Bioinformatics Lab

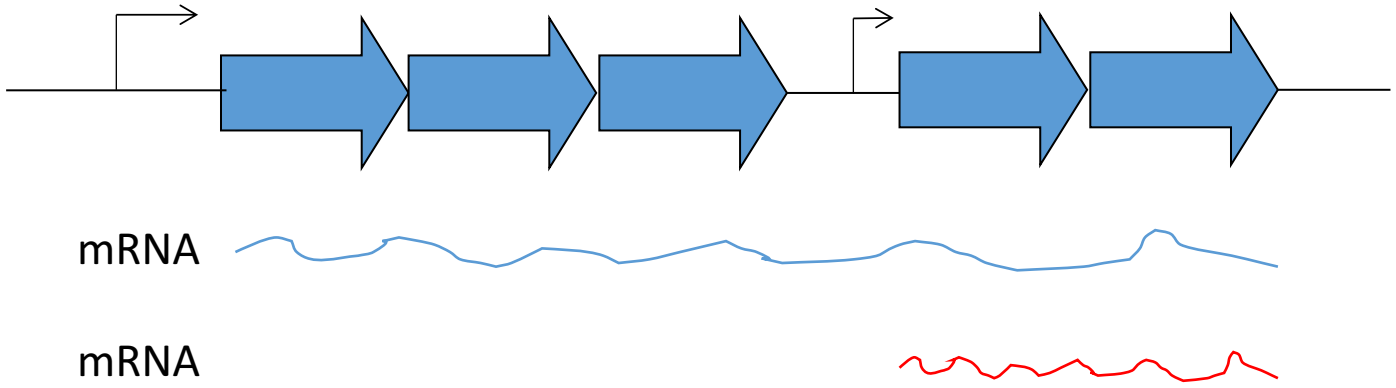
Week 7 Session 2

Instructor:
Arturo Medrano
l1medranosoto@ucsd.edu

Instructional Assistant:
Hanbin Lu
hal213@ucsd.edu

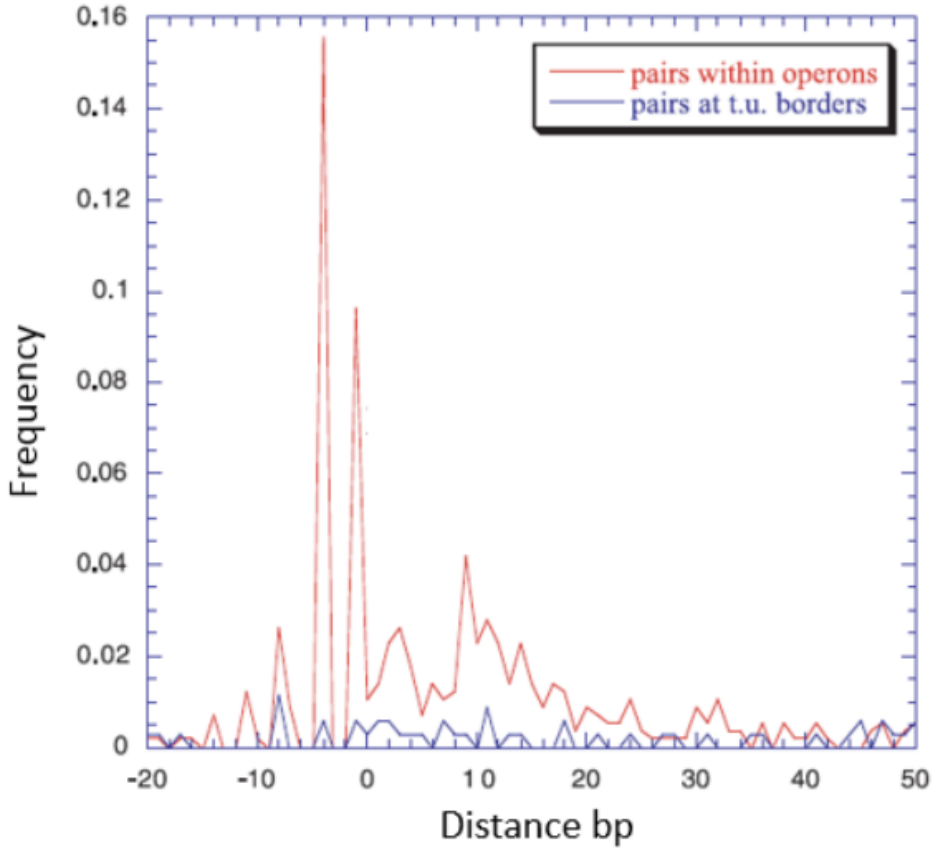
Inference of operons

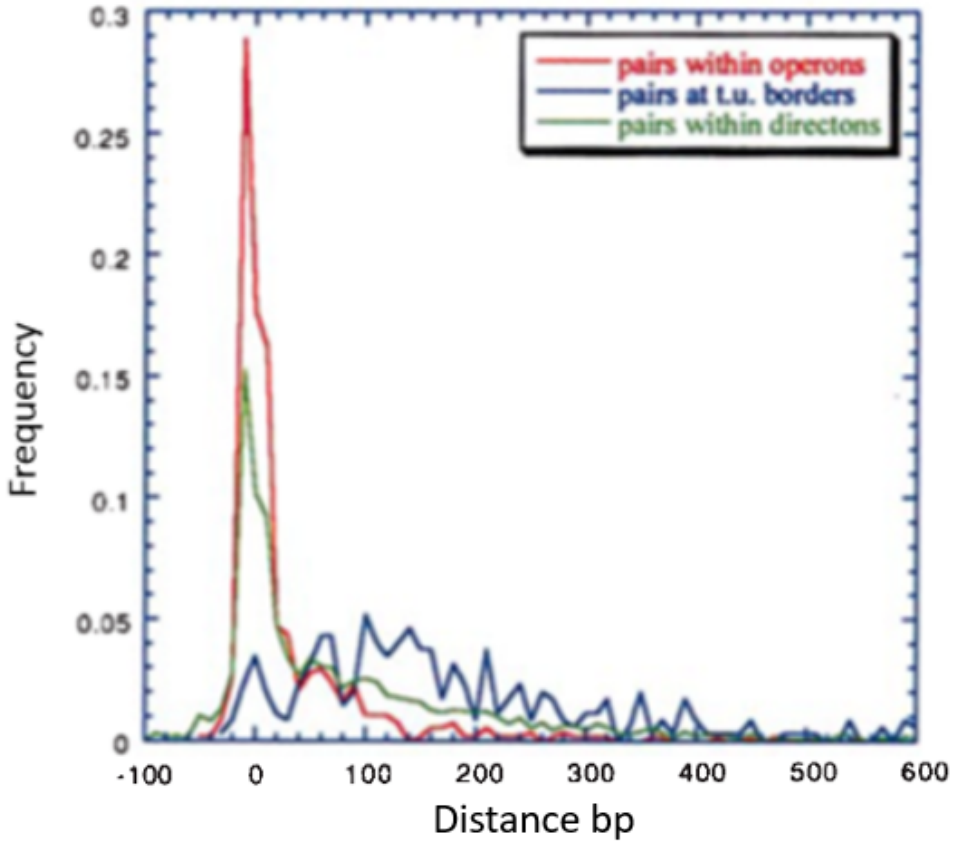
We want to design a method to estimate the probability that two adjacent genes (in the same strand) are in the same operon.



Feasibility Analysis on the selected relevant variable (intergenic distance)

Is there a significant difference in the distances between adjacent genes in the same operon versus adjacent genes (still on the same strand) that do not belong to same operon (operon borders)?

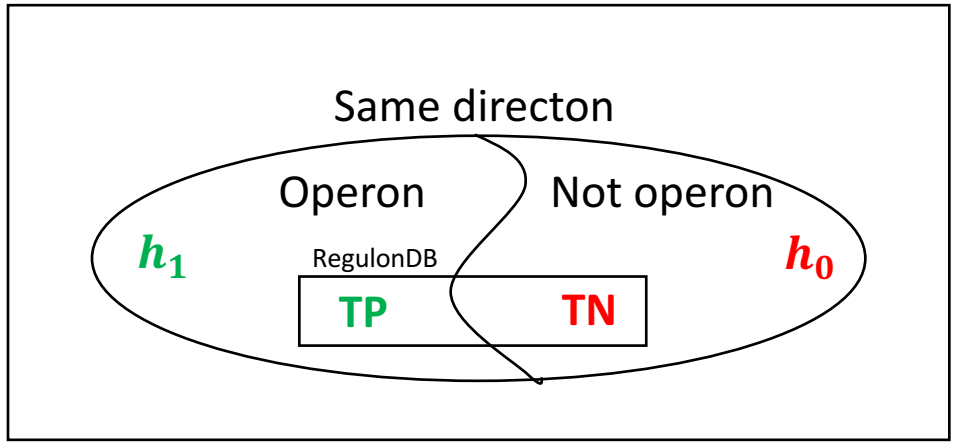




2000, PNAS 97(12): 6652-57

Our Bayesian model

Distances between adjacent genes i, j



The competing hypotheses

$h_1 = OP_{i,j}$ Adjacent genes i,j are in the same operon

$h_0 = NOP_{i,j}$ Adjacent genes i,j are NOT in the same operon.

What we want to know:

$p(h_1|d_{i,j} = x)$ Posterior probability that genes i and j are in the same operons given that they have intergenic distance x.

Applying the Bayes theorem:

$$p(h_1|d_{i,j} = x) = \frac{p(d_{i,j} = x|h_1)p(h_1)}{\sum_{k=0}^{1} p(d_{i,j} = x|h_k)p(h_k)}$$

$p(d_{i,j} = x|h_1)$: Likelihood of observing a distance x between genes i,j under the hypothesis that they are in the same operon. That is, our model of how the distances distribute given h_1.

$p(h_1)$: Prior probability of h_1

Now back to reality: we don't know the operon partition! Therefore we must work with samples and assume they are representative of the genomic distribution of distances in operons. We'll take this sample from RegulonDB.

What about the prior $p(h_1)$?

The prior presents an opportunity to include what we currently know about $p(h_1)$ before looking at the data. The prior for the null hypothesis would be:

$$p(h_0) = 1 - p(h_1)$$

Getting the distances for the positive and negative controls

- Download the set of known operons for E. coli K12 from RegulonDB (http://regulondb.ccg.unam.mx/) the following the list of operons and TUs from the experimental datasets link:
  - http://regulondb.ccg.unam.mx/menu/download/datasets/files/TUSet.txt
  - http://regulondb.ccg.unam.mx/menu/download/datasets/files/OperonSet.txt
  - http://regulondb.ccg.unam.mx/menu/download/datasets/files/GeneProductSet.txt
- File **TUSet.txt** contains information on internal Transcription Units (TUs) within the same operon. This can be helpful to interpret our results but it will not be used directly in the calculations.
- File **OperonSet.txt** contains the genes that compose each known operon in the genome. We will work only with operons with evidence “Strong” or “Confirmed”.
- File **GeneProductSet.txt** contains columns that map the gene names in file **OperonSet.txt** to bnumbers.
  - We need this because it is not guaranteed that the gene name in the **OperonSet.txt** file is the same name in our SQL ‘genes’ table (it could be a synonym). Therefore we can use the bnumbers to query our SQL table ‘genes’ using the column locus_tag and get the coordinates of those genes from our SQL ‘exons’ table.

Map gene name to be numbers

- To prevent the case where a gene name in file ***OperonSet.txt*** is not included in our SQL tables ‘genes’ or ‘gene_synonyms’, we use the file ***GeneProductSet.txt*** to create a dictionary that maps gene name to the locus_tag (b-number).

GeneProductSet.txt

```
ECK120000130 carA b0032 29651 30799 forward carbamoyl phosphate synthetase, &alpha; chain
ECK120000131 carB b0033 30817 34038 forward carbamoyl phosphate synthetase, &beta; chain
ECK120002708 caiF b0034 34300 34695 forward CaiF transcriptional activator 10564497,8631699,9573142,
ECK120002330 caiE b0035 34781 35371 reverse predicted acyl transferase
ECK120001510 caiD b0036 35377 36162 reverse crotonobetainyl-CoA hydratase
ECK120001511 caiC b0037 36271 37824 reverse carnitine-CoA ligase
ECK120001512 caiB b0038 37898 39115 reverse &gamma;-butyrobetainyl-CoA:carnitine CoA transferase
ECK120001513 caiA b0039 39244 40386 reverse crotonobetainyl-CoA reductase 11551212,23718679,
ECK120001514 caiT b0040 40417 41931 reverse L-carnitine : &gamma;-butyrobetaine antiporter
ECK120001515 fixA b0041 42403 43173 forward predicted elecron transfer flavoprotein subunit, ETFP adenine nucleotide-binding domain
```

- With this dictionary in memory we can now proceed to parse file **OperonSet.txt**

Extracting genes in curated operons

- From file ***OperonSet.txt***, extract all the rows with evidence "**Strong**" or "**Confirmed**".
  - For example, you can cut the columns with the operon name (col 1), the genes in the operon (col. 6) and the evidence (col. 7). Then just grep for the words **Strong** or **Confirmed**.

OperonSet.txt

```
fear      1446378  1447283 reverse 1 feaR [BTEI|W|Boundaries of transcription experimentally identified]Weak
fecABCDE  4510690  4516677 reverse 5 fecA,fecB,fecC,fecD,fecE [LTED|S|Length of transcript experimentally determined] Confirmed
fecIR     4516764  4518235 reverse 2 fecI,fecR [LTED|S|Length of transcript experimentally determined] Strong
aceBAK    4215478  4220332 forward 3 aceB,aceA,aceK [BTEI|W|Boundaries of transcription experimentally identified] Strong
feoABC    3540163  3542964 forward 3 feoA,feoB,feoC [PM|S|Polar mutation]; [IHBCE|W|Inferred by a human based on computational evidence] Weak
```

- From the resulting three-column table, read the genes in each operon and substitute each gene by its corresponding locus_tag (b-number):

```
fecABCD  b4291,b4290,b4289,b4288,b4287      Confirmed
fecIR    b4293,b4292                        Strong
aceBAK   b4014,b4015,b4016                  Strong
```

- **Note:** this list will contain operons with only one gene, which is fine because we will need those later when we create our negative control.

We have all the information we need to create the Positive Control.

- For each operon with two or more genes calculate the intergenic distances.

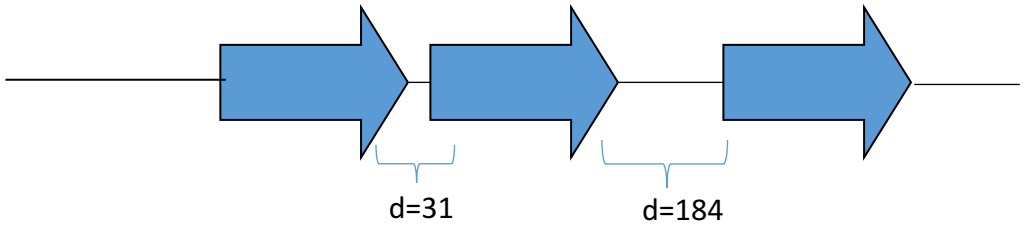
1. Get the left and right positions for all genes within the same operon:

SELECT g.gene_id,e.left_pos,e.right_pos,g.strand **FROM** genes g **JOIN** exons e **USING**(gene_id)
WHERE g.locus_tag **IN (**'b4014','b4015','b4016'**) ORDER BY** e.left **ASC**;

```
+---------+----------+-----------+--------+
| gene_id | left     | right     | strand |
+---------+----------+-----------+--------+
| 2471658 |  4213501 |   4215102 | F      |
| 2471659 |  4215132 |   4216436 | F      |
| 2471660 |  4216619 |   4218355 | F      |
+---------+----------+-----------+--------+
```

NOTE: If a gene has 2 or more exons, You need to take the left of the first exon and right of the last exons as coordinates for that gene.

2. Calculate the intergenic distances between the genes in the operon (left - right + 1)



3. Store all distances in an array. After processing all operons of two or more genes we can proceed to estimate the likelihood function for h_1 (histogram or model of the positive control). But first, let's get the data for the negative control.

Getting the data for the Negative Control.

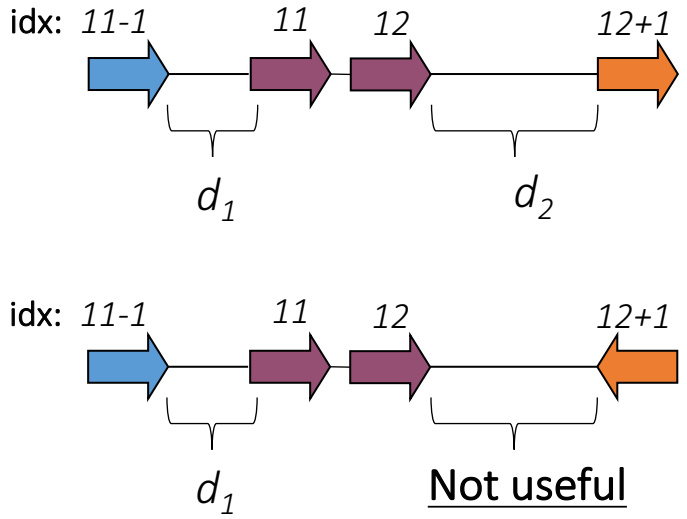
- Get all the genes in the genome (replicon) and sort them by left position. This facilitates getting the adjacent genes to the left and right of the borders of any operon in the genome.

```
SET @a:=0;
SELECT @a:=@a+1 as idx, g.gene_id,e.left,e.right,g.strand  FROM genes g JOIN exons e USING(gene_id)
WHERE g.genome_id=1 ORDER BY e.left_pos ASC;
```

The column **idx** can be used to identify the genes immediately to the left and right of a known operon. These are the genes that qualify as adjacent to the borders of the operon.

idx	gene_id	left_pos	right_pos	strand
1	2467885	190	255	F
2	2467886	337	2799	F
3	2467887	2801	3733	F
4	2467888	3734	5020	F
5	2467889	5234	5530	F
6	2467890	5683	6459	R
7	2467891	6529	7959	R
8	2467892	8238	9191	F
9	2467893	9306	9893	F
10	2467894	9928	10494	R

All genes in operons can be located in this indexed list of the genome. And we can verify that the neighbor genes are in the same strand than the operon.



With this we obtain all the distances between operon borders.
Exactly what we need to model our h_0.

Estimating the likelihoods for h_1 and h_0

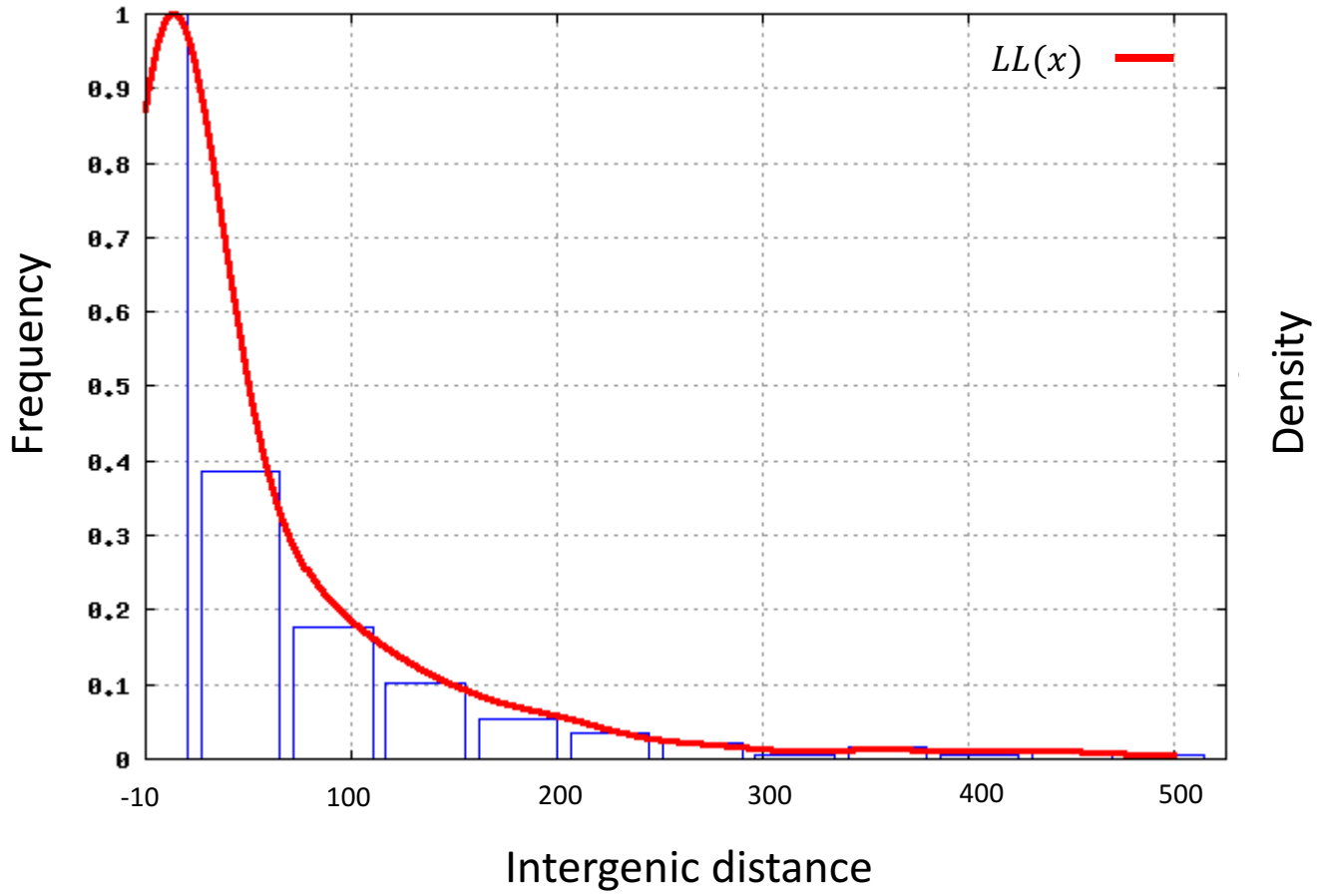
- We could estimate histograms where we show the frequency of distances with bin size of one base pair, and technically that would be enough. However:
  - Some bins could be empty (e.g. no data for distance 38) and we would be forced to make interpolations.
  - The histogram could be noisy with sudden jumps in frequencies.
- A better approach is to use Kernel Density Estimation (KDE) to model the histogram. This is a standard way to approach the probability density functions and it tackles the problem of smoothing noisy histograms.
  - Use python to calculate the density function (see for example: https://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation).
  - You only have to give the list with the distances in your control to the function and it will create the density function for you. For example:

```
kde = density (data, adjust=0.5);
LL = approxfun(kde)
```

Now, to get the density (equivalent to the frequency in a histogram) for a given distance x:

```
dens = LL(x)
```

Plotting the likelihood functions



Building our model

h_1

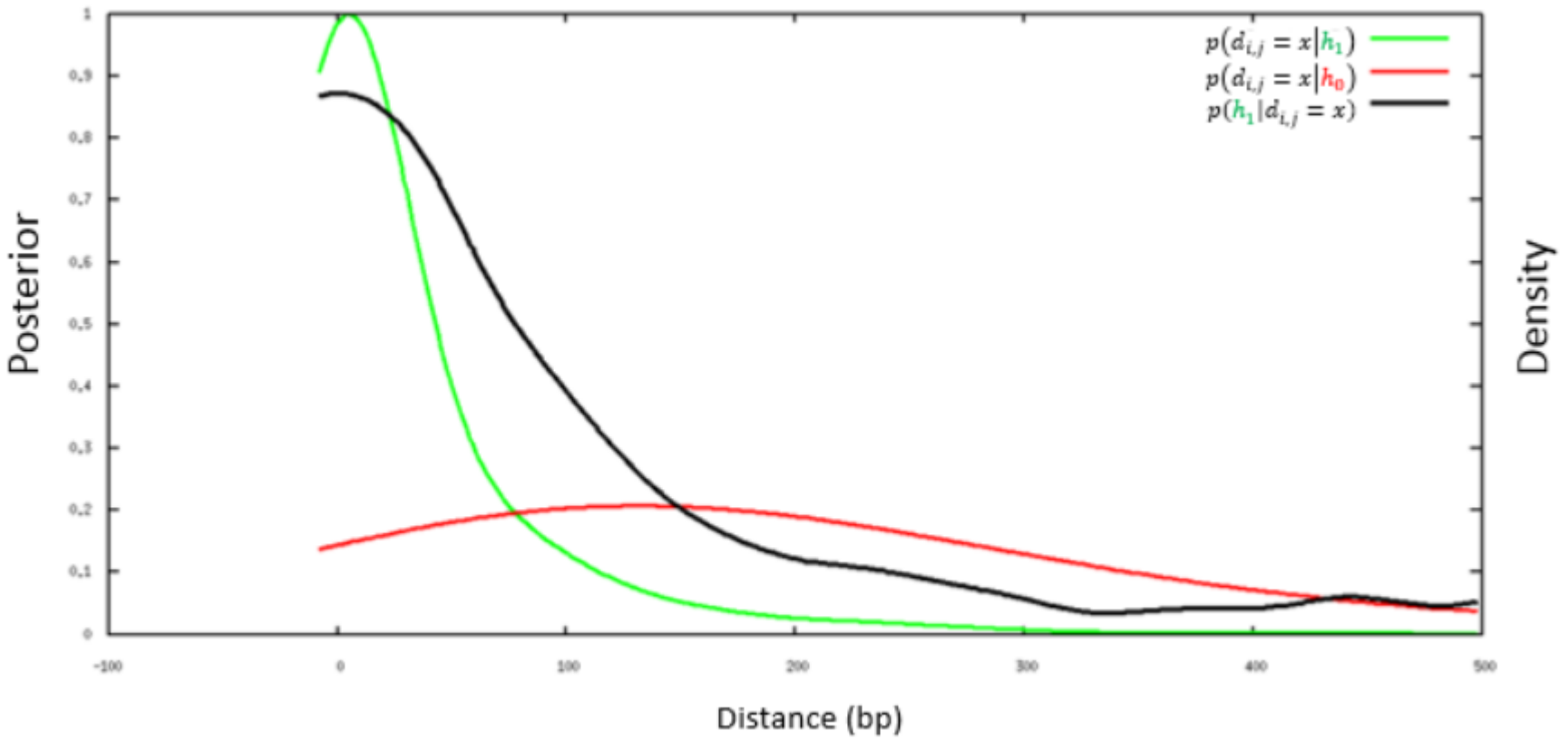
- Take the list of distances between genes in the same operons and build the density function $LL\_h_1(d_{i,j} = x)$.
- $LL\_h_1(x) = p(d_{i,j} = x | h_1)$

h_0

- Take the list of distances between genes at operon borders and build the density function $LL\_h_0(d_{i,j} = x)$.
- $LL\_h_0(x) = p(d_{i,j} = x | h_0)$

$$p(h_1 | d_{i,j} = x) = \frac{LL\_h_1(x)(0.6)}{LL\_h_1(x)(0.6) + LL\_h_0(x)(0.4)}$$

The posterior probability

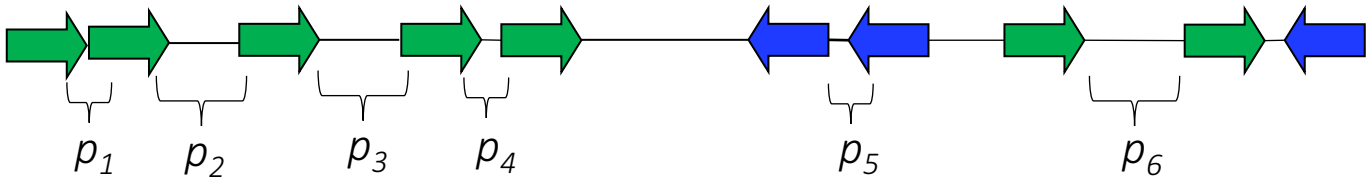


Making predictions in the complete genome

- Using the array that we created slide 17 we estimate all the directons in the genome first:

idx	gene_id	left_pos	right_pos	strand	
1	2467885	190	255	F	Directon 1
2	2467886	337	2799	F	
3	2467887	2801	3733	F	
4	2467888	3734	5020	F	
5	2467889	5234	5530	F	
6	2467890	5683	6459	R	Directon 2
7	2467891	6529	7959	R	
8	2467892	8238	9191	F	Directon 3
9	2467893	9306	9893	F	
10	2467894	9928	10494	R	Directon 4

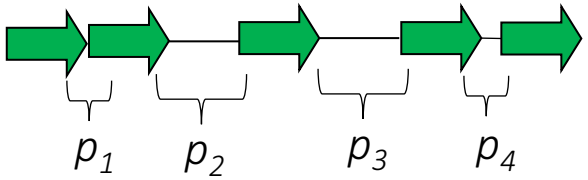
- For each directon with two or more genes we calculate the distances between pairs of adjacent genes and calculate their posterior probability of membership to the same operon.



Create a SQL table to put your predictions

```
CREATE TABLE tus (
  gid_1        INT   (10) UNSIGNED NOT NULL,
  gid_2        INT   (10) UNSIGNED NOT NULL,
  distance     INT   (10) UNSIGNED NOT NULL,
  status       ENUM('TP', 'TN') NOT NULL,
  prob         DOUBLE PRECISION NOT NULL,
  KEY (gid_1),
  KEY (gid_2)
) ENGINE=InnoDB;
```

- Fill the table with your inferences for each pair of genes. Notice that I didn't name the table operons. That is because our predictions are based on pairs of genes that may form a transcription unit, to get operons we would need to concatenate inferences at least a given probability value:



All five genes would form an operon if their posterior probability $p(h_1|d_{i,j})$ >= threshold.

Sensitivity

- Also known as the **true positive rate**, hit rate, recall or probability of detection.
- It measures the **fraction of correct inferences** detected by our model. In our case, the fraction of pairs of genes in the positive control set that were correctly classified as belonging to the same operon.
- This can be seen as **the extent to which our method did not miss true positives** (implying that false positives are few).
- A highly sensitive method rarely misses a true positive (e.g., it rarely infers that two genes are not in the same operon when the actually are).

$$\boldsymbol{Sensitiviy} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}$$

$\boldsymbol{TP}$: the number of True Positives detected by our model at a given classification threshold.

$\boldsymbol{FN}$: the number of False Negatives, or total true positives missed by our model at a given classification threshold.

Therefore $\boldsymbol{TP} + \boldsymbol{FN}$ is the total size of our positive test set.

Specificity

- Also known as the **true negative rate**.
- It measures the **fraction of negatives** that are correctly identified as true negatives (e.g., the fraction of adjacent pairs of genes at operon borders, which were correctly inferred as not belonging to the same operon).
- A highly specific method rarely confuses a true positive with a true negative (e.g., inferring that two genes are in the same operon when they the are not).

$$\boldsymbol{Specificity} = \frac{\boldsymbol{TN}}{\boldsymbol{TN} + \boldsymbol{FP}}$$

$\boldsymbol{TN}$: the number of True Negatives detected by our model at a given classification threshold.

$\boldsymbol{FP}$: the number of False Positives, or total true negatives missed by our model at a given classification threshold.

Therefore $\boldsymbol{TN} + \boldsymbol{FP}$ is the total size of our negative test set.

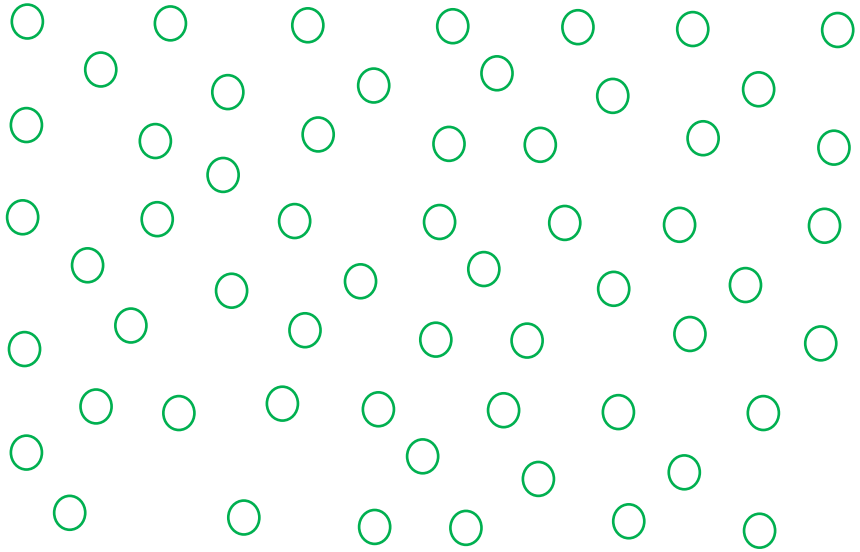
High Sensitivity

Posterior probability threshold

$$Sensitiviy = \frac{TP}{TP + FN}$$

Few false negatives (●)

True positives (○)



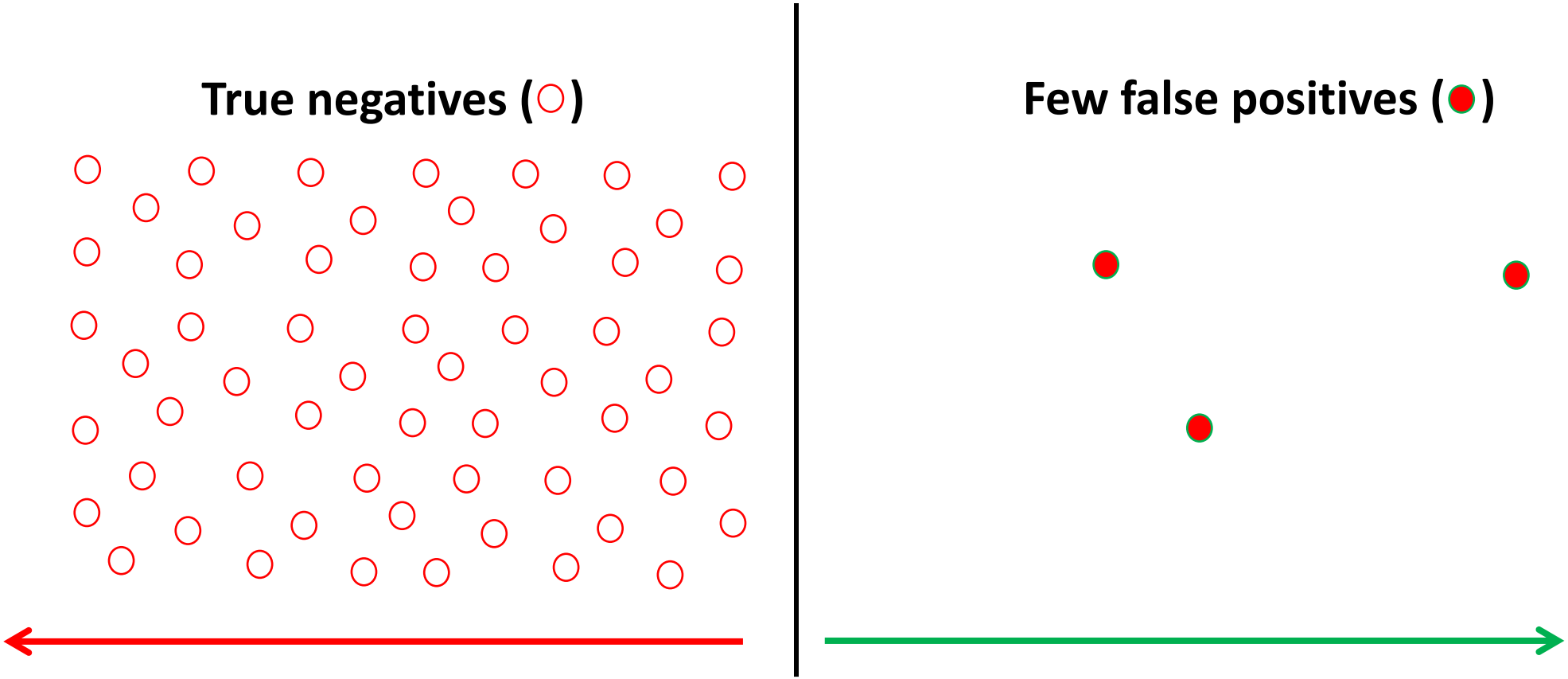
Genes at operon borders

Genes within the same operon

High Specificity

$$Specificity = \frac{TN}{TN + FP}$$

Posterior probability threshold



Sensitivity vs Specificity

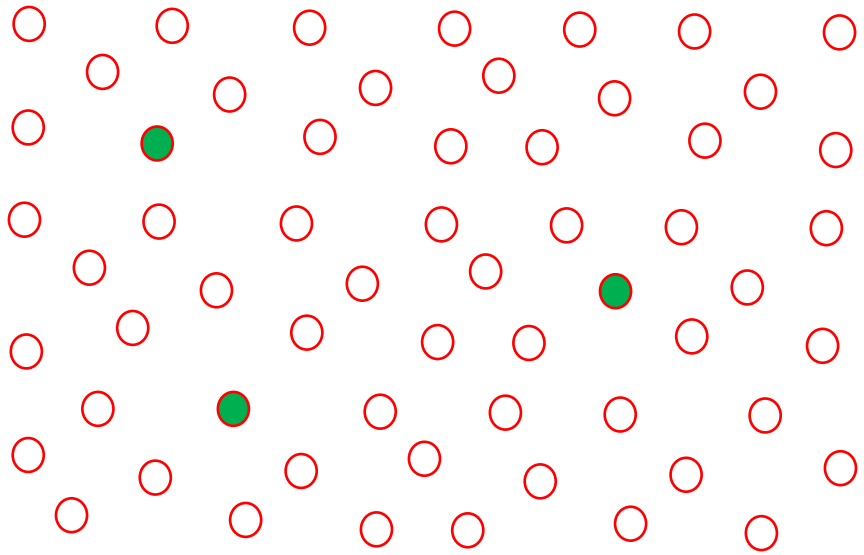
$$Specificity = \frac{TN}{TN + FP}$$

Posterior probability threshold

$$Sensitiviy = \frac{TP}{TP + FN}$$

High sensitivity

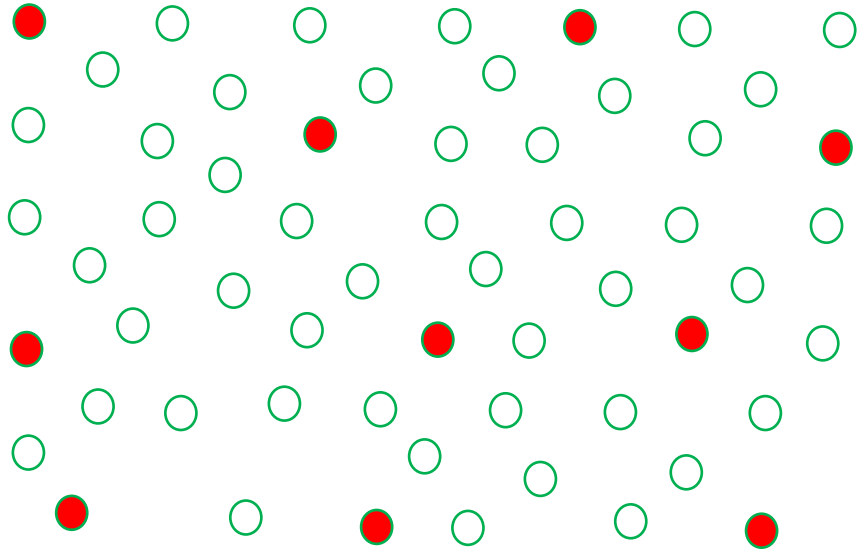
Few false negatives ( )



Genes at operon borders

Low specificity

Many false Positives ( )



Genes within the same operon

Sensitivity vs Specificity

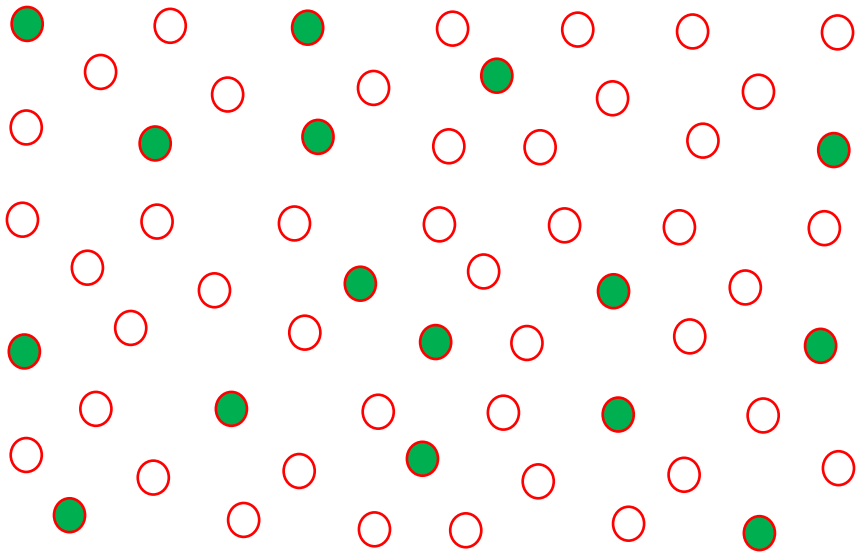
$$Specificity = \frac{TN}{TN + FP}$$

Posterior probability threshold

$$Sensitiviy = \frac{TP}{TP + FN}$$

Low sensitivity

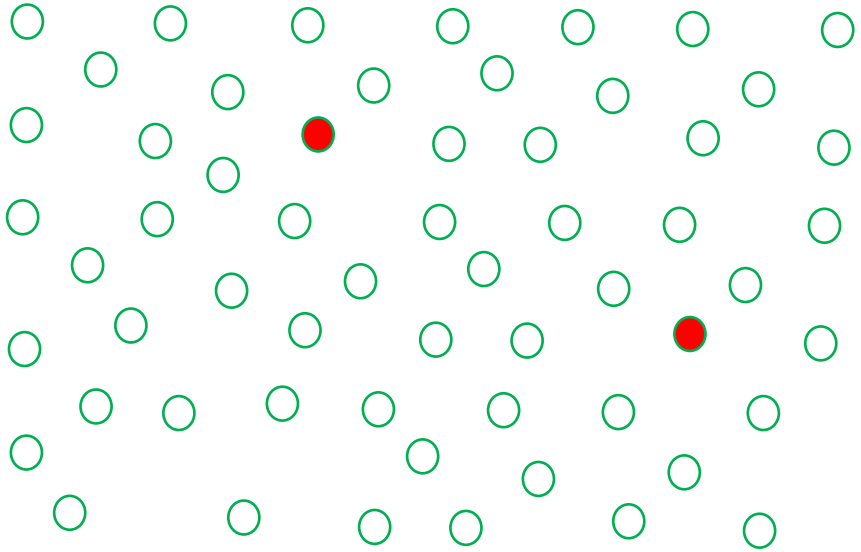
Many false negatives ( )



Genes at operon borders

High specificity

Few false Positives ( )



Genes within the same operon

Benchmarking the model

		Controls from RegulonDB		
		Genes in operons	Genes at operon borders	
Predictions of Transcription Units by our model	Positive prediction	TP	FP	$PPV = \frac{TP}{TP + FP}$
	Negative prediction	FN	TN	$NPV = \frac{TN}{TN + FN}$
		Sensitivity: $\frac{TP}{TP + FN}$	Specificity: $\frac{TN}{TN + FP}$	

Related Calculations

$$\boldsymbol{Sensitiviy} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}$$

$$\boldsymbol{Specificity} = \frac{\boldsymbol{TN}}{\boldsymbol{TN} + \boldsymbol{FP}}$$

False Negative rate (FNr):

$$\boldsymbol{FNr} = \frac{FN}{TP + FN} = 1 - Sensitivity$$

False Positive rate (FPr):

$$FPr = \frac{FP}{TN + FP} = 1 - Specificity$$

Likelihood ratio positive (LLp):

$$\boldsymbol{LL}p = \frac{Senstivity}{1 - Specificity} = \frac{True\ Positive\ rate}{False\ positive\ rate}$$

Likelihood ratio negative (LLn):

$$\boldsymbol{LL}n = \frac{1 - Senstivity}{Specificity} = \frac{False\ negative\ rate}{True\ negative\ rate}$$

Positive predictive value (PPV):

$$PPV = \frac{TP}{TP + FP}$$ (also known as **Precision**)

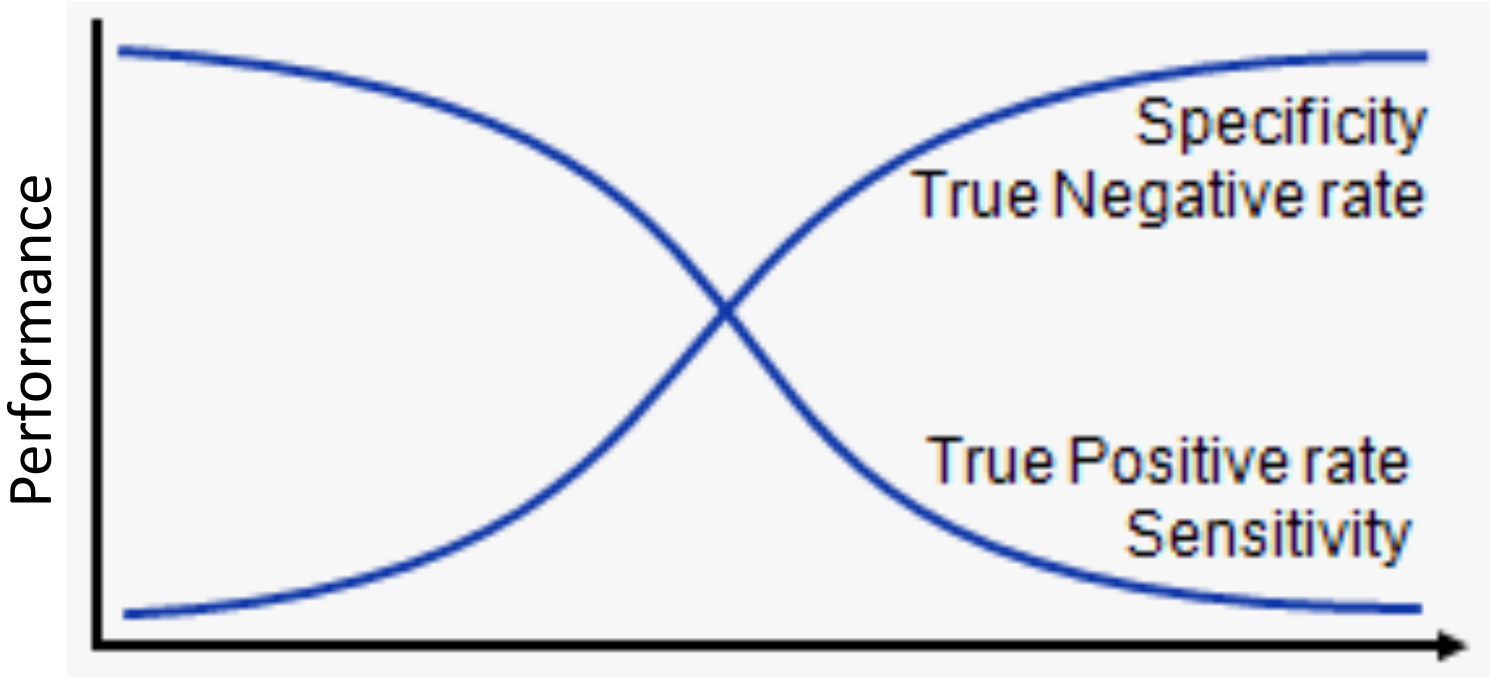
Negative predictive value (NPV):

$$NPV = \frac{TN}{TN + FN}$$

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

Sensitivity vs Specificity



$$Specificity = \frac{TN}{TN + FP}$$

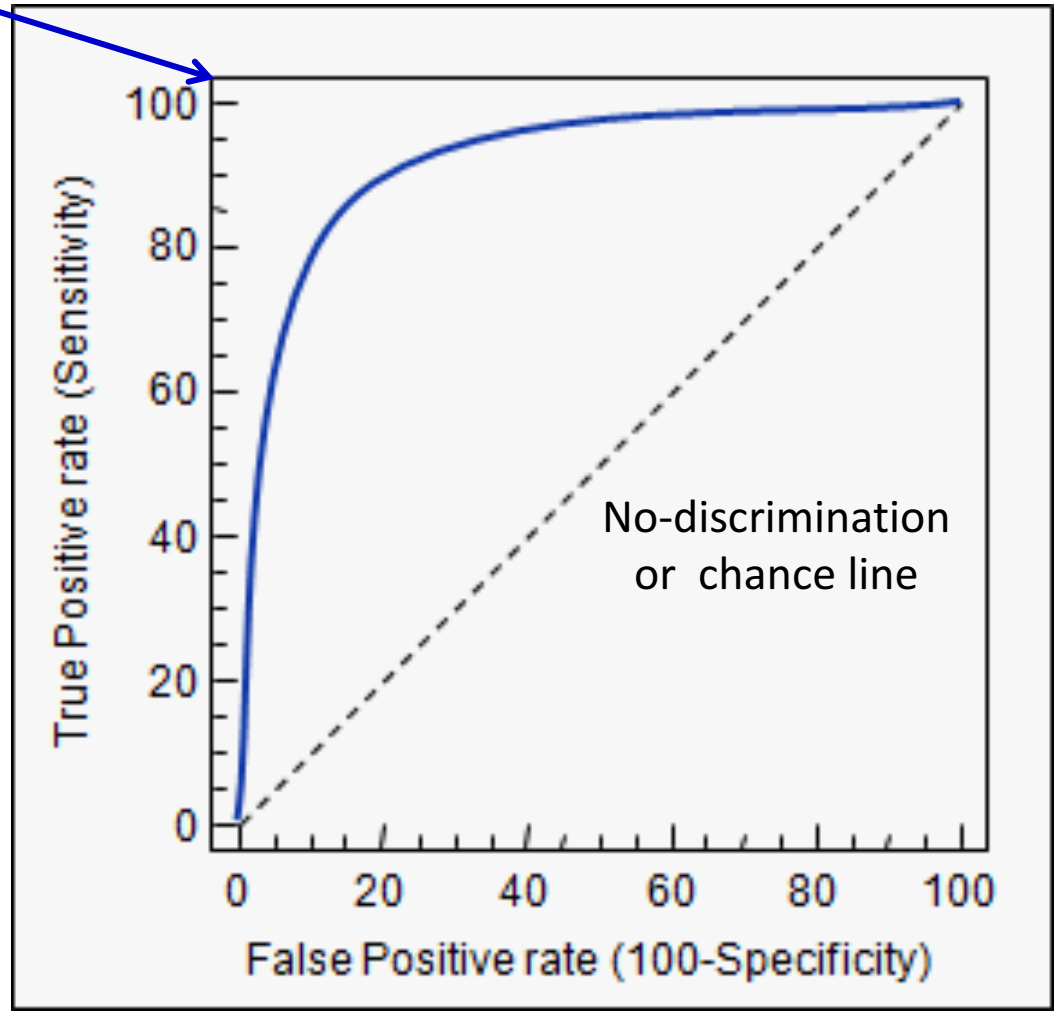
$$Sensitiviy = \frac{TP}{TP + FN}$$

Receiver Operator Characteristic Curve

Illustrates the discrimination ability of a binary classifier as the decision threshold is varied.

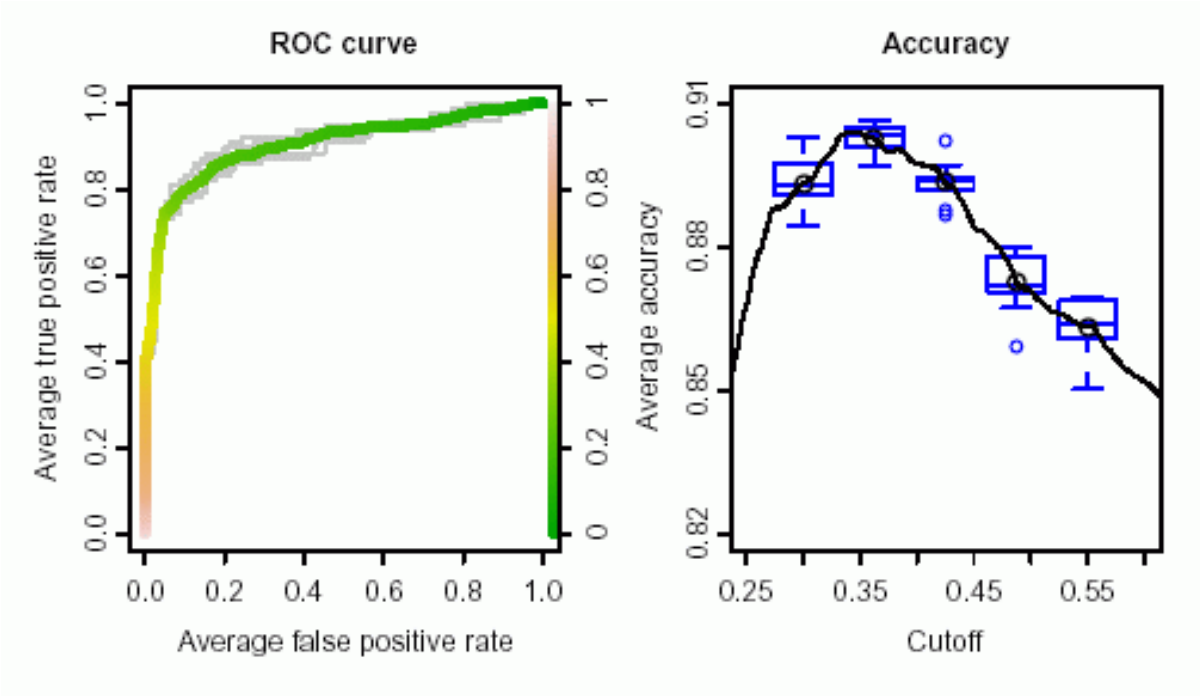
$$TPr = \frac{TP}{TP + FN}$$

$$FPr = \frac{FP}{TN + FP}$$



How to determine the best cutoff?

- This is still very arbitrary because it depends on our goals.
  - Maximize the sensitivity (detect all true pairs of genes within operons) at the expense of specificity (lots of false positives)
  - If both are important than we can chose the value that maximizes the distance between the ROC curve and the upper left corner of the graph. Or we can estimate the average accuracy per threshold using a bootstrapping.



Create a SQL table to put your predictions

```
CREATE TABLE tus (
  gid_1        INT   (10) UNSIGNED NOT NULL,
  gid_2        INT   (10) UNSIGNED NOT NULL,
  distance     INT   (10) UNSIGNED NOT NULL,
  status       ENUM('TP', 'TN') NOT NULL,
  prob         DOUBLE PRECISION NOT NULL,
  KEY (gid_1),
  KEY (gid_2)
) ENGINE=InnoDB;
```

Load your predictions and tag TP and TN

gid_1	gid_1	distance	status	prob
4	5	78		0.711570943478582
5	6	55	TP	0.80288879781484
11	12	93	TN	0.636225579410182
73	74	31		0.858797972342769
83	84	75	TP	0.725431332494521
84	85	23	TP	0.869403445736612
95	96	-4	TP	0.880789111794476
96	97	-17		0.873049916357744
151	152	31	TP	0.858797972342769
170	171	-4	TP	0.880789111794476
171	172	55		0.80288879781484
176	177	-11	TP	0.877786345937628
177	178	13		0.877837169972694
183	184	64	TP	0.771477259200511
203	204	-23		0.866079687248429
220	221	-4		0.880789111794476
292	293	152	TN	0.34348877466121
372	373	161	TN	0.30671284394195
1618	1619	156	TN	0.326829484898327
1650	1651	228		0.13289716513914
1834	1835	164	TN	0.295033074688328
2608	2609	268		0.100210089789783

- With all pairs of adjacent genes in the genome and the list of TP and TN in our controls we can add the status TP and TN to our table.
- For every probability threshold we define we can now estimate the Sensitivity, Specificity, Accuracy, etc.

Benchmarking the model

- As starting point we will use all the gene pairs in our Positive and Negative controls.
- For increments of 0.05 in the posterior probability calculate:
  - Sensitivity
  - Specificity
  - Positive predictive value (precision)
  - Accuracy.
- Create the plot of Sensitivity vs Specificity in slide 11
- Create the ROC curve in slide 12
- Plot accuracy as in slide 13