Positional proteomics in R

Dr. Tobias Kockmann, D-BIOL, ETHZ

Annotating peptides by position

- Problem: Find all perfect matches for a large collection of peptides (e.g. 20 k) in a reference proteome containing thousands of entries
- The naive approach takes hours/days to compute!
- neXtProt Peptide-to-protein mapper can not do it for more than 1000 queries (https://www.nextprot.org/tools/unicity-checker)…presented at Swiss Proteomics meeting.
- Solution: AhoCorasickSearch in R by Matt Chambers

Code bricks

```
# The Magic ####
# -1- use AhoCorasick to search for perfect matche
library("AhoCorasickTrie")
system.time(l <- AhoCorasickSearch(keywords = s$s.pep, text = proteins, alphabet = "aminoacid", groupByKeywor
```

A full proteome !

All quantified stripped peptide sequence

```
> system.time(l <- AhoCorasic
UE))
   user  system elapsed
  1.274   0.008   1.285
>
```

1.2 s on my laptop !!!

Next steps

- Flatten list-of-lists into dataframe using recursive split+apply+combine

```r
#helper functions to transform list returned by ACS to data frame
sec.split <- function(x){
  y <- ldply(.data = x, .fun = function(x){data.frame(prot = x$Text, offset=x$Offset)})
  return(y)
}
as.df <- function(x){
  y <- ldply(.data = x, .fun = sec.split, .id = "s.pep")
  return(y)
}
df <- as.df(l)
```

List-of-list is split in a recursive manner…call of ldply in ldply

- merge with query table by SQL-type left join

```r
# join annotations and comparision results
annotated.results <- dplyr::left_join(x = results, y = anno, by = "s.pep")
```

The central data structure – IRanges

- mapped peptides can be represented as IRanges in R
- Uses Bioconductor IRanges infrastructure (S4 Objects)
- Each protein of a proteome is represented by one list entry -> list of IRanges
- Any attributes of peptides become element metadata in IRanges object.
- Data structure is build from a data frame using the split+apply+combine principle implemented in the plyr package.

Code bricks

Uses power of split+apply+combine and can be extended easily to mc processing !!!

```r
df2IRangesList <- function(x, u = "proteome"){
  x <- x[!is.na(x$prot),] #remove cases with missing protein annotation, can not split here!!!
  y <- dlply(.data = x, .variables = "prot", .fun = failwith(NA, .make.ranges), .progress = "text")
  y <- as(y, "RangesList")
  universe(y) <- u
  return(y)
}
```

```r
## central function to confert tidy df into a list of ranges
.make.ranges <- function(x){
  ## helper function for split+apply+combine
  ## transforms df into ranges object with metadata
  r <- IRanges(start = x$offset, width = nchar(x$s.pep), names = x$ID)
  elementMetadata(r) <- x[c(4,9,13:15,19:22)]
  return(r)
}
```

Peptide IDs are retained

Any kind of metadata can be incorporated

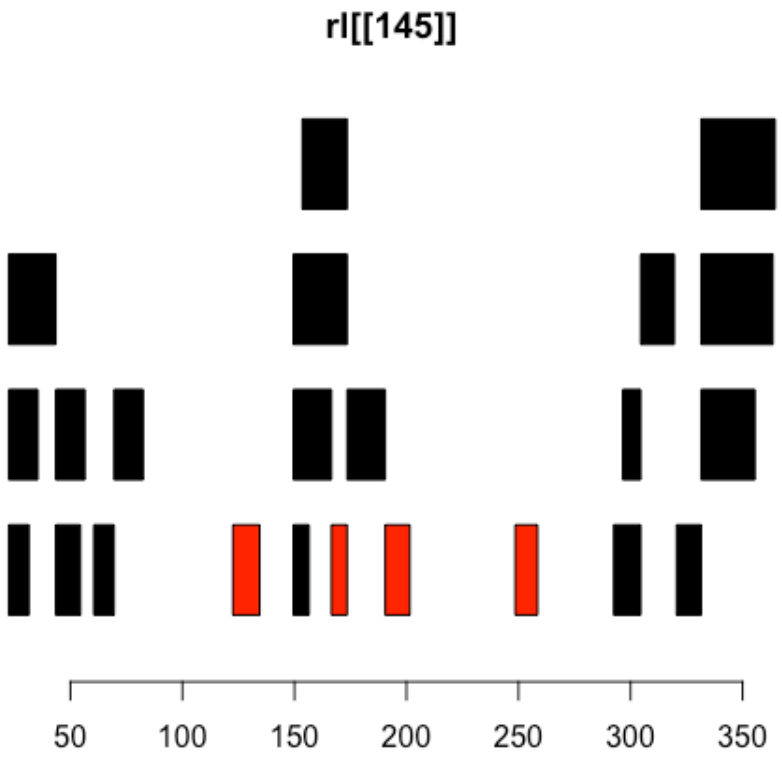
Example

```
Console ~/Documents/RStudio/Projects/p2095/
+   return(y)
+ }
> rl <- df2IRangesList(annotated.results, "UP000000589")
  |======================================================================================| 100%
> rl[[1]]
IRanges object with 3 ranges and 9 metadata columns:
           start       end     width |    log2FC adj.pvalue                          s.pep
       <integer> <integer> <integer> | <numeric>  <numeric>                    <character>
  1748       378       409        32 |      -Inf          0 GSYGDLGGPIITTQVTIPKDLAGSIIGKGGQR
  3424       383       409        27 |       Inf          0      LGGPIITTQVTIPKDLAGSIIGKGGQR
  8031        38        46         9 |      -Inf          0                          NTDEMVELR
                                                 m.pep nterm.label    first.aa     last.aa
                                               <character> <character> <character> <character>
  1748   GSYGDLGGPIITTQVTIPK[+28.0]DLAGSIIGK[+28.0]GGQR        free           G           R
  3424 L[+28.0]GGPIITTQVTIPK[+28.0]DLAGSIIGK[+28.0]GGQR       label           L           R
  8031                                          NTDEMVELR        free           N           R
           up.aa     down.aa
       <character> <character>
  1748          R           I
  3424          D           I
  8031          R           I
>
```

What is so great about IRanges?

- One can use all the fancy functions to look for patterns (Ranges that have a certain orientation towards one another, but are tight to the same reference (protein).
- One can process things like overalps, unios, …

Example – neoN/neoC pairs



Can you see pairs of black and red peptides oriented head-to-tail at zero distance?

Code bricks

Split+apply+ combine

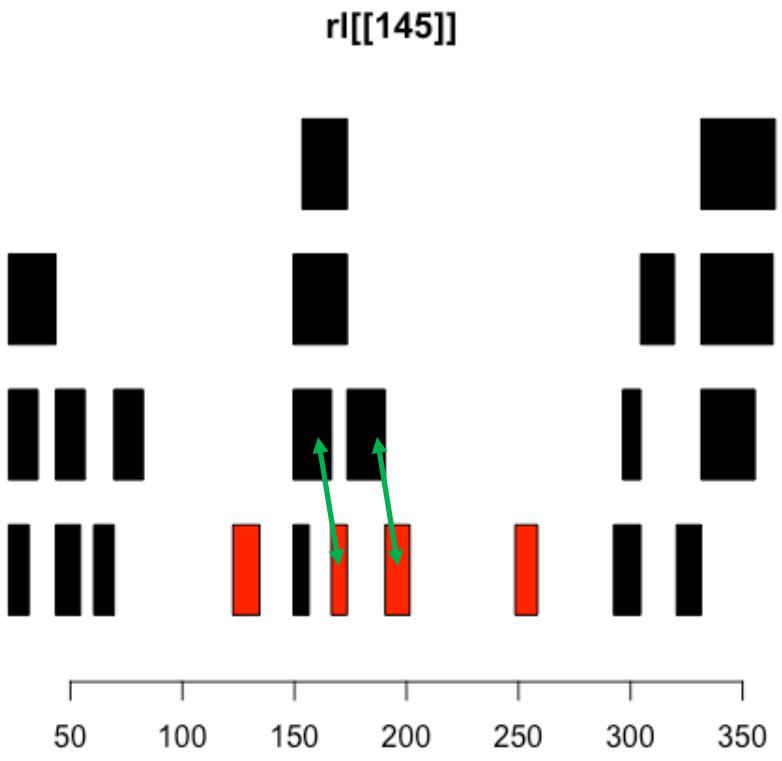
```
filter.neoCN.pairs <- function(x){
  y <- llply(.data = x, .fun = failwith(default = NA, f = .experimental), .progress = "text")
  return(y)
}
```

Tests for specificity and orientation

```
.experimental <- function(x){
  m <- elementMetadata(x)
  a <- x[m$nterm.label == "label" & m$last.aa %in% work.prot.spec & m$up.aa %in% test.prot.spec]
  b <- x[m$nterm.label == "free" & m$last.aa %in% test.prot.spec & m$up.aa %in% work.prot.spec]
  p <- findOverlapPairs(a, b, maxgap=1L)
  #p <- punion(subset(p, start(first) == end(second) + 1L | end(first) == start(second) - 1L))
  p <- punion(subset(p, start(first) == end(second) + 1L))
  elementMetadata(p) <- data.frame(nterm.label = "neoNC")
  return(p)
}
```

Returns intervals corresponding to pairs

```
> neoCN <- filter.neoCN.pairs(rl)
  |======================================================================================================| 100%
> neoCN[[145]]
IRanges object with 2 ranges and 1 metadata column:
           start       end     width | nterm.label
       <integer> <integer> <integer> |    <factor>
  2495       174       201        28 |       neoNC
  3904       150       173        24 |       neoNC
> 
```



…and one can apply IRanges as mask on the sequence

AAStrings instance

IRanges representing pairs

```
> Views(ref.proteome[[poi]], neoCN[[145]])
  Views on a 364-letter AAString subject
subject: MPHPYPALTPEQKKELSDIAHRIVAPGKGILAADESTGSIAKRLQSIGTE...GKKENLKAAQEEYIKRALANSLACQGKYTPSGQSGAAASESLFISNHAY
views:
    start end width
[1]   174 201    28 [YASICQQNGIVPIVEPEILPDGDHDLKR]
[2]   150 173    24 [CVLKIGEHTPSALAIMENANVLAR]
>
```

Counting on IRangesList instances...so easy!

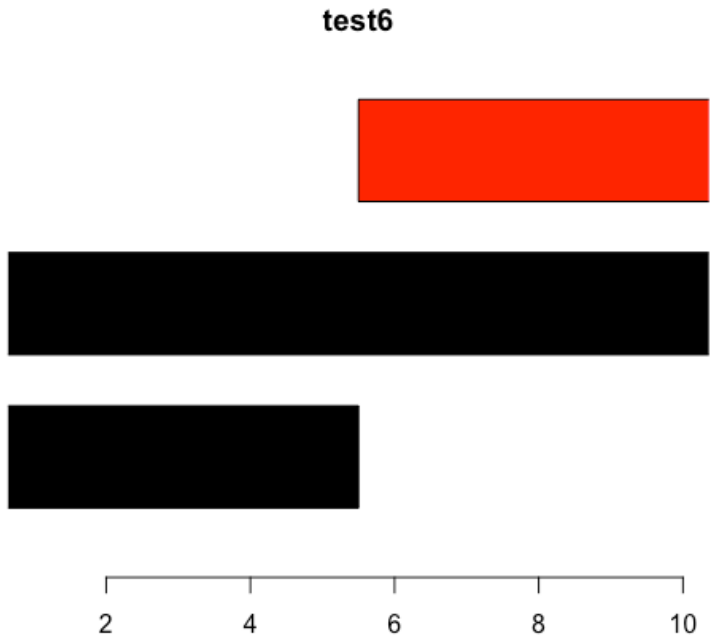
```
Console ~/Documents/RStudio/Projects/p2095/
> count.ranges <- function(x){
+   d <- ldply(.data = x, .fun = failwith(NA, length), .progress = "text")
+   return(d$V1)
+ }
> sum(count.ranges(neoCN))
  |=================================================================================================| 100%
[1] 50
> count.ranges(neoCN)
  |=================================================================================================| 100%
   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [53] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 [105] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 2 2 1 0 1 2 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0
 [157] 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [209] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [261] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Similar functions for

- neoN-spanning pairs
- neoC-spanning pairs

Find triplets of neoCN & spanning peptide

- Find to start-to-end overlaps between neoCN pairs and spanning peptides



To do

- Include quant. element metadata into filter functions
  - log2FC
  - adj.pvalue
  - neoNC -> values need to be bigger than critical values
- Make filter function work with unknown test protease specificity.
  - incl. with NA
  - test if NA, if yes do not incl. in selection