Repeat, Repeat!

Erik S. Wright

June 12, 2024

Contents

1 Introduction

Jacques Monod famously penned an essay in 1977 likening the evolutionary process to tinkering and boldly claiming that "evolution does not produce novelties from scratch." He cites molecular evidence:

> This is exemplified by the finding that large segments of genetic information, that is, of DNA, turn out to be homologous, not only in the same organism, but also among different organisms, even among those that are phylogenetically distant. Similarly, as more is known about amino acid sequences in proteins, it appears not only that proteins fulfilling similar functions in different organisms have frequently similar sequences, but also that proteins with different functions also exhibit rather large segments in common.

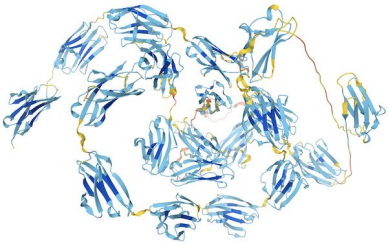

Figure 1: Partial predicted structure of the human protein Obscurin showing the repeated Ig domains.

We now know that proteins can, and regularly do, evolve *de novo*. Yet, repetition of existing sequence, followed by divergence (i.e., tinkering), is still a major source of evolutionary innovation. DNA slippage during replication is one mechanism that can result in repeated sequences, in particular adjacent repeats in the DNA. These tandem repeats, especially in the form of short microsatellites, underly several human diseases including fragile X syndrome and Huntington's disease.

Microsatellites are easy to spot by eye, as they contain repetitive stretches of a few nucleotides repeated many times in a row (e.g., CAG CAG CAG CAG CAG [...] in the protein Huntintin). Harder to detect are tandem repeats that have evolved considerably since their origin, such that the repeated copies no longer look anything like their shared ancestral sequence. It is estimated that over half of the human proteome contains tandem repeats, many of which share an ancient origin and are undetectable by eye.

Obscurin is one human protein containing several repeats that are difficult to spot by eye. It contains 68 Ig (Immunoglobin) domains, which each contain multiple antiparallel β-sheets (Fig. 1). Many of these Ig domains are found adjacent to each other and likely arose from a common ancestor. This vignette describes how to detect these obscure repeats using the `DetectRepeats` function in the R package DECIPHER.

2 Getting Started

2.1 Startup

To get started we need to load the DECIPHER package, which automatically loads a few other required packages.

```
> library(DECIPHER)
```

Detecting repeats is performed with the `DetectRepeats` function. Its help page can be accessed through:

```
> ? DetectRepeats
```

Once DECIPHER is installed, the code in this tutorial can be obtained via:

```
> browseVignettes("DECIPHER")
```

3 Taking a first look at Obscurin

Here, we will take a deep dive into the human protein Obscurin, which is expressed in muscle tissue. Obscurin is a large protein primarily composed of Ig domains, an example of which looks like:

```
> Ig <- AAStringSet("VRALPARFTEGLRNEEAMEGATATLQCELSKAAPVEWRKGLEALRDGDKYSLRQDGAVCELQIHGLAI
> Ig
AAStringSet object of length 1:
    width seq
[1]    88 VRALPARFTEGLRNEEAMEGATATLQCELSKAAP...GAVCELQIHGLAMADNGVYSCVCGQERTSATLT
```

Since we know the form of the repeat, we can find others like it in the sequence by aligning the repeat to a sliding window along the entire protein sequence. First, we load our sequences with:

```
> # specify the path to your file of sequences:
> fas <- "<<path to training FASTA file>>"
> # OR use the example DNA sequences:
> fas <- system.file("extdata",
          "LongHumanProteins.fas.gz",
          package="DECIPHER")
> # read the sequences into memory
> DNA <- readDNAStringSet(fas)
> DNA
DNAStringSet object of length 11:
       width seq                                                names
 [1] 107976 ATGACAACTCAAGCACCGACGT...TATACATATTCGATCCATTTAA titin isoform IC
 [2]  26778 ATGGATCAGCCACAGTTCAGCG...CAACCTGGCCCAGGTGCGCTGA obscurin isoform c
 [3]  16908 ATGATTTCCTGGGAAGTTGTCC...TGTGTCCGCCTATCCATACTAA hemicentin-1 prec...
 [4]   5847 ATGGCGCCCACCTGGGGCCCTG...CTTTGACCACTATGCAACCTAA receptor-type tyr...
 [5]   5745 ATGGGGGATGTGAAGCTGGTTG...AGGGGAAGAGGAAGAAGAGTGA myosin light chai...
 ...     ... ...
 [7]   3825 ATGCCTGAGCCGGGGAAGAAGC...GGAGGTGCGAGTGCCTCAGTGA myosin-binding pr...
 [8]   1497 ATGTTTAACTACGAACGTCCAA...CTATGAAAGTGAAGAACTTTAA myotilin isoform a
 [9]   1263 ATGCTCAAAGCGGTGATCCTGA...CACCAACCAGATCATCCTCTGA mannose-1-phospha...
[10]   1101 ATGGTGCTGCACCTACTGCTCT...GCAGCACTGTGCTTGTATCTAA inhibin alpha cha...
[11]    717 ATGTCTAAAGCAGCTCCTGCCA...CGCTCGCGTCGACGTGGCCTGA SPEG neighbor pro...
```

Although `DetectRepeats` works on nucleotide and protein sequences, it is generally easier to find tandem repeats in amino acid sequences than it is to find them in DNA sequences. We can easily translate our proteins:

```
> AA <- translate(DNA)
> AA
AAStringSet object of length 11:
     width seq                                               names
 [1] 35992 MTTQAPTFTQPLQSVVVLEGSTA...LSLGNEFGSDSATVNIHIRSI* titin isoform IC
 [2]  8926 MDQPQFSGAPRFLTRPKAFVVSV...VRNREKRRALLYKRHNLAQVR* obscurin isoform c
 [3]  5636 MISWEVVHTVFLFALLYSSLAQD...NGTIEYQTTFIVYIAVSAYPY* hemicentin-1 prec...
 [4]  1949 MAPTWGPGMVSVVGPMGLLVVLL...EYQFCYQAALEYLGSFDHYAT* receptor-type tyr...
 [5]  1915 MGDVKLVASSHISKTSLSVDPSR...TAELIVETMEEGEGEGEEEEE* myosin light chai...
 ...   ... ...
 [7]  1275 MPEPGKKPVSAFSKKPRSVEVAA...RATNLQGEARCECRLEVRVPQ* myosin-binding pr...
 [8]   499 MFNYERPKHFIQSQNPCGSRLQP...NPEGEFQRLAAQSGLYESEEL* myotilin isoform a
 [9]   421 MLKAVILIGGPQKGTRFRPLSFE...LNSIVLPHKELSRSFTNQIIL* mannose-1-phospha...
[10]   367 MVLHLLLFLLLTPQGGHSCQGLE...GGYSFKYETVPNLLTQHCACI* inhibin alpha cha...
[11]   239 MSKAAPAKKPVAVAPAPGCTLDI...YEVYVENSLGMDQSFARVDVA* SPEG neighbor pro...
> names(AA)
 [1] "titin isoform IC"
 [2] "obscurin isoform c"
 [3] "hemicentin-1 precursor"
 [4] "receptor-type tyrosine-protein phosphatase S isoform 1 precursor"
 [5] "myosin light chain kinase, smooth muscle isoform 1"
 [6] "roundabout homolog 1 isoform a precursor"
 [7] "myosin-binding protein C, cardiac-type"
 [8] "myotilin isoform a"
 [9] "mannose-1-phosphate guanyltransferase alpha"
[10] "inhibin alpha chain preproprotein"
[11] "SPEG neighbor protein"
> index <- 2
> AA <- AA[index]
```

We see that Obscurin is the second protein in the *AAStringSet*. The others are related human proteins containing Ig domains. It is possible to analyze many proteins at the same time, but this vignette focuses on only a single protein so the examples run quickly. To detect repeats in a different protein, or multiple proteins, simply change the value of `index` above.

We can find the repeated sequence using pairwise alignment by aligning each of the windows along the sequence to the Ig domain and plot their scores. The alignment score quantifies how well each window along the protein matches the Ig domain. Windows matching Ig domains will show up as peaks in the score.

We can make a couple of observations from the plot. First, many of the Ig domains are located adjacent to each other, likely as a result of tandem repeats, although some are isolated to other parts of the protein. Second, the tandem repeats around the main peak have higher scores, suggesting these orthologous repeats share a more recent evolutionary ancestor. Many of the other repeats have negative scores even though they are also Ig domains.

```
> windows <- extractAt(AA[[1]],
         IRanges(seq_len(width(AA[1]) - width(Ig)),
                 width=width(Ig)))
> p <- AlignPairs(windows,
         Ig,
         pairs=data.frame(Pattern=seq_along(windows), Subject=1),
         verbose=FALSE)
> plot(p$Score, ylab="Score", xlab="Amino acid position", type="l")
```

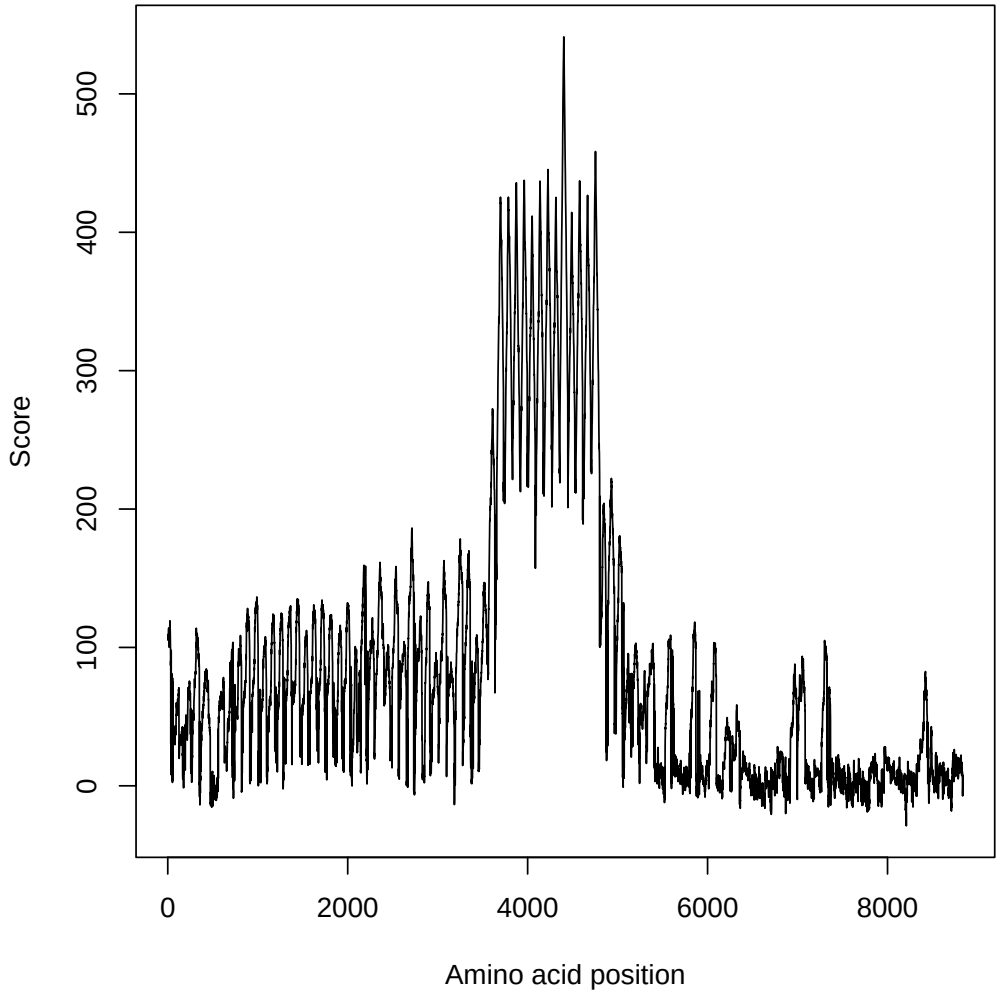

Figure 2: Peaks marking the start of Ig domains in the human protein Obscurin.

4 Detecting repeats automatically with DetectRepeats

With Obscurin we have the advantage that we already know to look for repeated Ig domains. But how can we find tandem repeats when we don't know what to expect? This is when it comes in handy to use the `DetectRepeats` function to find tandem repeats. It takes sequences as input and then outputs the predicted tandem repeats and their associated scores. Let's take a look at how it does with Obscurin:

```
> reps <- DetectRepeats(AA, allScores=TRUE)
================================================================================

Time difference of 61.94 secs
> reps[which.max(reps[, "Score"]),]
   Index Begin  End         Left        Right    Score
73     1   620 5668 620, 714.... 713, 804.... 673.4108
```

The result is a *data.frame* giving the beginning and ending positions of the tandem repeat, as well as every repeat within it. The top scoring repeat spans the Ig domains we found earlier. Since we asked for *allScores*, the output contains all repeats with scores above *minScore*, even if they are overlapping. This allows us to easily plot the different repeats that were identified in Obscurin:

All of the adjacent Ig domains were identified multiple times with different beginning and ending positions. If we set *allScores* to `FALSE` (the default) then we would only obtain the top scoring repeat in each region.

```
> plot(NA,
         xlim=c(0, max(width(AA))),
         ylim=range(0, reps[, "Score"]),
         xlab="Position in protein",
         ylab="Tandem repeat score")
> for (i in seq_len(nrow(reps)))
         segments(reps[[i, "Left"]],
                  reps[i, "Score"],
                  reps[[i, "Right"]],
                  reps[i, "Score"],
                  col=seq_along(reps[[i, "Left"]]))
```

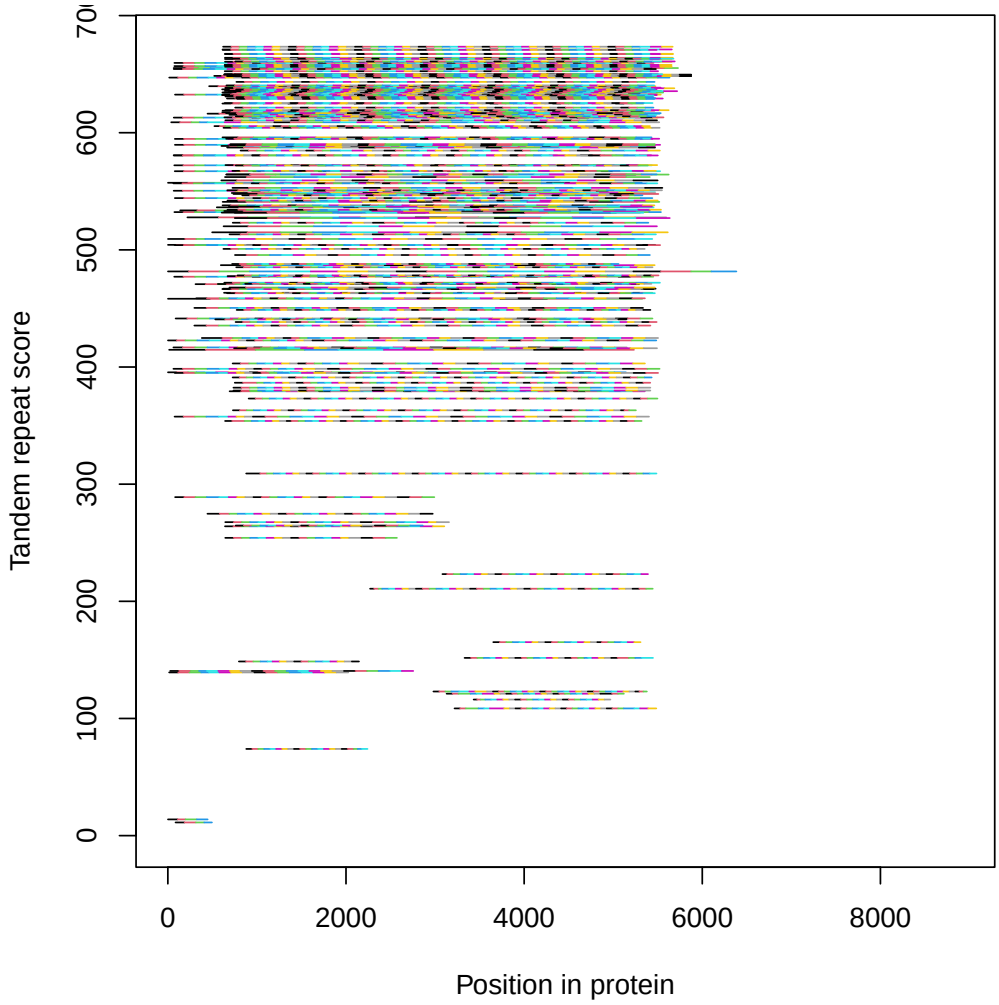

Figure 3: The different repeats detected in the human protein Obscurin.

5 Inferring the History of Repeats in Obscurin

It is feasible to infer the order of repeats from a phylogeny built from the repeat sequences. Repeats that duplicated more recently will presumably be more closely related on a phylogenetic tree. To construct a tree, first we must extract the repeats.

```
> i <- which.max(reps[, "Score"])
> seqs <- extractAt(AA[[reps[i, "Index"]]],
          IRanges(reps[[i, "Left"]], reps[[i, "Right"]]))
> seqs <- AlignSeqs(seqs, verbose=FALSE) # align the repeats
> names(seqs) <- seq_along(seqs) # number from left to right
> seqs
AAStringSet object of length 55:
     width seq                                                names
 [1]   215 ----------------------...---------------------- 1
 [2]   215 ----------------------...---------------------- 2
 [3]   215 ----------------------...------------------PAC- 3
 [4]   215 ----------------------...------------------LAR- 4
 [5]   215 ----------------------...------------------VAH- 5
 ...   ... ...
[51]   215 ----------------------...---------------------- 51
[52]   215 ----------------------...---------------------- 52
[53]   215 ----------------------...AVRAGAQACFTCTLSEAVPVGE 53
[54]   215 ----------------------...----GPE-CVVDGLA------- 54
[55]   215 PGETYRFRVAAVGPVGAGEPVHL...---------------------- 55
```

Note that the repeats are numbered from leftmost to rightmost in the tandem repeat. Then we can infer a phylogenetic tree using the `TreeLine` function in DECIPHER:

There is a signal of adjacent repeats branching later from each other in the phylogeny. From this it is possible to hypothesize the set of duplication events that resulted in the observed set and locations of repeats in Obscurin.

```
> d <- DistanceMatrix(seqs)
================================================================================

Time difference of 0 secs
> dend <- TreeLine(myDistMatrix=d, method="NJ", verbose=FALSE)
> dend <- reorder(dend, seq_along(seqs))
> layout(matrix(1:2))
> plot(dend)
> plot(unlist(dend), xlab="Position on tree", ylab="Repeat number")
```

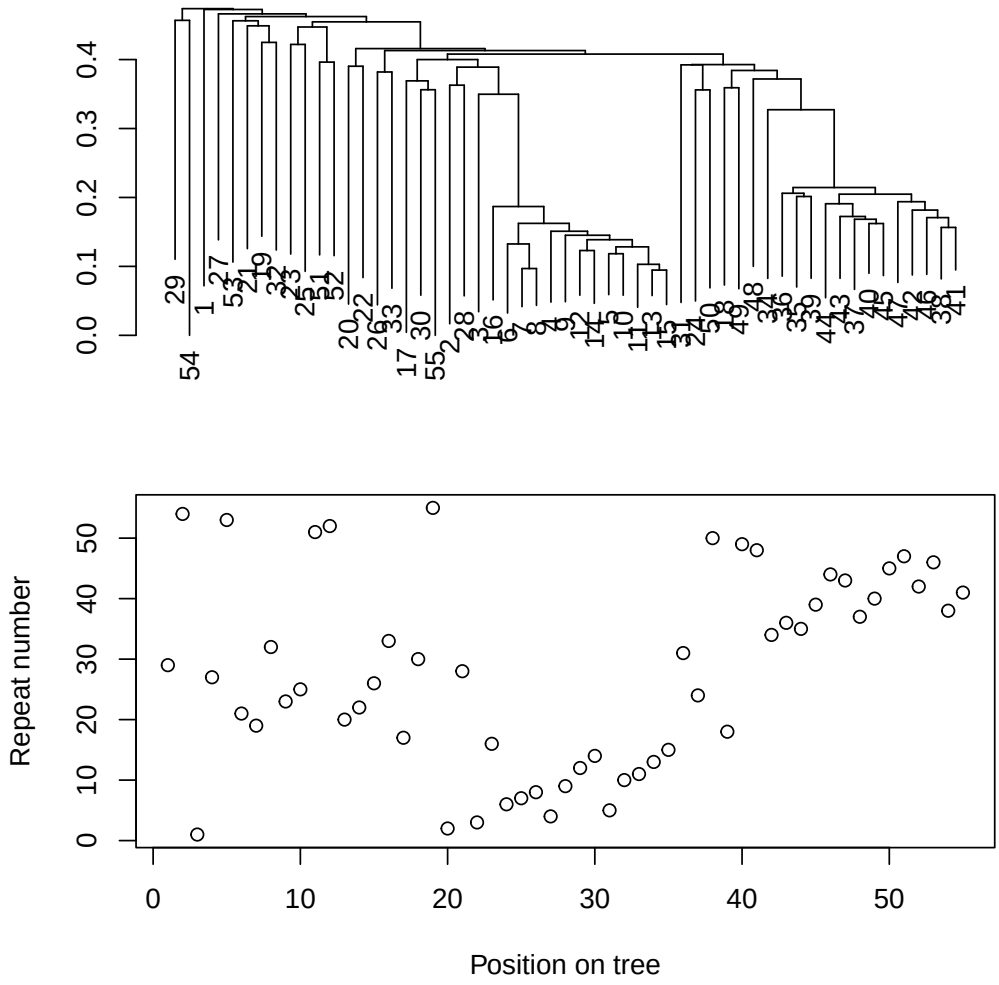

Figure 4: Inferred phylogeny of Ig domain repeats detected in Obscurin.

6 Session Information

All of the output in this vignette was produced under the following conditions:

- R version 4.4.0 (2024-04-24), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 24.04 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so` ; LAPACK version3.12.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: BiocGenerics 0.51.0, Biostrings 2.73.1, DECIPHER 3.1.4, GenomeInfoDb 1.41.1, IRanges 2.39.0, S4Vectors 0.43.0, XVector 0.45.0
- Loaded via a namespace (and not attached): DBI 1.2.3, GenomeInfoDbData 1.2.12, KernSmooth 2.23-22, R6 2.5.1, UCSC.utils 1.1.0, buildtools 1.0.0, compiler 4.4.0, crayon 1.5.2, httr 1.4.7, jsonlite 1.8.8, knitr 1.47, maketools 1.3.0, sys 3.4.2, tools 4.4.0, xfun 0.44, zlibbioc 1.51.1