



MATLAB

Class 6: statistics basics





**Statistics aims to understand your data by
describing it and making predictions**



Descriptive stats aims to describe data's distribution by **central tendency** (location in a plane) and **dispersion** (spread) around the location



Inferential stats aims to predict future outcomes or observations based on your data





Useful descriptive stats and associated functions

Data Structures

Mean

Average
value of
array

Median

Returns the
middle
value of
array

Mode

Returns
most
common
value of
array

Data Structures

Median

Note: that for MATLAB V 2018b and newer you can use the input 'all' to take the mean, median, etc across ALL elements. Previously you need to script this

array

common
value of
array

Data Structures

Max

Find largest
elements of
an array

Maxk

Find the
largest k
elements of
an array

Min(k)

Find the
smallest (k)
elements of
an array

Data spread

STD

Computes
standard
deviation of
array

range

Difference
between
max and
min

var

Computes
variance of
array

Data spread

STD

standard
deviation of
array

range

min

var

variance of
array

prctile(x,p) returns the p percentile of the data vector X



Dealing with missing data

Missing data in MATLAB generally takes the form of
NaN values

You need to address these before running analysis

Ismissing()







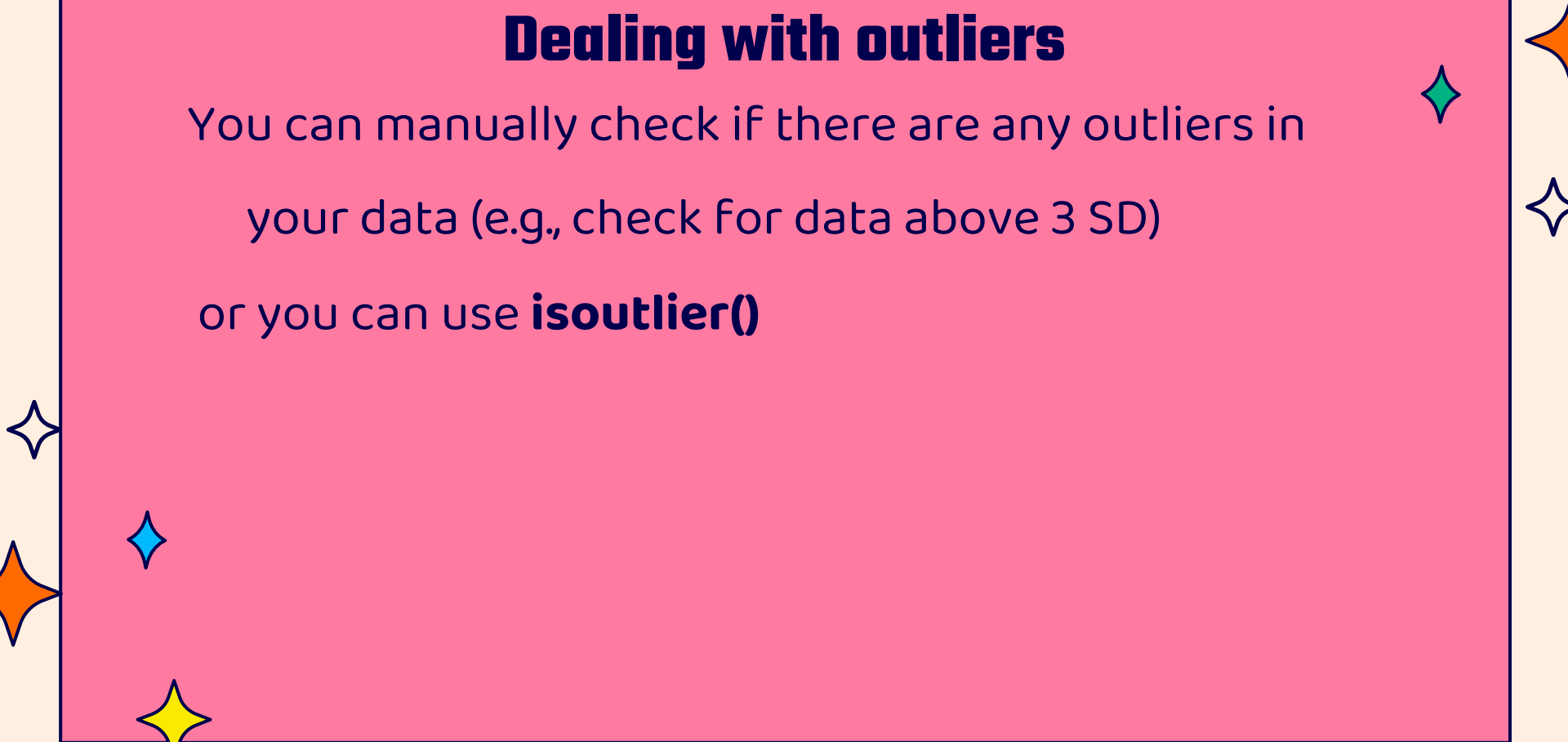
Rmmissing





Dealing with outliers

You can manually check if there are any outliers in
your data (e.g., check for data above 3 SD)
or you can use **isoutlier()**





Correlations

Correlations is the relationship between any two random variables

They can be described in different ways:

Pearson—linear relationship between continuous variables

Spearman Rho—nonparametric rank correlation, describing two variables as a monotonic function



Correlations in MATLAB

corr — returns a matrix of pairwise correlations between columns

corrcoef — Returns the correlation between vectorized matrices

corr2 — returns correlation coefficient for matrices (i.e., one value for its 2-d inputs)





Reminder: Reshape can help you

Reshape is a useful function to transform any sized matrix into a different shape

Reshape(X, [new dimensions])

Note that the new dimensions need to be consistent with the previous ones

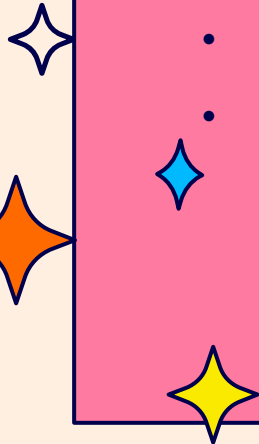
i.e., $\text{dim1} * \text{dim2} * \text{dim3} == \text{newdim1} * \text{newdim2}$ etc..



T-tests

Student t-test allows you to test mean differences between normal distributions

There are many different **flavours** of t's

- one-sample vs two samples
 - paired vs unpaired
 - one tail vs two
- 



T-test

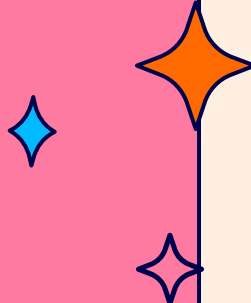
T-tests assume that your data come from a **normal** distribution and the observations are sampled **independently** from one another

These assumptions apply for both paired and unpaired test

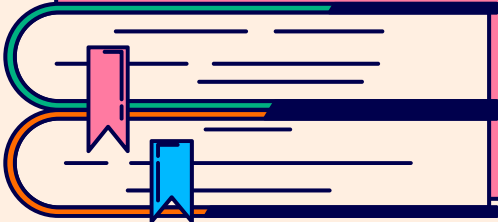


T-tests

One-sample t-tests test the hypothesis that mean is different than a prespecified μ_0






Independent-sample t-tests test the hypothesis that the mean difference between both samples is not 0





T-tests

Paired t-tests are used when the observations are repeated, i.e., each person is sampled twice thus each observation comes in a pair



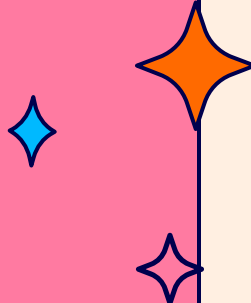
Unpaired t-tests are used when the observations are independent





T-tests

One tail tests are used when you have **directional** hypotheses (i.e., group A is bigger than group B)



Two tail tests are used when you have **non-directional** hypotheses (i.e., group A is different than group B, but you don't care if it's bigger or smaller)





T-tests

In MATLAB there are two functions for the student
t-test

ttest() is used for one-sample and paired tests

while **ttest2()** is used for independent sample (i.e.,
two-sample) tests





Ttest0

One sample and paired t-tests

Ttest(x) runs a one-sampled t-test against zero

Ttest(x, y) paired t-test

'Alpha'

'Tail' can specify 'left', 'right', 'both'



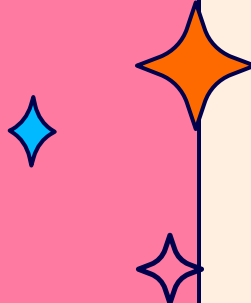


Ttest20

Ttest2() runs a two independent samples t-test.

Has the same specifiers as ttest() including:

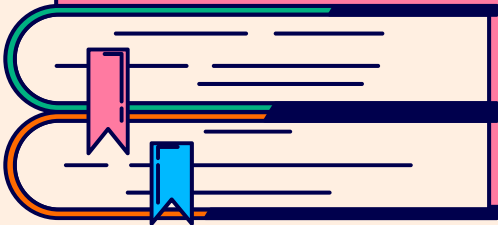

'Dim' to specify a dimension along which to run the test
and 'Vartype' for 'equal' and 'unequal' variances








T-tests

Given the same number of data which type of t-test is more stringent? Which one requires a bigger mean diff for the same value of t ?





T-tests

Let us assume we collect data from 20 people (paired) 
and observe a mean diff of 3.75, and a sd of 1
Then the t value for a paired test would be: 


$$t = \frac{\text{mean diff}}{\frac{sd}{\sqrt{n}}} \text{ with df } n - 1$$

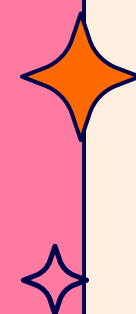




T-tests

Let us assume we collect data from 20 people (paired) and observe a mean diff of 3.75, and a sd of 1

Then the t value for a paired test would be:



$$t = \frac{3.75}{\frac{1}{\sqrt{20}}} = \text{with df 19}$$



T-tests

Let us assume we collect data from 20 people
(independent groups) and observe a mean diff of 3.75,
and a sd of 1 (assuming equal variance)

Then the t value for an unpaired test would be:

$$t = \frac{\text{mean diff}}{sp * \sqrt{\frac{1}{n1} + \frac{1}{n2}}} \text{ with } sp = \sqrt{\frac{(n1 - 1) * std1^2 + (n2 - 1) * std2^2}{n1 + n2 - 2}} \text{ df } n1 + n2$$

T-tests

Let us assume we collect data from 20 people
(independent groups) and observe a mean diff of 3.75,
and a sd of 1 (assuming equal variance)

Then the t value for an unpaired test would be:




$$sp = \sqrt{\frac{(20 - 1) * 1 + (20 - 1) * 1}{20 + 20 - 2}}$$

$$t = \frac{3.75}{sp * \sqrt{\frac{1}{20} + \frac{1}{20}}} \quad df \ 38$$



T-tests

T value for paired is larger, in comparison to t values of unpaired or independent tests



This is because **within-person** designs have more **power** as they control for more noise by observing the same person twice



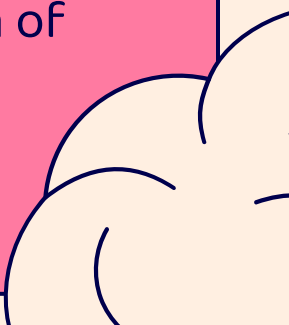



Non-parametric tests

In stats there are some tests that are **non-parametric**, by this statisticians mean that the test does not require assuming a specific model or distribution for your data

These are sometimes referred to as **non-distributional** tests

These tests are used when you do not want to assume a specific distribution (e.g., violation), or do not know the distribution of your data





Non-parametric t-tests

Regular t-tests assume your data is normally distributed

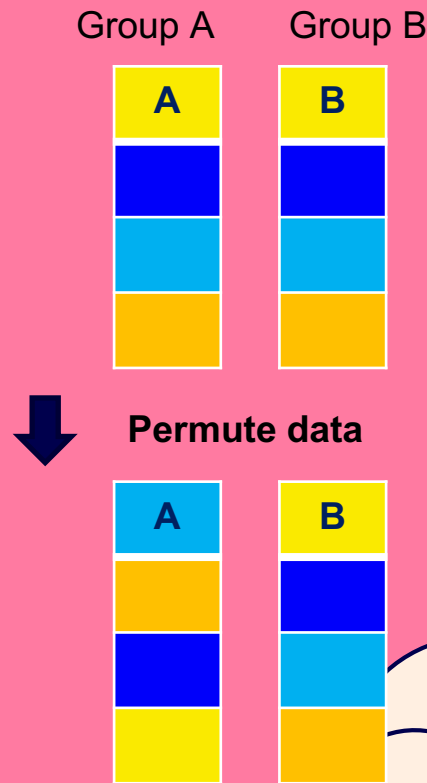
There is a *non-parametric equivalent of a t-test* called the

permutation t-test. This test works on the premise that you can observe a *null distribution* from your own data by randomly permuting groups.

Reminder: a null distribution is the distribution when the *null hypothesis* is true

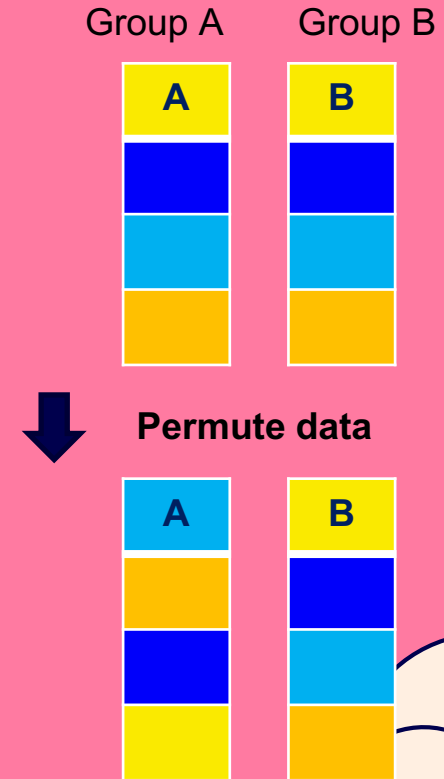
permutations

Permutations build a *null distribution* of data based on your observations under the assumption that randomly shuffling your data will void the effect of interest. Thus, you can measure how surprising the effect you observe is given your data based on the computed null distribution

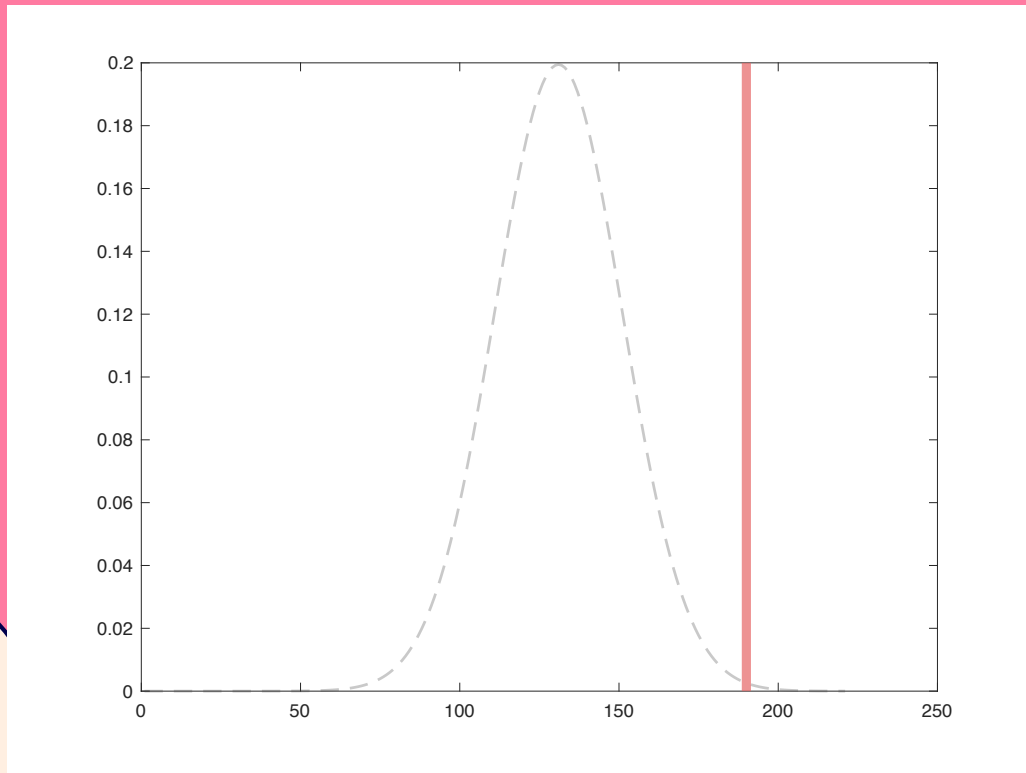


permutations

This technique is very similar to bootstrapping without replacement (i.e., no duplicate observations).

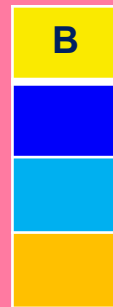


permutations

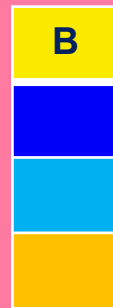


Group A

Group B



Permute data





Bootstrapping

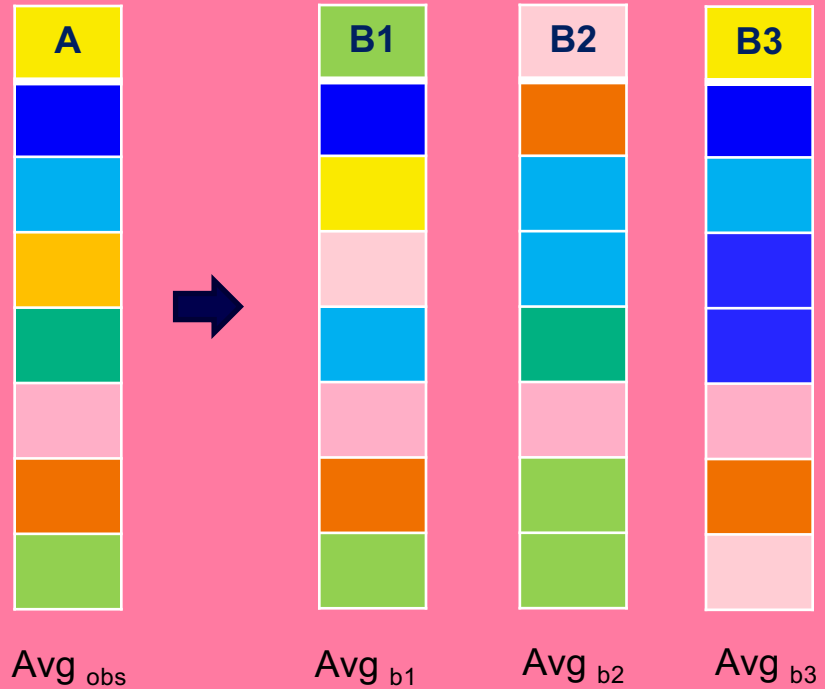
A very similar concept in statistics is the idea of **bootstrapping** to get a measure of uncertainty around an estimate (e.g., CI)

Bootstrapping is like permutations in that they resample your data, however bootstrapping requires **replacement**

It is often used to calculate the error associated to an estimate, effect, or performance of an algorithm and allows you to know if one given data point is driving the effect you see

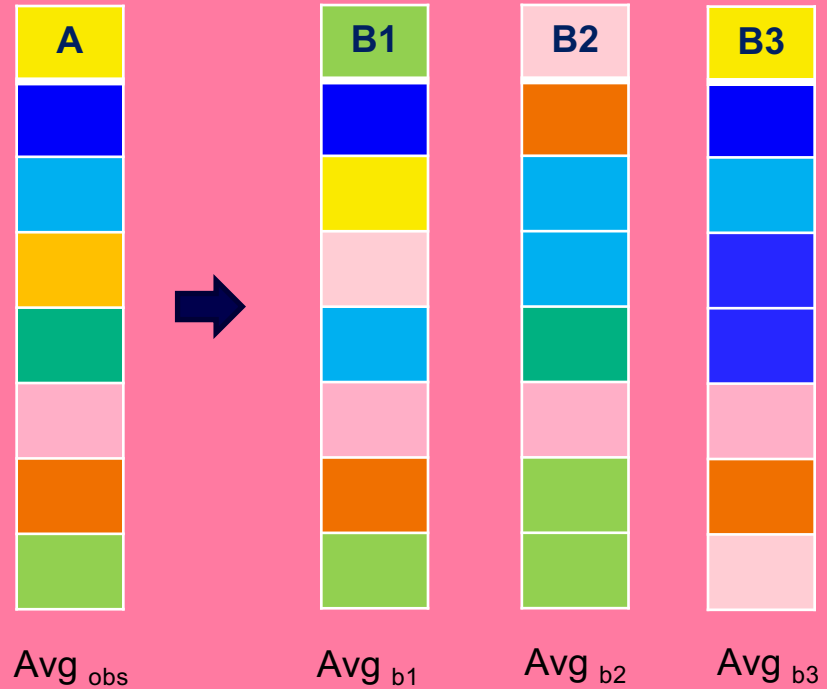
Bootstrapping

Bootstrapping is like
permutations in that
they resample your
data, however
bootstrapping
requires
replacement



Bootstrapping

The computed Avg for each **bootstrap** will allow you to get an idea of what **range** of values you could expect given your data





Signal Detection Theory

An analytic tool used to analyze data in its ability to discriminate between two signals or signal and noise

Assumes there is an inherent uncertainty in the classification

We will look at the case where there are two classes to categorize



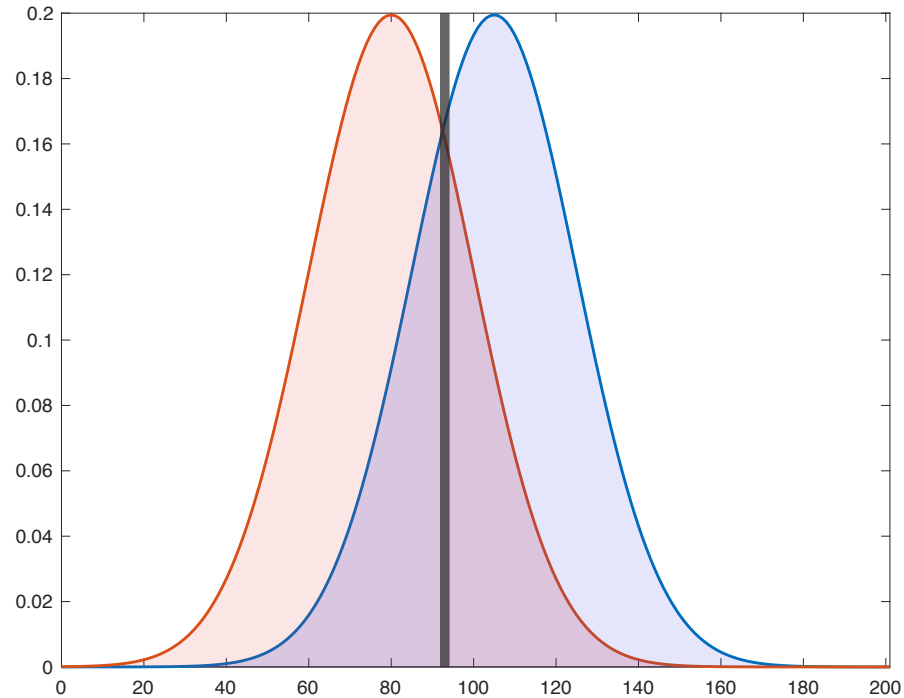
Signal Detection Theory

There are two parameters that describe signal detection theory

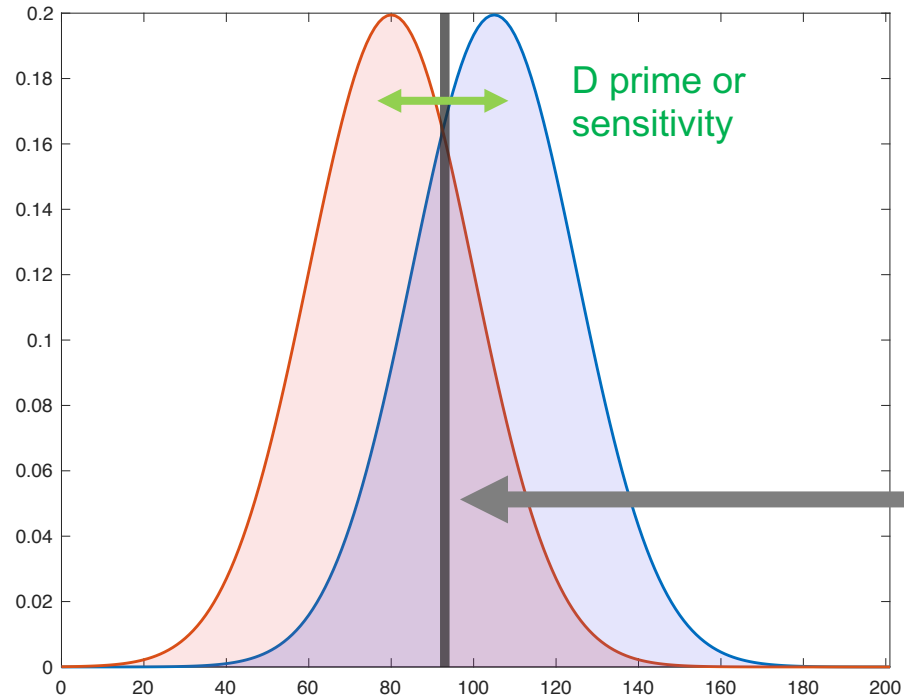
The criterion: where you draw the boundary between signal and noise

Sensitivity: one's ability to discriminate between signal and noise

Signal Detection Theory

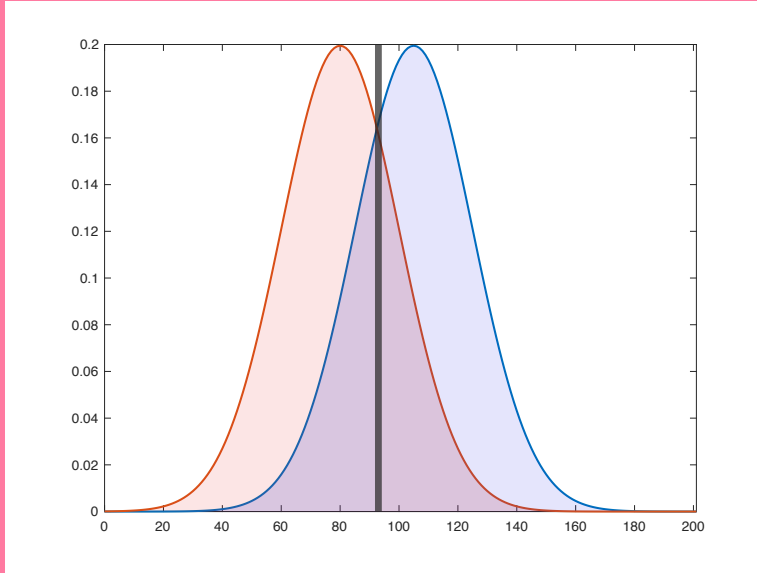


Signal Detection Theory



criterion

Signal Detection Theory



	Signal	Noise
Present	Hits	False Alarms
Absent	Misses	Correct Rejection

Signal Detection Theory

Sensitivity:

$$d_{prime} = (z(Hits) - z(False Alarms))$$

Criterion:

$$c = -\frac{1}{2} (z(Hits) + z(False Alarms))$$

Where $z()$ is the inverse of the cumulative normal distribution



Signal Detection Theory

What to do when you get values of 0 or 1 as probabilities?

You cannot take the inverse cumulative normal distribution of 0 or 1 as it returns infinite values.

We therefore must apply a correction to our data

Signal Detection Theory

We assume that if we double the number of trials, by chance someone would have guessed the right answer i.e., if p_{HIT} or $p_{FA} = 0$, then we correct to

$$\frac{1}{2 * Num Trials}$$

Signal Detection Theory

Similarlry we assume that if we double the number of trials, someone would have made one mistake i.e., if p_{HIT} or $p_{FA} = 1$, then correct with

$$\frac{(2 * \text{Nume Trials}) - 1}{2 * \text{Num Trials}}$$

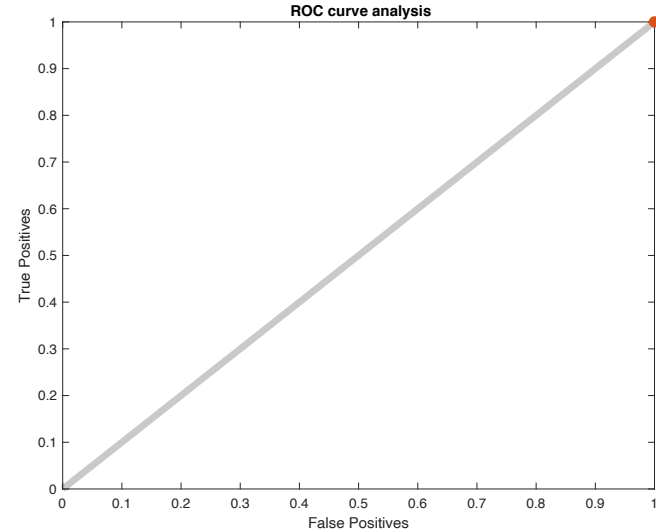
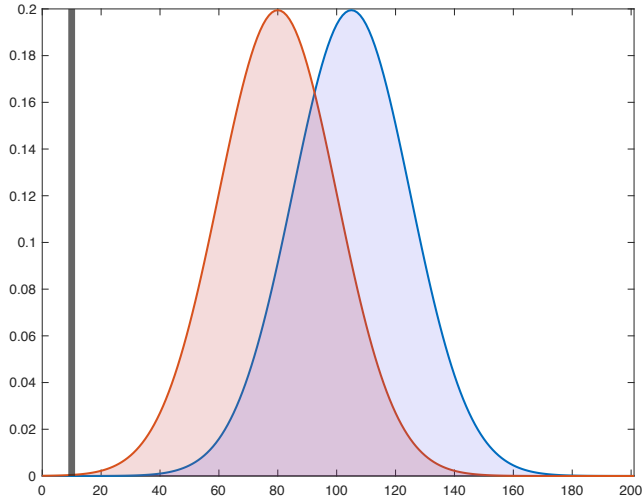


Receiver Operating Characteristic Curves

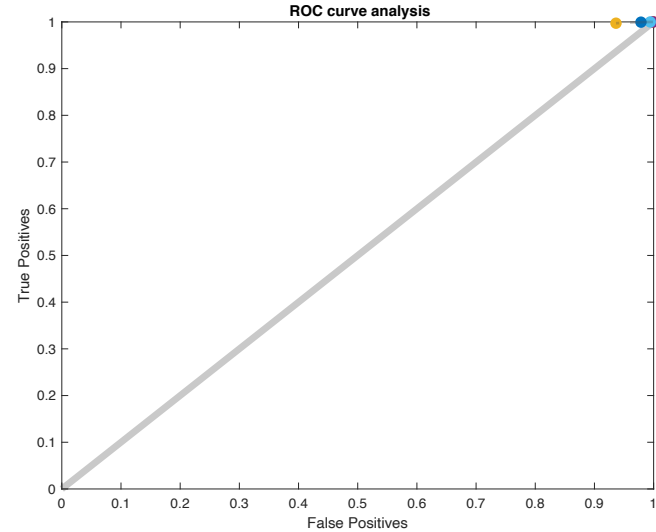
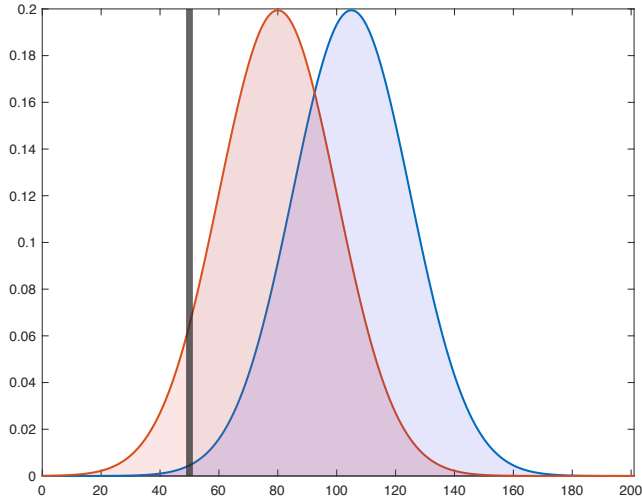
Is a diagnostic plot that helps visualize the ability of a binary classifier to separate two classes

This is achieved by plotting the rate of **True Positives** against **false positives**

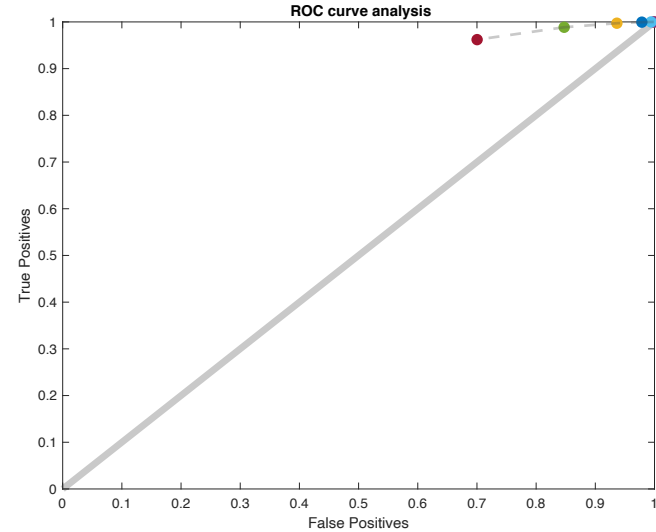
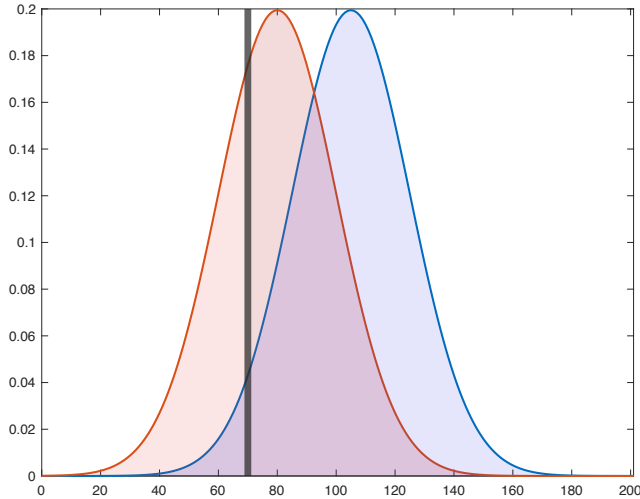
Receiver Operating Characteristic Curves



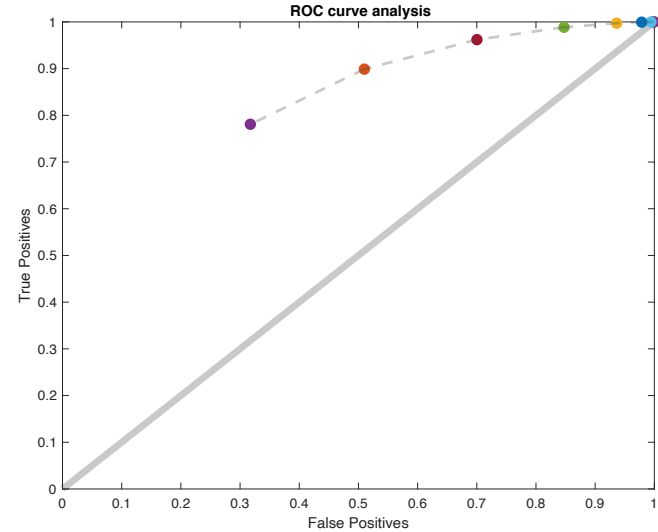
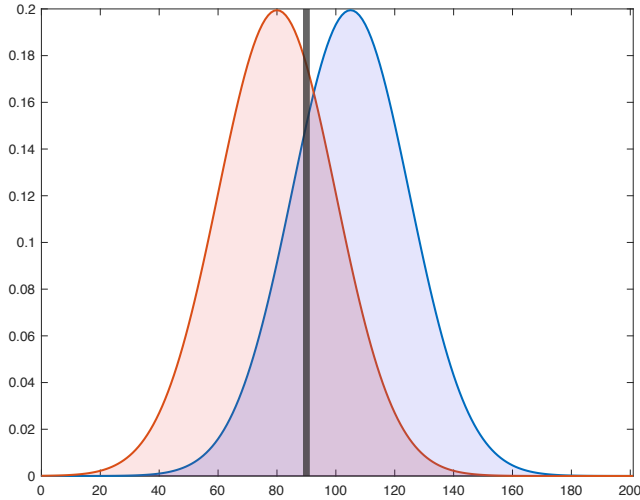
Receiver Operating Characteristic Curves



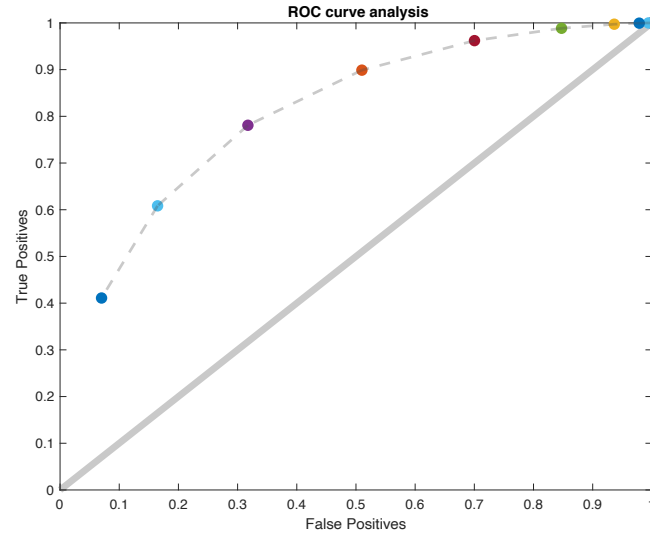
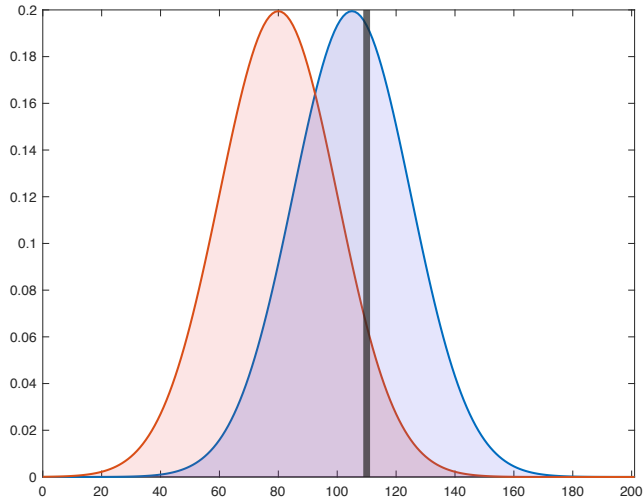
Receiver Operating Characteristic Curves



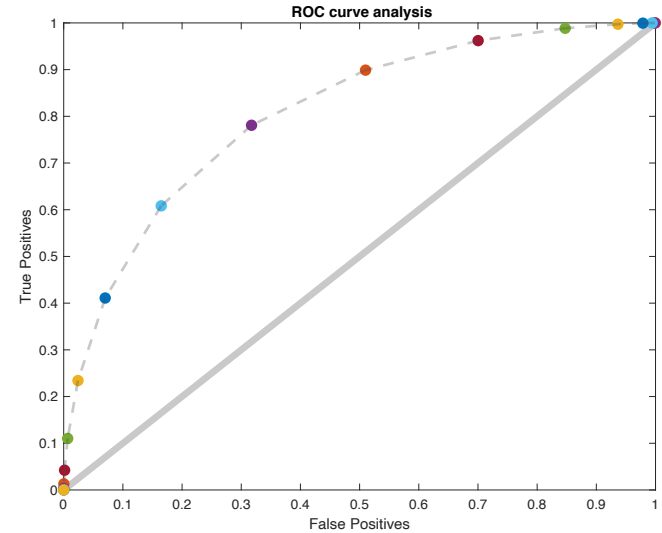
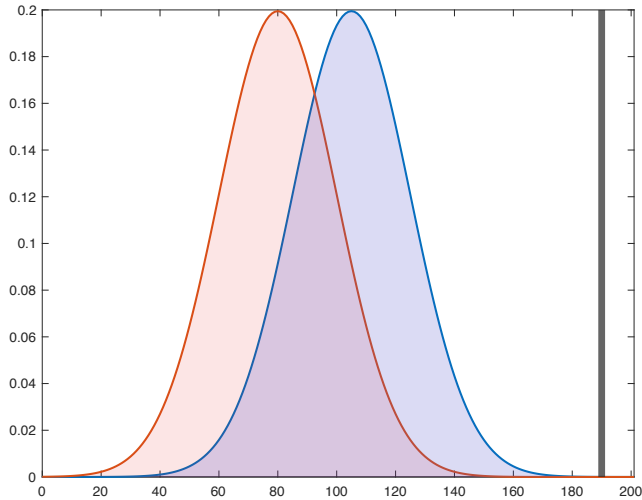
Receiver Operating Characteristic Curves



Receiver Operating Characteristic Curves



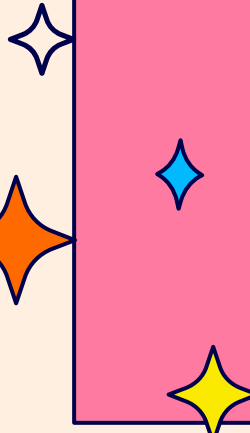
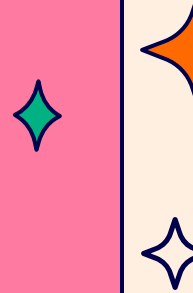
Receiver Operating Characteristic Curves



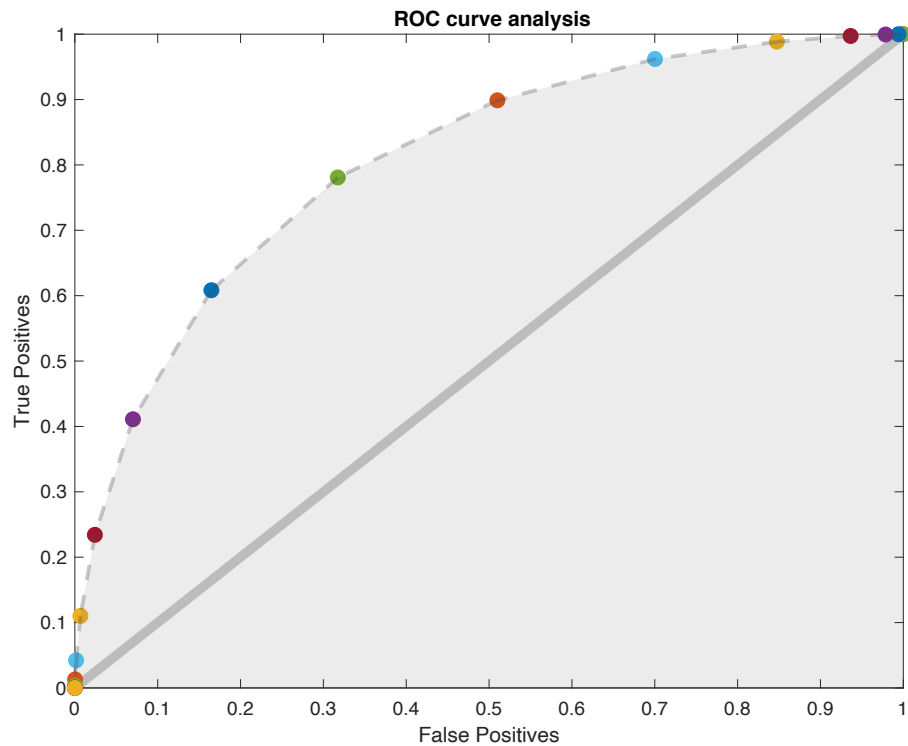


Area Under the Curve

Measures the area under the ROC curve and reflects
one's ability to classify classes given a variable
AUC of 1 or 0 is perfect classification while 0.5 is
chance level (i.e., along the diagonal)



Area Under the Curve



ROC in MATLAB

[X,Y,T,AUC] = perfcurve(labels,scores,posclass)

labels—the two classes to be discriminated

scores—the 'x' values used in discrimination

posclass—which class is larger of the two

ROC in MATLAB

`[X,Y,T,AUC] = perfcurve(labels,scores,posclass)`

X— x values of ROC curve

Y— y values of ROC curve

T—array of thresholds used

AUC—returns the value of AUC