TGAC

The Genome Analysis Centre™

BBSRC

Greater Norwich Development Partnership

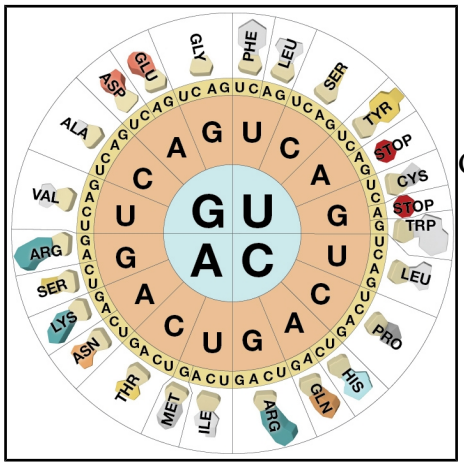
TGAC/EMBL-EBI Summer School in Bioinformatics for Biological Researchers

25-29 July 2016

Basic Bioinformatics Sessions

Practical 2: Pairwise Alignment

Monday 4 July 2016



Homo sapiens oculorhombin (PAX6) mRNA, complete cds, alternatively spliced

GenBank: M77844.1

Sensitive Pairwise Alignment

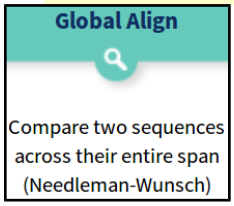
The purpose of this exercise is to look at some aspects of **Pairwise Sequence Alignment** using the most accurate methods available.

As hopefully has been discussed, sequences can be aligned using a **global** strategy, in which the two sequences being aligned are assumed to be homologous from end to end, or using a **local** approach, in which the sequences are assumed to just have homologous region(s).

Global Pairwise Sequence Comparison

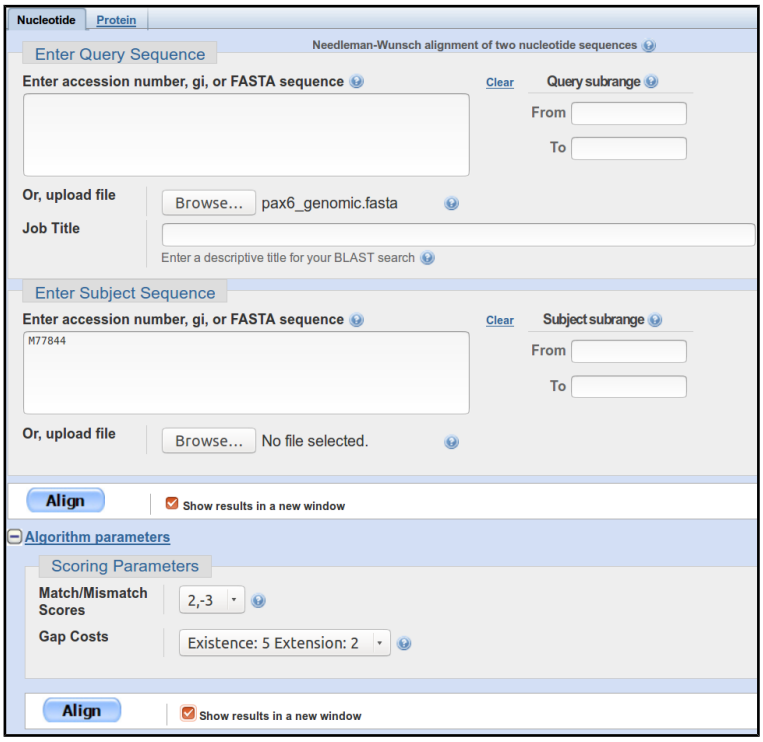
First the **global** approach. In a previous exercise, you already have used the **blast** facility at the **NCBI** to perform crude pairwise alignment. **blast** also offers a sensitive option, so maybe that would be a good place to start.

So, once more to the **NCBI** home page (**http://www.ncbi.nlm.nih.gov/**). From there chose **BLAST** from the **Popular Resources** list. Scroll down to the **Specialized searches** section and chose the



option.

A choice of settings for **Nucleotide** or **Protein** alignment is offered. As we are going to investigate the alignment of DNA sequences, the default choice is fine. For the first sequence, browse for the file **pax6_genomic.fasta**, which you created when looking at Ensembl. It contains the region of **Chromosome 11** containing the entire **PAX6** gene (with a few extra base pairs either end).

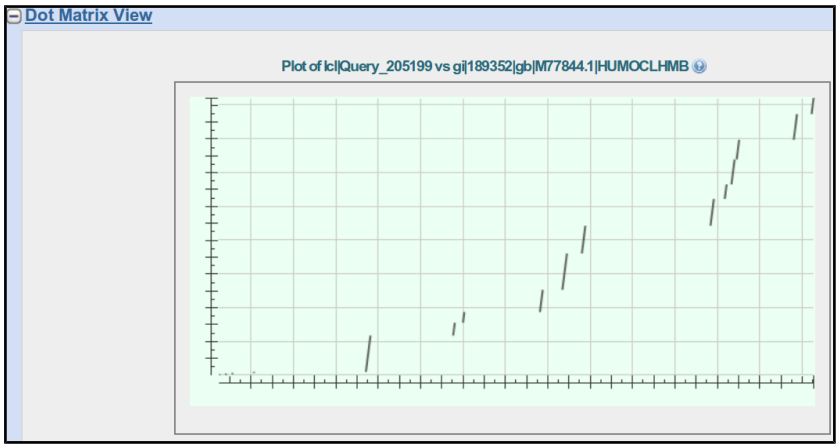


To specify the second sequence, you could load the file **pax6_mrna.fasta**, but just typing the corresponding **Accession** code in the appropriate box seems far more sophisticated, so that is what I chose to do.

Open the **Algorithmic Parameters** section, and see that they are as one might expect. The defaults are fine here as the alignment to be computed is trivial (given the way **blast** will go about the task), so anything not outrageous should work.

Ask to **Show results in a new window** and then click on the **Align** button.

After some significant Rollin' and Tumblin' **blast** will proclaim its lyrical conclusions. First examine the **Dot Matrix View**. This sort of representation has rather gone out of fashion in recent years. A shame, I say, this picture represents such a succinct summary of what should be expected of the textual aligment(s) that are the "real" detailed output of this sort of program.



How would you interpret this picture?

What do the diagonal(ish) lines represent?

What are the gaps in between the lines?

Which axis represents the genomic sequence and which the mrna?

Pairwise Sequence Alignment (NUCLEOTIDE)

EMBOSS Needle reads two input sequences and writes their optimal global sequence alignment to file.

This is the form for nucleotide sequences. Please go to the protein form if you wish to align protein sequences.

STEP 1 - Enter your nucleotide sequences

Enter or paste your first **nucleotide** sequence in any supported format:

Or, upload a file: Browse... pax6_genomic.fasta

AND

Enter or paste your second **nucleotide** sequence in any supported format:

Or, upload a file: Browse... pax6_mrna.fasta

STEP 2 - Set your pairwise alignment options

MATRIX	GAP OPEN	GAP EXTEND	OUTPUT FORMAT
DNAfull	10	0.5	pair

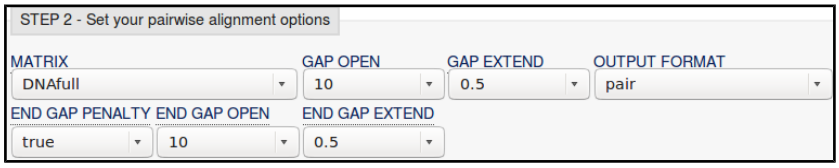
END GAP PENALTY	END GAP OPEN	END GAP EXTEND
false	10	0.5

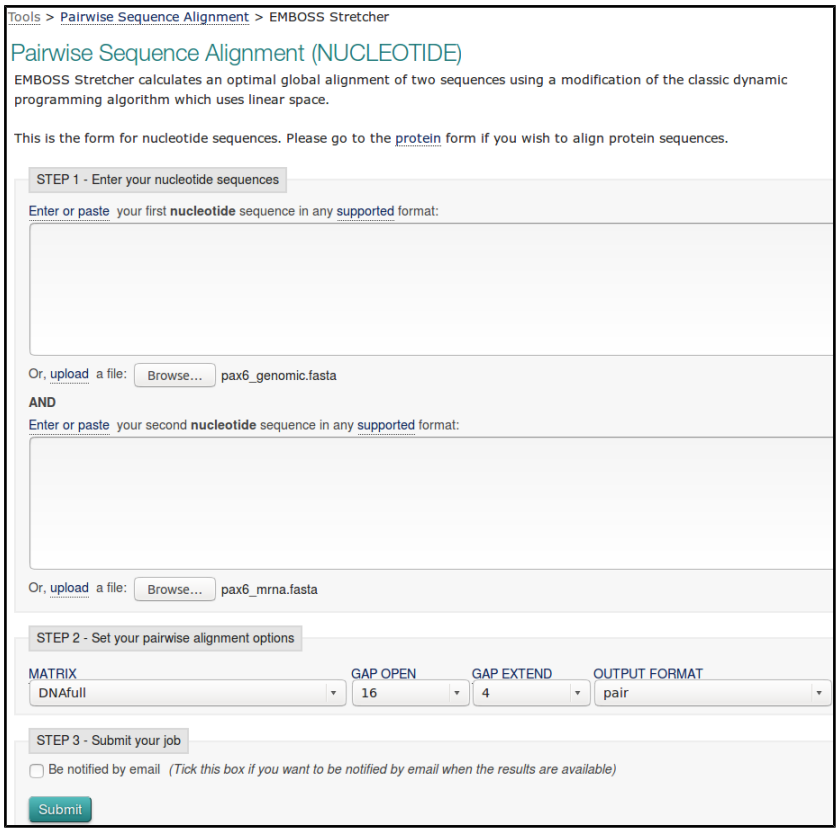
STEP 3 - Submit your job

Be notified by email *(Tick this box if you want to be notified by email when the results are available)*

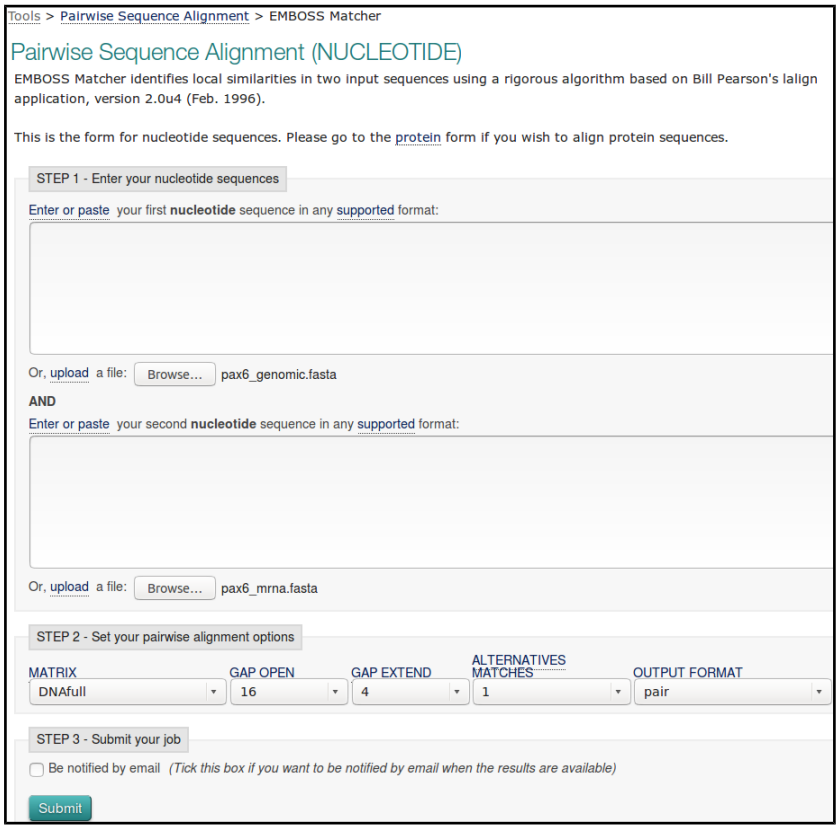
Submit

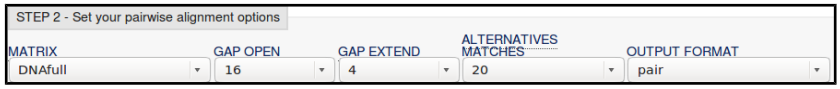
```
11              22101 TCTTGTAACACAATGTGGCCCGCTGCACGCCTCAAGAGAATC------CT  22144
                                                                .|||      .|
HUMOCLHMB           1 ------------------------------------------TATCGATAAGTT     12

11              22145 TTTGTTGTCCGCGCTCATTGTAGCCTCAAAAT-TCTGCCCACGAAAGTTT  22193
                      |||.||        |.|||||       .||| ||||
HUMOCLHMB          13 TTTTTT--------TTATTGT-------CAATCTCTG-------------     34

11              22194 GCCAACGCTCCTGCCCCAGGAGTTTAATAGTTTCCCTTACTCGCGGGGCA  22243
                      .|||||.|||||||.|.|.|...||.|.|||||.|.|
HUMOCLHMB          35 ------TCTCCT-TCCCAGGAATCTGAGGATTGCTCTTACACAC------     71

11              22244 TTGTGCAGCGCTGAAAAGCAGCCCCTCGCTATTCAAGTGTTGGTGGTCA-  22292
                      ||.|.|.|.||  ||.||                ||||..|
HUMOCLHMB          72 -----CAACCCAGCAA--CATCC-------------------GTGGAGAA     95

11              22293 ---TCTCAATAG-ATCTCCAAGGGCCCATATGGTGGCCAGTGCCGATGAA  22338
                         |||||..|| |.||||
HUMOCLHMB          96 AACTCTCACCAGCAACTCC-----------------------------     114

11              22339 TCCGCCTGTTTAAATGGGGGAGAAAGTTGGGGTTTTAAAACAT-------  22381
                            ||||||      |.|..||   ..|||.|||.|||
HUMOCLHMB         115 --------TTTAAA------ACACCGT---CATTTCAAACCATTGTGGTC    147

11              22382 TTCAA--------AGTTCCTGAAAAGATCCC-----------ACT-----  22407
                      |||||        ||...|..||||.|.|||           |||
HUMOCLHMB         148 TTCAAGCAACAACAGCAGCACAAAAAAACCCCAACCAAACAAAACTCTTGA    197

11              23546 TACCTTGGGAATGTTTTGGTGA---GGCTGTCGGGATATAATGCTCTTGG  23592
                            ||||     |||   |||.|.|.||     |.||   |||
HUMOCLHMB         804 ----------ATGT-----TGAACGGGCAGACCGG-----AAGC---TGG    830

11              23593 AGTTTAAGACTACACCAGGCCCCT-TTTGGAGGCTCCAAGTTAATCC--A  23639
                      .|          ||||.|  |||| .|||           .||||  .
HUMOCLHMB         831 GG----------CACCCG--CCCTGGTTGG-----------TATCCGGG    856

11              23640 AATTTCTCTTAC----CATCCTATTCTTTTTGTTCCAGATGGCTGCCAGC  23685
                      .|.|||..|..|    ||.||||          ..|.|||||||||||||
HUMOCLHMB         857 GACTTCGGTGCCAGGGCAACCTA---------CGCAAGATGGCTGCCAGC    897

11              23686 AACAGGAAGGAGGGGGGAGAGAATACCAACTCCATCAGTTCCAACGGAGAA  23735
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         898 AACAGGAAGGAGGGGGGAGAGAATACCAACTCCATCAGTTCCAACGGAGAA    947

11              23736 GATTCAGATGAGGCTCAAATGCGACTTCAGCTGAAGCGGAAGCTGCAAAG  23785
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         948 GATTCAGATGAGGCTCAAATGCGACTTCAGCTGAAGCGGAAGCTGCAAAG    997

11              23786 AAATAGAACATCCTTTACCCAAGAGCAAATTGAGGCCCTGGAGAAAGGTG  23835
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         998 AAATAGAACATCCTTTACCCAAGAGCAAATTGAGGCCCTGGAGAA-----   1042

11              23836 ATAGAGTTTTTCAAAGTAGAGAAGCAGTAAATCAAAGTAAATGCCACATC  23885

HUMOCLHMB        1043 ------------------------------------------------     1042

11              24636 TAACCTGTCCCACCTGATTTCCAGGTATGGTTTTCTAATCGAAGGGCCAA  24685
                                      |||||||||||||||||||||||||||||||
HUMOCLHMB        1125 ---------------------CAGGTATGGTTTTCTAATCGAAGGGCCAA   1153

11              24686 ATGGAGAAGAGAAGAAAAACTGAGGAATCAGAGAAGACAGGCCAGCAACA  24735
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB        1154 ATGGAGAAGAGAAGAAAAACTGAGGAATCAGAGAAGACAGGCCAGCAACA   1203

11              24736 CACCTAGTCATATTCCTATCAGCAGTAGTTTCAGCACCAGTGTCTACCAA  24785
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB        1204 CACCTAGTCATATTCCTATCAGCAGTAGTTTCAGCACCAGTGTCTACCAA   1253

11              24786 CCAATTCCACAACCCACCACACCGGGTAATTTGAAATACTAATACTACGA  24835
                      |||||||||||||||||||||||||
HUMOCLHMB        1254 CCAATTCCACAACCCACCACACCG-------------------------   1277

11              24836 ATCAATGTCTTTAAACCTGTTTGCTCCGGGCTCTGACTCTCACTCTGACT  24885

HUMOCLHMB        1278 ------------------------------------------------     1277

11              24886 ACTGTCATTTCTCTTGCCCTCAGTTTCCTCCTTCACATCTGGCTCCATGT  24935
                                      |||||||||||||||||||||||||||||||
HUMOCLHMB        1278 ----------------------GTTTCCTCCTTCACATCTGGCTCCATGT   1305

11              24936 TGGGCCGAACAGACACAGCCCTCACAAACACCTACAGCGCTCTGCCGCCT  24985
                      ||||||-|||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB        1306 TGGGCCTAACAGACACAGCCCTCACAAACACCTACAGCGCTCTGCCGCCT   1355

11              24986 ATGCCCAGCTTCACCATGGCAAATAACCTGCCTATGCAAGTAAGTGCGGC  25035
                      ||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB        1356 ATGCCCAGCTTCACCATGGCAAATAACCTGCCTATGCAA-----------   1394

11              25036 TGGTGGTGGCCTGCATAACCCAGGCCCCAG--AGAAGTGAGGAGTGGCTC  25083
                            ||       |||||.||||||  |||
HUMOCLHMB        1395 ---------CC-------CCCAGTCCCCAGCCAGA---------------   1413

11              25084 AGGGCCTGCGGACCTCAT-----TGGCTGTGTCTG--CACCCTTGAGAGC  25126
                            |||     ||||||     ||..||   |||  ||||      |||
HUMOCLHMB        1414 ----CCT-----CCTCATACTCCTGCATG---CTGCCCACC------AGC   1445
```


STEP 2 - Set your pairwise alignment options
MATRIX
DNAfull
GAP OPEN
10
GAP EXTEND
0.5
OUTPUT FORMAT
pair
END GAP PENALTY
true
END GAP OPEN
10
END GAP EXTEND
0.5


Tools > Pairwise Sequence Alignment > EMBOSS Stretcher
Pairwise Sequence Alignment (NUCLEOTIDE)
EMBOSS Stretcher calculates an optimal global alignment of two sequences using a modification of the classic dynamic programming algorithm which uses linear space.
This is the form for nucleotide sequences. Please go to the protein form if you wish to align protein sequences.
STEP 1 - Enter your nucleotide sequences
Enter or paste your first nucleotide sequence in any supported format:
Or, upload a file: Browse... pax6_genomic.fasta
AND
Enter or paste your second nucleotide sequence in any supported format:
Or, upload a file: Browse... pax6_mrna.fasta
STEP 2 - Set your pairwise alignment options
MATRIX
DNAfull
GAP OPEN
16
GAP EXTEND
4
OUTPUT FORMAT
pair
STEP 3 - Submit your job
Be notified by email (Tick this box if you want to be notified by email when the results are available)
Submit

```
11              16670 CACTTCCCCTAT---GCAGGTGTCCAACGGATGTGTGAGTAAAATTCTGG  16716
                      ||.|||||..||   ||||||||||||||||||||||||||||||||||
HUMOCLHMB         485 CATTTCCCGAATTCTGCAGGTGTCCAACGGATGTGTGAGTAAAATTCTGG    534

11              16717 GCAGGTATTACGAGACTGGCTCCATCAGACCCAGGGCAATCGGTGGTAGT  16766
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         535 GCAGGTATTACGAGACTGGCTCCATCAGACCCAGGGCAATCGGTGGTAGT    584

11              16767 AAACCGAGAGTAGCGACTCCAGAAGTTGTAAGCAAAATAGCCCAGTATAA  16816
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         585 AAACCGAGAGTAGCGACTCCAGAAGTTGTAAGCAAAATAGCCCAGTATAA    634

11              16817 GCGGGAGTGCCCGTCCATCTTTGCTTGGGAAATCCGAGACAGATTACTGT  16866
                      ||||||||||||||||||||||||||||||||||||||||||||||||||||
HUMOCLHMB         635 GCGGGAGTGCCCGTCCATCTTTGCTTGGGAAATCCGAGACAGATTACTGT    684

11              16867 CCGAGGGGGTCTGTACCAACGATAACATACCAAGCGTAAGTTCATTGAGA  16916
                      ||||||||||||||||||||||||||||||||||||....|||.|.|..
HUMOCLHMB         685 CCGAGGGGGTCTGTACCAACGATAACATACCAAGCGTGTCATCAATAAAC    734

11              16917 ACA--TCTGCCCTCCCTGCC  16934
                      |.|  |||-|.|..||||.|
HUMOCLHMB         735 AGAGTTCTTCGCAACCTGGC    754
```






```
########################################
#
# Aligned_sequences: 2
# 1: 11
# 2: HUMOCLHMB
# Matrix: EDNAFULL
# Gap_penalty: 16
# Extend_penalty: 4
#
# Length: 58
# Identity:      39/58 (67.2%)
# Similarity:    39/58 (67.2%)
# Gaps:           6/58 (10.3%)
# Score: 83
#
#
########################################

11              2353 GCTGGACGCCACCCGGCGCCAGA--GCCGGGC---CTGAGGAGCGGGGTC   2397
                     ||.||||.|||||||||...|||  |..|.||   ||.|. |||||||.|
HUMOCLHMB        425 GCCGGACTCCACCCGGCAGAAGATTGTAGAGCTAGCTCAC-AGCGGGGCC    473

11              2398 TGGCCGGG   2405
                     .|||||.|
HUMOCLHMB        474 CGGCCGTG    481


########################################
#
# Aligned_sequences: 2
# 1: 11
# 2: HUMOCLHMB
# Matrix: EDNAFULL
# Gap_penalty: 16
# Extend_penalty: 4
#
# Length: 46
# Identity:      31/46 (67.4%)
# Similarity:    31/46 (67.4%)
# Gaps:           1/46 ( 2.2%)
# Score: 83
#
#
########################################

11              11417 ACAGTTTGACTGAGCCCTAGATGCATGTGTTTTT-CCTGAGAGTGA  11461
                      |.|||||..||.|||...||.||..|||.||| ||.|||||.||
HUMOCLHMB        1043 AGAGTTTGAGAGAACCCATTATCCAGATGTGTTTGCCCGAGAAAGA   1088
```

Sequence Analysis

The overall target remains to discover all we can about **Aniridia** and its genetic causes. Here we will see how information can be derived from analysis of sequence data directly. Many of the analysis tools and methods we will use and discuss will be those that were used to generate the "ready made" answers you have already investigated. The conclusions of this section should not therefore contradict those you have already reached.

To start the analysis section, you should have data files including:

Two PCR primers associated with the **PAX6** gene in a way we have not yet fully determined, in a file called:

pax6_primers.fasta

The sequence of the Paired box domain of the most prolific human **PAX6** isoform in a file called:

pax_domain.fasta

The genomic sequence of the whole of the **PAX6** gene, plus an extra 500 bases in each direction, in a file called

pax6_genomic.fasta

The canonical human **PAX6** isoform in a file called:

pax6_human.fasta

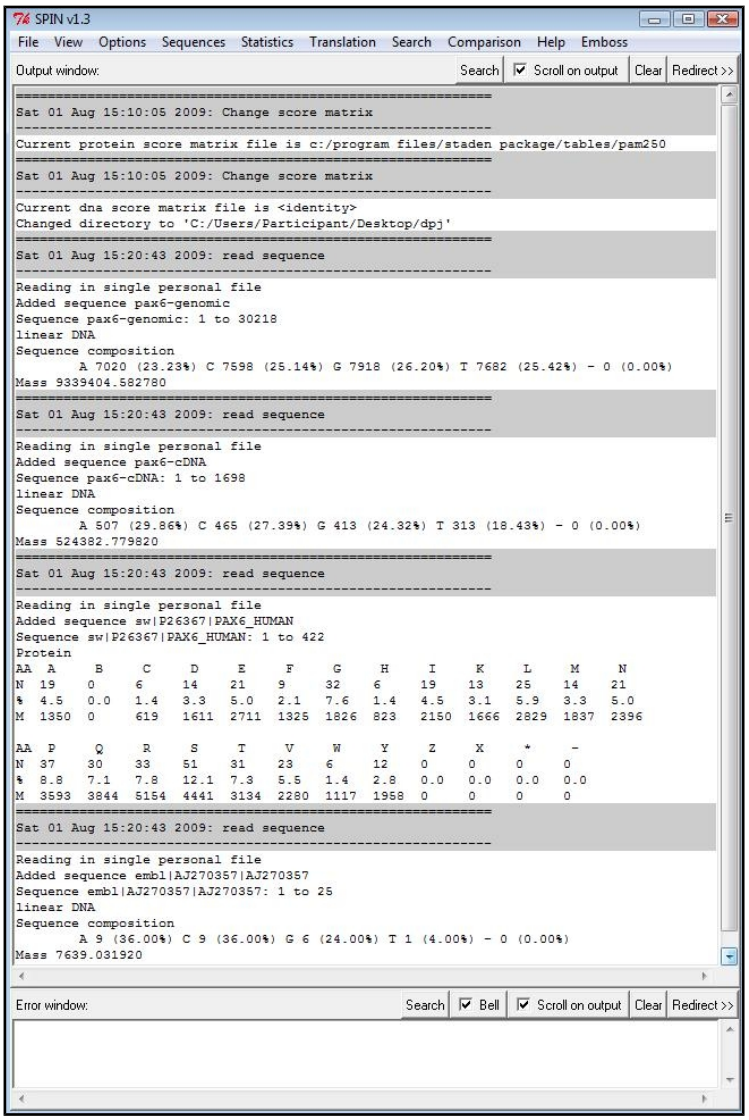
The file **pax6_genomic.fasta** contains the sequence of **PAX6** from a person(s) who, it can be assumed, does not suffer from **Aniridia**. Given a sequence from a person who **does** suffer from **Aniridia**, an obvious first step of investigation might be to compare the sequence from the affected source with the wild type. It would be reasonable to speculate that differences might explain the disease. Here we have to cheat a bit. I have provided an mRNA sequence we will suppose you sequenced from an **Aniridia** patient. It is stored in a file called:

pax6_cdna.fasta

which you will find in a directory called **Working Directory** on your **Desktop**.

Internet resources will be employed for many analyses. Software that runs under windows on your workstation will be used very occasionally. All windows software has been installed, without **Administrator** privileges, using an Installer that will be made available to you.

The **EMBOSS** package[1], using a program called **spin**[2] to provide a **G**raphical **U**ser **I**nterface (**GUI**) will be used for most of the workstation analyses. This is not ideal, particularly as support for **spin** is defunct, but the options[3] are few.



Start **Spin** in the usual windows way (it is part of the **The Staden Package**, which is in the **Start menu** under **Sequence Analysis Tools Installer**). Proceed by setting **spin** to manage its files somewhere sensible. Strangely, its default inclination is to create user output files in its installation directory? This is silly, so select **Change directory** from the **File** pull down menu and select a more sensible default directory. I suggest you might make a directory in your desktop named as you please for this purpose[4].

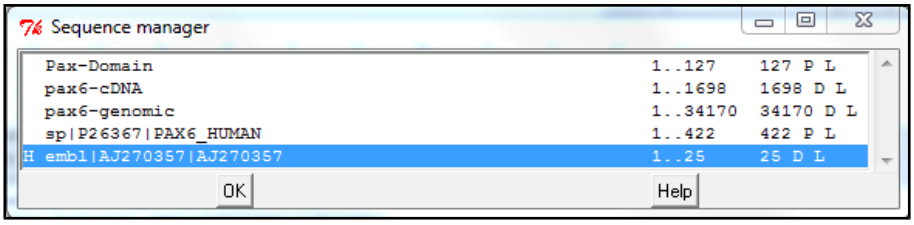
1 Documentation pages for **EMBOSS** can be accessed at the **EMBOSS** home page at: **http://emboss.sourceforge.net**

2 A part of the **Staden Package**.

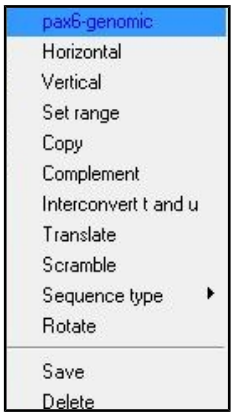
3 That is **FREE** options!

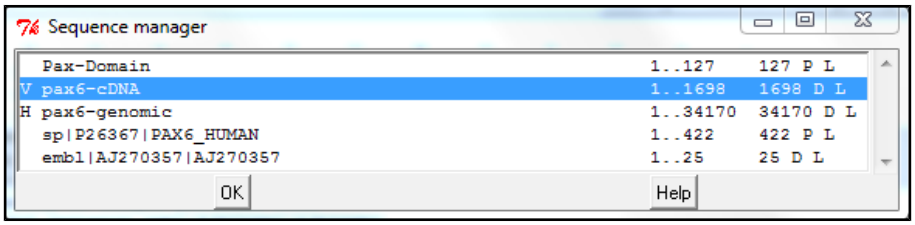
4 It would be very sensible to move **pax6_cdna.fasta** and all the files you have created thus far (**pax6_genomic.fasta**, **pax6_primers.fasta**, **pax_domain.fasta** and **pax6_human.fasta** in particular) into this directory.

Begin by loading all the sequence files you have gathered thus far. From the **File** drop down menu select **Load Sequence** and then **Simple**. This will present a new dialogue box for you to specify a sequence to be loaded. Leave the choice of **Database** as the default **Personal File**, as the all the sequences needed initially are stored in local files. To the right a text box awaits the file path for a sequence file. Use the **Browse** button to find and enter the directory you set up as your default[5]. Select all the sequence files listed above[6] and click **Open** and then **OK** in the **Load Sequence** window. From this point onwards the package will refer to these sequences by their **fasta** titles (e.g. **pax6-cDNA** and **pax6-genomic)**[7].



Take some time to become acquainted with the **spin Sequence manager**. This provides access to all loaded sequences. Its main purpose is to enable sequences to be selected for analysis. From the **File** drop down menu, select **Sequence Manager**. A window will appear listing the sequences showing their start and end, length and type (**D** for DNA or **P** for Protein). The last sequence to be loaded will be labelled **H** (for **H**orizontal)[8]. The sequence set to **H**orizontal is the one that **spin** will analyse by default. Ensure that **pax6-genomic** is set to **H**orizontal by right clicking its name and selecting **Horizontal**. Right clicking an entry brings forth a menu offering several more possibilities than we will use in these exercises. For example, it is possible to **Set range**, that is specify regions of entries to be analysed separately. Also it is possible to **Translate** or **Compliment** DNA sequences to create new entities for analysis.





In similar fashion, make **pax6-cDNA** the **V** (for **V**ertical) sequence[9].

Leave the **Sequence manager** by clicking **OK**. You are now be ready to proceed.

5 If you did things correctly, you should be in the right place without having to move anywhere.

6 You can select multiple files in a number of ways including holding the **Ctrl** key down whilst clicking on each of the files you require.

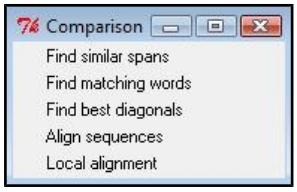
7 **spin** will describe all the sequences it has loaded. Note that only the first of the primer sequences was loaded, sadly **spin** does not recognise that **pax6_primers.fasta** is a multiple sequence file. This will not matter to us as the exercises stand at present.

8 This refers, somewhat bizarrely, to the placement of the sequence on Cartesian axes.

9 This only really makes complete sense when you are computing dotplots, which you are just about to do. **spin** is limited in that it allows a maximum of 2 sequences to be selected. When only one sequence is required, the **H**orizontal selection is used.

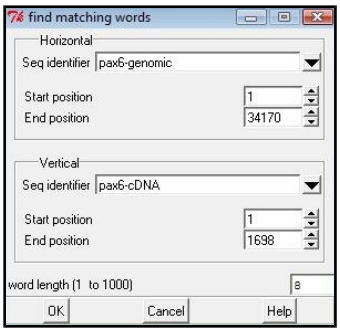
Pairwise Sequence Comparison

You will be using options from **spin**'s **Comparison** menu for the next few steps. To set up that menu for easy access, click on **spin**'s **Comparison** button and then on the dotted line of the menu that will appear. Position the menu window that "tears off" conveniently.

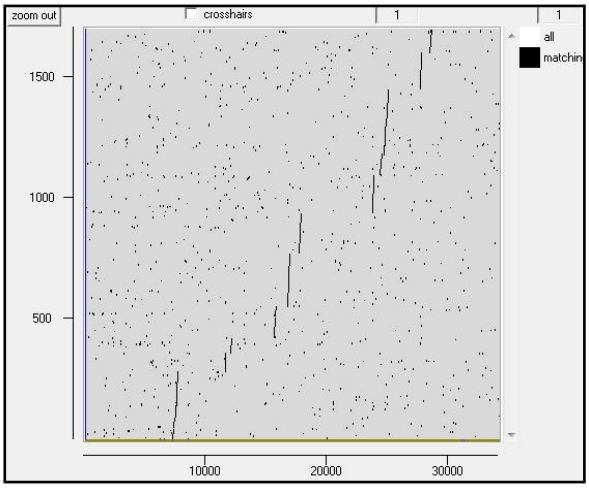


Graphical pairwise sequence comparison - DotPlots

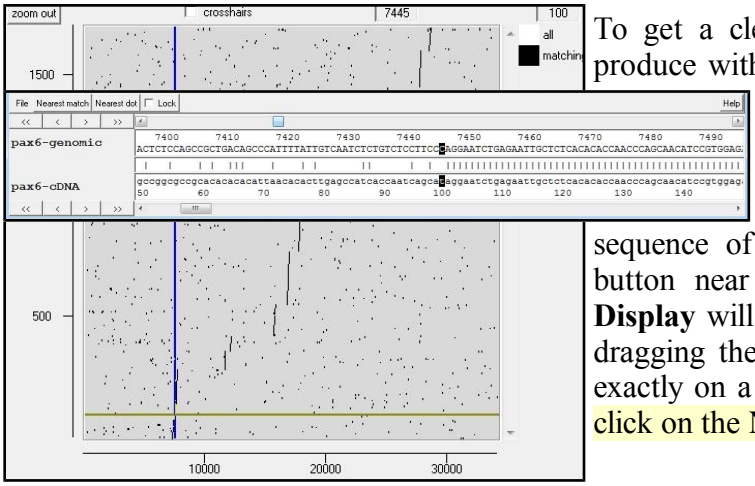
An intuitive graphical representation of the comparison between two sequences is a **dotplot**. One sequence is represented on each of two Cartesian axes and significant matching regions are indicated as a plot. Further detail will be provided by presentation. These plots are used to indicate which sections of sequence might align with each other convincingly and thus warrant further investigation.



From your **Comparison** menu, select **Find matching words**, which offers a simple and fast way to draw a dot plot. Check that you have the sequences you expect in the **Horizontal** and **Vertical** positions[10]. Accept the default **word length** of **8**[11] and click **OK**. A **dotplot** will burst forth with the genomic sequence along the bottom axis, and the cDNA sequence on the vertical axis, as illustrated. As far as the resolution of the picture allows one to judge, there appears to be **11** matching regions.

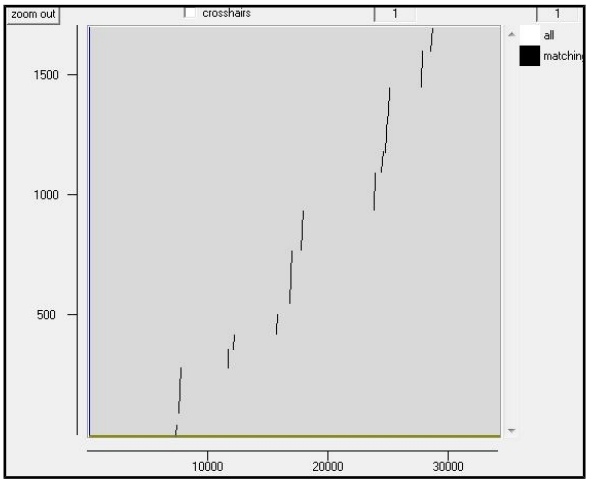


What do you suppose these regions represent?_____



To get a clearer view of the features of any of the graphics you produce with **spin**, you can always make the picture window bigger, use the **X** and **Y** controls at the bottom to reshape the view, hold the **Ctrl** key down and select a region with your right mouse button to **zoom in** to a feature[12]. Try any or all of these possibilities. You can also view the sequence of any feature by double clicking your left hand mouse button near that feature. Try this and a **Sequence Comparison Display** will appear. You can control the sequence region in view by dragging the cross hairs that will appear around. To fix the display exactly on a given feature, get as near as you can manually and then click on the **Nearest match** button.

Using a **word length** as small as **8** shows the meaningfully matching regions quite well in this case, but there is quite a bit of unnecessary "noise"[13]. The features we wish to be shown are long, so it is possible to use a bigger word size without losing matches of importance.

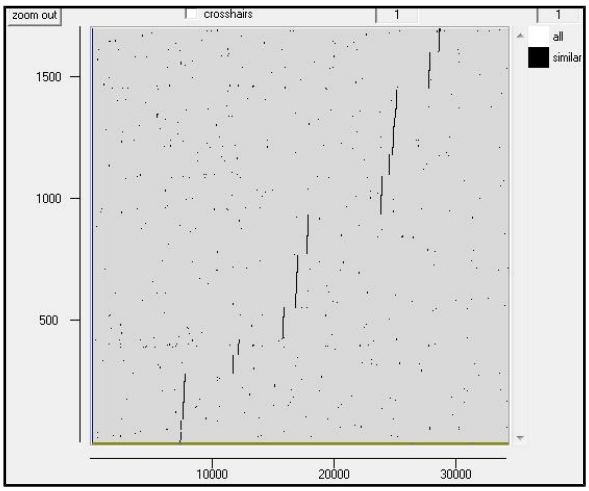


Re-run **Find matching words**, exactly as before but this time, select a **word length** of **50**. Your new dotplot will be drawn on top of you first and so will thus be difficult to see. A plot can be repositioned by moving around its configuration button (the corresponding square in the top right hand corner) with the middle mouse button. You could move the new dotplot so it is above or below the first or into a separate window (the best choice) by dragging its configuration button into an unoccupied place on your desktop. Your second plot is essentially as the first, without the noisy background. Now it looks more like twelve matching regions. We compare genomic sequence with cDNA, so it is reasonable to assume the matching regions of this plot represent the exons found in a transcript of **PAX6**.

10 If you need to, use the pull down menus provided to select from any of the sequences in your **Sequence manager**.
11 Thus asking the program to indicate the presence of exactly matching regions of **8** base pairs between the chosen sequences.
12 There is a **zoom out** button at the top of the display to undo any **zooming in** you try.
13 I.e. dots that do not represent anything interesting.

Is the number of matching regions consistent with the number of exons you might expect?_____

If not, can you explain the discrepancy?_____

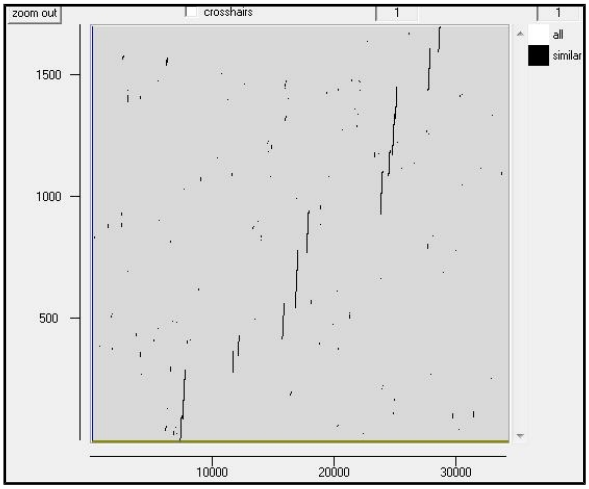


The second plot is clearer. Noise has been eliminated without loss of "real" features. The choice is not always so clear. It can be necessary to experiment using different parameters.

The algorithm you have used thus far is fast but crude. Adequate for these sequences between which all real matches are long and almost exact. However, in most circumstances a more meticulous slower[14] approach might be required in which "similar" as well as exactly matching words are recognised. From your **Comparison** menu, select **Find similar spans**, accepting the default **window length** of **11** and **minimum score** of **10**[15], click **OK**[16]. Your plot should be as illustrated.

Again, from your **Comparison** menu, select **Find similar spans**. This time, set the **window length** to **50**. **spin** automatically resets its **minimum score** field to a new default of **28**. Click **OK**. Separate out your plots so that they can be seen clearly. No surprises? A less noisy background is the most obvious effect. Also you should be able to see that the "real" features appear to have grown a little.

Why do you suppose that might be?_____



The dotplot can be useful as a rapid overview of the similarity between two sequences. Alone, it provides insufficient detail. For this, we need to generate textual alignments.

spin produces helpful graphics when textually aligning sequences. These graphics are most informative when superimposed over a corresponding dotplot. Here, the most useful dotplot is the simplest. The one you generated using the "**Find matching words**" algorithm with a word length of **50**. If you have not already done so, throw all the others away. If necessary, recreate the required dotplot.

Pairwise textual sequence alignment

Now make some alignments that will show the detail of the dotplot comparisons. The algorithms used are more rigorous than those used for searching databases, often producing more revealing alignments. Thus it can be a good idea to use these tools on similar sequences identified by database similarity searches

Global sequence alignment

A global alignment is one that aligns two sequences over their entire lengths. In **spin**'s **Comparison** menu, the option **Align sequences**[17]. Click on **spin**'s **Align sequences** option. Click **OK** accepting the defaults[18]:

score for match	**(4)**	value added to the alignment score for each correctly matched base
score for mis-match	**(-2)**	value added to the alignment score for each incorrectly matched base
penalty for starting gap	**(8)**	value subtracted from the alignment score for each gap
penalty for each residue in gap	**(1)**	value subtracted from the alignment score for each element of a gap beyond the first

spin represents the computed alignment graphically, over the dotplot. It should suggest that it has successfully aligned the group of exons around **23,500** to **25,000** in the genomic sequence and **940** to **1,450** in the mRNA[19], but then failed with the rest.

14 Not that the speed difference will be apparent with sequences of the size we are comparing here.

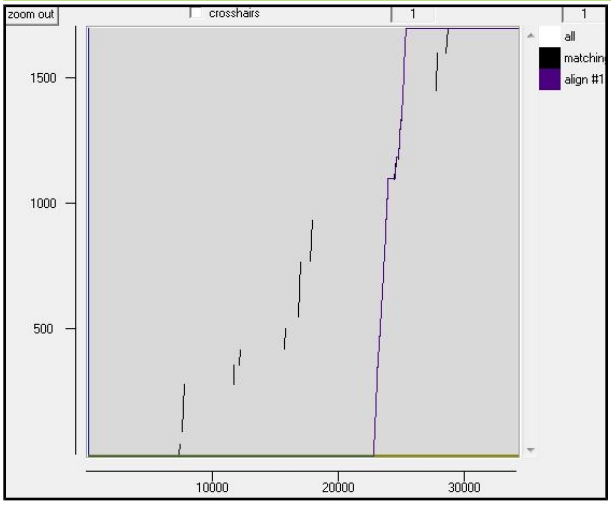
15 Thus matched windows of length **11** in which **10** of the aligned bases are identical are regarded as significant. By default, **spin** awards **1** point to matching bases and **0** points to mismatched bases. The equivalent program in the **EMBOSS** package, **dotmatcher**, awards matching bases **+5** and penalises mismatched bases **-4**. The full **EMBOSS** default DNA comparison scoring matrix can be found in an appendix.

16 Depending how you left you displays, you might need to separate out your dotplots in order to see them clearly.

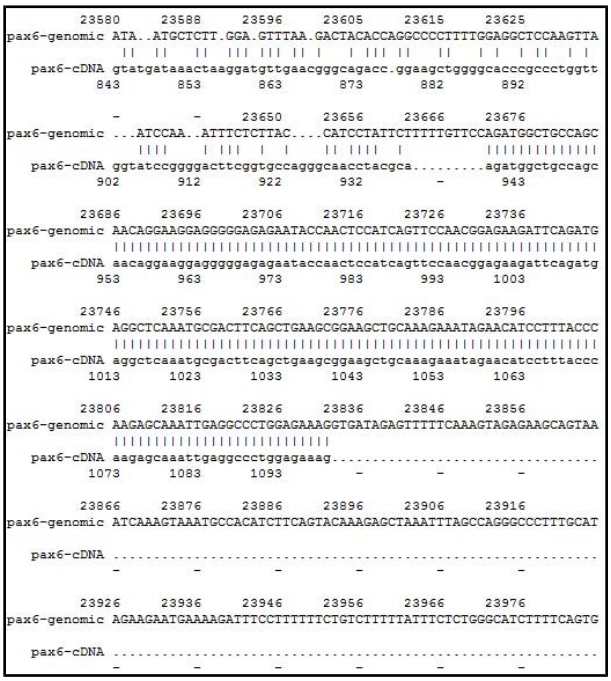
17 The **EMBOSS** package offers the program **needle**, which is a rigorous implementation of the Needleman-Wunsch algorithm for global alignment. **EMBOSS** also offers a less rigorous (i.e. faster and sloppier) program called **stretcher**.

18 A complete description of these parameters will be included in a presentation. Soon, if it has not already occurred.

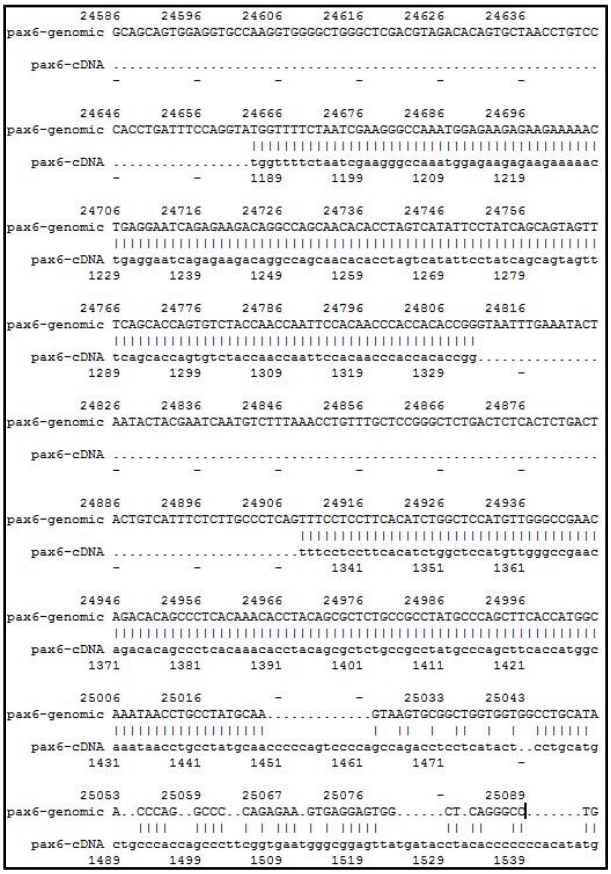
19 Click in the **crosshair** box and use the crosshair to get a good idea of where features are in your plots. I think the cross-hair is very irritating to leave active, so I suggest you click it off again once its purpose is fulfilled.



Examine how your alignment graphic matches the dotplot by right clicking its configuration button and using the **Hide**/**Reveal** option.



Now inspect the textual alignment in **spin**'s **Output window**. As the graphic suggests, the mRNA has been prefixed with over **22,000** minus signs to make it of equal length to the genomic sequence. There follows an entirely unconvincing (given the evidence of the dotplots) alignment of the first few exons of the mRNA with the region around **23,000** in the genomic sequence.



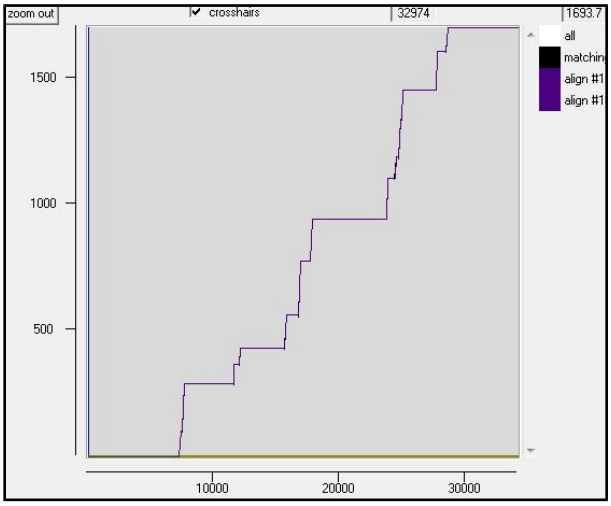
The **4** exons around **23,500-25,000** in the genomic sequence and **940-1,500** in the mRNA are aligned convincingly. The final mRNA exons are aligned, with an excess of poetry, with the bases after **25,000** of the genomic sequence.

The mRNA is then desperately padded to reach the end of the alignment. The bizarre padding at each end of the mRNA is required to fulfil the requirement of a global alignment to be “end to end”. The entirety of both sequences must be included in the alignment.

How many convincingly aligned regions did you see?_____

Roughly how many did you expect?_____

Clearly, this alignment is not correct. Can you explain why?_____



Try the alignment again, this time with much "cheaper" gaps. First **Hide** the plot associated with your first global alignment attempt. This should leave the dotplot in clear view. Now click on **spin**'s **Align sequences** option once again. This time, set the **penalty for each residue in gap** to **0**. This amounts to asking that all gaps be penalised exactly **8** points no matter how long they be. Click **OK**.

Now the alignment graphic shows the alignment dutifully passing though all the matching regions identified by the dotplot. Take a look at the textual output associated with this second alignment.

How many matching regions are there this time?_____

Is the count **now** roughly as you would expect?_____

These global alignments show how misleading it can be to run programs without carefully considering their assumptions and parameter values. The second time around, you achieved the "correct" alignment, but only by making the cost of gaps so cheap that huge introns could be mistaken for normal insertion/deletion events. This only worked because the mRNA/genomic sequences came from the same organism and the corresponding exons were effectively identical[20]. If the comparison was between homologous sequences from different organisms[21], it would usually be necessary to use gap penalties that reflected the way real insertion/deletion events occurred in the exons. Declaring almost free gaps to cope with introns would rarely succeed[22].

The best solution is to accept that the alignment between a cDNA/mRNA and genomic sequence is a special case. A general alignment program will not make the right assumptions for such alignments and is thus the wrong tool for the job. There is a program in the **EMBOSS** package, called **est2genome**, which is specifically designed for the alignment of cDNA/mRNA and genomic sequences. **est2genome** (and similar programs) may assume much more about the sequences to be aligned than can a general purpose alignment program. Gaps representing introns can be placed far more accurately if they are **known** to represent introns. Programs such as **est2genome** seek the highly conserved bases that occur at intron/exon boundaries, **C**/**T** rich intronic regions, **polyA** regions and **Stop**/**Start** codons to assist its detection of exons and gene structures.

est2genome is a fine program, but there are two other options offered at the **NCBI** in America that do the same job, I think, somewhat more nicely. Of these, I choose the program called **splign** for this exercise on the grounds it is the more sophisticated service[23]. To investigate, go to the home of **splign** at:

http://www.ncbi.nlm.nih.gov/sutils/splign

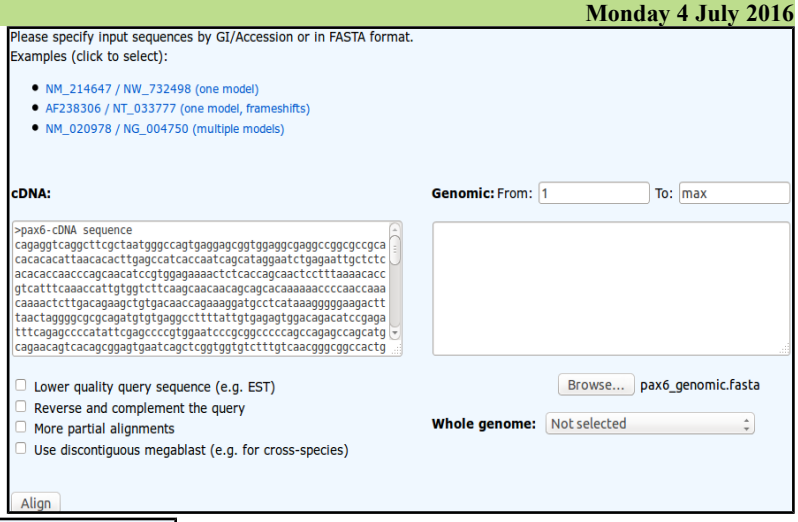
20 and so would align correctly given almost any gap penalties, or other parameter settings that might have been chosen.

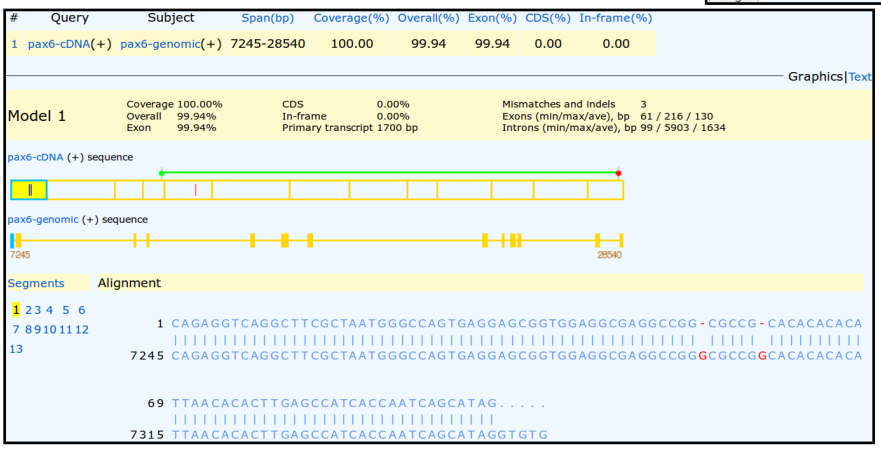
21 Comparing mRNA from one organism with genomic sequence from another is one way of investigating gene structure in newly sequenced genomic sequence.

22 One very simple solution to this problem offered by the **GCG** (now defunct) package version of **needle** (a program called **gap**) was to offer a gap penalty ceiling. The program would compute a gap penalty in the normal way until a given "cut off" was reached. **gap** would then assume it was trying to stretch over an intron rather than allow for an insertion/deletion, so it would allow the gap to increase at will without further raising the penalty. This worked much better than my intuition suggested it should!

23 The other is called **spidey**. The **spidey** home page is:
http://www.ncbi.nlm.nih.gov/IEB/Research/Ostell/Spidey/
It is pretty straight forward to use, just follow your nose. **est2genome**, **splign** and **spidey** should all give the same answer for this simple alignment and very similar answers for all such problems.

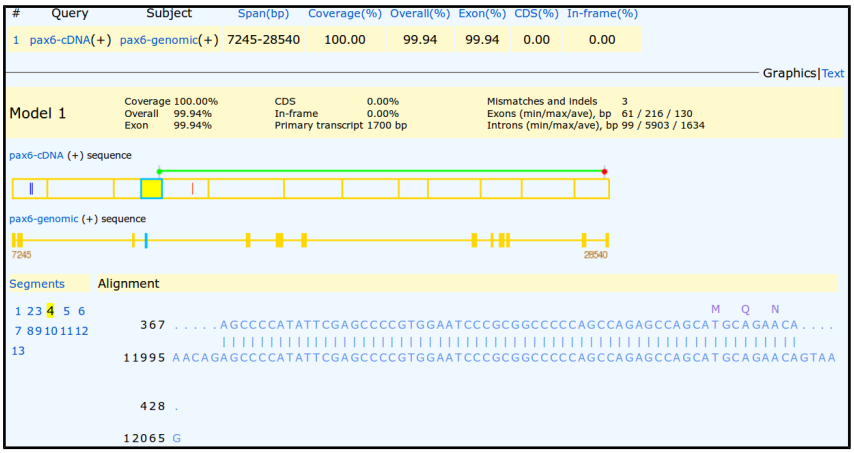
Click on the **Online** button. In the **Genomic** section, **Browse** to upload **pax6_genomic.fasta**. In the **cDNA** section, paste the sequence **pax6_cdna.fasta**. Where **cDNA** and **Genomic** sequences share exons that are nearly identical, **splign** uses the comparison algorithm **megablast** (default choice). Where exons are less similar (e.g. when the **cDNA** and **Genomic** sequences are from different organisms) the more sensitive option **discontinuous megablast**, might be a better choice[24]. Note the option to compare your **cDNA** with a **Whole genome** (including Human). Today, the default options are fine. Click the **Align** button.

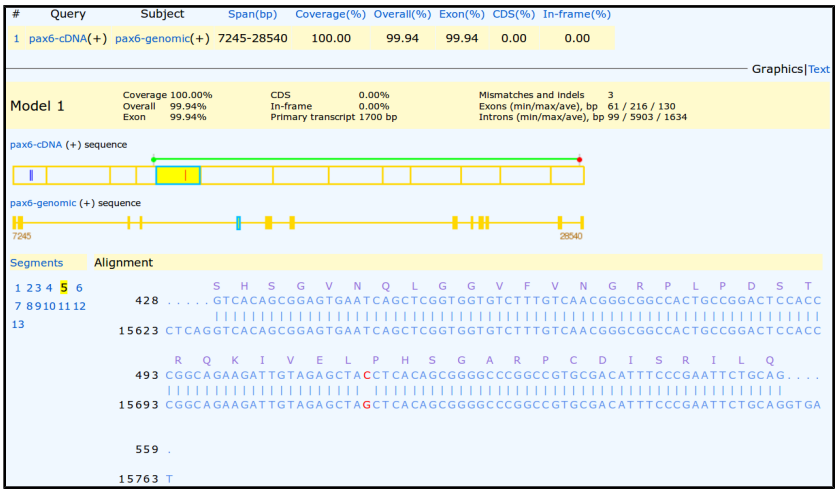




Your results will appear showing the cDNA split into **13** sections (the predicted exons) corresponding to **13** regions of the genomic sequence indicated by yellow rectangles. The first exon alignment is displayed showing two **cDNA** deletions. Though these are in a non-coding region, they could easily still be very significant. However, for the purposes of this exercise, let us assume they are not. The **Start** (green) and **Stop** (red) codons of the cDNA are illustrated by the bar above the cDNA display.

Click on the exon including the green **Start** codon (the fourth). The first coding exon is now displayed. The statistics at the top of the display include the claim that there are three discrepancies (**Mismatches and indels**[25]) between the **cDNA** and **Genomic** sequences. Two of these are the deletions we have already seen in the first exon of the cDNA. The third is indicated by the red bar in the fifth exon of the **cDNA** display.





Click on the fifth exon section of the cDNA display.

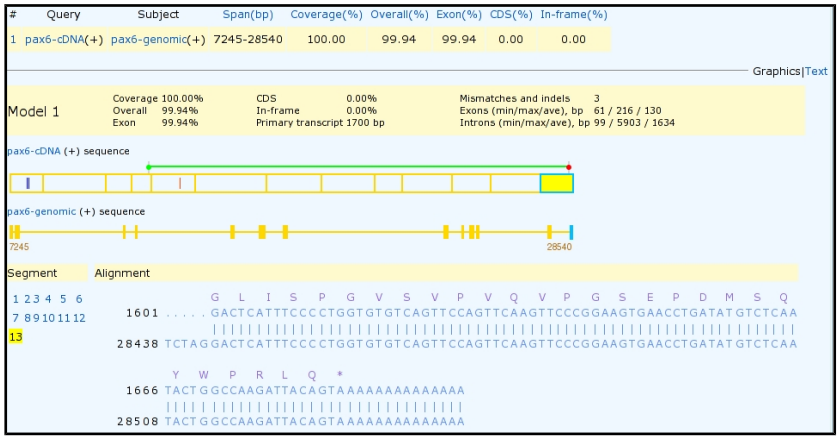
The third difference, a substitution, should be clear to see. Given it changes the coded protein, this substitution is likely to be the most significant.

What is the amino acid corresponding to this position in the mRNA of the aniridia patient[26]?_____

24 Why this is so will be considered later when we look at the database searching program **blast**.

25 An **indel** is either an **in**sertion or a **del**etion, depending upon which of the aligned sequences you are considering.

26 How this substitution affects the protein is not clear. Translating both the **Genomic** sequence and the **cDNA** might have helped? The **Standard Genetic Code** is in an Appendix, or look back to the **Uniprot** feature table for **PAX6_HUMAN**. It is the **Natural Variant** at position **33**.



Click on the last exon section in the cDNA display. You should now see the final exon of the cDNA with the **Stop** codon and polyA region.

Finally, click on the **Text** link to view the textual summary of the **splign** results.

#	Query	Subject	Span(bp)	Coverage(%)	Overall(%)	Exon(%)	CDS(%)	In-frame(%)
1	pax6-cDNA(+)	pax6-genomic(+)	7245-28540	100.00	99.94	99.94	0.00	0.00

Graphics|Text

#	Query	Subject	Idty	Len	Q.Start	Q.Fin	S.Start	S.Fin	Type	Details
+1	pax6-cDNA	pax6-genomic	0.981	103	1	101	7245	7347	<exon>GT	M53IM5IM43
+1	pax6-cDNA	pax6-genomic	1	188	102	289	7447	7634	AG<exon>GT	M188
+1	pax6-cDNA	pax6-genomic	1	77	290	366	11537	11613	AG<exon>GC	M77
+1	pax6-cDNA	pax6-genomic	1	61	367	427	12000	12060	AG<exon>GT	M61
+1	pax6-cDNA	pax6-genomic	0.992	131	428	558	15628	15758	AG<exon>GT	M86RM44
+1	pax6-cDNA	pax6-genomic	1	216	559	774	16686	16901	AG<exon>GT	M216
+1	pax6-cDNA	pax6-genomic	1	166	775	940	17606	17771	AG<exon>GT	M166
+1	pax6-cDNA	pax6-genomic	1	159	941	1099	23674	23832	AG<exon>GT	M159
+1	pax6-cDNA	pax6-genomic	1	83	1100	1182	24348	24430	AG<exon>GT	M83
+1	pax6-cDNA	pax6-genomic	1	151	1183	1333	24660	24810	AG<exon>GT	M151
+1	pax6-cDNA	pax6-genomic	1	116	1334	1449	24909	25024	AG<exon>GT	M116
+1	pax6-cDNA	pax6-genomic	1	151	1450	1600	27602	27752	AG<exon>GT	M151
+1	pax6-cDNA	pax6-genomic	1	98	1601	1698	28443	28540	AG<exon>AA	M98

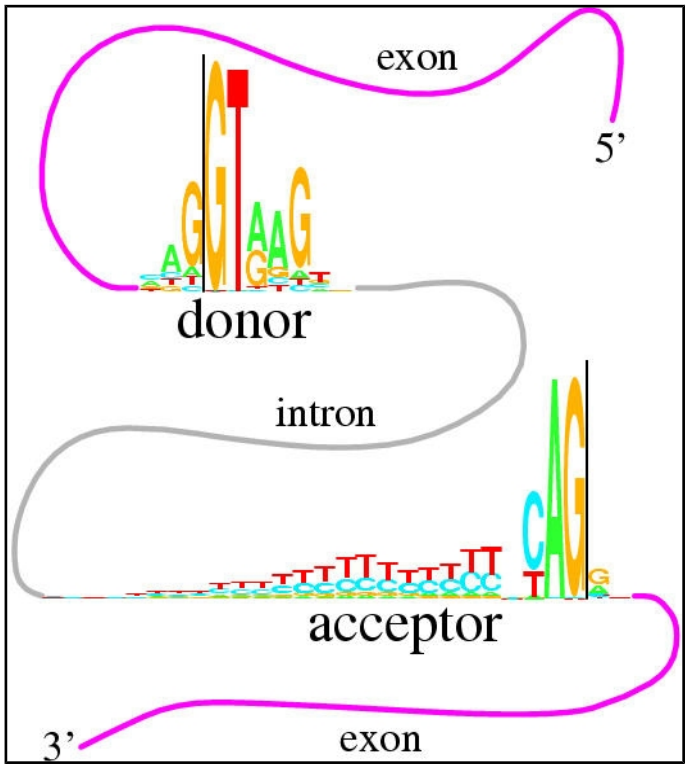
How do you interpret the **Details** column for exons 1 and 5?_____

Where is the substitution in the aniridia patient mRNA?_____

Where is the substitution in the Genomic Sequence?_____

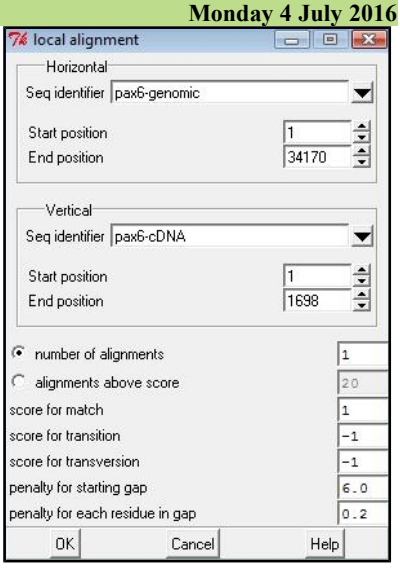
Compare the predicted **splign** intron/exon boundaries with the conservation suggested by the logo[27]?_____

What deviation(s) from the model suggested by the logo can you see?_____



27 The original label for this very nice graphic is:
This figure shows two ''sequence logos'' which represent sequence conservation at the 5' (donor) and 3' (acceptor) ends of human introns. The region between the black vertical bars is removed during mRNA splicing. The logos graphically demonstrate that most of the pattern for locating the intron ends resides on the intron. This allows more codon choices in the protein-coding exons. The logos also show a common pattern ''**CAG|GT**'', which suggests that the mechanisms that recognize the two ends of the intron had a common ancestor. See R. M. Stephens and T. D. Schneider, "Features of spliceosome evolution and function inferred from an analysis of the information at human splice sites", J. Mol. Biol., 228, 1124-1136, (1992).

Local sequence alignment

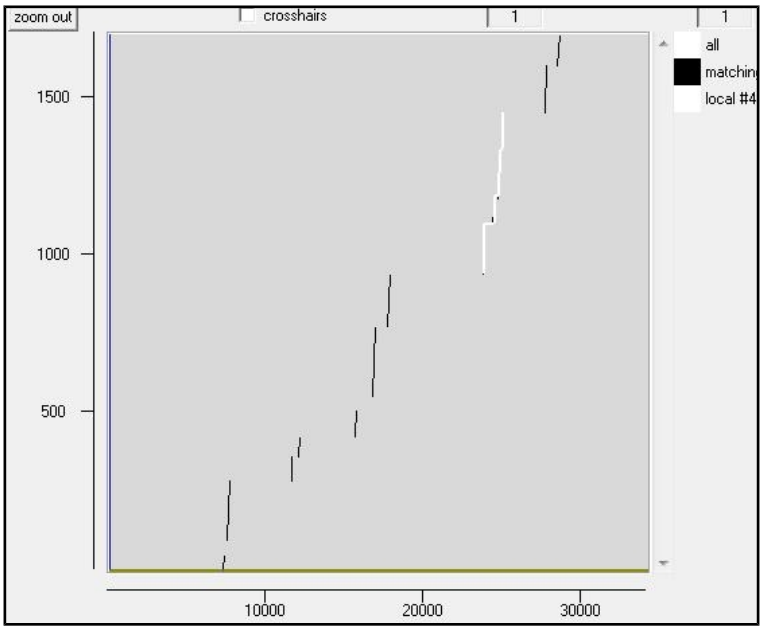


Global sequence alignment algorithms align sequences over their entire lengths. **Local** alignment methods searches for regions of similarity and need not include the entire length of the sequences. It is important to select the type of alignment that makes best sense for your sequences. Here, we expect an ordered sequence of matching regions (the exons) to occur in both sequences but spaced very differently. With an effort (or preferably appropriate software), we can get reasonably sensible alignments using a global or a local approach. However, if our sequences represented multi-domain proteins that shared some domains but not others, or the same domains but in differing orders, or proteins where there have been regions of duplication, a single sensible global alignment would not exist. A local approach would be essential.

Move back to **spin**, tidy away all previous graphics plots except one clear **dotplot** between **pax6-genomic** and **pax6-cDNA**. From **spin**'s **Comparison** menu, select **Local alignment**. Note that:

- in common with most local alignment programs, the default **number of alignments** to be reported is **1**[28]
- **transversions** and **transitions** may be penalized differently
- gaps are slightly cheaper than for **gobal** alignments

Accept all the defaults and click **OK**. The graphic suggests that the single "best" region that **spin** has elected to locally align, is the region around **25,000** in the genomic sequence where there are **4** exons that are close together. It was this region around which the global alignment algorithm chose to build its first attempt.

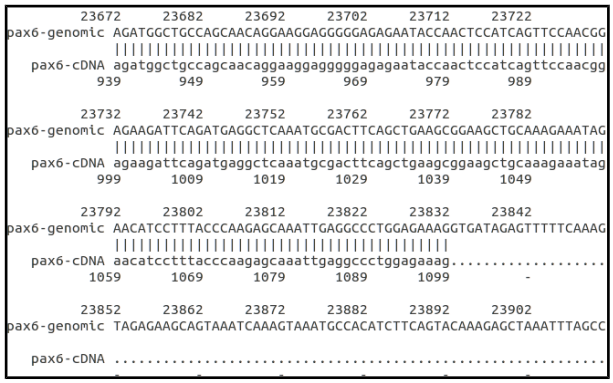


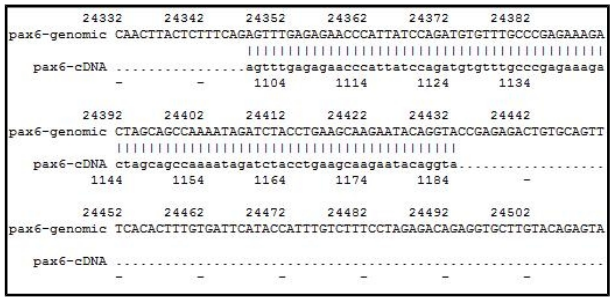
Check that the textual and graphic outputs agree.

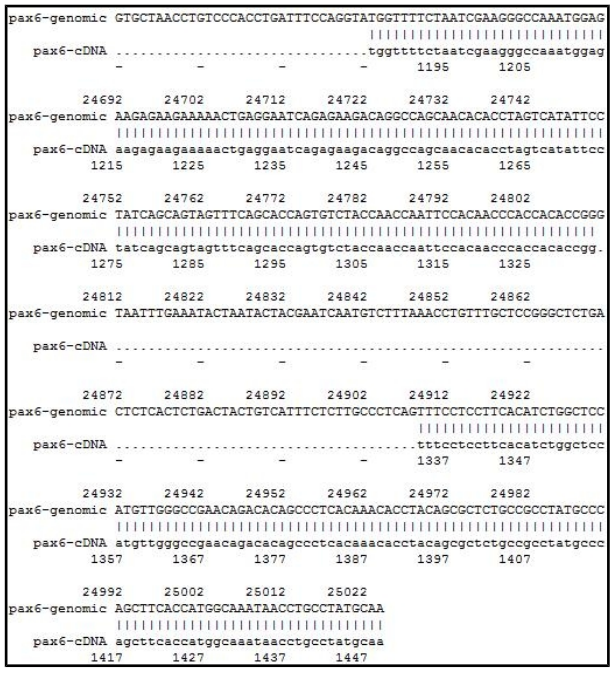
How might the gap around **24,600** in the genomic sequence been positioned more intelligently? _____

_____

The choice of this region is primarily due to the choice of gap penalties. If bigger gap penalties were selected (increase **penalty for each residue in gap** to **1**, say) **spin** would cease to regard the **4** exons around **25,000** as one entity. Their individual alignment scores would look less attractive. Eventually, the region around **16,700** (the longest single exon) would easily outscore the rest. If you have time, try it.

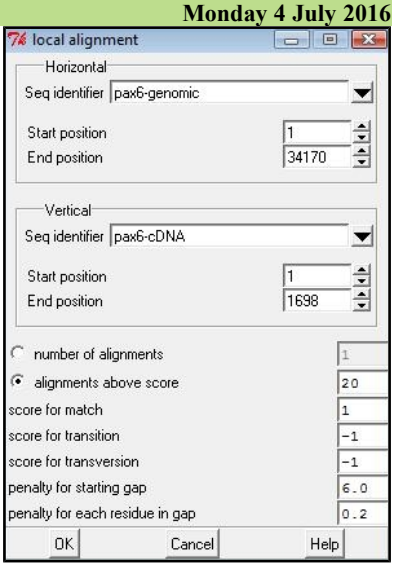






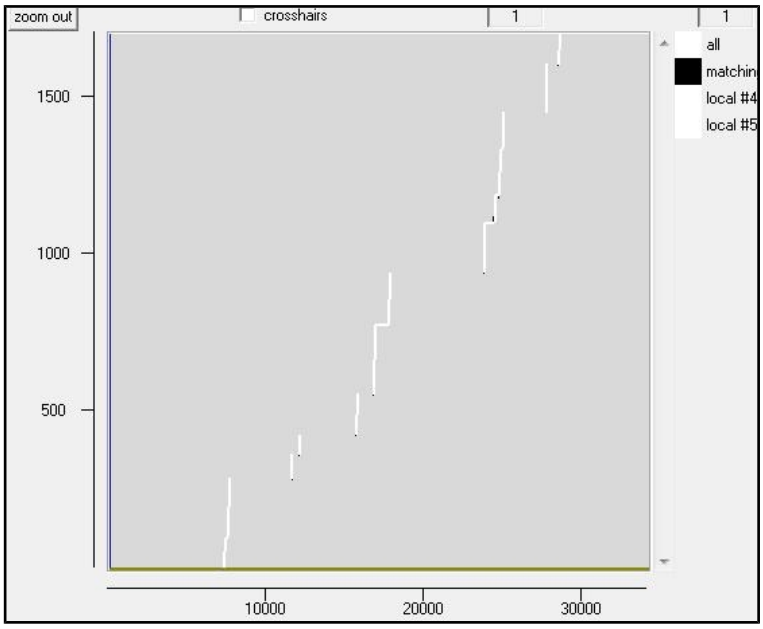
28 I.e. report the best matching region and ignore any others, however deserving they might be? This has always seemed an odd default to me.

From **spin**'s **Comparison** menu, select again **Local alignment**. Clearly, there is more than **1** meaningful local alignment. To see more, **spin** offers two options.



You could guess the **number of alignments** value. However, this is not trivial. You know the number of exons (and so the number of logical alignments expected) with some certainty by now, but this does not necessarily correspond to the expected number of local alignments. **spin** will combine close exons into single alignments if gap penalties are low enough. **spin** assumes that your choice of **number of alignments** is exactly correct. Too low and you will miss real alignments. Too high and **spin** will show alignments of ever increasing fantasy.

It is normally better, therefore, to specify the alignments you wish **spin** to display by providing a quality cut off rather than a volume cut off. To do this, turn on the **alignments above score** option. Use the default value of **20**.



It is not obvious how one might choose a sensible value. In such circumstances, the "lazy" option of just accepting the default is fine. Maybe someone sensible chose it? If not, one can always iterate to a value that works. I.e. if you would like more alignments, lower the value and try again. Too many? raise it. Repeat until the program agrees with what you wished to be correct in the first place. Is not science wonderful? With hope in your heart, click **OK**.

Looking at the graphic, I would say we got lucky this time[29]. There is a local alignment that covers every exon identified by the **dotplot** and no extra results requiring explanation.

Scan your results to find the exons. You should see that all have been correctly aligned.

Why do you suppose your aligned exons are not presented in the correct positional order?_____

_____

29 If you wish to be meticulously honest. You should **Hide** the graphic of your first **Local alignment** (right click its configuration box, choose **Hide**). Not that it will change anything as you will have found exactly the same alignment the second time round – along with the others.

Model Answers to Questions in the Instructions Text.

Notes:

For the most part, these "**Model Answers**" just provide the reactions/solutions I hoped you would work out for yourselves. However, sometime I have tried to offer a bit moer back ground and material for thought? Occasionally, I have rambled off into some rather self indulgent investigations that even I would not want to try and justify as pertenent to the objective of these exercises. I like to keep these meanders, as they help and entertain me, but I wish to warn you to only take regard of them if you are feeling particularly strong and have time to burn. Certainly not a good idea to indulge here during a time constrained course event!

Where things have got extreme, I am going to make two versions of the answer. One starting:

Summary:

Which has the answer with only a reasonably digestible volume of deep thought. Read this one.

The other will start:

Full Answer:

Beware of entering here! I do not hold back. Nothing complicated, but it will be long and full of pedantry.

This makes the Model answers section very big. **BUT**, it is not intended for printing or for reading serially, so I submit, being long and wordy does not matter. Feel free to disagree.

From your investigations of **Global Alignment**:

What do you suppose these regions represent?

Exons

Is the number of matching regions consistent with the number of exons you might expect?

Well, I think the only point left to this question is to say that the resolution of the graphics is insufficient to make a proper judgement concerning how many exons there might be. From looking at **Ensembl**, I would expect about **13** or **14**? From looking at this picture … I am simply not sure? I seem to have counted **12** when I first wrote this section.

I leave the question, but the need to know the exact number of exons, at this stage has become vanishingly small over the eons of evolution endured by these exercises. Let us say … **12**! Which I think is right.

If not, can you explain the discrepancy?

What discrepancy? I pause for thought, but why would I have an expectation of exon count here anyway? Depends which **transcript** and which **isoform**.

So … I decline to try to explain anything further. Stupid question!

Why do you suppose that might be?

The larger the **window size**, the less likely it is that small features, or noise, will be detected. The signal from a small feature (noise or otherwise) will be too weak to influence the general impression of a relatively large window. Therefore. Ther longer the **window size** you choose, the stronger/longer the feature has to be before it will be noticed.

The presence of an extended strong feature will be detected as long as some part of it remains in a scoring window. The longer the **window**, the further the **window** must be from the feature before its influence is insignificant. Therefore, the longer the **window**, the longer strong features will appear to be.

How many convincingly aligned regions did you see?

4

Roughly how many did you expect?

12 or so … one per exon anyway.

Clearly, this alignment is not correct. Can you explain why?

This alignment algorithm only wishes to maximise an alignment score. It sees ***ALL*** the high scoring exon regions, however, as the gaps between many of the exons (introns that is) are so long that the penalties for representing them correctly are greater than the gain achieved by the inclusion the extra exons in the alignment. Arithmetically, it is better to align all the exons either side of the **4** exons that were aligned sensibly, in the biologically improbably fashion shown. Arithmetically the best alignment, biologically ridiculous!

This behaviour is exaggerated because this program regards the enormous gaps in has suggested at the start and end of the alignments as “free”. Some global alignment programs offer the option of penalising the ends gaps in the same way as for internal gaps. Normally, not penalising end gaps is sensible as it allows for the sequences to have slightly different lengths. In this case, penalising end gaps should have resulted in a better alignment.

Had you used stretcher (the faster, less vigorous algorithm offered by the **EMBOSS** package) you would have got a much improved answer in this case (but not generally). This is because **stretcher** works in a way far closer to the way an informed human might think. **stretcher** does not mindlessly insist of the highest alignment score. Instead, it looks for all the high scoring regions (i.e. all the exons) and then computes the best way to link them together. The result is a far more convincing alignment, but not the arithmetically best scoring answer.

How many matching regions are there this time?

Where you to trawl though your textual output carefully (or simply take my immaculate word for it), you would find **13** perfectly (or nearly so) aligned regions, implying **13** exons.

To be pedantic, the nicely aligned regions do not match the exons exactly (as will become apparent later), so it is not possible to claim definite evidence for any particular number of exons. However, **13** has to be a pretty confident estimate.

Is the count **now** roughly as you would expect?

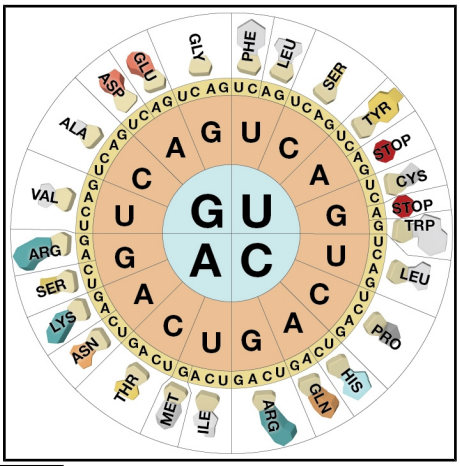
Yes, roughly as suggested by **Ensembl**.

From your investigations comparing mRNA/cDNA with genomic DNA:

What is the amino acid corresponding to this position in the mRNA of the aniridia patient?

This is easy to answer. The top sequence is the mRNA from the **aniridia** patient. **spline** is kind enough to explicitly inform us that the mutated codon, **CCT**, will be expressed a **P**roline.

So, why not translate the wild type genomic sequence also **spline**?! Easy enough to look up. But I resent having to do so!



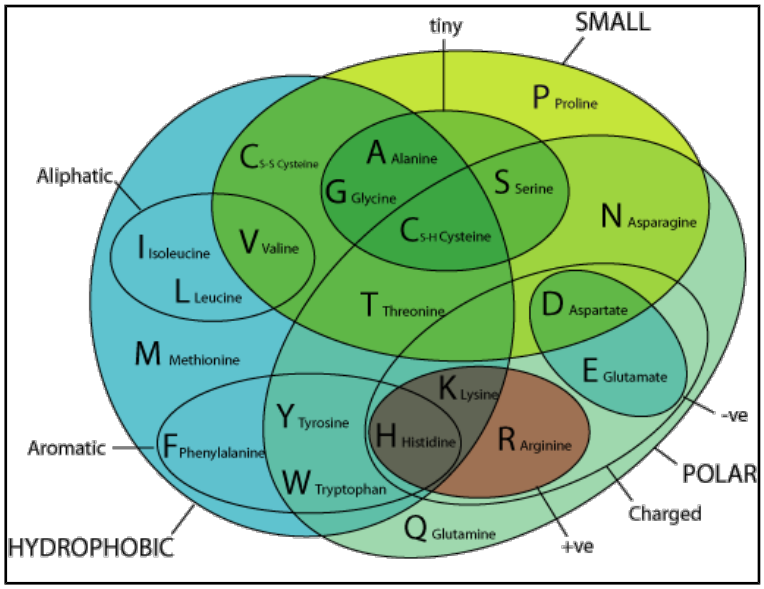
From this rather beautiful representation of the **Genetic Code**, I conclude:

Patient **CCT** → **Proline** (**P**)
Wild Type **GCT** → **Alanine** (**A**)

Feature key	Position(s)	Length	Description	Graphical view	Feature identifier
Natural variant	17 – 17	1	N → S in AN. 1 Publication		VAR_003808
Natural variant	18 – 18	1	G → W in AN. 1 Publication		VAR_003809
Natural variant	19 – 19	1	R → P in AN. 1 Publication		VAR_047860
Natural variant	22 – 26	5	Missing in AN. 2 Publications		VAR_008693
Natural variant	26 – 26	1	R → G in PETAN. 2 Publications		VAR_003810
Natural variant	29 – 29	1	I → S in AN. 1 Publication		VAR_008694
Natural variant	29 – 29	1	I → V in AN. 1 Publication		VAR_003811
Natural variant	33 – 33	1	A → P in AN. 1 Publication		VAR_008695
Natural variant	37 – 39	3	Missing in AN. 1 Publication		VAR_008696

Or, looking back at the relevant **Uniprot Feature Table**, I come to an identical conclusion. This is the Natural Variant at position **33**, as you could easily confirm with a bit of counting along the alignment.

Either way, it could be a quite serious mutation. **Alanine** and **Proline** having quite distinct properties.



Also, according to the font of all easily packaged knowledge, Wikipedia:

> "**Proline** and **glycine** are sometimes known as "**helix breakers**" because they disrupt the regularity of the α helical backbone conformation; however, both have unusual conformational abilities and are commonly found in **turns**."

The **helices** of this protein are of particular importance to its vital **DNA Binding** role.

How do you interpret the **Details** column for exons 1 and 5?

Summary:

The **Details** column shows the alignments of each exon in a compressed format described in the **spline** documentation as illustrated.

11. Alignment transcript	Alignment transcript represents full details of the alignment in a form of a string composed of characters 'M', 'R', 'I' and 'D' where each character corresponds to an elementary command (Match, Replace, Insert or Delete) needed to transform the query segment into the subject segment. The string is encoded with RLE.

The majority of the exon alignments are trivial.

#	Query	Subject	Span(bp)	Coverage(%)	Overall(%)	Exon(%)	CDS(%)	In-frame(%)
1	pax6-cDNA(+)	pax6-genomic(+)	7245-28540	100.00	99.94	99.94	0.00	0.00

Graphics|Text

#	Query	Subject	Idty	Len	Q.Start	Q.Fin	S.Start	S.Fin	Type	Details
+1	pax6-cDNA	pax6-genomic	0.981	103	1	101	7245	7347	<exon>GT	M53IM5IM43
+1	pax6-cDNA	pax6-genomic	1	188	102	289	7447	7634	AG<exon>GT	M188
+1	pax6-cDNA	pax6-genomic	1	77	290	366	11537	11613	AG<exon>GC	M77
+1	pax6-cDNA	pax6-genomic	1	61	367	427	12000	12060	AG<exon>GT	M61
+1	pax6-cDNA	pax6-genomic	0.992	131	428	558	15628	15758	AG<exon>GT	M86RM44
+1	pax6-cDNA	pax6-genomic	1	216	559	774	16686	16901	AG<exon>GT	M216
+1	pax6-cDNA	pax6-genomic	1	166	775	940	17606	17771	AG<exon>GT	M166
+1	pax6-cDNA	pax6-genomic	1	159	941	1099	23674	23832	AG<exon>GT	M159
+1	pax6-cDNA	pax6-genomic	1	83	1100	1182	24348	24430	AG<exon>GT	M83
+1	pax6-cDNA	pax6-genomic	1	151	1183	1333	24660	24810	AG<exon>GT	M151
+1	pax6-cDNA	pax6-genomic	1	116	1334	1449	24909	25024	AG<exon>GT	M116
+1	pax6-cDNA	pax6-genomic	1	151	1450	1600	27602	27752	AG<exon>GT	M151
+1	pax6-cDNA	pax6-genomic	1	98	1601	1698	28443	28540	AG<exon>AA	M98

For example:

For **Exon 2,** **spline** informs us **M188**, meaning "There are **188** bases aligned and they all **M**atch perfectly".

For **Exon 3**, **spline** informs us **M77**, meaning "There are **77** bases aligned and they all **M**atch perfectly".

The only **2** interesting entries are those were there are some disagreements. That is, the entries for **Exons 1** and **5**, which, following the documentation, I translate thus:

Exon 1 – M53IM5IM43

An alignment of **103** bases, the first **53** of which **M**atch perfectly (**M53**), there then follows an **I**nsertion (**I**), a further **5 M**atched bases(**M5**), a second **I**nsertion (**I**) all finished off with **43 M**atched bases (**M43**).

Exon 5 – M86RM4R

An alignment of **131** bases, the first **86** of which **M**atch perfectly (**M86**), there them follows a **R**eplacement (**R**) and a further **44 M**atched bases(**M44**).

Full Answer:

From the individual **Exon 1** display, it can be inferred that the declaration of an **I**nsertion or a **D**eletion is made to describe the type of variation required to transform the **cDNA** (**Query**) sequence into the **genomic** (**Subject**). Hence the two **indels** (**I**nsertions or **D**eletions) are considered to be **I**nsertions.

```
   1 CAGAGGTCAGGCTTCGCTAATGGGCCAGTGAGGAGCGGTGGAGGCGAGGCCGG-CGCCG-CACACACACA
     ||||||||||||||||||||||||||||||||||||||||||||||||||||| ||||| ||||||||||
7245 CAGAGGTCAGGCTTCGCTAATGGGCCAGTGAGGAGCGGTGGAGGCGAGGCCGGGCGCCGGCACACACACA
```

Not that it is a vital issue, but I would have thought the other way around was more logical? That is, to consider the **genomic** sequence as the **reference** against which a particular **mRNA** might vary. In other words, what we see here would surely be more relevantly recorded as "This **mRNA/cDNA** has two **D**eletions relative to the **genomic** sequence which, presumably, attempts to represent the norm in the general population"? Just the reflection of an irretrievable pedant, but I am right, nevertheless!!!

In the documentation it enigmatically states "The string is encoded with **RLE**.". Just in case, **RLE** stands for **Run-length encoding** which is succinctly defined by **Wikipedia**. In a nutshell, it is a very simple form of data compression that recognizes that:

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

can be compressed to:

```
60x
```

which has to be very effective for any data that has runs of identical characters of significant length. This is certainly the case here where one would expect long stretches of **M**s in most alignments. Of course, life would get tricky if the data included numeric characters, but that is not an issue here[30].

I think it worth mentioning, that this way of representing an alignment is a simplification of **CIGAR** format[31]. This format is used for **SAM** (**S**equence **A**lignment **M**ap) and **BAM** (**B**inary **A**lignment **M**ap, exactly the same as **SAM**, except compressed) files. You will be engulfed in **SAM/BAM** files if you ever do any **N**ext **G**eneration **S**equencing (**NGS**).

So, straight form the **SAM/BAM Format Specification** I copy the table of **CIGAR** enlightenment.

CIGAR: CIGAR string. The CIGAR operations are given in the following table (set '*' if unavailable):

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

- H can only be present as the first and/or last operation.
- S may only have H operations between them and the ends of the CIGAR string.
- For mRNA-to-genome alignment, an N operation represents an intron. For other types of alignments, the interpretation of N is not defined.
- Sum of lengths of the M/I/S/=/X operations shall equal the length of SEQ.

Note, in particular, the extended range of **Op**erators and the different meaning associated with the operator '**M**'. The operators '**=**' and '**X**' are such that any '**M**' is either an '**=**' or and '**X**' but never both. Which leaves one pondering whem one might use '**M**' in preference to either an '**=**' or an '**X**'?

Where is the substitution in the aniridia patient mRNA?
Where is the substitution in the Genomic Sequence?

spline makes one work quite hard to answer this one! Unless I am missing something.

From the alignment of **Exon 5**, the exon including the **R**eplacement, with a bit of squinting, it can be confirmed that the **R**eplacement is at:

```
        R  Q  K  I  V  E  L  P  H  S  G  A  R  P  C  D  I  S  R  I  L  Q
  493 CGGCAGAAGATTGTAGAGCTACCTCACAGCGGGGCCCGGCCGTGCGACATTTCCCGAATTCTGCAG....
      ||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||
15693 CGGCAGAAGATTGTAGAGCTAGCTCACAGCGGGGCCCGGCCGTGCGACATTTCCCGAATTCTGCAGGTGA
```

Base pair position **514** of the aniridia patient's **mRNA**

Base pair position **15714** of the **genomic** sequence

It might also have been relevant to ask which amino acid position corresponded to the **R**eplacement. To discover this one would need to look at the alignment of **Exon 3**, where the coding begins.

```
                                                          M  Q  N
  367 .....AGCCCCATATTCGAGCCCCGTGGAATCCCGCGGCCCCCAGCCAGAGCCAGCATGCAGAACA....
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
11995 AACAGAGCCCCATATTCGAGCCCCGTGGAATCCCGCGGCCCCCAGCCAGAGCCAGCATGCAGAACAGTAA
```

More squinting, and I conclude the **A** of the **ATG** representing the initial **Methionine** of the protein coding region is at position **418**. That is, the **3' UTR** ends at position **417**. So the **R**eplacement is at:

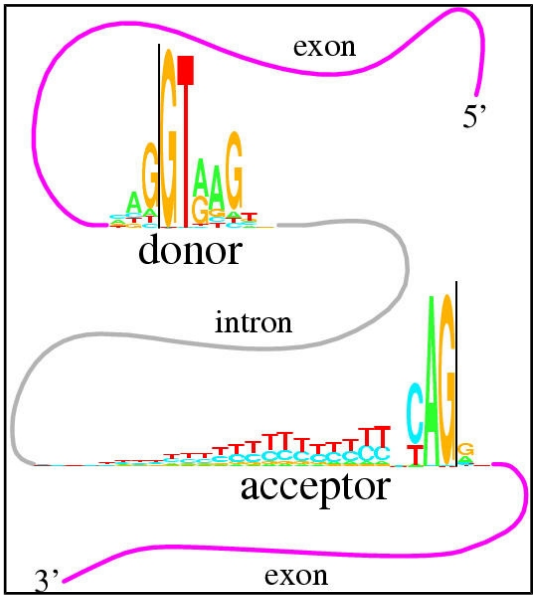
Base position **514** – **417** = **97** of the protein coding region of the **mRNA**.

As **97** / **3** is **32** remainder **1**, the **R**eplacement is at codon position **1** of the **33rd** amino acid of the protein.

Which you knew already, of course! Cannot help thinking that **spline** might have helped a bit more here?

30 The **Wikipedia** article shows how this complication might be overcome.
31 There may or may not be some justification for calling the format **CIGAR**, but if there is, I have no idea what it might be.

Compare the predicted **splign** intron/exon boundaries with the conservation suggested by the logo?
What deviation(s) from the model suggested by the logo can you see?



You may have gathered, I rather like this logo, although I rather think it is leading me to make the same point a trifle to often?

The logo is in almost **100%** agreement with the predictions of **spline**.

As you will have noted previously, when looking at the **Ensembl** predictions of exons locations of a similar transcript of the **PAX6** human gene, there is a single exception.

Type
<exon>GT
AG<exon>GT
AG<exon>GC
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>GT
AG<exon>

The easiest way to show this in the **spline** output is to look at the **spline** text output again.

The **Type** column records the type of all the **<exon>** alignments it predicts. It also records **2 flanking intron base pairs**.

It is clear that the only time the spline prediction deviates from the model suggested by the logo is at the end of the **3rd** exon. Here there is **GC** rather then **GT**. Well, nothing is perfect!

From your investigations of **Local Alignment**:

How might the gap around **24,600** in the genomic sequence been positioned more intelligently?

spin has positioned a gap in this region merely to maximize the overall alignment score. There is more than one way of achieving this simple goal. However, if it were to be recognized that the gap to be positioned was to represent an intron, then one of the arithmetically equivalent options becomes far more attractive than the others. This "best" option is not the one chosen by **spin**, which is forgiveable as **spin** had nor reason to expect an intron and was not written to understand the properties of introns anyway.

The alignment chosen for this region by spin was:

```
          24392     24402     24412     24422     24432     24442
pax6-genomic CTAGCAGCCAAAATAGATCTACCTGAAGCAAGAATACAGGTACCGAGAGACTGTGCAGTT
             |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
   pax6-cDNA ctagcagccaaaatagatctacctgaagcaagaatacaggta..................
          1144      1154      1164      1174      1184          -

                                    • • •

          24632     24642     24652     24662     24672     24682
pax6-genomic GTGCTAACCTGTCCCACCTGATTTCCAGGTATGGTTTTCTAATCGAAGGGCCAAATGGAG
                                       ||||||||||||||||||||||||||||
   pax6-cDNA ..............................tggttttctaatcgaagggccaaatggag
             -         -         -         -      1195      1205
```

Shifting the gap 3 places to the left neither changes the size of the gap nor the perfection of the alignment either side of the gap and so does not affect the alignment score. However, it does mean the gap begins with an **GT** and ends with a **AG** which is what one might expect if it were known that the gap represented an intron. So, if **spin** was a little better informed, the improved alignment would have been:

```
          24392     24402     24412     24422     24432     24442
pax6-genomic CTAGCAGCCAAAATAGATCTACCTGAAGCAAGAATACAGGTACCGAGAGACTGTGCAGTT
             |||||||||||||||||||||||||||||||||||||||
   pax6-cDNA ctagcagccaaaatagatctacctgaagcaagaatacag.....................
          1144      1154      1164      1174      1184          -

                                    • • •

          24632     24642     24652     24662     24672     24682
pax6-genomic GTGCTAACCTGTCCCACCTGATTTCCAGGTATGGTTTTCTAATCGAAGGGCCAAATGGAG
                                        |||||||||||||||||||||||||||||
   pax6-cDNA ............................gtatggttttctaatcgaagggccaaatggag
             -         -         -         -      1195      1205
```

This is the alignment that the customized program **splign** chose as **splign** understands something of the expected properties of introns. **spin** was confused because it is a general alignment program concerned only with the simple arithmetic of maximising alignment scores.

Why do you suppose your aligned exons are not presented in the correct positional order?

To **spin**, the logical order in which to present the alignments is that governed by quality rather than position. So, the highest scoring alignment, rather than the first exon alignment, will be at the top of the list. I think this is generally logical. Once again, the program **splign**, knowing it was looking for an ordered set of exons, was more obliging.

From your investigations of **ORF detection**:

At what base position does the coding sequence of the mRNA commence?

From the illustration in the notes above, it can be determined (with a bit of pain filled arithmetic) that the **A** base of the **ATG Methionine** codon that starts the coding region of this mRNA is at position **418**.

Note that the nearest **Stop** Codon before the **Methionine** is well before the start of the ORF indicated by **plotorf**. This shows that the definition of an ORF being used by **plotorf** is **Start → Stop**, rather than **Stop → Stop**. As you can see from the output below (generated by another **Emboss** program called **showorf**).

```
            ---------|---------|---------|---------|---------|
       301 TAActaggggcgcgcagatgtgtgaggccttttattgtgagagtggacag 350
F1       1 *  L  G  A  R  R  C  V  R  P  F  I  V  R  V  D  R  16

            ---------|---------|---------|---------|---------|
       351 acatccgagatttcagagccccatattcgagccccgtggaatcccgcggc 400
F1      17  H  P  R  F  Q  S  P  I  F  E  P  R  G  I  P  R  P 33

            ---------|---------|---------|---------|---------|
       401 ccccagccagagccagcATGcagaacagtcacagcggagtgaatcagctc 450
F1      34   P  A  R  A  S  M  Q  N  S  H  S  G  V  N  Q  L   49
```

What is the base position of the last base of the coding sequence?

Slide along the sequence display and you should be able to conform the coding region of this mRNA ends at position **1683**. To complete the dreadful pedantic excess, I support his with more **showorf** output.

```
            ---------|---------|---------|---------|--------
      1651 cctgatatgtctcaatactggccaagattacagTAAaaaaaaaaaaa 1698
F1     450 P  D  M  S  Q  Y  W  P  R  L  Q  *  K  K  K  K   4
```

Leaving only the mystery of what **showorf** might be trying to convey by the **"4**" at the end of the display? I find my concern drifting somewhat.

What is the single amino acid difference between the two sequences?

It has been established that there is but one base pair difference in the coding region of this cDNA and the wild type genomic sequence. There can therefore be, at most, one amino acid that is different between the two protein sequences. Surely a case for a global rather than a local alignment strategy? Not that the choice should be of any real consequence with sequences that are this similar.

```
                    1        11        21        31        41        51
sp|P26367|PAX6_H MQNSHSGVNQLGGVFVNGRPLPDSTRQKIVELAHSGARPCDISRILQVSNGCVSKILGRY
                 ********************************:***************************
      pax6 MQNSHSGVNQLGGVFVNGRPLPDSTRQKIVELPHSGARPCDISRILQVSNGCVSKILGRY
                    1        11        21        31        41        51
```

From the alignment I generated with the global option of **spin**, there is a **A → P** (**Alanine → Proline**) substitution evident. **Alanine** and **Proline** are amino acids with quite different properties, so it is reasonable to suppose that the this substitution will be significant.

What is the position of the difference?

The substitution is at residue **33**, as you will have seen reported by various sources previously.

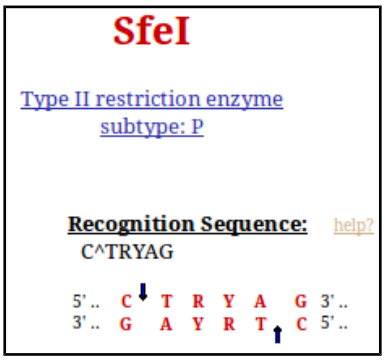
From your investigations of **Restriction Mapping**

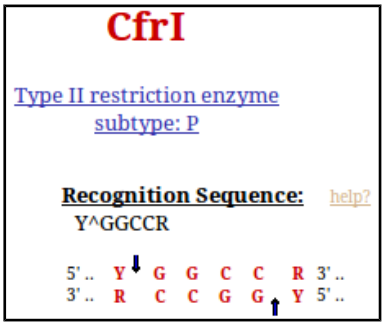
There are quite a few less enzymes mentioned in the map you have just made compared to that generated by **remap**. Can you speculate what the main reason for this might be?

Probably differences in the definition of a **6** cutter.

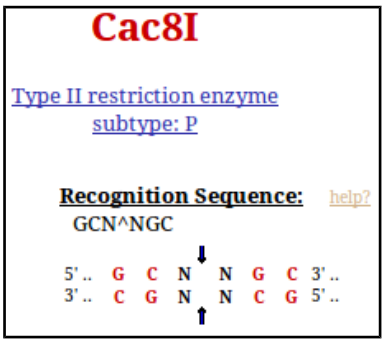
E.g. **BfuAI** has a recognition site of **ACCTGCNNNN_NNNN-**, is this a **6** base pair recognition site or **14**?

Enzymes in remap maps but not reported by nebcutter include:





6 cutters according to **remap**? Surely **R** & **Y** only count **0.5**? I agree with **nebcutter**, these are **5 cutter**s.



Oh come on **remap**!! you cannot count **N**s!! This is a **5 cutter** also.

Some enzymes in this map appear in the same place as **remap** predicted, but have different names. Can you explain why?

Restriction mapping programs, by default, only map one member of each **isoschizomer** family. There is no consistency between programs in the choice of the representative enzyme.

For example:

Commercially Available:

Enzymes	Cloned	Sequenced	Recognition Sequence	Suppliers
BseX3I	-	-	C↓GGCCG	IV
BstZI	-	-	C↓GGCCG	R
EagI	yes	yes	C↓GGCCG	N
EclXI	-	-	C↓GGCCG	MS
Eco52I	-	-	C↓GGCCG	FK

Count: 5

remap chooses to show just **XmaIII**

Not Commercially Available:

Enzymes	Cloned	Sequenced	Recognition Sequence
AaaI	-	-	C↓GGCCG
BsoDI	-	-	CGGCCG
SenPT16I	-	-	C↓GGCCG
TauII	-	-	CGGCCG
Tsp504I	-	-	CGGCCG
XmaIII	-	-	C↓GGCCG

Count: 6

nebcutter chooses **EagI**, probably because it is commercially available.

They both have the same recognition site and so would both cut at the indicated position.

From your investigations of **Searching for sequence similarities in databases**

When would **Mask lower case letters** be a useful thing to do?

Generally, whenever one might suspect the automatic masking algorithms of **blast** might miss a non informative region in a specific query sequence, obviously.

A specific example might be when a query sequence contained a significant informative region that was known to be common amongst the sequences being searched. If this region was left unmasked, **blast** would pick up so many similar matches to this one region that other interesting similarities might be obscured. By manually masking such a region by changing it to lower case, its matches would not be seen by **blast** and matches with other regions of the query sequence should be more apparent.

Which parameters would **blast** need to **automatically adjust** to cater for short input sequences (such as primers being tested for uniqueness), and why?

The **word size**: Clearly, if you are trying to find matches for a primer (for example) of around **20** base pairs, it would be pretty silly to use a **word size** of **28** (default for **megablast**). A **word** the same size as the primer would find only exact matches. A **word** of about **7** would allow a couple of mismatches and would probably be most generally appropriate.

The **expect score**: As good chance matches between between a short query sequence and a large database will be abundant, it would not be sensible to choose a demanding (i.e. small) **expect score** to represent the limit of significance. In particular, a primer sized query sequence of around **20** base pairs might easily exactly match more than **10** times (generally the default maximum expect score for a significant match) just by chance. After all, there are only **4** bases, a string of **20** is not that long and the databases can be huge! Typically **blast** chooses very high **expect score** cut off for short query sequences, effectively removing the **expect score** filter altogether.

Earlier versions of **blast** did not automatically adjust these parameters. When a short query sequences were selected, suitable adjustment was left to the user. Without sensible parameter adjustment, results could be greatly confusing. For example, a **21** base pair primer could easily match perfectly more than **10** times against a large DNA sequence database. **blast** is set to ignore matches that are expected to occur more than **10** times by chance. Thus even exact matches with such a small sequences would be ignored! Now automatic parameter adjustment is undertaken by **blast**, the user does not really have to think too hard. However, it does seem to be a good idea to know what **blast** is doing and why.

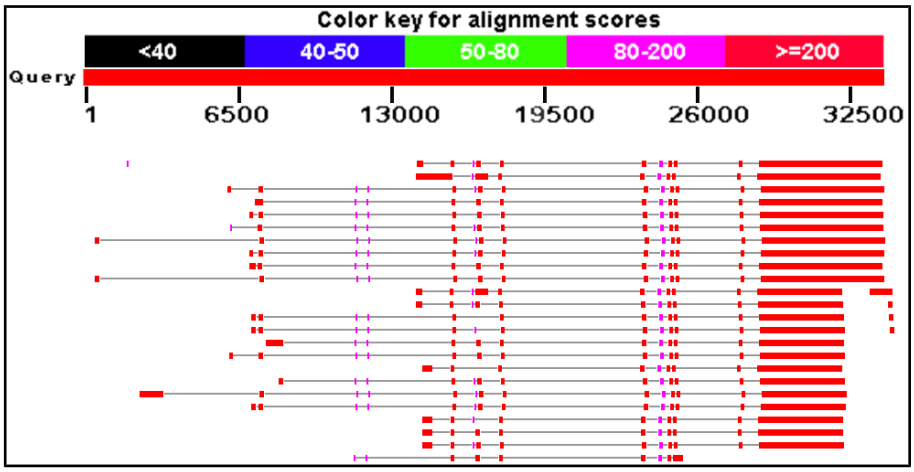
Why do you suppose that a few of the exons do not achieve the maximum score?

Summary:

Each local region of significant alignment between a database entry and a query sequence is scored independently. The scoring method that governs the alignment score colour in this graphic, reflects both the quality of the match ***and*** its length. Unless a particular region is of sufficient length, it cannot achieve the **200 bit** threshold even if the alignment is perfect. Note that is is the shorter regions that fail to reach the >=200 status. All of the illustrated local alignments associated with **PAX6** transcripts are essentially perfect.

Full Answer:

In common with most database searching programs, **blast** compares query sequences with database entries using a local strategy. The overall evaluation of a particular query sequence is taken to be the highest local score.



Individual local matches are coloured according to individual quality. In this query, all true matches should be perfect, or very nearly so. Scores might therefore be expected to be maximal (>=200). However, they are not? Some only manage a score in the range 80-200.

The score referenced for this purpose is the **bit score**. For a full, no holds barred definition of this score, try here. I prefer this somewhat gentler version:

> "The **bit score** gives an indication of how good the alignment is; the higher the score, the better the alignment. In general terms, this score is calculated from a formula that takes into account the alignment of similar or identical residues, as well as any gaps introduced to align the sequences. A key element in this calculation is the "substitution matrix ", which assigns a score for aligning any possible pair of residues. The **BLOSUM62** matrix is the default for most **BLAST** programs, the exceptions being **blastn** and **MegaBLAST** (programs that perform **nucleotide–nucleotide** comparisons and hence do not use protein-specific matrices). Bit scores are normalized, which means that the bit scores from different alignments can be compared, even if different scoring matrices have been used."

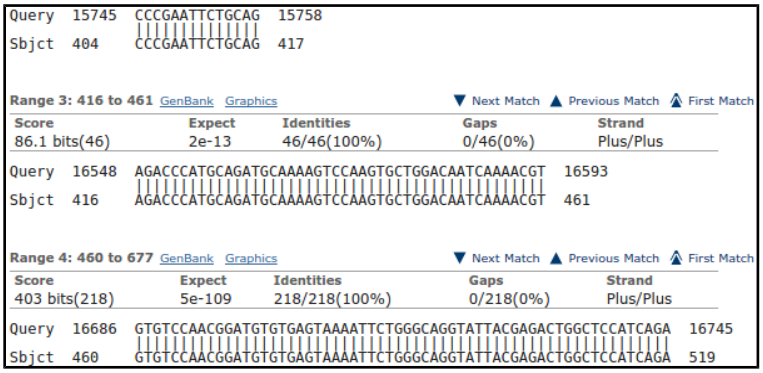
Still too scary? The important things to note are that:

- These scores are based on a simple DNA scoring matrix (**1** for a match, **-2** for a mismatch by default for **megablast**), plus penalties for gaps. So scores will be limited by the length of the alignment, ignoring gaps.
- The scores reflect penalties for **indel**s (**in**sertions or **del**etions).
- The scores are normalised so that they do not depend on the chosen scoring matrix. This allows bits scores from searches using different scoring matrices to be compared.

This being so, **bit scores** will reflect the length of an alignment as well as its quality. If an alignment is very short, it might be perfect but still not achieve a very high value. **bit scores** are designed to reflect significance, not just local quality. A short perfect match clearly can be less significant than a longer less perfect match. That is what you see illustrated here.

You can see evidence of what is occurring in the alignments further down your results. Here is illustrated one of the 80-200 exons that occur in all transcripts at position **24,346**[32]. The match is perfect, but the length of the exon is consistently just to short to get to the heady >=200 level.

Note how imperfectly **blast** finds exon/intron boundaries. If the start of an intron happens to match the start of the next exon, **blast** will included the bases in two alignments[33]. It is not looking for exons and introns as was **spline**, it just mindlessly seeks matches.



For a further example, look at the exon that is found only in the **isoform 5a** transcripts. It is tiny (**42** base pairs) and scores well below >=200 even thought it is a perfect match.

Note that the alignment is **46** base pairs long due to **blast** adding on two bases either side that are actually the highly conserved intron start and end base pairs. As you can see, these extra base pairs occur in the preceding and succeeding alignment also.

Explain why one exon in the reasonably consistent region, does not appear in all of the transcript matches?

Oh dear oh dear! Not this again.

Well I refer to the **isoform 5a** exon, of course. The tiny inconsistent one about **9** exons in from the right (when it exists). This will, clearly, only occur in **isoform 5a** transcripts. One day I will tidy these questions up a trifle!

Why were you not surprised to discover **24 PAX6** transcripts in **Refseq** matching this sequence?

Repetitive? True. This question is more "interesting" when the resources you have visited do not agree. That is, most of the time. Whilst the situation is in balance, the answer is:

Because **GeneCards** says there should be **11** quality and a further **13** less supported **PAX6 mRNA** sequences in **RefSeq**. A total of **24 PAX6** implied transcripts in total. In passing, you could have discovered the number of **PAX6 mRNA** sequences in **RefSeq** but asking **RefSeq** directly. Probably a more sensible and certainly a more reliable approach.

32 In order to make this illustration, I needed set **Sort by**: (top of the alignments) to **Query start position**.

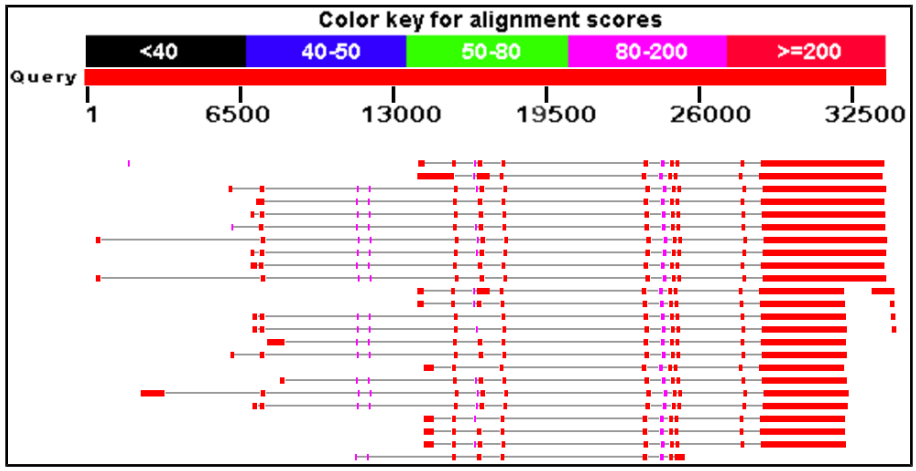
33 **6** base pairs (**Sbjct**: **1081-1086**, **CAGGTA**) occur in both the first two matches illustrated. Just **1** base pair is shared between the **2nd** and **3rd** match (**Sbjct**: **1234**, **G**).

Which of the **Refseq PAX6** transcripts corresponds to **isoform 5a**?

Summary:

As I am sure you are tired of noting by now, all the transcripts with the extra tiny exon around position **1,600** in the genomic sequence are **isoform 5a** transcripts.

Full Answer:



The **isoform 5a** transcripts can be spotted most easily from the graphic. They are the ones with the extra small exon slightly to the left of middle (around base position **1,600**). For example, the **first**, **second** and **third blast** matches displayed. If you hover over these matches with your mouse, you will see that they are t**ranscript variants 1**, **2**, **3**, **6**, **7**, **8**, **11**, **12**, **14**, **18**, **19**, **20**, **21** and **23** (in the vertical order of the graphic).

Stated with the unequalled poetry of **RefSeq Accession Code** and lyrical **Title** Line, that becomes:

TITLE	ACCESSION CODE
Homo sapiens paired box 6 (PAX6), transcript variant 11, mRNA	NM_001310161.1
Homo sapiens paired box 6 (PAX6), transcript variant 10, mRNA	NM_001310160.1
Homo sapiens paired box 6 (PAX6), transcript variant 8, mRNA	NM_001310158.1
Homo sapiens paired box 6 (PAX6), transcript variant 5, mRNA	NM_001258463.1
Homo sapiens paired box 6 (PAX6), transcript variant 4, mRNA	NM_001258462.1
Homo sapiens paired box 6 (PAX6), transcript variant 2, mRNA	NM_001604.5
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X13, mRNA	XM_005252958.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X12, mRNA	XM_011520153.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X10, mRNA	XM_011520152.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X6, mRNA	XM_011520150.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X5, mRNA	XM_011520149.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X4, mRNA	XM_005252954.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X3, mRNA	XM_011520148.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X1, mRNA	XM_011520146.1

Yes well, that was fun? When I first wrote this question, there was only **5** or **6 RefSeq** mRNAs! The message of the question was to ensure you could see how to spot the **isoform 5a** transcripts (again!), not to list them! But, never mind, doing so was in fine tune with the ennui of the moment.

Additional Meanderings:

That really should not detain anyone but me? They belong after the consideration of masking the run of As for the **megablast** you ran. I just enjoyed this detour, so I keep it somewhere low profile. The whole journey leads nowhere of any note, so, should you decide to read, expect little!

The **500** base pairs of 3' flanking sequence added on to the **Ensembl** sequence for "good measure", is not part of the alignment (as would be expected). This can be seen easily if you look at the end of the alignment illustrated above, which is the alignment of the last exon of a transcript.

The entire length of this transcript is **6,732** base pairs.

Homo sapiens paired box 6 (PAX6), transcript variant 11, mRNA
Sequence ID: ref|NM_001310161.1| Length: 6732 Number of Matches: 11

The entire length of the genomic query sequence is **34,170** base pairs.

Query ID	lcl|Query_71179
Description	pax6-genomic sequence
Molecule type	nucleic acid
Query Length	34170

The alignment ends at position position **6,729** of the mRNA (**3** from the end) and position **33,670** of the genomic sequence (exactly **500** base pairs from the end).

```
Query  33601  ATTTGACATCCTGGCAAATCACTGTCATTGATTCAATTATTCTAATTCTGAATAAAAGCT  33660
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  6660   ATTTGACATCCTGGCAAATCACTGTCATTGATTCAATTATTCTAATTCTGAATAAAAGCT  6719

Query  33661  GTATACAGTA  33670
              ||||||||||
Sbjct  6720   GTATACAGTA  6729
```

The **3** missing base pairs of the mRNA are all **A**s, due to **polyadenylation**. Position **6,729** being recorded as a **polyA** site by **RefSeq**. A very short **polyA** tail surely? But there is no telling what stage of the mRNA is recorded on **RefSeq**. **Wikipedia** says:

```
polyA_site      2495
                /gene="PAX6"
                /gene_synonym="AN; AN2; D11S812E; FVH1; MGDA; WAGR"
```

```
    6541 aaaaaaatag aataagaaac ctgattttta gtactaatga aatagcgggt gacaaaatag
    6601 ttgtcttttt gattttgatc acaaaaaata aactggtagt gacaggatat gatggagaga
    6661 tttgacatcc tggcaaatca ctgtcattga ttcaattatt ctaattctga ataaaagctg
    6721 tatacagtaa aa
//
```

> "The tail is shortened over time, and, when it is short enough, the mRNA is enzymatically degraded."

Of course, the neatness of this observation does reflect less some profund biological truth than it does that this mRNA just happens to be one that extends furthest to the right in the genome, and there is no chance match between the **polyA** tail and the extra **500** bases of genomic sequence you added on when extracting it from **Ensembl**.

The journey was fun even though the destination was dubious. Much the way of a considerable portion of life in general one might reflect?

What are the **9** stronger matches around base position **16,000**?

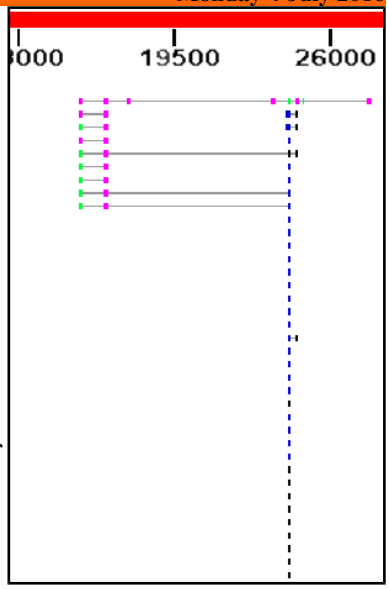
Matches between the regions of genomic DNA encoding **Paired Box** domains.

Why would you expect exactly **9** matches around this point?

Because that is how many **Paired box** domains are suggested to be in the human genome by counting the number of quality **mRNA** sequences in **RefSeq** claiming to include a **Paired box** coding region. There is **PAX6** plus its **8** paralogues, imaginatively all named:

PAX1, **PAX2**, **PAX3**, **PAX4**, **PAX5**, **PAX6**, **PAX7**, **PAX8** & **PAX9**

What do you make of the plethora of matches around **24,000**?



These are matches between human **mRNA** sequences with the regions of genomic DNA encoding **Homeo Box** domains. As you discovered from **Interpro**, there are many of these.

The thin line joining features implies that those features relate to the same database entry.

Notice that **4** of the **9** proteins matching a **Paired box** genomic region also match a **Homeo box** region. the remaining **5** do not. This implies that **4** of the **9** proteins corresponding to the hits detected here have a **Paired box** domain near the start of the protein and a **Homeo box** domain further along. This is exactly as was suggested by the **PROSITE** annotation you examined.

Why do you suppose the **Paired box** matches precede the **Homeobox** matches?

Because they score more highly and so, in the opinion of **blast**, are more worthy. Primarily, they score more highly because they are longer. The list is ranked by **E Value**. Good matches with long sequence are less likely to occur by chance than equally good matches with shorter sequences.

Possibly a more interesting question[34] might have been: "Why are not all the hits which include both domains at the top of the list?". Surely they should be, as they match over a longer proportion of the query sequence and so must, in general at least, be of the greatest significance.

They do not always come at the top of the list because **blast** scores each matching region individually and uses the ranking scores associated with the single region with the highest **E Value** to evaluate the similarity of the entire database entry with the query. This has to be a dubious practice surely? But, it appears to work, so why complain.

To justify this last assertion, Look at your top hit.

Description	Max score	Total score	Query cover	E value	Ident	Accession
RecName: Full=Paired box protein Pax-6; AltName: Full=Aniridia type II protein; AltName: Full=Oculorhombin	160	767	3%	2e-40	97%	P26367.2

E Val = 2e-40, **Max score = 160**, **Total score 767** associated with the whole of **P26367.2**

RecName: Full=Paired box protein Pax-6; AltName: Full=Aniridia type II protein; AltName: Full=Oculorhombin
Sequence ID: sp|P26367.2|PAX6_HUMAN Length: 422 Number of Matches: 8

Range 1: 46 to 123 GenPept Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Method	Identities	Positives	Gaps	Frame
160 bits(406)	2e-40	Compositional matrix adjust.	76/78(97%)	78/78(100%)	0/78(0%)	+3

```
Query  16680  MQVSNGCVSKILGRYYETGSIRPRAIGGSKPRVATPEVVSKIAQYKRECPSIFAWEIRDR  16859
              +QVSNGCVSKILGRYYETGSIRPRAIGGSKPRVATPEVVSKIAQYKRECPSIFAWEIRDR
Sbjct  46     LQVSNGCVSKILGRYYETGSIRPRAIGGSKPRVATPEVVSKIAQYKRECPSIFAWEIRDR  105

Query  16860  LLSEGVCTNDNIPSVSSL  16913
              LLSEGVCTNDNIPSVSS+
Sbjct  106    LLSEGVCTNDNIPSVSSI  123
```

Range 2: 254 to 305 GenPept Graphics ▼ Next Match ▲ Previous Match ▲ First Match

Score	Expect	Method	Identities	Positives	Gaps	Frame
81.3 bits(199)	5e-29	Compositional matrix adjust.	51/52(98%)	51/52(98%)	0/52(0%)	+3

```
Query  24654  FQVWFSNRRAKWRREEKLRNQRRQASNtpshipisssfstsVYQPIPQPTTP  24809
              QVWFSNRRAKWRREEKLRNQRRQASNTPSHIPISSSFSTSVYQPIPQPTTP
Sbjct  254    IQVWFSNRRAKWRREEKLRNQRRQASNTPSHIPISSSFSTSVYQPIPQPTTP  305
```

Range 3: 312 to 344 GenPept Graphics ▼ Next Match ▲ Previous Match ▲ First Match

Score	Expect	Method	Identities	Positives	Gaps	Frame
70.5 bits(171)	5e-29	Compositional matrix adjust.	33/33(100%)	33/33(100%)	0/33(0%)	+2

```
Query  24926  GSMLGRTDTALTNTYSALPPMPSFTMANNLPMQ  25024
              GSMLGRTDTALTNTYSALPPMPSFTMANNLPMQ
Sbjct  312    GSMLGRTDTALTNTYSALPPMPSFTMANNLPMQ  344
```

Now look at the first few individual regional alignments for this hit.

As you can see, the **E Value** and **Max score** values used to evaluate the whole protein were computed from just the best (ranked by **E Value**) local alignment! Crude, but never mind.

The **Total score** for the entire protein is the sum (rounded up to the nearest integer) of all the bit scores for all **8** local alignments computed for this protein (I suggest you just trust me on this assertion).

34 That I did not ask, because I only just thought of it.

How do you suppose the **Max matches in a query range** parameter might be of value if this order was reversed?

If **Paired boxes** had been more prolific, then the number of **Paired box** matches might have filled the **blast** hit list before the highest scoring **Homeo box** hit was registered.

If **Homeo boxes** were longer, and so justified a better **E value**, then the number of **Homeo box** matches might have filled the **blast** hit list before the highest scoring **Paired box** hit was registered.

Either of these situations would be very unfortunate, but easily avoided by setting the **Max matches in a query range** parameter to something sensible (**50** say). This would ensure that only the top **50** items in the **blast** hit list would be dominated by the strongest hit.

For further discussion of the parameter, see above.

How does this "non-informative" region match expectations suggested by **Prosite** and the **Feature table** of **Uniprot** for **PAX6_HUMAN**?

blast identifies two non-informative regions. I only discussed the prettiest one above. The region discussed is comprised largely of **Serines**, **Prolines**, **Threonines** & **Isoleucines** the **15** residues between **294-308**.

Score	Expect	Method	Identities	Positives	Gaps	Frame
81.3 bits(199)	5e-29	Compositional matrix adjust.	51/52(98%)	51/52(98%)	0/52(0%)	+3

```
Query  24654  FQVWFSNRRAKWRREEKLRNQRRQASNtpshipisssfstsVYQPIPQPTTP  24809
              QVWFSNRRAKWRREEKLRNQRRQASNTPSHIPISSSFSTSVYQPIPQPTTP
Sbjct  254    IQVWFSNRRAKWRREEKLRNQRRQASNTPSHIPISSSFSTSVYQPIPQPTTP  305
```

The second (to be found much further down your **blast Alignments** output) is comprised entirely of **Arginines**, **Luecines** and **Lysines** and **Glutamines**, the **10** residues between **203 - 212**.

Score	Expect	Method	Identities	Positives	Gaps	Frame
85.9 bits(211)	3e-16	Compositional matrix adjust.	56/66(85%)	58/66(87%)	5/66(7%)	+3

```
Query  23649  YHPILFVP-----DGCQQQEGGGENTNSISSNGEDSDEAQMrlqlkrklqrNRTSFTQEQ  23813
              ++P   VP     DGCQQQEGGGENTNSISSNGEDSDEAQMRLQLKRKLQRNRTSFTQEQ
Sbjct  162    WYPGTSVPGQPTQDGCQQQEGGGENTNSISSNGEDSDEAQMRLQLKRKLQRNRTSFTQEQ  221

Query  23814  IEALEK  23831
              IEALEK
Sbjct  222    IEALEK  227
```

Uniprotkb also suggests there are two **compositionally biased regions**.

Compositional bias	131 – 209	79	Gln/Gly-rich
Compositional bias	279 – 422	144	Pro/Ser/Thr-rich

Well, hardly an exact match, but there is approximate agreement? One would certainly suppose that **blast** is only willing to mask fairly severe cases of **compositional bias**. It is also probable that **blast** has a rather more mechanistic (i.e. non-biological) interpretation of what **computational bias** is?

PROSITE also predicts the more obvious region of **computational bias**, rather more generally:

> "An octapeptide and/or a homeodomain can occur C-terminal to the paired domain, as well as a Pro-Ser-Thr-rich C-terminus"

From your investigations of **Primer Design**

Do you think **10** primer pair suggestions is sufficient? If not, what number would you choose?

Until very recently, the default here was **5**. That seemed rather low to me. I included this question to solicit opinion rather than to impart knowledge. A default of **10** seems more in line with my instincts, but people who use this program seriously mostly tell me that they can select suitable primers from the first **2** or **3** suggestions of the program. So, **5** would seem a good choice and **10** would be moving towards cautiously overdoing things.

On the whole, informed opinion suggests that **10** suggestions will be more than enough in most circumstances.

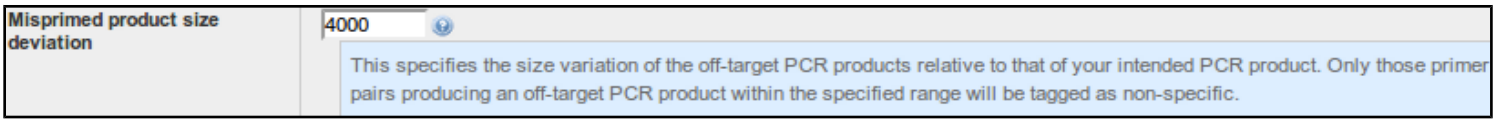
What value would you choose here if you were looking for uncluttered results?

Summary:

Clearly, the smaller the number chosen, the shorter will be the list of spurious products. However, pick something too small and you risk including unintended product(s) that could cause confusion. The size selected must be sufficient that larger unwanted PCR product(s) could easily be spotted by other means (simply by size?).

Full Answer:

Well, mostly for me, and just in case you were curious, when I first wrote the question, the parameter was very different and not so easy to understand. Pure self indulgence, I know, but here is the history. The parameter explained itself, via the button, thus:



I interpreted this to mean that only **blast** predicted products of up to **X+4,000** base pairs, where **X** base pairs is the length of the intended target, will be given any regard. It is thus assumed that a difference of **4,000** base pairs between an intended PCR product (predicted by **primer3**) and a spurious product (detected by **blast**) can easily be detected simply by size difference.

Of course this parameter also will reject unwanted **blast** predicted products that are less than **X–4,000** base pairs will be given any regard. Given the largest possible **primer3** suggestion will be **1,000** base pairs (the form setting for the exercise specifies products of between **100**[35] and **1,000** base pairs), this is hardly an issue here.

Comment upon the small default value for the **Blast word size**?

By default, **blast** will be looking for aligned exactly matching blocks of **7** nucleotides when identifying where a primer might match a database entry. The entire primer match with the template sequence does not have to be exact for the primer to be acceptable. The entire primer is typically only around **20** bases long. And word size much more that **7** would clearly miss too much to be effective.

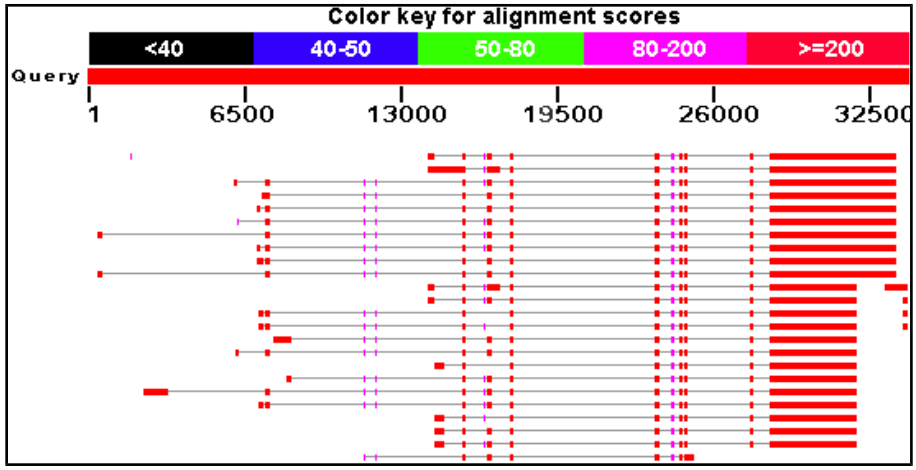
35 The form explicitly declares a minimum of **70**, but the ranges from which the **forward** & **reverse** primers must come (**15000-15700** & **15800-16500**) make the smallest possible **primer3** prediction **100** base pairs long.

Why do you suppose **blast** did not pick up all the transcripts?

Summary:

Well, the simple answer is that the transcripts that were not detected as unwanted products cannot include either the forward primer, or the reverse primer, or both. This is, almost, the only possible explanation.

Full Answer:



Of course, for this run, you did specify that you were not interested in products longer than **4,000** base pairs, so it could be that one or more products were possible but longer than that? I suspect this would only be feasible if there were retained introns involved, but previous **blast** results do not suggest this to be the case. I would say the only possible candidate for an over-length product might be the second hit down in the graphical representation generated previously by **blast**. The first and third exons from the left look a bit bloated, but not really sufficiently to cause a problem.

Description	Max score	Total score	Query cover	E value	Ident	Accession
Homo sapiens paired box 6 (PAX6), transcript variant 11, mRNA	9659	12484	19%	0.0	100%	NM_001310161.1
Homo sapiens paired box 6 (PAX6), transcript variant 10, mRNA	9659	15161	24%	0.0	100%	NM_001310160.1
Homo sapiens paired box 6 (PAX6), transcript variant 8, mRNA	9659	12929	20%	0.0	100%	NM_001310158.1
Homo sapiens paired box 6 (PAX6), transcript variant 7, mRNA	9659	12729	20%	0.0	100%	NM_001258465.1
Homo sapiens paired box 6 (PAX6), transcript variant 6, mRNA	9659	12761	20%	0.0	100%	NM_001258464.1
Homo sapiens paired box 6 (PAX6), transcript variant 5, mRNA	9659	12737	20%	0.0	100%	NM_001258463.1
Homo sapiens paired box 6 (PAX6), transcript variant 4, mRNA	9659	12862	20%	0.0	100%	NM_001258462.1
Homo sapiens paired box 6 (PAX6), transcript variant 2, mRNA	9659	12833	20%	0.0	100%	NM_001604.5
Homo sapiens paired box 6 (PAX6), transcript variant 1, mRNA	9659	12942	20%	0.0	100%	NM_000280.4
Homo sapiens paired box 6 (PAX6), transcript variant 3, mRNA	9659	12791	20%	0.0	100%	NM_001127612.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X13, mRNA	6613	10063	15%	0.0	100%	XM_005252958.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X12, mRNA	6613	9439	14%	0.0	100%	XM_011520153.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X11, mRNA	6613	9329	14%	0.0	100%	XM_006718246.2
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X10, mRNA	6613	9410	14%	0.0	100%	XM_011520152.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X9, mRNA	6613	10507	16%	0.0	100%	XM_005252956.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X8, mRNA	6613	9783	15%	0.0	100%	XM_005252955.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X7, mRNA	6613	9091	14%	0.0	100%	XM_011520151.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X6, mRNA	6613	9637	15%	0.0	100%	XM_011520150.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X5, mRNA	6613	11324	17%	0.0	100%	XM_011520149.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X4, mRNA	6613	9814	15%	0.0	100%	XM_005252954.3
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X3, mRNA	6613	9172	14%	0.0	100%	XM_011520148.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X2, mRNA	6613	9502	15%	0.0	100%	XM_011520147.1
PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X1, mRNA	6613	9576	15%	0.0	100%	XM_011520146.1
PREDICTED: Homo sapiens elongator acetyltransferase complex subunit 4 (ELP4), transc	1775	1775	2%	0.0	100%	XM_005252865.2
Homo sapiens paired box 6 (PAX6), transcript variant 9, mRNA	647	2630	4%	0.0	100%	NM_001310159.1

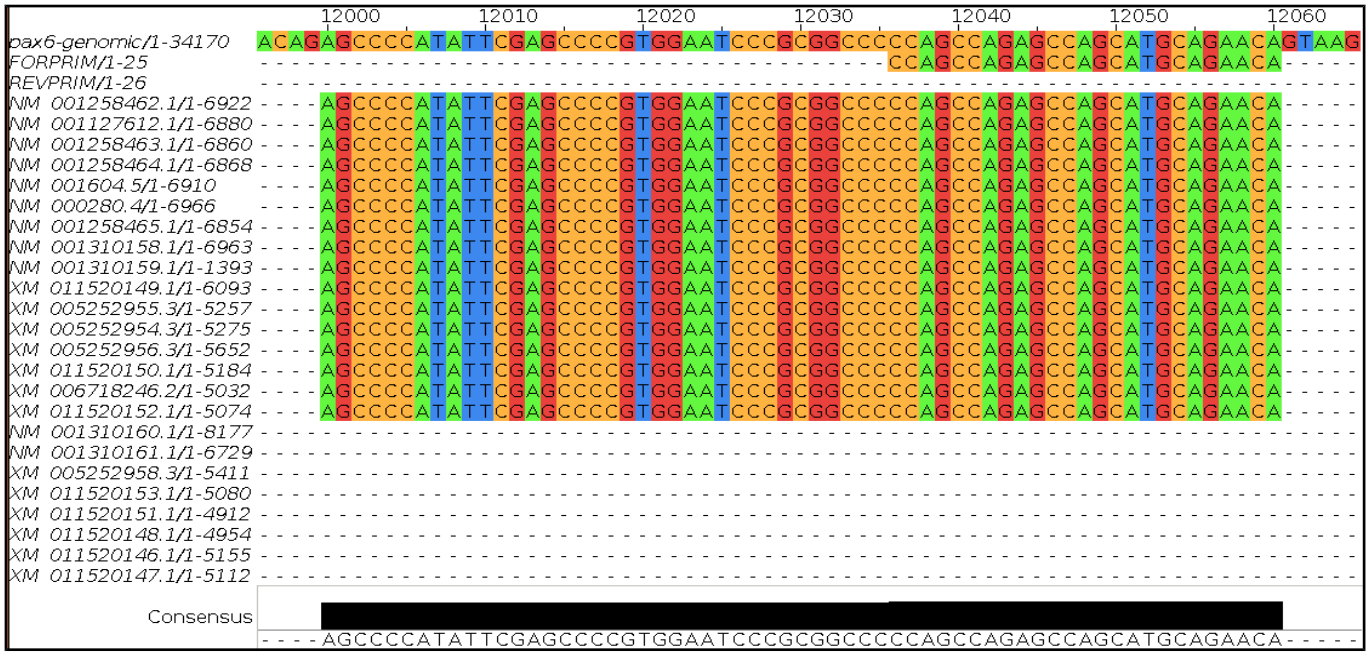
It might also be that unwanted PCR products are eliminated/introduced due to variations in the predicted transcripts. However, this can be ruled out as previous experiments, **blast** assures us that all **24** potential transcripts match the genomic sequence exactly.

Enough! Only because I want to, I will compute the alignments to prove the missing primer matches. Read no further unless you are truly in the mood. Much of the reason for recording the rest of this answer is that, apart from enjoying the pursuit of irrelevant detail, I also wanted to remember how I made the alignments and certainly feel I could have made both these, and my point much more simply? Suggestions welcome.

OK, I started by computing an alignment that was a mapping of all **24** transcripts onto the **PAX6** genomic regions as represented in the file **pax6_genomic.fasta**. I used a program called **gmap**, which like **spline** (used in the exercise) is designed to align cDNA/mRNA sequences with corresponding genomic sequences. The version of **gmap** I used runs under **linux** from the command line. It has the advantage over **spline** that is will align more than one cDNA/mRNA sequence against the genome in one run. Unfortunately, it does not generate an output format that can be easily displayed in the way I required here. I did try to persuade a couple of general multiple alignment programs (**clustalw** & **muscle**) to make me a usable alignment, but ran into the same difficulties we experienced in the exercise. I failed to find gap penalties that would get the programs to gap the larger introns. Even if I had succeeded to get the gaps in the right place, I would not have believed them to be placed with sufficient accuracy for the same reasons this was not possible when we tried the same trick with general alignment software for just one cDNA sequence against the genome in the exercise.

So, I made a rough alignment with **clustalw** and edited it to exactly what was suggested by **gmap** using **jalview**. This took **HOURS**. There has to be a better way!! You have already used all the software mentioned except **gmap** and **clustalw**. You will use **clustalw** and see how **jalview** can be use to edit, as well as just view, alignments a little later.

All that effort to show that the region around the forward primer looks like this:

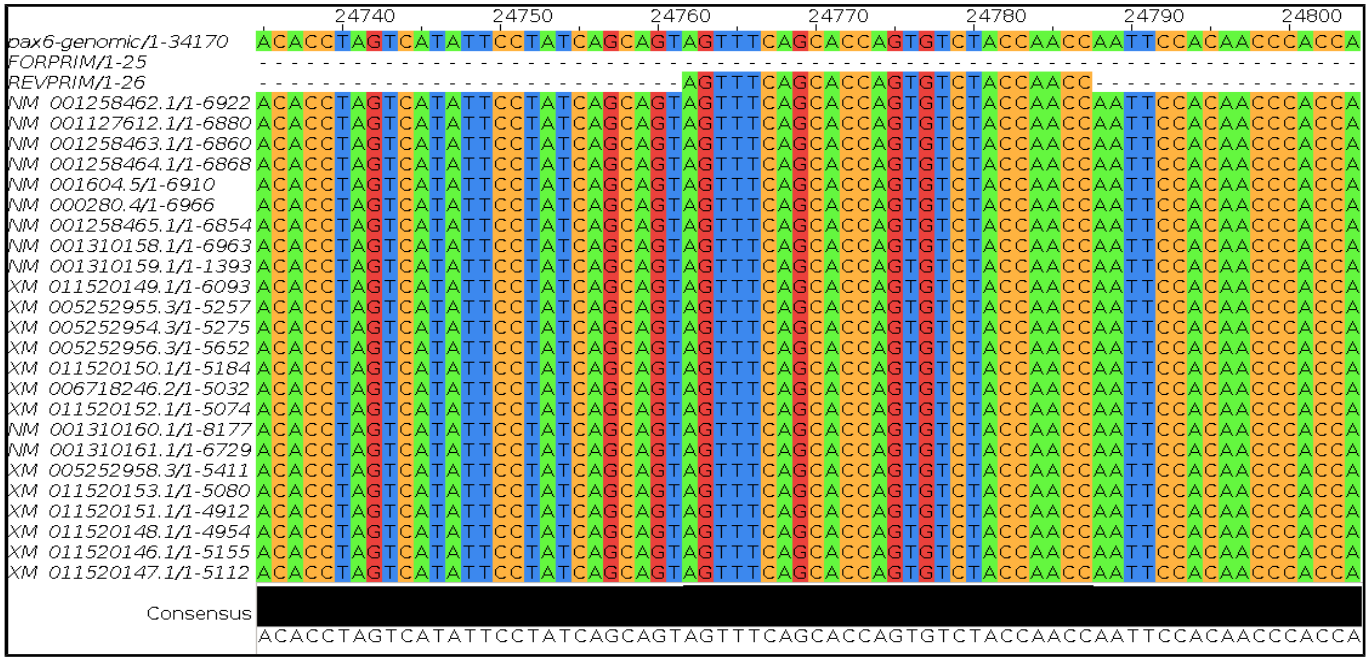


Showing clearly that the **8** transcripts:

NM_001310160.1
NM_001310161.1
XM_005252958.3
XM_011520153.1
XM_011520151.1
XM_011520148.1
XM_011520146.1
XM_011520147.1

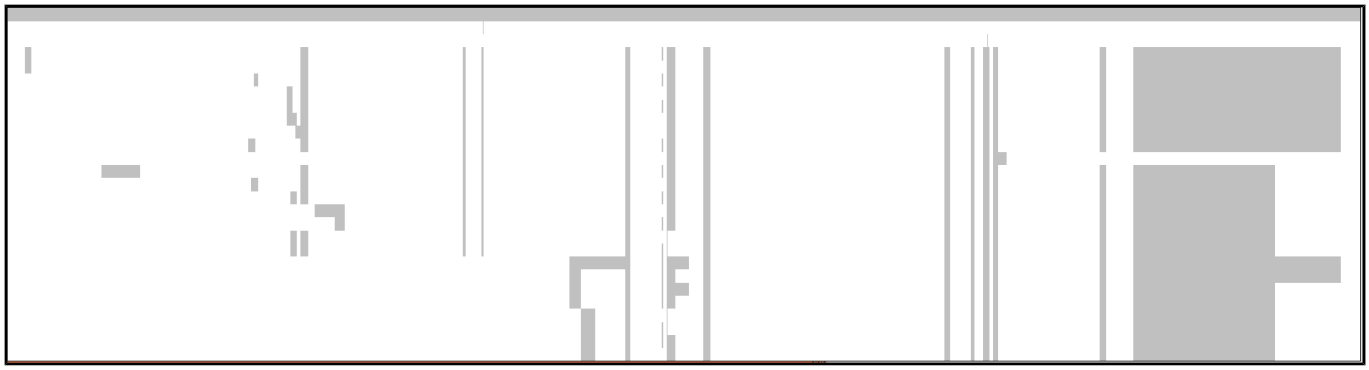
Have the exon that includes the forward primer spliced out and so will not produce any PCR product. Feel free to check this by comparing the textual results of your **blast** of the genomic sequence against the **RefSeq** mRNAs and the results of **PRIMER-BLAST**. I did, it was lots and lots of fun and I ended up content that all was logically consistent.

The alignment around the reverse primer looks like this:



All **24** putative transcripts match the reverse primer perfectly. So blast should indeed find **16** of the **24** transcripts sequences in **RefSeq**, and it does.

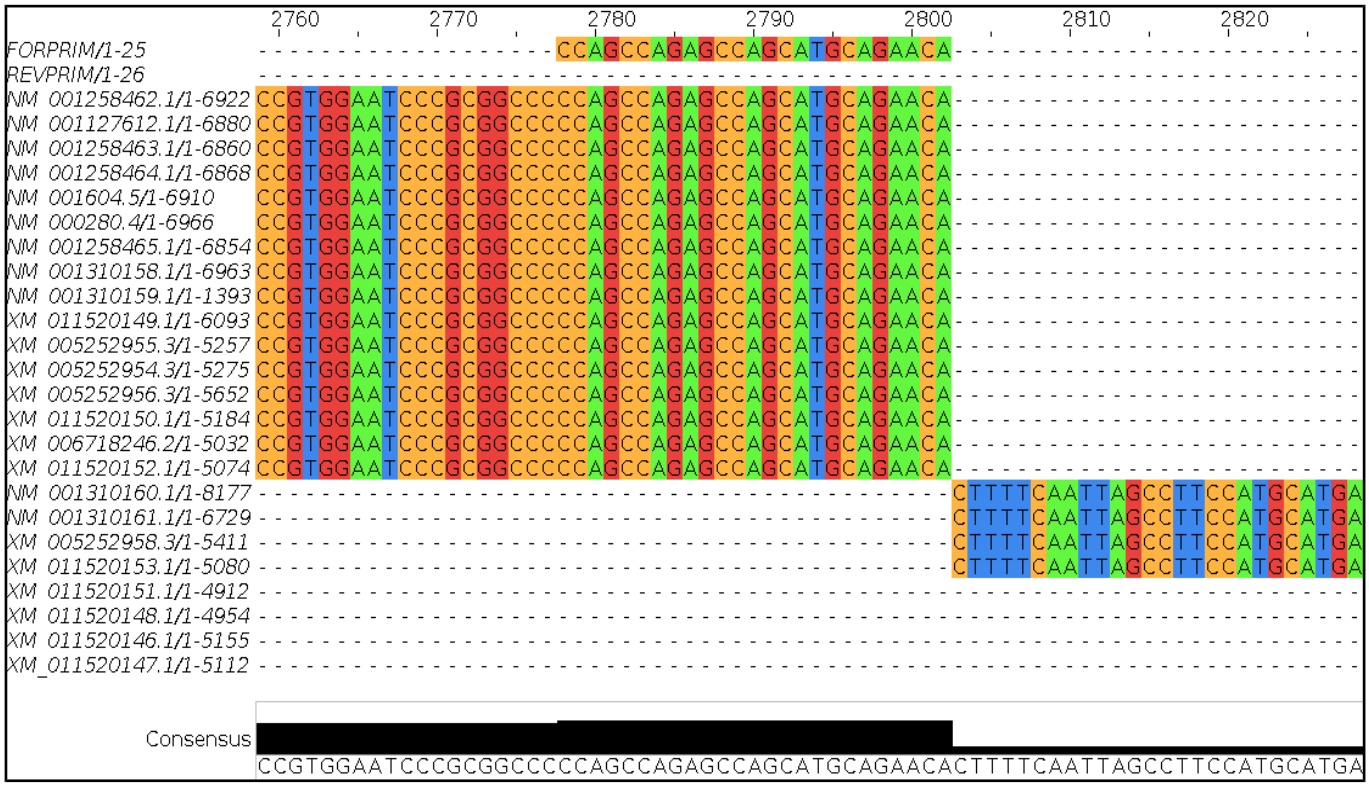
Jalview offers an overview of the entire alignment. The top row shows the genomic sequence. The second row shows the position of the forward primer. The third row shows the position of the reverse primer.



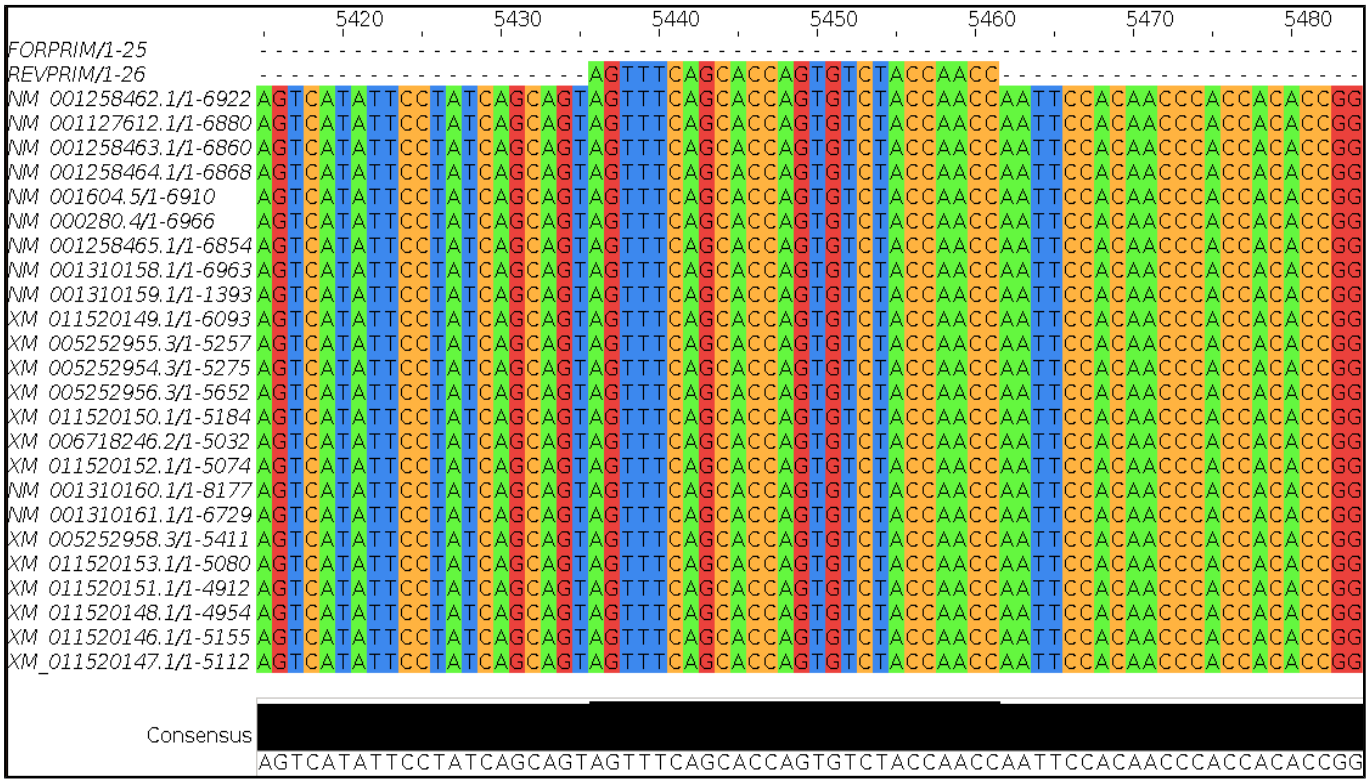
Except for the order of the transcripts, this view is very similar to the overview graphic generated by **blast**. The transcripts missing the forward primer, which isoform each transcripts represents and the fact that all transcripts match the reverse primer should be very clear.

Finished Dave? Well no, not quite. I wondered why I had included the genomic sequence in my alignment. Finding no answer to that question, I tried to make an alignment of just the primer sequences and the mRNAs. I thought this would be easy. I was wrong. The general programs are still going to get the gaps wrong whatever penalties are used. Some transcripts have exons entirely missing in all other transcripts leaving no clues as to which way round they should be aligned. The scaffold provided by the genomic sequence was essential. So, I made an mRNA only alignment by editing the alignment discussed above with **jalview**. This was easy (although you would not think so given the time it took me to work out how to do it!). I loaded the alignment into **jalview**, deleted the genomic sequence and then removed all empty columns (that is, all columns with no bases in them due the the removal of the genomic sequence). Clever eh? Just because it is there, here are the pictures.

Forward primer region (the primer is right at the end of an exon):

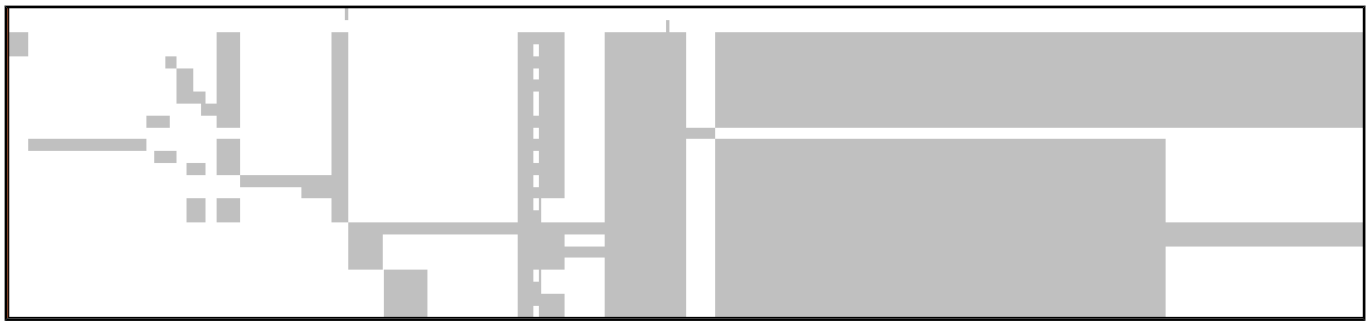


Reverse primer region:



Overview:

Without the evidence of the genomic sequence, the two leftmost exons could logically swap position. There is no transcript that includes both these exons and no overlap between either and any other exon in any transcript (most clearly verified from the previous **Overview** plot). Thus, there is no exon evidence of the order in which the two should appear.



Now I am done! This has to be the most over the top answer yet, but at least it kept me out of trouble for a while.

How would you tell quickly which isoform was represented by each mRNA listed here?

Summary:

All the mRNAs reported were of length **908**, **950**, **707** or **749**.

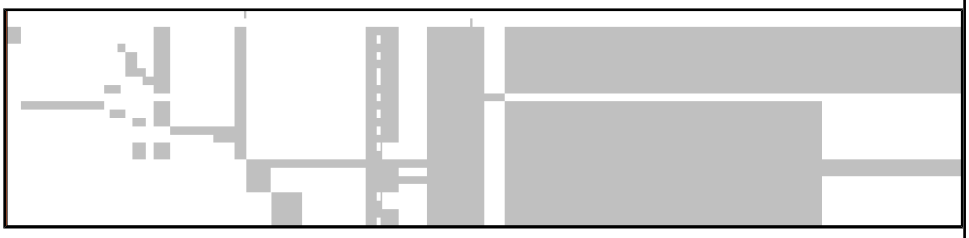
A reasonable guess might be based on the length of the products? All those that are **908** bases might be assume to produce the **422** amino acid **canonical isoform**. All those that are **950** (i.e. **42** base pairs longer) might be assumed to **436** produce amino acid **isoform 5a** proteins (i.e. **14** amino acids longer).

Analogous reasoning might be applied to the mRNAs that are either **707** or **749** base pairs in length.

Just a guess of course, but one I would be happy to have faith in. To be certain, one would need to read the annotations of each listed **RefSeq** entry!

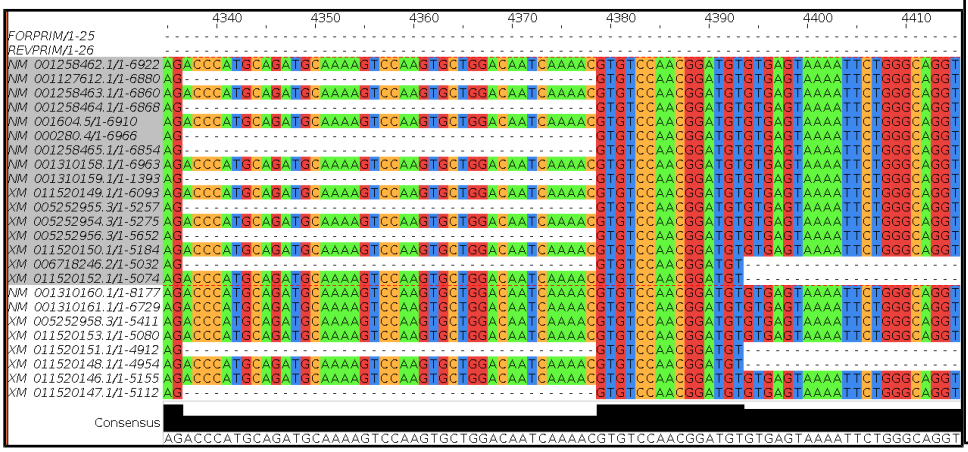
Full Answer:

From the illustrations of the last "**Full Answer**" (in particular, the **jalview** overviews), it is clear that all the mRNAs that produce a product include the region that determines which isoform is represented. That is, all are one isoform or the other.



The last two of the mRNAs that produce a PCR product, have a bit chewed out just after the isoform defining region (an exon spliced out, if you prefer). It is logical to suppose these would be the mRNAs from which the two shorter products were generated.

Indeed, looking at the relevant part of the mRNA only alignment shows them to be **XM_006718246** (product length **707**, excluding the **isoform 5a** exon that suggest it codes for a **canonical** protein) and **XM_011520152** (product length **749**, including the extra **42** base pairs suggesting it codes for an **isoform 5a** protein).



Products on potentially unintended templates

>NM_001310159.1 Homo sapiens paired box 6 (PAX6), transcript variant 9, mRNA

product length = 908
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 114 138

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1021 996

>NM_001310158.1 Homo sapiens paired box 6 (PAX6), transcript variant 8, mRNA

product length = 950
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 496 520

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1445 1420

>XM_006718246.2 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X11, mRNA

product length = 707
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 457 481

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1163 1138

>XM_011520152.1 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X10, mRNA

product length = 749
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 457 481

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1205 1180

>XM_005252956.3 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X9, mRNA

product length = 908
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 876 900

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1783 1758

>XM_005252955.3 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X8, mRNA

product length = 908
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 481 505

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1388 1363

>XM_011520150.1 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X6, mRNA

product length = 950
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 366 390

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1315 1290

>XM_011520149.1 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X5, mRNA

product length = 950
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 1275 1299

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 2224 2199

>XM_005252954.3 PREDICTED: Homo sapiens paired box 6 (PAX6), transcript variant X4, mRNA

product length = 950
Forward primer 1 CCAGCCAGAGCCAGCATGCAGAACA 25
Template 457 481

Reverse primer 1 GGTTGGTAGACACTGGTGCTGAAACT 26
Template 1406 1381

All other transcripts that generate PCR products generate products of length either **908** or **950**. Given the difference (**42** base pairs) is exactly the size of the **isoform 5a** exon, it is reasonable to assume the transcripts generating PCR products of length **908** represent **canonical** proteins, whereas the transcripts generating PCR products of length **950** represent **isoform 5a** proteins.

Prettier? True, and I submit including sufficient evidence to be more that just a guess now.

Is the number of "**potentially unintended products**" as you would you expect, given the evidence from **GeneCards**, **Ensembl** and **blast**?

Yes, I think so, given you accept my investigation (see above) as to why there were only **16** "**potentially unintended products**" when you might have expected **24**, given your **blast** results. **GeneCards** now encourages an initial expectation of **24** "**potentially unintended products**". **Ensembl** only uses the higher quality **RefSeq** mRNAs. Currently, **Ensembl** uses **10** of the **11** good quality **RefSeq** mRNAs to make its transcripts predictions. Close enough?

For all the "**potentially unintended products**", the selected primers match exactly. Can you explain this?

Well, of course they do??? All the transcripts found are generated from the same region of genomic DNA and therefore will be identical in all shared regions, including the primer regions. I suppose, in other instances, it would be possible to have transcripts with variation in the regions matching the primers insufficient to stop the primers working? But not in this case.

One might conclude there are no genuinely "unintended" products? All are real **PAX6** transcripts of varying certainty. A genuine unintended product would come from an entirely different part of the genome and would not necessarily match exactly with respect to the primers. They would just need to be "good enough to work".

The "**potentially unintended products**" are of different sizes. Can you explain the difference between the possible product lengths?
Are the numbers of "**potentially unintended products**" of each possible length consistent with your **blast** results?

Yes yes yes! I think both these questions made a bit more sense a few generations of these notes ago. We have already answered them sufficiently I suggest. I refer you to the answers above.

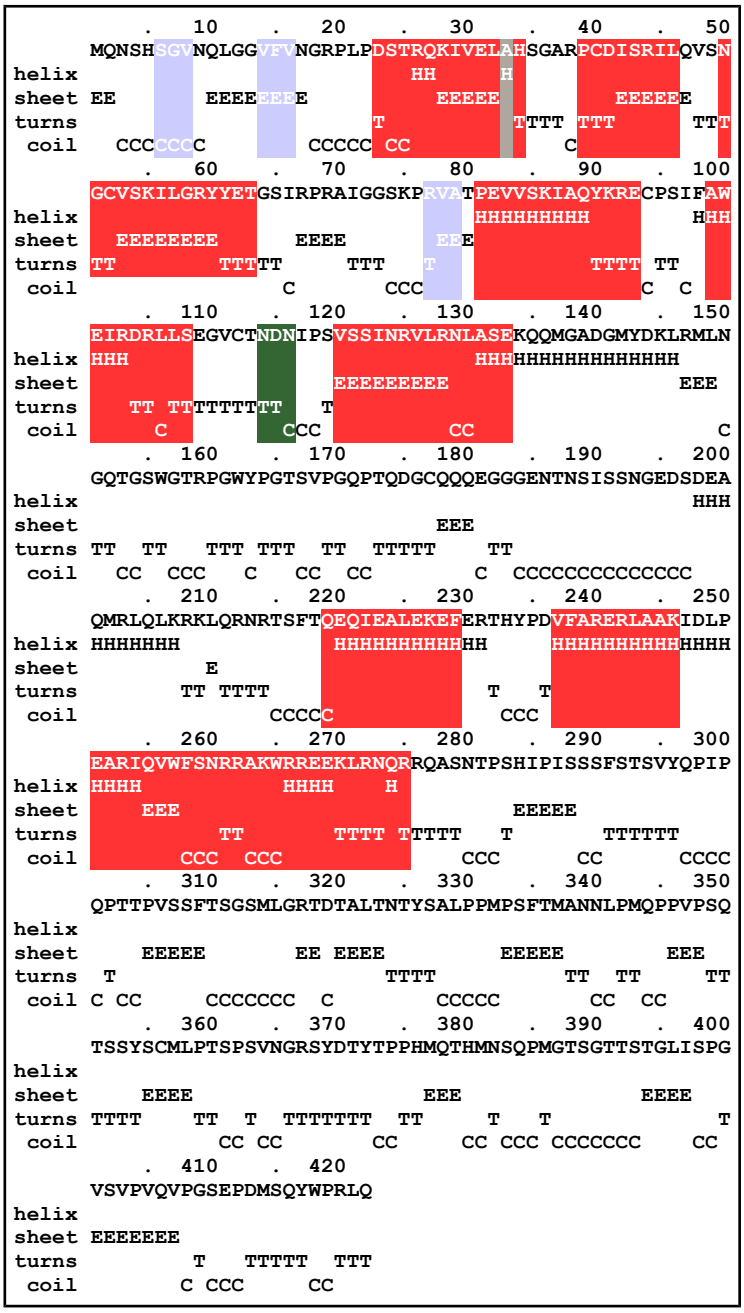
From your investigations of **Protein Secondary Structure Prediction with GOR I**

How credible would you say was the prediction at amino acid 33?

The original **GOR** predicts a Helix of length **1** at position **33**! This is just daft! A single amino acid cannot form a helix all by itself, it must have a few friends to make a believable Helix … or Beta Sheet, come to that.

Later versions of **GOR** made such silly predictions impossible, preferring to take the second best possibility for all insane predictions. In this case, positions **26**, **27** & **33** would all probably be predicted as Beta Sheet (**E** – for **E**xtended). Wrong (according to **UniprotKB**) but more logical.

Position **33** is, as you have discovered, the amino acid that, when mutated from an **A**lanine to a **P**roline, is the major cause of **Aniridia**. As you will confirm later, it is at the end of a Helix critical to to the DNA binding properties of the protein. **A**lanines are fine in Helices. **P**rolines are not.



How would you rate the prediction overall?

Beta strand	6 - 8	3
Beta strand	14 - 16	3
Helix	23 - 34	12
Helix	39 - 46	8
Helix	50 - 63	14
Beta strand	77 - 79	3
Helix	81 - 93	13
Helix	99 - 108	10
Turn	114 - 116	3
Helix	120 - 133	14
Helix	219 - 229	11
Helix	237 - 246	10
Helix	251 - 275	25

Referring to the structure according to **UniProtKB**, I colour the Helices Red and the Beta Sheets Blue. Position **33** I colour Grey. I do not believe the turn predictions, there are far too many! However, as the one predicted does, almost, match the **UniProtKB** assertion, I will highlight it with Green.

The **C**oils are boring, so I do not colour them at all!

Well, what do you think? It might reach the **60%** or so accuracy claimed by the version of **GOR**, but what does that really mean? There are only four possibilities so **25%** "accuracy" can be expected by chance. Add a small insight into the relative abundance of each possibility and you could improve on that before doing anything clever.

My, intuitive only, impression is that the program has found some useful insight into the problem, but I cannot say I am overly impressed. Of course, this is just one example. It is unreasonable to make harsh general judgement based on such light evidence. This is, after all, the crudest implementation of a simple method and I have seen it do much better with more compliant proteins.

From your investigations of **Protein Secondary Structure Prediction with GOR IV**

How does the prediction at position **33** compare with that of **garnier**?

Well, at least there is no attempt to suggest a one amino acid helix in this case! **GOR IV**, with justice, does not regard this as possible. Such silly predictions suggested by the arithmetic are rejected and replaced by more sensible alternatives computed by considering the likelihoods computed for other possibilities.

In a nutshell, **GOR IV** predicts position **33** to be at the end of a helix whose existence is supported reasonably well by **UniProtKB**. This is good! **GOR I** predicts position **33** to be a helix of length **1** amino acid in a region weakly suggested to have some sort of structure. This is weak, to say the least.

How would you rate the prediction overall?

I have coloured up the **GOR IV** results in the same way as I did for the **GOR I** results discussed above. Colouring the turn was unnecessary as **GOR IV** does not even try to predict turns.

This prediction is clearly loads better than the original **GOR** managed. In particular:

- The region after the last helix (position **280** or so onwards) is, give or take a few spurious attempts at Beta Sheets, predicted accurately as unstructured (i.e. Coil).
- The predictions for the Beta sheets are slightly worse that **GOR I** managed. Both programs got the second of the three and missed the first entirely, but **GOR I** did pick two of the three positions for the third Beta sheet, **GOR IV** missed it altogether!

```
        10        20        30        40        50        60
70
         |         |         |         |         |         |
|
MQNSHSGVNQLGGVFVNGRPLPDSTRQKIVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIRP
RA
ccccccccccccceeecccccccchhhhhhhhccccccceeeeeeecccceeeeeccceecccccccc
ee
IGGSKPRVATPEVVSKIAQYKRECPSIFAWEIRDRLLSEGVCTNDNIPSVSSINRVLRNLASEKQQMG
AD
ccccccccccchhhhhhhhhccccceeehhhhhceeccccccccccccccchhhhhhhhhhhhhhhhh
hh
GMYDKLRMLNGQTGSWGTRPGWYPGTSVPGQPTQDGCQQQEGGGENTNSISSNGEDSDEAQMRLQLKR
KL
chhhheeeecccccccccccccccccccccccccccceecccccccceeeecccccchhhhhhhhhhhh
hh
QRNRTSFTQEQIEALEKEFERTHYPDVFARERLAAKIDLPEARIQVWFSNRRAKWRREEKLRNQRRQA
SN
hhccccchhhhhhhhhhhhcccccccchhhhhhhhhhhcccccchhhhhhcchhhhhhhhhhhhhhhcc
cc
TPSHIPISSSFSTSVYQPIPQPTTPVSSFTSGSMLGRTDTALTNTYSALPPMPSFTMANNLPMQPPVP
SQ
ccccccccccccccccccccccccccceeecccccccccceecccccccccceecccccccccccccc
cc
TSSYSCMLPTSPSVNGRSYDTYTPPHMQTHMNSQPMGTSGTTSTGLISPGVSVPVQVPGSEPDMSQYW
PR
cceeeeeecccccccccceecccccceeecceecccccccccceeeecccccceeecccccccccccc
ee
LQ
ec
```

- Neither program got the helices quite right, but **GOR IV** was much closer to the right answer (according to **UniProtKB**). The two helices **GOR IV** missed altogether were predicted as largely structured (i.e. Beta Sheet instead of Helix). Generally, **GOR** is pretty reliable at picking structure, but not always the right structure. Knowing this, you could claim that predicting the helices as Beta Sheets beats predicting them as Coil.

I conclude, this is a usefully accurate prediction (as long as you do not take it ***TOO*** seriously) and certainly a big improvement on **GOR I**.

From your investigations of **Protein Secondary Structure Prediction with Jpred**

Some Notes on colouring the MSA generated by Jpred:

(Click here to return to the Instructions.)

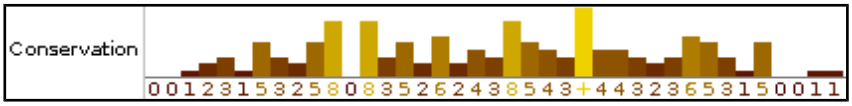
Mostly to remind me why and how I decided to colour the **MSA** as I did. My objective was purely to make obvious where the family of proteins were meaningfully similar. If you are happy that this objective was achieved, it is probably best to read no further.

I discovered most of what follows by Selecting the **Help** (easiest way is to press **F1** key, otherwise there is a pull down option at the top of the display, choose **Documentation** option) and searching for "**conservation**". From the list of hits, I first selected "**Alignment Conservation Annotation**". There it says:

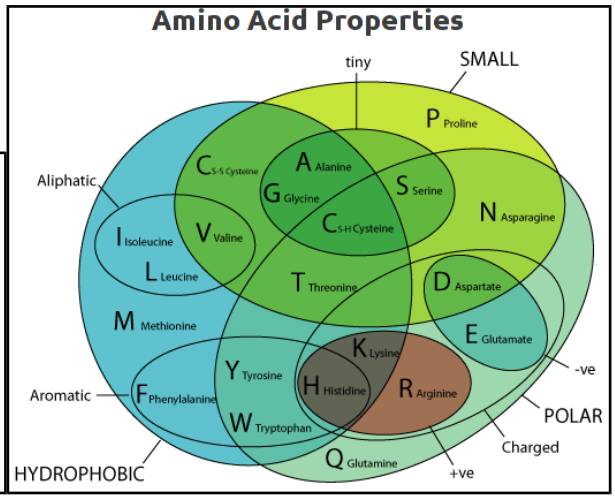
> "**Conservation** is measured as a numerical index reflecting the **conservation** of physico-chemical properties in the alignment: Identities score highest, and the next most conserved group contain substitutions to amino acids lying in the same physico-chemical class.
>
> **Conservation** is visualised on the alignment or a sequence group as a histogram giving the score for each column. Conserved columns are indicated by '*' (score of **11** with default amino acid property grouping), and columns with mutations where all properties are conserved are marked with a '+' (score of **10**, indicating all properties are conserved).
>
> Mousing over a conservation histogram reveals a tooltip which contains a series of symbols corresponding to the physico-chemical properties that are conserved amongst the amino acids observed at each position. In these tooltips, the presence of ! implies that the lack of a particular physico-chemical property is conserved (e.g. !proline)."

I think to understand the detail of the scoring, one would have to read the paper quoted in the **Help**. I think I will leave that until another day! For now, I just make a few notes.

- The numbers under the histogram columns appear to represent simply the number of phsico-chemical properties considered to be conserved. At least, this is consistently true for this example, shown by hovering the mouse over the histogram columns.

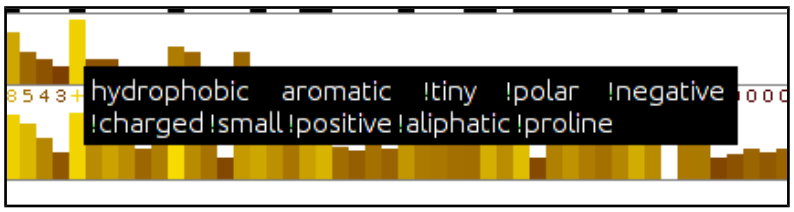


- **Jalview** admits to exactly **10** phsico-chemical properties that must be one of "**Not conserved**", "**positively conserved**" or "**negatively conserved**".




```
ILVCAGMFYWHKREQDNSTPBZX-
XXXXXXXXXXX·····X···XX Hydrophobic
·····XXXXXXXXXX·XXXXX Polar
··XXXX·····XXXXX··XX Small
·····X··XX Proline
····XX·····X····XX Tiny
XXX·····XX Aliphatic
·····XXXX·····XX Aromatic
·····XXX·····XX Positive
·····X·X·····XX Negative
·····XXXX·X·····XX Charged
```

- The column achieving a "+" has all **10** conserved phsico-chemical properties either **positively** or **negatively conserved**. It is a highly, but not completely **conserved** "**F**". This would appear to agree with the **Help**? There is no example of a **100%** conserved column in this example. If there was, I would expect it would be represented by a "*" representing a score of **11**.

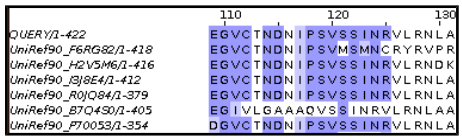


- Conservation of any given property does not have to be **100%** and gaps are tolerated. Reasonable as to be too exacting would eliminating. I expect the details are explained in the original paper. I justify this statement, unnecessarily, by claiming there are both gaps and a **Proline** in the column represented by a "+".
- I am still uncertain about the difference between a "**0**" column and a "–" column? I decide to believe they are both columns were there is no measurable conservation, but "**0**" columns are in regions where they are surrounded by significant conservation? One day, I will read the paper.
- By observation, it can be seen that "conservation" is measured relative to the consensus sequence rather than the query sequence. This seems a reasonable choice to me.

Well that was fun? Now I write some instructions to turn the nasty bland alignment into one that glows blue.

Click here to return to the Instructions.

What protein database has **Jpred** chosen to search for protein sequences for the alignment upon which its predictions will be based?



The database **Jpred** instructed **PSI-blast** to use to seek proteins homologous to the **PAX6** query can be determined by looking at the sequence identifiers displayed down the left hand side of the alignment in **Jalview**. The identifiers are constructed from the name of the database and the entry identifier separated by an underline character. So the database is the **UniRef90** cluster database built from the **UniProtKB** database.

The **UniRef** cluster databases comprise entries that are not individual protein sequences, but cluster of similar sequences. In the case of the **UniRef90** database, each entry includes all sequences **90%** identical to a given seed sequence. A representative sequence is elected as the only one of the cluster to be considered by such as **PSI-blast**, but clearly, a hit with any representative sequence implies significant similarity with all the sequences of its cluster.

I offer a supplementary exercise to investigate these cluster databases for those to whom they might be of particular interest.

Why do you suppose this database was used in preference to, say **UniprotKB**?

The reason **Jpred** runs **PSI-blast** is to identify sequences representing as wide a family of proteins as possible, to which a **Query** sequence belongs. For the purpose of structure prediction, there is little value in this collection including many sequences that are essentially identical. A wide variety of sequences, as long as they still are likely to be homologous, is of far greater value than a huge number of sequences. Using a **UniRef** database allows that only the **Representative** sequence of each cluster of very similar sequences will be recognised and aligned by **PSI-blast**. This allows the **PSI-blast MSA** to include an extensive range of variation without being bloated by sequences too similar to be individually interesting.

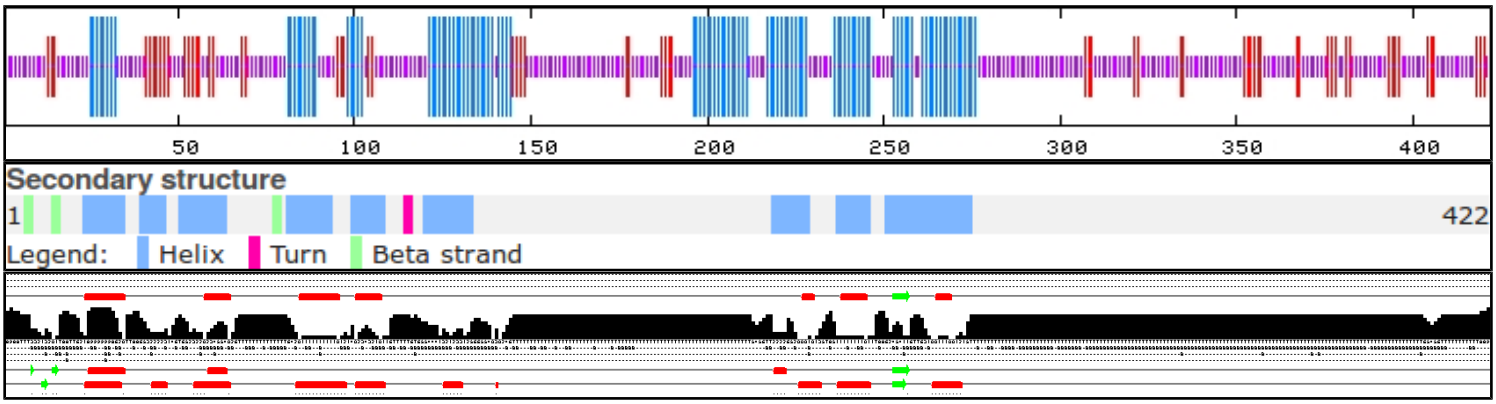
How would you rate the **Jpred** prediction overall?

Well, frankly, not as wonderful as I was expecting. Better than **GOR IV** (investigated in a supplementary exercise), but still leaves room for improvement? **jnetpred** (essentially the answer) is reasonable. It misses a couple of helices including one that **GOR IV** also overlooks. However, it has considerably less false positive prediction tendencies than **GOR IV**. The **JNETHMM** predictions are particularly poor, saved by the much more accurate deliberations of **JNETPSSM**.

JNETHMM is a prediction computed from the **H**idden **M**arkov **M**odel (**HMM**) representation of the final **PSI-blast MSA**.

JNETPSSM is a prediction computed from the **P**osition **S**pecific **S**coring **M**atrix (**PSSM**) representation of the final **PSI-blast MSA**. **PSI-blast** uses **PSSM**s of the **MSA** of each iteration of its search as a **Query** for the next iteration.

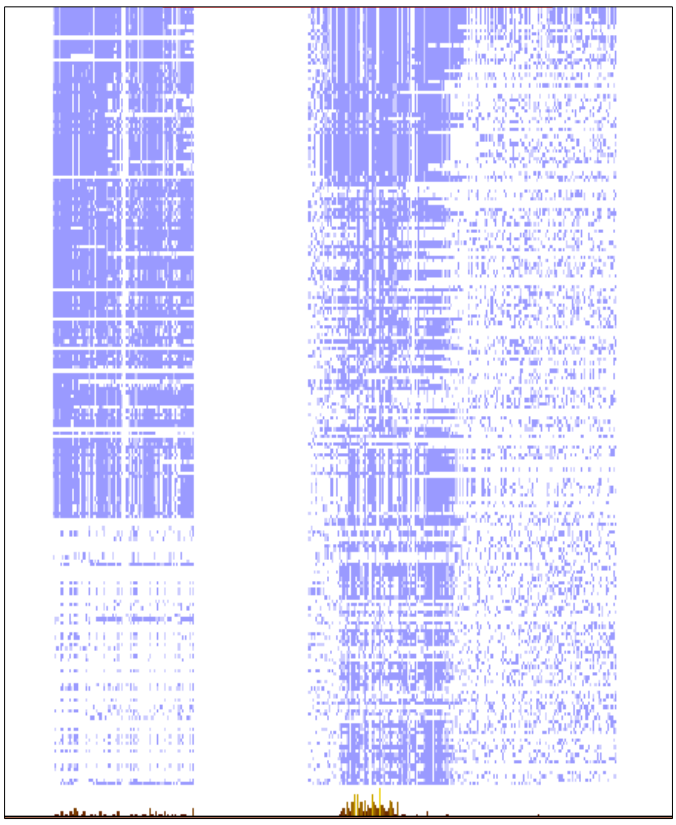
The **jnetpred** prediction is effectively the consensus of the predictions of **JNETHMM** and **JNETPSSM**.



Here I have aligned the **GOR IV** and **Jpred predictions** with the secondary structure as recorded by **UniProtKB**.

So, can the prediction be improved? **Jpred** is better than this result suggests!

On reflection, maybe just throwing in the entire sequence of **PAX6_HUMAN** and hoping for the best was a little crude? Our protein has two major domains whose secondary structure one might expect to be conserved. **PSI-blast** will gather together a mountain of sequences that have one, or the other, or both of the domains and try to align them as if they were homologous over their entire length (a **global alignment**). ***BUT***, they are not all globally homologous! This means that the alignment of both the domains regions will be polluted by sequence that represent proteins that do not include that domain. This must substantially reduce the quality of the prediction?



This phenomena can be illustrated by choosing to view the **Jalview Overview Window** (available from the **View** pull down menu).

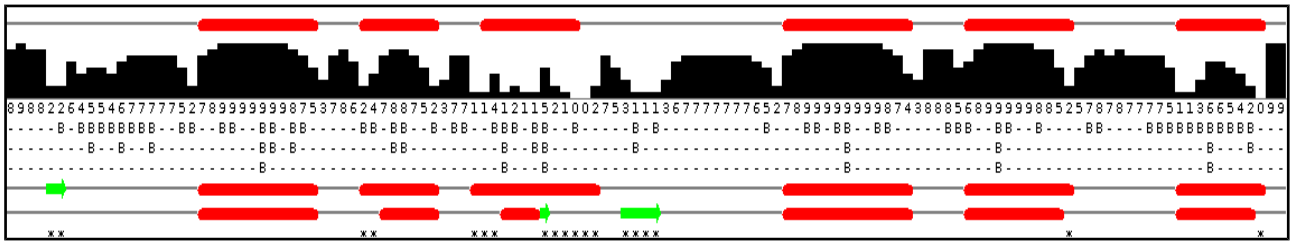
The wider column of blueness at the start of the alignment represents the **paired box** domains. The picture suggests about one third of the aligned sequences do not have a **paired box** domain, but those sequences will have unrelated sequence in that region that will reduce the degree to which the alignment represents the properties of a **paired box** and so also the likelihood of a sensible structure prediction[36].

The problem for the more common **homebox** domain looks less severe, however, the alignment clearly includes many sequences that do not look to have a **homeobox** domain.

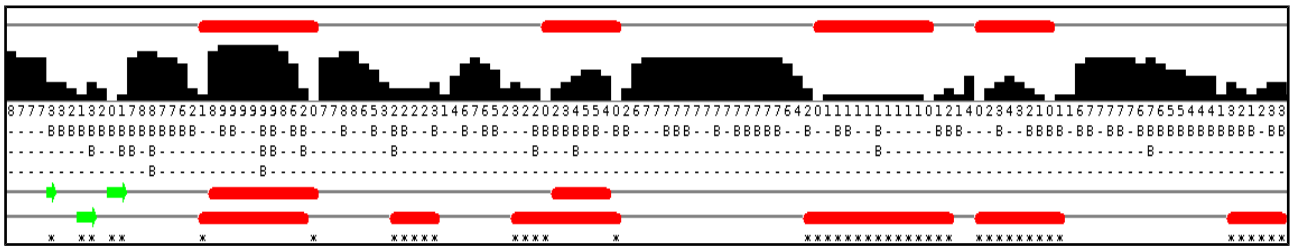
So, what to do? I suggest the two domains might be investigated separately? Why not run **Jpred** twice, once with just the **PAX6_HUMAN** paired box region and then again with just the **homebox** region.

I have done this for you and will now show you the results, however, should you wish to try it yourself, you already have the isolated sequence of both domains saved in local files. The sequence of the **paired box** region should be in a file called **pax_domain.fasta**. The **homeobox** sequence should be in a file called **homeobox_domain.fasta**. Run **Jpred** again with each sequence and you should get results very similar to mine.

First the new **paired box** prediction (top) compared to the original (bottom).



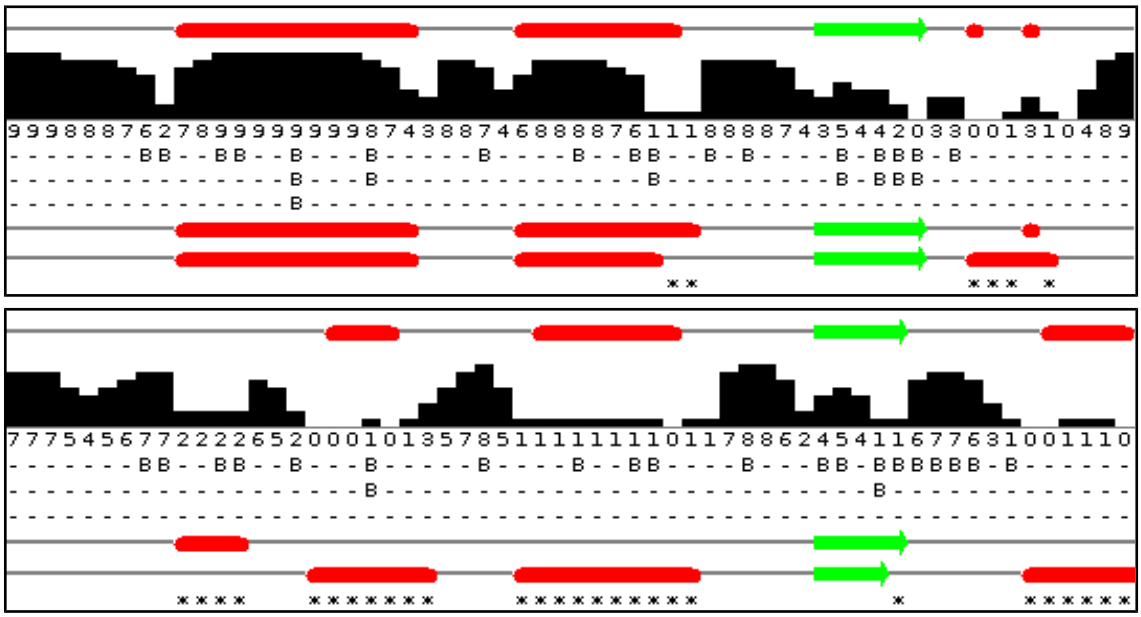
Massively improved I would suggest. All helices present and accurately placed. The **JPREDHMM** prediction, in



particular, is very much improved. The Beta Sheet predictions seem weak? It finds only one (accurately) of the three that **UniProtKB** suggests to be present. I wonder why, but the helices for the paired box domain specific prediction are excellent.

36 This could be why, as noted in the instructions, the start of the **PAX** region was considered insignificantly conserved by **Jalview**.

And so to the **homeobox** specific results. Once more, the new **homeobox** prediction (top) compared to the original (bottom).



As the **homeobox**s are significantly more numerous than the **paired box**s, less interference from sequences not including a **homeobox** might have been expected. I imagined the improvement in prediction would be minimal. However, it is very much better! All three helices are predicted in the correct positions, although **Jpred** appears to be a little reluctant about the third helix? There is a rather strong beta sheet prediction that is unsupported by **UniProtKB**. There is no reason to suppose that **UniProtKB** is **100%** correct, of course, but nothing I can find suggests that a beta sheet should appear in the middle of a homeobox. An enigma for another day.

So I conclude that this sort of protein analysis requires a little bit more than just throwing an entire sequence at a dumb program and assuming something marvellous will occur. In this case, considering the regions of the protein that are expected to be homologus separately is a very logical thing to do (and entirely obvious, retrospectively at least). Geoff Barton, whose group is responsible for **Jpred** agrees. He says[37]:

> " … Always split proteins into domains when searching. … "

So for both domains the prediction of the helices is far more accurate when each domain is considered separately. However, it is not just the red bars indicating the position of the helical predictions that should be noted. Look also at the confidence histogram. It indicates clearly that with more specific data to work on, better predictions can be made with much improved confidence (i.e. likelihood of being correct!).

37 As does the **Jpred Help**.

Searching PROSITE

What is the signature pattern for **N-myristoylation** site?

From the database entry, it can be seen that the pattern is **6** positions wide. **2** of those positions can be any amino acid. Only one position is fully specified. Not too demanding on the whole. I would expect this to match most proteins of any size and not always because there was an **N-myristoylation** site.

MYRISTYL, PS00008; N-myristoylation site (PATTERN with a high probability of occurrence!)

- Consensus pattern:
  G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}[GistheN-myristoylationsite]

How would you interpret this pattern?

The pattern is explained in the database thus.

- The N-terminal residue must be glycine.
- In position 2, uncharged residues are allowed. Charged residues, proline and large hydrophobic residues are not allowed.
- In positions 3 and 4, most, if not all, residues are allowed.
- In position 5, small uncharged residues are allowed (Ala, Ser, Thr, Cys, Asn and Gly). Serine is favored.
- In position 6, proline is not allowed.

The description is not entirely an honest reflection of the information to which the scanning software will respond. The software is given to understand that **ANY** amino acid can occur in positions **3** and **4**. The software has no way to know that "**Serine** is favoured" in position **5**! Maybe you think that my pointing out these transparent truths makes me an intolerable pedant? Well … so is the computer!

How many **N-myristoylation** sites did **ScanProsite** suggest there might be in **PAX6_HUMAN**?

11, is the short answer. A site every **40** amino acids or so.

PS00008 MYRISTYL *N-myristoylation site :*	
13 - 18:	GVfvNG
36 - 41:	GArpCD
110 - 115:	GVctND
151 - 156:	GQtgSW
154 - 159:	GSwgTR
157 - 162:	GTrpGW
182 - 187:	GGgeNT
183 - 188:	GGenTN
312 - 317:	GSmlGR
387 - 392:	GTsgTT
390 - 395:	GTtsTG

How many real **N-myristoylation** sites would you guess there might be in **PAX6_HUMAN**?

It is not really possible to answer this question from the evidence of this exercise. Intuitively, I would expect a large number of false positives from as weakly specified motif as this one. It has been suggested of this **PROSITE** pattern, by researchers looking at more sophisticated detection methods, that:

> "**PS00008** of **PROSITE** constructed from a small dataset … produces a great number of not only false positive but false negative predictions."

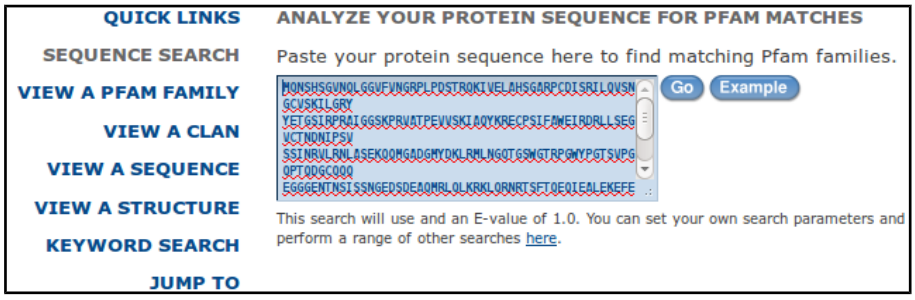
Good enough for me not to believe the majority of these predictions to be reliable. Not good enough for me to hazard a meaningful guess as to how many real sites I would expect in this particular protein.

Searching Pfam

Would you have found anything of interest had you chosen "to search for Pfam-B matches"?

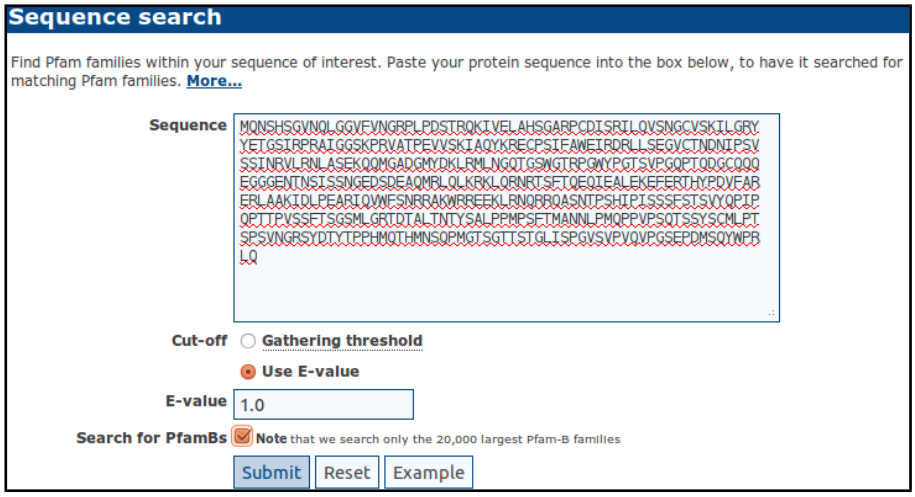
I did not find it straight forward to persuade the search to include **Pfam-B** entries.

In the end, I set up the search as required just to search **Pfam-A.** Then, click on the truly tiny **here** link at the end of the sentence suggesting *"You can set up your own search parameters ..."*.



A new search form! I pause to feel badly treated in that I have to re-enter my sequence! I click the Search for **PfamB**s box, reflecting as I do so, how nice that button would have looked on the previous search form?

With a triumphant flourish, I click on the **Submit** button.



NOTHING!! Well one use to find a **PfamB** hit that made all the fiddling around make a very small amount of sense, but now??? **NOTHING**!!! They claim the hit one got previously was not a worthy hit … but … but … it was a hit!!! AND, I had a very nice story about how it fitted in with all that we have discovered previously. One, in common with dear Victoria, is NOT amused!

2015.08.08: So you thought that was bad Dave Mawr!!! … Now the little button has gone altogether and cannot search **PfamB** at all???? Query sent to Help.

2015.08.10: In reply to the question " … Searching **PfamB** - Is this no longer possible? … "

> " … Correct - we are not longer producing **Pfam-B**s, largely because most of the clusters not covered by **Pfam** are rarely meaningful potential new domains. Our recommendation now, is to take the piece of sequence of interest and run it using the **HMMER** webserver.
>
> **http://www.ebi.ac.uk/Tools/hmmer/search/phmmer**
>
> This will then produce a set of results that essentially provide you with the same information as a **Pfam** entry. … "

This makes sense only if you change **really** to **rarely** in line 2? I assume this is what was meant (indeed it is, confirmed later). I also assume **PfamB** still to exists, in spirit at least, … as a source to discover new **PfamA** entries? But, users must effectively run **HMMER** themselves if they wish to search beyond what is offered by **PfamA** … now consistently referred to as **Pfam**? Hmm, maybe I try to clear this up a bit sometime?

Searching for Helix-Turn-Helix motifs

To which, if any, of the expected **HTH** motifs does this prediction correspond?

Using all the defaults, there is just one prediction:

```
Sequence: FARERLAAKIDLPEARIQVWFS
            |                   |
          238                 259
```

Looking again at the reported secondary structure in **UniProtKB**, I would expect there to be 3 **HTH** motifs roughly at:

```
39  →  63
99  → 116
237 → 275
```

So here we are looking at the second expected **HTH**. That is, the one associated with the helical triplet of the **HomeoBox** domain.

STRAND	6	8	3
STRAND	14	16	3
HELIX	23	34	12
HELIX	39	46	8
HELIX	50	63	14
STRAND	77	79	3
HELIX	81	93	13
HELIX	99	108	10
TURN	114	116	3
HELIX	120	133	14
HELIX	219	229	11
HELIX	237	246	10
HELIX	251	275	25

To which, if any, of the expected **HTH** motifs does the new prediction correspond?

The new prediction is

```
Sequence: PCDISRILQVSNGCVSKILG
            |                 |
           39                58
```

Near enough to the **HTH** expected for the first helical triplet of the **PairedBox** domain for me.

Can you suggest anything particular about the third putative **HTH** motif that might explain its reluctance to be discovered?

Nope …. which is why one should answer the questions straight away! I must have had a theory at one time?

It is possible to find this third **HTH** by using the old scoring matrix and an **SD** limit of **0.8**

However, by loosening things up this far, one gets seven answers instead of three. Fours out of the seven are wrong (well, one supposes so at least).

From your investigations of **Domain & Motif identification** using **Interpro**

Do you think it a good idea for **Interpro** to offer feature prediction programs as well as domain database searches?

Well … why not? The purpose of **InterProScan** is to associate regions of query proteins with **Interpro** domains. This was originally achieved, exclusively, by simply comparing a query sequence with all entries of relevant individual domain databases. These entries being representations of alignments of examples of specific domains constructed by homology searching (i.e. **blast** and similar).

I would suggest including a few predictor programs would provide extra evidence gathered from more general, more theoretical definitions of domains. I would imagine the inclusion of these program has improved and widened the picture provided by **InterProScan**.

Searching domain databases, typically composed of **HMM profiles**, such as **Pfam**, **Prosite** and **PRINTS** is quite different to running the predictor programs. As I cannot improve on the justification of this claim offered to me by Geoff Barton (Head of the group responsible for **Jalview**, **Jpred**, **Jnet** and much more), I will just reproduce his explanation here:

> " … The main difference is that with an **HMM profile** you have a "specific" example of a domain or motif whereas with something like **COILS**, you have something trained across all examples.
>
> For example, for secondary structure prediction, you could (a) do predictions of alpha-helix and beta-strand just by aligning a sequence to a protein of known structure, or an **HMM** from a family of aligned proteins of known structure. This is a specific case of secondary structure in the context of one protein family. Or (b) you can train a predictor from **ALL** protein families and then apply this. The advantage of (a) is it is very specific to the individual protein family and so should be more accurate for that family. The disadvantage is that it does not generalise to proteins that are not very like the specific example. The advantage of (b) is that it will work with any protein but will likely be less accurate than (a) for proteins that fit into the (a) category. … "

Do you think the Coil prediction might be correct?

I do not recall anything in what we have discovered thus far that would directly suggest there should be a **coiled coil** here, in the middle of the **HTH**. However, wikipedia does suggest **coiled coils** are associated with **transcription factors** (which **pax6_human** is).

" ... Many **coiled coil**-type proteins are involved in important biological functions such as the regulation of **gene expression**, e.g. **transcription factors**. ... "

I think I would not be overly convinced by this prediction, but I would not make that judgement with any great confidence. The all knowing **wikipedia** says:

" … **Coiled coils** usually contain a repeated pattern, ***hxxhcxc***, of **h**ydrophobic (***h***) and **c**harged (***c***) amino-acid residues, referred to as a **heptad repeat**. … "

Geoff Barton comments:

> " … Sometimes the pattern that is particular to **coiled-coils** also turns up in other helices that pack against each other. You would need to look at some examples of coiled-coil structures to see if the example you are using fits structurally. … "

Which seems very reasonable. The **heptad repeat** pattern could easily occur just by chance. **COILS** surely cannot predict the structure of the helices well enough to make an assured judgement? **COILS** offers a suggestion the user must follow up with other resources.

There is also the evidence that **Jpred**, possibly using the **COILS** program disguised as **LUPAS**, did not detect any coiled coils. This could be for a number of reasons. Possibly **LUPAS** is not the same program as **COILS**, or it is a different version, or **Jpred** runs **COILS**, but with different parameters.

Not many clear and confident answers in Bioinformatics are there!

Why might you suppose **Interpro** predicts only **2** of the **3 helix-turn-helix domain**s that might be expected?

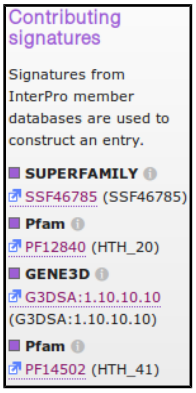
2 Winged helix-turn-helix (wHTH) DNA-binding domains are predicted coincident with the helical triplets of the **Paired domain**. This should broadly match your expectations.

No **helix-turn-helix (HTH) domain** is detected coincident with the **Homeobox domain**, where one might also have been expected?

I am not entirely certain why this might be, so I speculate.

Pfam attempts to classify a variety of types of **HTH**, and offers a range of **HTH** domain models (**HTH_17**, **HTH_38**, **HTH_39** and **HTH_40** to name but a few) and a number of **wHTH** domain models (including **HTH_33** and **HTH_24**).

Interpro also has a considerable number of **HTH** entries (**IPR017895**, **IPR032877**, **IPR007394**, **IPR013197** and more) and **wHTH** entries (**IPR005104**, **IPR023120** to name but **2**).



Interpro does use **Pfam** models to detect its various flavours of **HTH/wHTH** domain, but it does so selectively. For example, to detect the **wHTH** domains discovered here, only two **Pfam** families were used **HTH_20** and **HTH_41**, see illustration). These appear not to have matched in this instant as only a **G3DSA** entry is quoted.

All the above suggests that no one model exists to pick up all **HTH** domains? Possibly also, the fact that **HTH** domains come in such a variety of forms makes them difficult to detect reliably?

If you did the relevant supplementary exercise, you would find that the **EMBOSS** program used easily detected the **Homeobox domain HTH** but essentially failed to detect the **wHTH**s recorded here. This must be because the, very simple, model used by the program only reliably applies to a specific range of **HTH** domains/motifs that includes the one in the **Homeobox domain** of the human **PAX6** protein?

I am very open to better explanations. I am not completely convinced by the discussion above.

From your investigations of **Multiple Sequence Alignment**

Clustalx

How might you have saved the need to recompute the alignment by selecting an **Edit** option other than **Remove All Gaps**?

Pretty sure using **Remove Gap-Only columns** rather than **Remove All Gaps** would have got to the second alignment in one step. I struggle to see how this could fail, but cannot convince myself it would be absolutely certain to work. Maybe some of the sequences that were removed would subtly alter the alignment calculations? Doubtful, given how crude the whole thing is. Anyway, it would have been good enough for me in "Real Life" for this data. I did not suggest this approach as we had to make a **Fasta** format file with the entire sequence set to be aligned by the alternative software as a by product of working with **clustalx**. This would not be easily possible if the **Remove Gap-Only columns** short cut was taken.

Muscle

How do the options for the **OUTPUT TREE** relate to the output files of **ClustalX** and the difference between the way that **ClustalX** and **muscle** work?

I leave this question here in the hope that one day I will be able to offer a sensible answer. First draft answer below.

Essentially, both **ClustalX** and **MUSCLE** work in two stages. First they create **Guide Tree(s)** (**ClustalX** just one, **MUSCLE** creates **2**). Then from the **Guide Tree(s)** they compute the final **MSA**. The purpose of saving the **Guide Tree(s)** to a file is to enable a rerun of the second phase with new parameter settings without having to rerun the first **Guide Tree** generation stage.

Of course, as mentioned above, utterly pointless if there is no way to change the parameters? but that is the theory.

More investigation by me and expansion of this answer required. Discussion with EBI current (2016.04.20).

Comment on how one might choose between the range of options offered for the aligned parameter?

I cannot … beyond suggesting it simply does not make sense? Going by what is offered at Wageningen, the choice should be between **aligned** and **input order**. i.e. the order of the original set of sequences to be aligned or the order after they have all been compared with each other and arranged into a **Guide Tree** … or two.

Interpro:

Relationships: PARENT/CHILD; CONTAINS/FOUND IN

The PARENT/CHILD relationship is used to indicate family/subfamily relationships defined by the member database methods in the entries. To be a CHILD, >75% of the protein set of the CHILD must be represented in the PARENT. If an InterPro PARENT has more than one InterPro CHILD a protein sequence should not be found in the match table of more than one of these children. If one InterPro entry is described as the child of another InterPro entry, this implies that the child entry is more specific than the parent, and that in all cases a protein sequence match to the child entry implies a match to the parent. Signatures for the parent and child entries must overlap by >50% of their sequence and there must be no adjacent signatures to the CHILD that are also covered by one or more of the PARENT signatures. A list of the PARENT/CHILD relationships by InterPro entry accession is available from the ftp site: ParentChildTreeFile.txt.

The CONTAINS/FOUND IN relationship is used to indicate features of the entry that are not defined by Parent/Child relationships, these include: Regions, Domains, Repeats and Sites. For a CONTAINS/FOUND IN relationship to be established, 40% or above of the proteins in the InterPro entry must contain the feature. Some features can be found in more than one type of protein or family of proteins, but is not children in the family sense. Features may be structural or functional and can be found in proteins with different domain organisations. The CONTAINS/FOUND IN relationship, is therefore useful in linking InterPro entries which are associated by composition but are not related hierarchically.

Taxonomy coverage:

Taxonomy Coverage

The Taxonomy Coverage aims to provide 'at a glance' view of the taxonomic range of the sequences associated with each InterPro entry and the number of sequences associated with each lineage. The taxonomic lineages are 'clickable' and provide a pop-up, which displays the tax-ID, the taxonomy and taxonomic subgroup(s)/species having matches to proteins, the protein match counts and a FASTA link. Clicking on the taxonomy or taxonomic subgroup(s)/species links to the protein overview matches for the selected taxonomy. Clicking on the FASTA box will download the complete set of FASTA sequences for the selected taxonomy of the entry.

The lineages were carefully selected to provide a view of the major groups of organisms. The circular display has the taxonomy-tree root as its centre. Selected model organisms populate the outer most circle. Nodes of the taxonomy-tree are placed on the inner circles. Radial lines lead to the description for each node. No significance is attached to the position of the node on a particular inner-circle, other than convenience, though some attempt has been made to group nodes. The nodes themselves are either true taxonomy nodes and have a NCBI taxonomy number or are artificial nodes created for this display; of which there are three: **Unclassified**, **Other Eukaryotes** (Non-Metazoa) and the **Plastid Group**.

Artificial Taxon: **Unclassified** contains the following NCBI taxon groups:

- Taxonomy:12884 Viroids
- Taxonomy:12908 unclassified sequences
- Taxonomy:28384 other sequences

The Eukaryota (TAXONOMY:2759) comprises 29 taxons, these have been grouped into two artificial taxons and one existing taxon:

Fungi/Metazoa (TAXONOMY:33154); Node **Metazoa**

Artificial Taxon; **Plastid Group**, this contains the following NCBI taxon groups:

- Taxonomy: 2763 Rhodophyta
- Taxonomy: 2830 Haptophyceae
- Taxonomy: 3027 Cryptophyta
- Taxonomy: 33090 Viridiplantae
- Taxonomy: 33630 Alveolata
- Taxonomy: 33634 stramenopiles
- Taxonomy: 33682 Euglenozoa

- Taxonomy: 38254 Glaucocystophyceae
- Taxonomy: 339960 Katablepharidophyta

Each taxonomic group within this artificial taxon contains organisms that have a plastid.

Artificial Taxon; **Other Eukaryotes** (Non-Metazoa), this comprises the following NCBI taxon groups:

- Taxonomy: 5719 Parabasalidea
- Taxonomy: 5752 Heterolobosea
- Taxonomy: 66288 Oxymonadida
- Taxonomy: 136087 Malawimonadidae
- Taxonomy: 154966 Nucleariidae
- Taxonomy: 193537 Centroheliozoa
- Taxonomy: 207245 Diplomonadida group
- Taxonomy: 543769 Rhizaria
- Taxonomy: 554296 Apusozoa
- Taxonomy: 554915 Amoebozoa
- Taxonomy: 556282 Jakobida

Each taxonomic group within this artificial taxon are the remaining taxonomic groups of the NCBI taxon:2759, which are not in the Plastid Group and are not Fungi/Metazoa (TAXONOMY:33154).

Overlapping Interpro entries

Overlapping InterPro Entries

This section displays entries that share more than 70% of their proteins. Such overlaps define PARENT/CHILD and CONTAINS/FOUND IN relationships between InterPro entries.

IPR009007	Numbers of overlapping proteins	Average numbers of overlapping amino acids

In the above example, InterPro entry IPR011969 contains proteins which are also found in IPR009007 as a result of the protein signatures of the two entries overlapping.

The two entries have been compared firstly by counting the number of proteins which are common to both, the results of which are displayed in the Venn diagram on the left, and secondly by calculating the average overlap of the protein signatures, in amino acids, with the results displayed in the bar diagram on the right.

Venn diagram display of the overlap of proteins common to both entries:

- The purple intersection contains the number of overlapping proteins common to both IPR009007 and IPR011969, which is 31 in this case.
- The pink section on the left is the number of proteins found in IPR009007 but not IPR011969, which is 35378.

Bar diagram display of the average amino acid overlap between the protein signatures:

The average number of amino acids overlapping in the sequences of the 31 proteins common to both entries is then calculated, with the results displayed in the bar diagram on the right. The bar diagram display is only shown for 'Domain - Domain' relationships.

- The purple segment in the middle shows the average number of amino acids overlapping between IPR009007 and IPR011969 for the 31 proteins, in this case 104.
- The pink segment shows the average number of amino acids found in IPR009007, but not IPR011969, for the 31 proteins, which is 0.
- The blue segment shows the average number of amino acids found in IPR011969, but not IPR009007, for the 31 proteins, which is 15.

The results of these comparisons are used to calculate the percentage overlap score, with all scores greater than 70% displayed on the InterPro pages. In this example, since all proteins found in IPR011969 are also found in IPR009007, and all the amino acids from IPR009007 overlap with those from IPR011969, the percentage overlap score is 100%.

Muscle:

- The blue section on the right is the number of proteins found in IPR011969 but not IPR009007, which is 0; i.e. all proteins associated with IPR011969 occur in IPR009007.

How do the options for the **OUTPUT TREE** relate to the output files of **ClustalX** and the difference between the way that **ClustalX** and **muscle** work?_____

none
From first iteration
From second iteration

Comment on how one might choose between the range of options offered for the aligned parameter?_____

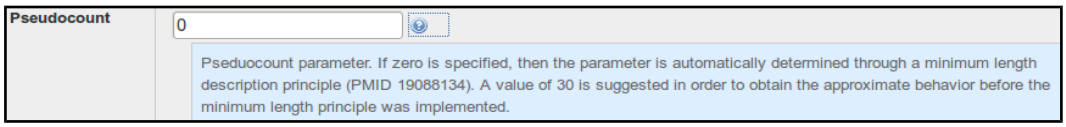
Clearly?

aligned

PSI-Blast

What do you suppose the choice of **Pseudocount** might influence?

I clicked with confidences upon the link to the help. It opined as illustrated.



I suppose the next step is to read **PMID 19088134**? There is most certainly no elucidation amongst the strangle of words offered here?

The article **Abstract** says:

"**P**osition **s**pecific **s**core **m**atrices (**PSSM**s) are derived from multiple sequence alignments to aid in the recognition of distant protein sequence relationships. The **PSI-BLAST** protein database search program derives the column scores of its **PSSM**s with the aid of **pseudocounts**, added to the observed amino acid counts in a multiple alignment column. In the absence of theory, the number of **pseudocounts** used has been a completely empirical parameter. This article argues that the minimum description length principle can motivate the choice of this parameter. Specifically, for realistic alignments, the principle supports the practice of using a number of **pseudocounts** essentially independent of alignment size. However, it also implies that more highly conserved columns should use fewer **pseudocounts**, increasing the inter-column contrast of the implied **PSSM**s. A new method for calculating **pseudocounts** that significantly improves **PSI-BLAST**'s; retrieval accuracy is now employed by default."

The article itself, continues in like vein ….. how about we close our eyes and accept the defaults? I would just wonder why the whole thing does not commence with, at least an attempt, to answer the question in the forefront of my inquiry, which is .. "**WHAT, in the current context, IS a pseudocount?**". I do not believe it is as tricky as they appear to wish us to believe. I will try again later, when my view of the world is less storm infested.

******

**********NOTES**********

HelixTurnHelix

helixturnhelix uses the method of Dodd and Egan and finds helix-turn-helix nucleic acid binding motifs in proteins.

The helix-turn-helix motif was originally identified as the DNA-binding domain of phage repressors. One alpha-helix lies in the wide groove of DNA; the other lies at an angle across DNA.

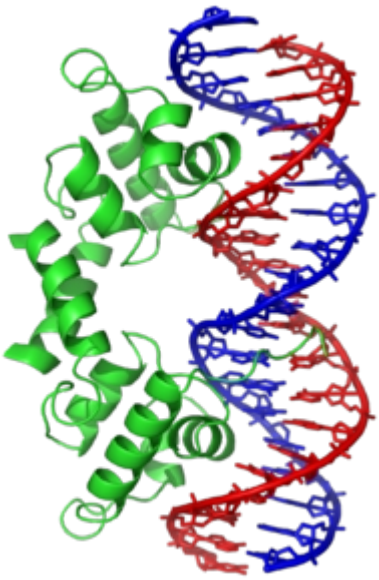
The old (1987) data has a motif length of 20 residues, whilst the default data (Ehth.dat) has a motif length of 22 residues.

With care these can be replaced to suit your data sets. If the files are placed in the following directories they will be used in preference to the files in the EMBOSS distribution data directory:

- . (your current directory)
- .embossdata
- ~/ (your home directory)
- ~/.embossdata

Here is the default file:

```
# Amino acid counts for 91 Helix-turn-helix (presumed) protein motifs
# from Dodd IB and Egan JB (1990) Nucl. Acids. Res. 18:5019-5026.
#
Sample: 91 aligned sequences
#
# R  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 Total Exp
# - -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- ----- ---
  A  2  1  3 14 10 12 75  6 15  9  1  1  4  3  8 15  4  4  4 11  0 10   212 995
  C  0  0  1  1  0  0  0  0  0  3  3  1  1  0  0  0  0  0  0  1  0  3    14 106
  D  0  1  0  1 14  0  0 14  1  0  5  0  1  2  0  0  0  0  1  1  0  2    43 556
  E  4  5  0 11 26  0  0 16  9  3  3  0  3 12 13  0  0  2  0  1 13  6   127 669
  F  4  0  4  0  0  4  0  1  0 10  0  0  0  0  1  0  0  1  1  1 22  0    49 358
  G  9  7  1  4  0  0  8  0  0  0 50  0  6  0  7  1  0  3  1  1  0  4   102 761
  H  4  3  1  1  2  0  0  3  2  0  5  0  3  3  0  2  0  2  4  5  0  2    42 225
  I 10  0 13  3  2 15  0  4  9  4  0 17  0  2  0  1 31  1  4  8 16  1   141 583
  K  4  4  6 11 12  1  1 14 11  0  5  2  2  7  2  1  0  5  8  4  5 15   120 516
  L 16  1 17  0  1 35  0  3 12 31  0 22  0  2  1  1 22  1  1 12 20  0   198 954
  M  7  0  2  1  1  1  0  0  5  7  1 10  0  0  2  0  2  0  0  2  0  1    42 275
  N  0  8  0  1  0  0  0  2  1  1 14  0  8  1  4  2  0  4  9  0  0 11    66 383
  P  1  6  0  1  0  0  0  0  0  0  0  0  3 13  7  0  0  0  0  0  0  3    34 403
  Q  2  1 21  9 11  0  0  9  8  0  0  2  1 17  7 12  0  3 12  5  3  9   132 437
  R  9 10 14  9  5  0  1 16 10  0  1  0  1 17  8  7  0 17 28  3  0 16   172 609
  S  2 17  0  8  4  1  6  1  2  2  3  0 37  1 25  5  0 29  3  0  1  5   152 552
  T  6 24  3 12  1  5  0  2  2  4  0  5 20  4  3 39  0  4  1  0  4  3   142 512
  V  7  3  1  1  2 16  0  0  2 12  0 29  0  5  3  3 32  0  7  8  7  0   138 724
  W  2  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  2 21  0  0    27 105
  Y  2  0  4  3  0  1  0  0  2  4  0  1  1  2  0  2  0 15  5  7  0  0    49 267
```



The λ repressor of bacteriophage lambda employs a helix-turn-helix (top; green) to bind DNA (bottom; blue and red).

In proteins, the helix-turn-helix (HTH) is a major structural motif capable of binding DNA. It is composed of two α helices joined by a short strand of amino acids and is found in many proteins that regulate gene expression. It should not to be confused with the helix-loop-helix domain.

Its discovery was based on similarities between the genes for Cro, CAP, and λ repressor, which share a common 20-25 amino acid sequence that facilitates DNA recognition. In particular, recognition and binding to DNA is done by the two α helices, one occupying the N-terminal end of the motif, the other at the C-terminus. In most cases, such as in the Cro repressor, the second helix contributes most to DNA recognition, and hence it is often called the "recognition helix". It binds to the major groove of DNA through a series of hydrogen bonds and various Van der Waals interactions with exposed bases. The other α helix stabilizes the interaction between protein and DNA, but does not play a particularly strong role in its recognition.

DPJ – 2015.09.14