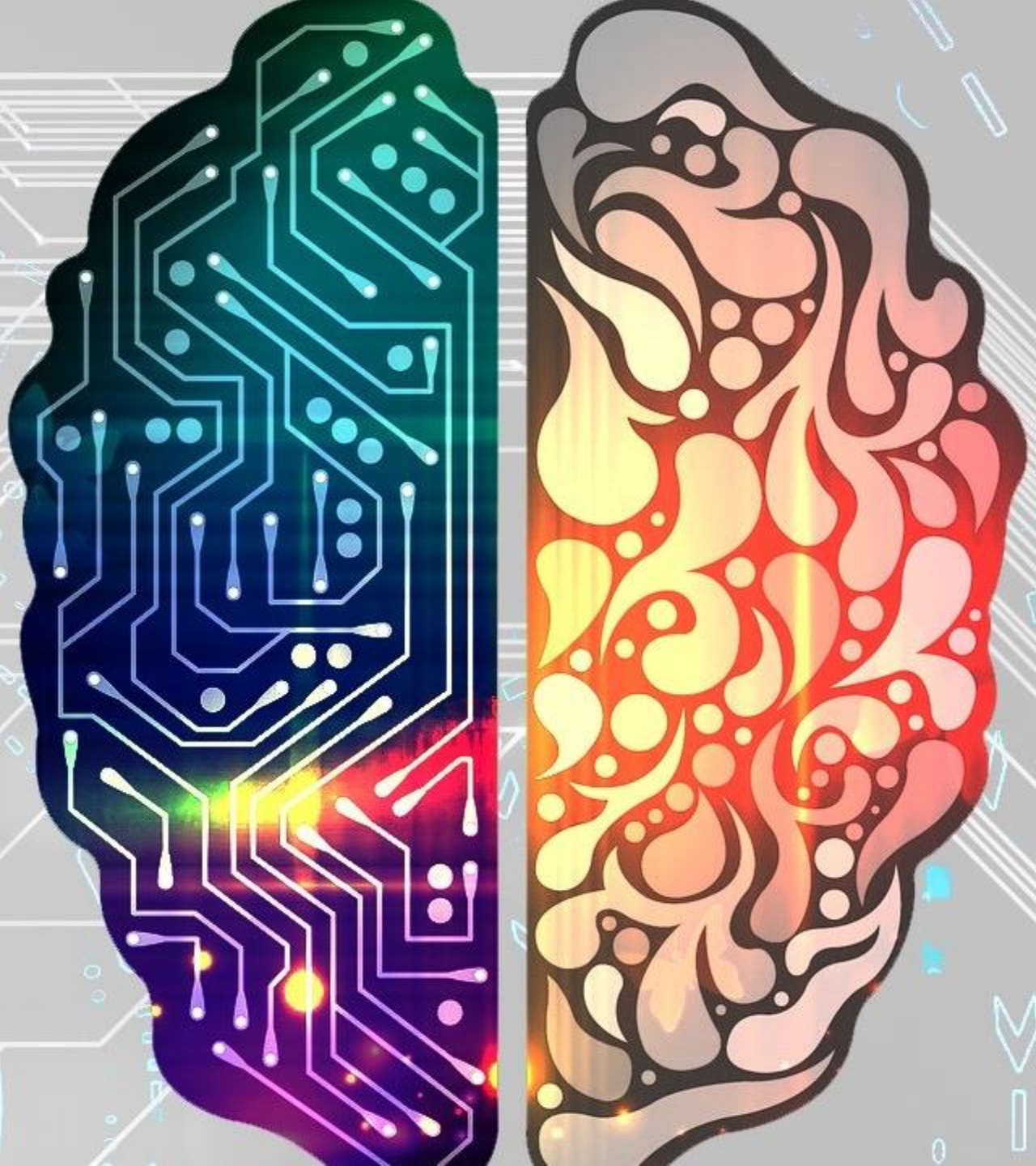
MACHINE LEARNING

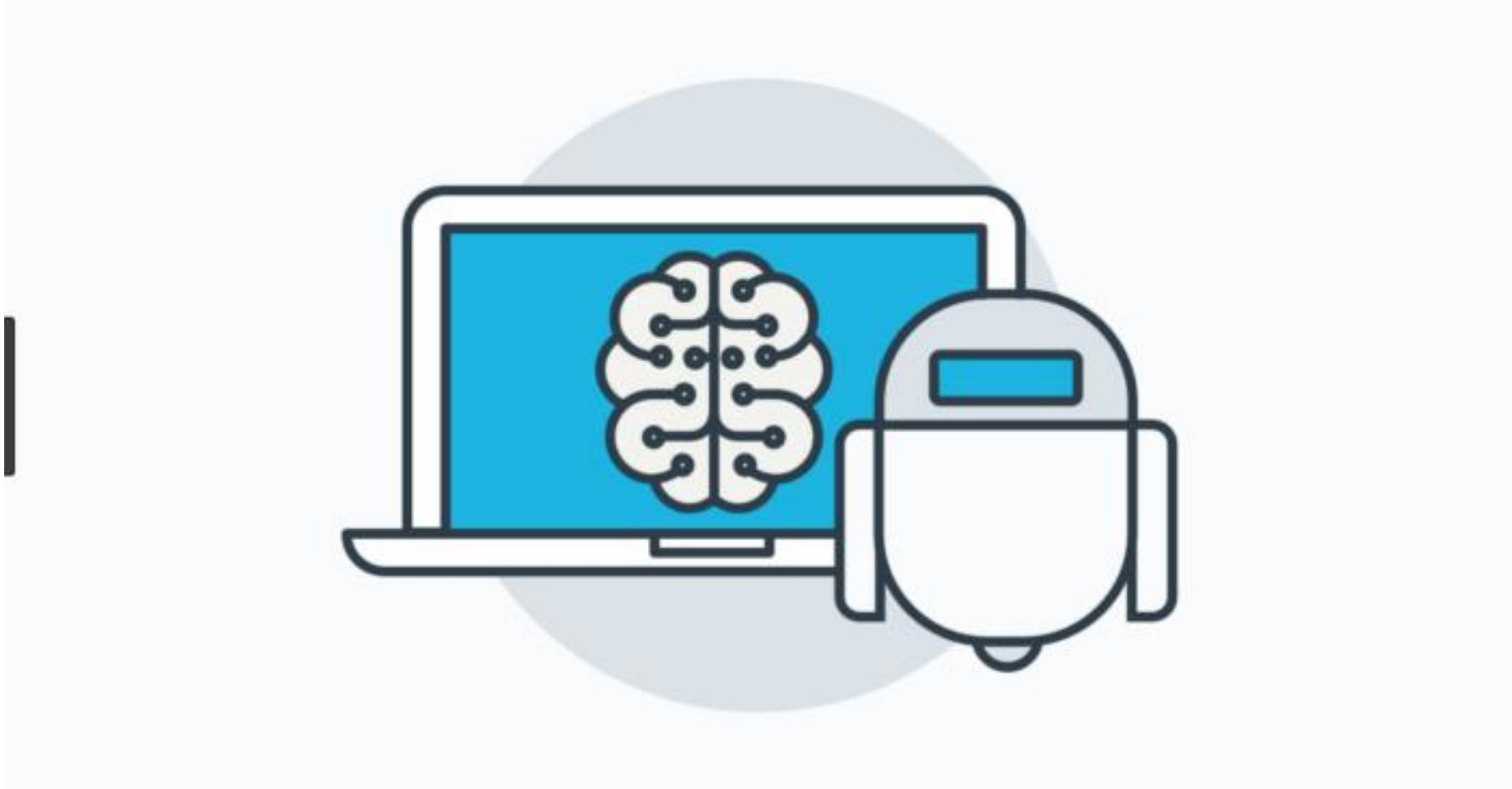


Jingjing Yang, PhD
Department of Human Genetics
Emory University School of Medicine

Outline

- Overview
- Supervised Machine Learning
  - Regression
  - Classification
- Unsupervised Machine Learning
  - Association Study
  - Clustering

Overview



Machine Learning

Prediction

- Product recommendation

Image Recognition

- Face ID

Speech Recognition

- Siri

Medical Diagnoses

- Risk score for Type 2 Diabetes

Financial Trading

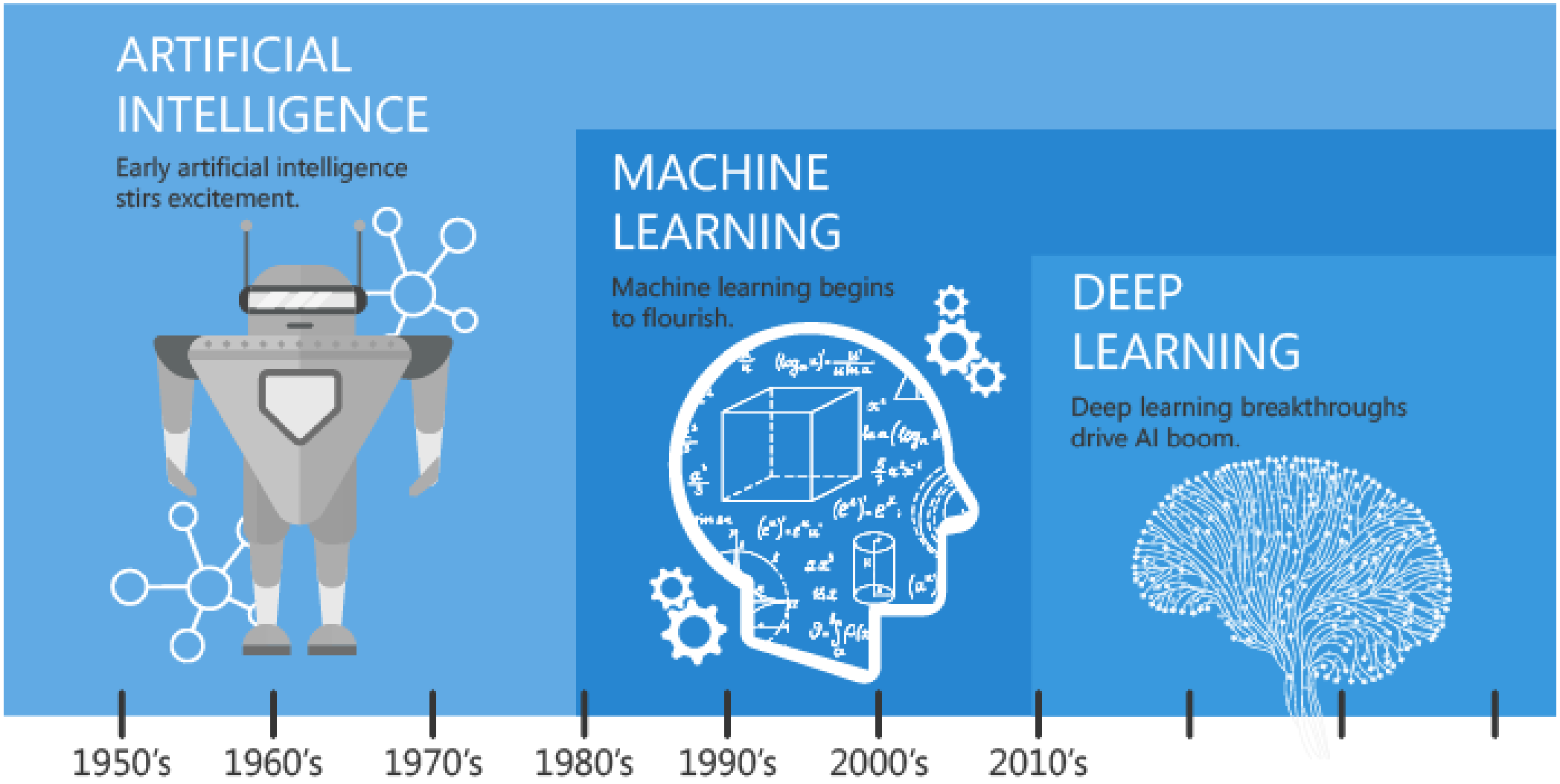
Machine Learning vs. Statistical Learning

- According to **Arthur Samuel**, Machine Learning algorithms enable the computers to learn from data, and even improve themselves, without being explicitly programmed.
- According to "**The Elements of Statistical Learning**", the bible of Statistical Learning, Statistical Learning is referred to using statistical methods to extract important patterns and trends, and understand data that were generated in many fields.
- The intersection of **Computer Science** and **Statistics** gave birth to probabilistic approaches in **Artificial Intelligence**.
- **Key message**: Learning from the DATA, Statistical Methods, Computational Algorithms

Machine Learning

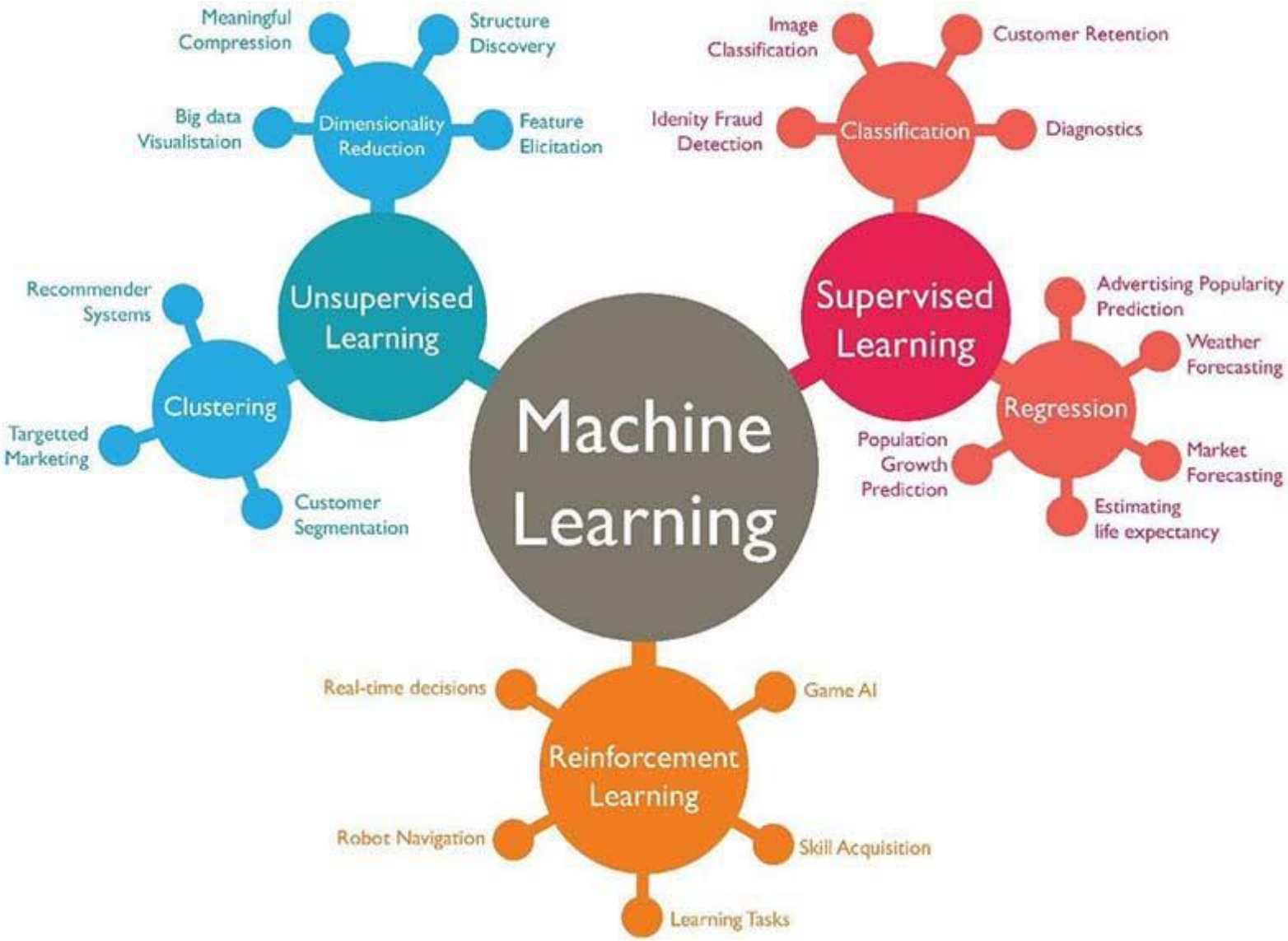
- **Machine Learning (ML)** is a category of algorithms that allow software applications to become more accurate in predicting outcomes without being explicitly programmed.
- Basic premise of machine learning is to build algorithms that can
  - Receive input data
  - Use statistical analysis
  - Predict an output
  - Updating outputs as new data becomes available.

Quick History about Machine Learning

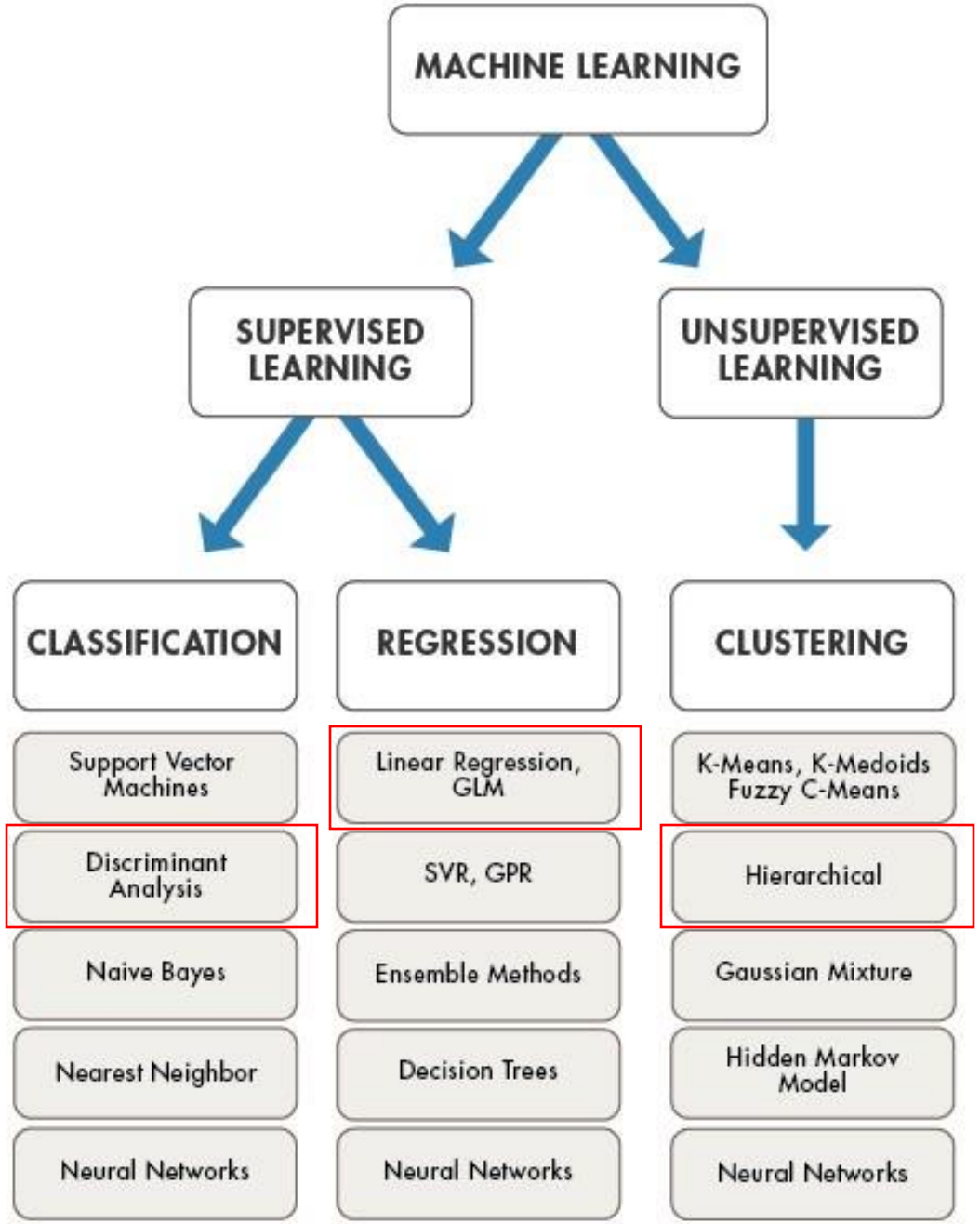


Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Types of Learning : Supervised, Unsupervised, Reinforcement



Machine Learning Methods

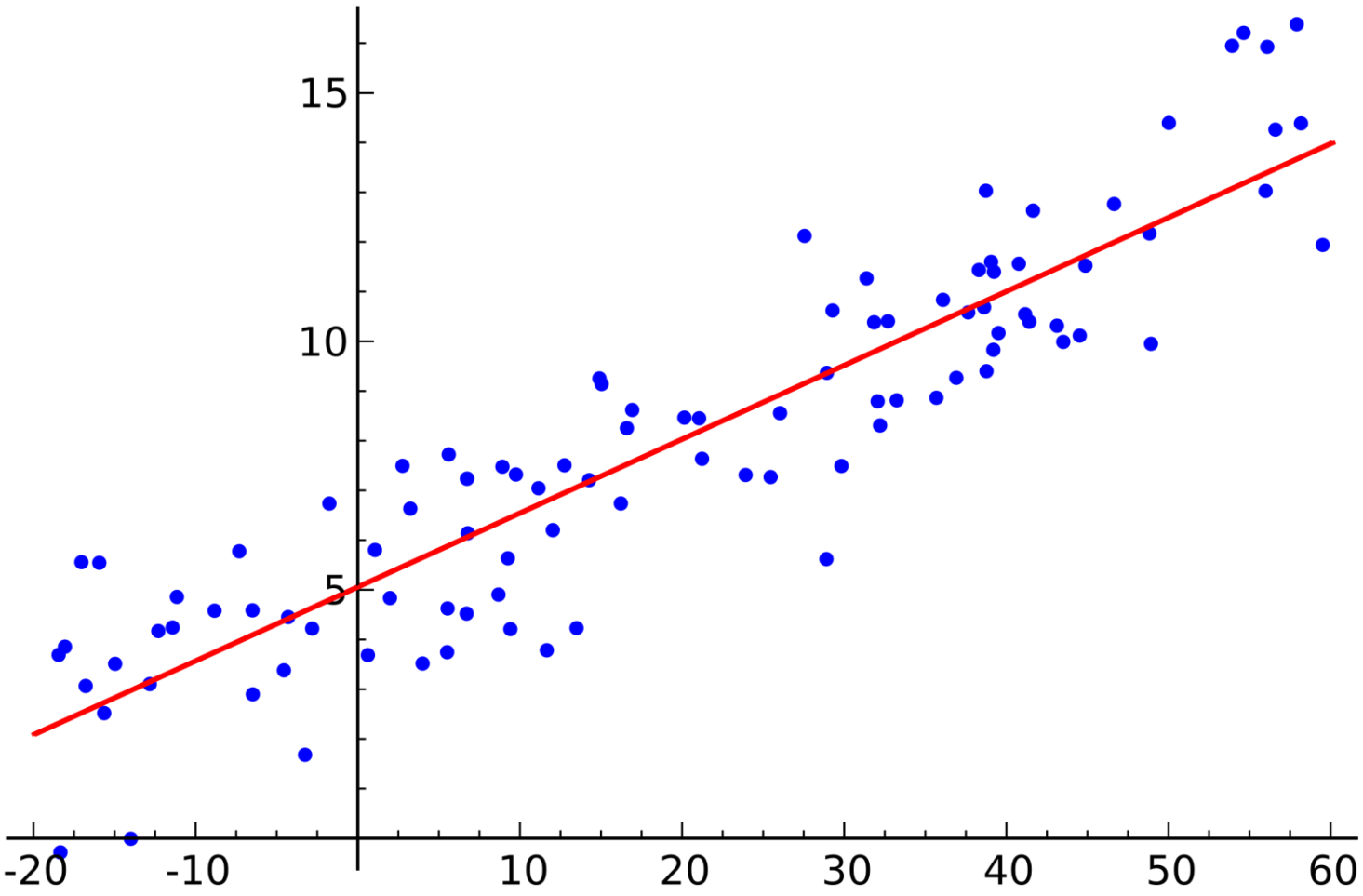


Supervised Learning

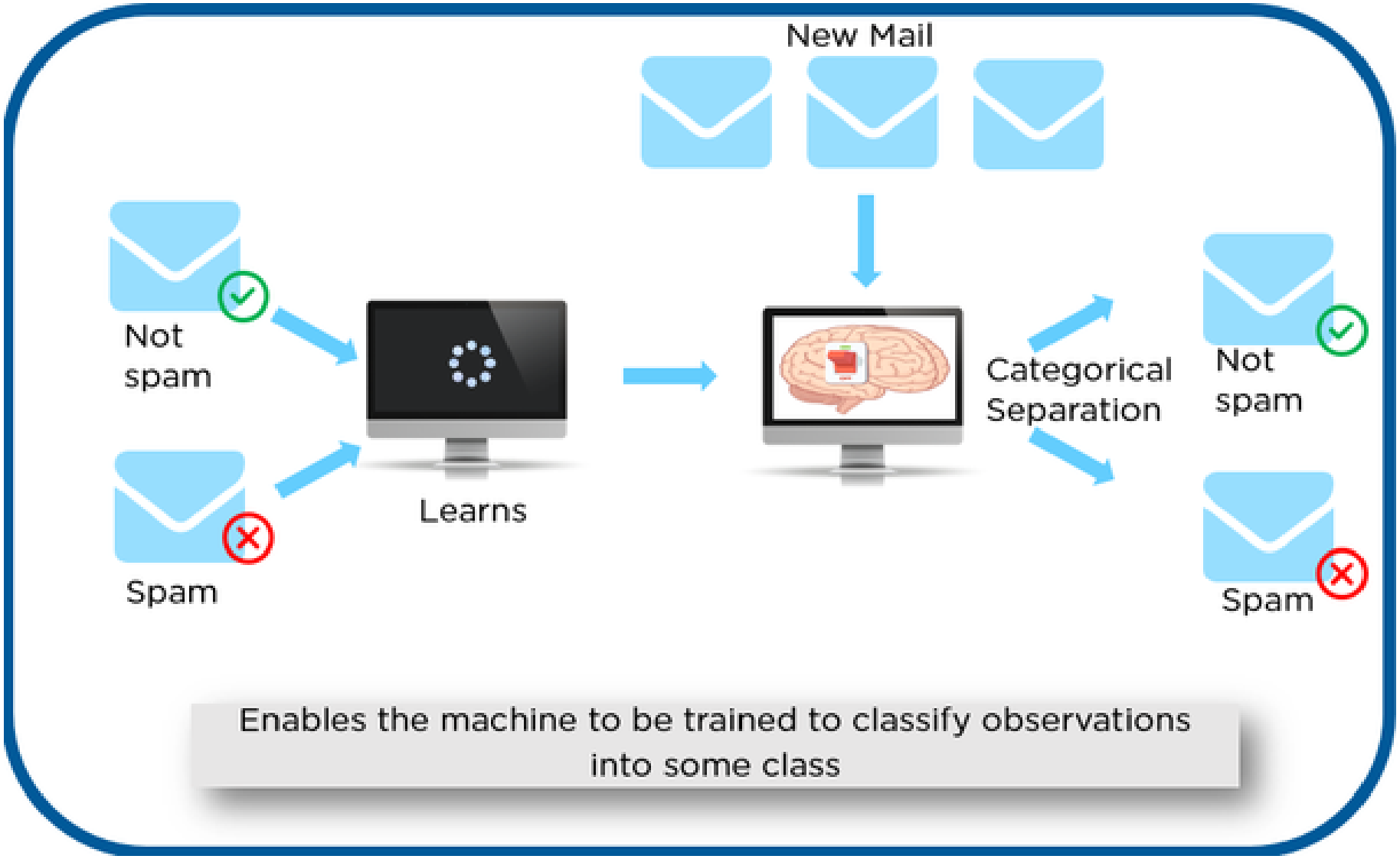
Supervised Learning

- **Regression**
  - A regression problem is when the output variable is a real measured value, such as “weight”, “BMI”, “blood pressure”.
  - Regression analysis is a form of predictive modelling technique which investigates the relationship between dependent and independent variables.
- **Classification**
  - A classification problem is when the output variable is a category, such as “disease” or “no disease”.

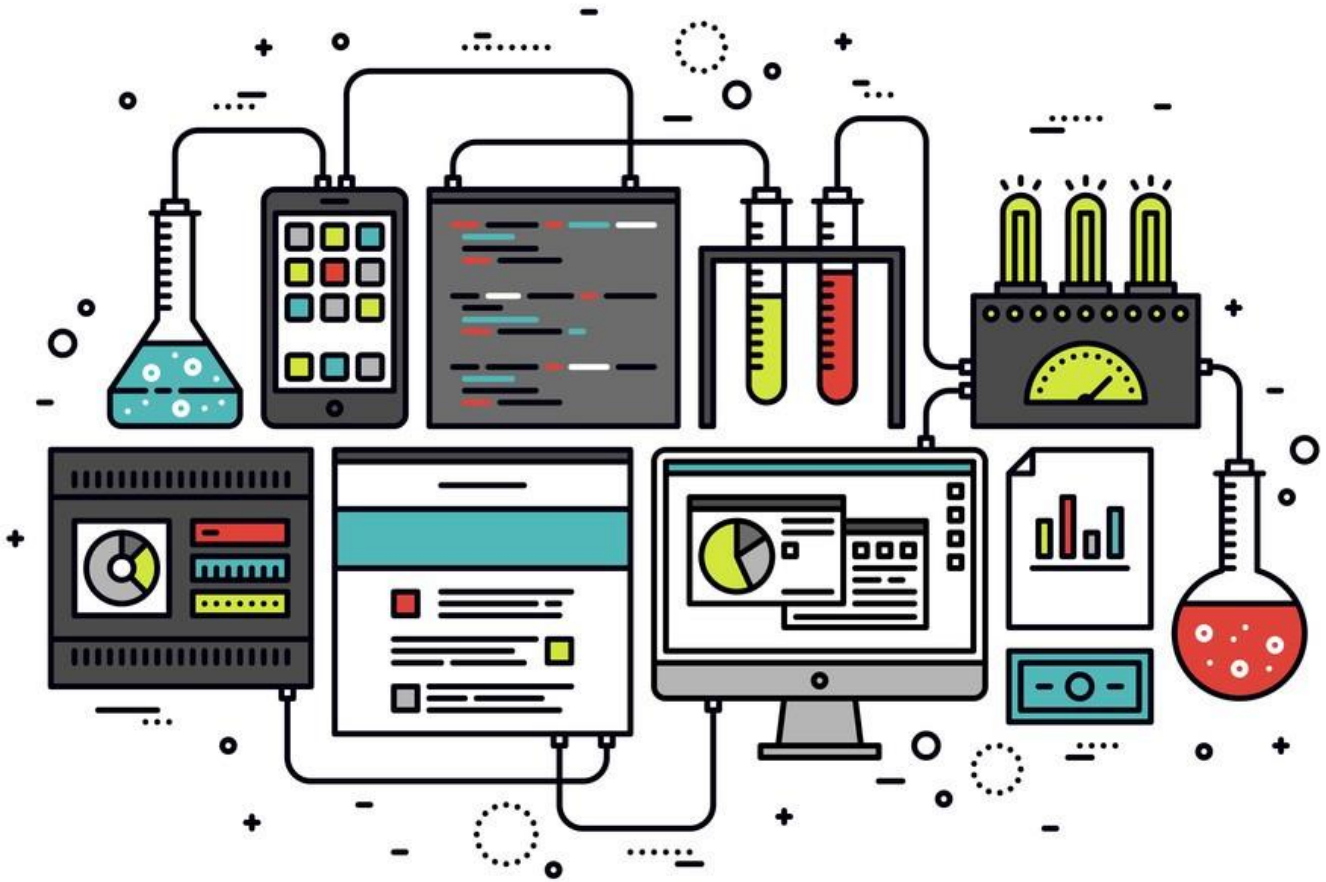
Regression



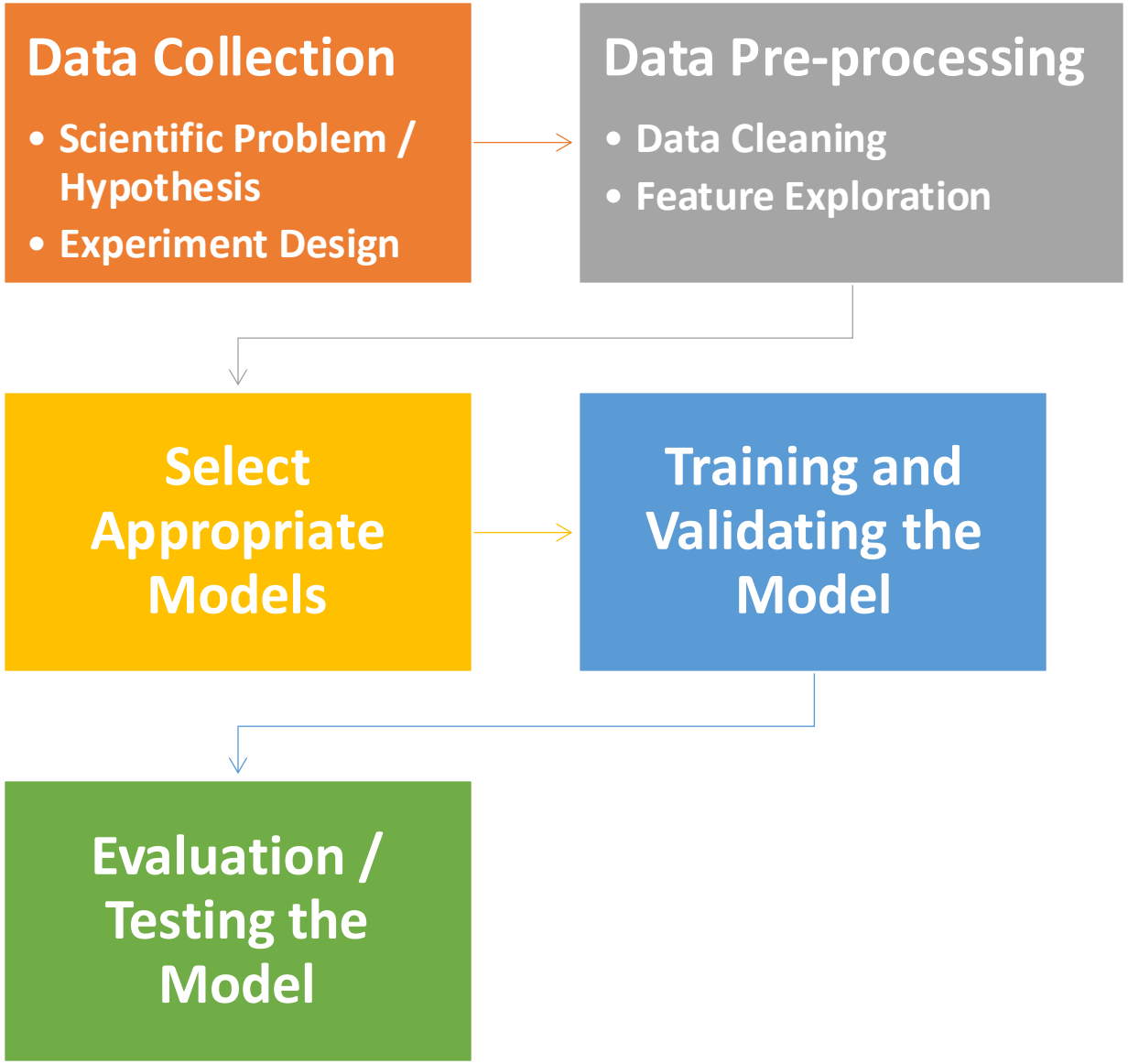
Classification



Machine Learning Workflow



Machine Learning Workflow



Data Pre-processing (80% time)

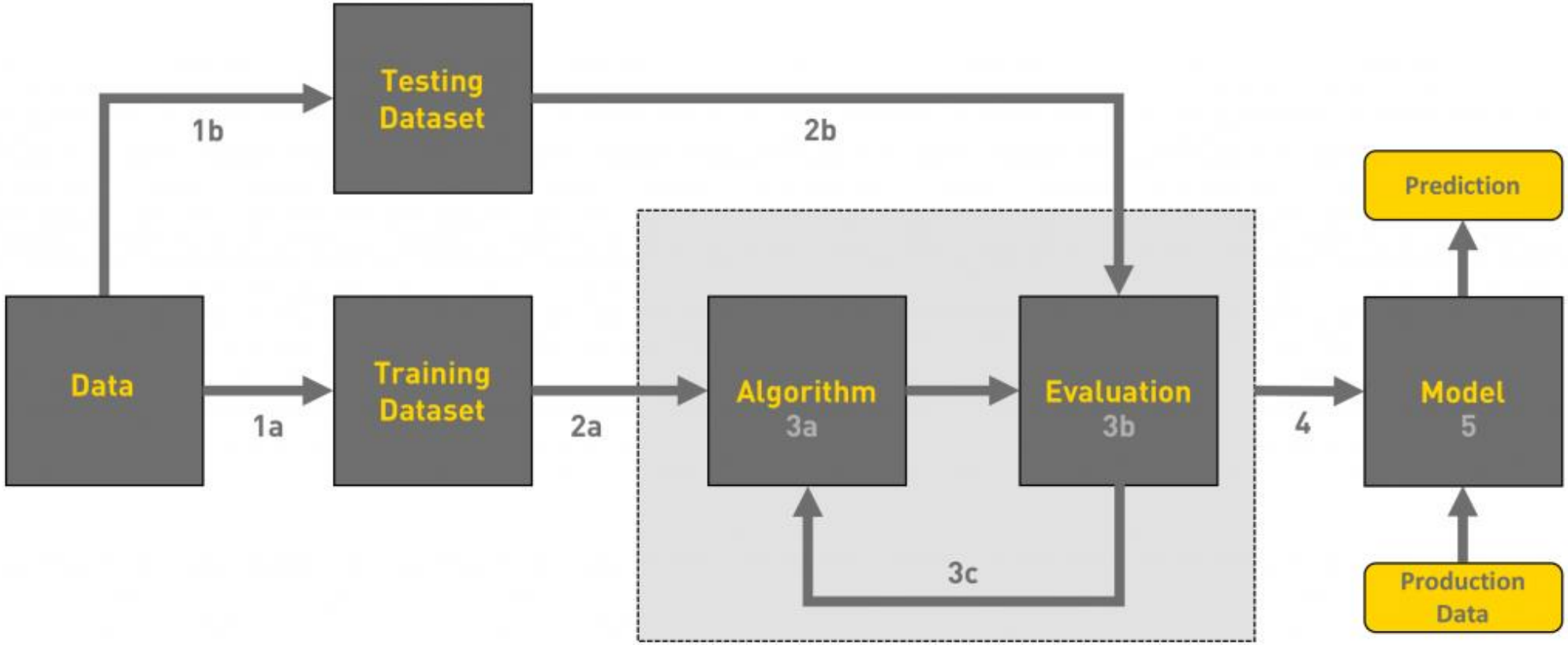
Possible data problems

- Missing data: Ignoring or Imputing?
- Noisy data: Excluding or Smoothing?
- Inconsistent data: Excluding or Correcting?
- Outliers : Excluding?

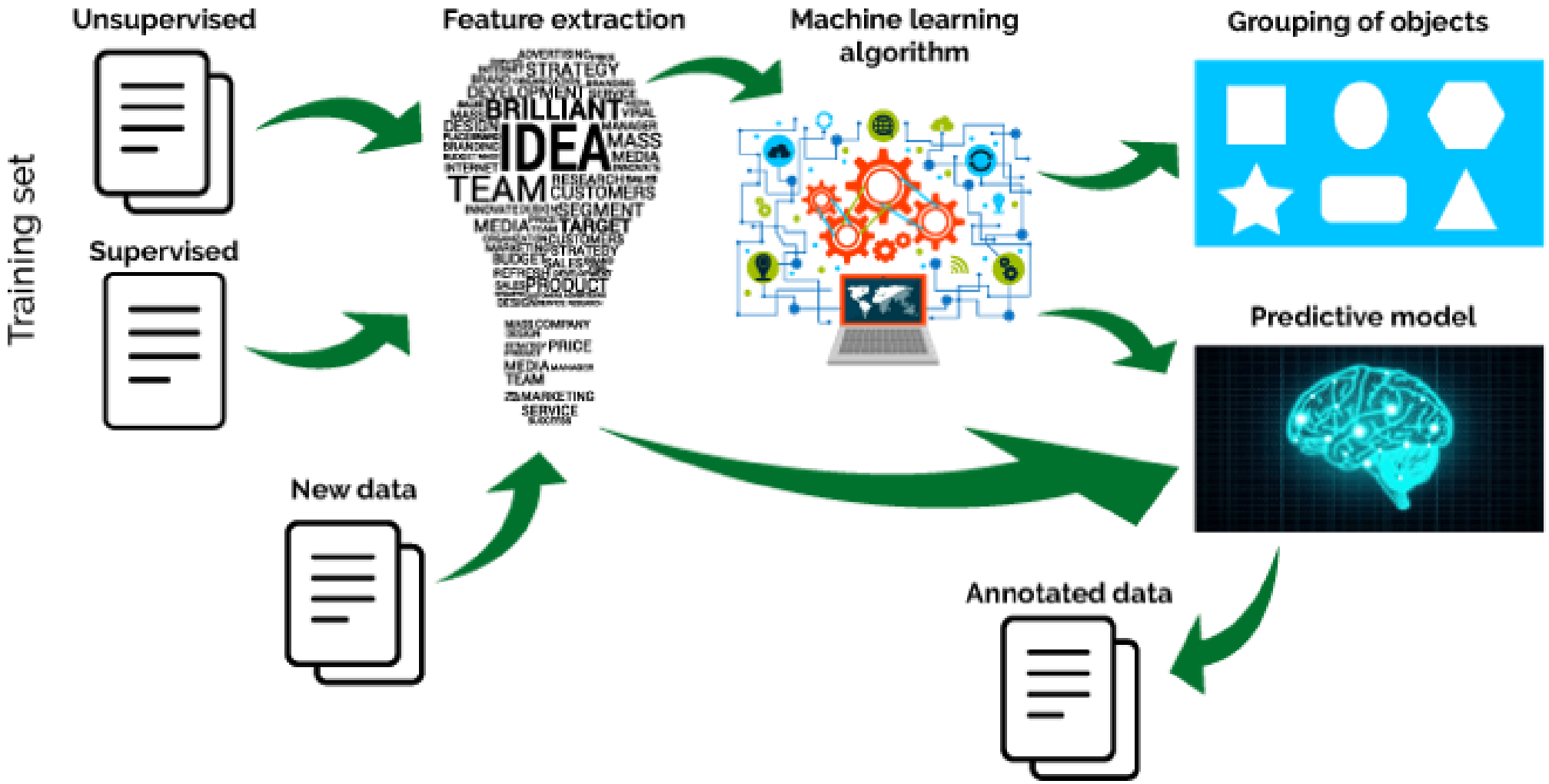
Data types

- Numeric, e.g., age, height, weight
- Categorical, e.g., gender, ethnicity; generally coded as 0/1
- Ordinal, e.g., low/medium/high; generally coded as consecutive numbers such as 0/1/2

Machine Learning Workflow

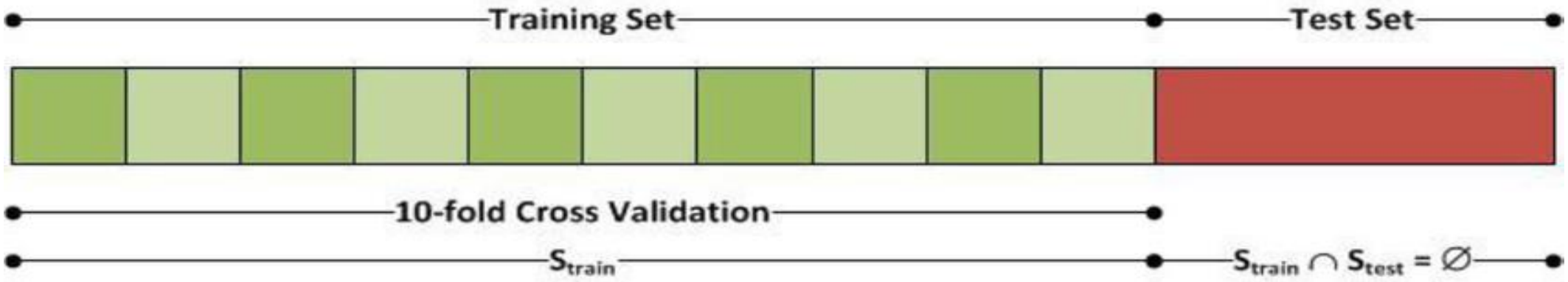


Machine Learning

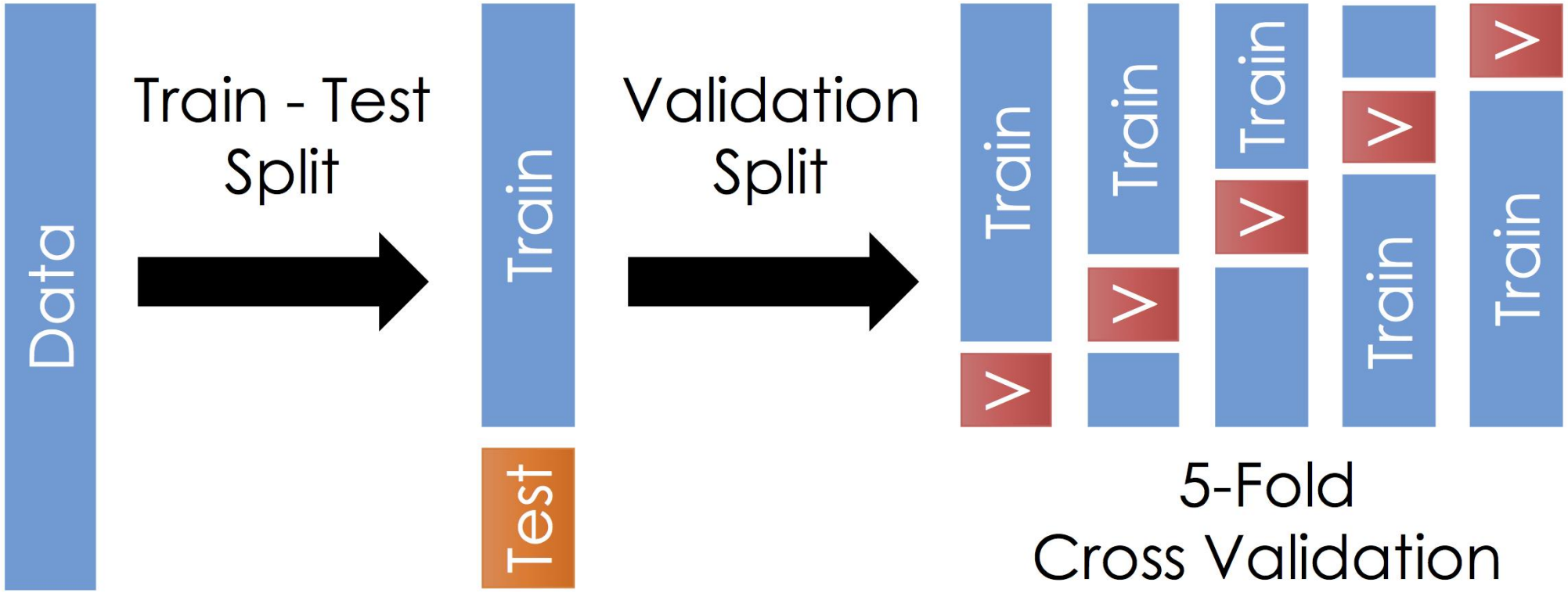


Machine Learning Data Structure

- **Training set:** Data used for learning, that is to fit the parameters of the classifier/model.
- **Validation set:** Independent data (different from the training data) used to tune the parameters of a classifier/model (cross-validation is primarily used).
- **Test set:** Independent data (different from the training and validation data) used only to assess the performance of a fully-trained classifier.



Cross Validation



Tuning Parameters & Model Selection

- **Tuning Parameters**
  - Train a set of models with respect to a range of parameters
  - Use validation data to select best parameters leading to the best performance
- **Model Selection**
  - Train multiple models with respect to different settings
    - For example, different sets of predictive features might be considered
    - Different methods/models might be considered
  - Select based on residual deviance, regression R2, or AIC based on the training data
  - <u>Preferred:</u> use test data to select a best model with best performance

Model Evaluation

- Test model performance using a test data set that is independent of the training/validation data sets
- **Evaluation criteria**
  - **Regression**
    - Mean Squared Error (MSE)

$$\frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n}$$

  - **Classification**
    - Misclassification Rate

$$\frac{FalsePositives + FalseNegatives}{N}$$

    - ROC/AUC

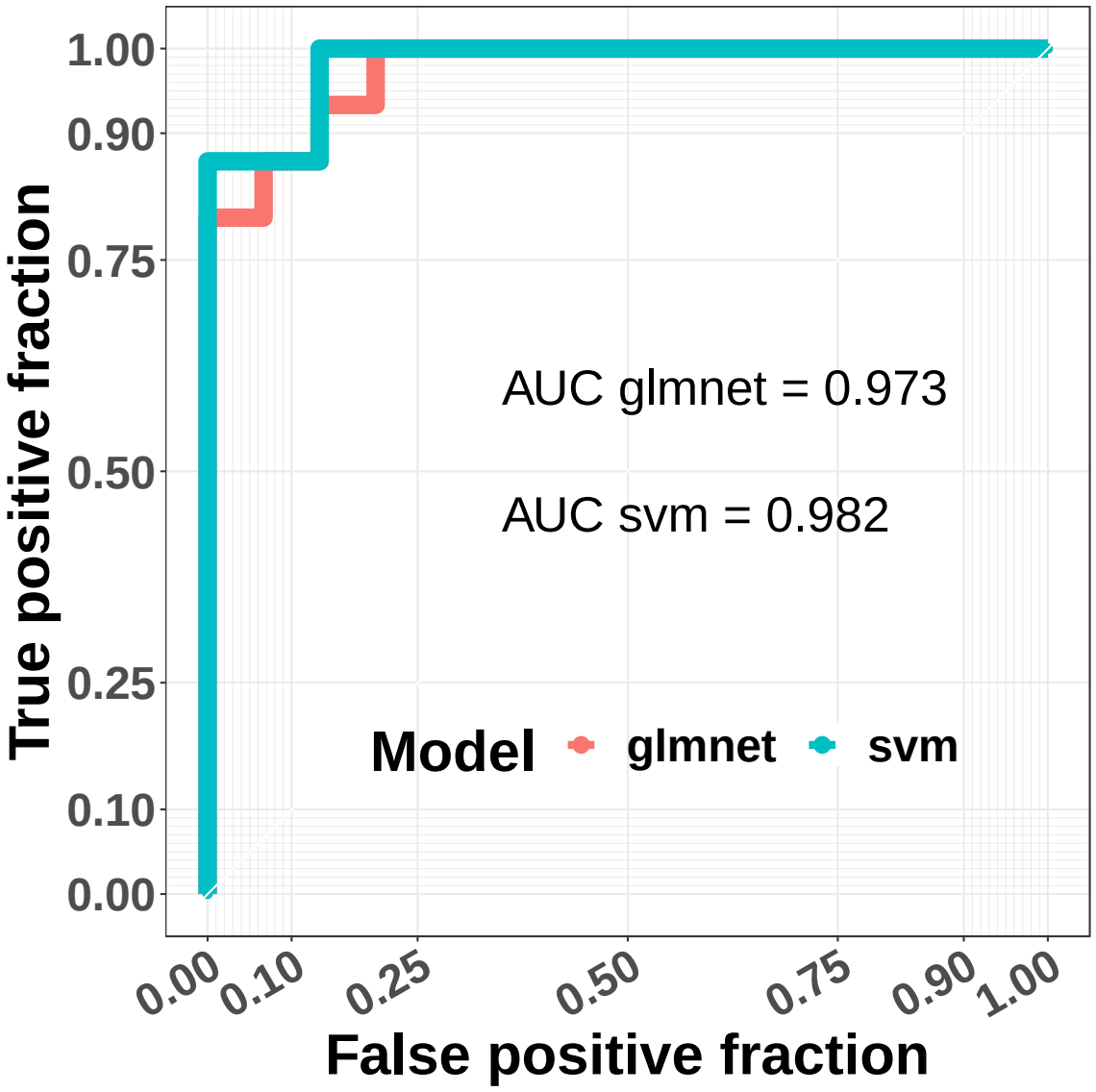
Confusion Matrix for Two-group Classification

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

ROC Curve

- **Receiver operating characteristic (ROC)** curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
- Plot **True Positive Rate** (TPR, sensitivity, recall rate, probability of detection, power) against the **False Positive Rate** (FPR, 1-specificity, probability of false alarm, type I error) **at various threshold settings**.
- **Area under the curve (AUC, C statistic),** the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative').
- https://en.wikipedia.org/wiki/Receiver_operating_characteristic

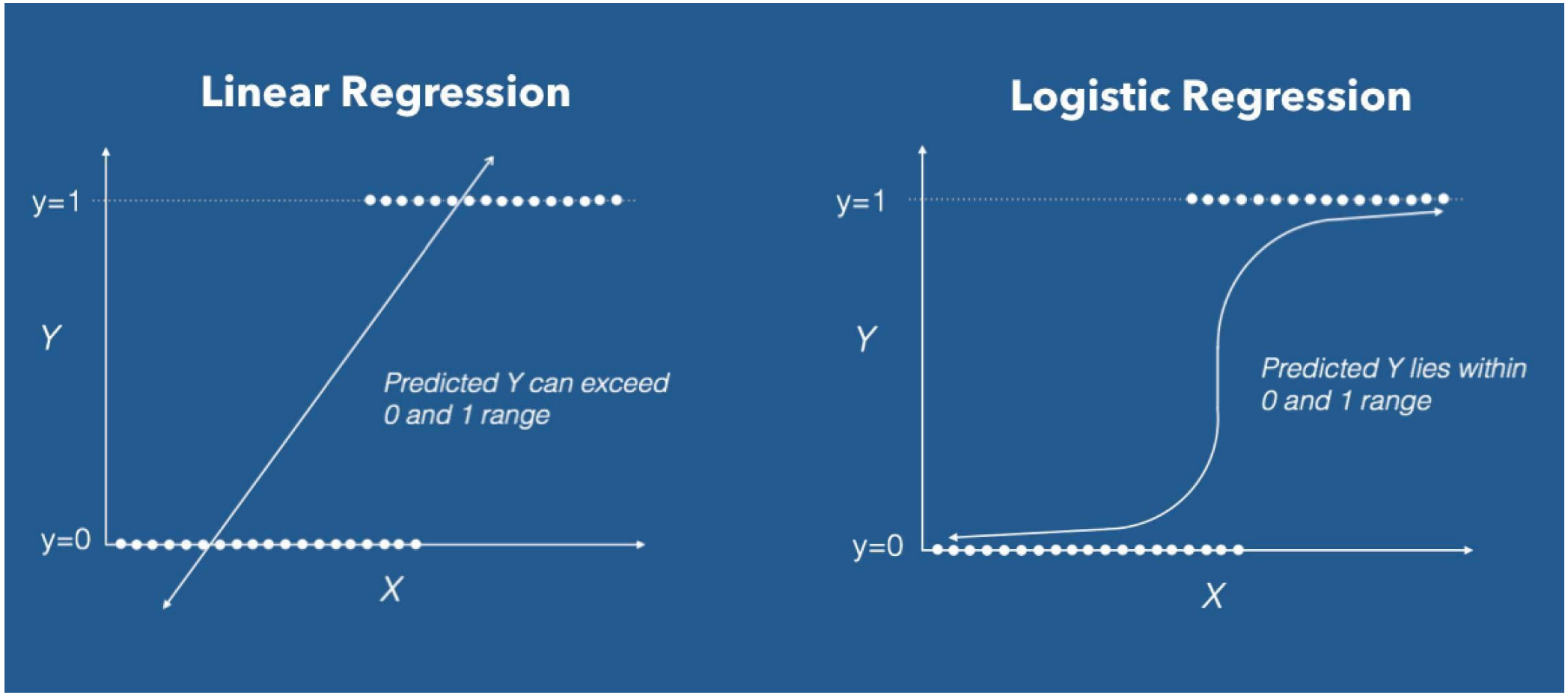
Example ROC Plot



Classification Method

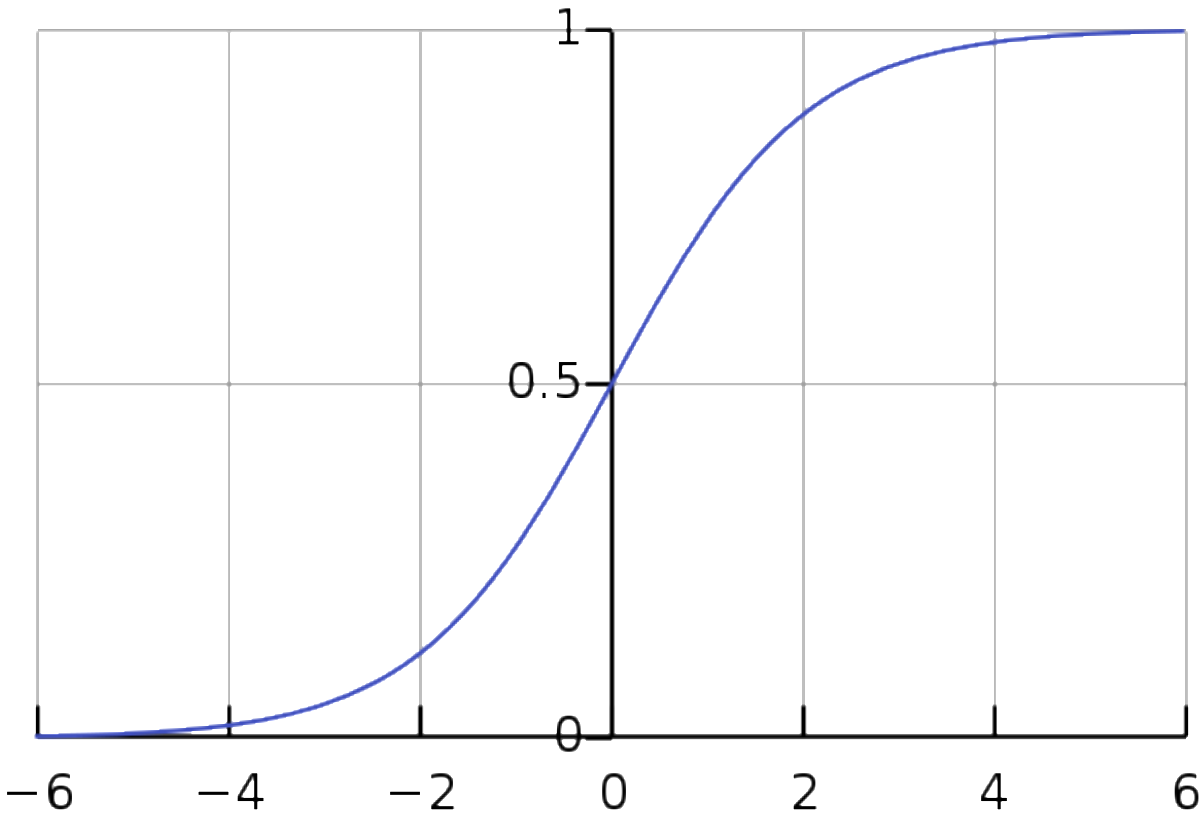
- **Logistic Regression** (Generalized linear regression model with binary responses)
  - https://en.wikipedia.org/wiki/Logistic_regression

Logistic Regression



Logistic Regression

- $l_{\text{LogOdds}} = \log\left(\frac{p}{1-p}\right) = X\beta$
- $p = Prob(Y = 1)$
- $p = \frac{1}{1+e^{-X\beta}} = \sigma(X\beta)$,
  Sigmoid function of $X\beta$



Elastic-Net Penalized Regression

- Penalized regression with a combined L1 penalty (LASSO) and L2 penalty (Ridge) on coefficients

$$\min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right],$$

- Variable Selection for using L1 penalty (LASSO)
- Account for Highly Correlated variables for using L2 penalty (Ridge)
- Need to tune penalty parameters $\boldsymbol{\lambda}, \boldsymbol{\alpha}$ by cross validation
- β_0, β will be estimated by using the above objective function for each unique pair of parameter values of $\boldsymbol{\lambda}, \boldsymbol{\alpha}$

Elastic-Net Penalized Regression

- R package “glmnet”
  - Fits a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso or elastic-net penalty at a grid of values for the regularization parameter lambda.
  - The algorithm is extremely fast, and can exploit sparsity in the input matrix x.
  - It fits linear, logistic and multinomial, Poisson, and Cox regression models.
  - https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html#top

R package *caret*

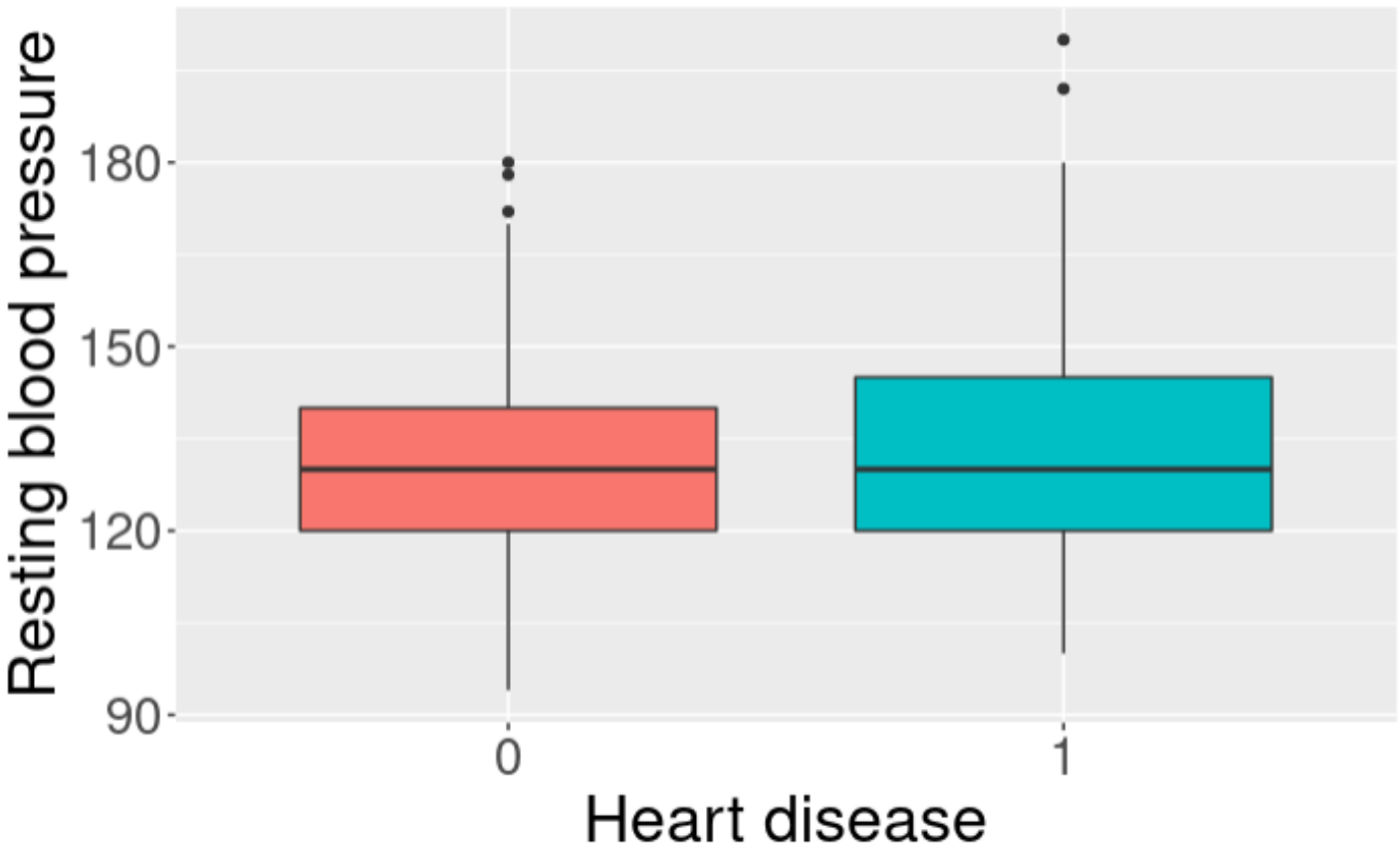
R package *caret*

- The **caret** package (**C**lassification **A**nd **RE**gression **T**raining) is a set of functions that attempt to streamline the process for creating predictive models.
- Integrates almost all Machine Learning models
- The package contains tools for:
  - data splitting (training vs. test)
  - pre-processing (quality control, imputing missing values)
  - feature selection
  - model tuning using resampling
  - variable importance estimation (R function "varImp()")
- https://topepo.github.io/caret/index.html

Example Dataset : Cleveland Heart Disease

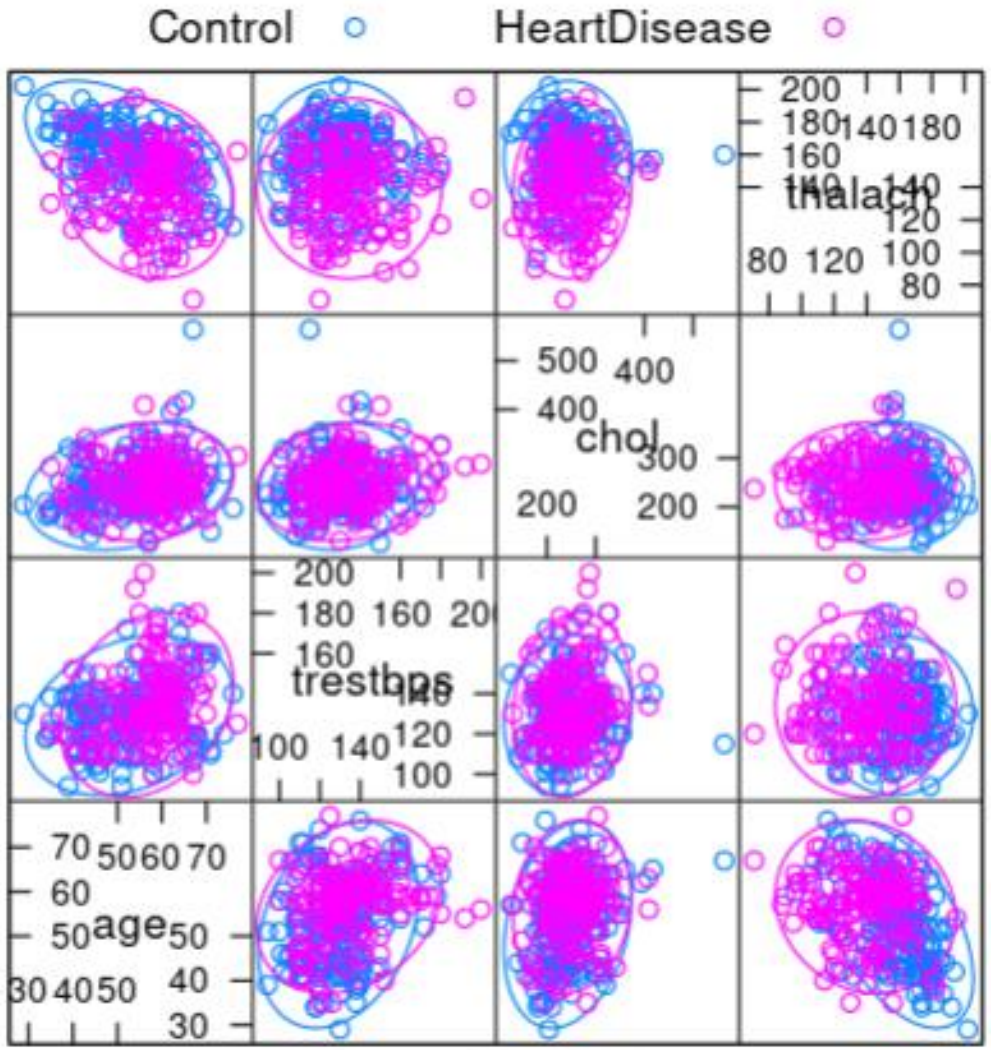
Name	Data Type	Description
age	continuous	age in years
sex	binary	1 = male; 0 = female
cp	categorical	chest pain type – 1: typical angina; 2: atypical angina; 3: non-anginal pain; 4: asymptomatic
trestbps	continuous	resting blood pressure (in mm Hg on admission to the hospital)
chol	continuous	serum cholesterol in mg/dl
fbs	binary	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg	categorical	resting electrocardiograph results – 0: normal; 1: having ST-T wave abnormality; 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
thalach	continuous	maximum heart rate achieved
exang	binary	exercise induced angina (1 = yes; 0 = no)
oldpeak	continuous	ST depression induced by exercise relative to rest
slope	categorical	the slope of the peak exercise ST segment– 1: up sloping; 2: flat; 3: down sloping
ca	continuous	number of major vessels (0-3) colored by fluoroscope
thal	categorical	Thallium heart scan – 3 = normal; 6 = fixed defect; 7 = reversible defect
disease	categorical	absence (0) vs. presence (1, 2, 3, 4)

Study the relationship between resting blood pressure would affect heart disease presence



Lattice Graph with Four Continuous Variables

age
resting blood pressure (trestbps)
cholesterol (chol)
maximum heart rate (thalach)



Scatter Plot Matrix

Partition Training and Test Data

Data splitting

```
## Exclude samples with NAs
dim(cleveland)
```

```
## [1] 303  15
```

```
cleveland <- na.omit(cleveland)
dim(cleveland)
```

```
## [1] 297  15
```

```
## Select training data by sample indexes
set.seed(2021)
trainIndex_2class <- createDataPartition(cleveland$HD, p = .7,
                                         list = FALSE,
                                         times = 1)
head(trainIndex_2class)
```

Resample1
1
2
3
4
5
7

Setup Arguments for Model Training

```
## set model training parameters
fitControl <- trainControl(## 5-fold CV
                           method = "cv",
                           number = 5,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = twoClassSummary)
```

Train the classification model by "glmnet" method

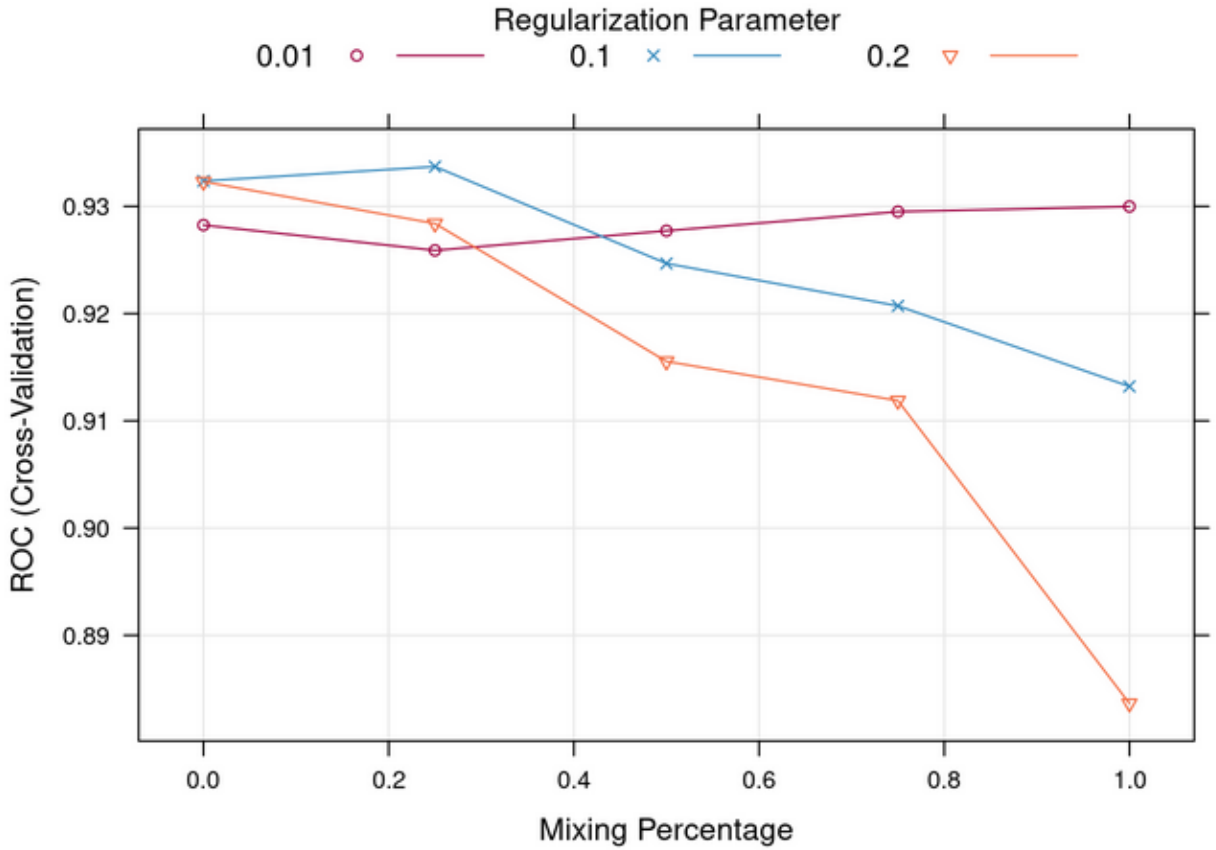
```
## Train the classification model by "glmnet" method
glmnet.fit <- train(HD ~ age + sex + cp + trestbps + chol +
                        fbs + restecg + thalach + exang + oldpeak +
                        slope + ca + thal , data = cleveland[trainIndex_2class, ],
                    method = "glmnet",
                    trControl = fitControl,
                    preProc = c("center", "scale"),
                    ## set tuning parameter grid
                    tuneGrid = expand.grid(alpha = seq(0, 1, length.out = 5),
                           lambda = c(0.01, 0.1, 0.2)),
                    ## Specify which metric to optimize
                    metric = "ROC")
print(glmnet.fit)
```

Trained classification model by "glmnet" method

```
## glmnet
##
## 208 samples
##  13 predictor
##   2 classes: 'Control', 'HeartDisease'
##
## Pre-processing: centered (18), scaled (18)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 167, 165, 167, 167, 166
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0.00   0.01    0.9282505  0.8857708  0.8331579
##   0.00   0.10    0.9323653  0.8853755  0.8326316
##   0.00   0.20    0.9322987  0.8940711  0.8010526
##   0.25   0.01    0.9258956  0.8857708  0.8331579
##   0.25   0.10    0.9337071  0.8940711  0.7910526
##   0.25   0.20    0.9283961  0.9205534  0.7805263
##   0.50   0.01    0.9277221  0.8857708  0.8331579
##   0.50   0.10    0.9246744  0.8944664  0.7910526
##   0.50   0.20    0.9155419  0.9122530  0.7600000
##   0.75   0.01    0.9295070  0.8857708  0.8331579
##   0.75   0.10    0.9207177  0.9035573  0.7910526
##   0.75   0.20    0.9118848  0.9296443  0.6652632
##   1.00   0.01    0.9299854  0.8857708  0.8331579
##   1.00   0.10    0.9132120  0.8948617  0.7600000
##   1.00   0.20    0.8836530  0.9470356  0.4689474
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.25 and lambda = 0.1.
```

Parameter Tuning Results

```
## Plot tuning results
trellis.par.set(caretTheme())
plot(glmnet.fit)
```




```
## Best tuned parameters
glmnet.fit$bestTune
```

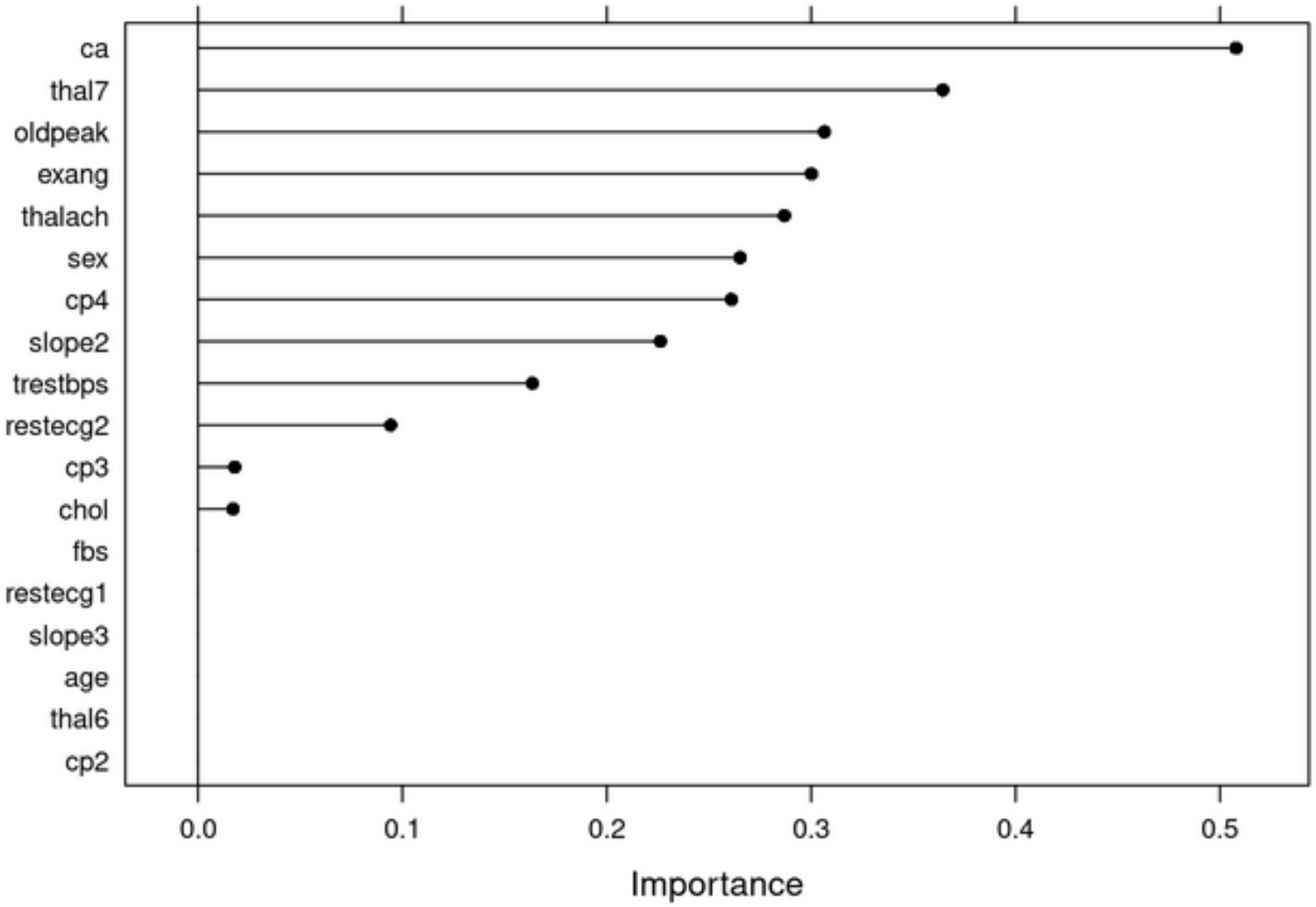
	alpha	lambda
5	0.25	0.1

Predictor Importance

```r
## Plot important predictors
roc_imp1 <- varImp(glmnet.fit, scale = FALSE)
roc_imp1
```

```
## glmnet variable importance
##
##          Overall
## ca       0.50793
## thal7    0.36437
## oldpeak  0.30643
## exang    0.30013
## thalach  0.28687
## sex      0.26519
## cp4      0.26093
## slope2   0.22626
## trestbps 0.16359
## restecg2 0.09428
## cp3      0.01793
## chol     0.01717
## age      0.00000
## fbs      0.00000
## cp2      0.00000
## restecg1 0.00000
## slope3   0.00000
## thal6    0.00000
```

```r
plot(roc_imp1)
```



Prediction in Test Data

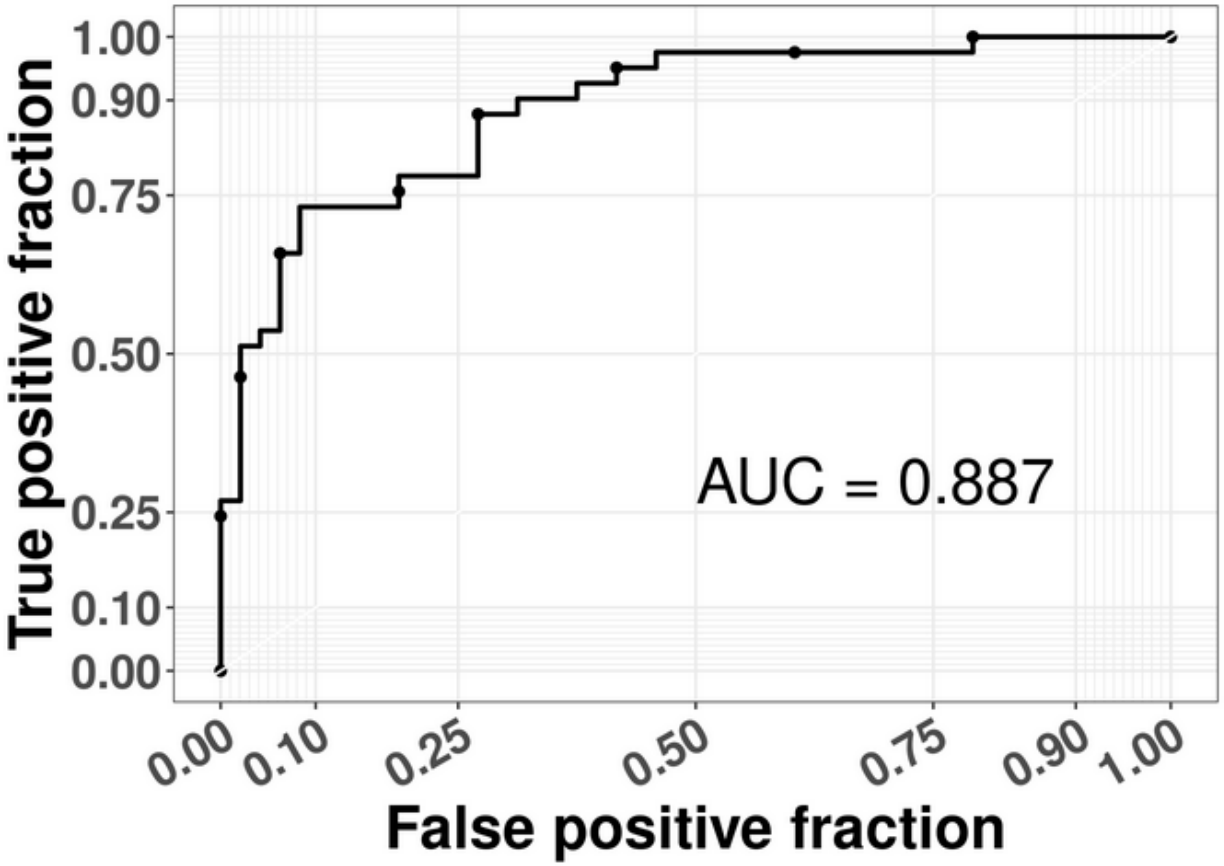
```
true.class <- cleveland$HD[-trainIndex_2class]
head(true.class)
```

```
## [1] Control      Control      HeartDisease HeartDisease Control
## [6] HeartDisease
## Levels: Control HeartDisease
```

```
## Predict labels
pred.class.glmnet <- predict(glmnet.fit, newdata = cleveland[-trainIndex_2class, ])
### Confusion Matrix
confusionMatrix(pred.class.glmnet, true.class)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction     Control HeartDisease
##   Control           44           11
##   HeartDisease       4           30
##
##                Accuracy : 0.8315
##                  95% CI : (0.7373, 0.9025)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 6.345e-09
##
##                   Kappa : 0.6565
##
##  Mcnemar's Test P-Value : 0.1213
##
##             Sensitivity : 0.9167
##             Specificity : 0.7317
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8824
##              Prevalence : 0.5393
##          Detection Rate : 0.4944
##    Detection Prevalence : 0.6180
##       Balanced Accuracy : 0.8242
##
##        'Positive' Class : Control
##
```

ROC Plot for Prediction Results



In-Class Activity

Unsupervised Learning

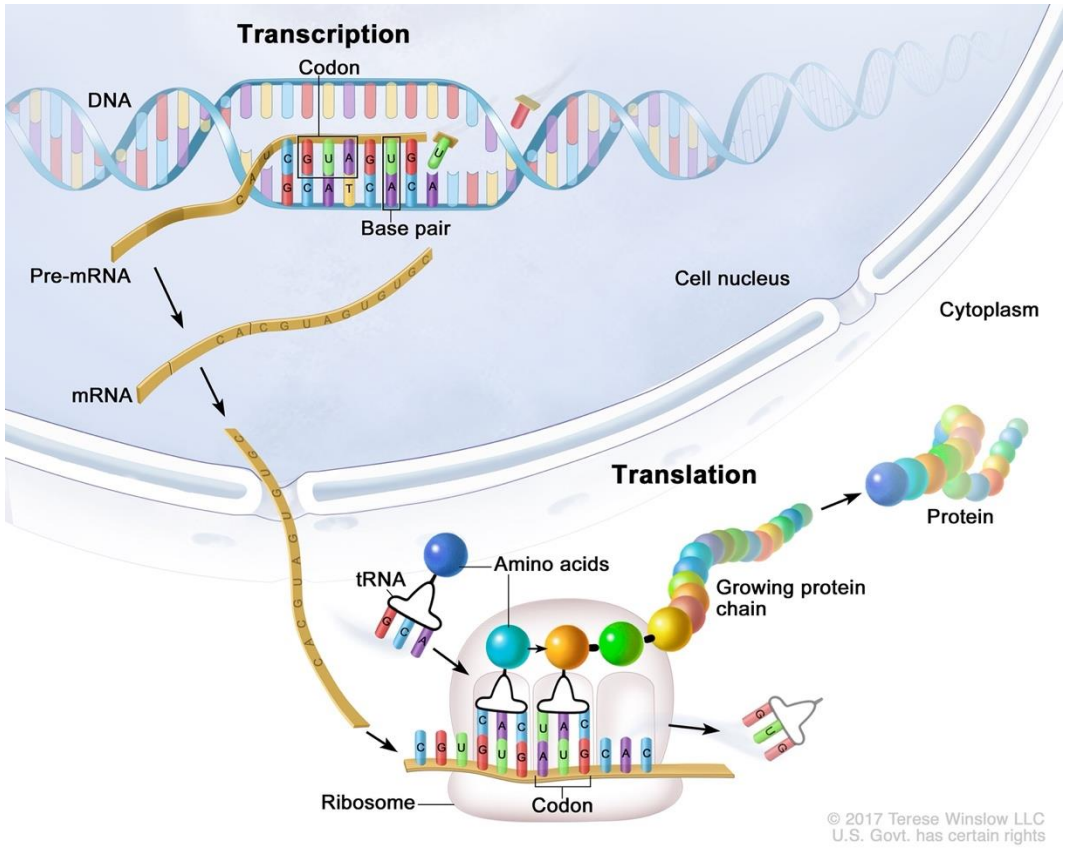
Biomedical Research Problems

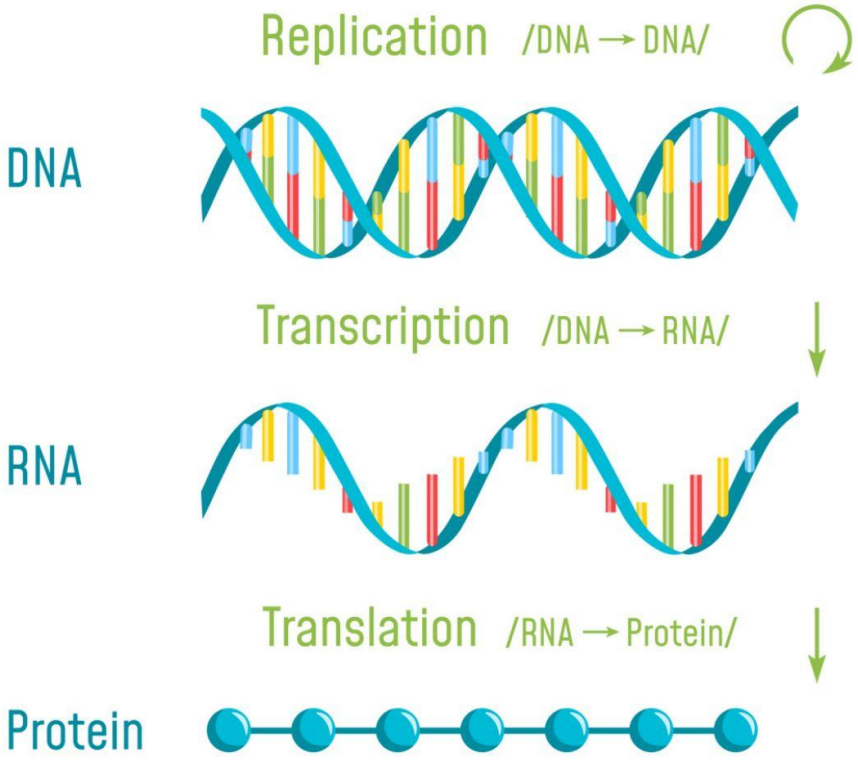
- Data quality assessment
  - Clustering samples according to their ancestry
  - Clustering sequence samples
- Clustering genes or samples using bulk RNAseq data
- Clustering single cells using single cell RNAseq (scRNAseq) data

RNA-Seq Data

- Gene expression Quantitative Traits
  - Profiled by RNA sequencing (RNA-seq)
  - CPM (Counts Per Million) per gene
    - Count up all the read counts in a sample (library size) and divide this number by 1,000,000. This is your “per million” scaling factor.
    - Divide the read counts per gene by the “per million” scaling factor. This gives you CPM.
- 20K ~ 25K genes in human genome
- Bulk RNA-seq; single cell RNAseq

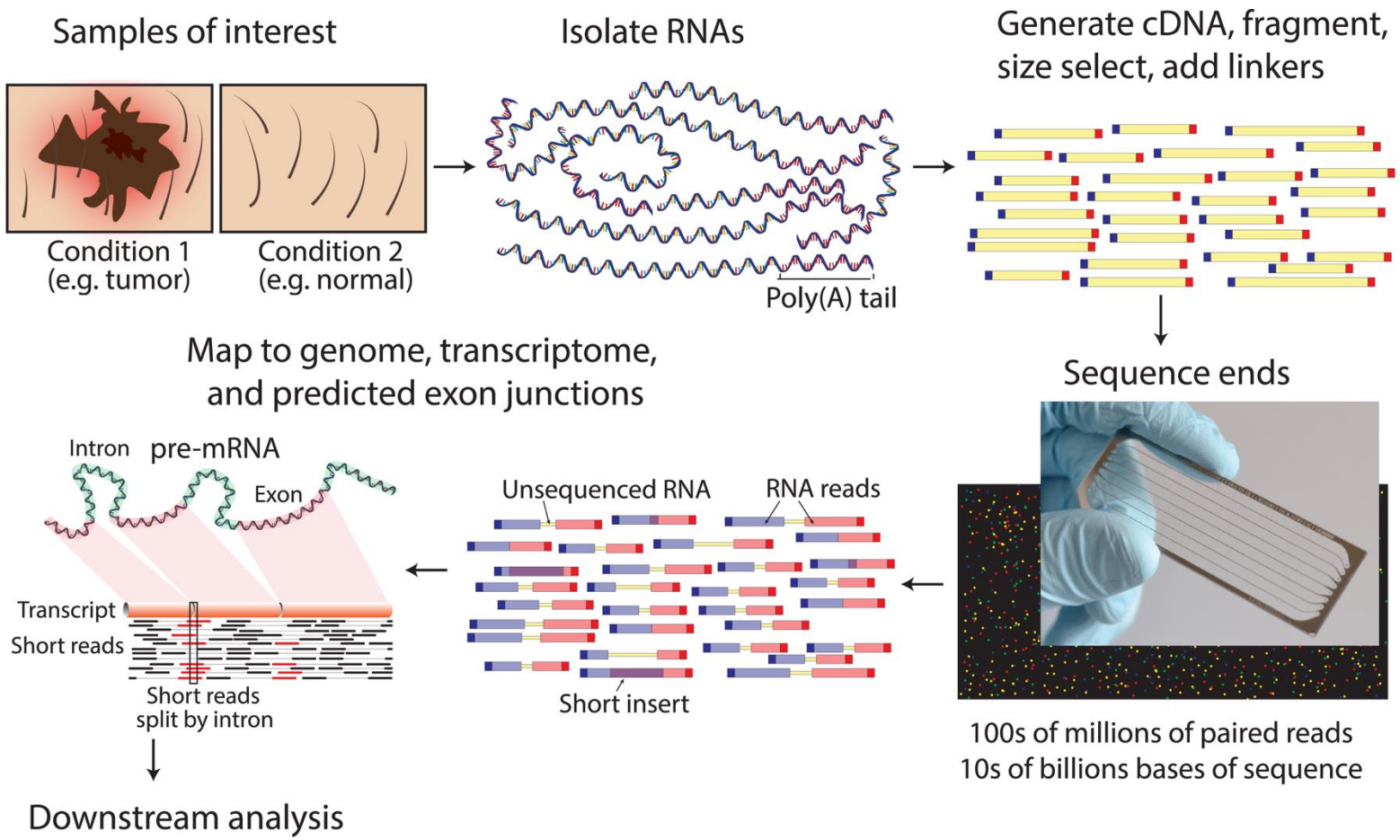
Transcription





https://www.thoughtco.com/dna-transcription-373398

Profile Gene Expression Levels by RNA-sequencing



Example RNA-Seq Data

- Example RNA-Seq data from: David K. Lau et. al., 2019. Genomic Profiling of Biliary Tract Cancer Cell Lines Reveals Molecular Subtypes and Actionable Drug Targets. PMID: 31731200 ; DOI: 10.1016/j.isci.2019.10.044
- Samples from 20 biliary track cancer cell lines were profiled for gene expression data by RNA sequencing
- 24222 genes in the raw data

Inspecting Raw RNA-Seq Data

```
head(RNAseq_dt)
```

```
##      RefSeq EGI1 G415 HUCCT1 HUH28 MZCHA2  NOZ OCUG1   OZ SKCHA1 SNU1079
## 1 NM_000014    1    0     10    25   8935    1     0    1      3       0
## 2 NM_000015    4    0      1     0      5    0     0    0      2       3
## 3 NM_000017  120  154    132    46    240  163    47  188    195     293
## 4 NM_000019  444 1246    467   426    350 1245   470  286    783     843
## 5 NM_000020    0  250      1     0     85    4     0    0      0      39
## 6 NM_000021 2373 1989   2648   796   1083  958   933 1539   2454    1555
##   SNU1196 SNU245 SNU308 SNU478 SNU869 TFK1 TGBC14TKB TGBC18TKB TGBC2TKB TKKK
## 1       6      0      0     58      2    0         3         1      185    0
## 2       1      2      1      1      0    2         4         1        2    5
## 3     411    497    211    160    161  369       354       212       99  344
## 4     418    175    586    854    712  740       751       604      451 1034
## 5       2      0      2      2      5    4        27         1        0    0
## 6    1618   1539   1421   1282   1525 2151      1158      1590      804 1339
```

Normalize Raw RNA-Seq Data

```
apply(RNAseq_matrix, 2, sum)
```

```
##      EGI1      G415    HUCCT1     HUH28    MZCHA2       NOZ     OCUG1        OZ
##  12480812  14289644  11524427  14247144  12296380  13698008  13120204  11866325
##    SKCHA1   SNU1079   SNU1196    SNU245    SNU308    SNU478    SNU869      TFK1
##  13501752  11876625  12732950  10469013  13972069  11309785  12500489  15414668
## TGBC14TKB TGBC18TKB  TGBC2TKB      TKKK
##  13224759  10735498  11619162  11487514
```

- Summarizing read counts per column/sample gives us the **library size**. The total number of mapped read counts per sample.
- Various library sizes make the raw read counts per gene are not comparable across all samples/cell-lines.
- Need to **Normalize** read counts to Counts Per Million (CPM)

Get RNA-Seq Data in CPM

```
class(RNAseq_matrix)
```

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

```
RNAseq_CPM <- cpm(RNAseq_matrix)
class(RNAseq_CPM)
```

```
## [1] "matrix" "array"
```

```
head(RNAseq_CPM)
```

```
##                   EGI1      G415       HUCCT1     HUH28      MZCHA2         NOZ
## NM_000014   0.08012299   0.00000   0.86772210  1.754738 726.6366199  0.07300332
## NM_000015   0.32049197   0.00000   0.08677221  0.000000   0.4066237  0.00000000
## NM_000017   9.61475904  10.77704  11.45393172  3.228717  19.5179394 11.89954043
## NM_000019  35.57460845  87.19601  40.52262208 29.900730  28.4636617 90.88912782
## NM_000020   0.00000000  17.49519   0.08677221  0.000000   6.9126035  0.29201326
## NM_000021 190.13186001 139.19171 229.77281213 55.870847  88.0747017 69.93717627
##               OCUG1           OZ      SKCHA1    SNU1079     SNU1196    SNU245
## NM_000014  0.000000   0.08427209   0.2221934   0.000000   0.4712184   0.00000
## NM_000015  0.000000   0.00000000   0.1481289   0.252597   0.0785364   0.19104
## NM_000017  3.582261  15.84315279  14.4425701  24.670308  32.2784586  47.47343
## NM_000019 35.822614  24.10181754  57.9924739  70.979761  32.8282134  16.71600
## NM_000020  0.000000   0.00000000   0.0000000   3.283761   0.1570728   0.00000
## NM_000021 71.111699 129.69474542 181.7541901 130.929452 127.0718883 147.00526
##                 SNU308       SNU478      SNU869        TFK1  TGBC14TKB
## NM_000014   0.00000000   5.12830262   0.1599937   0.0000000  0.2268472
## NM_000015   0.07157136   0.08841901   0.0000000   0.1297466  0.3024630
## NM_000017  15.10155726  14.14704170  12.8794962  23.9382386 26.7679736
## NM_000019  41.94081778  75.50983507  56.9577718  48.0062237 56.7874243
## NM_000020   0.14314272   0.17683802   0.3999844   0.2594931  2.0416251
## NM_000021 101.70290456 113.35317161 121.9952275 139.5424151 87.5630323
##              TGBC18TKB   TGBC2TKB        TKKK
## NM_000014   0.09314892 15.9219744   0.0000000
## NM_000015   0.09314892  0.1721295   0.4352552
## NM_000017  19.74757016  8.5204079  29.9455565
## NM_000019  56.26194518 38.8151917  90.0107717
## NM_000020   0.09314892  0.0000000   0.0000000
## NM_000021 148.10677623 69.1960401 116.5613378
```

```
apply(RNAseq_CPM, 2, sum)
```

```
##      EGI1      G415    HUCCT1     HUH28    MZCHA2       NOZ     OCUG1        OZ
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##    SKCHA1   SNU1079   SNU1196    SNU245    SNU308    SNU478    SNU869      TFK1
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
## TGBC14TKB TGBC18TKB  TGBC2TKB      TKKK
##     1e+06     1e+06     1e+06     1e+06
```

Data Cleaning : filtering out genes with low CPM

- Low read counts are more likely to add noises.
- As a general rule, a good threshold can be chosen for a CPM value that corresponds to 10 raw read counts.

Library Size	Count	CPM
1M	1	1
10M	**10**	**1**
20M	20	1

Data Cleaning : filtering out genes with low CPM in any samples

```
thresh <- RNAseq_CPM > 1
class(thresh)
```

```
## [1] "matrix" "array"
```

```
head(thresh)
```

```
##            EGI1  G415 HUCCT1 HUH28 MZCHA2   NOZ OCUG1    OZ SKCHA1 SNU1079
## NM_000014 FALSE FALSE  FALSE  TRUE   TRUE FALSE FALSE FALSE  FALSE   FALSE
## NM_000015 FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE FALSE  FALSE   FALSE
## NM_000017  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
## NM_000019  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
## NM_000020 FALSE  TRUE  FALSE FALSE   TRUE FALSE FALSE FALSE  FALSE    TRUE
## NM_000021  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
##           SNU1196 SNU245 SNU308 SNU478 SNU869  TFK1 TGBC14TKB TGBC18TKB
## NM_000014   FALSE  FALSE  FALSE   TRUE  FALSE FALSE     FALSE     FALSE
## NM_000015   FALSE  FALSE  FALSE  FALSE  FALSE FALSE     FALSE     FALSE
## NM_000017    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
## NM_000019    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
## NM_000020   FALSE  FALSE  FALSE  FALSE  FALSE FALSE      TRUE     FALSE
## NM_000021    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
##           TGBC2TKB  TKKK
## NM_000014     TRUE FALSE
## NM_000015    FALSE FALSE
## NM_000017     TRUE  TRUE
## NM_000019     TRUE  TRUE
## NM_000020    FALSE FALSE
## NM_000021     TRUE  TRUE
```

```
RNAseq_CPM.keep <- RNAseq_CPM[keep,]
class(RNAseq_CPM.keep)
```

```
## [1] "matrix" "array"
```

```
dim(RNAseq_CPM.keep)
```

```
## [1] 10034    20
```

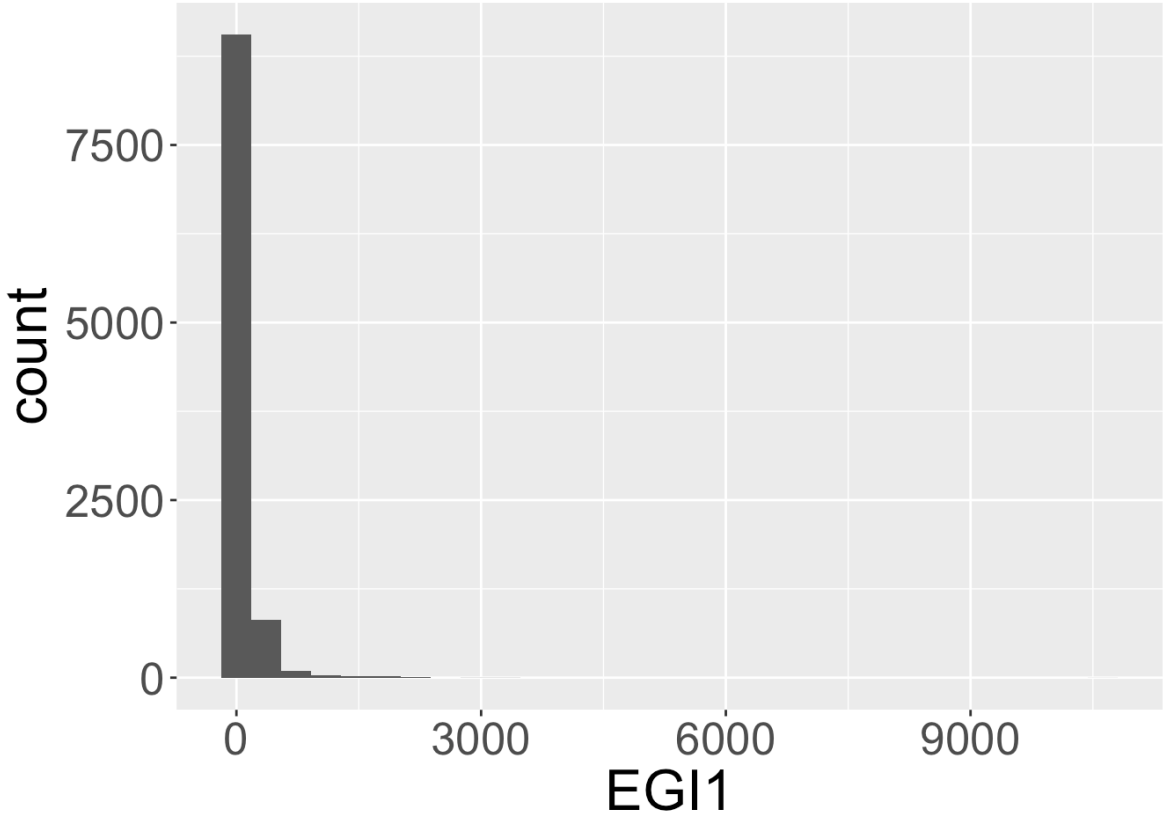
```
head(RNAseq_CPM.keep)
```

```
##                EGI1       G415    HUCCT1     HUH28    MZCHA2      NOZ
## NM_000017   9.614759  10.777035  11.45393  3.228717  19.51794 11.89954
## NM_000019  35.574608  87.196014  40.52262 29.900730  28.46366 90.88913
## NM_000021 190.131860 139.191711 229.77281 55.870847  88.07470 69.93718
## NM_000026  43.747154  77.958555  52.49719 48.430759  53.18638 89.79408
## NM_000027  17.226443   7.557921  21.17242 17.687756  19.19264 14.67367
## NM_000028  49.836501  52.975428  32.10572 58.116911 114.26127 13.87063
##               OCUG1        OZ    SKCHA1   SNU1079   SNU1196    SNU245    SNU308
## NM_000017  3.582261  15.84315  14.44257  24.67031  32.27846  47.47343  15.10156
## NM_000019 35.822614  24.10182  57.99247  70.97976  32.82821  16.71600  41.94082
## NM_000021 71.111699 129.69475 181.75419 130.92945 127.07189 147.00526 101.70290
## NM_000026 71.797664  48.96208 108.57850  67.44340  78.77200  48.33311  44.37424
## NM_000027 35.060430  29.15814  47.77158  26.01749  13.82241  13.18176  12.31027
## NM_000028 46.035870  35.05719  46.51248  34.35319  26.38823  34.86480 646.71882
##              SNU478    SNU869      TFK1 TGBC14TKB TGBC18TKB  TGBC2TKB      TKKK
## NM_000017  14.14704  12.87950  23.93824  26.76797  19.74757  8.520408  29.94556
## NM_000019  75.50984  56.95777  48.00622  56.78742  56.26195 38.815192  90.01077
## NM_000021 113.35317 121.99523 139.54242  87.56303 148.10678 69.196040 116.56134
## NM_000026 138.19891 108.95574  68.18181  39.62265  44.89778 49.745412  35.69092
## NM_000027  18.56799  14.87942  22.57590  14.14014  59.14956 46.561017  17.58431
## NM_000028  65.87216  39.75844  72.52832  57.24112  28.13097 89.507316  54.58100
```

Data Visualization : Histogram plot per sample

```
ggplot(data.frame(RNAseq_CPM.keep), aes(x = EGI1)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Normally distributed?

Log2 Transform

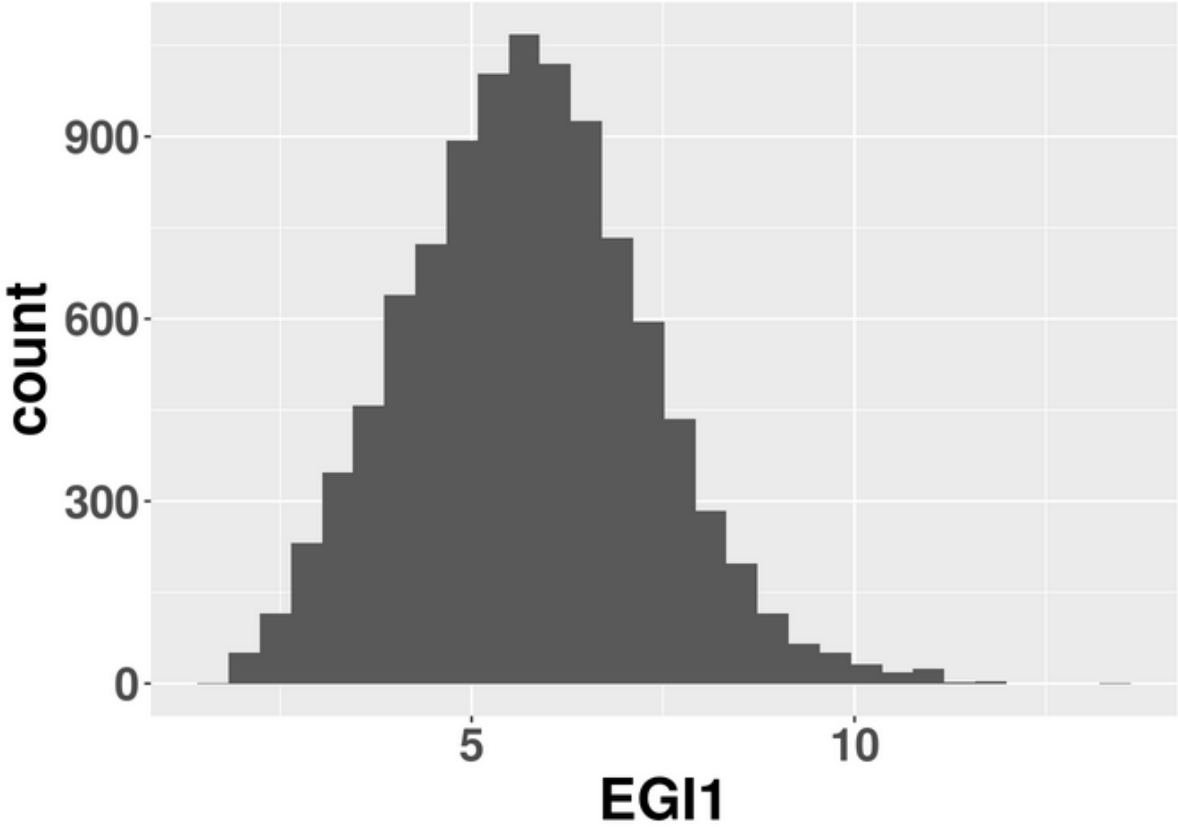
```
RNAseq_CPM.keep.log2 <- cpm(RNAseq_CPM.keep, log = TRUE)
head(RNAseq_CPM.keep.log2)
```

	EGI1	G415	HUCCT1	HUH28	MZCHA2	NOZ	OCUG1	OZ	SKCHA1	SNU1079
NM_000017	3.748058	3.816137	3.965549	2.681242	4.640084	3.918829	2.679221	4.340983	4.246274	4.929370
NM_000019	5.443631	6.610270	5.627865	5.338095	5.142077	6.648118	5.439834	4.889197	6.114535	6.381345
NM_000021	7.798535	7.272300	8.075071	6.201472	6.706817	6.279921	6.390711	7.223192	7.729691	7.246363
NM_000026	5.727696	6.452723	5.986030	6.002192	5.999779	6.631033	6.404184	5.853875	6.996910	6.309694
NM_000027	4.476208	3.400990	4.751226	4.636076	4.618084	4.179182	5.410464	5.144458	5.845004	5.000441
NM_000028	5.908059	5.913048	5.309406	6.256596	7.075075	4.108523	5.784708	5.394439	5.808027	5.376097

Data Transformation: Log2

```
ggplot(data.frame(RNAseq_CPM.keep.log2), aes(x = EGI1)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



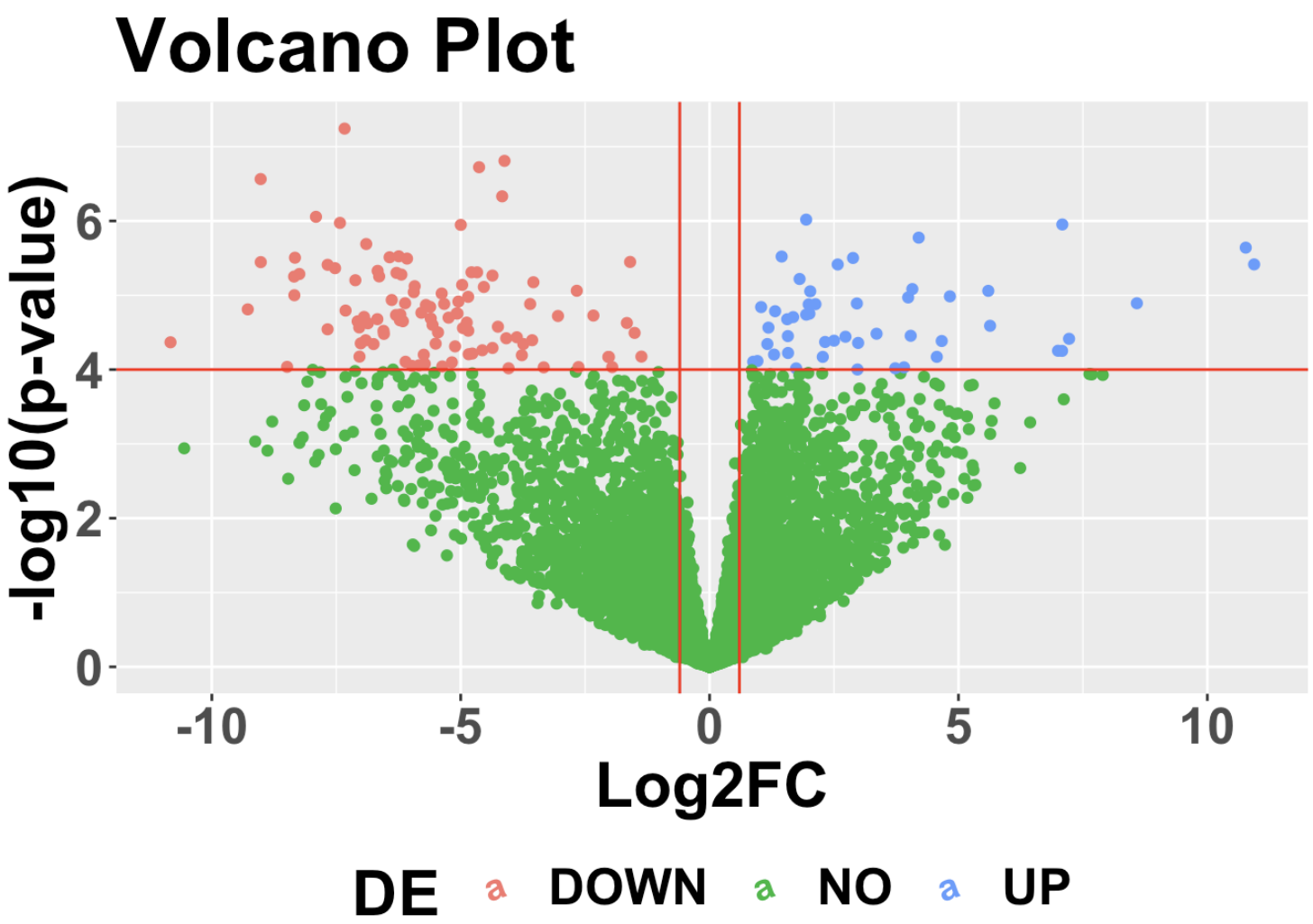
Normally distributed?

Unsupervised Learning

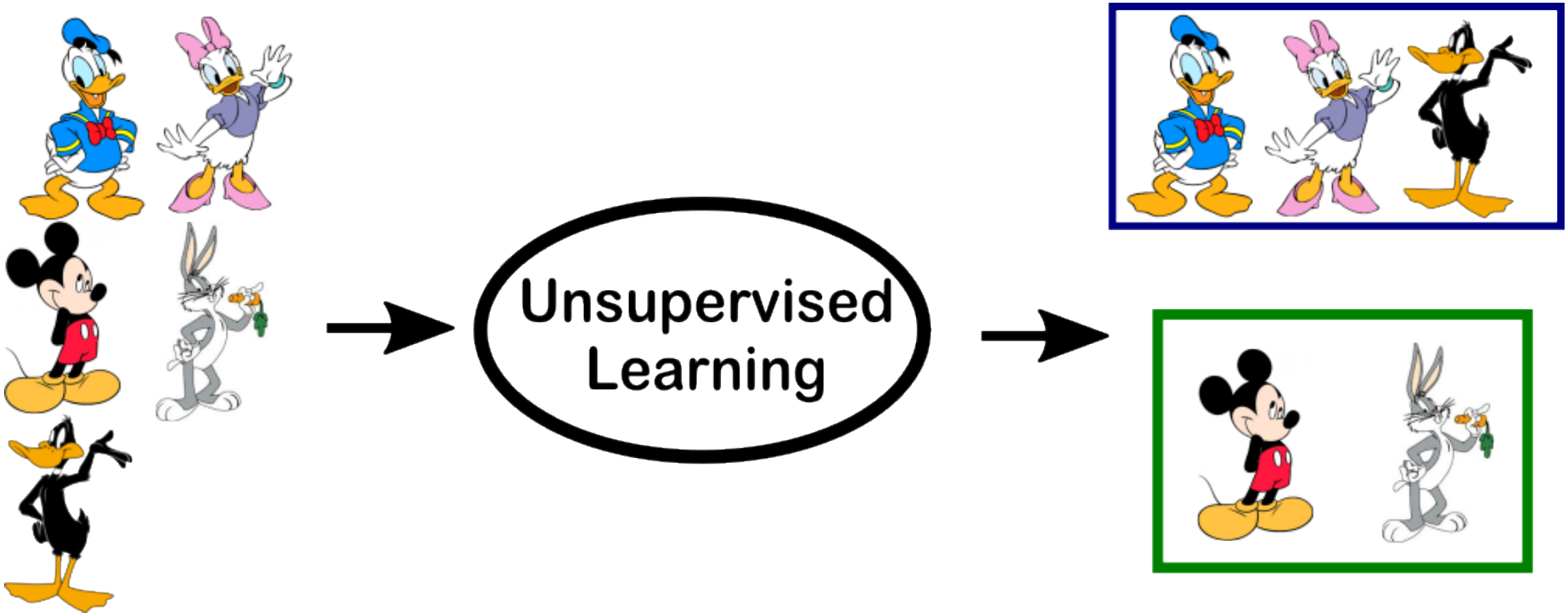
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data
  - Test if people with genetic variation X are more likely to have disease Y
  - Test if a treatment will be effective in clinical trials

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data
  - Grouping cells with respect different characteristics

Association Study: Differential Gene Expression Analysis



Clustering : Ducks vs. Not Ducks



Clustering Methods

1. **Hierarchical Clustering**
   - Build a hierarchy from the bottom-up and doesn't require us to specify the number of clusters beforehand.
   - Put each data point in its own cluster.
   - Identify the closest two clusters and combine them into one cluster.
   - Repeat the above step till all the data points are in a single cluster.
2. **Uniform Manifold Approximation and Projection (UMAP)**
   - Dimension reduction. Projecting high dimensional features to 2-dimension
   - Competitive with t-SNE method. UMAP preserves more of the global structure with superior run time performance.
   - Widely used in single cell RNAseq studies

1.Hierarchical Clustering

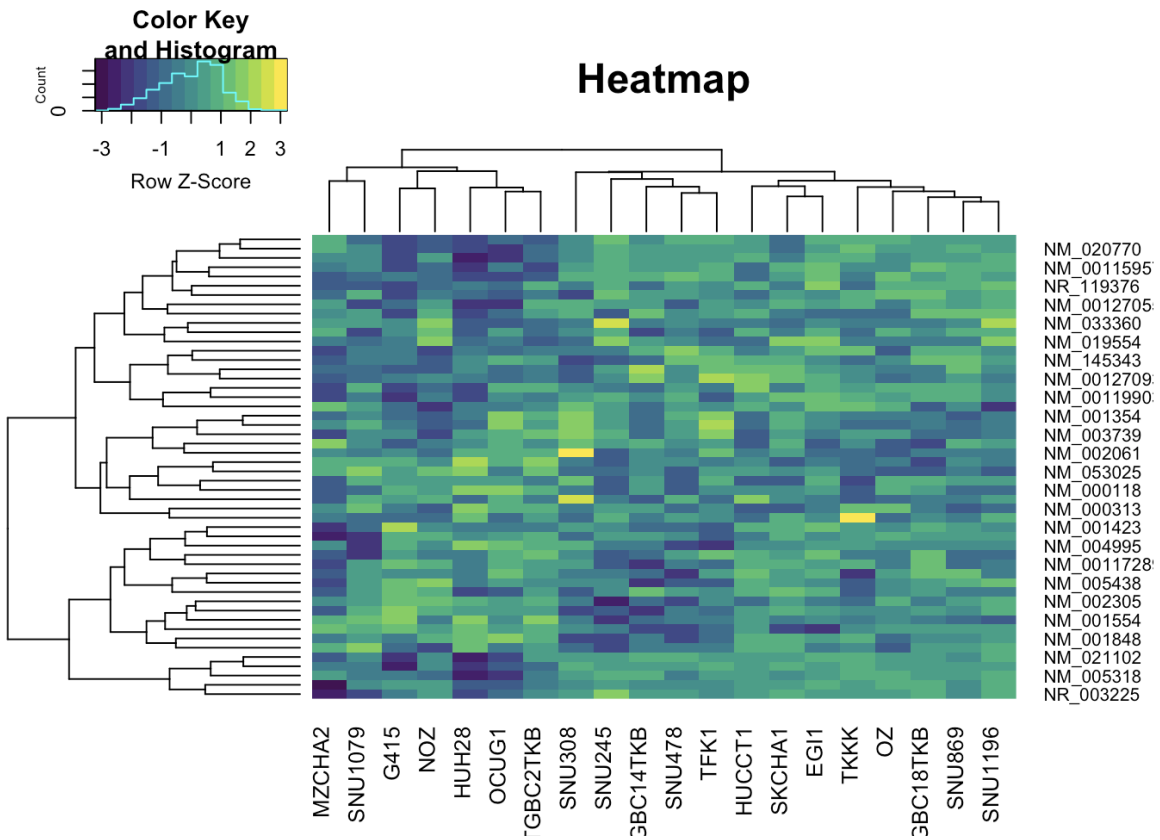
There are a few ways to determine how close two clusters are:

- Complete linkage clustering: Find the maximum possible distance between points belonging to two different clusters.
- Mean linkage clustering: Find all possible pairwise distances for points belonging to two different clusters and then calculate the average.

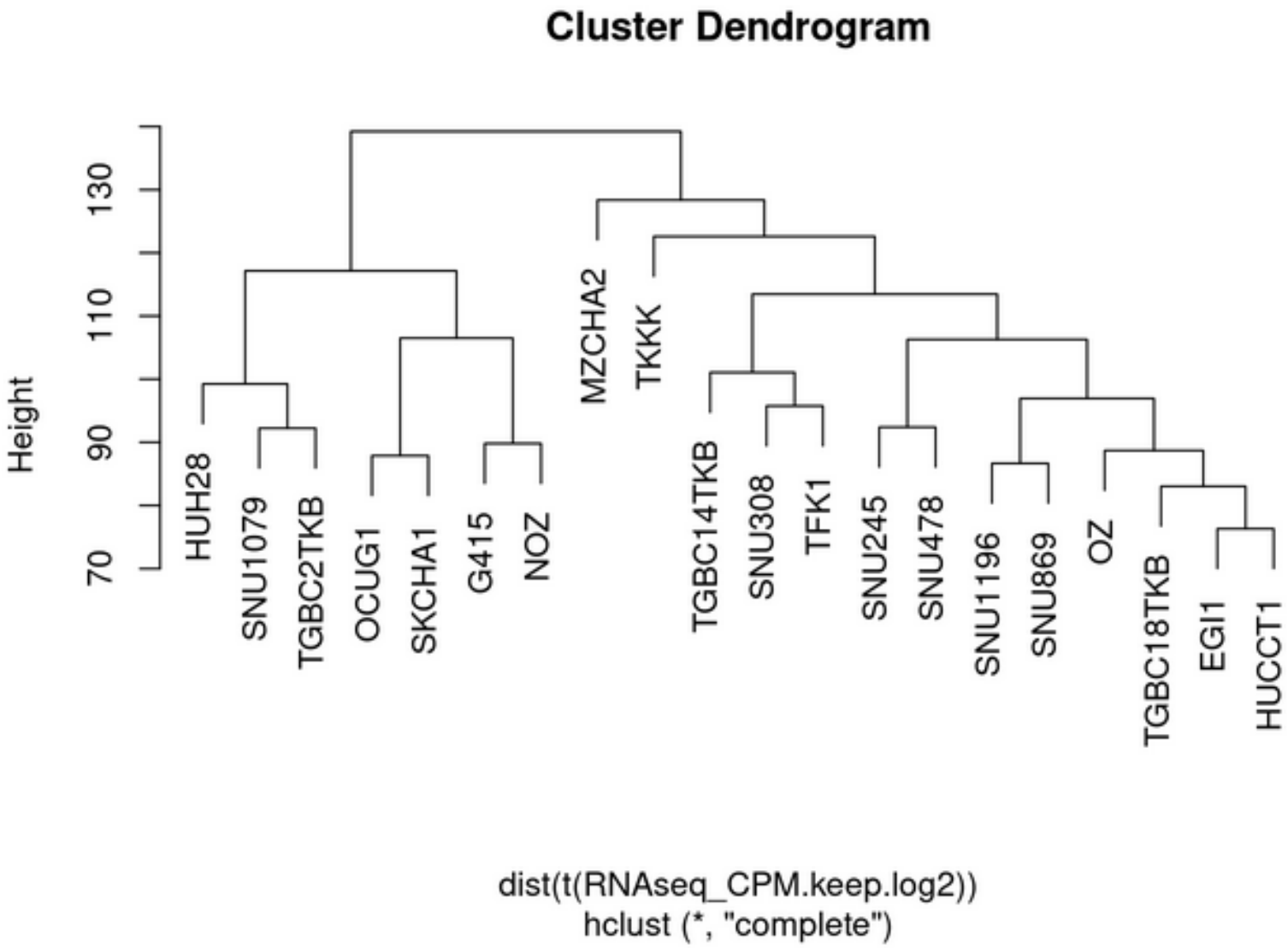
Complete linkage and **mean linkage** clustering are the ones used most often.

Data Visualization: Heatmap for highly variable genes

```
hvlcpm <- RNAseq_CPM.keep.log2[select_genes, ]
gplots::heatmap.2(hvlcpm,
          col=viridis,
          trace="none",
          main="Heatmap",
          scale="row")
```



Hierarchical Clustering with Complete Linkage




```
## Using complete linkage clustering
clusters_complete <- hclust(dist(t(RNAseq_CPM.keep.log2)), method = "complete")
plot(clusters_complete)
```

How to visualize sample relationship using all gene expression data?

- 10034 genes left after filtering out low expressed ones
- Still high dimensional data
- Project high dimensional data to two dimensions (dimension reduction), and then a scatter plot will work.
- How?

2. UMAP (Uniform Manifold Approximation and Projection)

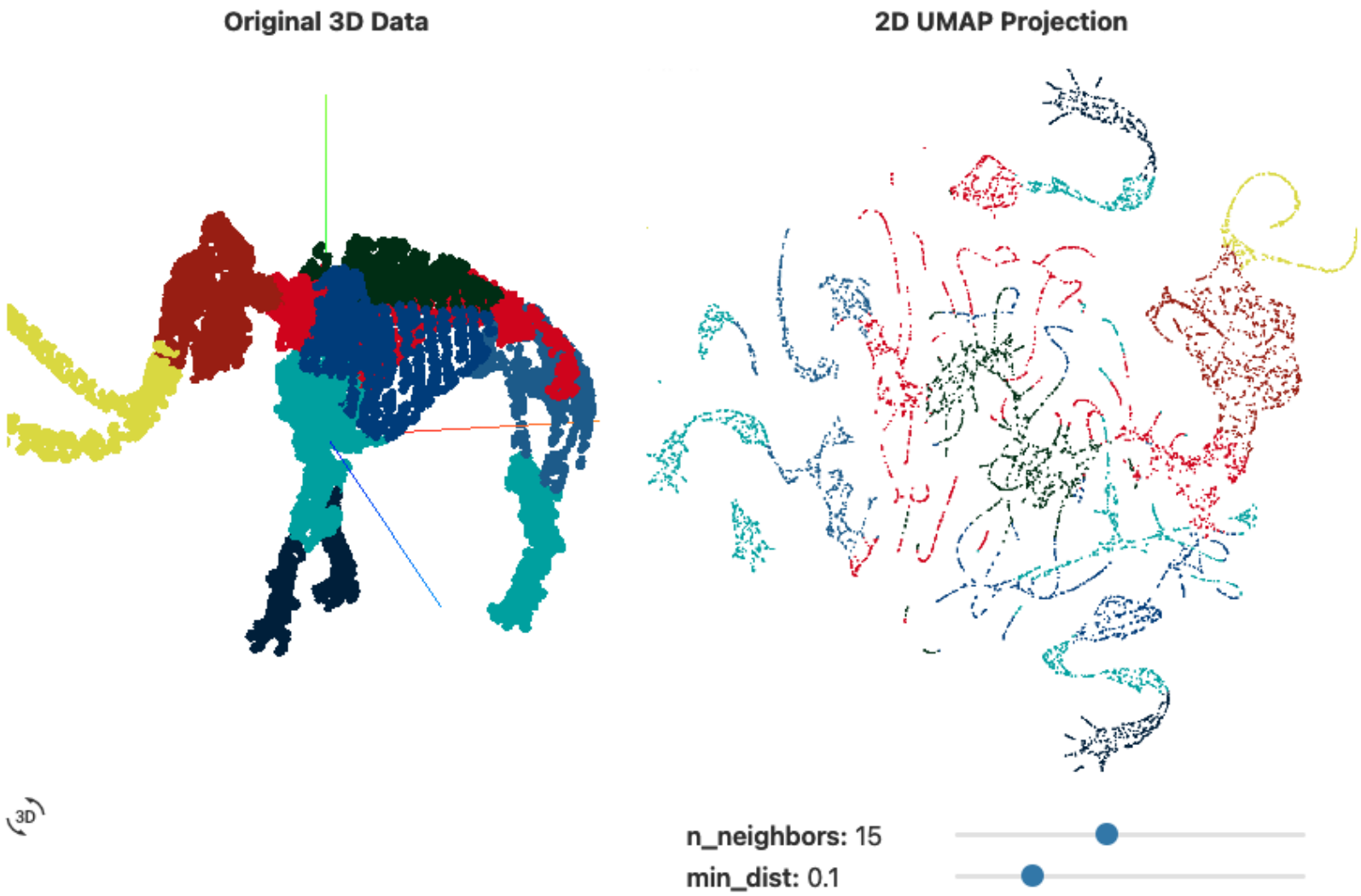
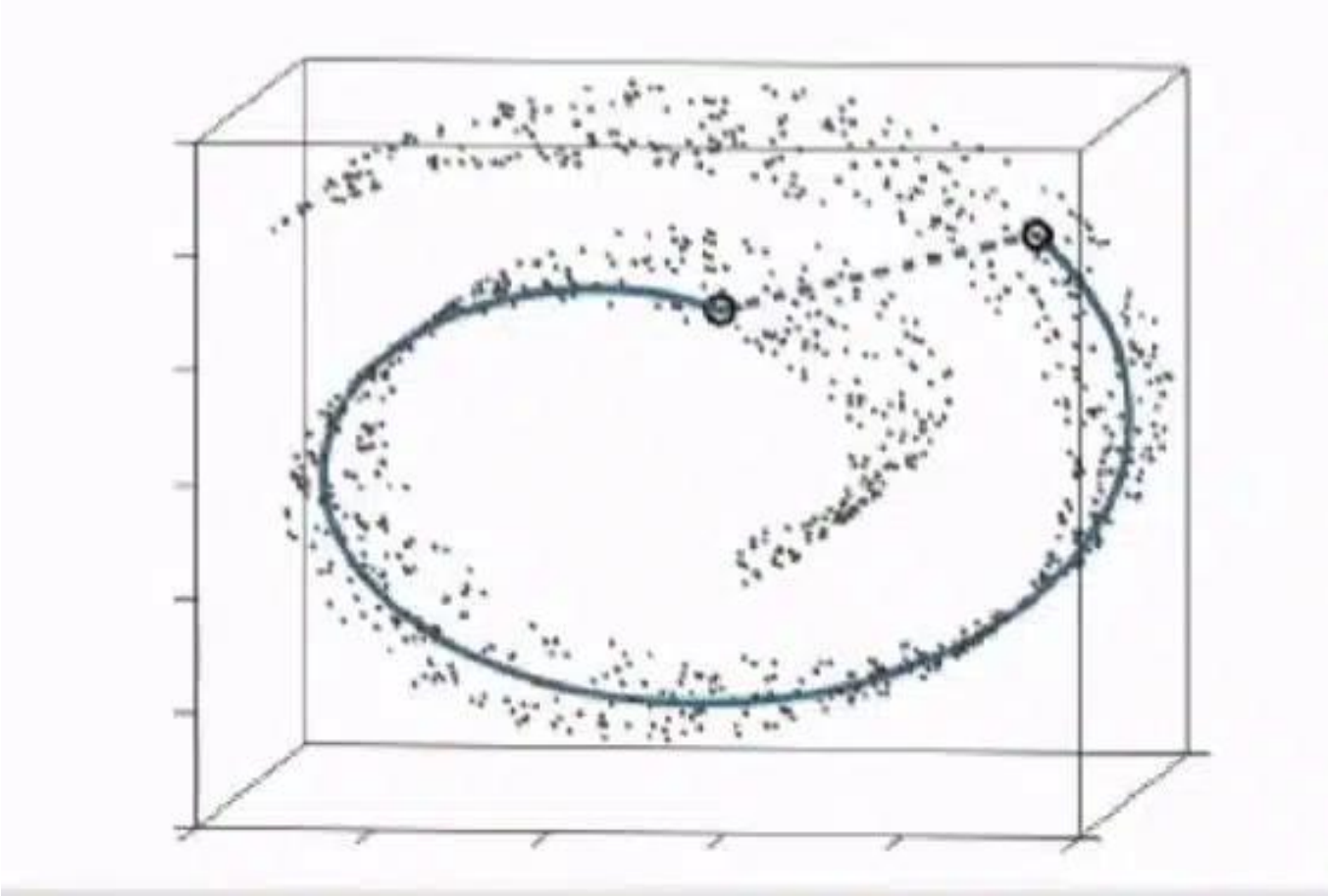


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` *and* `min_dist` *parameters.*

https://pair-code.github.io/understanding-umap/

How does UMAP work?

By learning the manifold, or shape, of the high-dimensional dataset, the UMAP algorithm calculates the correct distance (solid line) between points in complex patterns instead of its linear distance (dashed line).



What does the name UMAP mean?

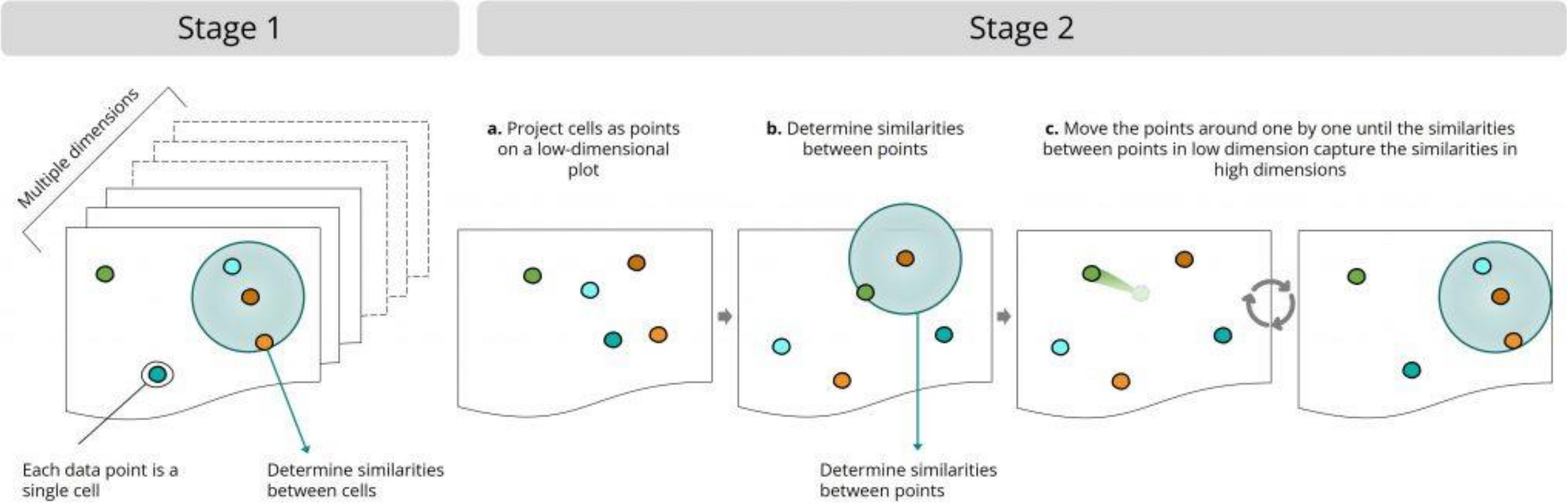
- *Uniform* refers to a mathematical assumption UMAP uses. It assumes that all data points distribute evenly across the manifold. In reality, this is almost never the case. So, the consequence of this assumption is that the points are uniformly distributed, but the space between the points is artificially warped. Hence, the distance across the manifold varies. UMAP uses this distance to calculate the similarities between cells.
- *Manifold* means the shape of the data points. UMAP learns that shape to determine the similarities between cells.
- *Approximation* refers to the fact that the algorithm should approximate the data's manifold, or shape, to determine similarities between cells.
- *Projection* happens after the cells' similarities are determined. UMAP projects the cells as points on a 2D or 3D plot. Then, it moves the points around until the 2D or 3D plot captures the similarities of the high-dimensional data.

How does UMAP work?

- In the simplest sense, UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.
- UMAP builds something called a "fuzzy simplicial complex". This is really just a representation of a weighted graph, with edge weights representing the likelihood that two points are connected.
  - To determine connectedness, UMAP extends a radius outwards from each point, connecting points when those radii overlap.
  - Choosing this radius is critical - too small a choice will lead to small, isolated clusters, while too large a choice will connect everything together.
  - UMAP overcomes this challenge by choosing a radius locally, based on the distance to each point's *n*th nearest neighbor.
  - UMAP then makes the graph "fuzzy" by decreasing the likelihood of connection as the radius grows.
  - Finally, by stipulating that each point must be connected to at least its closest neighbor, UMAP ensures that local structure is preserved in balance with global structure.
- Once the high-dimensional graph is constructed, UMAP optimizes the layout of a low-dimensional analogue to be as similar as possible.

https://pair-code.github.io/understanding-umap/supplement.html

Cluster Single Cell RNA-seq Data by UMAP



Cluster Single Cell RNA-seq Data by UMAP

Clustering Bulk-RNAseq Samples by UMAP

```
# Generate UMAP data object
RNAseq.umap = umap(t(RNAseq_CPM.keep.log2))
RNAseq.umap
```

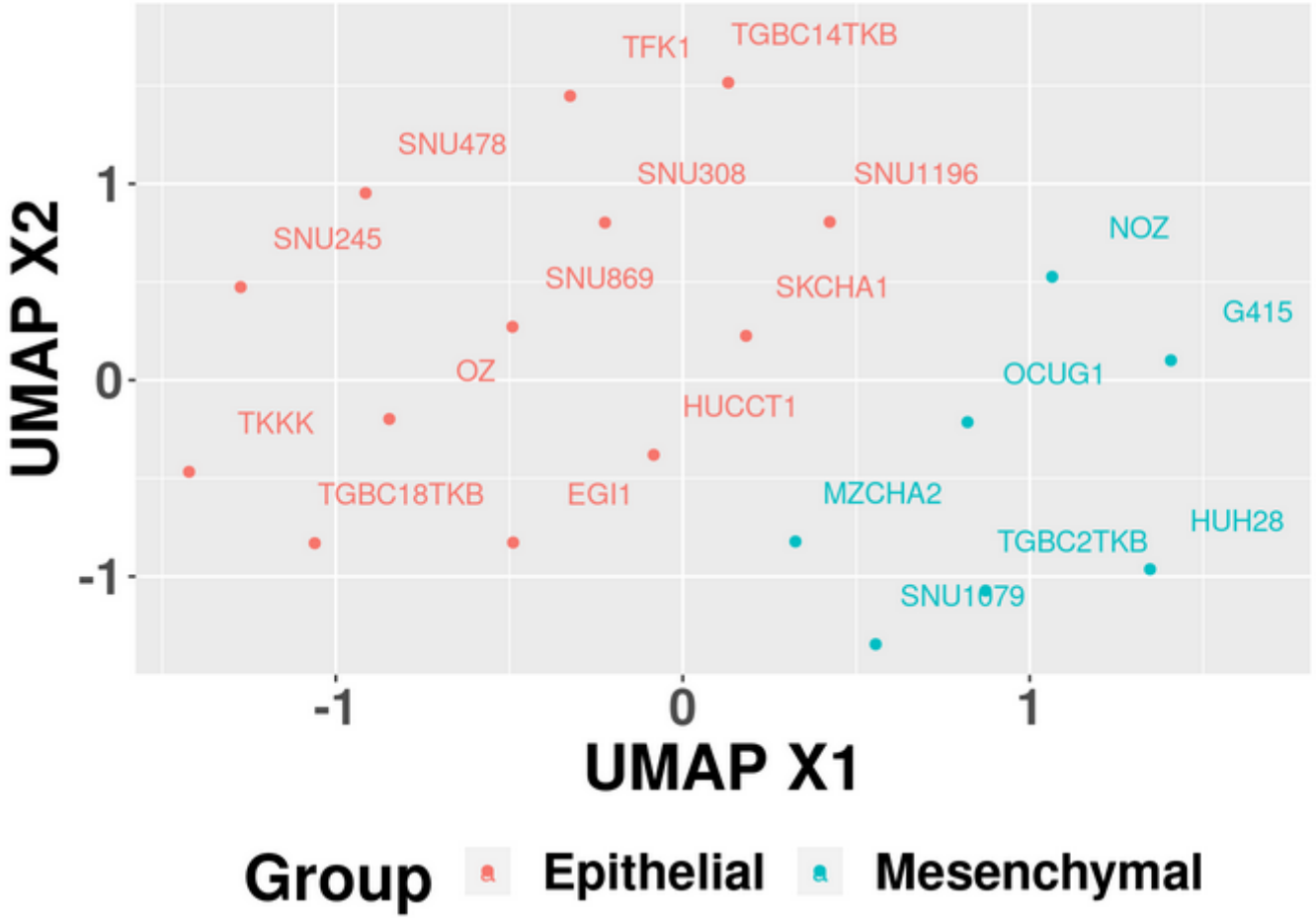
```
## umap embedding of 20 items in 2 dimensions
## object components: layout, data, knn, config
```

```
head(RNAseq.umap$layout)
```

EGI1	-0.4888411	-0.8272848
G415	1.4069668	0.1008201
HUCCT1	-0.0836996	-0.3807546
HUH28	1.3468850	-0.9633277
MZCHA2	0.3247851	-0.8221387
NOZ	1.0647610	0.5263747

Clustering RNAseq Samples by UMAP

```
# Plot UMAP X1 vs. X2
ggplot(data.frame(RNAseq.umap$layout),
       aes(x = X1, y = X2, colour = cell_group$group_label)) +
    geom_point() + labs(x = "UMAP X1", y = "UMAP X2") +
    geom_text(label = rownames(RNAseq.umap$layout), nudge_x = 0.25, nudge_y = 0.25,
    check_overlap = T) + labs(colour = "Group")
```



References

- Towards Data Science Blogs: https://medium.com/@NotAyushXD
- Kaggle: https://www.kaggle.com/
- Introduction to R library “caret”
  - https://topepo.github.io/caret/index.html
- Extra Resources: https://hbctraining.github.io/main/
- Clustering Single Cells with scRNAseq Data by UMAP:
  - https://hbctraining.github.io/scRNA-seq_online/schedule/links-to-lessons.html

In-Class Activity