Solutions for chapter Annotation and Metadata

Exercise 1

```
> hist(rt$statistic, breaks=100, col="skyblue",
       main="", xlab="t-statistic")
```

```
> hist(rt$p.value, breaks=100, col="mistyrose",
       main="", xlab="p-value")
```

Exercise 2

```
> sel = order(rt$p.value)[1:400]
> ALLsub = ALLfilt_af4bcr[sel,]
```

Exercise 3

First, we map from the Affymetrix identifiers to EntrezGene IDs.

```
> EG    = as.character(hgu95av2ENTREZID[featureNames(ALL)])
> EGsub = as.character(hgu95av2ENTREZID[featureNames(ALLsub)])
```

Then, we find the multiplicity by a frequently used and efficient idiom of the R language. The two calls to the function `table` work as follows. The inner one counts for each EntrezGene ID the number of probe sets that are mapped to it. The outer one tabulates how often each count is seen, one, two, three, ... times.

```
> table(table(EG))
   1    2    3    4    5    6    7    8    9
6891 1495  468   97   25   13    5    5    1
> table(table(EGsub))
  1
400
```

There are 6891 instances of EntrezGene IDs that are matched by exactly one probe set in `ALL`, whereas 1495 EntrezGene IDs are matched by two probe sets. That the probe sets in `ALLsub` all map to a unique EntrezGene ID is no coincidence. This has been achieved by our call to the `nsFilter` function above (type `? nsFilter` to find out more about this).

Exercise 4

```
> syms = as.character(hgu95av2SYMBOL[featureNames(ALLsub)])
> whFeat = names(which(syms =="CD44"))
> ordSamp = order(ALLsub$mol.biol)
> CD44 = ALLsub[whFeat, ordSamp]
> plot(as.vector(exprs(CD44)), main=whFeat,
      col=c("sienna", "tomato")[CD44$mol.biol],
      pch=c(15, 16)[CD44$mol.biol], ylab="expression")
```

Exercise 5

First, we create the *data.frame* `z` that contains the mapping between probe sets and chromosome identifiers; then we use the function `table` to produce the table of frequencies.

```
> z = toTable(hgu95av2CHR[featureNames(ALLsub)])
> chrtab = table(z$chromosome)
> chrtab
```

```
 1 10 11 12 13 14 15 16 17 18 19  2 20 21 22  3  4  5  6  7
43 23 23 20  9 20  5 12 17  6 14 26  9  7 13 18 14 11 39 22
 8  9  X  Y
14 20 15  1
```

To plot the frequencies entries in the numeric order of the chromosomes, we need one extra step constructing `chridx`, as in the code below.

```
> chridx = sub("X", "23", names(chrtab))
> chridx = sub("Y", "24", chridx)
> barplot(chrtab[order(as.integer(chridx))])
```

Exercise 6

First, we compute a list `probeSetsPerGene` that contains, for each EntrezGene ID, the list of probe sets that are mapped to it.

```
> probeSetsPerGene = split(names(EG), EG)
> j = probeSetsPerGene$"7013"
> j
[1] "1329_s_at"  "1342_g_at"  "1361_at"    "32255_i_at"
[5] "32256_r_at" "32257_f_at" "32258_r_at"
```

Then we plot the data for the first and seventh probe set of Entrez Gene ID 7013.

```
> plot(t(exprs(ALL_af4bcr)[j[c(1,7)], ]), asp=1, pch=16,
      col=ifelse(ALL_af4bcr$mol.biol=="ALL1/AF4", "black",
      "grey"))
```

We can also consider a heatmap.

```
> library("lattice")
> mat = exprs(ALL_af4bcr)[j,]
> mat = mat - rowMedians(mat)
> ro = order.dendrogram(as.dendrogram(hclust(dist(mat))))
> co = order.dendrogram(as.dendrogram(hclust(dist(t(mat)))))
> at = seq(-1, 1, length=21) * max(abs(mat))
> lp = levelplot(t(mat[ro, co]),
    aspect = "fill", at = at,
    scales = list(x = list(rot = 90)),
    colorkey = list(space = "left"))
> print(lp)
```

What is the effect of the median centering? What does the heatmap look like if you do not do the centering?

Exercise 7

```
> ps_chr = toTable(hgu95av2CHR)
> ps_eg  = toTable(hgu95av2ENTREZID)
> chr = merge(ps_chr, ps_eg)
> chr = unique(chr[, colnames(chr)!="probe_id"])
> head(chr)
  chromosome gene_id
1         14    5875
2         16    5595
3          1    7075
4         10    1557
5         11     643
7          5    1843
```

We see that in `chr` some EntrezGene IDs are mapped to multiple chromosomes (you might want to investigate which ones):

```
> table(table(chr$gene_id))
   1    2
8985   12
```

Here, for simplicity, we just remove conflicting mappings.

```
> chr = chr[!duplicated(chr$gene_id), ]
```

Exercise 8

```
> isdiff = chr$gene_id %in% EGsub
> tab = table(isdiff, chr$chromosome)
> tab
isdiff    1  10  11  12  13  14  15  16  17  18  19   2  20
  FALSE 898 304 498 474 150 271 256 366 512 122 543 547 221
  TRUE   43  23  23  20   9  20   5  12  17   6  14  26   9

isdiff   21  22   3   4   5   6   7   8   9  Un   X   Y
  FALSE  93 249 461 326 390 490 406 297 311   4 384  24
  TRUE    7  13  18  14  11  39  22  14  20   0  15   0
> fisher.test(tab, simulate.p.value=TRUE)
        Fisher's Exact Test for Count Data with simulated
        p-value (based on 2000 replicates)

data:  tab
p-value = 0.01399
alternative hypothesis: two.sided
> chisq.test(tab)
        Pearson's Chi-squared test

data:  tab
X-squared = 42.2, df = 24, p-value = 0.01213
```

Exercise 9

```
> chrloc = toTable(hgu95av2CHRLOC[featureNames(ALLsub)])
> head(chrloc)
  probe_id start_location Chromosome
1  1635_at      132579088          9
2  1635_at      132700651          9
3 39329_at      -68410592         14
4 40797_at      -56675801         15
5 33800_at       -3952652         16
6 34777_at       10283217         11
```

A little complication arises because some genes, and hence some probe sets, have multiple (alternative) transcription start sites and therefore are annotated at multiple locations.

```
> table(table(chrloc$probe_id))
  1   2   3   4   5   6   9
285  66  33   9   3   3   1
```

We can collapse this table such that for each probe set we only record the strand, which is unique.

```
> strds = with(chrloc,
    unique(cbind(probe_id, sign(start_location))))
> table(strds[,2])
```

```
 -1   1
194 206
```

Exercise 10

We call the summary method, with $p = 0.001$.

```
> sum = summary(mfhyper, p=0.001)
> head(sum)
      GOBPID   Pvalue OddsRatio ExpCount Count Size
1 GO:0007154 3.05e-09      1.91   116.13   168 1089
2 GO:0007165 4.33e-09      1.91   109.41   160 1026
3 GO:0006955 2.84e-07      2.43    27.30    54  256
4 GO:0019882 1.37e-06      6.18     3.84    15   36
5 GO:0006687 1.42e-06       Inf     0.64     6    6
6 GO:0006664 4.44e-06     29.83     0.96     7    9
                                Term
1                  cell communication
2                 signal transduction
3                     immune response
4 antigen processing and presentation
5 glycosphingolipid metabolic process
6        glycolipid metabolic process
```

In total, the table `sum` contains 28 categories. Several relate to the immune system and lymphocyte proliferation. This is not surprising given the role that B-cells play and the fact that the disease studied is a leukemia.

Exercise 11

For each GO identifier, an object of class *GOTerms* can be retrieved from the `GOTERM` annotation object that is supplied in the **GO.db** package. It contains various pieces of information about that category, as shown below.

```
> GOTERM[["GO:0032945"]]
GOID: GO:0032945
Term: negative regulation of mononuclear cell
    proliferation
Ontology: BP
Definition: Any process that stops, prevents or
    reduces the frequency, rate or extent of
    mononuclear cell proliferation.
Synonym: negative regulation of PBMC proliferation
Synonym: negative regulation of peripheral blood
    mononuclear cell proliferation
```

Exercise 12

```
> utr = getSequence(id=EGsub, seqType="3utr",
      mart=ensembl, type="entrezgene")
> utr[1,]
           1
3utr       "AGAATGATCCTGTTCAACCTCCTAG..."
entrezgene "  224"
           2
3utr       "ATCCCATCCTGGAATGGAAGGTGCA..."
entrezgene " 3117"
           3
3utr       "TCCACCCCGCCCGGCCGCCCTCGTC..."
entrezgene "   87"
           4
```

```
3utr       "GGGACCCCTGAGAAGATGCCAGGAC..."
entrezgene "23152"
           5
3utr       "CCCGAGGCCCACGGGGCCCGCGCCT..."
entrezgene " 9744"
```

Exercise 13

```
> domains = getBM(attributes=c("entrezgene", "pfam",
      "prosite", "interpro"), filters="entrezgene",
      value=EGsub, mart=ensembl)
> interpro = split(domains$interpro, domains$entrezgene)
> interpro[1]
```

```
$`25`
  [1] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
  [5] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
  [9] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [13] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [17] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [21] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [25] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [29] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [33] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [37] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [41] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [45] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [49] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [53] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [57] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [61] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [65] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [69] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [73] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [77] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [81] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [85] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
 [89] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
 [93] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
 [97] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
[101] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
[105] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
[109] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
[113] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
[117] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
[121] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
[125] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
[129] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
[133] "IPR001245" "IPR000980" "IPR001720" "IPR001452"
[137] "IPR000719" "IPR017441" "IPR008266" "IPR011009"
[141] "IPR011511" "IPR017442" "IPR015015" "IPR002290"
```

Exercise 14

We can use the same type of query as for finding terms that contain the word chromosome. The % wild card matches zero or more arbitrary characters, hence we are looking for all terms that contain the words *transcription factor* at their beginning, in the middle, or in the end.

```
> query = paste("select term from go_term where term",
      "like '%transcription factor%'")
> tf = dbGetQuery(GO_dbconn(), query)
> nrow(tf)
[1] 43
> head(tf)
                                          term
1   RNA polymerase I transcription factor complex
2            transcription factor TFIIIB complex
3            transcription factor TFIIIC complex
4                  transcription factor activity
5  RNA polymerase I transcription factor activity
6 RNA polymerase II transcription factor activity
```