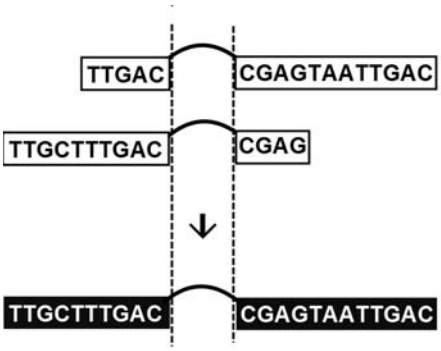
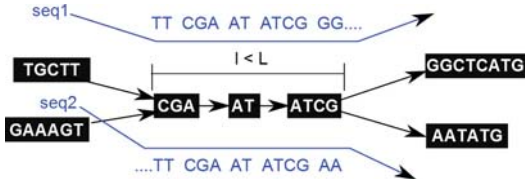
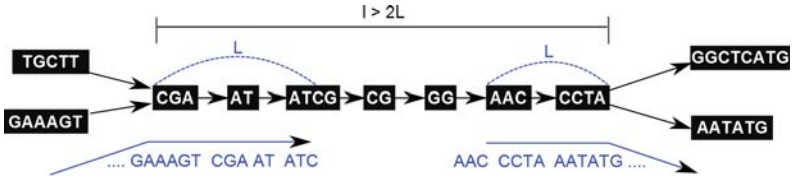


Figure S.1: In filtering stage, RNA-seq reads that have identical splice junctions are merged, and extended in both ends



(a) $\ell < L$

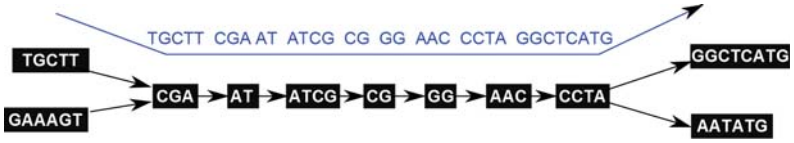


(b) $l \geq 2L$

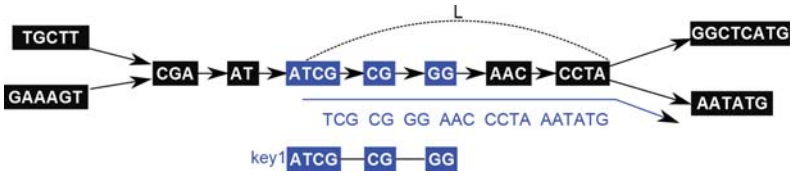
Figure S.2: Combining pairs of sequences that share a prefix and suffix string. First, we identify *overlap-node-pairs* as pairs of *merge* nodes (out degree 1) and *split* nodes (in-degree 1) with length ℓ ($L \leq \ell < 2L$) sequence in between the two. (a) If $\ell < L$, the generated sequences cannot share an identical prefix and suffix. (b) If $l \geq 2L$, the prefix and suffix of generated sequences will not overlap

Comparison with other gene prediction methods

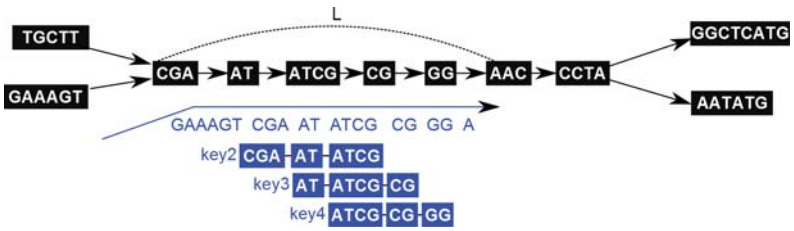
We compared the list of identified novel peptides using our proteogenomics pipeline against various gene prediction results provided from other groups using identical RNA-seq data-sets. The file of gene predictions is available,[17] and includes GeneFinder,[17] single exon gene predictions, predictions based on RNA-seq, and predictions from conserved ORFs (against *C. briggsae*). A



(a) Sequence generated from the first visited nodes



(b) Sequence generated from split nodes



(c) Sequence generated from merge nodes

Figure S.3: Illustration of hashing technique to rapidly identify overlap-node-pairs. (a) For the first visited node path from a start to an end node, the generated sequence is the full path from the corresponding start to end node. This full path cannot be merged with others. (b) In traversing the graph in a depth first fashion, we store all the split nodes present in a candidate list. For each split node u, we hash the prefix string using the first 3 nodes as key(key1), so that each key contains the list of the paths such that prefix of the paths same as the corresponding key. (c) Every time a merge node is encountered in the DFS, we traverse the subsequent path, querying the hash table continuously using 3 node triplets(key2, key3, and key4) to query the hash table. When a match is found (key4 and key1), the hash table returns a list of sequences that corresponding paths starting with the appropriate key. ('TCG'+'CG'+'GG'+'AAC'+'CCTA'+'AATATG'). We search each sequence within the returned sequences, using remaining suffix of the queried sequence. In our example, the remaining sequence is 'A' which appears right after key4. We merge the matched sequence with queried sequence and output to a FASTA file.

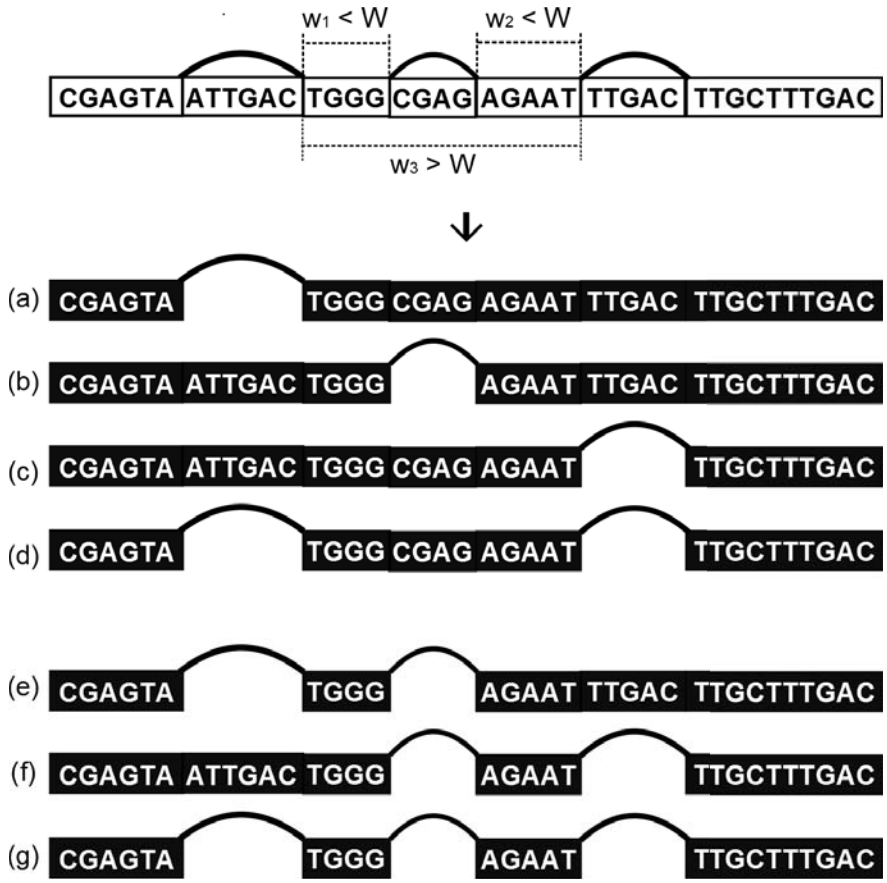


Figure S.4: Description of parameter W. In this example, W is set to 10bp. From the splice graph all possible combinations of the resulting sequences, considering all splicings, total 7 as shown above ((a) through (g)). If multiple spice edges exist within Wbp, and only when the corresponding node has a following consecutive node, then the splice path will be ignored. As a result, (a), (b), (c), and (d), are converted and expressed to the FASTA file. On the other hand, (e), (f), and (g) are discarded.

```
Query  1375  NLVTPLFGILIRARCYRYIIVSDIEKAFHQVRLQKAFRNVTQFLWIQDPSKPTVEDNLCR
             N++TP+FGIL+R R     I+V+DIEKAFHQVRLQ  FRNVT FLW++D + P   DN+
Sbjct  1383  NMITPIFGILVRVRFPPIIVVADIEKAFHQVRLQPEFRNVTMFLWLKDVTAPATADNIQV
```

Figure S.5: Alignment result of novel gene example. The highlighted region corresponds to the alignment of identified peptide 'R.CYRYIIVSDIEKAFHQVRLQKAFR.N' against the sequence of hypothetical protein CRE_ 09558 [Caenorhabditis remanei].

Table S.1: Overall statistics of splice graph data structure

Number of components(G)	116355
Number of nodes	652936
Number of edges	337648
Average node length	57.00 bp
Average number of edges per node	0.44

total of 688 novel peptides were matched to 1194 different predicted gene sequences (Table S.2).

Table S.2: Number of overlapping sequences between identified novel peptides using our proteogenomics pipeline versus protein sequences generated from other gene prediction methods.

Prediction methods[17]	# of overlapped sequence
GeneFinder	286
Conserved *C. briggsae* ORFs	364
RNA-seq data	543
Single exon gene predictions	1

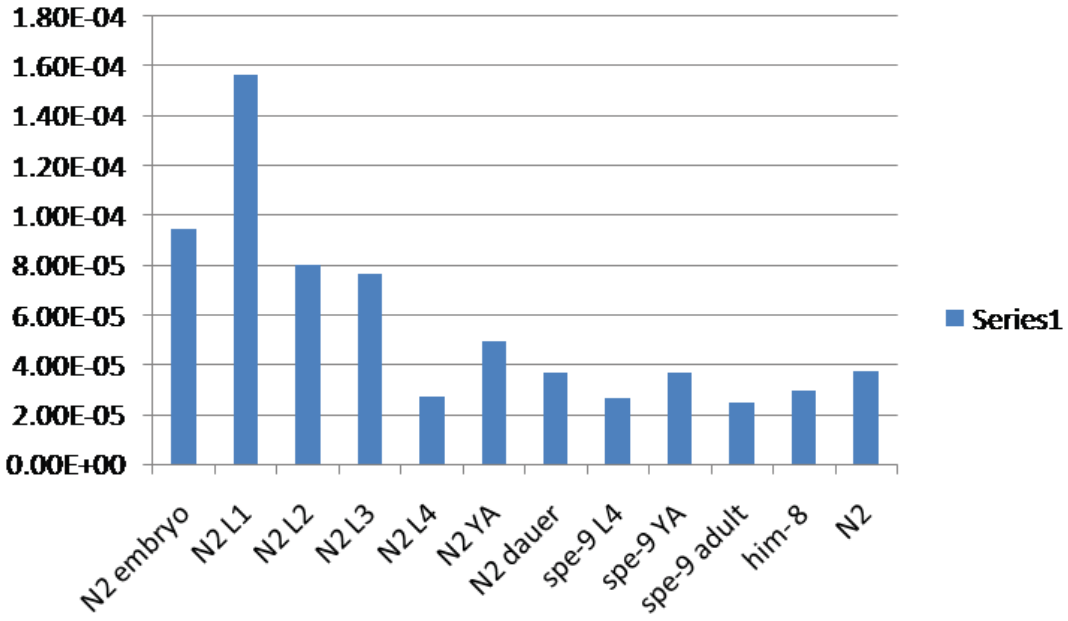


Figure S.6: Translated UTR spectral counts throughout different developmental stages

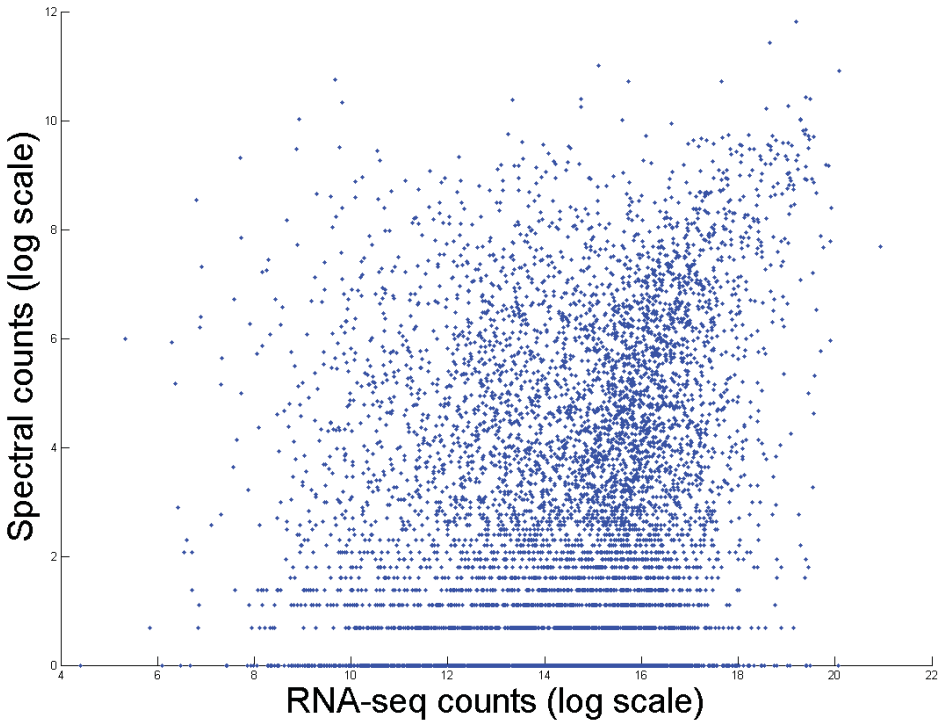


Figure S.7: RNA vs peptide transcription level

Calculation of split mapped coordinates from CIGAR string in SAM file format

The CIGAR string of the SAM format file is used to determine splice junctions. For example, consider a match, starting at coordinate x, with the accompanying CIGAR string given by '35M1000N35M' which is translated to "match 35bp", "skip 1000bp", followed by "match 35bp". We convert this string to two GFF lines, denoting the intervals $[x, x+35]$ and $[x+1035, x+1070]$. In this example, $[x+35, x+1035]$ represents a splice junction.

Detailed RNA-seq methods

RNA-seq Alignment methods:

Step 1:

1. identify all reads beginning with at least 4 TTs
2. identify all reads that begin with at least 6 bases of SL on the front
3. identify adaptor sequence on the 5' and 3' ends of the reads
4. align the reads against the WS220 genome using cross-match
5. align the reads against the AG1003 aggregate genelet transcriptome (transformed into WS220 coordinates) using cross-match

From combining the information from the output of these steps, if z<=5 bases on either end of the read are unclassified/unaligned, then the read is considered to be mapped.
The non-C. elegans worms, C. briggsae and C. remanei, were searched against the WS225 database for each of those genomes. C. japonica and C. brenneri, were searched against the WS227 database.

Step 2:

Next, all reads that had at least 30 bases of match to the genome but were not yet successfully placed, are aligned to the WS220 genome using splice-aware cross-match. Those results are in-

tegrated with the alignments to step 1 to again decide which reads are now considered to be fully mapped.

Step 3:

Reads still not placed but with at least 30 bases of match in the genome, are aligned against a splice junction database using cross-match. The splice junction database contains all confirmed and predicted splice junctions (wormbase and RNAseq etc.) plus all possible novel combinations of those junctions (within 4kb of one another) with 75 bases appended on either side of the junction. Combine these alignments with the information from step 1 to determine if the read is fully accounted for (<=5 bases unaligned from either end).

Step 4:

For reads still not placed, look for multi-segment alignments from bwasw that suggest multiple as yet unpredicted exon pairs and identify splice junctions to join those multi-segment alignments. Combine these alignments with the information from step 1 to determine if the read is fully accounted for (<=5 bases unaligned from either end).

Comparison of spectra dataset used in this study with Merrihew *et al.* (2008)[17]

The Merrihew *et al.* (2008)[17] paper used different fractionation methods, samples and data analysis than the current submission to discover novelty.

The 2008 paper used biochemical fractionation of all stages of C. elegans to improve identifications while the new dataset uses molecular weight fractionation of different stages of C. elegans. The molecular weight fractionation gives us information about the protein before digestion and giving us the potential to map the peptides back to different isoforms. Sampling the different stages of C. elegans improves the identifications and provides information about when proteins are expressed.

The 2008 data analysis relied on a search database made up of the following components: Wormbase (version WS150) protein-coding genes, less conservative predictions from a version of Genefinder, and intergenic ORFs from Wormbase (version WS130) greater than 30 codons with homology between C. elegans and C. briggsae. All of the above components are outdated and most, if not all of the novel findings from the 2008 paper have been confirmed by other experimental methods provided by the modENCODE project. Wormbase is currently on version WS236 which incorporates all these modENCODE findings. This paper describes a different method for assessing novelty using a non-redundant compact database of information from RNA-seq reads to identify novel events in mass spectrometry data. The 2008 paper used some RNA-seq data from the Green lab but it only used the data to confirm the novel events found based on our database search. Also when the 2008 paper was written the Green lab had only RNA-seq data for part of the C. elegans genome.

Additionally our chromatography conditions and mass spectrometers have improved tremendously since 2008. The new data was collected using nano-flow liquid chromatography and using a mass spectrometer with higher resolution, increased sensitivity and faster scanning. The 2008 data used standard flow liquid chromatography and a standard mass spectrometer.

Proof of correctness and completeness in applying Rule1, Rule2, and Rule3

We use three rules to eliminate shared sub-paths.

1. For a pair of paths, xz and yz with a shared string z, we generate two FASTA strings xz, and $y \cdot \text{pref}_L(z)$, where $\text{pref}_L(z)$ denotes a length $L-1$ prefix of string z.
2. For a pair of paths, xz and xy with a shared prefix x, we generate two FASTA strings xz, and $\text{suff}_L(x) \cdot y$, where $\text{suff}_L(x)$ denotes a length $L-1$ suffix of string x.
3. For paths xy and yz, which have a prefix-suffix match with $y \geq L$, generate the FASTA string xyz.

Claim: Applying rules 1,2, and 3, doesn't violate completeness and correctness

Proof: Let G be a splice graph with nodes and edges. Each node represents exons containing sequence of nucleotides, and edges represents the possible event of splicing.

First we'll define the followings,

- S is a set of every sequences from graph G.
- S_1 is a set of sequences from S with applying rule 1.
- S_2 is a set of sequences from S_1 with applying rule 2.
- S_3 is a set of sequences from S_2 with applying rule 3.
- $S_\alpha(l)$ is a set of length l sequences from S_α.

It is clear that $S(l)$ contains every length l sequences that can be generated from G, and doesn't have any sequences that cannot be generated from G. So, the claim will be satisfied if $S(l) = S_3(l)$

To show $S(l) = S_3(l)$, we need to show $S(l) \supset S_1(l) \supset S_2(l) \supset S_3(l)$ and $S(l) \subset S_1(l) \subset S_2(l) \subset S_3(l)$

Because of rule 1, 2 eliminating the sequence or subsequence of elements in S_1 and S_2, $S(l) \supset S_1(l) \supset S_2(l)$ is clear. Also, rule 3 may produce extra length l path during the combining procedure, $S_2(l) \subset S_3(l)$ is also clear.

What we need to show are the following,

1. $S(l) \subset S_1(l)$
2. $S_1(l) \subset S_2(l)$
3. $S_3(l) \subset S_2(l)$

1. $S(l) \subset S_1(l)$

It is clear that x is an element of $S_1(l)$ if S_1 has at least one element y which has x as a subsequence. So, it is enough to show that for $\forall x \in S(l)$, $\exists y$ such that $y \in S_1$ and x is a subsequence of y. Recall that S_1 is generated from S with applying rule 1 which preserve one sequence for the shared suffix. This means that rule 1 eliminates the suffix only when there is at least one sequence which contains the same suffix. Hence y exists in S_1.

2. $S_1(l) \subset S_2(l)$

Recall that every element in S_1 has distinct length l suffix and S_2 is generated from S_1 with rule 2. So $\mid S_2 \mid = \mid S_1 \mid$. Set a bijection between S_2 and S_1 which has same length l suffix. Then the only difference between these sets is part of the prefix that is eliminated by rule 2. But rule 2 also never eliminates the prefix until one of the other elements in S_2 has same prefix. Therefore every length l sequence in $S_1(l)$ is also in $S_2(l)$.

3. $S_3(l) \subset S_2(l)$

Rule 3 is applied only when two elements in S_2 shares the node as their suffix and prefix. So, newly generated length l sequences have their corresponding location in graph G. By 1 and 2, $S_2(l)$ have every length l sequnce in G. So there is no element x such that $x \in S_3(l)$, $x \notin S_2(l)$.

Proof of correctness and completeness in DFS algorithm implementation of Rule1, Rule2, and Rule3

Claim: Our algorithm doesn't violate the completeness and correctness.

Proof: From above, we have shown that application of Rule 1, 2, and 3 doesn't violate the constraints. Here, we want to show our algorithm correctly apply rule 1, 2, and 3.

We begin with the two functions used in our implementation which follows conventional DFS algorithm. DFŚ: An algorithm same as DFS, but stop when it evumerate visited node

DFSFiniteLength : An algorithm same sa DFS, but stop when it enumerate length L further from desired node.

Rule 1:

For Rule 1, we will show this in a two step.

Step 1: For a certain merge node n, if every input edges are retrived, then keep search DFS for one edge and call DFSFiniteLength for others. We want to show this is same as application of Rule 1 for the sequence set generated from n by DFS.

Step 2: For every merge node in G, DFSŕetrieve every input edges of all merge nodes.

If Step 1 and Step 2 are true, DFSánd DFSFiniteLength correctly apply rule 1 for every merge nodes. Step 1 and Step 2 are shown as the following.

1. Assume we have merge node n and multiple input edges $e_1, e_2, ..., e_k$.

Define:

- path p_i : a path that passes through e_i and stops at node n.
- set E_i : set of paths such that all elements in the set have p_i as a prefix and are a result of enumeration from n by DFS.

$| E_i |$ are all identical because they are generated at same node n by DFS. Therefore, we can set a bijenction for any element from E_i to E_j which shares the suffix. Keep E_1, and for $E_2, ..., E_k$,

eliminate the suffix of all elements except $(L-1)$ from corresponding sequence of n.
This is the same as application of Rule 1 for same merge node n.

2. Define: DFS´ is an algorithm the same as DFS, but stopping when it enumerates a visited node. If we color the edges enumerated by DFS´ , then we can easily see that all edges in G will be colored. This means that every merge node in G will be visited by DFSfrom all its incoming edges.

By 1. and 2., Rule 1 is implemented except for some path set that shares the suffix but does not share the path.

Rule 2:

Whenever DFS´ visit splitting node during the enumeration, every sequence set from that node will share the prefix. Therefore keep one and eliminate the prefix except $(L-1)$ from the node is same as application if Rule 2 for that node.
DFSánd DFSFiniteLength apply this for every splitting node, hence Rule 2 is implemented for all splitting nodes.

Rule 3:

Candidate pair set contains every possible prefix-suffix 'coordinate' overlap which shares the sequence, so combining prefix-suffix pair in candidate pair set implements Rule 3.

Note that our algorithm applies Rule 1, Rule 2, and Rule 3 only when sequences have 'coordinate' overlap (having only sequence level overlap does not satisfy this condition, and having coordinate overlap guarantees that the sequences have sequence level overlap within the same reference DNA

system). This is different then Edwards and Lippert[16] where they merge all overlapping sequences (less than a certain parameter), and do not consider coordinate information. Again, unlike Edwards and Lippert,[16] our method uses a genomic coordinate-based data structure(represented in base pairs) rather than minimizing the amino acid sequence overlap. We claim that for proteogenomic analysis the coordinate based approach is more appropriate since it can easily reconstruct the original genomic coordinate of the identified peptide.