R implementation

Sleiman Bassim

July 2, 2015

Loaded functions:

```
#source("/media/Data/Dropbox/humanR/01funcs.R")
rm(list=ls())
#setwd("/media/Data/Dropbox/humanR/PD/")
#setwd("~/Dropbox/humanR/PD/")
###load("PD.Rdata", .GlobalEnv)
#lsos(pat="")
```

1 Data preprocessing: load R packages

Load packages.

```
pkgs <- c('xlsx','lattice','latticeExtra',
          'ggplot2', 'dplyr', 'vegan', 'tidyr',
          'ggbiplot')
lapply(pkgs, require, character.only = TRUE)
```

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

[[2]]
[1] TRUE

[[3]]
[1] TRUE

[[4]]
[1] TRUE

[[5]]
[1] TRUE

[[6]]
[1] TRUE

[[7]]
[1] TRUE

[[8]]
[1] TRUE
```

1.1 Load gff3 sequence length data for all mapped libraries and references

GFF3 files contains the sequence length of each contig. These contigs belong to Steve Roberts genome v017 and transcriptome v22 of QPX. GFF3 were generated with an in-house perl script.

```
genome <- read.table("./QPX_Genome_v017.gff3")
```

```
head(genome)
```

```
                    V1 V2  V3 V4    V5 V6 V7 V8
1 QPX_v017_contig_1007  . CDS  1 15433  .  .  .
2 QPX_v017_contig_1043  . CDS  1 11565  .  .  .
3 QPX_v017_contig_1050  . CDS  1 12908  .  .  .
4 QPX_v017_contig_1087  . CDS  1 12852  .  .  .
5 QPX_v017_contig_1094  . CDS  1 10365  .  .  .
6 QPX_v017_contig_1128  . CDS  1 10580  .  .  .
                                                 V9
1 ID=QPX_v017_contig_1007;Name=QPX_v017_contig_1007
2 ID=QPX_v017_contig_1043;Name=QPX_v017_contig_1043
3 ID=QPX_v017_contig_1050;Name=QPX_v017_contig_1050
4 ID=QPX_v017_contig_1087;Name=QPX_v017_contig_1087
5 ID=QPX_v017_contig_1094;Name=QPX_v017_contig_1094
6 ID=QPX_v017_contig_1128;Name=QPX_v017_contig_1128
```

```
transcriptome <- read.table("./QPX_transcriptome_v2orf.gff3")
head(transcriptome)
```

```
                                  V1 V2  V3 V4   V5 V6 V7 V8
1  QPX_transcriptome_v2_Contig_1335_1  . CDS  1  210  .  .  .
2  QPX_transcriptome_v2_Contig_1456_7  . CDS  1  285  .  .  .
3  QPX_transcriptome_v2_Contig_1465_1  . CDS  1 1107  .  .  .
4  QPX_transcriptome_v2_Contig_1887_1  . CDS  1  243  .  .  .
5 QPX_transcriptome_v2_Contig_1941_15  . CDS  1  621  .  .  .
6  QPX_transcriptome_v2_Contig_1952_2  . CDS  1  330  .  .  .
                                                                                                  V9
1   ID=QPX_transcriptome_v2_Contig_1335_1;Name=QPX_transcriptome_v2_Contig_1335_1
2   ID=QPX_transcriptome_v2_Contig_1456_7;Name=QPX_transcriptome_v2_Contig_1456_7
3   ID=QPX_transcriptome_v2_Contig_1465_1;Name=QPX_transcriptome_v2_Contig_1465_1
4   ID=QPX_transcriptome_v2_Contig_1887_1;Name=QPX_transcriptome_v2_Contig_1887_1
5 ID=QPX_transcriptome_v2_Contig_1941_15;Name=QPX_transcriptome_v2_Contig_1941_15
6   ID=QPX_transcriptome_v2_Contig_1952_2;Name=QPX_transcriptome_v2_Contig_1952_2
```

GFF3 counts of MME transcriptomes **MMETSP0098** and **MMETSP00992**, and the custom assembly that *I* did for MMETSP0098.

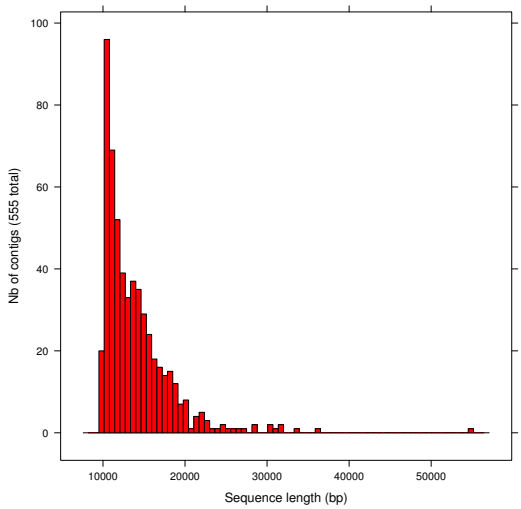
```
mme98 <- read.table("./MMETSP0098.gff3")
mme99 <- read.table("./MMETSP0099_2.gff3")
mme98c <- read.table("./mme98cust.gff3")
genomv015 <- read.table("./QPX_v015.gff3")
```

1.2 Distribution of contig length for libraries per reference

The number of bases has been counted and published elsewhere by the authors who assembled the references and sequenced the libraries. Working through their data, we provide a distribution of contig length for genome of Steve's QPX. The purpose of this analysis is to identify 2 things:

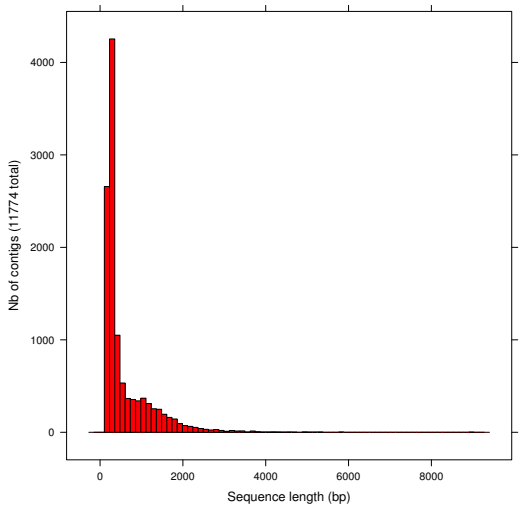
- Biases in contig length
- Comparison of parameters used for assembling the references

```
histogram(~ (genome$V5),
          type= 'count',
          nint = 75,
          data = genome,
          xlab = 'Sequence length (bp)',
          ylab = 'Nb of contigs (555 total)',
          col = 'red')
```



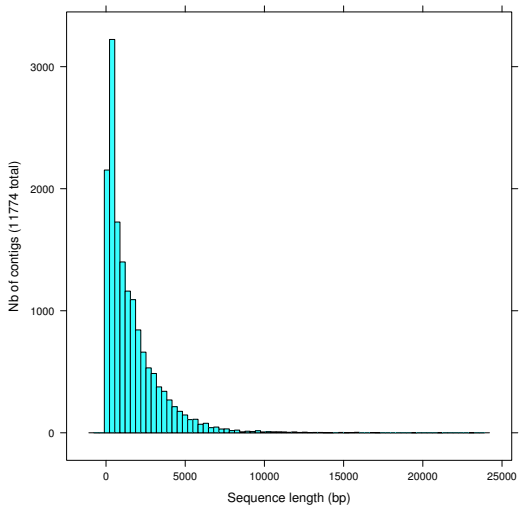
Distribution of Steve's QPX transcriptome.

```
histogram(~ transcriptome$V5,
          type = 'count',
          col = 'red',
          data = transcriptome,
          nint = 75,
          xlab = 'Sequence length (bp)',
          ylab = 'Nb of contigs (11774 total)')
```



Distribution of length of MMETSP0098.

```
histogram(~ mme98$V5,
          type = 'count',
          nint = 75,
          data = mme98,
          xlab = 'Sequence length (bp)',
          ylab = 'Nb of contigs (11774 total)')
```



Superpose length of contigs in:

- Steve's genome v017 (555 contigs)
- Steve's transcriptome
- MMEtsp0098 transcriptome
- MMEtsp00992 transcriptome
- MMEtsp0098 custom transcriptome
- Steve's Genome v015 (approx 22,000 contigs)

Merge datasets. Then add new column that designs the nature of each contig.

```
lsos()
```

```
                    Type     Size PrettySize  Rows Columns
genome         data.frame   128048     125 Kb   555       9
genomv015      data.frame  4840272     4.6 Mb 21280       9
mme98          data.frame  3535944     3.4 Mb 15489       9
mme98c         data.frame  8259976     7.9 Mb 39946       9
mme99          data.frame  2687328     2.6 Mb 11767       9
pkgs            character      504  504 bytes     8      NA
transcriptome  data.frame  3630424     3.5 Mb 11774       9
```

```
grouping <- rbind(genome[, c(1,5)],
                  transcriptome[, c(1, 5)],
                  mme98[, c(1,5)],
                  mme99[, c(1,5)],
                  mme98c[, c(1,5)],
                  genomv015[, c(1,5)])
grouping <- data.frame(grouping,
                       y = c(rep("GenomeV17", nrow(genome)),
                           rep("TrxV22", nrow(transcriptome)),
                           rep("MME98", nrow(mme98)),
                           rep("MME99", nrow(mme99)),
                           rep("MME98custom", nrow(mme98c)),
                             rep("(GenomeV15)", nrow(genomv015))))

dim(grouping)
```

```
[1] 100811      3
```

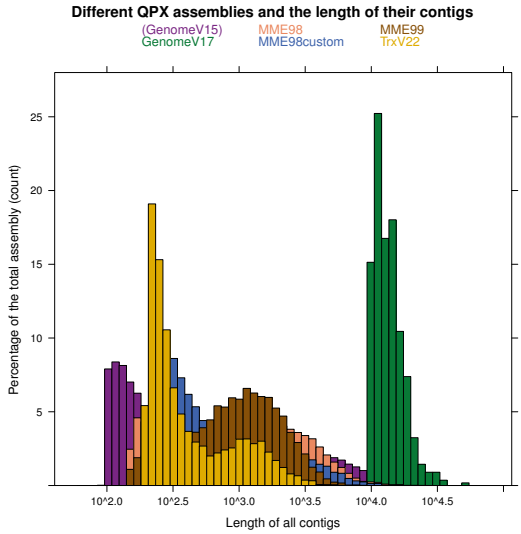
```
colnames(grouping)
```

```
[1] "V1" "V5" "y"
```

Plot length of the5 assembly including one genome.

```
custom.colors <- c(col1 = "#762a83",
                   col2 = "#1b7837",
                   col3 = "#ef8a62",
                   col4 = "#2166ac",
                   col5 = "#8c510a",
                   col6 = "#e6ab02")

histogram( V1 ~ V5,
     data = grouping,
     nint = 55,
     scales = list(log = 10),
     type = "p",
     #breaks = seq(4,8,by=0.2),
     ylim = c(0,28),
     groups = grouping$y,
     panel = function(...) panel.superpose(...,
         panel.groups = panel.histogram,
         col = custom.colors,
         alpha = 1),
     auto.key=list(columns=3,
         rectangles = FALSE,
         col = custom.colors),
     main = 'Different QPX assemblies and the length of their contigs',
     xlab = 'Length of all contigs',
     ylab = 'Percentage of the total assembly (count)'
     )
```

```
Warning in histogram.formula(V1 ~ V5, data = grouping, nint = 55, scales = list(log
= 10), :  Can't have log Y-scale
```



2 Quality controls after trimming bad regions in contigs

Many different options are available while trimming the contigs which have been already assembled.

- Nature of PCR adapters (trueSeq2 or trueSeq3)
- Sliding window while reading contigs
- Crop less than a desired contig length
- Minimum length for contigs
- Trailing is to remove ends of contigs with bad quality

```
lsos()
```

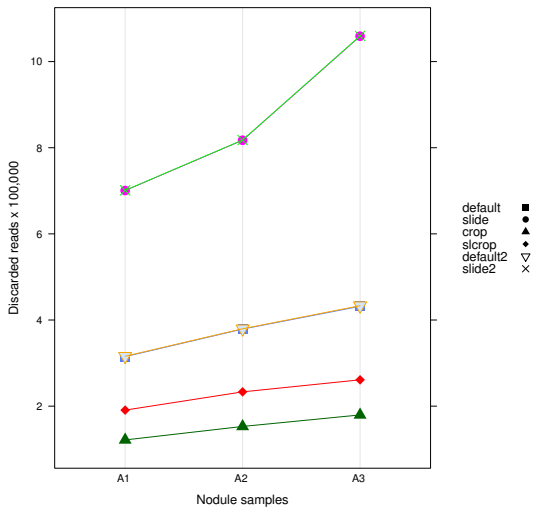
```
                    Type    Size PrettySize   Rows Columns
custom.colors  character     864  864 bytes      6      NA
genome        data.frame  128048     125 Kb    555       9
genomv015     data.frame 4840272     4.6 Mb  21280       9
grouping      data.frame 9251608     8.8 Mb 100811       3
mme98         data.frame 3535944     3.4 Mb  15489       9
mme98c        data.frame 8259976     7.9 Mb  39946       9
mme99         data.frame 2687328     2.6 Mb  11767       9
pkgs           character     504  504 bytes      8      NA
transcriptome data.frame 3630424     3.5 Mb  11774       9
```

```
trim <- read.xlsx("./Classeur1.xlsx", header = T, sheetName = "Feuil1")
trim <- trim[1:3, ]
trim
```

```
  Sample.    Total.  default    slide     crop   slcrop default2
1      A1  30491569 30176618 29790935 30370029 30300807 30175710
2      A2  34515597 34136650 33698155 34362754 34282437 34135918
3      A3  46861893 46430064 45802956 46682292 46600781 46428610
    slide2
1 29790528
2 33697784
3 45802292
```

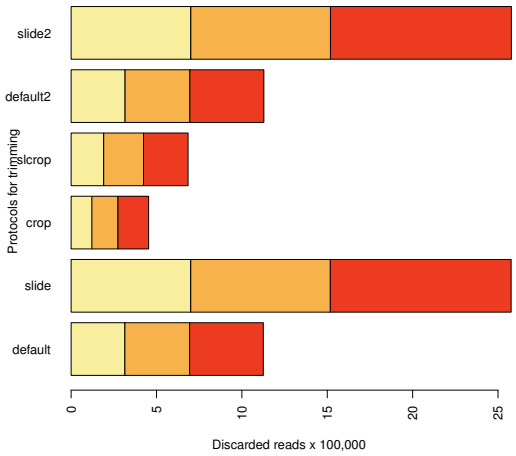
Plot the differences between nature off adapters (colors) and the combination of the other parameters (shapes).

```
key.variety <- list(space = "right",
                    text = list(colnames(trim[, -c(1:2)])),
                    points = list(pch = c(15:18,25,4)))

dotplot(c(trim$Total-trim$default)/100000 +
          c(trim$Total-trim$slide)/100000 +
              c(trim$Total-trim$crop)/100000 +
                  c(trim$Total-trim$slcrop)/100000 +
                      c(trim$Total-trim$default2)/100000 +
                          c(trim$Total-trim$slide2)/100000
        ~ trim$Sample,
        data = trim,
        type = 'o',
         pch = c(15:18,25,4),
         key = key.variety,
         lty = 1, cex = 1.5,
        xlab = 'Nodule samples',
        ylab = 'Discarded reads x 100,000')
```



Another way to visualize the discarded reads.

```
custom.colors <- c(col1 = "#ffeda0", col2 = "#feb24c", col3 = "#f03b20")

barplot(as.matrix((trim$Total-trim[, -c(1:2)])/100000),
        horiz = TRUE,
        col = custom.colors,
        xlab = 'Discarded reads x 100,000',
        ylab = 'Protocols for trimming',
        las = 2)
```



3 Aligning contigs to reference

Two sets of reads were mapped to 2 references. First batch from the trimmed reads with the default parameters (adapters clipping, trailing, and minimum length) and TrueSeq3 adapters (ones Bassem supplied). Second batch were also trimmed with default settings but using TrueSeq2 adapters (ones that sleiman supplied). With the second batch of adapters, more reads were trimmed and discarded. This analysis will try to show why by regressing the number of mapped reads over the length of each reference.

```
ref.genome1 <- read.table("./refGenome/A1.htseq.counts.txt")
ref.genome2 <- read.table("./refGenome/A2.htseq.counts.txt")
ref.genome3 <- read.table("./refGenome/A3.htseq.counts.txt")
ref.genome4 <- read.table("./refGenome/A1-4.htseq.counts.txt")
ref.genome5 <- read.table("./refGenome/A2-4.htseq.counts.txt")
ref.genome6 <- read.table("./refGenome/A3-4.htseq.counts.txt")
```

Merge all mapped reads on the genome

```
ref.genome <- data.frame(
    A1 = ref.genome1[-c(556:560), 2],
    A2 = ref.genome2[-c(556:560), 2],
    A3 = ref.genome3[-c(556:560), 2],
    A1.4 = ref.genome4[-c(556:560), 2],
    A2.4 = ref.genome5[-c(556:560), 2],
    A3.4 = ref.genome6[-c(556:560), 2],
    contigs = ref.genome1[-c(556:560), 1])
dim(ref.genome)

[1] 555   7

# just because im lazy
genome1 <- data.frame(contigs= genome[,1], length = genome$V5)
```

Merge length and number of mapped reads

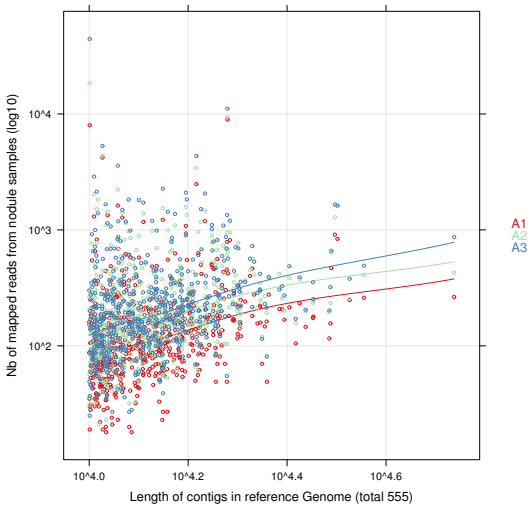
```
ref.genome.mix <- merge(genome1, ref.genome)
```

```
head(ref.genome.mix)
```

```
                contigs length  A1  A2  A3 A1.4 A2.4 A3.4
1 QPX_v017_contig_1007   15433 117 197 249  117  197  249
2 QPX_v017_contig_1020   12397 123 164 171  123  164  171
3 QPX_v017_contig_1021   18562 335 487 596  335  488  596
4 QPX_v017_contig_1023   19919 116 198 331  116  198  331
5  QPX_v017_contig_103   10989  71 111 107   71  111  105
6 QPX_v017_contig_1034   10178 196 289 655  196  289  655
```

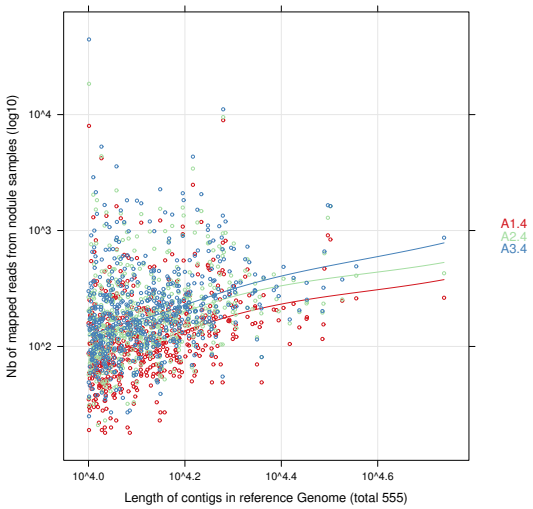
Plot the correlation between read length and number of mapped reads on the genome of QPX with the remaining reads from the default trimming with TrueSeq3 adapters.

```
custom.colors <- c('#d7191c', '#abdda4', '#2b83ba')
key.variety <- list(space = "right",
                    text = list(colnames(ref.genome.mix[, 3:5])),
                    col = custom.colors)
xyplot(A1 + A2 + A3 ~ length,
       data = ref.genome.mix,
       xlab = 'Length of contigs in reference Genome (total 555)',
       ylab = 'Nb of mapped reads from nodule samples (log10)',
       col = custom.colors,
       cex = 0.5,
       type = c("g", "p", "smooth"),
       scales = list(log = 10),
       key = key.variety)
```



Regression between reads and length of contigs in reference genome with adapters TrueSeq2 under default trimming settings.

```
custom.colors <- c('#d7191c', '#abdda4', '#2b83ba')
key.variety <- list(space = "right",
                    text = list(colnames(ref.genome.mix[, 6:8])),
                    col = custom.colors)
xyplot(A1.4 + A2.4 + A3.4 ~ length,
#                    alpha = .5,
       data = ref.genome.mix,
       xlab = 'Length of contigs in reference Genome (total 555)',
       ylab = 'Nb of mapped reads from nodule samples (log10)',
       col = custom.colors,
       cex = 0.5,
       type = c("g", "p", "smooth"),
       scales = list(log = 10),
       key = key.variety)
```



Previously i regressed the number of mapped reads of nodule samples over the reference genome of QPX. Now its time to do the same thing over the reference transcriptome of QPX. Both references belong the Steve Roberts.

```
ref.transcriptome1 <- read.table("./refTranscriptome/A1.htseq.counts.txt")
ref.transcriptome2 <- read.table("./refTranscriptome/A2.htseq.counts.txt")
ref.transcriptome3 <- read.table("./refTranscriptome/A3.htseq.counts.txt")
ref.transcriptome4 <- read.table("./refTranscriptome/A1-4.htseq.counts.txt")
ref.transcriptome5 <- read.table("./refTranscriptome/A2-4.htseq.counts.txt")
ref.transcriptome6 <- read.table("./refTranscriptome/A3-4.htseq.counts.txt")
dim(ref.transcriptome1)

[1] 11779     2

tail(ref.transcriptome1)

                                    V1       V2
11774 QPX_transcriptome_v2_Contig_9_3        6
11775                    __no_feature        0
11776                     __ambiguous      568
11777                  __too_low_aQual   147407
11778                    __not_aligned 29746511
11779           __alignment_not_unique        0
```

Merge all mapped reads on the transcriptome

```
ref.transcriptome <- data.frame(
```

```
    A1 = ref.transcriptome1[-c(11775:11779), 2],
    A2 = ref.transcriptome2[-c(11775:11779), 2],
    A3 = ref.transcriptome3[-c(11775:11779), 2],
    A1.4 = ref.transcriptome4[-c(11775:11779), 2],
    A2.4 = ref.transcriptome5[-c(11775:11779), 2],
    A3.4 = ref.transcriptome6[-c(11775:11779), 2],
    contigs = ref.transcriptome1[-c(11775:11779), 1])
dim(ref.transcriptome)
```

```
[1] 11774     7
```

```
head(ref.transcriptome)
```

```
  A1 A2 A3 A1.4 A2.4 A3.4                              contigs
1  0  2  0    0    2    0 QPX_transcriptome_v2_Contig_10002_1
2  0  0  0    0    0    0 QPX_transcriptome_v2_Contig_10002_2
3  2  2  7    2    2    7  QPX_transcriptome_v2_Contig_1000_1
4  1  0  0    1    0    0  QPX_transcriptome_v2_Contig_1000_2
5  0  0  0    0    0    0  QPX_transcriptome_v2_Contig_1000_3
6 58 72 99   59   72  100  QPX_transcriptome_v2_Contig_1000_4
```

```
# just because im lazy
transcriptome1 <- data.frame(contigs= transcriptome[,1], length = transcriptome$V5)
```

Merge length and number of mapped reads

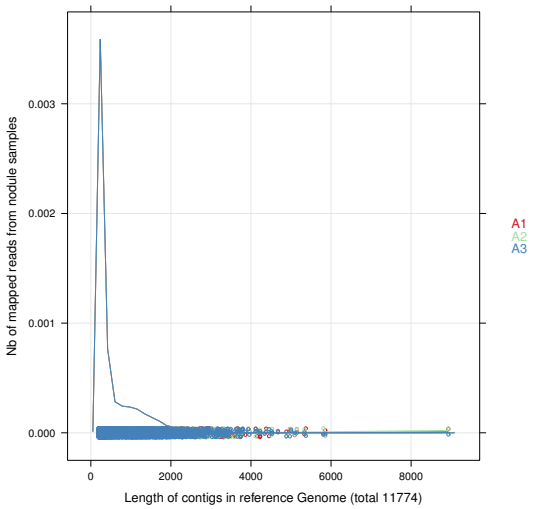
```
ref.transcriptome.mix <- merge(transcriptome1, ref.transcriptome)
head(ref.transcriptome.mix)
```

```
                              contigs length A1 A2 A3 A1.4 A2.4 A3.4
1  QPX_transcriptome_v2_Contig_1000_1    201  2  2  7    2    2    7
2  QPX_transcriptome_v2_Contig_1000_2    258  1  0  0    1    0    0
3 QPX_transcriptome_v2_Contig_10002_1    477  0  2  0    0    2    0
4 QPX_transcriptome_v2_Contig_10002_2    264  0  0  0    0    0    0
5  QPX_transcriptome_v2_Contig_1000_3    321  0  0  0    0    0    0
6  QPX_transcriptome_v2_Contig_1000_4   1473 58 72 99   59   72  100
```

3.1 Concentration of contigs in different libraries

Plot correlation between transcriptome contigs and assembled reads of nodule samples. Trimming parameters are of default with TrueSeq3 adapters.

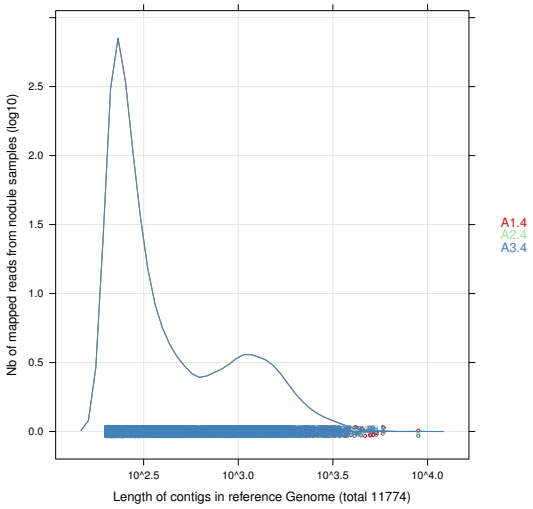
```
custom.colors <- c('#d7191c', '#abdda4', '#2b83ba')
key.variety <- list(space = "right",
                    text = list(colnames(ref.genome.mix[, 3:5])),
                    col = custom.colors)
densityplot(A1 + A2 + A3 ~ length,
       data = ref.transcriptome.mix,
#            alpha = .7,
            xlab = 'Length of contigs in reference Genome (total 11774)',
       ylab = 'Nb of mapped reads from nodule samples',
       col = custom.colors,
       cex = 0.5,
       type = c("g", "p", "smooth"),
#       scales = list(log = 10),
       key = key.variety)
```



Plot correlation same as above but with TrueSeq2 adapters with default parameters.

```
custom.colors <- c('#d7191c', '#abdda4', '#2b83ba')
key.variety <- list(space = "right",
                    text = list(colnames(ref.genome.mix[, 6:8])),
                    col = custom.colors)
densityplot(A1.4 + A2.4 + A3.4 ~ length,
       data = ref.transcriptome.mix,
#            alpha = .9,
       xlab = 'Length of contigs in reference Genome (total 11774)',
       ylab = 'Nb of mapped reads from nodule samples (log10)',
       col = custom.colors,
       cex = 0.5,
       type = c("g", "p", "smooth"),
       scales = list(log = 10),
       key = key.variety)
```

```
Warning in densityplot.formula(A1.4 + A2.4 + A3.4 ~ length, data = ref.transcriptome.mix,
:  Can't have log Y-scale
```



Load the number of mapped reads to the MMETSP0098 transcriptome before discarding duplicates.

```
ref.dupA1R3 <- read.table("./refMME98/A1.htseq.counts.txt")
ref.dupA2R3 <- read.table("./refMME98/A2.htseq.counts.txt")
ref.dupA3R3 <- read.table("./refMME98/A3.htseq.counts.txt")
```

Merge all mapped reads to MMETSP0098 reference transcriptome before discarding duplicates (ie, raw counts).

```
ref.mme98 <- data.frame(A1 = ref.dupA1R3$V2,
                        A2 = ref.dupA2R3$V2,
                        A3 = ref.dupA3R3$V2,
                        contigs = ref.dupA1R3$V1)
```

Add the length values to each contig mapped to MMETsp0098. But first remove extra rows.

```
nr <- nrow(ref.mme98)
ref.mme98 <- ref.mme98[1:(nr-5), ]
tail(ref.mme98)
```

```
      A1 A2 A3                   contigs
15484  0  0  0 MMETSP0098-20131031|9992
15485 12  6 17 MMETSP0098-20131031|9993
15486  0  0  0 MMETSP0098-20131031|9995
15487  3  3  3 MMETSP0098-20131031|9996
15488  0  0  0 MMETSP0098-20131031|9998
15489  2  9  5 MMETSP0098-20131031|9999
```

Merge length and counts.

```
#ref.mme98.mix <- merge(ref.mme98, mme98[, c(1,5)])
```

Load the number of mapped reads to the MMETSP0099_2 transcriptome before discarding duplicates.

```
ref.dupA1R4 <- read.table("./refMME992/A1.htseq.counts.txt")
ref.dupA2R4 <- read.table("./refMME992/A2.htseq.counts.txt")
ref.dupA3R4 <- read.table("./refMME992/A3.htseq.counts.txt")
```

Merge all mapped reads to MMETSP0099_2 reference transcriptome before discarding duplicates (ie, raw counts).

```
ref.mme992 <- data.frame(A1 = ref.dupA1R4$V2,
                         A2 = ref.dupA2R4$V2,
                         A3 = ref.dupA3R4$V2,
                         contigs = ref.dupA1R4$V1)
```

Add the length values to each contig mapped to MMETsp0099_2. But first remove extra rows.

```
nr <- nrow(ref.mme992)
ref.mme992 <- ref.mme992[1:(nr-5), ]
tail(ref.mme992)
```

```
      A1 A2 A3                     contigs
11762  0  0  0 MMETSP0099_2-20121227|9994
11763  0  8  9 MMETSP0099_2-20121227|9995
11764  0  0  0 MMETSP0099_2-20121227|9996
11765  0  0  1 MMETSP0099_2-20121227|9997
11766  0  0  0 MMETSP0099_2-20121227|9998
11767  0  0  0 MMETSP0099_2-20121227|9999
```

Load the number of mapped reads to SR v015 genome before discarding duplicates.

```
ref.dupA1R5 <- read.table("./refGenomV015/A1.htseq.counts.txt")
ref.dupA2R5 <- read.table("./refGenomV015/A2.htseq.counts.txt")
ref.dupA3R5 <- read.table("./refGenomV015/A3.htseq.counts.txt")
```

Merge all mapped reads to SR v015 reference genome before discarding duplicates (ie, raw counts).

```
ref.genomv015 <- data.frame(A1 = ref.dupA1R5$V2,
                         A2 = ref.dupA2R5$V2,
                         A3 = ref.dupA3R5$V2,
                         contigs = ref.dupA1R5$V1)
```

Add the length values to each contig mapped to SR v015 reference genome. But first remove extra rows.

```
nr <- nrow(ref.genomv015)
ref.genomv015 <- ref.genomv015[1:(nr-5), ]
tail(ref.genomv015)
```

```
      A1 A2 A3              contigs
21275  0  1  2 QPX_v015_contig_9994
21276  0  0  0 QPX_v015_contig_9995
21277  1  5  3 QPX_v015_contig_9996
21278  1  2  2 QPX_v015_contig_9997
21279  3  1  2 QPX_v015_contig_9998
21280  9  5 21 QPX_v015_contig_9999
```

4 Removing the duplicate reads and reducing bias for better coverage

After aligning the reads to a reference duplicates must be removed.
First load the sample reads mapped to reference genome (without duplication) of Steve Roberts.

```
nodupA1R1 <- read.table("./nodupR1/A1.htseq.nodup.counts.txt")
nodupA2R1 <- read.table("./nodupR1/A2.htseq.nodup.counts.txt")
nodupA3R1 <- read.table("./nodupR1/A3.htseq.nodup.counts.txt")
```

Second load the sample reads mapped to reference transcriptome (withtout duplication) of Steve Roberts.

```
nodupA1R2 <- read.table("./nodupR2/A1.htseq.nodup.counts.txt")
nodupA2R2 <- read.table("./nodupR2/A2.htseq.nodup.counts.txt")
nodupA3R2 <- read.table("./nodupR2/A3.htseq.nodup.counts.txt")
```

Third load the sample reads mapped to reference transcriptome (without duplication) of MMESTO0098.

```
nodupA1R3 <- read.table("./nodupR3/A1.htseq.counts.nodup.txt")
nodupA2R3 <- read.table("./nodupR3/A2.htseq.counts.nodup.txt")
nodupA3R3 <- read.table("./nodupR3/A3.htseq.counts.nodup.txt")
```

Forth load of sample reads mapped to reference transcriptome MMETSP0099_2.

```
nodupA1R4 <- read.table("./nodupR4/A1.htseq.counts.nodup.txt")
nodupA2R4 <- read.table("./nodupR4/A2.htseq.counts.nodup.txt")
nodupA3R4 <- read.table("./nodupR4/A3.htseq.counts.nodup.txt")
```

Forth load of sample reads mapped to reference SR genome v015 with approximately 21,000 contigs.

```
nodupA1R5 <- read.table("./nodupR5/A1.htseq.counts.nodup.txt")
nodupA2R5 <- read.table("./nodupR5/A2.htseq.counts.nodup.txt")
nodupA3R5 <- read.table("./nodupR5/A3.htseq.counts.nodup.txt")
```

Merge mapped samples relative to the reference.

- R1 = genome of QPX (steve roberts, 555 contigs)
- R2 = transcriptome of QPX (steve roberts)
- R3 = transcriptome of QPX MMETSP0098
- R4 = transcriptome of QPX MMETSP0099_2
- R5 = genome of QPX (steve roberts v015, approx. 21,000 contigs)

```
allR1 <- data.frame(A1n = nodupA1R1$V2,
```

```
                  A2n = nodupA2R1$V2,
                  A3n = nodupA3R1$V2,
                  reference = rep("genomSRv017", nrow(nodupA1R1)),
                    contigs = nodupA1R1$V1)
allR1 <- allR1[1:555, ]

allR2 <- data.frame(A1n = nodupA1R2$V2,
                  A2n = nodupA2R2$V2,
                  A3n = nodupA3R2$V2,
                  reference = rep("trxSRv022", nrow(nodupA1R2)),
                    contigs = nodupA1R2$V1)
allR2 <- allR2[1:11774, ]

allR3 <- data.frame(A1n = nodupA1R3$V2,
                  A2n = nodupA2R3$V2,
                  A3n = nodupA3R3$V2,
                  reference = rep("trxMME98", nrow(nodupA1R3)),
                    contigs = nodupA1R3$V1)
allR3 <- allR3[1:15489, ]

allR4 <- data.frame(A1n = nodupA1R4$V2,
                  A2n = nodupA2R4$V2,
                  A3n = nodupA3R4$V2,
                  reference = rep("trxMME992", nrow(nodupA1R4)),
                    contigs = nodupA1R4$V1)
allR4 <- allR4[1:c(nrow(nodupA1R4)-5), ]

allR5 <- data.frame(A1n = nodupA1R5$V2,
                  A2n = nodupA2R5$V2,
                  A3n = nodupA3R5$V2,
                  reference = rep("genomSRv015", nrow(nodupA1R5)),
                    contigs = nodupA1R5$V1)
allR5 <- allR5[1:c(nrow(nodupA1R5)-5), ]
```

95 Put before /after duplicates removal in one dataset for genome of Steve Roberts.

```
genomeSR <- merge(ref.genome.mix[, 1:5], allR1)
head(genomeSR)
```

```
               contigs length  A1  A2  A3 A1n A2n A3n   reference
1 QPX_v017_contig_1007  15433 117 197 249 109 191 240 genomSRv017
2 QPX_v017_contig_1020  12397 123 164 171 118 157 159 genomSRv017
3 QPX_v017_contig_1021  18562 335 487 596 319 460 568 genomSRv017
4 QPX_v017_contig_1023  19919 116 198 331 109 196 326 genomSRv017
5  QPX_v017_contig_103  10989  71 111 107  68 106 104 genomSRv017
6 QPX_v017_contig_1034  10178 196 289 655 185 279 592 genomSRv017
```

```
rownames(genomeSR) <- genomeSR$contigs
genomeSR <- t(genomeSR[, -c(1,9)])
genomeSR[, 1:3]
```

```
       QPX_v017_contig_1007 QPX_v017_contig_1020 QPX_v017_contig_1021
length                15433                12397                18562
A1                      117                  123                  335
A2                      197                  164                  487
A3                      249                  171                  596
A1n                     109                  118                  319
A2n                     191                  157                  460
A3n                     240                  159                  568
```

```
genomeSR <- data.frame(genomeSR,
                       y = c(2, rep(0,3), rep(1,3)))
```

96 Put the before /after duplicates removal in one datasetfor transcriptome of SR.

```
transcriptomeSR <- merge(ref.transcriptome.mix[, 1:5], allR2)
head(transcriptomeSR)

                             contigs length A1 A2 A3 A1n A2n A3n
1  QPX_transcriptome_v2_Contig_1000_1    201  2  2  7   2   2   7
2  QPX_transcriptome_v2_Contig_1000_2    258  1  0  0   1   0   0
3 QPX_transcriptome_v2_Contig_10002_1    477  0  2  0   0   2   0
4 QPX_transcriptome_v2_Contig_10002_2    264  0  0  0   0   0   0
5  QPX_transcriptome_v2_Contig_1000_3    321  0  0  0   0   0   0
6  QPX_transcriptome_v2_Contig_1000_4   1473 58 72 99  52  69  94
  reference
1 trxSRv022
2 trxSRv022
3 trxSRv022
4 trxSRv022
5 trxSRv022
6 trxSRv022
```

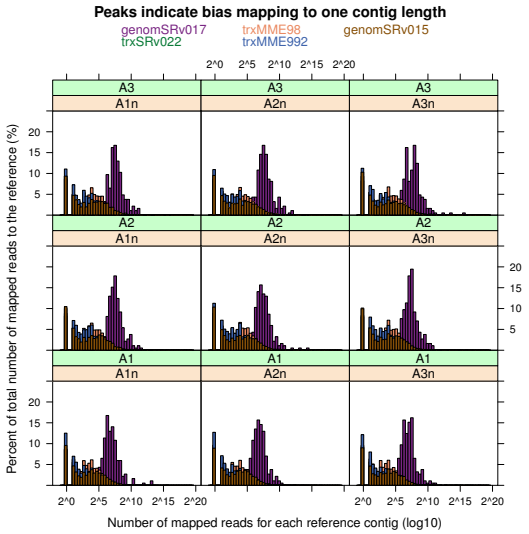
Present difference for each sample mapped to the references. First merge all samples before /after duplicates were removed.

```
allRefs <- rbind(allR1, allR2, allR3, allR4, allR5)
dim(allRefs)

[1] 60865     5

summary(allRefs$reference)

genomSRv017   trxSRv022    trxMME98   trxMME992 genomSRv015
        555       11774       15489       11767       21280

allRefs.raw <- rbind(ref.genome.mix[, 3:5],
                     ref.transcriptome.mix[, 3:5],
                     ref.mme98[, 1:3],
                     ref.mme992[, 1:3],
                     ref.genomv015[, 1:3])
dim(allRefs.raw)

[1] 60865     3

allDF <- cbind(allRefs, allRefs.raw)
allDF[sample(1:20000, 5), ]

      A1n A2n A3n reference                              contigs A1 A2
13661   0   0   0  trxMME98          MMETSP0098-20131031|11472  0  0
8346    4   8  12 trxSRv022 QPX_transcriptome_v2_Contig_5162_2  0  0
6367    0   4   7 trxSRv022 QPX_transcriptome_v2_Contig_3790_1 14 17
14112   0   1   0  trxMME98          MMETSP0098-20131031|12093  0  1
16724   0   0   3  trxMME98          MMETSP0098-20131031|15135  0  0
      A3
13661  0
8346   0
6367  20
14112  0
16724  3
```

Plot the difference before and after duplicates were discarded. The number of mapped reads to the reference contigs is descriptive for any bias in contig assembly. For example in the case of SR genome v017, more than 20 % of A1, A2, A3 reads align to a small set of contigs. We have to imagine that each vertical bar is a different reference contig. The best distribution is a constant one.
Even though we did not plot length of contigs, the analyzes above demonstrate that length is linearly correlated to the number of mapped reads. Therefore, peaks indicate a specific preference that reads have to map to the assembled reference.

```
custom.colors <- c(col1 = "#762a83",
```

```
                    col2 = "#1b7837",
                    col3 = "#ef8a62",
                    col4 = "#2166ac",
                    col5 = "#8c510a",
                    col6 = "#e6ab02")

histogram( ~ A1 + A2 + A3 | c('A1n', 'A2n', 'A3n'),
     data = allDF,
     nint = 50,
     scales = list(log = 2),
     type = "p",
     ylim = c(0,25),
     groups = allDF$reference,
     panel = function(...) panel.superpose(...,
         panel.groups = panel.histogram,
         col = custom.colors,
         alpha = 1),
     auto.key=list(columns=3,
         rectangles = FALSE,
         col = custom.colors),
          main = 'Peaks indicate bias mapping to one contig length',
          ylab = 'Percent of total number of mapped reads to the reference (%)',
          xlab = 'Number of mapped reads for each reference contig (log10)'
     )

Warning in histogram.formula(~A1 + A2 + A3 | c("A1n", "A2n", "A3n"), data = allDF,
:  Can't have log Y-scale
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
Warning in pmax(`c("A1n", "A2n", "A3n")` = c(FALSE, FALSE, FALSE, FALSE, :  an
argument will be fractionally recycled
Warning in id & (if (is.shingle(var)) ((var >= levels(var)[[levels[i]]][1]) &
:  longer object length is not a multiple of shorter object length
```



5 Calling SNPs: Testing tools, parameters, and filters

SNPs were called either with samtools *mpileup* function and the highest significant were selected with bcftools or they have been called with GATK. Either way SNP calling was done on each library separately. Libraries were:

- mmetsp0098
- mmetsp001433
- mmetsp00992
- mmetsp001002
- mmetsp0099
- mmetsp00100

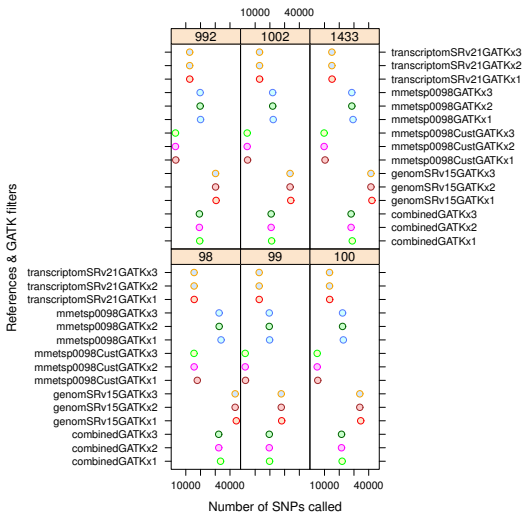
5.1 Load data

Number of SNPs called with either package were counted. Calls were done after read duplicates removal with Picard.

```
counts.SNP <- read.xlsx("./snp.counts.xlsx", sheetIndex = 1)
glimpse(counts.SNP)

Variables:
$ sample    (dbl) 98, 992, 1002, 1433, 99, 100, 98, 992, 1002, 14...
$ counts    (dbl) 351790, 395060, 427790, 389188, 309813, 425947,...
$ reference (fctr) trxSRv21, trxSRv21, trxSRv21, trxSRv21, trxSRv...
```

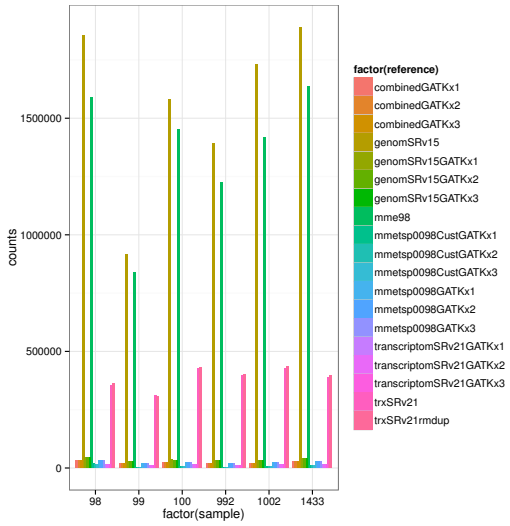
Histogram grouped by library showing difference in SNPs called relative to the reference used for mapping and the number of times GATK has been used to recalibrate calls.

```
xyplot( factor(reference) ~ as.matrix(counts) | factor(sample),
       data = counts.SNP[-c(1:24), ],
       groups = counts.SNP$reference,
       pch = 21,
       cex = 1,
       type = c("p"),
       xlab = 'Number of SNPs called',
       ylab = 'References & GATK filters')
```



Plot the difference between libraries and packages for the number of called SNPs.

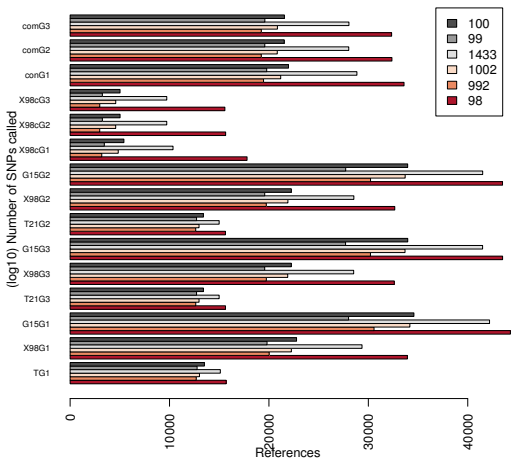
```
ggplot(counts.SNP,
       aes(x = factor(sample),
           y = counts,
           fill = factor(reference))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw()
```



Another plot for cluster analysis between references and SNPs called. I find this useful for a fast check of outliers and errors in importing data.

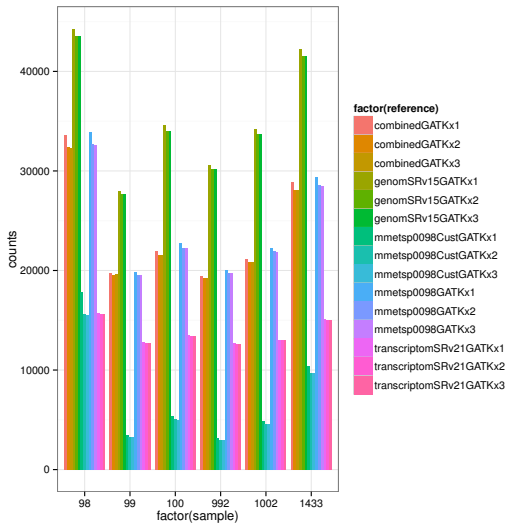
```
dat <- read.xlsx("./snp.counts.xlsx", sheetIndex = 4)
```

```
custom.colors <- c(col1 = "#b2182b",
                   col2 = "#ef8a62",
                   col3 = "#fddbc7",
                   col4 = "#e0e0e0",
                   col5 = "#999999",
                   col6 = "#4d4d4d")
barplot((as.matrix(dat[, -c(1:5)])),
        col = custom.colors,
        horiz = TRUE,
        las = 2,
        beside = T,
        legend.text = factor(dat[, 1]),
        cex.names = .7,
        ylab = '(log10) Number of SNPs called',
        xlab = 'References')
```



Plotting only the GATK called SNPs.

```
counts.SNP <- counts.SNP[-c(1:24), ]
ggplot(counts.SNP,
       aes(x = factor(sample),
           y = counts,
           fill = factor(reference))) +
    geom_bar(stat = "identity",
             position = "dodge") +
       theme_bw()
```



Plot the difference between the number of SNPs called on the 6 libraries using either the assembled or custom assembled mmetsp0098 reference. Also show the variation pattern with the number of reads used for calling SNPs. First, prepare SNP data.

```
x1 <- counts.SNP[counts.SNP$reference %in% "mmetsp0098GATKx1", ]
x2 <- counts.SNP[counts.SNP$reference %in% "mmetsp0098CustGATKx1", ]
```

Next, add the number of reads per library. This is the count of non duplicate reads that mapped to each of all the references used.

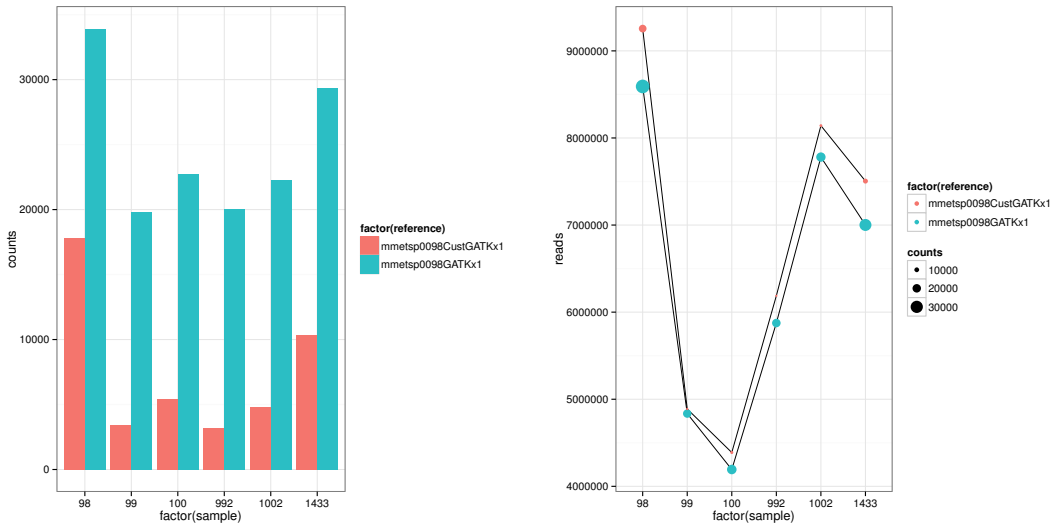
```
ref.reads <- read.xlsx("./refreads.xlsx", sheetIndex = 1)
head(ref.reads)
```

```
  sample  counts        reference
1     98 8591456 mmetsp0098GATKx1
2    992 5875110 mmetsp0098GATKx1
3   1002 7780584 mmetsp0098GATKx1
4   1433 7001081 mmetsp0098GATKx1
5     99 4835298 mmetsp0098GATKx1
6    100 4193326 mmetsp0098GATKx1
```

```
y <- ref.reads[1:12, ]
```

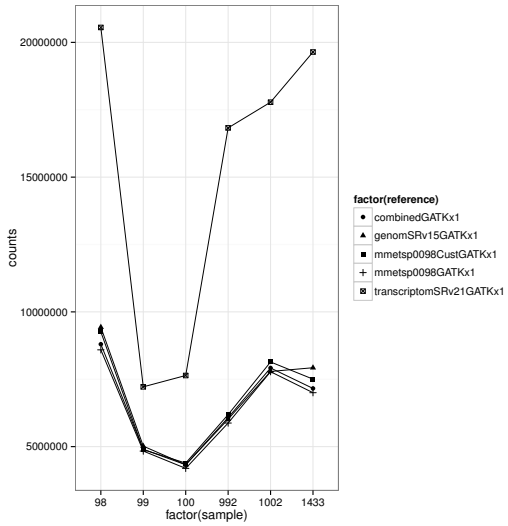
Plot difference.

```
dat <- data.frame(rbind(x1, x2), reads = y$counts)

ggplot(dat,
       aes(x = factor(sample),
           y = counts,
           fill = factor(reference))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw()

ggplot(dat,
       aes(x = factor(sample),
           y = reads,
           group = factor(reference))) +
    geom_line(size = .2)+
        geom_point(data = dat,
                   aes(x = factor(sample),
                       y = reads,
                       colour = factor(reference),
                       size = counts)) +
        theme_bw()
```



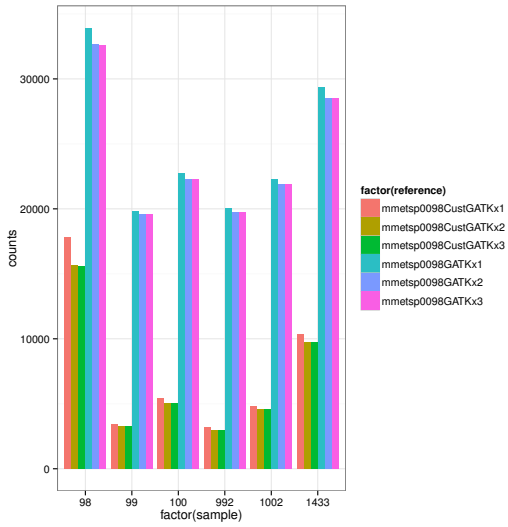
Plot number of all mapped reads for each library and for all 4 references.

```
ggplot(ref.reads,
       aes(x = factor(sample),
           y = counts,
           group = factor(reference))) +
    geom_line(size = .2) +
        geom_point(aes(shape = factor(reference))) +
            theme_bw()
```



Difference in SNPs called between the already assembled and the custom assembled *mmetsp0098* reference.

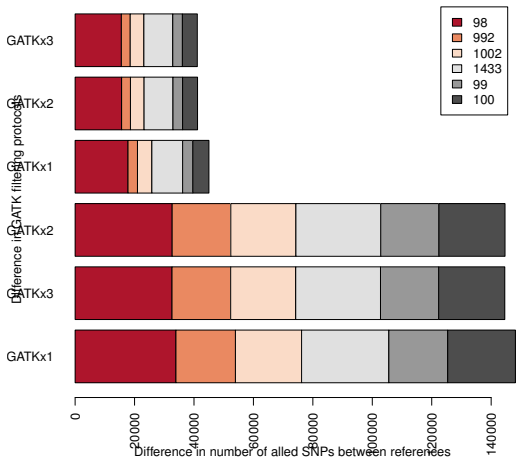
```
dat <- read.xlsx("./snp.counts.xlsx", sheetIndex = 2)
ggplot(dat,
       aes(x = factor(sample),
           y = counts,
           fill = factor(reference))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw()
```



Another way to show difference between GATK recalibration protocols decreasing the number of SNPs called after readjusting of nucleotide probabilities for each read.

```
dat <- read.xlsx("./snp.counts.xlsx", sheetIndex = 3)
```

```
custom.colors <- c(col1 = "#b2182b",
                   col2 = "#ef8a62",
                   col3 = "#fddbc7",
                   col4 = "#e0e0e0",
                   col5 = "#999999",
                   col6 = "#4d4d4d")

barplot(as.matrix(dat[, -1]),
        horiz = TRUE,
        col = custom.colors,
        xlab = "Difference in number of alled SNPs between references",
        ylab = 'Difference in GATK filtering protocols',
        las = 2,
        legend = dat$sample)
```



5.2 Final filtering

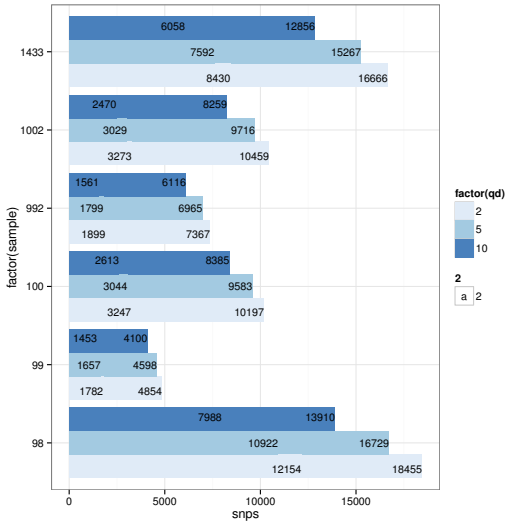
GATK hard filtering removes SNPs with low quality or confidence. This is calculated relatively to the depth of coverage. Using 3 three different thresholds for *QD* we get the number of SNPs that pass the filters.

$$QD = \frac{Confidence}{DepthCoverage} \tag{1}$$

$$DepthOfCoverage = \frac{NbOfReads \times ReadLength}{AssemblySize} \tag{2}$$

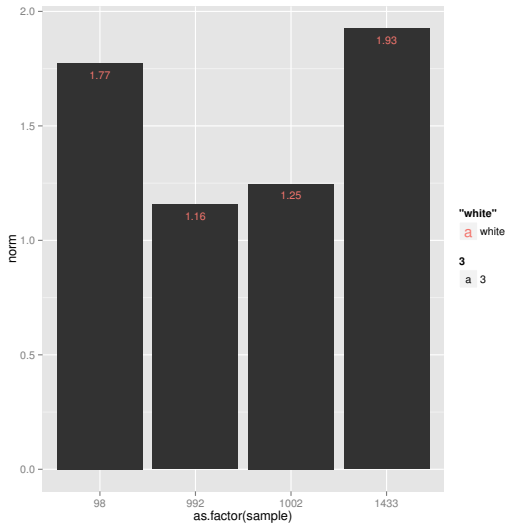
Only one reference was used here, that is Steve Roberts' genome v15.

```
dat <- read.xlsx("./hard.snps.xlsx", sheetIndex = 1)
ggplot(dat,
       aes(x = factor(sample),
           y = snps,
           fill = factor(qd)
#           group = factor(reference)
           )) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw() +
            geom_text(aes(x = factor(sample),
                          y = snps,
                          ymax = snps,
                          label = snps,
                          size = 2,
                          hjust = 1),
                      position = position_dodge(width=1)) +
                    coord_flip() +
                        scale_fill_brewer()
```



Number of SNPs per strain at *QD = 5*. SNPs called against *SR genome v15*. The number of reads (approx 100 nt) per library has been counted and plotted above, the data is in *refreads.xlsx*.

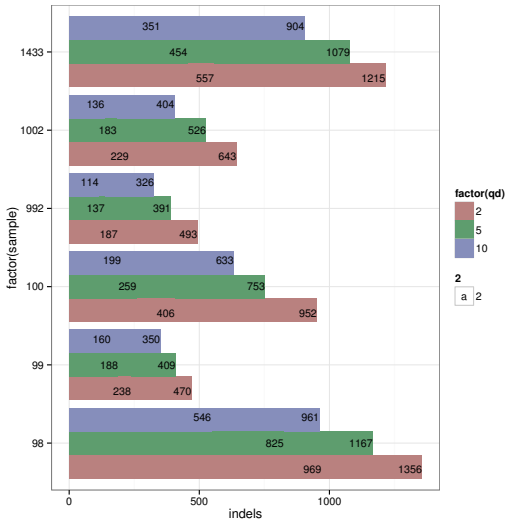
```
dat <- read.xlsx("./hard.snps.xlsx", sheetIndex = 1)
dat <- dat[7:10, 1:2]
dat$Treads <- ref.reads[c(19, 22, 20, 21), 2]
dat$norm <- with(dat, (snps/Treads)*1000)

ggplot(dat,
       aes(x = as.factor(sample),
           y = norm)) +
    geom_bar(stat = "identity") +
        geom_text(aes(x = as.factor(sample),
                      y = norm,
                      ymax = norm,
                      label = round(norm, digits = 2),
                      color = "white",
                      vjust = 2,
                      size = 3))
```



We can also do the same thing with indels.

```
dat <- read.xlsx("./hard.snps.xlsx", sheetIndex = 1)
```

```
ggplot(dat,
       aes(x = factor(sample),
           y = indels,
           fill = factor(qd))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw() +
            geom_text(aes(x = factor(sample),
                          y = indels,
                          ymax = indels,
                          label = indels,
                          size = 2,
                          hjust = 1),
                      position = position_dodge(width=1)) +
                    coord_flip() +
                        scale_fill_hue(c = 40, l = 60)
```



6 Working with a combined assembly

The combined assembly is already published. It is added here with the other references because it is heavily annotated and their contigs are extensively mapped. Future analysis of SNPs and QPX genome structure depends on good annotation data. Load in new mapped data to the combined reference:

```
combined <- read.xlsx("./snp.counts.xlsx", sheetIndex = 1)
glimpse(combined)

Variables:
$ sample    (dbl) 98, 992, 1002, 1433, 99, 100, 98, 992, 1002, 14...
$ counts    (dbl) 351790, 395060, 427790, 389188, 309813, 425947,...
$ reference (fctr) trxSRv21, trxSRv21, trxSRv21, trxSRv21, trxSRv...
```

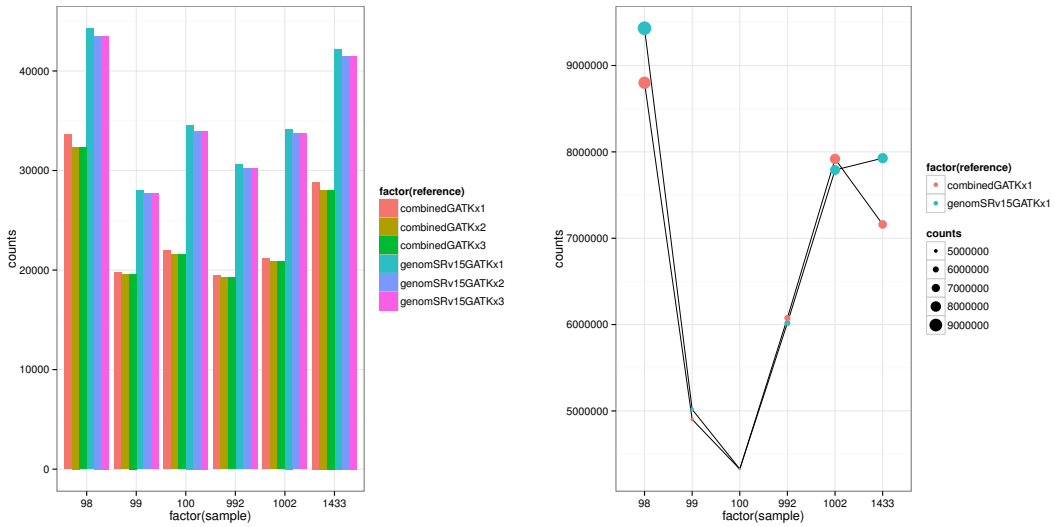
Difference in SNPs called between the genome v15 of S. Roberts and the official combined assembly. First extract relative rows.

```
dev <- paste("genomSRv15GATKx", seq(1,3,1), sep = "")
ser <- paste("combinedGATKx", seq(1,3,1), sep = "")
difference <- rbind(combined[combined$reference %in% dev, ],
                    combined[combined$reference %in% ser, ])

d.ref <- ref.reads[c(19:30), ]
```

Plot difference.

```
ggplot(difference,
```

```
       aes(x = factor(sample),
           y = counts,
           fill = factor(reference))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw()

ggplot(d.ref,
       aes(x = factor(sample),
           y = counts,
           group = factor(reference))) +
    geom_line(size = .2)+
        geom_point(data = d.ref,
                   aes(x = factor(sample),
                       y = counts,
                       colour = factor(reference),
                       size = counts)) +
        theme_bw()
```

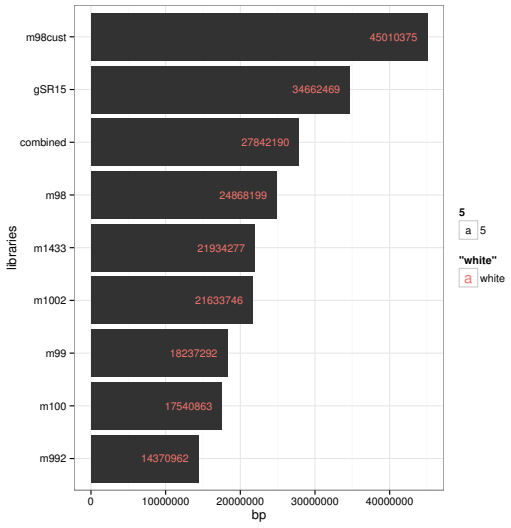


7 Descriptive stats of all processed libraries

This following section shows the mean length of all sequences assembled from each library, the number of base pairs per library, the identified protein features from these sequences, and the number of functional enzymes identified by mapping to public libraries. It is to note the number of predicted and identified rRNA features in each of these libraries is significantly low. Regress different variables on each others for visualization purposes.

```
stats <- read.xlsx("./libraries.xlsx", sheetIndex = 1)
```

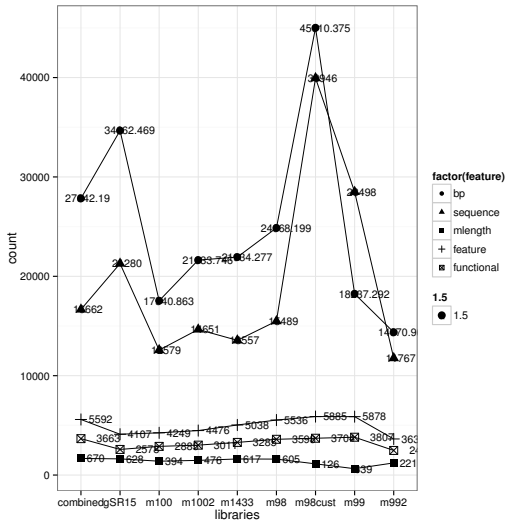
```
rstats <- stats[complete.cases(stats), ]
#rownames(rstats) <-  stats[, 1]

# The whole new magical script
# job: order columns
# dependecies: dplyr
rstats <- within(rstats,
                 libraries <- factor(libraries,
                                     levels = arrange(rstats,
                                         bp)$libraries))

ggplot(rstats,
       aes(x = libraries,
           y = bp)) +
    geom_bar(stat = "identity") +
        theme_bw() +
            coord_flip() +
                geom_text(aes(x = libraries,
                              y = bp,
                              ymax = bp,
                              label = bp,
                              size = 5,
                              color = "white",
                              hjust = 1.2))
```



Difference between libraries in the number of base pair (bp), which must be multiplied by 1000 bp, identified protein features inside assembled sequences (feature), functional sequences in the contigs (function), the mean length in each library, and the number of contigs (sequence) assembled from raw reads after trimming and duplicate removal (all basic quality controls).

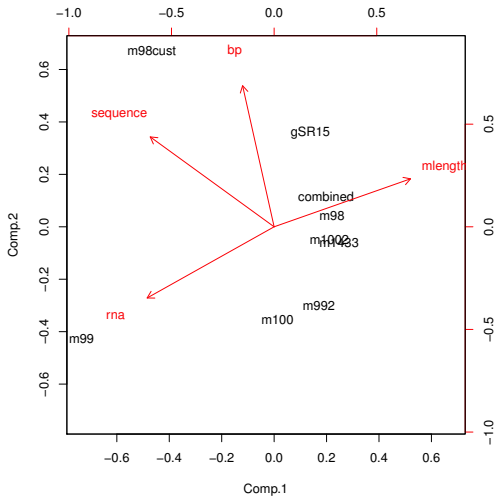
```
stats <- read.xlsx("./libraries.xlsx", sheetIndex = 1)
```

```
rstats <- stats[complete.cases(stats), ]
rstats$bp <- rstats$bp/1000
#rstats <- rename(rstats, bpx1000 = bp)
rstats <- gather(rstats, "feature", "count", c(2:4, 7:8))
ggplot(rstats,
       aes(x = libraries,
           y = count,
           group = factor(feature))) +
    geom_line(size = .2) +
        geom_point(aes(shape = factor(feature),
                       size = 1.5)) +
                       theme_bw() +
                           geom_text(aes(x = libraries,
                                         y = count,
                                         ymax = count,
                                         label = count,
                                         size = 1.5,
                                         hjust = ifelse(sign(count)>1, .5, 0)),
                                     position = position_dodge(width = 1))
```



Principal component analysis and diagnostics.

```
stats <- read.xlsx("./libraries.xlsx", sheetIndex = 1)
rownames(stats) <- stats$libraries
rstats <- stats[complete.cases(stats), -1]
rstats <- decostand(rstats, method = "range")
p = princomp(~bp + mlength + sequence + rna
   , data= rstats)
summary(p)

Importance of components:
                       Comp.1 Comp.2 Comp.3   Comp.4
Standard deviation      0.459  0.382 0.1401 0.017928
Proportion of Variance  0.560  0.387 0.0522 0.000855
Cumulative Proportion   0.560  0.947 0.9991 1.000000

biplot(p)
```



Finally a summary of all sequence data.

```
stats[, -1]

               bp sequence mlength   sd mgc feature functional rna
m98      24868199    15489    1605 1765  45    5536       3598  21
m98cust  45010375    39946    1126 1505  42    5885       3704  60
m992     14370962    11767    1221  921  46    3632       2476   7
m1433    21934277    13557    1617 1677  46    5038       3285  29
m1002    21633746    14651    1476 1133  45    4476       3011   9
m99      18237292    28498     639  504  49    5878       3807 229
m100     17540863    12579    1394 1042  46    4249       2885 121
gSR15    34662469    21280    1628 2907  44    4107       2578   9
combined 27842190    16662    1670 1908  45    5592       3663  34
```

8 Applied annotations, subsytem predictions, and taxonomic distribution

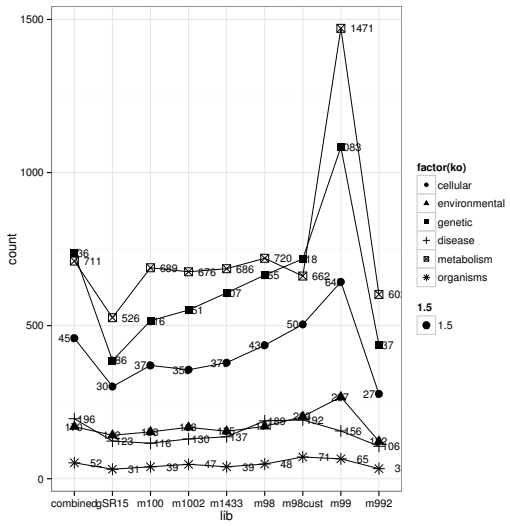
Like the title implies, identified and predicted annotations and protein features are mapped to public sequence libraries. Reshape data, transform columns into rows.

```
predicted <- read.xlsx("./libraries.xlsx", sheetIndex = 3)
predicted <- gather(predicted, "ko", "count", 3:8, na.rm = TRUE)
summary(predicted)

       lib      chart                ko         count
 combined: 6   ko:54   cellular      :9   Min.   :  31
 gSR15   : 6           environmental:9   1st Qu.: 132
 m100    : 6           genetic       :9   Median : 272
 m1002   : 6           disease       :9   Mean   : 361
 m1433   : 6           metabolism    :9   3rd Qu.: 589
 m98     : 6           organisms     :9   Max.   :1471
 (Other) :18
```

Plot difference in identified protein features between libraries.

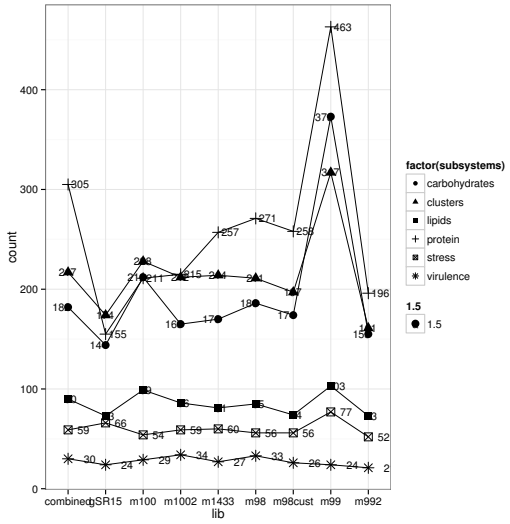
```
ggplot(predicted,
```

```
      aes(x = lib,
          y = count,
          group = factor(ko))) +
    geom_line(size = .2) +
      geom_point(aes(shape = factor(ko),
                     size = 1.5)) +
                     theme_bw() +
                         geom_text(aes(x = lib,
                                       y = count,
                                       ymax = count,
                                       label = count,
                                       size = 1.5,
                                       hjust = ifelse(sign(count)>1, .5, 0)),
                                   position = position_dodge(width = 1))
```



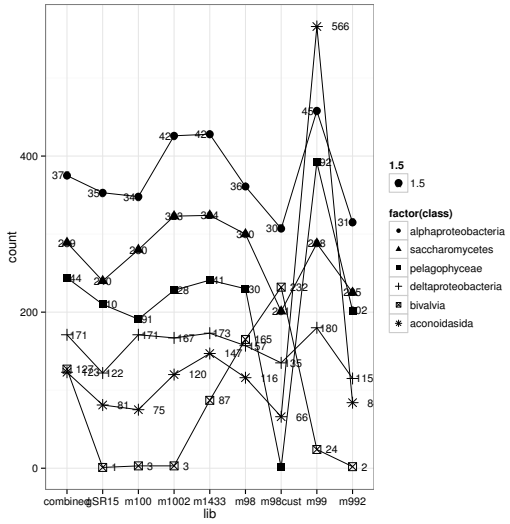
In this next snippet subsystems are discussed. Functional coupling and chromosomal clusters are shown for *clustering-based subsytems* among other subsytems.

```
predicted <- read.xlsx("./libraries.xlsx", sheetIndex = 4)
predicted <- gather(predicted, "subsystems", "count", 3:8, na.rm= TRUE)
ggplot(predicted,
       aes(x = lib,
           y = count,
           group = factor(subsystems))) +
    geom_line(size = .2) +
      geom_point(aes(shape = factor(subsystems),
                     size = 1.5)) +
                     theme_bw() +
                         geom_text(aes(x = lib,
                                       y = count,
                                       ymax = count,
                                       label = count,
                                       size = 1.5,
                                       hjust = ifelse(sign(count)>1, .5, 0)),
                                   position = position_dodge(width = 1))
```



Finally, a taxonomic classification on sequence similarities gives insights on sequence relatedness or sample contamination. Five classes were selected, bacteria, fungi, algae, parasite, and bivalvia.

```
predicted <- read.xlsx("./libraries.xlsx", sheetIndex = 5)
predicted <- gather(predicted, "class", "count", c(3,5:9), na.rm = TRUE)
ggplot(predicted,
       aes(x = lib,
           y = count,
           group = factor(class))) +
    geom_line(size = .2) +
        geom_point(aes(shape = factor(class),
                       size = 1.5)) +
                       theme_bw() +
                           geom_text(aes(x = lib,
                                         y = count,
                                         ymax = count,
                                         label = count,
                                         size = 1.5,
                                         hjust = ifelse(sign(count)>1, .5, 0)),
                                     position = position_dodge(width = 1))
```



9 Shared SNPs between libraries

Shared SNPs between libraries mapped to SR genome v15.

```
shared.snps <- read.table("./shared.snps.txt", fill =TRUE)
```

```
shared.snps

                  X132 X1433.0.9.. X992.1.9..
200           98(1.2%)   992(2.9%)
314         1002(3.2%)  1433(2.1%)
328         1002(3.4%)  1433(2.1%)  992(4.7%)
587         1002(6.0%)    98(3.5%)  992(8.4%)
589         1433(3.9%)    98(3.5%)  992(8.5%)
632         1002(6.5%)    98(3.8%)
655          992(9.4%)
825         1002(8.5%)  992(11.8%)
1679       1002(17.3%)
1702       1002(17.5%) 1433(11.1%)  98(10.2%)
2577       1433(16.9%)
3394         98(20.3%)
3649       1002(37.6%) 1433(23.9%)  98(21.8%)
992(52.4%)
5976       1433(39.1%)   98(35.7%)
```

Shared indels between libraries mapped to SR genome v15.

```
shared.indels <- read.table("./shared.indels.txt", fill = TRUE)
shared.indels

                    X8   X98.0.7.. X992.2.0..
12          1433(1.1%)   992(3.1%)
14          1002(2.7%)  1433(1.3%)  992(3.6%)
15          1002(2.9%)  1433(1.4%)
31          1002(5.9%)    98(2.7%)
40          1433(3.7%)    98(3.4%) 992(10.2%)
41          1002(7.8%)    98(3.5%) 992(10.5%)
62         1002(11.8%)  992(15.9%)
68          992(17.4%)
78         1002(14.8%)  1433(7.2%)   98(6.7%)
139        1002(26.4%)
146        1002(27.8%) 1433(13.5%)  98(12.5%)
992(37.3%)
267        1433(24.7%)
316          98(27.1%)
507        1433(47.0%)   98(43.4%)
```

10 Component analysis and sequence closeness

Import annotated data.

```
closeness <- read.csv("./pca.csv", sep = "\t")
```

```
summary(closeness)

      metagenome
 mmetsp1002:1141
 mmetsp1433:1278
 mmetsp98  :1279
 mmetsp992 :1088
 QPX_v15   :1030


                                                level.1
 Carbohydrates                                      : 848
 Amino Acids and Derivatives                        : 785
 Protein Metabolism                                 : 769
 Clustering-based subsystems                        : 559
 Miscellaneous                                      : 552
 Cofactors, Vitamins, Prosthetic Groups, Pigments: 462
 (Other)                                            :1841
                           level.2
 0                              : 750
 Plant-Prokaryote DOE project   : 506
 Protein biosynthesis           : 499
 RNA processing and modification: 346
 Central carbohydrate metabolism: 326
 Folate and pterines            : 278
 (Other)                        :3111
                                 level.3
 YgfZ                                  : 120
 Ribosome LSU eukaryotic and archaeal: 101
 Proteasome eukaryotic                 :  91
 Ribosome SSU eukaryotic and archaeal:  91
 Serine-glyoxylate cycle               :  88
 tRNA modification Bacteria            :  76
 (Other)                               :5249
                                                    function.
 GTP cyclohydrolase I (EC 3.5.4.16) type 1               :  70
 Acetyl-CoA acetyltransferase (EC 2.3.1.9)               :  55
 Serine hydroxymethyltransferase (EC 2.1.2.1)            :  50
 Cysteine desulfurase (EC 2.8.1.7)                       :  48
 3-ketoacyl-CoA thiolase (EC 2.3.1.16)                   :  40
 Branched-chain amino acid aminotransferase (EC 2.6.1.42):  40
 (Other)                                                 :5513
   abundance      avg.eValue    avg...ident     avg.align.len
 1      :3943   Min.   :-269   Min.   :-183.0   Min.   : 24
 2      :1140   1st Qu.: -57   1st Qu.:  63.2   1st Qu.: 61
 3      : 280   Median : -29   Median :  66.2   Median : 92
 4      : 155   Mean   : -41   Mean   :  65.7   Mean   :118
 5      :  69   3rd Qu.: -15   3rd Qu.:  70.2   3rd Qu.:156
 6      :  68   Max.   :   3   Max.   :  95.6   Max.   :544
 (Other): 161
    X..hits          X
 Min.   :  1   Min.   :1
 1st Qu.:  1   1st Qu.:1
 Median :  1   Median :1
 Mean   :  4   Mean   :1
 3rd Qu.:  2   3rd Qu.:2
 Max.   :457   Max.   :3
               NA's   :5727

closeness <- closeness[, c(1, 7:10)]
```

Principal component analysis on 5 libraries, 4 strains and the genome (v15), using an *identity score* for annotating a sequence and an *alignment length score* for similarities with a functional feature, an *e-value score* for functional similarities, and the *number of hits*, ie., the number of times a function is identified in a library.

```
rownames(closeness) <- paste(closeness[, 1], 1:nrow(closeness), sep = ".")
x=closeness[, -1]
head(x)
```

```
             avg.eValue avg...ident avg.align.len X..hits
mmetsp1433.1        -57        74.7           142       1
QPX_v15.2           -57        74.7           142       1
mmetsp98.3          -57        74.7           142       1
mmetsp1002.4        -57        74.7           142       1
mmetsp1433.5        -60        64.7           173       1
QPX_v15.6           -60        64.7           173       1
```

```
results <- decostand(x, method = "range")
head(results)
```

```
             avg.eValue avg...ident avg.align.len X..hits
mmetsp1433.1      0.779       0.925         0.227       0
QPX_v15.2         0.779       0.925         0.227       0
mmetsp98.3        0.779       0.925         0.227       0
mmetsp1002.4      0.779       0.925         0.227       0
mmetsp1433.5      0.768       0.889         0.287       0
QPX_v15.6         0.768       0.889         0.287       0
```

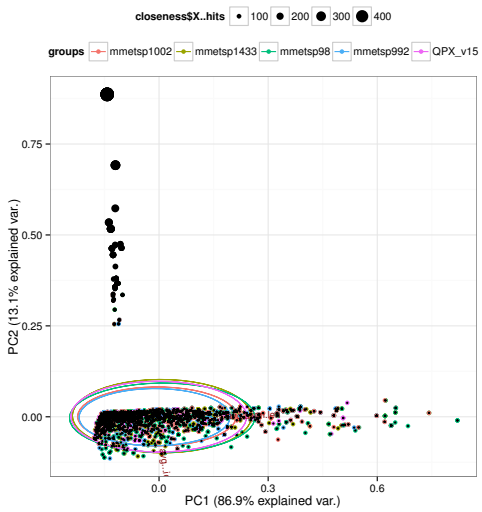
```
p = princomp(~ avg...ident + avg.align.len
   , data= results)
summary(p)
```

```
Importance of components:
                       Comp.1 Comp.2
Standard deviation      0.156 0.0605
Proportion of Variance  0.869 0.1309
Cumulative Proportion   0.869 1.0000
```

```
#plot(p, type = "l")
#biplot(p, cex = .4)
```

Clustering and visualization of all sequences without applying any filters.

```
ggbiplot(p, obs.scale = 1,
         var.scale = 1,
         groups = closeness$metagenome,
         ellipse = TRUE,
         circle = FALSE) +
        geom_point(aes(size = closeness$X..hits)) +
        theme_bw() +
            theme(legend.direction = 'horizontal',
                  legend.position = 'top')
```



Build a custom PCA function for repettive iterations.

```
customBiplot <- function(data, method){
x=data[, -1]
results <- decostand(x, method = method)
p = princomp(~ results[, 2] + results[, 3]
   , data= results)
ggbiplot(p, obs.scale = 1,
         var.scale = 1,
         groups = data$metagenome,
         ellipse = TRUE,
         circle = FALSE) +
        theme_bw() +
            theme(legend.direction = 'horizontal',
                  legend.position = 'top')
}
```

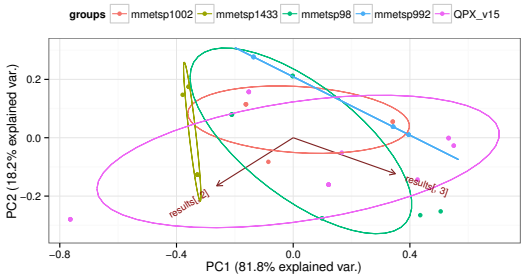
Filter sequences depending on their alignment length and the abundance of a function.

↰ results[,2] = identity and results[,3] = alignment length

```
closenessX <- filter(closeness, avg.align.len < 50, X..hits > 2)
dim(closenessX)[1]
```

```
[1] 57
```

```
customBiplot(closenessX, method = "range")
```

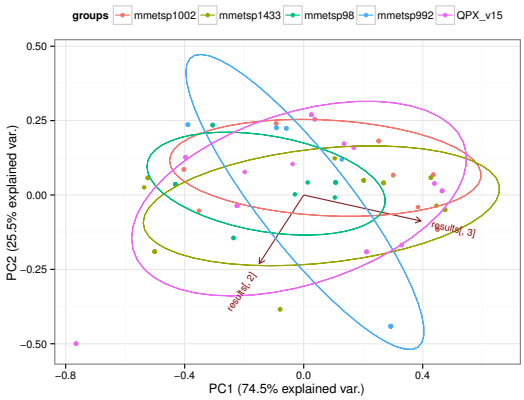


Select higher alignment similarities.

```
closenessX <- filter(closeness, avg.align.len < 60, X..hits > 2)
dim(closenessX)[1]
```

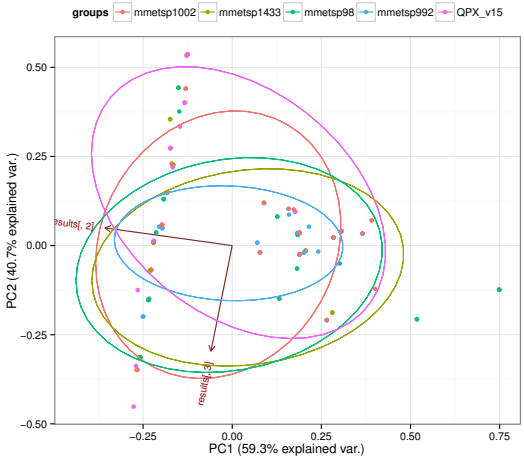
```
[1] 117
```

```
customBiplot(closenessX, method = "range")
```



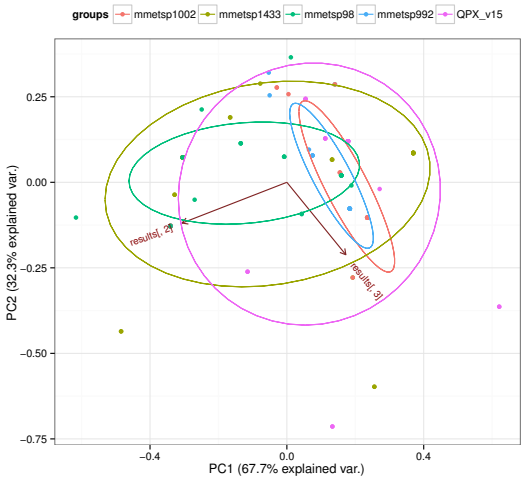
Select even higher alignment similarities.

```
closenessX <- filter(closeness, avg.align.len < 100, X..hits > 4)
dim(closenessX)[1]

[1] 199

customBiplot(closenessX, method = "range")
```



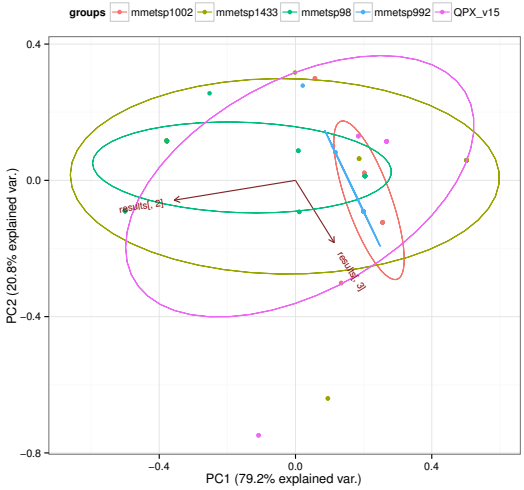
Select on the criteria of e-Value and abundance of a functional sequence.

```
closenessX <- filter(closeness, avg.eValue < -40, X..hits > 2)
dim(closenessX)[1]

[1] 152

customBiplot(closenessX, method = "range")
```



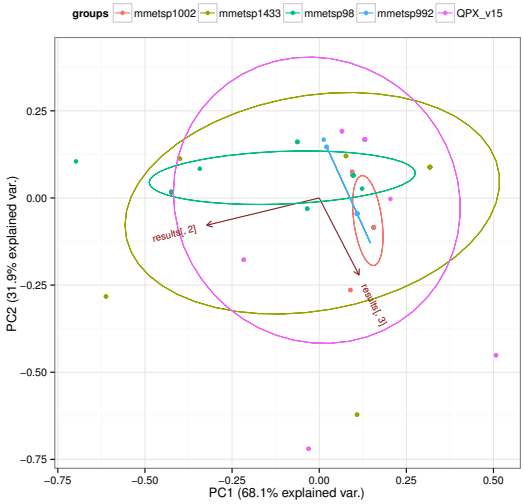
Select on the criteria of e-Value and abundance of a functional sequence.

```
closenessX <- filter(closeness, avg.eValue < -40, X..hits > 3)
dim(closenessX)[1]

[1] 126

customBiplot(closenessX, method = "range")
```



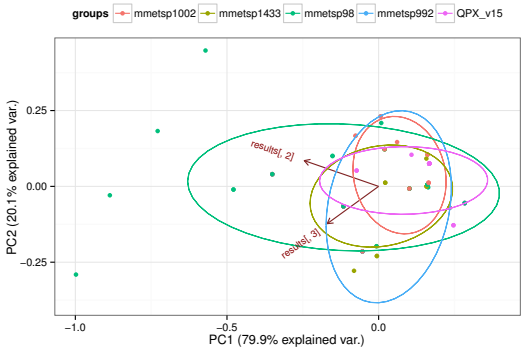
Select on the criteria of e-Value and abundance of a functional sequence.

```
closenessX <- filter(closeness, avg.eValue < -50, X..hits > 2)
dim(closenessX)[1]

[1] 109

customBiplot(closenessX, method = "range")
```



Select on the criteria of alignment length. Since SNP aggregation tests are the next step in this analysis, the length of a correct alignment is technically helpful in differentiating SNP position. And abundance will be more than 2 to increase probabilities of correct functional annotation.

```
closenessX <- filter(closeness, avg.align.len > 200, X..hits >= 2)
dim(closenessX)[1]

[1] 100

customBiplot(closenessX, method = "range")
```

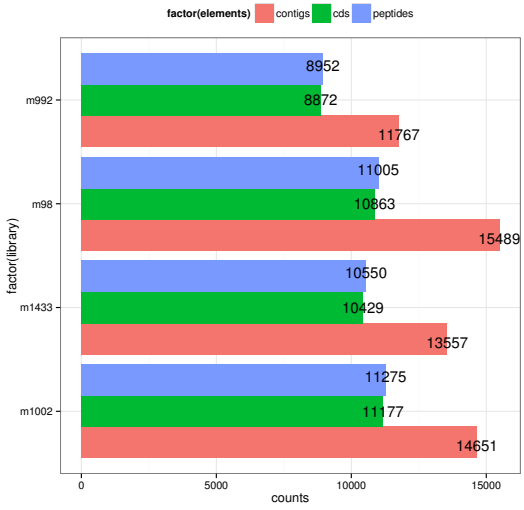


11 Aggregation analysis of SNPS

MMETSP libraries are already been annotated. How many contigs, peptide and cds elements are indexed?

```
contigs <- read.xlsx("./annot.stats.xlsx", sheetIndex = 1)
```

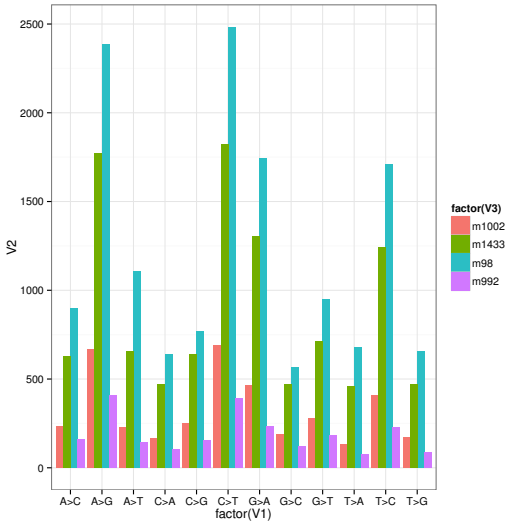
```
contigs <- gather(contigs, "elements", "counts", 2:4)
ggplot(contigs,
       aes(x = factor(library),
           y = counts,
           fill = factor(elements))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw() +
            theme(legend.direction = 'horizontal',
                  legend.position = 'top') +
                coord_flip() +
                    geom_text(aes(x = factor(library),
                                  y = counts,
                                  ymax = counts,
                                  label = counts,
                                  hjusts = ifelse(sign(counts)> 0,1,0)),
                              position = position_dodge(width = 1))
```



11.1 Preferential substitution

Preferential substitution of nucleotides. It should be noted that *mmetsp0098* and *mmetsp1433* are both bigger in library size than the others. Therefore comparison of SNPs should be done for each library separately. However there is a resemblance in substitution between libraries since the pattern is quite similar for all nucleotides.

```
prefs <- read.table("./all.stats.txt")
prefs$V3 <- c(rep("m1002", 12),
              rep("m98", 12),
              rep("m992", 12),
              rep("m1433", 12))
ggplot(prefs,
       aes(x = factor(V1),
           y = V2,
           fill = factor(V3))) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw()
```



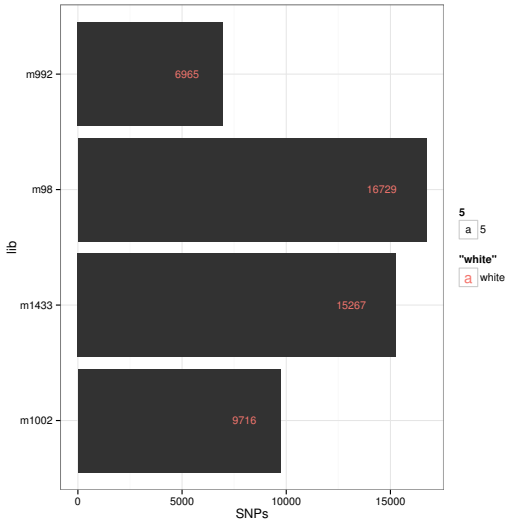
After hard filtering SNPs to the minimum from all 4 libraries, *DISCARD*-labelled SNPs were removed. The remaining were imported into data frames with the following columns.

1. CHROM: number of contig
2. POS: SNP position on that contig
3. ALT: alternative SNP to the reference
4. AD: allelic depth for the reference and ALT alleles
5. DP: approximate read depth
6. GQ: genotype quality
7. PL: normalized phred scaled likelihoods

The structure of the data frame is similar to the *iris* data.

↰ With low number of sample it is impossible to create a f(SNP)=strain machine learning framework. To make a binary table of SNPs at least 100 samples must be used.

```
x <- c('m98', 'm1433', 'm1002', 'm992')
y <- c(16729, 15267, 9716, 6965)
dat <- data.frame(lib = x, SNPs = y)
ggplot(dat,
       aes(x = lib,
           y = SNPs)) +
    geom_bar(stat = "identity") +
        theme_bw() +
            coord_flip() +
                geom_text(aes(x = lib,
                              y = SNPs,
                              ymax = SNPs,
                              label = SNPs,
                              size = 5,
                              col = "white",
                              hjust = 2))
```



Import SNP data: data manipulation process of removing NAs and getting the same number of SNPs across all samples.

```
m98 <- read.table("./m98.ml.txt", fill = NA)
m1433 <- read.table("./m1433.ml.txt", fill = NA)
m992 <- read.table("./m992.ml.txt", fill = NA)
m1002 <- read.table("./m1002.ml.txt", fill = NA)

colnames(m98) <- c('contigs', 'pos', 'ad1', 'ad2',
                   'dp', 'gq', 'pl1', 'pl2', 'pl3', 'lib')
colnames(m1433) <- c('contigs', 'pos', 'ad1', 'ad2',
                   'dp', 'gq', 'pl1', 'pl2', 'pl3', 'lib')
colnames(m992) <- c('contigs', 'pos', 'ad1', 'ad2',
                   'dp', 'gq', 'pl1', 'pl2', 'pl3', 'lib')
colnames(m1002) <- c('contigs', 'pos', 'ad1', 'ad2',
                   'dp', 'gq', 'pl1', 'pl2', 'pl3', 'lib')

m98 <- m98[complete.cases(m98), ]
m1433 <- m1433[complete.cases(m1433), ]
m992 <- m992[complete.cases(m992), ]
m1002 <- m1002[complete.cases(m1002), ]

m98 <- m98[! m98$lib != 'm98', ]
m1433 <- m1433[! m1433$lib != 'm1433', ]
m992 <- m992[! m992$lib != 'm992', ]
m1002 <- m1002[! m1002$lib != 'm1002', ]

m1433$pl3 <- as.numeric(m1433$pl3)

m98$lib <- factor(m98$lib, "m98")
m992$lib <- factor(m992$lib, "m992")
m1433$lib <- factor(m1433$lib, "m1433")
m1002$lib <- factor(m1002$lib, "m1002")

index <- min(dim(m98)[1], dim(m1433)[1],
             dim(m992)[1], dim(m1002)[1])
set.seed(123)
mall <- rbind(m98[sample(nrow(m98), index), ],
              m1433[sample(nrow(m1433), index), ],
              m992[sample(nrow(m992), index), ],
              m1002[sample(nrow(m1002), index), ])
dim(mall)

[1] 27844    10
```

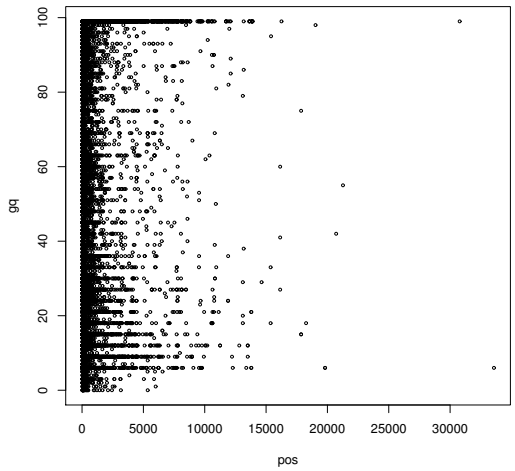
11.2 Available data

Regressing the genome quality of SNPs on the position of the SNPs inside a contig. This shows that SNPs are concentrated in the first 10 Kb.

```
head(mall)
```

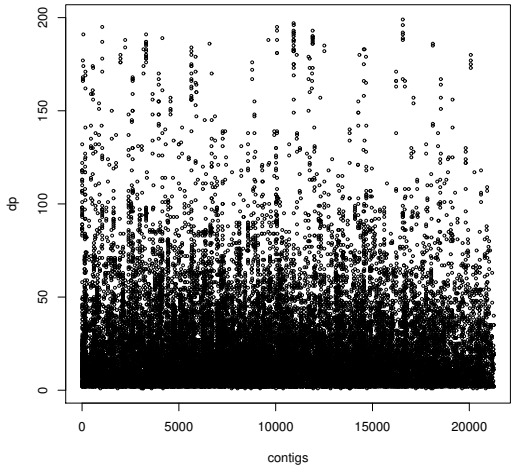
```
      contigs pos ad1 ad2 dp gq pl1 pl2 pl3 lib
4825     5776 244   5  18 23 99 786   0 150 m98
13214   14964  96   6   6 12 99 234   0 234 m98
6857     7810 176   3  11 14 68 392   0  68 m98
14801   17502  56   9   7 16 99 230   0 337 m98
15760   19016  80   0   7  7 21 315  21   0 m98
763       884 810  17  29 46 99 986   0 516 m98
```

```
with(mall, plot(pos, gq, cex = .5))
```



Regression of contigs and the read depth for each SNPs in those contigs. When using libraries mapped to the combined assembly (as a reference transcriptome) the plot shows that the depth of coverage distinguishes between 2 different subsets of contigs. However the regression is constant when using the genome of SR v15 as a reference for mapping the libraries (as shown below).

```
with(mall, plot(contigs, dp, cex = .5))
submall <- filter(mall, dp > 50, pos <= 10000)
```

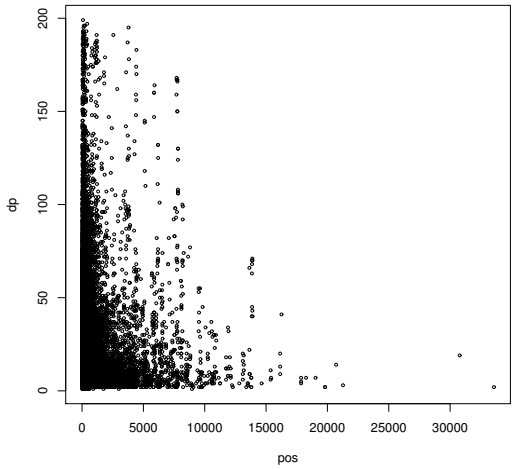


This plot shows that 11.41 % of the SNPs have a depth over 50 for the first 10 Kb.

```
with(mall, plot(pos, dp, cex = .5))
```

```
## percentage of SNPs with read depth higher than 35
(nrow(submall)/nrow(mall))*100
```

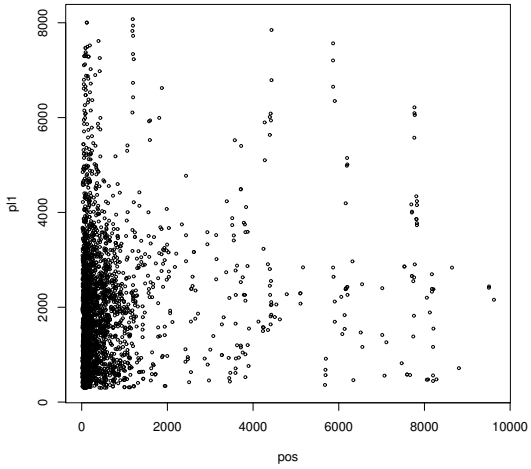
```
[1] 11.4
```



Plotting only SNPs with DP > 50 and in contigs which length <= 10 Kb, and regressing toward a phred-scaled adjusted likelihood for each variant or genotype likelihood.

```
with(submall, plot(pos, pl1, cex = .5))
summary(submall$lib)
```

```
  m98 m1433  m992 m1002
 1256  1127   417   377
```



Linear regression between position of the SNP and the normalized phred scaled likelihood, which on its own is an accuracy determination score. Phred likelihoods (PL) are computed for the REF/REF, REF/ALT, and ALT/ALT variants. To convert a PL to a raw likelihood L:

$$P(L|AA) = 10^{-P_{AA}/10} \tag{3}$$

These probabilities are adjusted with phred scores. They determine the probability of a base observed given a reference genotype, an heterozygous genotype or a non-reference genotype respectively (pl1, pl2, and pl3).
Accordingly, REF/REF (pl1) is significant. Meaning the genotype we have is homozygous for the reference nucleotide (not the variant), but if a variant exists, thus it represents a rare mutation (*reference needed*). Therefore, the raw likelihoods must be calculated with the equation above for the picked variants and the genotype with *P=1* is the most significant genotype at that nucleotide.

```
fit <- lm(pos~pl1, data = submall)
summary(fit)


Call:
lm(formula = pos ~ pl1, data = submall)

Residuals:
   Min     1Q Median     3Q    Max
 -1353   -484   -344    -97   8996

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 315.2290    39.9153    7.90  3.9e-15 ***
pl1           0.1442     0.0171    8.42  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1280 on 3175 degrees of freedom
Multiple R-squared:  0.0219,Adjusted R-squared:  0.0216
F-statistic:   71 on 1 and 3175 DF,  p-value: <2e-16
```

Lets get the variants with the highest probability that a genotype has been identified. *PL=1* determines the genotype, either homozygous for REF (pl1) or ALT (pl3) or heterozygous REF/ALT (pl2).

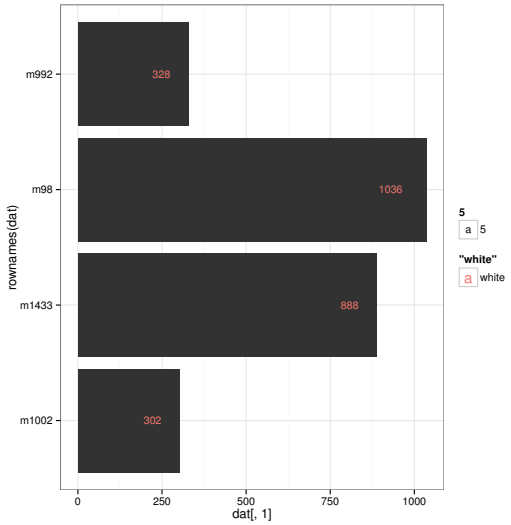
```
submall[, 7:9] <- apply(submall[, 7:9], 2, function(x) 10^(-x/1000))
head(submall)

  contigs pos ad1 ad2  dp gq     pl1 pl2        pl3 lib
1    6470 215  43  21  64 99 0.18113   1 0.01717908 m98
2   20921  54  88  21 109 99 0.22751   1 0.00000196 m98
3   13280  47  78  19  97 99 0.27542   1 0.00054954 m98
4   10194 378   5  49  54 63 0.01064   1 0.86496792 m98
5   19812  80  55  41  96 99 0.04027   1 0.00693426 m98
6    4446  65  15  65  80 99 0.00514   1 0.49545019 m98
```

Lets extract all heterozygous alleles with at least 90 % confidence.

```
heteromall <- filter(submall, pl2 >= .9)
```

The original number of SNPs was 3177 among which the number of variants with heterozygous genotype is 2554.

```
dat <- as.data.frame(summary(heteromall$lib))
ggplot(dat,
       aes(x = rownames(dat),
           y = dat[, 1])) +
    theme_bw() +
        geom_bar(stat = "identity") +
            coord_flip() +
                geom_text(aes(x = rownames(dat),
                              y = dat[, 1],
                              ymax = dat[, 1],
                              size = 5,
                              label = dat[, 1],
                              col = "white",
                              hjust = 2))
```

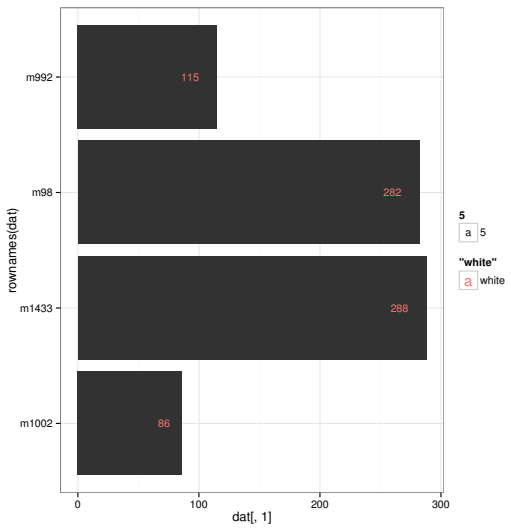


Now lets get the homozygous variants with genotype ALT/ALT with 90 %.

```
altmall <- filter(submall, p13 >= .9)
```

The number of variants ALT/ALT is 771. Interesting thing is that using the combined assembly as a reference, m1433 had also the highest number of homozygous alleles while m98 had half the number shown below.

```
dat <- as.data.frame(summary(altmall$lib))
ggplot(dat,
       aes(x = rownames(dat),
           y = dat[, 1])) +
    theme_bw() +
        coord_flip() +
            geom_bar(stat = "identity") +
                geom_text(aes(x = rownames(dat),
                              y = dat[, 1],
                              ymax = dat[, 1],
                              label = dat[, 1],
                              size = 5,
                              color = "white",
                              hjust = 2))
```



12 Protein domain annotation

Get the number of protein domains from the MMETSP strains. First, contigs must be translated into peptides. HMMER3.2b was used for annotation. Hidden Markov Models were generated on Pfam database. The table below lists old and new annotations against old and new Pfam v26 and v28 libraries. (> 2 years interval between versions).

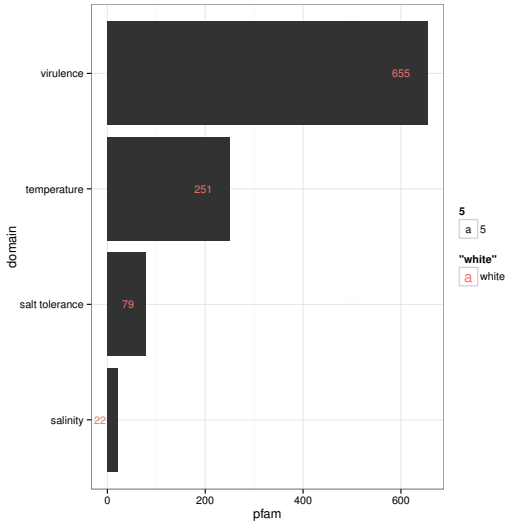
```
pfam <- read.xlsx("./pfam.xlsx", sheetIndex = 1)
pfam
```

```
          domain pfam  a98 s98 a992 s992 a1002 s1002 a1433 s1433
1      virulence  655 5098 313 3075  261  4606   291  4794   308
2    temperature  251 2484 168 1680  141  2283   164  2277   161
3       salinity   22  163  13   91    9   123    10   137    12
4 salt tolerance   79 2231  70 1422   64  2097    66  2078    66
5      virulence  655 5306 331 3185  275  4763   302  4973   326
6    temperature  251 2704 179 1771  145  2436   170  2478   170
7       salinity   22  161  12   97   10   128    10   138    10
8 salt tolerance   79 2267  73 1451   68  2108    67  2138    69
  annot
1   old
2   old
3   old
4   old
5   new
6   new
7   new
8   new
```

Number of domains found in Pfam v28 for :

- Virulence
- Temperature
- Salinity
- Salt tolerance

```
ggplot(pfam[1:4, ],
       aes(x = domain,
           y = pfam)) +
    coord_flip() +
        theme_bw() +
            geom_bar(stat = "identity") +
                geom_text(aes(x = domain,
                              y = pfam,
                              ymax = pfam,
                              label = pfam,
                              size = 5,
                              color = "white",
                              hjust = 2))
```

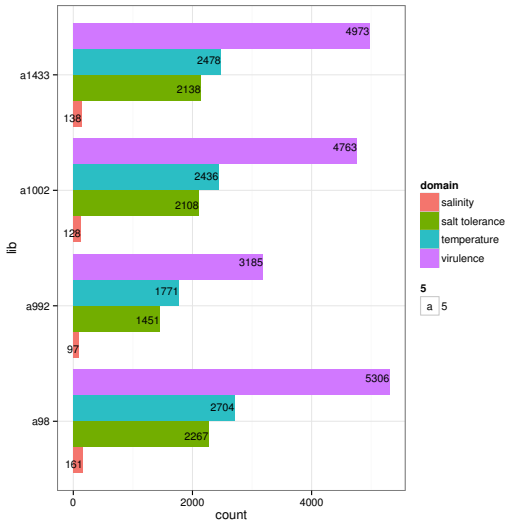


All domains of the above proteins found in the 4 strain libraries.

```
allpfam <- select(pfam, contains("a"))
allpfam <- filter(allpfam, annot == "new")
allpfam
```

```
          domain pfam  a98 a992 a1002 a1433 annot
1      virulence  655 5306 3185  4763  4973   new
2    temperature  251 2704 1771  2436  2478   new
3       salinity   22  161   97   128   138   new
4 salt tolerance   79 2267 1451  2108  2138   new
```

```
allpfam <- gather(allpfam[, -2], "lib", "count", 2:5)
ggplot(allpfam,
       aes(x = lib,
           y = count,
           fill = domain)) +
    theme_bw() +
        coord_flip() +
            geom_bar(stat = "identity",
                     position = "dodge") +
                geom_text(aes(x = lib,
                              y = count,
                              ymax = count,
                              label = count,
                              size = 5,
                              hjust = 1),
                               position = position_dodge(width = 1))
```



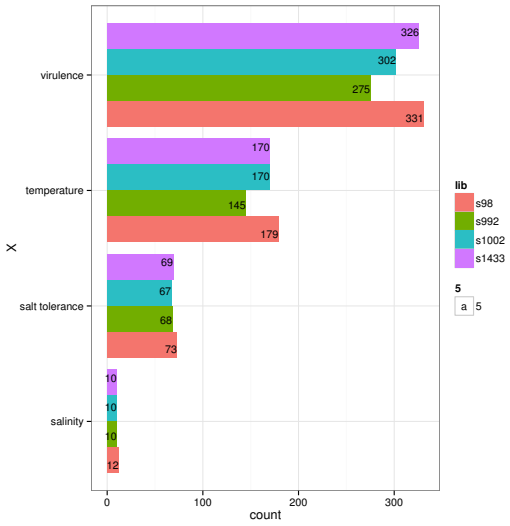
How many possible proteins can be found among the 4 strains.

```
singlepfam <- select(pfam, contains("s"))
```

```
singlepfam <- cbind(singlepfam, X = pfam$domain, Y = pfam$annot)
singlepfam <- filter(singlepfam, Y == "new")
singlepfam
```

```
  s98 s992 s1002 s1433              X   Y
1 331  275   302   326      virulence new
2 179  145   170   170    temperature new
3  12   10    10    10       salinity new
4  73   68    67    69 salt tolerance new
```

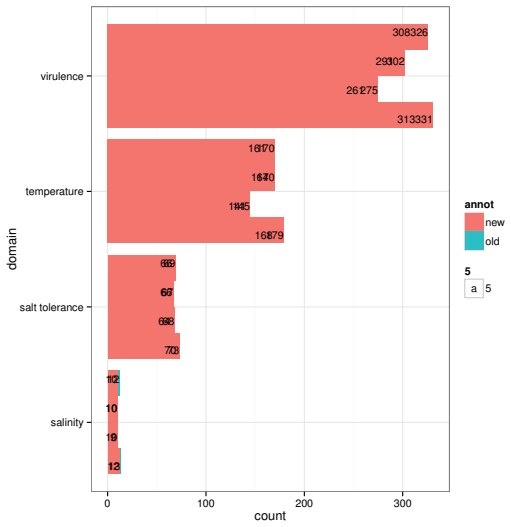
```
singlepfam <- gather(singlepfam, "lib", "count", 1:4)
ggplot(singlepfam,
       aes(x = X,
           y = count,
           fill = lib)) +
    theme_bw() +
        coord_flip() +
            geom_bar(stat = "identity",
                     position = "dodge") +
                geom_text(aes(x = X,
                              y = count,
                              ymax = count,
                              label = count,
                              size = 5,
                              hjust = 1),
                          position = position_dodge(width = 1))
```



Difference between old and new annotations against pfam database.

```
newpfam <- select(pfam, contains("s"))
```

```
newpfam <- cbind(newpfam, annot = pfam$annot, domain = pfam$domain)
newpfam <- gather(newpfam, "lib", "count", 1:4)
ggplot(newpfam,
       aes(x = domain,
           y = count,
           fill = annot,
           group = lib)) +
    theme_bw() +
        coord_flip() +
            geom_bar(stat = "identity",
                     position = "dodge") +
                geom_text(aes(x = domain,
                              y = count,
                              ymax = count,
                              label = count,
                              size = 5,
                              hjust = 1),
                          position = position_dodge(width = 1))
```



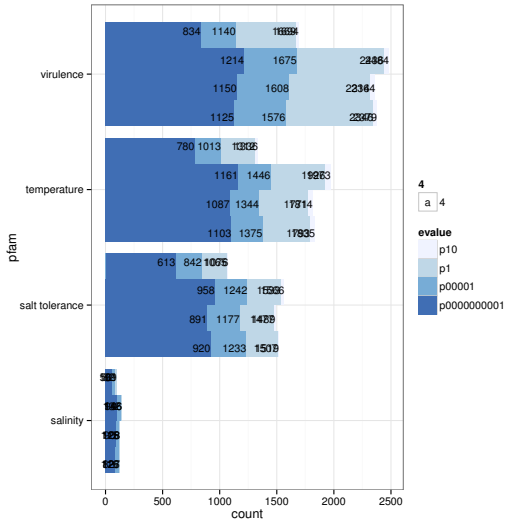
Get the number of contigs that match a significant e-value domain.

```
pfam2 <- read.xlsx("./pfam.xlsx", sheetIndex = 2)
```

```
pfam2
```

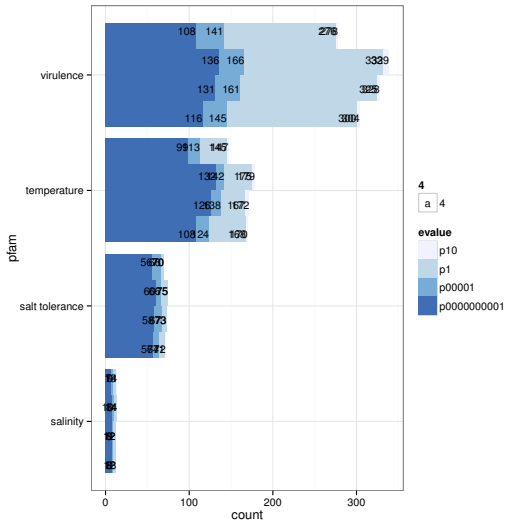
```
     lib  p10   p1 p00001 p0000000001  annot           pfam
1    m98  339  332    166         136 domain      virulence
2   m992  278  276    141         108 domain      virulence
3  m1002  304  300    145         116 domain      virulence
4  m1433  328  325    161         131 domain      virulence
5    m98 2484 2436   1675        1214 contig      virulence
6   m992 1694 1669   1140         834 contig      virulence
7  m1002 2379 2346   1576        1125 contig      virulence
8  m1433 2364 2314   1608        1150 contig      virulence
9    m98  179  175    142         132 domain    temperature
10  m992  147  145    113          99 domain    temperature
11 m1002  170  168    124         108 domain    temperature
12 m1433  172  167    138         126 domain    temperature
13   m98 1973 1926   1446        1161 contig    temperature
14  m992 1336 1312   1013         780 contig    temperature
15 m1002 1835 1793   1375        1103 contig    temperature
16 m1433 1814 1771   1344        1087 contig    temperature
17   m98   14   14     10           8 domain       salinity
18  m992   14   13      9           7 domain       salinity
19 m1002   13   12      9           8 domain       salinity
20 m1433   12   12      9           8 domain       salinity
21   m98  146  146    132          99 contig       salinity
22  m992  101   99     82          56 contig       salinity
23 m1002  127  126    116          82 contig       salinity
24 m1433  128  128    115          90 contig       salinity
25   m98   75   75     66          60 domain salt tolerance
26  m992   70   70     66          56 domain salt tolerance
27 m1002   72   71     64          57 domain salt tolerance
28 m1433   73   73     67          58 domain salt tolerance
29   m98 1566 1533   1242         958 contig salt tolerance
30  m992 1076 1065    842         613 contig salt tolerance
31 m1002 1519 1507   1233         920 contig salt tolerance
32 m1433 1489 1477   1177         891 contig salt tolerance
```

```
pfam2 <- filter(pfam2, annot == "contig")
pfam2 <- gather(pfam2, "evalue", "count", 2:5)
ggplot(pfam2,
       aes(x = pfam,
           y = count,
           fill = evalue,
           group = lib)) +
    theme_bw() +
        geom_bar(stat = "identity",
                 position = "dodge") +
            scale_fill_brewer() +
                coord_flip() +
                    geom_text(aes(x = pfam,
                                  y = count,
                                  ymax = count,
                                  label = count,
                                  size = 4,
                                  hjust = 1),
                              position = position_dodge(width = 1))
```



How many protein domains were found at different evalue significance.

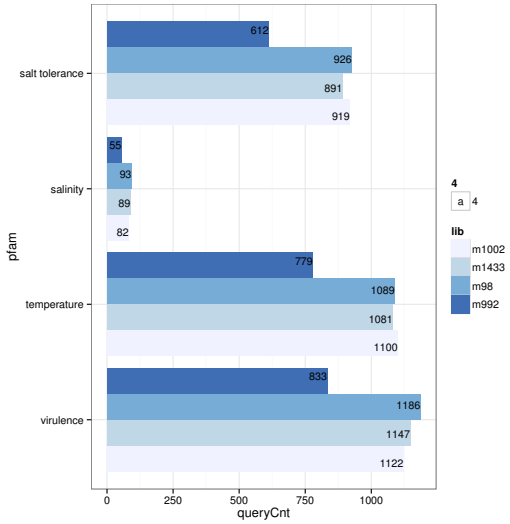
```
pfam2 <- read.xlsx("./pfam.xlsx", sheetIndex = 2)
pfam2 <- filter(pfam2, annot == "domain")
pfam2 <- gather(pfam2, "evalue", "count", 2:5)
ggplot(pfam2,
       aes(x = pfam,
           y = count,
           fill = evalue,
           group = lib)) +
    geom_bar(stat = "identity",
             position = "dodge") +
        scale_fill_brewer() +
            coord_flip() +
                theme_bw() +
                    geom_text(aes(x = pfam,
                                  y = count,
                                  ymax = count,
                                  size = 4,
                                  label = count,
                                  hjust = 1),
                              position = position_dodge(width = 1))
```



13 Map RNA contigs to Genome (v15) contigs

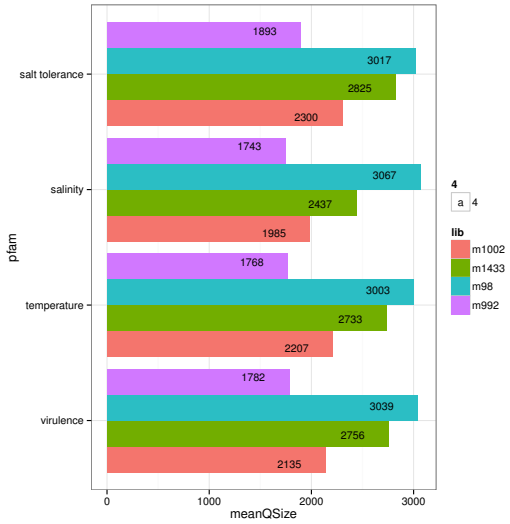
Here is the overall stats of the BLAT of the 4 strains RNA sequenced contigs against SR. genome v15. Until now the RNA contigs have been annotated with pfam, then reverse translated into DNA contigs then aligned on the reference genome for SNP localization.

```
blat <- read.table("./pfam.stats.genomics.txt", header = T)
x <- c("m98", "m992", "m1002", "m1433")
y <- gl(4, 4, 16, labels = c("virulence", "temperature", "salinity", "salt tolerance"))
blat <- data.frame(blat, lib = c(rep(x, 4)), pfam = y)
ggplot(blat,
       aes(x = pfam,
           y = queryCnt,
           fill = lib)) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw() +
            coord_flip() +
                scale_fill_brewer() +
                    geom_text(aes(x = pfam,
                                  y = queryCnt,
                                  ymax = queryCnt,
                                  label = queryCnt,
                                  size = 4,
                                  hjust = 1),
                              position = position_dodge(width = 1))
```



From the table data above the minimum identity of all contigs aligned is 0.9. The mean query is necessary to choose the number of contigs mapped. Since each contig can be found multiple times in the genome (at different alignment lengths of course) it is best if we choose the best contigs those that have a maximum alignment length. For those contigs must be mapped/aligned once and thus, no duplicate entries should be selected for whatever contig. For this reason choosing an alignment length equal to the half of the mean of the alignment length gives the minimum number of duplicate contigs.

```
ggplot(blat,
       aes(x = pfam,
           y = meanQSize,
           fill = lib)) +
    coord_flip() +
        theme_bw() +
            geom_bar(stat = "identity",
                     position = "dodge") +
                geom_text(aes(x = pfam,
                              y = meanQSize,
                              ymax = meanQSize,
                              label = meanQSize,
                              size = 4,
                              hjust = 2),
                          position = position_dodge(width = 1))
```



14 Assessing SNP hotspots in 4 QPX strains

QPX assemblies by MMETSP team were used for pfam annotation (with HMMER). SNP calling on the 4 strains used Steve Roberts reference genome v15 (called with GATK). Location of SNPs in the pfam domains was inferred after alignment of the QPX contigs on the reference genome (with BLAT). Finally all data were merged in one file grouped by 4 QPX strains (2 from NY, one from each VA and MA) and 3 pfam subset pathways (Virulence, salinity/salt-tolerance, temperature).

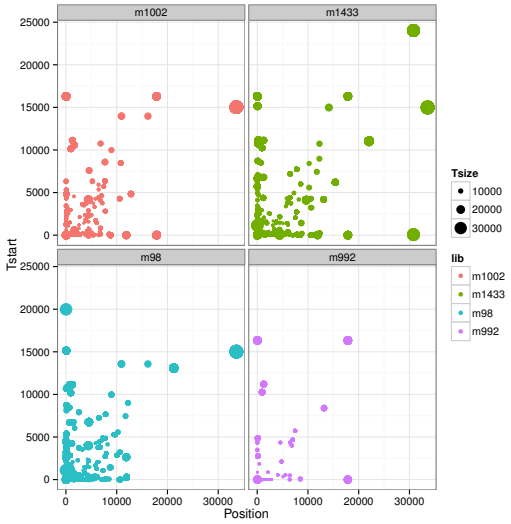
```
hotspots.raw <- read.table("./hotspots/all.pfam.snp.txt", header = TRUE)
```

```
head(hotspots.raw)
```

```
  row Qname                Tname match mismatch repmatch N QgapCount
1   1     5 QPX_v015_contig_5842  2327        0        0 0         0
2   2     5 QPX_v015_contig_5842  2327        0        0 0         0
3   3     5 QPX_v015_contig_5842  2327        0        0 0         0
4   4     5 QPX_v015_contig_5842  2327        0        0 0         0
5   5     5 QPX_v015_contig_5842  2327        0        0 0         0
6   6     5 QPX_v015_contig_5842  2327        0        0 0         0
  QgapBases TgapCount TgapBases Strand Qsize Qstart Qend Tsize Tstart
1         0         1        50      -  2372      0 2327  4816    104
2         0         1        50      -  2372      0 2327  4816    104
3         0         1        50      -  2372      0 2327  4816    104
4         0         1        50      -  2372      0 2327  4816    104
5         0         1        50      -  2372      0 2327  4816    104
6         0         1        50      -  2372      0 2327  4816    104
  Tend BlockCount BlockSize qStarts  tStarts         pfam.x lib.x
1 2481          2 376,1951, 45,421, 104,530, salt.tolerance   m98
2 2481          2 376,1951, 45,421, 104,530, salt.tolerance   m98
3 2481          2 376,1951, 45,421, 104,530, salt.tolerance   m98
4 2481          2 376,1951, 45,421, 104,530, salt.tolerance   m98
5 2481          2 376,1951, 45,421, 104,530,    temperature   m98
6 2481          2 376,1951, 45,421, 104,530,    temperature   m98
  Position REF ALT Quality lib.y Domain  accession tLen qLen  evalue
1       23   C   T    88.8   m98  SH3_1 PF00018.24   48  573 2.5e-25
2       23   C   T    88.8   m98  SH3_1 PF00018.24   48  573 7.9e-26
3       23   C   T    88.8   m98  SH3_1 PF00018.24   48  573 2.5e-25
4       23   C   T    88.8   m98  SH3_1 PF00018.24   48  573 7.9e-26
5       22   G   C    88.8   m98  SH3_1 PF00018.24   48  573 2.5e-25
6       22   G   C    88.8   m98  SH3_1 PF00018.24   48  573 7.9e-26
  score2 cEvalue   iEval score alnFrom alnTo  acc description
1   81.9 2.3e-16 2.8e-14  46.5     352   396 0.96         SH3
2   81.9 1.4e-12 1.1e-10  33.3     262   309 0.86         SH3
3   81.9 2.9e-12 3.6e-10  33.3     262   309 0.86         SH3
4   81.9 1.1e-16 8.9e-15  46.5     352   396 0.96         SH3
5   81.9 2.3e-16 2.8e-14  46.5     352   396 0.96         SH3
6   81.9 1.4e-12 1.1e-10  33.3     262   309 0.86         SH3
         pfam.y lib
1   temperature m98
2 salt.tolerance m98
3   temperature m98
4 salt.tolerance m98
5   temperature m98
6 salt.tolerance m98
```

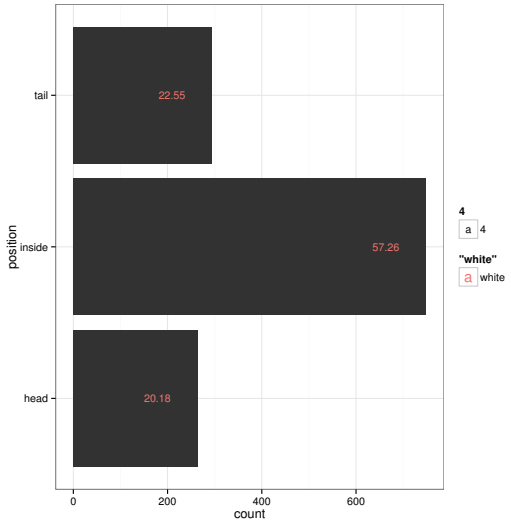
What is the correlation between a SNP position and the first reference nucleotide that align to contig containing domain?

```
ggplot(hotspots.raw,
       aes(x = Position,
           y = Tstart)) +
    theme_bw() +
        geom_point(aes(color = lib,
                       size = Tsize)) +
            facet_wrap( ~ lib, ncol = 2)
```



How many SNPs can be found inside and outside of protein domains?

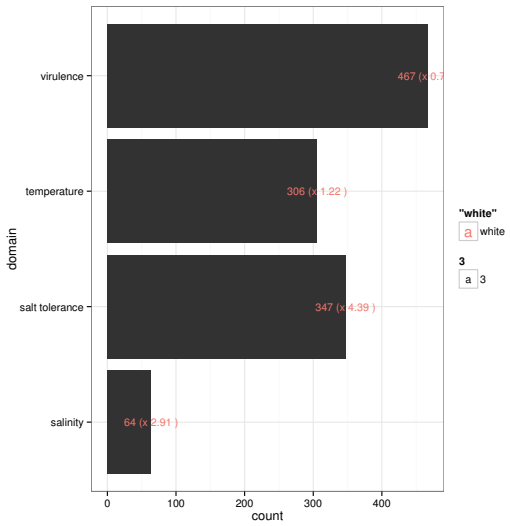
```
count <- c(264, 749, 295)
position <- c("head", "inside", "tail")
dat <- data.frame(position, count)
dat$per <- round((dat$count/sum(dat[, 2]))*100, digits = 2)
ggplot(dat,
       aes(x = position,
           y = count)) +
    theme_bw() +
        geom_bar(stat = "identity") +
            coord_flip() +
                geom_text(aes(x = position,
                              y = count,
                              ymax = count,
                              label = per,
                              size = 4,
                              hjust = 2,
                              color = "white"))
```



How are SNPs distributed between pfam domains? The between parenthesis score is an identifier for the highest SNP concentration. It has no units. It is just a score of normalized counts. The counts is relative to the position of SNPs inside the protein domains.

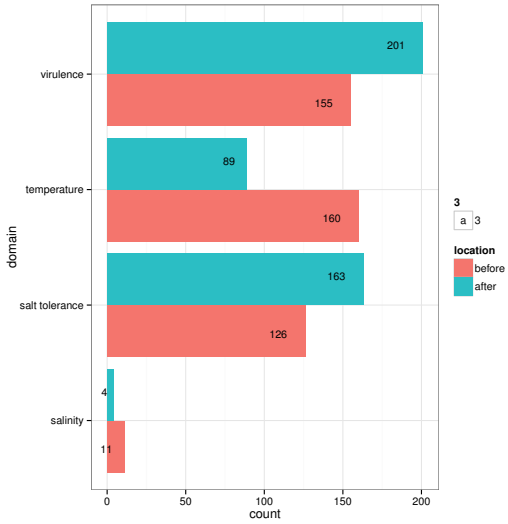
```
domain <- c("virulence", "temperature", "salinity", "salt tolerance")
```

```
count <- c(467, 306, 64, 347)
dat <- data.frame(domain, count)
dat$norm <- round(dat$count/pfam[1:4, 2], digits = 2)
ggplot(dat,
       aes(x = domain,
           y = count)) +
    theme_bw() +
        coord_flip() +
            geom_bar(stat = "identity") +
                geom_text(aes(x = domain,
                              y = count,
                              ymax = count,
                              label = paste(count, "(x", norm, ")"),
                              size = 3,
                              hjust= .5,
                              color = "white"))
```



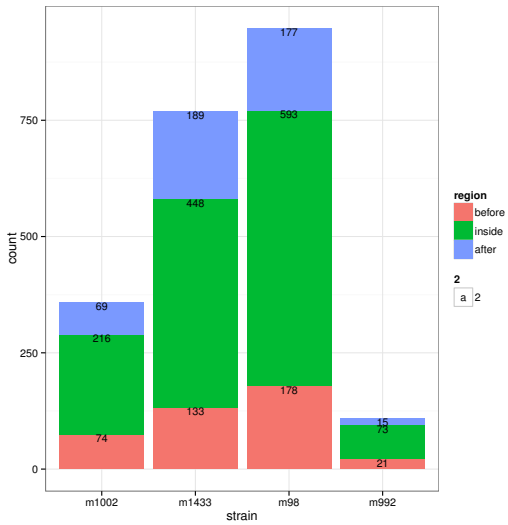
How many SNPs can be found outside of each domain? Separate analysis. The SNP position outside the domains is dependent on the Reference contig length, which was selected through alignment.

```
before <- c(155, 160, 11, 126)
after <- c(201, 89, 4, 163)
dat <- data.frame(domain, before, after)
dat <- gather(dat, "location", "count", 2:3)
ggplot(dat,
       aes(x = domain,
           y = count,
           fill = location)) +
    geom_bar(stat = "identity",
             position = "dodge") +
        theme_bw() +
            coord_flip() +
                geom_text(aes(x = domain,
                              y = count,
                              ymax = count,
                              label = count,
                              size = 3,
                              hjust = 2),
                          position = position_dodge(width = 1))
```



How many SNPs can be found inside and outside protein domains between strains?

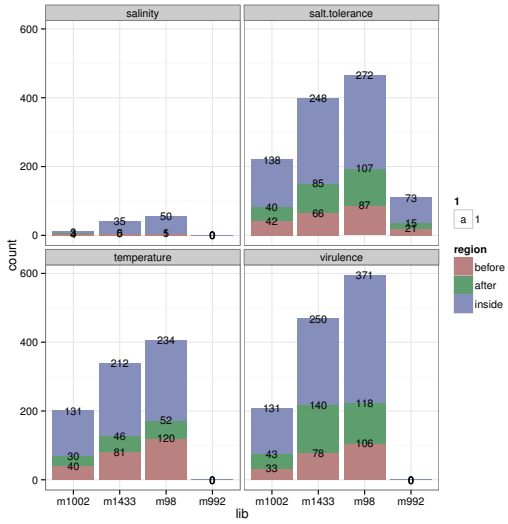
```
before <- c(178, 21, 74, 133)
after <- c(177, 15, 69, 189)
inside <- c(593, 73, 216, 448)
strain <- c("m98", "m992", "m1002", "m1433")
dat <- data.frame(strain, before, inside, after)
dat <- gather(dat, "region", "count", 2:4)
ggplot(dat,
       aes(x = strain,
           y = count,
           fill = region)) +
    geom_bar(stat = "identity") +
        geom_text(aes(x = strain,
                      y = count,
                      ymax = count,
                      label = count,
                      vjust = 1,
                      size = 2),
                  position = "stack") +
            theme_bw()
```



How many SNPs can be found inside and outside domains between virulence, temperature, salinity and within strain?

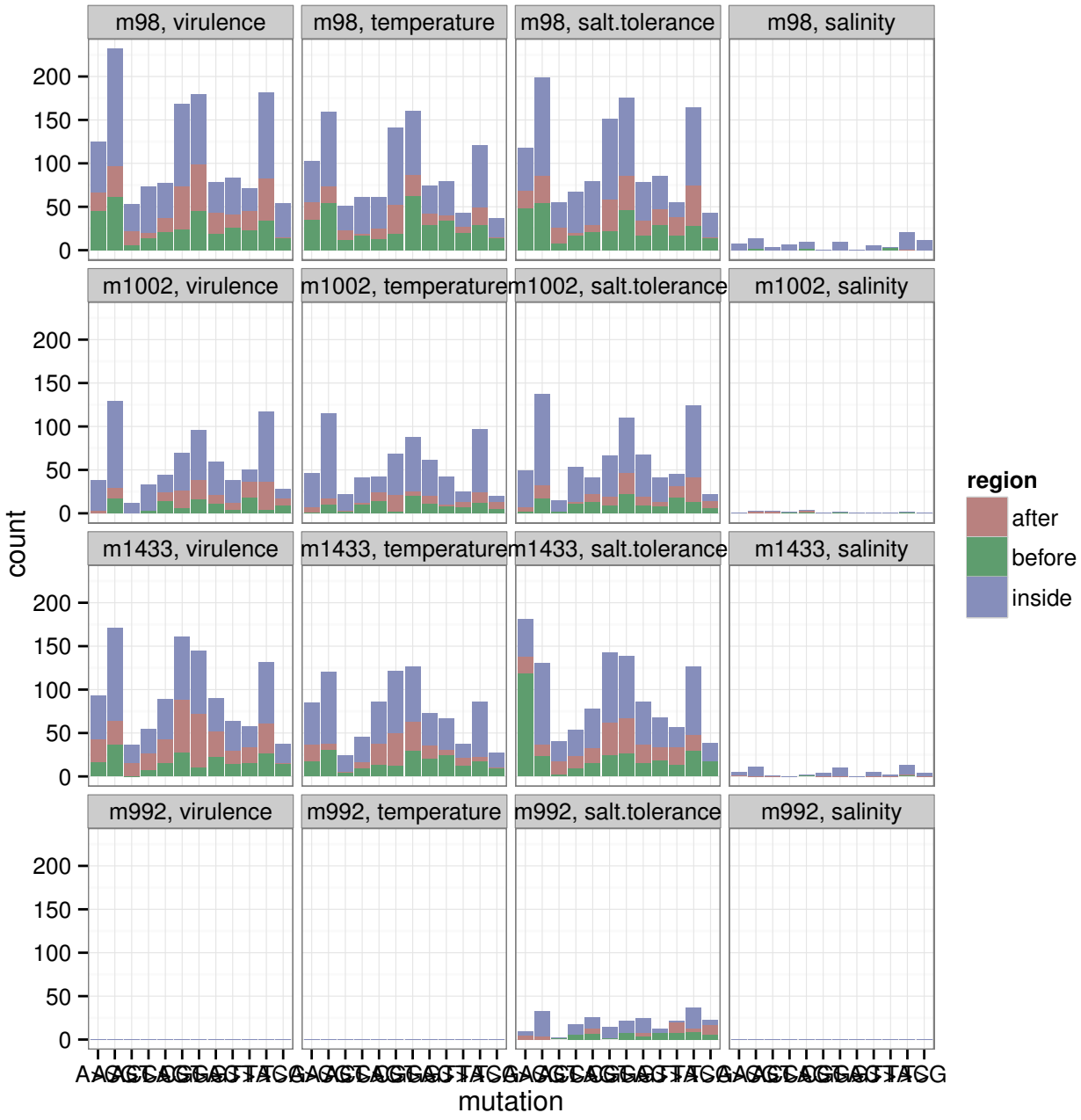
```
dat <- read.xlsx("./hotspots/snps.all.pfam.xlsx", sheetIndex = 1)
```

```
dat <- gather(dat, "region", "count", 3:5)
ggplot(dat,
       aes(x = lib,
           y = count,
           fill = region)) +
    geom_bar(stat = "identity") +
        geom_text(aes(x = lib,
                      y = count,
                      ymax = count,
                      label = count,
                      size = 1,
                      hjust = .5),
                  position = "stack") +
            facet_wrap( ~ pfam, ncol = 2) +
                theme_bw() +
                    scale_fill_hue(c = 40, l = 60)
```



Preferential substitution inside/outside domains per pfam subset for each strain.

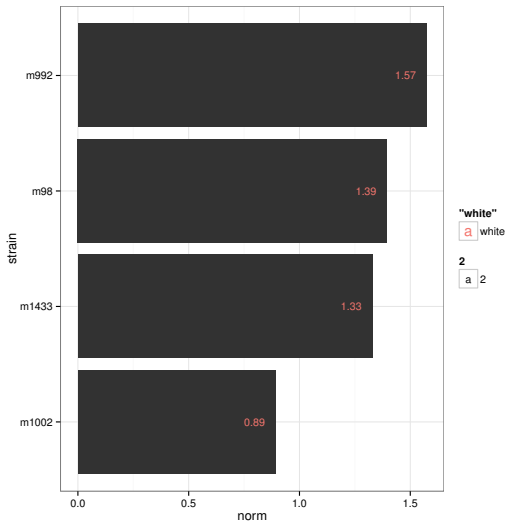
```
dat <- read.xlsx("./hotspots/snps.all.pfam.xlsx", sheetIndex = 2)
dat <- gather(dat, "mutation", "count", 3:14)
dat$mutation <- gsub(".", ">", dat$mutation, fixed = TRUE)
dat$pfam <- factor(dat$pfam, levels = c("virulence",
                                        "temperature",
                                        "salt.tolerance",
                                        "salinity"))
dat$lib <- factor(dat$lib, levels = c("m98",
                                      "m1002", "m1433", "m992"))
ggplot(dat,
       aes(x = mutation,
           y = count,
           fill = region)) +
    geom_bar(stat = "identity") +
            theme_bw() +
                facet_wrap(lib ~ pfam, ncol = 4) +
                    scale_fill_hue(c = 40,  l = 60)
```



How many SNPs can be found inside protein domains between strains, per 1 kb? The number showing has been normalized with the total sum of sizes of contigs containing those SNPs.

```
inside <- c(593, 73, 216, 448)
```

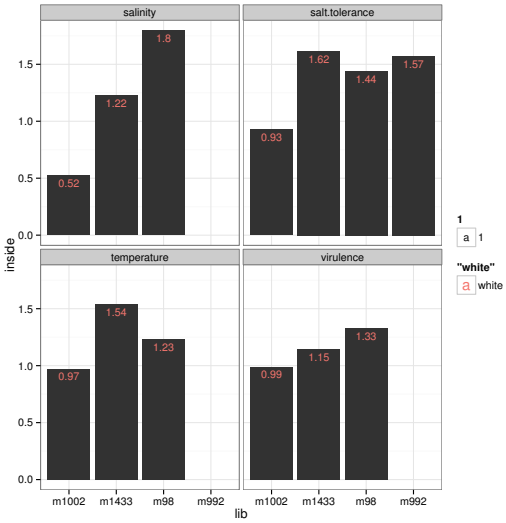
```
sizen <- c(425098, 46409, 242136, 337206)
strain <- c("m98", "m992", "m1002", "m1433")
dat <- data.frame(strain, inside, sizen)
dat$norm <- with(dat, (inside/sizen)*1000)

ggplot(dat,
       aes(x = strain,
           y = norm)) +
    coord_flip() +
    geom_bar(stat = "identity") +
        geom_text(aes(x = strain,
                      y = norm,
                      ymax =norm,
                      label = round(norm, digits = 2),
                      hjust = 1.5,
                      color = "white",
                      size = 2)) +
            theme_bw()
```



Frequency of SNPs inside pfam domains for each strain, per 1 Kb. Normalized by the total size of contigs for each strain.

```
dat <- read.xlsx("./hotspots/snps.all.pfam.xlsx", sheetIndex = 1)
dat$inside <- with(dat, (inside/tsum)*1000)
ggplot(dat,
       aes(x = lib,
           y = inside)) +
    geom_bar(stat = "identity") +
        geom_text(aes(x = lib,
                      y = inside,
                      ymax = inside,
                      label = round(inside, digits = 2),
                      size = 1,
                      color = "white",
                      vjust = 1.5)) +
            facet_wrap( ~ pfam, ncol = 2) +
                theme_bw()
```

```
Warning:  Removed 1 rows containing missing values (position_stack).
Warning:  Removed 1 rows containing missing values (position_stack).
Warning:  Removed 1 rows containing missing values (position_stack).
Warning:  Removed 1 rows containing missing values (geom_text).
Warning:  Removed 1 rows containing missing values (geom_text).
Warning:  Removed 1 rows containing missing values (geom_text).
```



15 Machine learning on 4 QPX strains

Lets try a support vector machine classifier to differentiate between the QPX strains using quality data (above) of the variants.

16 System Information

The version number of R and packages loaded for generating the vignette were:

```
###save(list=ls(pattern=".*|.*"),file="PD.Rdata")
sessionInfo()

R version 3.1.2 (2014-10-31)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8           LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8          LC_NAME=en_US.UTF-8
 [9] LC_ADDRESS=en_US.UTF-8        LC_TELEPHONE=en_US.UTF-8
[11] LC_MEASUREMENT=en_US.UTF-8    LC_IDENTIFICATION=en_US.UTF-8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets
[7] methods   base

other attached packages:
 [1] ggbiplot_0.55       scales_0.2.4        plyr_1.8.1
 [4] tidyr_0.1           vegan_2.2-0         permute_0.8-3
 [7] dplyr_0.3.0.2       ggplot2_1.0.0       latticeExtra_0.6-26
[10] RColorBrewer_1.0-5  lattice_0.20-29     xlsx_0.5.7
[13] xlsxjars_0.6.1      rJava_0.9-6         knitr_1.8

loaded via a namespace (and not attached):
 [1] assertthat_0.1   cluster_1.15.3   colorspace_1.2-4
 [4] compiler_3.1.2   DBI_0.3.1        digest_0.6.4
 [7] evaluate_0.5.5   formatR_1.0      gtable_0.1.2
[10] highr_0.4        labeling_0.3     lazyeval_0.1.9
[13] magrittr_1.5     MASS_7.3-35      Matrix_1.1-4
[16] mgcv_1.8-4       munsell_0.4.2    nlme_3.1-118
[19] parallel_3.1.2   proto_0.3-10     Rcpp_0.11.3
[22] reshape2_1.4     stringr_0.6.2    tcltk_3.1.2
[25] tools_3.1.2
```