sequence matching - test data

Kai He

Purpose: Match amino acid sequences identified by mass spectrometry to all

dependencies: seqinr, dplyr, tidyr, ggplot2, "assertthat", "stringr"

```
lapply(c("dplyr", "seqinr", "tidyr", "ggplot2","assertthat","stringr"), library, character.only = T)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'seqinr'

## The following objects are masked from 'package:dplyr':
##
##     count, query

## [[1]]
## [1] "dplyr"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
## [1] "seqinr"    "dplyr"     "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "tidyr"     "seqinr"    "dplyr"     "stats"     "graphics"
##  [6] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[4]]
##  [1] "ggplot2"   "tidyr"     "seqinr"    "dplyr"     "stats"
##  [6] "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [11] "base"
##
## [[5]]
##  [1] "assertthat" "ggplot2"    "tidyr"      "seqinr"     "dplyr"
##  [6] "stats"      "graphics"   "grDevices"  "utils"      "datasets"
## [11] "methods"    "base"
##
## [[6]]
##  [1] "stringr"    "assertthat" "ggplot2"    "tidyr"      "seqinr"
##  [6] "dplyr"      "stats"      "graphics"   "grDevices"  "utils"
## [11] "datasets"   "methods"    "base"
```

default arguments for automated build

```
lkup_path <- "../data/testData.csv" # use testdata
ref_path <- "../data/uniprot-all.fasta" # reference fasta file
```

read csv file from test data:

```
ori_lkup <- read.csv(lkup_path)
# ori_lkup <- filter(ori_lkup, ST.2 != "index")
```

show head of data:

```
head(ori_lkup)
```

```
##   index spectrum pulp.specimen preTAILS...TAILS probability start.scan
## 1    NA       NA            NA               NA          NA         NA
## 2    NA       NA            NA               NA          NA         NA
## 3    NA       NA            NA               NA          NA         NA
## 4    NA       NA            NA               NA          NA         NA
## 5    NA       NA            NA               NA          NA         NA
## 6    NA       NA            NA               NA          NA         NA
##   expect ions                  peptide protein leading.protein
## 1     NA   NA R.PPSPALASVLLALLLSGAAR.A      NA          P24158
## 2     NA   NA    K.PTHVNVSVVMAEVDGTCY.      NA          P01876
## 3     NA   NA          MFGGPGTASRPSSSR      NA          P08670
## 4     NA   NA           TYSLGSALRPSTSR      NA          P08670
## 5     NA   NA                    SSAVR      NA          P08670
## 6     NA   NA      LLQDSVDFSLADAINTEFK      NA          P08670
##   calc.neutral.pep.mass precursor.neutral.mass mz.ratio assumed.charge
## 1                    NA                     NA       NA             NA
## 2                    NA                     NA       NA             NA
## 3                    NA                     NA       NA             NA
## 4                    NA                     NA       NA             NA
## 5                    NA                     NA       NA             NA
## 6                    NA                     NA       NA             NA
##   massdiff ppm fval pI retention.time.sec num.tol.term X.missed.cleavages
## 1       NA  NA   NA NA                NA           NA                 NA
## 2       NA  NA   NA NA                NA           NA                 NA
## 3       NA  NA   NA NA                NA           NA                 NA
## 4       NA  NA   NA NA                NA           NA                 NA
## 5       NA  NA   NA NA                NA           NA                 NA
## 6       NA  NA   NA NA                NA           NA                 NA
##   ionscore identity.score homology.score hyperscore nextscore bscore
## 1       NA             NA             NA         NA        NA     NA
## 2       NA             NA             NA         NA        NA     NA
## 3       NA             NA             NA         NA        NA     NA
## 4       NA             NA             NA         NA        NA     NA
## 5       NA             NA             NA         NA        NA     NA
## 6       NA             NA             NA         NA        NA     NA
##   yscore raw denovo SpecEValue IsotopeError xcorr deltacn deltacnstar
## 1     NA  NA     NA         NA           NA    NA      NA          NA
## 2     NA  NA     NA         NA           NA    NA      NA          NA
## 3     NA  NA     NA         NA           NA    NA      NA          NA
```

```
## 4     NA  NA     NA          NA           NA    NA      NA          NA
## 5     NA  NA     NA          NA           NA    NA      NA          NA
## 6     NA  NA     NA          NA           NA    NA      NA          NA
##   spscore sprank pI_zscore RT RT_score
## 1      NA     NA        NA NA       NA
## 2      NA     NA        NA NA       NA
## 3      NA     NA        NA NA       NA
## 4      NA     NA        NA NA       NA
## 5      NA     NA        NA NA       NA
## 6      NA     NA        NA NA       NA
```

select only relevant columns. the name "leading protein" is converted to "leading.protein" by R default

```
lkup <- ori_lkup %>%
  select(peptide
         # = X.5
         , leading.protein
         # = X.7
         ) %>%
  # filter out proteins with no sequence information, doesn't matter for test but will become important
  filter(peptide != "") %>%

  # create new column indicating whether the N-terminus is blocked before tryptic digestion
  mutate(Ncapped = 0)

lkup$Ncapped[grepl("n",ignore.case = FALSE,lkup$peptide)] <- 1
```

variable that stores the peptide sequence in character

```
pepseq <- as.character(lkup$peptide)

# show peptide sequences:

head(pepseq)
```

```
## [1] "R.PPSPALASVLLALLLSGAAR.A" "K.PTHVNVSVVMAEVDGTCY."
## [3] "MFGGPGTASRPSSSR"          "TYSLGSALRPSTSR"
## [5] "SSAVR"                    "LLQDSVDFSLADAINTEFK"
```

temp variable for formatting sequence

```
tmp <- sub(".?\\.","",pepseq)
tmp <- sub("n\\[[0-9]*\\.?[0-9]*\\]","",tmp)
tmp <- gsub("\\[[0-9]*\\.?[0-9]*\\]","",tmp)
tmp <- sub("\\..*","",tmp)

# show tmp:
head(tmp)
```

```
## [1] "PPSPALASVLLALLLSGAAR" "PTHVNVSVVMAEVDGTCY"   "MFGGPGTASRPSSSR"
## [4] "TYSLGSALRPSTSR"       "SSAVR"                "LLQDSVDFSLADAINTEFK"
```

Add the formatted sequence as a new column named Lseq

```
# lead.protein <- sub(,"",lkup$Acc_id)

lkup <- lkup %>%
  mutate(Lseq=tmp) %>%
  rename(Acc_id = leading.protein) %>%
  filter(nchar(as.character(Acc_id))<=8) %>%
  mutate(Acc_id = sub("\\-[0-9]*","",Acc_id))

# write cleaned up lookup sequence to results folder
write.csv(lkup, "../results/lkup.csv")
```

show cleaned data:

```
head(lkup)
```

```
##                     peptide Acc_id Ncapped                 Lseq
## 1 R.PPSPALASVLLALLLSGAAR.A P24158       0 PPSPALASVLLALLLSGAAR
## 2    K.PTHVNVSVVMAEVDGTCY. P01876       0   PTHVNVSVVMAEVDGTCY
## 3          MFGGPGTASRPSSSR P08670       0      MFGGPGTASRPSSSR
## 4           TYSLGSALRPSTSR P08670       0       TYSLGSALRPSTSR
## 5                    SSAVR P08670       0                SSAVR
## 6      LLQDSVDFSLADAINTEFK P08670       0  LLQDSVDFSLADAINTEFK
```

Load and show the ref dataset.

```
orig_ref <- read.fasta(ref_path, seqtype = "AA", as.string = T)

head(orig_ref)
```

```
## $`sp|P31946|1433B_HUMAN`
## [1] "MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKNVVGARRSSWRVISSIEQKTERNEKKQQMGKEYREKIEAELQDI
## attr(,"name")
## [1] "sp|P31946|1433B_HUMAN"
## attr(,"Annot")
## [1] ">sp|P31946|1433B_HUMAN 14-3-3 protein beta/alpha OS=Homo sapiens GN=YWHAB PE=1 SV=3"
## attr(,"class")
## [1] "SeqFastaAA"
##
## $`sp|P62258|1433E_HUMAN`
## [1] "MDDREDLVYQAKLAEQAERYDEMVESMKKVAGMDVELTVEERNLLSVAYKNVIGARRASWRIISSIEQKEENKGGEDKLKMIREYRQMVETELKL
## attr(,"name")
## [1] "sp|P62258|1433E_HUMAN"
## attr(,"Annot")
## [1] ">sp|P62258|1433E_HUMAN 14-3-3 protein epsilon OS=Homo sapiens GN=YWHAE PE=1 SV=1"
## attr(,"class")
## [1] "SeqFastaAA"
##
## $`sp|Q04917|1433F_HUMAN`
## [1] "MGDREQLLQRARLAEQAERYDDMASAMKAVTELNEPLSNEDRNLLSVAYKNVVGARRSSWRVISSIEQKTMADGNEKKLEKVKAYREKIEKELET
## attr(,"name")
```

```
## [1] "sp|Q04917|1433F_HUMAN"
## attr(,"Annot")
## [1] ">sp|Q04917|1433F_HUMAN 14-3-3 protein eta OS=Homo sapiens GN=YWHAH PE=1 SV=4"
## attr(,"class")
## [1] "SeqFastaAA"
##
## $`sp|P61981|1433G_HUMAN`
## [1] "MVDREQLVQKARLAEQAERYDDMAAAMKNVTELNEPLSNEERNLLSVAYKNVVGARRSSWRVISSIEQKTSADGNEKKIEMVRAYREKIEKELEA
## attr(,"name")
## [1] "sp|P61981|1433G_HUMAN"
## attr(,"Annot")
## [1] ">sp|P61981|1433G_HUMAN 14-3-3 protein gamma OS=Homo sapiens GN=YWHAG PE=1 SV=2"
## attr(,"class")
## [1] "SeqFastaAA"
##
## $`sp|P31947|1433S_HUMAN`
## [1] "MERASLIQKAKLAEQAERYEDMAAFMKGAVEKGEELSCEERNLLSVAYKNVVGGQRAAWRVLSSIEQKSNEEGSEEKGPEVREYREKVETELQGV
## attr(,"name")
## [1] "sp|P31947|1433S_HUMAN"
## attr(,"Annot")
## [1] ">sp|P31947|1433S_HUMAN 14-3-3 protein sigma OS=Homo sapiens GN=SFN PE=1 SV=1"
## attr(,"class")
## [1] "SeqFastaAA"
##
## $`sp|P27348|1433T_HUMAN`
## [1] "MEKTELIQKAKLAEQAERYDDMATCMKAVTEQGAELSNEERNLLSVAYKNVVGGRRSAWRVISSIEQKTDTSDKKLQLIKDYREKVESELRSICT
## attr(,"name")
## [1] "sp|P27348|1433T_HUMAN"
## attr(,"Annot")
## [1] ">sp|P27348|1433T_HUMAN 14-3-3 protein theta OS=Homo sapiens GN=YWHAQ PE=1 SV=1"
## attr(,"class")
## [1] "SeqFastaAA"
```

clean the reference

```
ref <- data.frame(Acc_id = getName(orig_ref),
                  Rseq = rapply(getSequence(orig_ref, as.string = T), c)) %>%
  ### Clean Acc_id.
  mutate(Acc_id = substr(as.character(Acc_id), 4, 9)) %>%
  ### Clean Rseq.
  mutate(Rseq = toupper(Rseq)) %>%
  ### Remove duplicates.
  distinct(Acc_id, Rseq)
```

Write the cleaned reference to results folder

```
write.csv(ref,"../results/ref.csv")
```

show cleaned ref:

```
head(ref)
```

```
##   Acc_id
```

```
## 1 P31946
## 2 P62258
## 3 Q04917
## 4 P61981
## 5 P31947
## 6 P27348
##
## 1           MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKNVVGARRSSWRVISSIEQKTERNEKKQQMGKEYREKIE
## 2 MDDREDLVYQAKLAEQAERYDEMVESMKKVAGMDVELTVEERNLLSVAYKNVIGARRASWRIISSIEQKEENKGGEDKLKMIREYRQMVETELKLICC
## 3           MGDREQLLQRARLAEQAERYDDMASAMKAVTELNEPLSNEDRNLLSVAYKNVVGARRSSWRVISSIEQKTMADGNEKKLEKVKAYREKI
## 4          MVDREQLVQKARLAEQAERYDDMAAAMKNVTELNEPLSNEERNLLSVAYKNVVGARRSSWRVISSIEQKTSADGNEKKIEMVRAYREKIE
## 5         MERASLIQKAKLAEQAERYEDMAAFMKGAVEKGEELSCEERNLLSVAYKNVVGGQRAAWRVLSSIEQKSNEEGSEEKGPEVREYREKVETE
## 6            MEKTELIQKAKLAEQAERYDDMATCMKAVTEQGAELSNEERNLLSVAYKNVVGGRRSAWRVISSIEQKTDTSDKKLQLIKDYREKVESE
```

Merge the lkup and ref datasets by Acc_id.

```
lkup <- read.csv("../results/lkup.csv")
ref <- read.csv("../results/ref.csv")

combo <- merge(lkup,ref, by = "Acc_id") %>%
  ### Keep obs with unique values in Lseq and Rseq only.
  group_by(Rseq) %>%
  filter(!duplicated(Lseq))
  # select(Acc_id,Rseq,Lseq) %>%
  # unique()

combo$X.x = NULL
combo$X.y = NULL
```

Check if all Acc_ids from lkup are in combo. Must be TRUE.

```
assert_that(length(intersect(combo$Acc_id, lkup$Acc_id)) == length(unique(lkup$Acc_id)))
```

```
## [1] TRUE
```

Check if all Acc_ids from ref are in combo. Must be FALSE.

```
assert_that(length(intersect(combo$Acc_id, as.character(ref$Acc_id))) != length(unique(ref$Acc_id)))
```

```
## [1] TRUE
```

Set Match to 1 if Lseq is a substring of Rseq; 0 otherwise.

```
combo$Lseq <- as.character(combo$Lseq)
combo$Rseq <- as.character(combo$Rseq)
```

combo data frame summarising matched Lseqs and Rseqs

```
combo <- combo %>%
  mutate(Match = mapply(grepl,Lseq,Rseq)) %>%
  ### Keep only matched obs.
  filter(Match == TRUE) %>%
  ### Count character length of Lseq and Rseq.
  mutate(Llen = nchar(Lseq),
         Rlen = nchar(Rseq),
         ### Get start and end indices of matched Lseq.
         Lstart = mapply(regexpr,Lseq,Rseq,fixed = TRUE),
         Lend = Lstart + Llen -1) %>%
  ### Remove case with Lstart < 0.
  filter(Lstart>=0) %>%
  ### Sort by Acc_id, Lstart and Lend.
  arrange(Acc_id,Lstart,Lend)
```

Check if all obs are matched. Must be TRUE.

```
assert_that(sum(combo$Match) == nrow(combo))
```

```
## [1] TRUE
```

```
### See which obs are not matched.
which(combo$Match == FALSE)
```

```
## integer(0)
```

Check if each Acc_id is associated with a unique Rseq. Must be TRUE.

```
assert_that(length(unique(combo$Acc_id)) == length(unique(combo$Rseq)))
```

```
## [1] TRUE
```

show combo:

```
head(combo)
```

```
## Source: local data frame [6 x 10]
## Groups: Rseq [2]
##
##   Acc_id               peptide Ncapped              Lseq
##   <fctr>                <fctr>   <int>             <chr>
## 1 P01876 K.PTHVNVSVVMAEVDGTCY.       0 PTHVNVSVVMAEVDGTCY
## 2 P04083  T.n[35.07]SDTSGDFR.N       1           SDTSGDFR
## 3 P04083              NALLSLAK       0           NALLSLAK
## 4 P04083              ALYEAGER       0           ALYEAGER
## 5 P04083               VLDLELK       0            VLDLELK
## 6 P04083             SEIDMNDIK       0          SEIDMNDIK
## # ... with 6 more variables: Rseq <chr>, Match <lgl>, Llen <int>,
## #   Rlen <int>, Lstart <int>, Lend <dbl>
```

get unique reference sequences only.

```
matchedRseq <- combo %>%
  select(Acc_id,Rseq) %>%
  distinct(.keep_all = TRUE)
```

show ref seqs:

```
(matchedRseq)
```

```
## Source: local data frame [6 x 2]
## Groups: Rseq [6]
##
##   Acc_id
##   <fctr>
## 1 P01876
## 2 P04083
## 3 P08670
## 4 P12814
## 5 P24158
## 6 Q16512
## # ... with 1 more variables: Rseq <chr>
```

create a vector of access ids repeated by length of corresponding reference sequences.

```
accid <- sapply(as.integer(rownames(matchedRseq)),
                function(i){
                  replicate(nchar(matchedRseq$Rseq[i]),
                            expr = matchedRseq$Acc_id[i])})
```

show accid

```
str(accid)
```

```
## List of 6
##  $ : Factor w/ 6 levels "P01876","P04083",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ : Factor w/ 6 levels "P01876","P04083",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ : Factor w/ 6 levels "P01876","P04083",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ : Factor w/ 6 levels "P01876","P04083",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ : Factor w/ 6 levels "P01876","P04083",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ : Factor w/ 6 levels "P01876","P04083",..: 6 6 6 6 6 6 6 6 6 6 ...
```

```
summary(accid)
```

```
##      Length Class  Mode
## [1,] 353    factor numeric
## [2,] 346    factor numeric
## [3,] 466    factor numeric
## [4,] 892    factor numeric
## [5,] 256    factor numeric
## [6,] 942    factor numeric
```

store the level of accid in a tmp data frame df.

```
df <- data.frame(ACID <- c(unlist(accid)))

df <- df %>%
  # add Actual access id column
  mutate(Acc_id = as.factor(levels(unlist(accid))[ACID....c.unlist.accid..]),
         # add index for each accid
         index = rownames(df)) %>%
  # arrange by Access id and index
  arrange(Acc_id,index) %>%
  # group by access id.
  group_by(Acc_id) %>%
  # recreate index column, remove the level numbers, create new variable indicating matched or not
  mutate(index = row_number(),
         ACID....c.unlist.accid..=NULL)
```

tmp vector storing start location of each Lseq

```
# Lseq_starts <- (sapply(tdf$Acc_id, function(i) combo$Lstart[combo$Acc_id == i & combo$Ncapped == 0]))
Lseq_starts <- (sapply(df$Acc_id, function(i) combo$Lstart[combo$Acc_id == i & combo$Ncapped == 0]))
```

show Lseq_starts

```
head(Lseq_starts)
```

```
## [[1]]
## [1] 336
##
## [[2]]
## [1] 336
##
## [[3]]
## [1] 336
##
## [[4]]
## [1] 336
##
## [[5]]
## [1] 336
##
## [[6]]
## [1] 336
```

new column storing first matching positions between Lseq and Rseq by getting first start location of matching Lseq

```
df$firstmatch <- sapply(Lseq_starts,function(i) i[[1]][1])
```

divide the Rseq index into groups by first match of Lseq

```
df$group <- replicate(nrow(df),0)
df$group[df$index > df$firstmatch-1] <- 1
```

find strat position of each possible Rseq marked by a K or R

```
df$Rpep_start <- 0
```

get Rpep_index

```
index_lst <- lapply(matchedRseq$Rseq,function(i) {
  which(df$Acc_id == as.character(matchedRseq$Acc_id[which(matchedRseq$Rseq == i)]) &
          df$index %in% unlist(gregexpr("[K,R]",i)))
})


df$Rpep_start[as.integer(unlist(index_lst))+1] <- 1

df <- df %>%   group_by(Acc_id) %>%   mutate(Rpep_index = sapply(index, function(i) sum(Rpep_start[1:i]]
```

rename the tmp dataframe to grouped_peps, make new column stating if the peptide is matched

```
grouped_peps <- df %>%
  group_by(Acc_id) %>%
  mutate(match = Rpep_index %in% Rpep_index[index %in% combo$Lstart[combo$Acc_id == Acc_id[1] & combo$N
```

show grouped_peps:

```
head(grouped_peps)
```

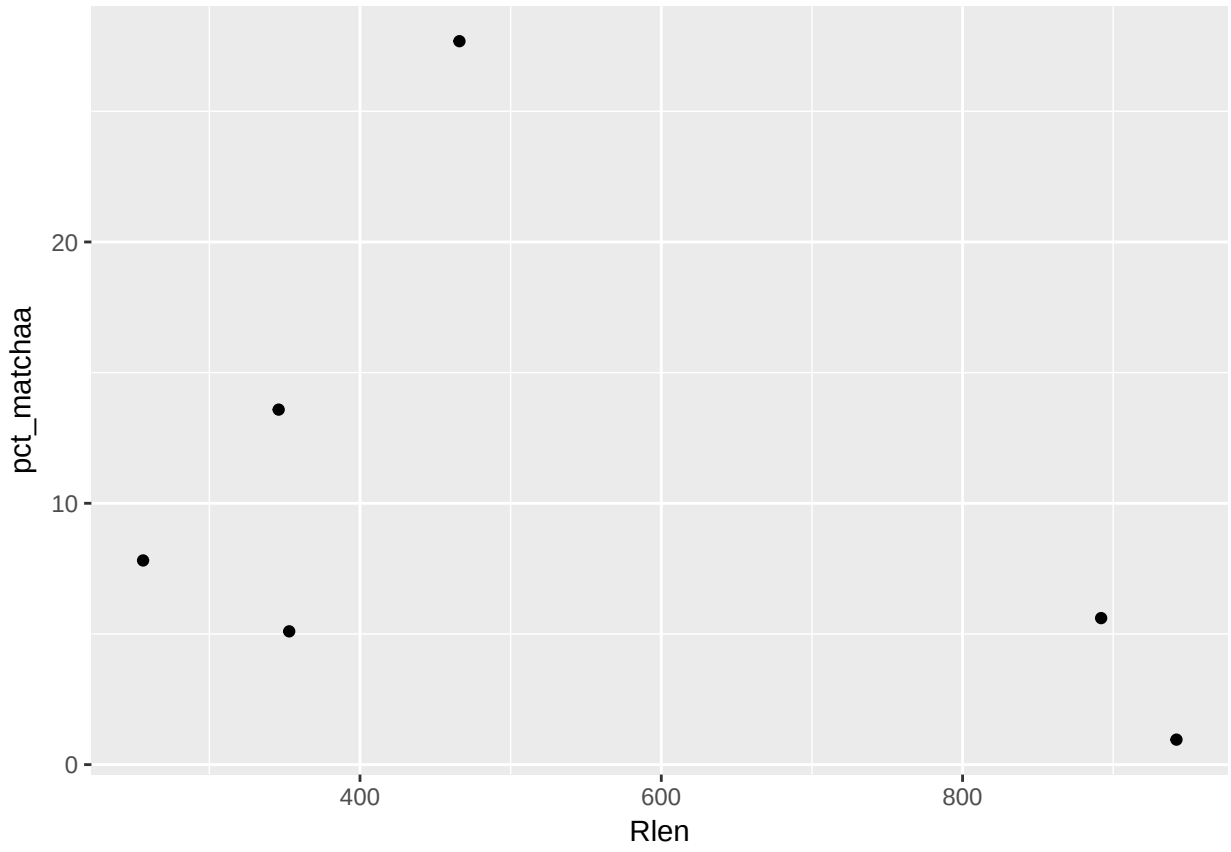
```
## Source: local data frame [6 x 7]
## Groups: Acc_id [1]
##
##   Acc_id index firstmatch group Rpep_start Rpep_index match
##   <fctr> <int>      <int> <dbl>      <dbl>      <dbl> <lgl>
## 1 P01876     1        336     0          0          1 FALSE
## 2 P01876     2        336     0          0          1 FALSE
## 3 P01876     3        336     0          0          1 FALSE
## 4 P01876     4        336     0          0          1 FALSE
## 5 P01876     5        336     0          0          1 FALSE
## 6 P01876     6        336     0          0          1 FALSE
```

basic statistics

- % of amino acids matched [plot as scatterplot vs total number of amino acids]

assume no overlaps between Lseqs (will break Lseqs manually at possible cut sites) % calculated as sum of Llen over length of corresponding Rseq.

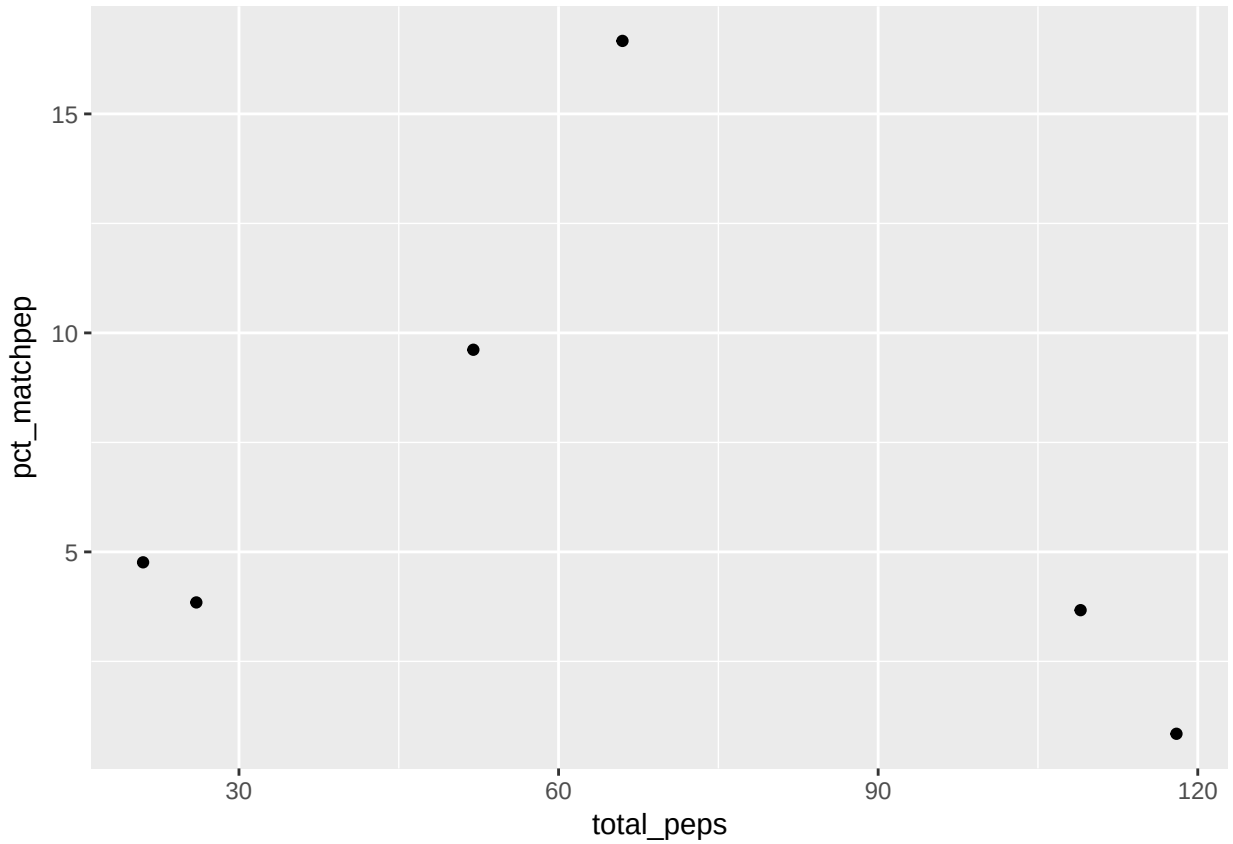
```
matchedRseq %>%
  mutate(pct_matchaa = sum(combo$Llen[combo$Acc_id == Acc_id & combo$Ncapped == 0]) / combo$Rlen[combo$A
         Rlen = combo$Rlen[combo$Acc_id == Acc_id][1]) %>%
  ggplot(aes(x=Rlen, y = pct_matchaa)) +
  geom_point()
```



- % of peptides matched [plot as scatterplot vs total number of peptides]

```
pctpep <- grouped_peps %>%
  group_by(Acc_id) %>%
  summarise(match_peps = nlevels(as.factor(Rpep_index[match])),
            total_peps = nlevels(as.factor(Rpep_index)),
            pct_matchpep = match_peps / total_peps * 100)

ggplot(pctpep,aes(x = total_peps, y = pct_matchpep)) + geom_point()
```

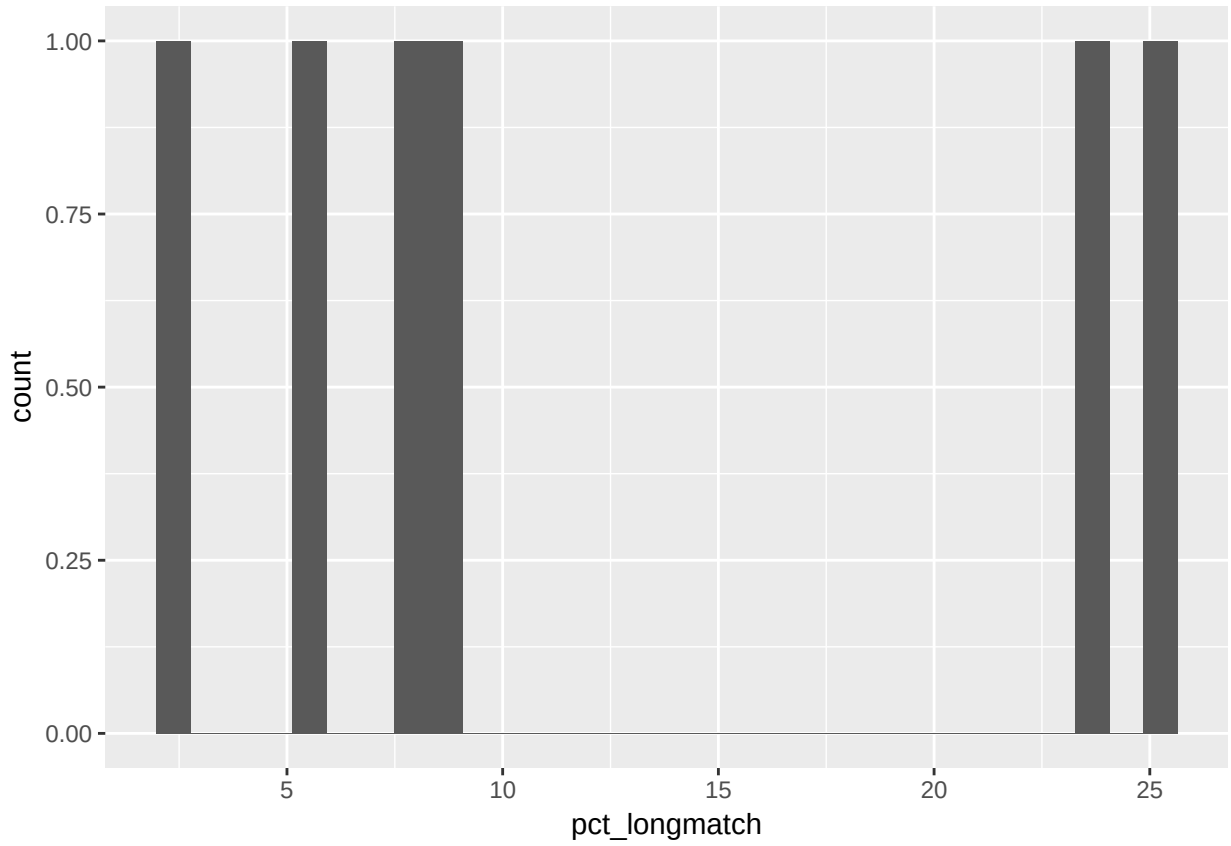


- % of peptides with identification probability =1 matched [plot as scatterplot vs total number of peptides]

```
pctlongpep <- grouped_peps %>%
  group_by(Acc_id,Rpep_index) %>%
  summarise(peplen = n(),matched = any(match)) %>%
  group_by(Acc_id) %>%
  summarise(total_peps = n(),
            long_peps = sum(peplen>6),
            matched_long_peps = sum(peplen>6 & matched),
            pct_longmatch = matched_long_peps / long_peps * 100,
            pct_longpep = long_peps / total_peps * 100)

ggplot(pctlongpep,aes(x= pct_longmatch)) + geom_histogram()
```

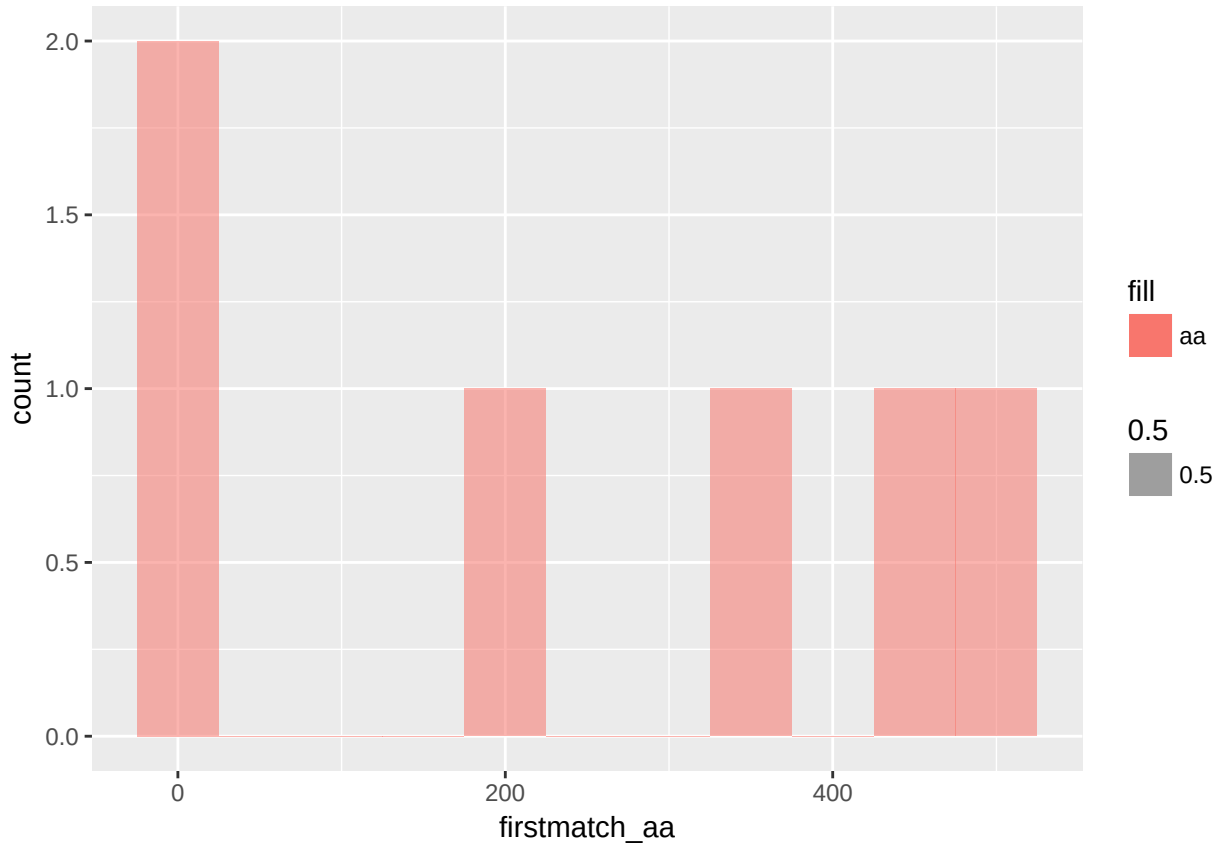
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



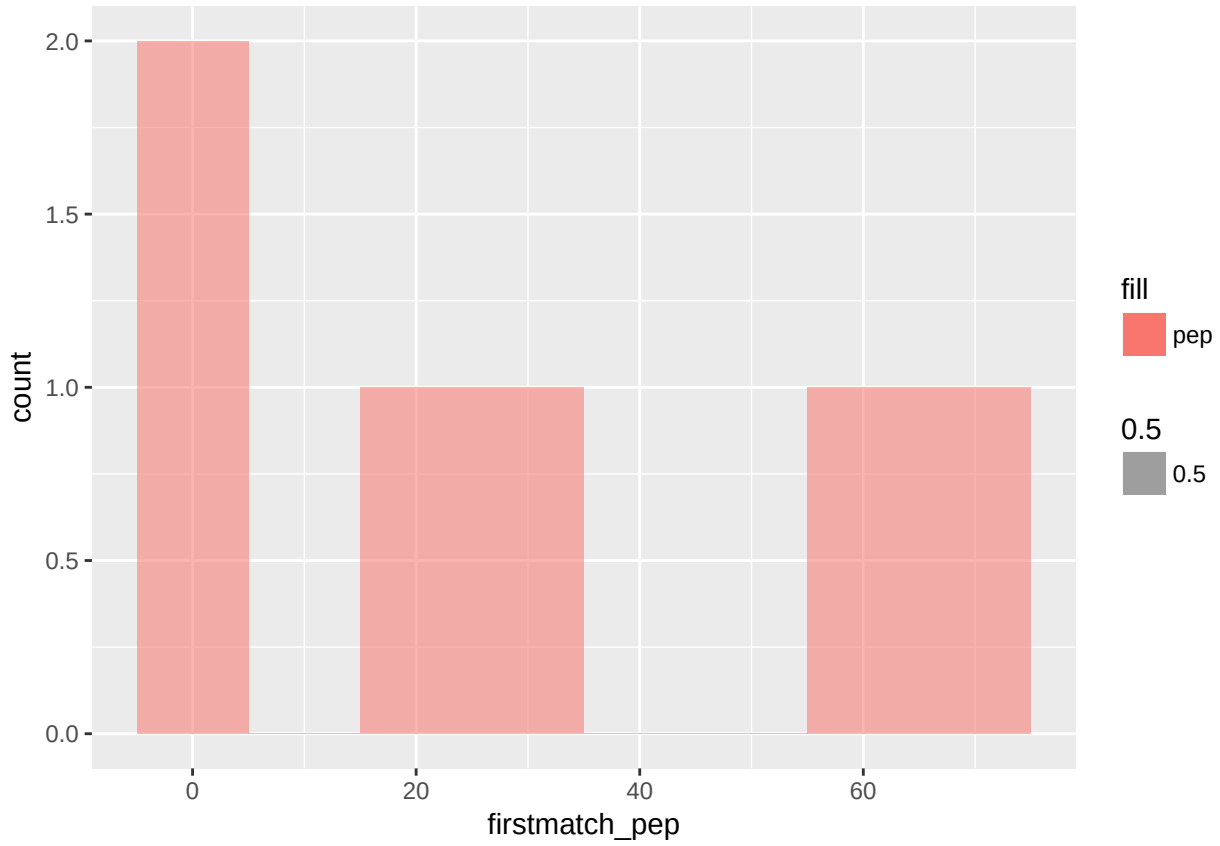
- start position of first match (both as Nth amino acid and Nth peptide) (old GAPS I believe) [plot as histogram]

```
first_match_start <- grouped_peps %>%
  group_by(Acc_id) %>%
  summarise(firstmatch_aa = min(firstmatch),
            firstmatch_pep = min(Rpep_index[group == 1]),
            percentile_aa = firstmatch_aa / n(),
            percentile_pep = (firstmatch_pep - 1) / max(Rpep_index))

p <- ggplot(first_match_start)
p+  geom_histogram(aes(x=firstmatch_aa, alpha = 0.5, fill = "aa"),binwidth = 50)
```

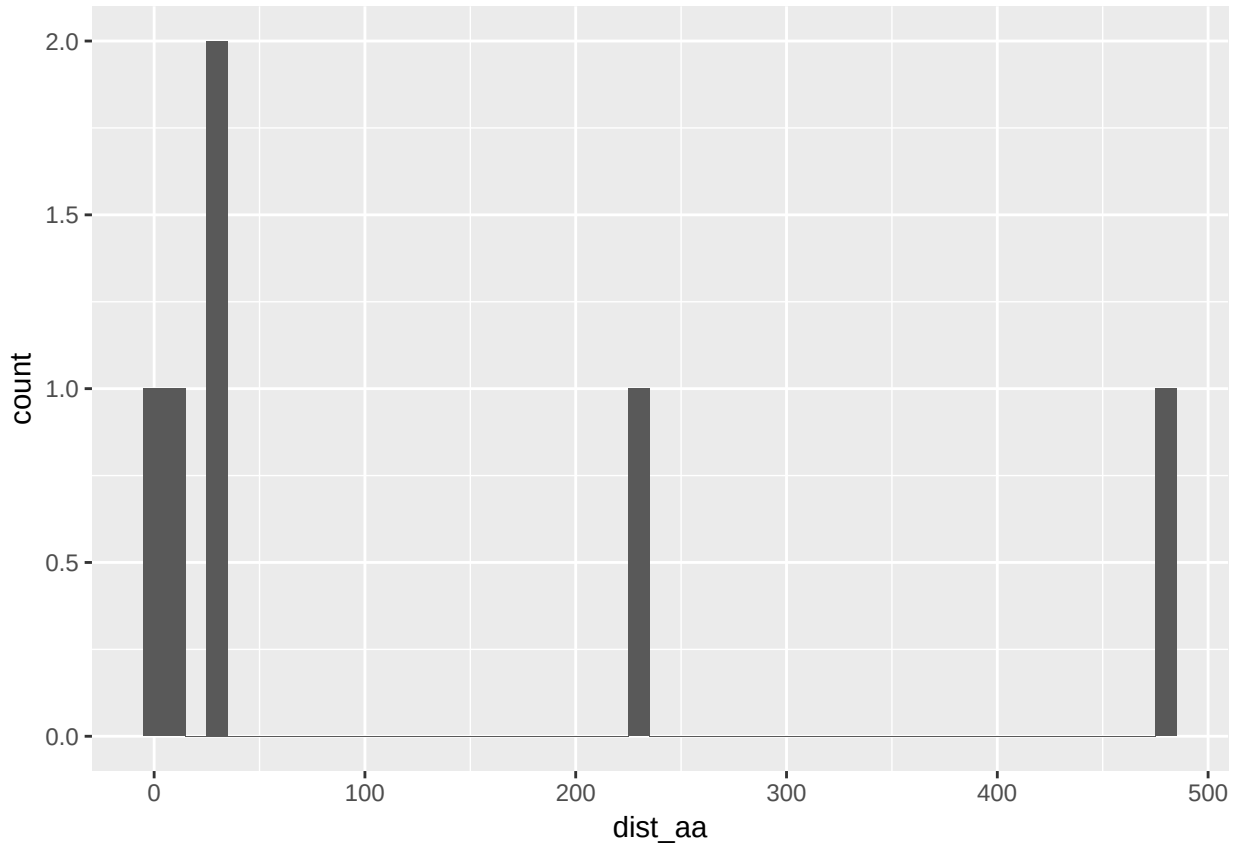



```
p+  geom_histogram(aes(x=firstmatch_pep, alpha = 0.5, fill = "pep"),binwidth = 10)
```

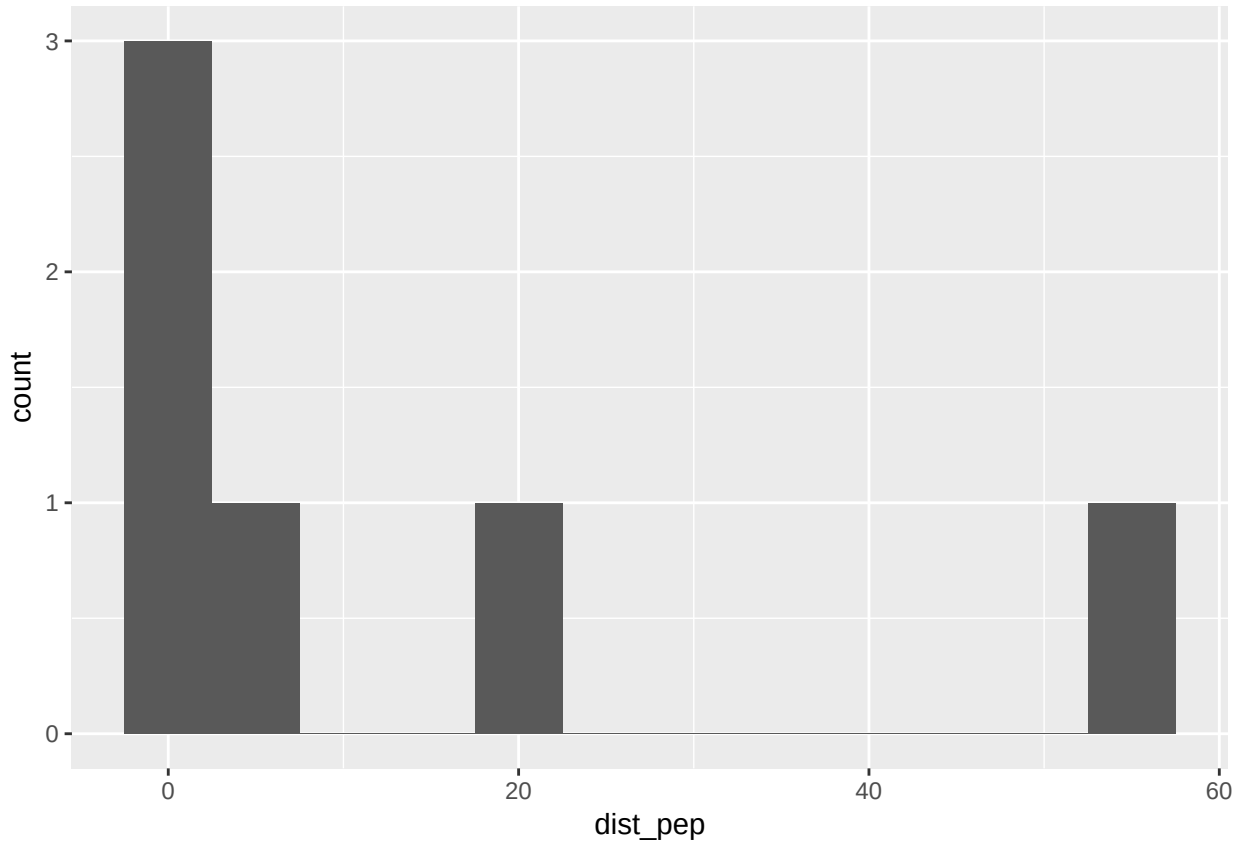


- distance of endposition of last match (both as Nth amino acid and Nth peptide) for resed end (old GAPE I believe) [plot as histogram]

```
combo %>%
  group_by(Acc_id) %>%
  summarise(lastmatch_aa = max(Lend),
            dist_aa = Rlen[1] - lastmatch_aa) %>%
  ggplot(aes(x=dist_aa)) + geom_histogram(binwidth = 10)
```



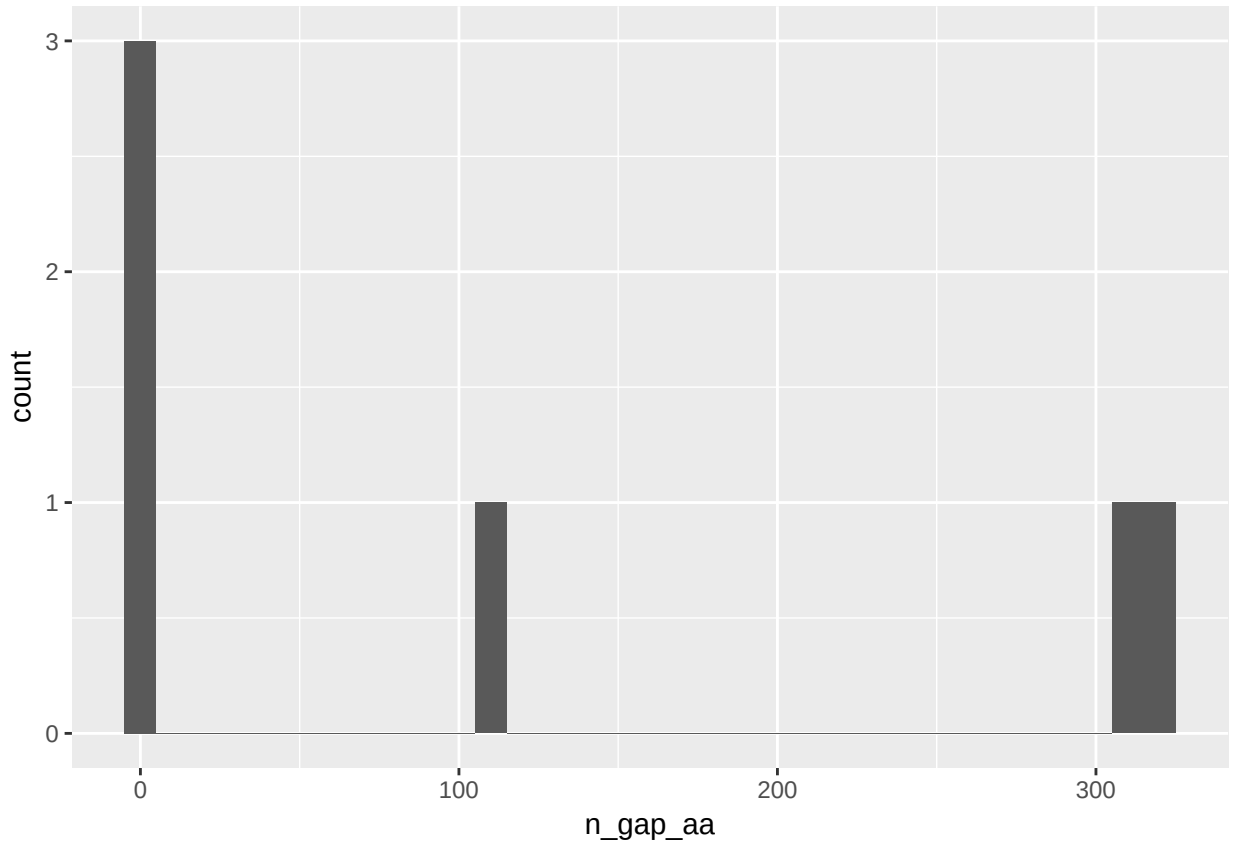

```
grouped_peps %>%
  group_by(Acc_id) %>%
  summarise(lastmatch_pep = max(Rpep_index[match]),
            dist_pep = max(Rpep_index) - lastmatch_pep) %>%
  ggplot(aes(x=dist_pep)) + geom_histogram(binwidth = 5)
```



- # and percentage of inner gaps (both in terms of amino acids and #peptides) [plot as histogram]

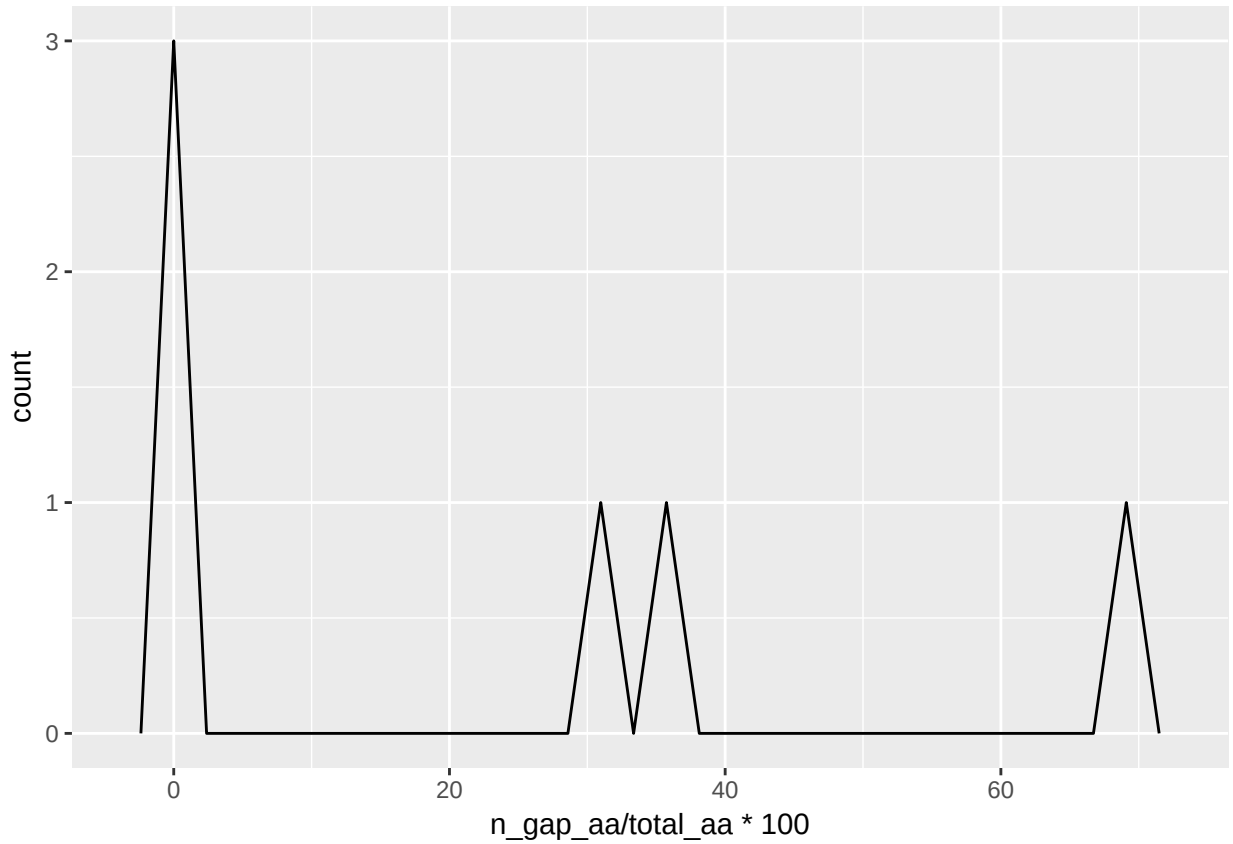
```
pgap <- grouped_peps %>%
  group_by(Acc_id) %>%
  summarise(n_gap_aa = max(index[match])-min(index[match]) - sum(match) + 1,
            total_aa = n(),
            n_gap_pep = sum(diff(match) > 0 ) - 1,
            total_pep = nlevels(as.factor(Rpep_index))) %>%
  ggplot()


pgap + geom_histogram(aes(x = n_gap_aa),binwidth = 10)
```

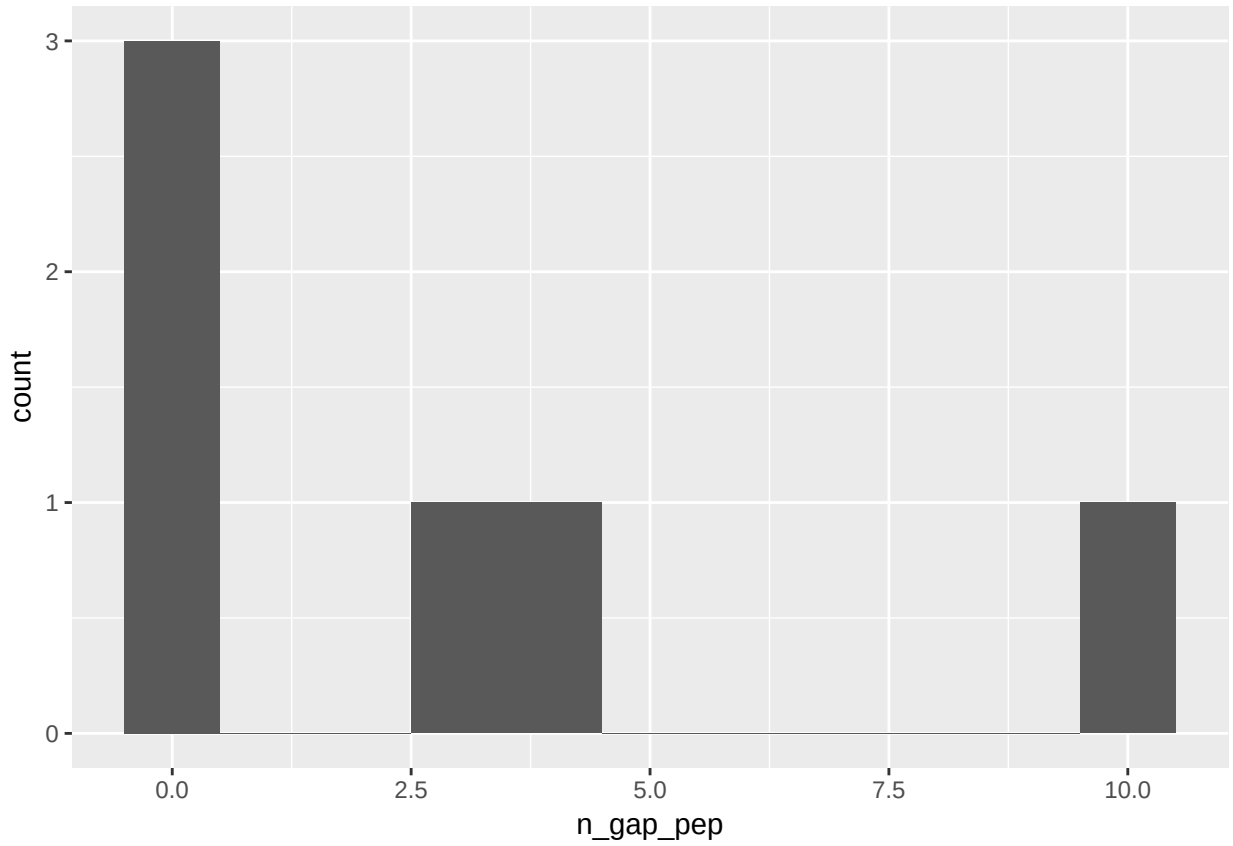



```
pgap + geom_freqpoly(aes(x = n_gap_aa / total_aa * 100))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

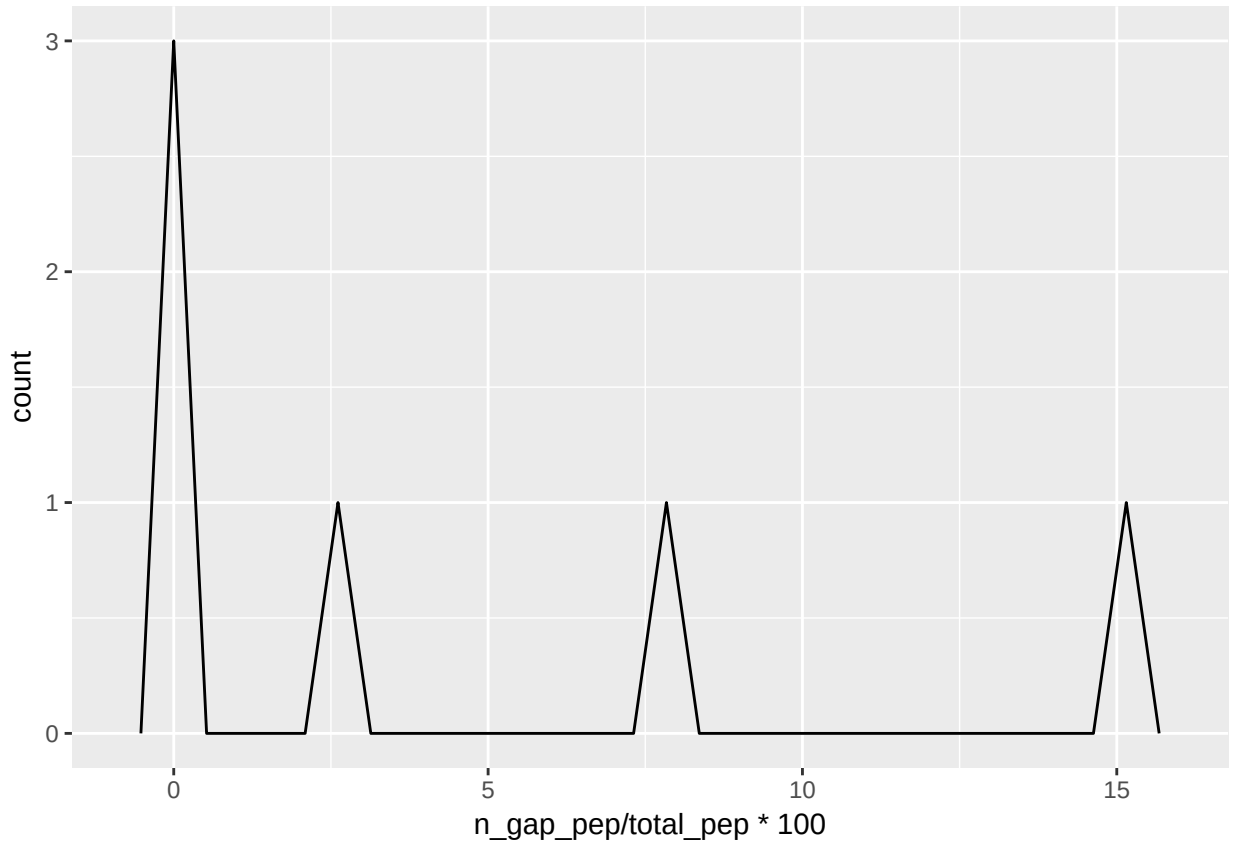



```
pgap + geom_histogram(aes(x = n_gap_pep),binwidth = 1)
```




```
pgap + geom_freqpoly(aes(x = n_gap_pep / total_pep * 100))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

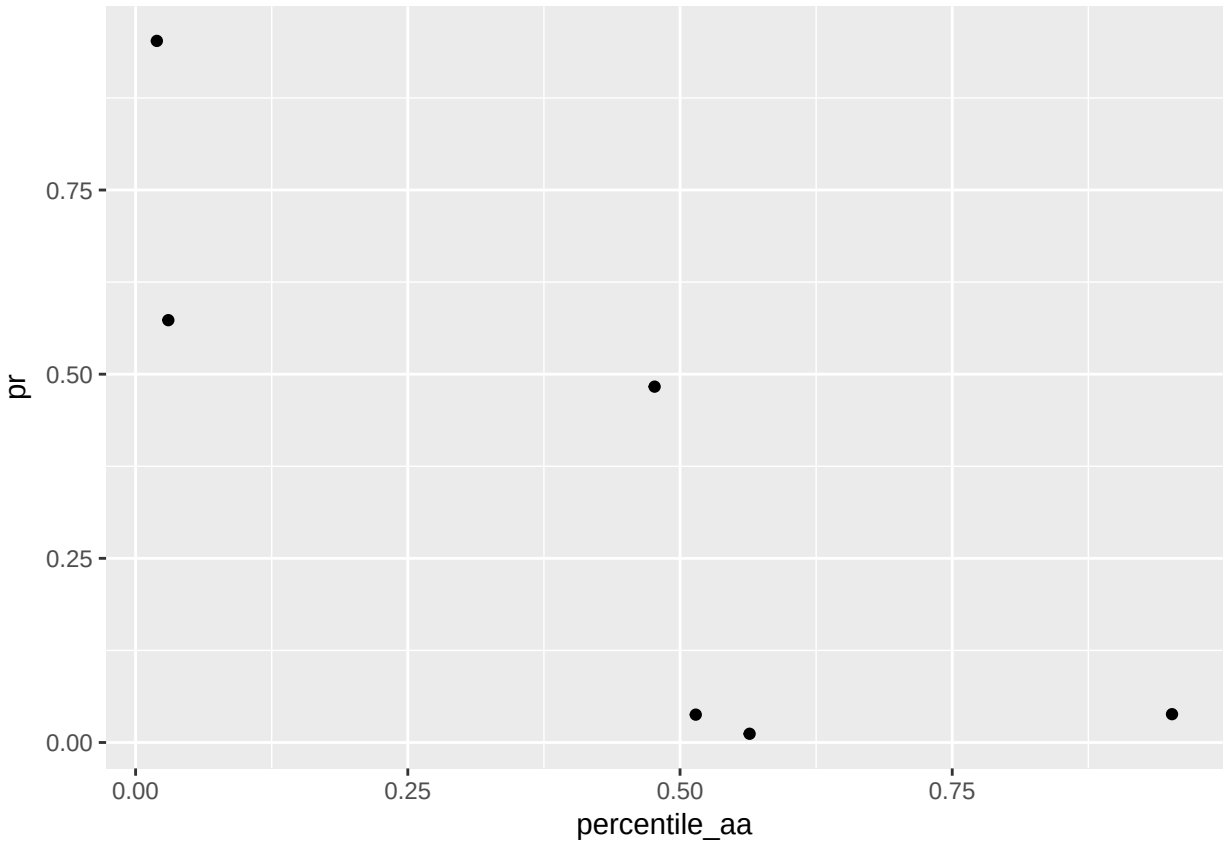


- Hypergeometric test of peptide distribution between the two groups (start to before first match, first match to end) as discussed [plot as scatter vs some of the above values, e.g. %peptides matched, start position of first match, . . .]

plot the probability that the observed first match position is due to random chance, against the first match position as a percentile in the protein sequence.

```
grouped_peps %>%
  group_by(Acc_id) %>%
  summarise(q = nlevels(as.factor(Rpep_index[match])), # number of peptides matched
            m = nlevels(as.factor(Rpep_index[group==1])), # number of peptides in group 1
            n = nlevels(as.factor(Rpep_index[group==0])), # number of peptides in group 0
            k = nlevels(as.factor(Rpep_index[match])), # numbero of peptides matched
            pr = phyper(q,m,n,k) - phyper(q-1,m,n,k)) %>%
  full_join(first_match_start) %>%
  ggplot(aes(x = percentile_aa, y = pr))+
  geom_point()
```

```
## Joining, by = "Acc_id"
```


0.75
0.50
0.25
0.00
pr
0.00
0.25
0.50
0.75
percentile_aa