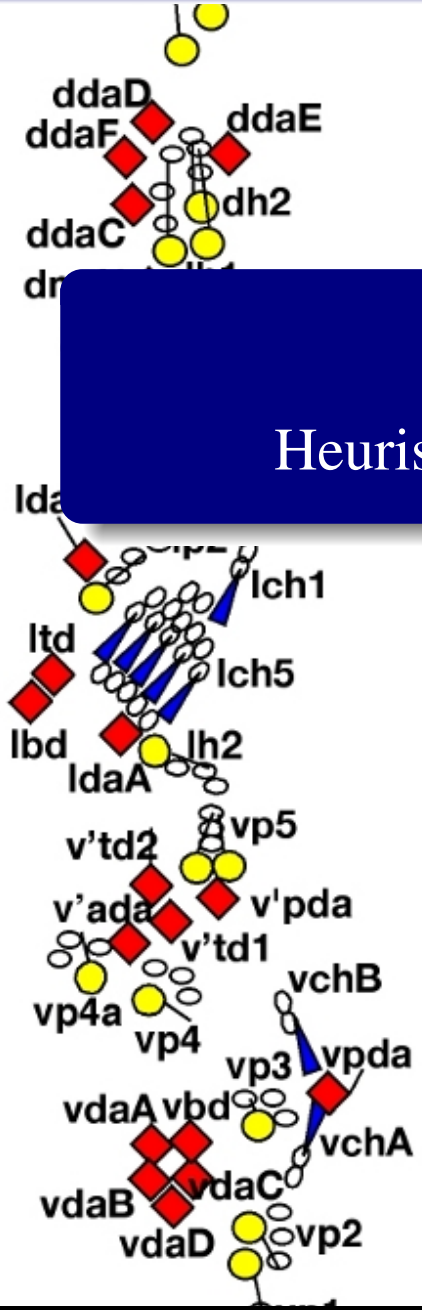


Bioinformatics 2 - Lecture 7

Heuristic methods, clustering and gene feature finding

Dr. Ian Simpson

Centre for Integrative Physiology
University of Edinburgh

Structure of the Lecture

Structure of the Lecture

1. Heuristics
   - Introduction to heuristics
   - Heuristics and Computational Biology
   - FASTA (fast all) - the original heuristic
   - Summary
2. Clustering
   - Introduction to clustering in Biology
   - Clustering example - Drosophila PNS development
   - Summary

Structure of the Lecture

1. Heuristics
   - Introduction to heuristics
   - Heuristics and Computational Biology
   - FASTA (fast all) - the original heuristic
   - Summary
2. Clustering
   - Introduction to clustering in Biology
   - Clustering example - Drosophila PNS development
   - Summary
3. Gene feature finding
   - Primer on gene regulation
   - DNA sequence searching
   - Transcription factor binding site prediction
   - Summary

What is a Heuristic?

a heuristic is

"..a method for problem solving...often involving experimentation and trial and error.."

and a heuristic algorithm is

"a heuristic, is an algorithm that is able to produce an acceptable solution to a problem in many practical scenarios, but for which there is no formal proof of its correctness"

Why use Heuristics ?

- Heuristics are typically used when there is no known method to find an optimal solution, under the given constraints or at all

Why use Heuristics ?

- Heuristics are typically used when there is no known method to find an optimal solution, under the given constraints or at all
- They are nearly always used for problems that are or are thought to be NP-hard (roughly, not computable in polynomial time)

Why use Heuristics ?

- Heuristics are typically used when there is no known method to find an optimal solution, under the given constraints or at all
- They are nearly always used for problems that are or are thought to be NP-hard (roughly, not computable in polynomial time)
- Allow us to incorporate knowledge about a problem or system to reduce the overall complexity of the task

Why use Heuristics ?

- Heuristics are typically used when there is no known method to find an optimal solution, under the given constraints or at all
- They are nearly always used for problems that are or are thought to be NP-hard (roughly, not computable in polynomial time)
- Allow us to incorporate knowledge about a problem or system to reduce the overall complexity of the task
- Can help to constrain search space and/or possible solution space to avoid erroneous solutions

What are the problems with Heuristics ?

- what comes next in the sequence : 1 2 4 .... ?

What are the problems with Heuristics ?

- what comes next in the sequence : 1 2 4 .... ?

  is it...

  1 2 4 7 11 16 22

What are the problems with Heuristics ?

- what comes next in the sequence : 1 2 4 .... ?

  is it...

  1 2 4 7 11 16 22

  or is it...

  1 2 4 8 16 32 64

What are the problems with Heuristics ?

- what comes next in the sequence : 1 2 4 .... ?

  is it...

  1 2 4 7 11 16 22

  or is it...

  1 2 4 8 16 32 64

  or is it...

  something completely different !?

What are the problems with Heuristics ?

- when working with heuristic algorithms you want speed and accuracy (optimal solutions), in reality you often lose one or both

What are the problems with Heuristics ?

- when working with heuristic algorithms you want speed and accuracy (optimal solutions), in reality you often lose one or both
- you cannot formally prove the solution is optimal and you cannot know that the algorithm will always be fast

What are the problems with Heuristics ?

- when working with heuristic algorithms you want speed and accuracy (optimal solutions), in reality you often lose one or both
- you cannot formally prove the solution is optimal and you cannot know that the algorithm will always be fast
- do not perform well when the underlying sample is small or the problem is ill defined

What are the problems with Heuristics ?

- when working with heuristic algorithms you want speed and accuracy (optimal solutions), in reality you often lose one or both
- you cannot formally prove the solution is optimal and you cannot know that the algorithm will always be fast
- do not perform well when the underlying sample is small or the problem is ill defined
- need to develop customised statistical models to go alongside the analysis to have confidence, normally randomisation based with it's associated sampling problems

The introduction of heuristics to the biology domain

- Dynamic programming was first used for accurate alignment of two sequences

  globally - Needleman Wunsch (1970)
  locally - Smith Waterman (1981)

The introduction of heuristics to the biology domain

- Dynamic programming was first used for accurate alignment of two sequences

  globally - Needleman Wunsch (1970)
  locally - Smith Waterman (1981)

- First heuristic algorithms developed in sequence analysis used both heuristics and dynamic programming

  FASTA - Lipman and Pearson 1985,1988
  Clustal - Higgins et al. 1988
  BLAST - Altschul et al. 1990

The introduction of heuristics to the biology domain

- Dynamic programming was first used for accurate alignment of two sequences
  - globally - Needleman Wunsch (1970)
  - locally - Smith Waterman (1981)
- First heuristic algorithms developed in sequence analysis used both heuristics and dynamic programming
  - FASTA - Lipman and Pearson 1985,1988
  - Clustal - Higgins et al. 1988
  - BLAST - Altschul et al. 1990
- Heuristics are now epidemic in Bioinformatics applied to
  - classic alignment and sequence search problems
  - cluster editing, partitioning problem solving
  - phylogenetic parsimony
  - motif detection
  - protein docking
  - protein structure resolution

FASTA - a heuristic sequence searching algorithm

- used to query large sequence databases with sequences DNA/Protein
  - for example searching for a 20mer oligo in a genome of 150Mb

FASTA - a heuristic sequence searching algorithm

- used to query large sequence databases with sequences DNA/Protein
  - for example searching for a 20mer oligo in a genome of 150Mb
- can perform gapped local alignments

FASTA - a heuristic sequence searching algorithm

- used to query large sequence databases with sequences DNA/Protein
  - for example searching for a 20mer oligo in a genome of 150Mb
- can perform gapped local alignments
- performs optimized searches for local alignment using substitution matrices (identity for DNA, BLOSUM/PAM for protein)

FASTA - a heuristic sequence searching algorithm

- used to query large sequence databases with sequences DNA/Protein
  - for example searching for a 20mer oligo in a genome of 150Mb
- can perform gapped local alignments
- performs optimized searches for local alignment using substitution matrices (identity for DNA, BLOSUM/PAM for protein)
- slower than BLAST, but more sensitive for nucleotides and particularly good for repetitive sequence

FASTA - a heuristic sequence searching algorithm

- Variables
  - ktup: word-length (similar to BLAST)
    1-2 for proteins, 4-6 for nucleotides
  - gap opening penalties : -12 (protein) and -16 (DNA)
  - gap extension penalties : -2 (protein) and -4 (DNA)

FASTA - a heuristic sequence searching algorithm

- Variables
  - ktup: word-length (similar to BLAST)
    1-2 for proteins, 4-6 for nucleotides
  - gap opening penalties : -12 (protein) and -16 (DNA)
  - gap extension penalties : -2 (protein) and -4 (DNA)
- Statistics
  - Z-scores : calculated normalised by sequence length
  - E (expectation) scores : number of sequences expect with same score by chance

The main steps of the FASTA algorithm

- Step One
  - Find exact matches of word size between query and target, record in a hash/lookup table
  - hash/lookup can be pre-computed for different searches k=1 (oligo 20nt), k=6 (normal 100-500nt)

The main steps of the FASTA algorithm

- Step One
  - Find exact matches of word size between query and target, record in a hash/lookup table
  - hash/lookup can be pre-computed for different searches k=1 (oligo 20nt), k=6 (normal 100-500nt)
- Step Two
  - cluster the 'hot-spots' into diagonals by making a matrix of 1s and 0s by position
  - score all of the diagonals with each region + and each gap -
  - find the 10 best diagonals and then perform a local alignment with no indels
  - the best partial alignment is called init1 and is used in Step Four

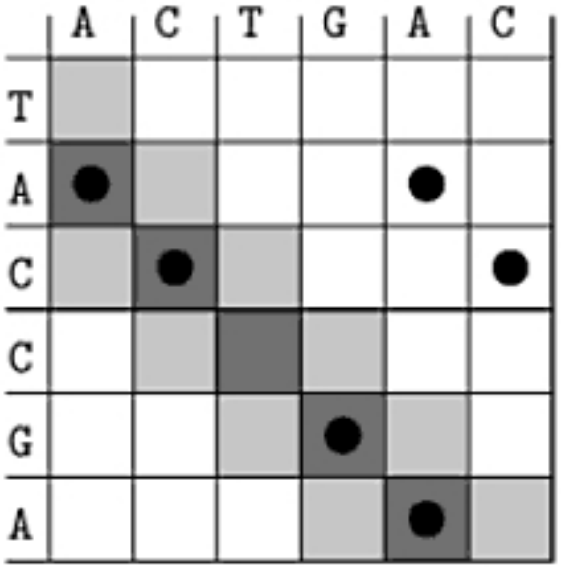
The main steps of the FASTA algorithm

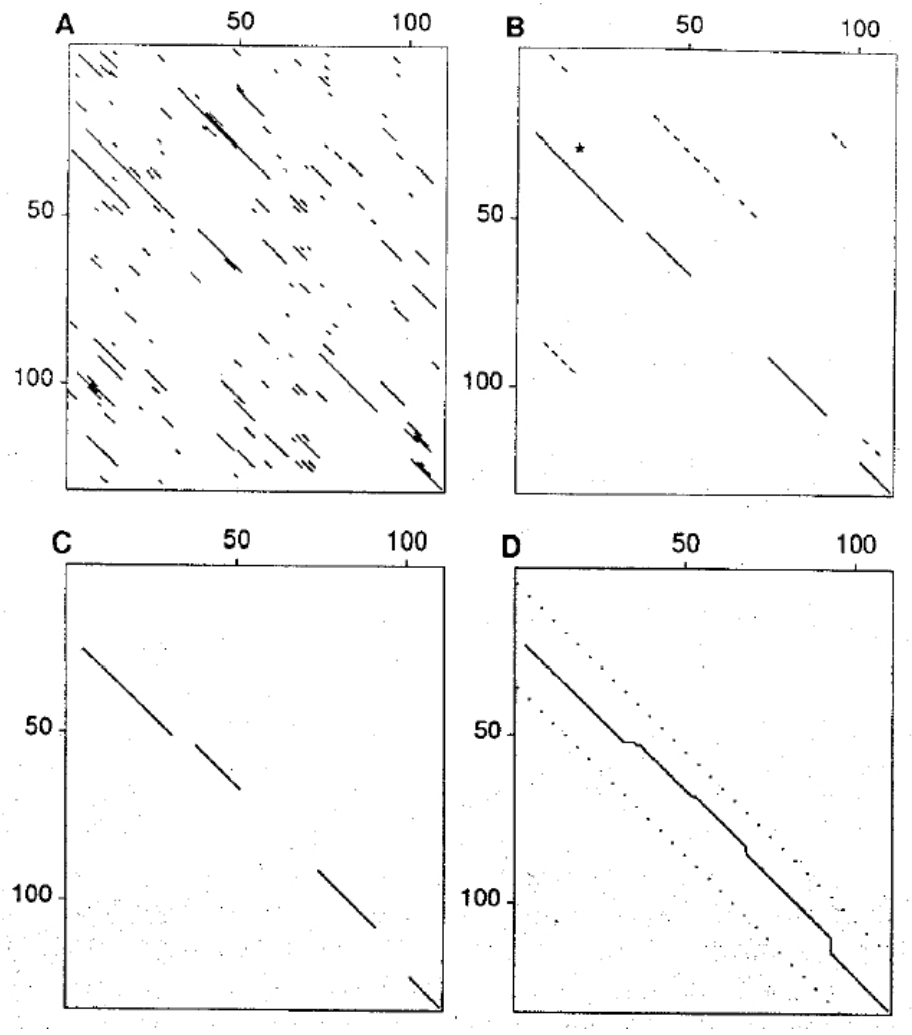
- Step Three
  - going back to the 10 partial alignments, the algorithm now takes any that exceed a certain score cut-off and tries to make them into longer alignment runs
  - if a longer partial can be made (and this is a graph theoretic problem) it is optimally aligned and returned as one result from the algorithm

The main steps of the FASTA algorithm

- Step Three
  - going back to the 10 partial alignments, the algorithm now takes any that exceed a certain score cut-off and tries to make them into longer alignment runs
  - if a longer partial can be made (and this is a graph theoretic problem) it is optimally aligned and returned as one result from the algorithm
- Step Four
  - picking up the init1 partial alignment from Step Two the algorithm performs a banded Smith Waterman
  - a window of alignment space either side of the init1 diagonal is identified and optimal local alignments are performed throughout the space
  - the alignments are scored by matrix and statistics are calculated, normalised Z-score and E value

Schematic of the Fasta matrix process





Example FASTA result

```
>>UNIPROT:FOSB_MOUSE P13346 Protein fosB.                    (338 aa)
 initn: 2268 init1: 2268 opt: 2268  Z-score: 1866.3  bits: 353.7 E(): 3.3e-96
Smith-Waterman score: 2268;  100.000% identity (100.000% ungapped) in 338 aa overlap (1-338:1-338)

               10        20        30        40        50        60
FOSB_M MFQAFPGDYDSGSRCSSSPSAESQYLSSVDSFGSPPTAAASQECAGLGEMPGSFVPTVTA
       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
UNIPRO MFQAFPGDYDSGSRCSSSPSAESQYLSSVDSFGSPPTAAASQECAGLGEMPGSFVPTVTA
               10        20        30        40        50        60

               70        80        90       100       110       120
FOSB_M ITTSQDLQWLVQPTLISSMAQSQGQPLASQPPAVDPYDMPGTSYSTPGLSAYSTGGASGS
       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
UNIPRO ITTSQDLQWLVQPTLISSMAQSQGQPLASQPPAVDPYDMPGTSYSTPGLSAYSTGGASGS
               70        80        90       100       110       120

              130       140       150       160       170       180
FOSB_M GGPSTSTTTSGPVSARPARARPRRPREETLTPEEEEKRRVRRERNKLAAAKCRNRRRELT
       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
UNIPRO GGPSTSTTTSGPVSARPARARPRRPREETLTPEEEEKRRVRRERNKLAAAKCRNRRRELT
              130       140       150       160       170       180

              190       200       210       220       230       240
FOSB_M DRLQAETDQLEEEKAELESEIAELQKEKERLEFVLVAHKPGCKIPYEEGPGPGPLAEVRD
       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
UNIPRO DRLQAETDQLEEEKAELESEIAELQKEKERLEFVLVAHKPGCKIPYEEGPGPGPLAEVRD
              190       200       210       220       230       240
```

Example FASTA histogram output

```
       opt      E()
< 20  1040     0:=
  22     0     0:           one = represents 1534 library sequences
  24     4     1:*
  26    14    19:*
  28    53   201:*
  30   227  1223:*
  32  1161  4728:=  *
  34  5065 12823:====    *
  36 16019 26336:==========      *
  38 33416 43523:======================      *
  40 58656 60711:=======================================*
  42 81562 74211:================================================*=====
  44 87125 81862:=======================================================*===
  46 92000 83378:=======================================================*=====
  48 83023 79825:=====================================================*==
  50 83389 72841:=================================================*=======
  52 68683 64039:=========================================*===
  54 58489 54701:===================================*===
  56 48841 45692:=============================*==
  58 36330 37512:======================*
  60 26755 30387:================= *
  62 21306 24361:============== *
  64 17453 19374:============*
  66 13701 15313:=========*
  68 10436 12045:=======*
  70  8222  9439:======*
  72  6047  7376:====*
```

Heuristics summary

- Heuristics are used to reduce the complexity of problems that are not computationally tractable

Heuristics summary

- Heuristics are used to reduce the complexity of problems that are not computationally tractable
- Prior knowledge and reasonable assumptions about the system and charcateristics of likely solutions are needed

Heuristics summary

- Heuristics are used to reduce the complexity of problems that are not computationally tractable
- Prior knowledge and reasonable assumptions about the system and charcateristics of likely solutions are needed
- Statistical methods need to be developed to test the fidelity of the heuristic results, these are typically randomisation or bootstrap type methods

Heuristics summary

- Heuristics are used to reduce the complexity of problems that are not computationally tractable
- Prior knowledge and reasonable assumptions about the system and charcateristics of likely solutions are needed
- Statistical methods need to be developed to test the fidelity of the heuristic results, these are typically randomisation or bootstrap type methods
- Heuristics are used widely in computational biology especially in studies using genome scale data, proteomics, transcriptomics, phylgenetics etc..

Finding groups in data

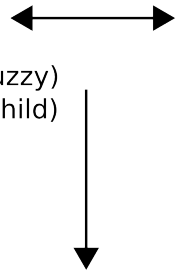
- finding trends and groupings within high order complex datasets is fundamental to many computational biology projects
  - Proteomics - protein-protein interaction data
  - Functional annotation clustering - grouping genes by function
  - Transcriptomics - grouping genes by expression profile by condition
- in large datsets the ditinction between groupings can be obtuse and many parallel methods are often used to try to validate the clustering results
  - protein-protein interactions tend to be multplicitous and single group membership may not be appropriate
  - functional annotation clustering is constrained by ontologies and provides a unique, and unsolved? problem
  - gene expression data - expression on a continuous scale with high noise, often need to pre-transform data to reduce dimensionality and/or exacerbate distinctions between groups

Classic clustering methodologies

- Slides adapted from to Dr. Dirk Husmeier, BioSS Scotland
- Would take a long time to do this in Beamer.....

MAGIC - multi algorithmic grouping with integrity checking

multi-algorithm clustering
agnes - heirarchical (agg)
PAM - partional(medoid)
fuzzy-c - partial membership (fuzzy)
hpach - combnination (parent-child)

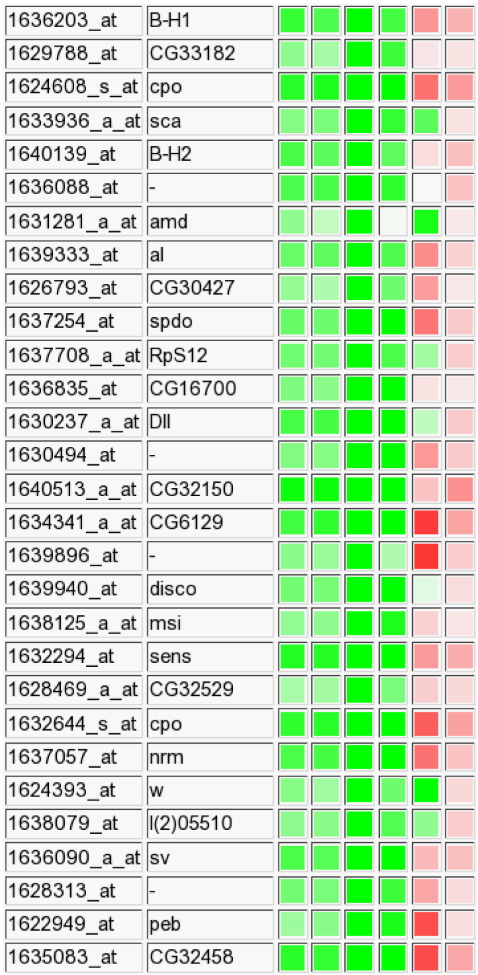


parameter optimisation
average silhouette width
median split silhouette
elbow-plot

experimental cluster sets

cluster comparison
set-operation cluster list comparison
matrix of unit sets
expands into dimensions (robustness)
conservation scoring by membership
cluster ranking
bootstrap resampling - confidence intervals

ranked conserved cluster sets



hopach5 - bristle morphogenesis

MAGIC - finishing off the pipeline with a statistical analysis

- Determine the statistical significance of the scores for each cluster - bootstrap confidence estimation
  - generate many random cluster sets with the same pool of members and the same structure for each cluster and each clustering experiment
  - score each of the random sets to build up a distribution that estimates the true distribution of scores
  - fit a probability density function to the bootstrap distribution
  - calculate p-values for the scores generated from the clusters of the experimental data sets

MAGIC - finishing off the pipeline with a statistical analysis

- Determine the statistical significance of the scores for each cluster - bootstrap confidence estimation
  - generate many random cluster sets with the same pool of members and the same structure for each cluster and each clustering experiment
  - score each of the random sets to build up a distribution that estimates the true distribution of scores
  - fit a probability density function to the bootstrap distribution
  - calculate p-values for the scores generated from the clusters of the experimental data sets
- rank clusters by score with an associated p-value that is a measure of how far away the cluster membership is than randomly populated clusters of the same structure

Clustering summary

- Many methodologically distinct methods have been developed both classical and modelled
  - heirarchical, partitioning, fuzzy, combinatorial

Clustering summary

- Many methodologically distinct methods have been developed both classical and modelled
  - heirarchical, partitioning, fuzzy, combinatorial
- Many distance measures can be used depending on the distribution of the data
  - euclidean, mahalanobis, cosine..

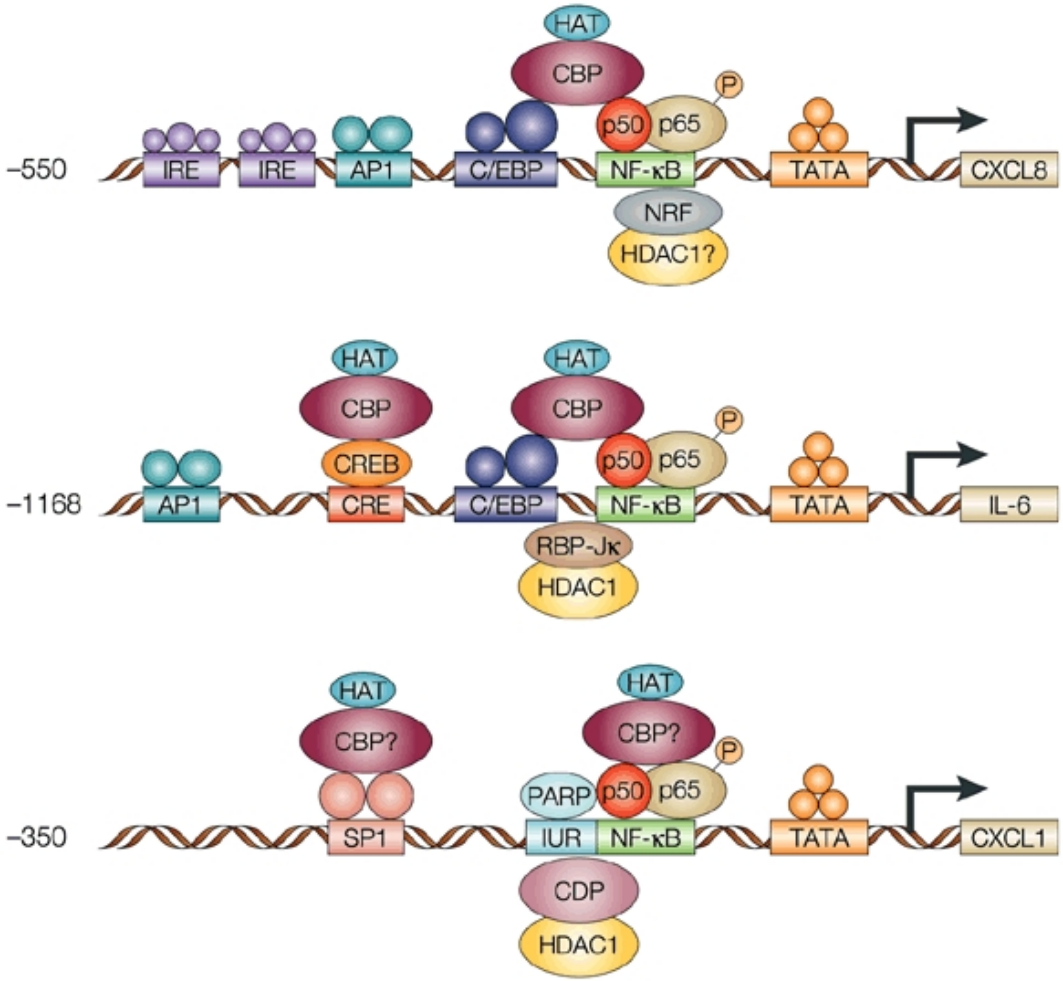
Clustering summary

- Many methodologically distinct methods have been developed both classical and modelled
  - heirarchical, partitioning, fuzzy, combinatorial
- Many distance measures can be used depending on the distribution of the data
  - euclidean, mahalanobis, cosine..
- Many parameter optimisation methods have been developed
  - median split-silhouette, elbow plot, GAP statistics...

Clustering summary

- Many methodologically distinct methods have been developed both classical and modelled
  - heirarchical, partitioning, fuzzy, combinatorial
- Many distance measures can be used depending on the distribution of the data
  - euclidean, mahalanobis, cosine..
- Many parameter optimisation methods have been developed
  - median split-silhouette, elbow plot, GAP statistics...
- Now integrative pipelines are being developed to cross-compare results from clustering using a whole range of algorithms, variables and measures - so called consensus clustering

Anatomy of a promoter/enhancer



Promoter/enhancer features

- promoters and enhancers contain binding sites for transcription (TFBS) and transcription associated factors

Promoter/enhancer features

- promoters and enhancers contain binding sites for transcription (TFBS) and transcription associated factors
- promoters are close to the transcriptional start of the gene

Promoter/enhancer features

- promoters and enhancers contain binding sites for transcription (TFBS) and transcription associated factors
- promoters are close to the transcriptional start of the gene
- enhancers can be very far away from the gene

Promoter/enhancer features

- promoters and enhancers contain binding sites for transcription (TFBS) and transcription associated factors
- promoters are close to the transcriptional start of the gene
- enhancers can be very far away from the gene
- TFBS sites are used in complex combinations to modulate the time, location and level of expression of genes

Promoter/enhancer features

- promoters and enhancers contain binding sites for transcription (TFBS) and transcription associated factors
- promoters are close to the transcriptional start of the gene
- enhancers can be very far away from the gene
- TFBS sites are used in complex combinations to modulate the time, location and level of expression of genes
- TFBS sites are generally small 6-8nt and are also degenerate (more than one sequence can perform the same or similar task)

Examples of TFBS binding sites

Atonal E-box

Scute E-box

Problems finding TFBS sites in promoters and enhancers

- objective is to find TFBS sites according to defined criteria and predict which are functional

Problems finding TFBS sites in promoters and enhancers

- objective is to find TFBS sites according to defined criteria and predict which are functional
- by chance TFBS sites are found with relatively high frequency in the genome as they are small. This means that finding the true TFBS sites is an inheritantly noisy process

Problems finding TFBS sites in promoters and enhancers

- objective is to find TFBS sites according to defined criteria and predict which are functional
- by chance TFBS sites are found with relatively high frequency in the genome as they are small. This means that finding the true TFBS sites is an inheritantly noisy process
- sites for a particular transcription factor are most commonly defined as regular expressions or position weight matrices (PWMs)

Problems finding TFBS sites in promoters and enhancers

- objective is to find TFBS sites according to defined criteria and predict which are functional
- by chance TFBS sites are found with relatively high frequency in the genome as they are small. This means that finding the true TFBS sites is an inheritantly noisy process
- sites for a particular transcription factor are most commonly defined as regular expressions or position weight matrices (PWMs)
- reg exps produce binary results (0,1), but searches with PWMs produce continuous scores and probabilities, i.e. uncertainty

Problems finding TFBS sites in promoters and enhancers

- objective is to find TFBS sites according to defined criteria and predict which are functional
- by chance TFBS sites are found with relatively high frequency in the genome as they are small. This means that finding the true TFBS sites is an inheritantly noisy process
- sites for a particular transcription factor are most commonly defined as regular expressions or position weight matrices (PWMs)
- reg exps produce binary results (0,1), but searches with PWMs produce continuous scores and probabilities, i.e. uncertainty
- need ways to reduce complexity and or search space

Regular expressions

- a simple consensus is essentially a regular expression such as for example CANNTG, the E-box consensus
  - possibilities are CAAATG, CAATTG...etc
  - you could express this as a regular expression CA[ACTG]{2}TG and search for matches
  - the result is a hit sequence and a location, it's binary

Regular expressions

- a simple consensus is essentially a regular expression such as for example CANNTG, the E-box consensus
  - possibilities are CAAATG, CAATTG...etc
  - you could express this as a regular expression CA[ACTG]{2}TG and search for matches
  - the result is a hit sequence and a location, it's binary
- Problems with regular expressions
  - assume that all possible permutations are equal
  - in order to be informative you have to exclude what could be informative, but low frequency, sequences from the consensus (so that you don't have an E-box of [ACTG]{6} for example !
  - there are currently two main solutions to this problem, position weight matrices and hidden markov model profiles

Weight matrices, PWMs

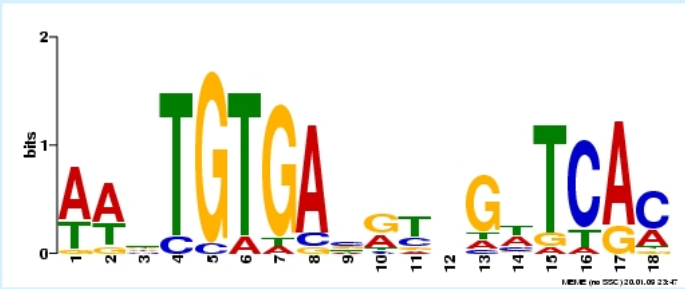
	A	C	T	G
1	101	-1081	-182	13
2	87	-1081	-82	13
3	-45	-23	18	35
4	-1081	-82	-1081	155
5	-1081	-182	227	-1081
6	-145	-1081	-1081	155
7	-245	-1081	218	-245
8	145	-82	-182	-1081

SEQUENCE LOGO

Information Content
14.7 (bits)

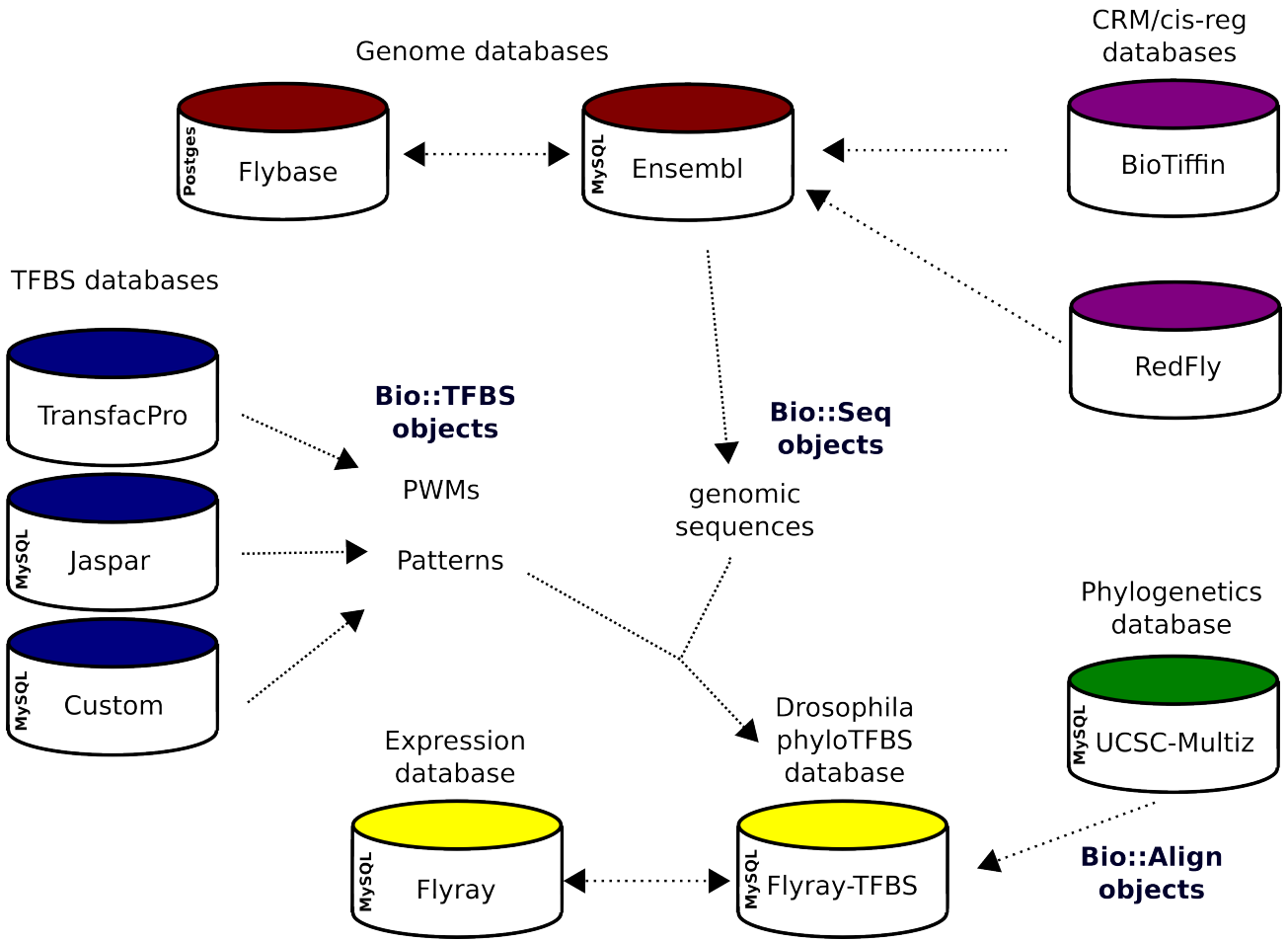
Relative Entropy
14.4 (bits)

Download LOGO
Without SSC:[EPS][PNG]
With SSC:[EPS][PNG]



NAME	STRAND	START	P-VALUE	SITES
ara	-	58	2.51e-07	TGGCATAGCA AAGTGTGACGCCGTGCAA ATAATCAATG
lac	+	8	5.35e-07	AACGCAAT TAATGTGAGTTAGCTCAC TCATTAGGCA
malt	+	40	8.61e-07	AAAGATTTGG AATTGTGACACAGTGCAA ATTCAGACAC
ilv	-	42	1.69e-06	GCAAAGGGAA AATTGAGGGGTTGATCAC GTTTTGTACT
pbr322	-	56	2.85e-06	CTCCTTACGC ATCTGTGCGGTATTTCAC ACCGCATATG
deop2	+	59	2.85e-06	AGATTTCCTT AATTGTGATGTGTATCGA AGTGTGTTGC
uxu1	+	16	5.17e-06	AGAGTGAAAT TGTTGTGATGTGGTTAAC CCAATTAGAA
trn9cat	+	83	5.69e-06	CTTTTGGCGA AAATGAGACGTTGATCGG CACG
ce1cg	-	64	7.54e-06	GGACTTCCAT TTTTGTGAAAACGATCAA AAAAACAGTC
ompa	+	47	9.04e-06	TTTTTTTCAT ATGCCTGACGGAGTTCAC ACTTGTAAGT

Apply a PWMSearch but with databases



TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream
  - Avoided CRM data as it assumes levels of conservation to define ranges, not objective

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream
  - Avoided CRM data as it assumes levels of conservation to define ranges, not objective

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream
  - Avoided CRM data as it assumes levels of conservation to define ranges, not objective
- TFBS PWM and pattern screen
  - Screened >20,000 sites/patterns and >800 PWMs from TransfacPro

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream
  - Avoided CRM data as it assumes levels of conservation to define ranges, not objective
- TFBS PWM and pattern screen
  - Screened >20,000 sites/patterns and >800 PWMs from TransfacPro
  - Screened 123 PWMs from Jaspar and our in-house patterns

TFBS screening - searching for sites

- Genomic sequence selection
  - pre-screened genome to determine size range for 1kbp upstream - end of intron 1
  - TESS uses proximal 300bp upstream
  - Pre-computes from Flybase and AAA less than 1kbp upstream
  - Settled on primary screen of 1kbp upstream to 1kbp downstream
  - Avoided CRM data as it assumes levels of conservation to define ranges, not objective
- TFBS PWM and pattern screen
  - Screened >20,000 sites/patterns and >800 PWMs from TransfacPro
  - Screened 123 PWMs from Jaspar and our in-house patterns
  - Total of 20 percent of the Drosophila melanogaster genome screened producing $3.5x10^6$ hits

Calculating TFBS site conservation, phylogenetics

- UCSC 15-way Multiz alignments - phylogenetic analysis
  - comprises approximately 2 million alignment blocks (MSAs) indexed by Dmel scaffold

Calculating TFBS site conservation, phylogenetics

- UCSC 15-way Multiz alignments - phylogenetic analysis
  - comprises approximately 2 million alignment blocks (MSAs) indexed by Dmel scaffold
  - converted Multiz files into Bio::Align objects indexed in a MySQL database by Dmel chr:start:end

Calculating TFBS site conservation, phylogenetics

- UCSC 15-way Multiz alignments - phylogenetic analysis
  - comprises approximately 2 million alignment blocks (MSAs) indexed by Dmel scaffold
  - converted Multiz files into Bio::Align objects indexed in a MySQL database by Dmel chr:start:end
  - for each sequence range to be analysed pulled all of the alignment blocks, stripped out clean sequences and re-aligned, pairwise between Dmel and the other 11 species sequences

Calculating TFBS site conservation, phylogenetics

- UCSC 15-way Multiz alignments - phylogenetic analysis
  - comprises approximately 2 million alignment blocks (MSAs) indexed by Dmel scaffold
  - converted Multiz files into Bio::Align objects indexed in a MySQL database by Dmel chr:start:end
  - for each sequence range to be analysed pulled all of the alignment blocks, stripped out clean sequences and re-aligned, pairwise between Dmel and the other 11 species sequences
  - retrieved all TFBS site hit data from screen and scored every site to every pairwise alignment

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation
- Calculating the best measure of conservation
  - Optimise the way in which conservation score is calculated to maximise true positives and minimise false negatives

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation
- Calculating the best measure of conservation
  - Optimise the way in which conservation score is calculated to maximise true positives and minimise false negatives
  - Use summation, average, weighted summation and weighted average where the weighting is a function of species divergence

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation
- Calculating the best measure of conservation
  - Optimise the way in which conservation score is calculated to maximise true positives and minimise false negatives
  - Use summation, average, weighted summation and weighted average where the weighting is a function of species divergence
  - Consider using micro-sequence evolution in windows around sites

Sequence conservation as a proxy for sequence function

- Assessing the relationship between conservation and function
  - In order to meaningfully use any sequence conservation as a proxy for function we need to determine as best we can the relationship between conservation and function
  - TransfacPro Site data - a positive training set
  - Map TFPro DMel sites onto the genome and calculate pairwise conservation scores - Multiz - positive training set
  - Randomise sequence sets from the whole genome (non-coding) and score from Multiz - background estimation
- Calculating the best measure of conservation
  - Optimise the way in which conservation score is calculated to maximise true positives and minimise false negatives
  - Use summation, average, weighted summation and weighted average where the weighting is a function of species divergence
  - Consider using micro-sequence evolution in windows around sites
  - Calculate on a per site basis (i.e. randomise per site not for all sites) some sites will be more informative than others, drop the uninformative ones

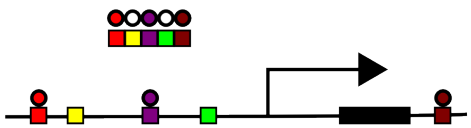
Rfx, the X-box and ciliated sensory neuron development

- Ciliated sensory neurons
  - Most sensory neurons have cilia at their dendritic tips
  - Cilia play crucial and highly conserved roles in motility, molecular transport and developmental processes such as left-right symmetry and sense organ development
  - Mutations in Rfx proteins are associated with defects in ciliogenesis in many organisms including Drosophila

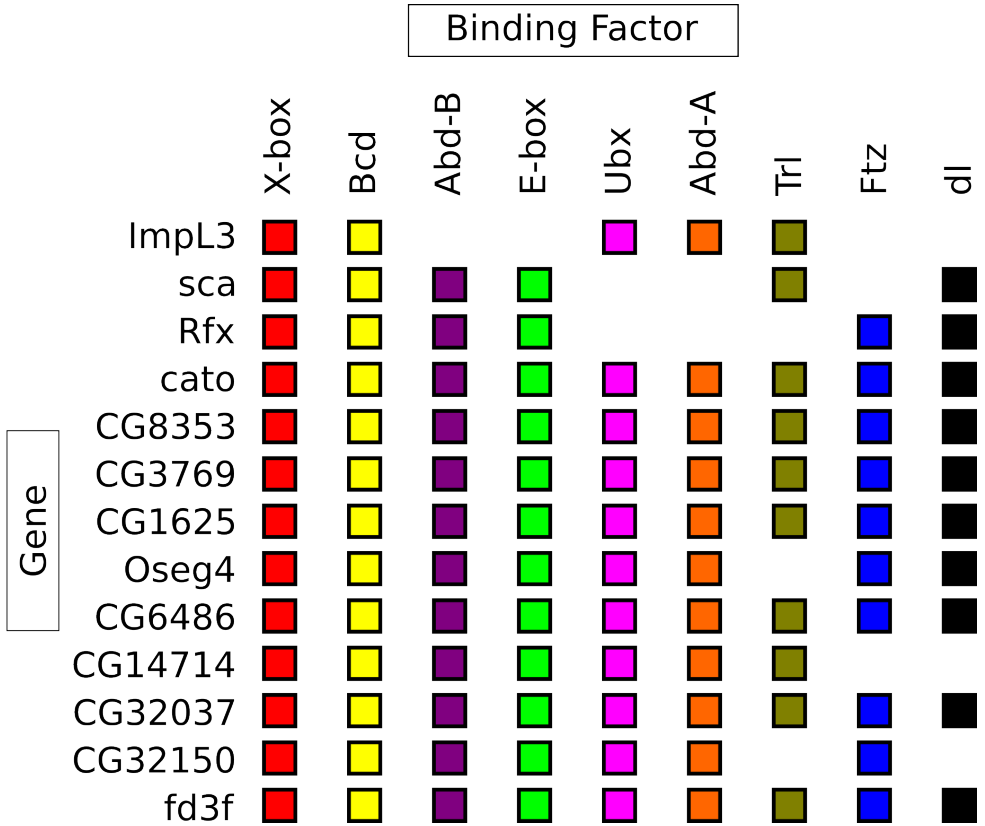
Rfx, the X-box and ciliated sensory neuron development

- Ciliated sensory neurons
  - Most sensory neurons have cilia at their dendritic tips
  - Cilia play crucial and highly conserved roles in motility, molecular transport and developmental processes such as left-right symmetry and sense organ development
  - Mutations in Rfx proteins are associated with defects in ciliogenesis in many organisms including Drosophila
- The X-box, comparative genetics and the ciliome
  - Rfx proteins bind to the X-box RYYNYYN[1-3]RRNRAC is bound by Rfx proteins
  - Genome screens for conserved X-boxes have recently been used to identify novel targets of Rfx proteins in Drosophila (Laurencon et al. Genome Biology(2007)**8**,R195)
  - Compared D.mel and D.pse common ancestor 40-60 mya
  - intron sequences 40% identical, known binding sites from the literature mapped on are 63% identical

cis-regulatory modules (CRMs) an entry point for network assembly



Sites >=75% identical between D.mel and D.Pse that for genes that also contain an X-box (13/27) from the sensory cilium biogenesis cluster.



- based on 75% conservation there are 7823 X-boxes in the fly genome (0.5/gene) so we expect 13 in list of 27
- sensory cluster has 50 conserved X-boxes an enrichment of x3.8

Gene feature finding summary

- heuristics, Gibbs samplers, dynamic programming, Markov chains and randomisation/bootstrap methods are commonly integrated into pipelines to study a series of connected processes from beginnning of analysis to the end

Gene feature finding summary

- heuristics, Gibbs samplers, dynamic programming, Markov chains and randomisation/bootstrap methods are commonly integrated into pipelines to study a series of connected processes from beginnning of analysis to the end
- here we have looked at a specific (and unsolved) instance of TFBS searching (and prediction)

Gene feature finding summary

- heuristics, Gibbs samplers, dynamic programming, Markov chains and randomisation/bootstrap methods are commonly integrated into pipelines to study a series of connected processes from beginnning of analysis to the end
- here we have looked at a specific (and unsolved) instance of TFBS searching (and prediction)
- these methods are rapidly evolving in all areas, most are useable on a standard workstation and most have programmatic access through BioJava, BioPerl and of course C