Practical Bioinformatics

Mark Voorhies

5/19/2015

Review – Documentation

A program should communicate its intent to both the computer and human programmers.

- Comments
- Docstrings

Review – Documentation

A program should communicate its intent to both the computer and human programmers.

- Comments
- Docstrings
- Code and inputs defining a complete protocol

Review – Documentation

A program should communicate its intent to both the computer and human programmers.

- Comments
- Docstrings
- Code and inputs defining a complete protocol
- Positive and negative controls

Review – "Top Down" design

- Experiment in the shell
- Factor working code into functions and modules
- Refine from problem-specific to generally applicable functions

Review – "Top Down" design

- Experiment in the shell
- Factor working code into functions and modules
- Refine from problem-specific to generally applicable functions
  "As simple as possible, but no simpler"

Dictionaries

```
dictionary = {"A":"T","T":"A","G":"C","C":"G"}
dictionary["G"]
dictionary["N"] = "N"
dictionary.has_key("C")
```

Dictionaries

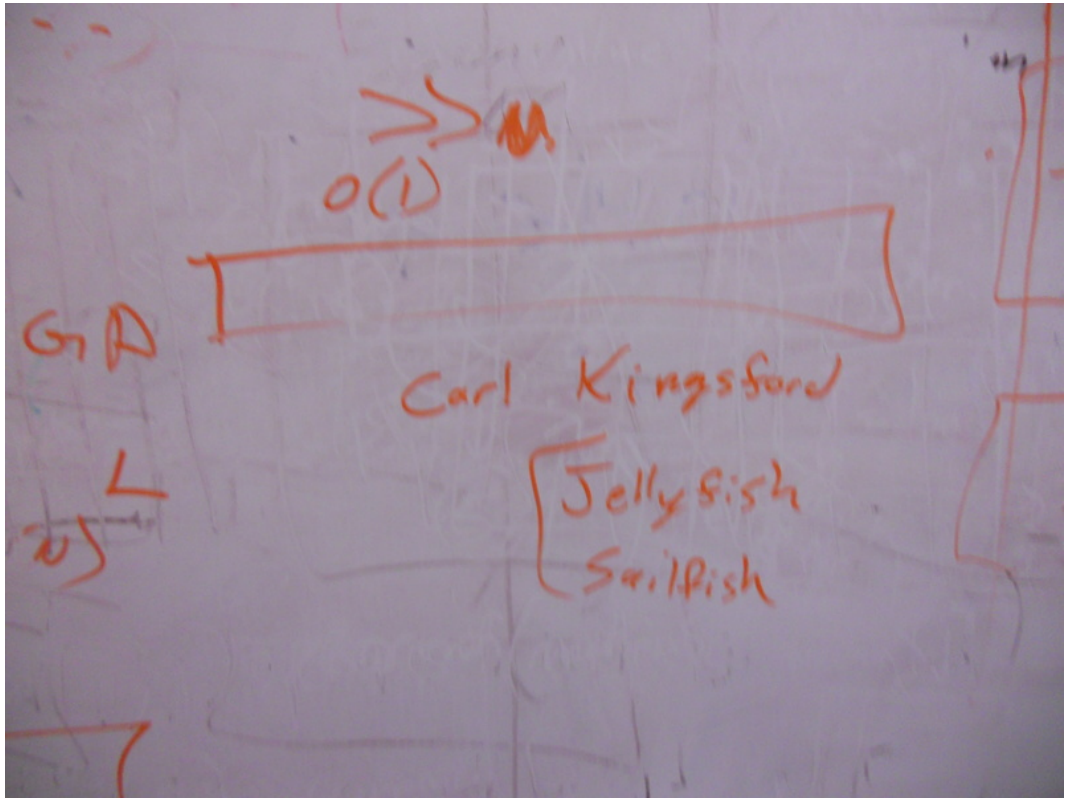
```
geneticCode = {"TTT":"F","TTC":"F","TTA":"L","TTG":"L",
               "CTT":"L","CTC":"L","CTA":"L","CTG":"L",
               "ATT":"I","ATC":"I","ATA":"I","ATG":"M",
               "GTT":"V","GTC":"V","GTA":"V","GTG":"V",

               "TCT":"S","TCC":"S","TCA":"S","TCG":"S",
               "CCT":"P","CCC":"P","CCA":"P","CCG":"P",
               "ACT":"T","ACC":"T","ACA":"T","ACG":"T",
               "GCT":"A","GCC":"A","GCA":"A","GCG":"A",

               "TAT":"Y","TAC":"Y","TAA":"*","TAG":"*",
               "CAT":"H","CAC":"H","CAA":"Q","CAG":"Q",
               "AAT":"N","AAC":"N","AAA":"K","AAG":"K",
               "GAT":"D","GAC":"D","GAA":"E","GAG":"E",

               "TGT":"C","TGC":"C","TGA":"*","TGG":"W",
               "CGT":"R","CGC":"R","CGA":"R","CGG":"R",
               "AGT":"S","AGC":"S","AGA":"R","AGG":"R",
               "GGT":"G","GGC":"G","GGA":"G","GGG":"G"}
```

Whiteboard Image



Exercise: Transforming sequences

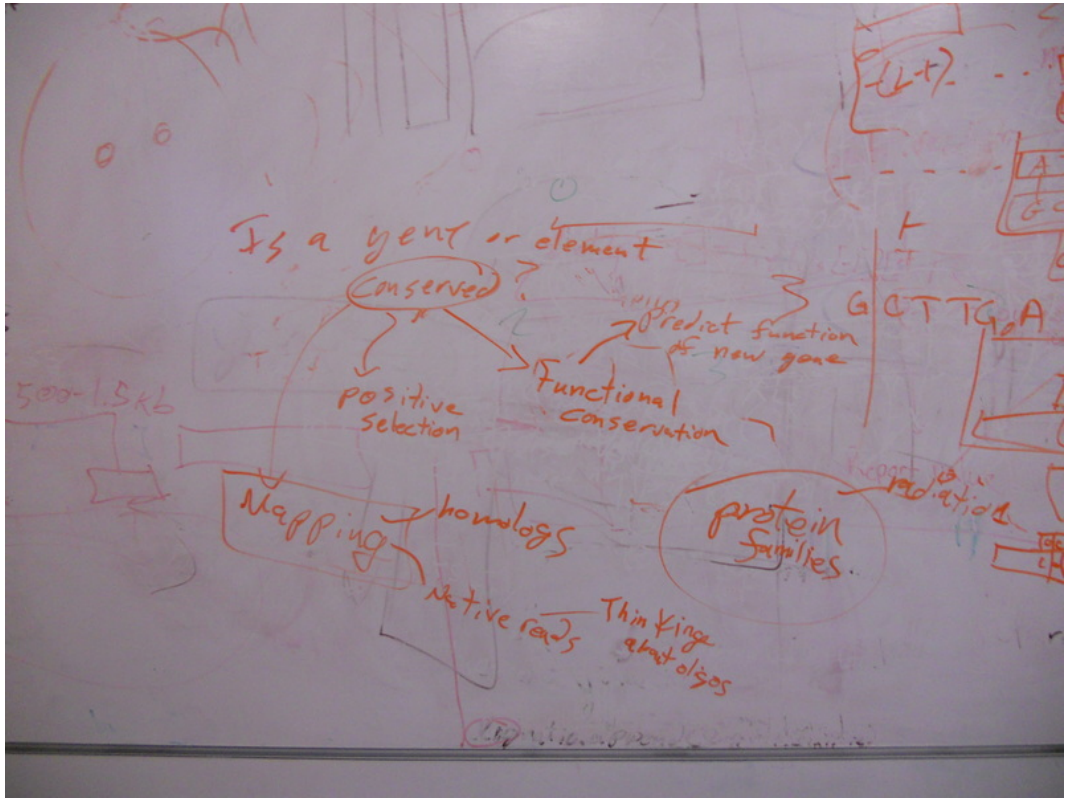
1. Write a function to return the antisense strand of a DNA sequence in 3'→5' orientation.
2. Write a function to return the complement of a DNA sequence in 5'→3' orientation.
3. Write a function to translate a DNA sequence

Why compare sequences?

Why compare sequences?

- To find genes with a common ancestor
- To infer conserved molecular mechanism and biological function
- To find short functional motifs
- To find repetitive elements within a sequence
- To predict cross-hybridizing sequences (*e.g.*, in microarray design)
- To find genomic origin of imperfectly sequenced fragments (*e.g.*, in deep sequencing experiments)
- To predict nucleotide secondary structure

Whiteboard Image



Nomenclature

Homologs heritable elements with a common evolutionary origin.

Nomenclature

Homologs heritable elements with a common evolutionary origin.

Orthologs homologs arising from speciation.

Paralogs homologs arising from duplication and divergence within a single genome.

Nomenclature

Homologs heritable elements with a common evolutionary origin.

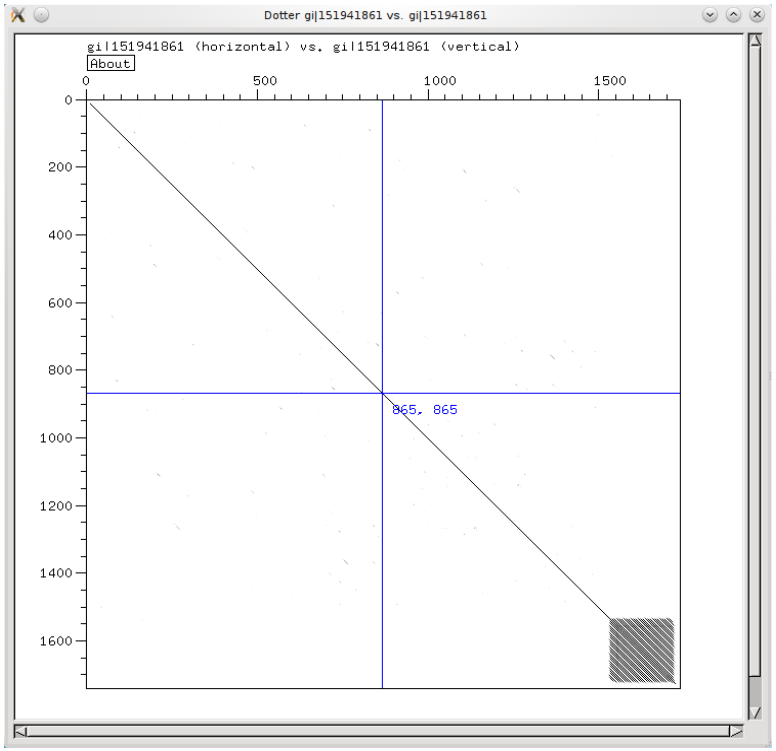
Orthologs homologs arising from speciation.

Paralogs homologs arising from duplication and divergence within a single genome.

Xenologs homologs arising from horizontal transfer.

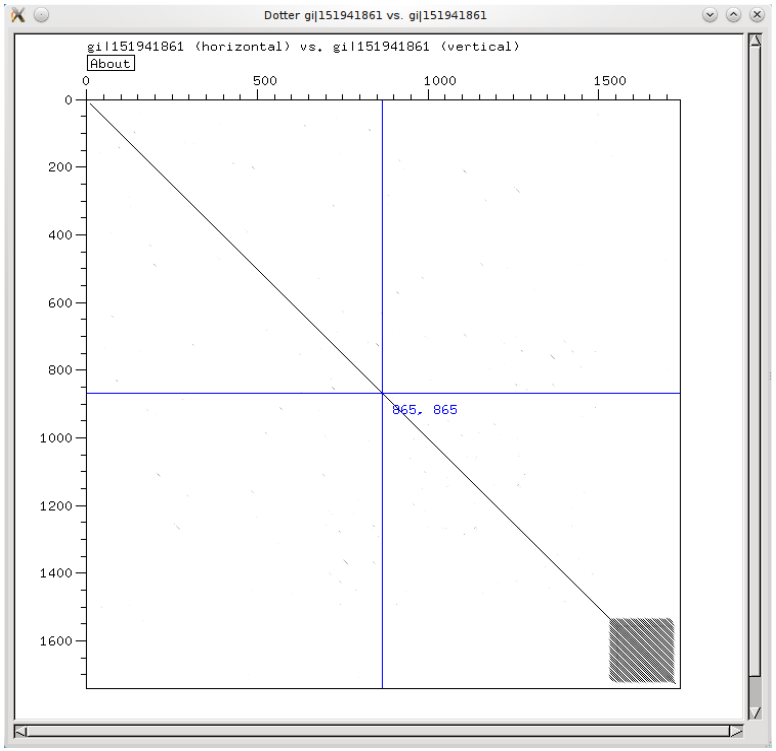
Onologs homologs arising from whole genome duplication.

Dotplots



1. Unbiased view of all ungapped alignments of two sequences

Dotplots



1. Unbiased view of all ungapped alignments of two sequences
2. Noise can be filtered by applying a smoothing window to the diagonals.

Types of alignments

Global Alignment Each letter of each sequence is aligned to a letter or a gap (*e.g.*, Needleman-Wunsch)

Local Alignment An optimal pair of subsequences is taken from the two sequences and globally aligned (*e.g.*, Smith-Waterman)

Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

$$S(x, y) = \sum_{i}^{N} s(x_i, y_i)$$

Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

$$S(x, y) = \sum_{i}^{N} s(x_i, y_i)$$

1. Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.

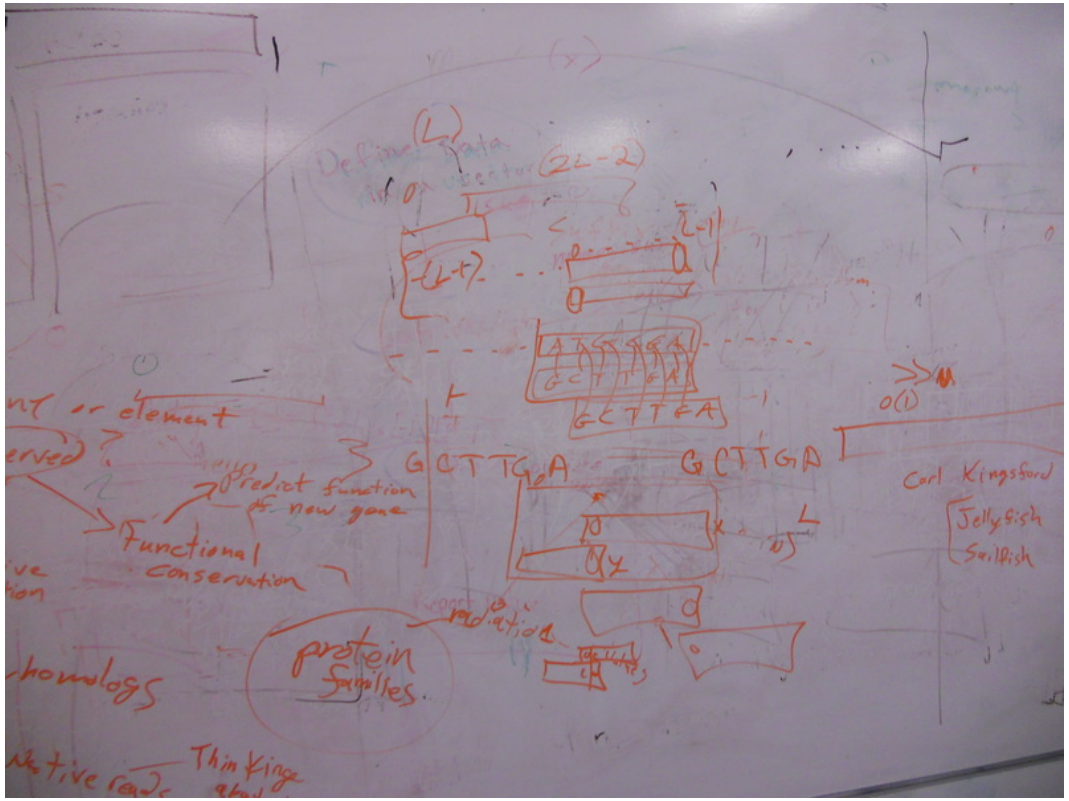
Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

$$S(x, y) = \sum_{i}^{N} s(x_i, y_i)$$

1. Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.
2. Given two sequences and a scoring matrix, find the offset that yields the best scoring ungapped alignment.

Whiteboard Image



Exercise: Scoring a gapped alignment

1. Given two equal length gapped sequences (where "-" represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.

Exercise: Scoring a gapped alignment

1. Given two equal length gapped sequences (where "-" represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.
2. Write a new scoring function with separate penalties for opening a zero length gap (*e.g.*, G = -11) and extending an open gap by one base (*e.g.*, E = -1).

$$S_{gapped}(x, y) = S(x, y) + \sum_{i}^{gaps} (G + E * len(i))$$