Chapter 2

Data Display and Descriptive Statistics

In this chapter we will introduce a few essential methods commonly used to first explore a new dataset - a process often called "exploratory analysis". These methods provide quick and easy means to display and visualize data in order to quickly answers questions like:

- How are my data distributed?
- Are there outliers in my data?
- Does the distribution of my data resemble that of a bell-shaped curve?
- Are there differences between gene expression values taken from two groups of patients?
- What are the different nucleotide frequencies contained within the coding DNA sequence of a particular gene?

In addition, we will introduce the most important measures of central tendency (mean, trimmed mean, median) together with the most important measures of spread (standard deviation, variance, inter quartile range, and median absolute deviation) to answer some of the above questions.

2.1 Descriptive statistics

There exist various ways to describe the central tendency and spread of datapoints within a dataset. For example, the central tendency can be described by the *mean*, *trimmed mean*, or the *median*, and the spread by the *variance*, *standard deviation*, *interquartile range*, or *median absolute deviation (mad)*. In this section we will define each of these and illustrate their usage.

2.1.1 Measures of central tendency

The most important descriptive statistics for central tendency are the *mean*, *trimmed mean*, and *median*. The *sample mean* of the data values $x_1, \cdots, x_n$ is defined as:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n).$$

Thus the *sample mean* $\overline{x}$ is simply the average of the n data values. Since it is the sum of all the data values divided by the sample size, a few extreme data values may greatly influence the value of the mean. In other words, the mean is not robust against outliers. Note that while the *sample mean* is referred to as $\overline{x}$, the *population mean* is referred to as the Greek letter μ (mu).

The *median* is defined as the second quartile or the 50th percentile, and is denoted by $x_{0.50}$. When the data are symmetrically distributed around the mean, then the mean and the median are equal. Since extreme data values do not influence the value of the median, it is very robust against outliers. Robustness is important in bioinformatics because biological data are frequently contaminated by extreme or otherwise influential datapoints.

The *trimmed mean* is a compromise between the *mean* and *median*. The *trimmed mean* is calculated as the mean after excluding (dropping) the top n percentile and bottom m percentile of datapoints. If no percentile of datapoints is dropped then the trimmed mean is the same as the mean, and as higher percentiles of datapoints are dropped the trimmed mean approaches the value of the median. Since the trimmed mean drops extreme datapoints, it is a more robust version of the mean.

Example 1: CCND3 gene expression. To compute the mean, trimmed mean, and median of the ALL expression values of gene CCND3 (Cyclin D3) we can use the `mean()` and `median()` functions:

```
> ccnd3 = grep("CCND3",golub.gnames[ ,2], ignore.case = TRUE) #find the index for CCND3
> ccnd3
[1] 1042
> mean(golub[ccnd3, golubFactor=="ALL"])
[1] 1.89
> mean(golub[ccnd3, golubFactor=="ALL"], trim=.10) #drop the top and bottom 10% of
    ↪ datapoints
[1] 1.922224
> median(golub[ccnd3, golubFactor=="ALL"])
[1] 1.93
```

Note that: (1) the value of the trimmed mean lies between the values of the mean and median, and (2) the mean and the median do not differ very much, so we can assume that the distribution of datapoints around the median is quite symmetric.

2.1.2 Measures of spread

The most important measures of spread (or dispersion) are the *standard deviation*, the *interquartile range*, and the *median absolute deviation (mad)*. The *sample standard deviation* is the square root of the *sample variance*, which is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{1}{n-1}\left((x_1 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2\right).$$

Hence, it is the average of the squared differences between the data values and the sample mean. The *sample standard deviation* s is the square root of the sample variance and may be interpreted as a distance of the data values from the *sample mean* $\overline{x}$. The *sample standard deviation* s has many characterics in common with the mean absolute distance from the sample mean. Note that while the *sample standard deviation* is referred to as s, the *population standard deviation* is referred to as the Greek letter σ (sigma). Like the mean, the variance and the standard deviation are not robust against outliers.

The *interquartile range* is defined as the difference between the third and the first quartile, that is $x_{0.75} - x_{0.25}$. It can be computed by the function `IQR()`. For normally distributed data, the value $IQR/1.349$ is a robust estimator of the standard deviation. For other types of distributions, the constant $k = 1.349$ will take on different values.

The *median absolute deviation (MAD)* about the median is defined as the median of the absolute deviations of the data from the median, and is computed as $median(|x_1 - x_{0.50}|, \cdots, |x_n - x_{0.50}|)$ (e.g. Jurečková & Picek,

2006, p. 63). In R, it is computed by the function `mad()`. For normally distributed data, the value $1.4826 \times MAD$ is a robust estimator of the standard deviation. And just like with IQR, for other types of distributions the constant $k = 1.4826$ will take on different values. However, it's important to note that the default bahavior of the R implementation of the `mad()` function automatically includes the multiplicative scaler $k = 1.4826$ in its calculation.

When a dataset becomes more like a bell-shaped (normal) distribution, both the $IQR/1.349$ and the $1.4826 \times MAD$ calculations will become more equal to the standard deviation (see Section 3.2.2). Lastly, the median absolute deviation (MAD) and interquartile range (IQR) are more robust against outliers compared to the standard deviation because the MAD and IQR are based on quantiles.

Example 1: CCND3 gene expression. These measures of spread for the ALL expression values of gene CCND3 (Cyclin D3) can be computed as follows:

```
> sd(golub[ccnd3, golubFactor=="ALL"])
[1] 0.491
> IQR(golub[ccnd3, golubFactor=="ALL"]) / 1.349
[1] 0.284
> mad(golub[ccnd3, golubFactor=="ALL"])
[1] 0.368
```

Due to the three outliers in the data (cf. Figure 2.5), the standard deviation is larger than the interquartile range and the mean absolute deviation. That is, the absolute differences with respect to the median are somewhat smaller than the root of the squared differences.

2.1.3 Measures of correlation

It is often useful to know how well two datasets are correlated with each other. The two most common measures of correlation are the Pearson's and Spearman's correlations.

The Pearson's correlation coefficient ρ (or r) measures the linear correlation (dependence) between two variables X and Y. ρ can take any value between +1 and -1 inclusive with the following characteristics:

- 1 = total positive correlation
- 0 = no correlation

- -1 = total negative correlation.

When applied to a population, the Pearson's correlation coefficient is commonly represented by the Greek letter ρ (rho) and is referred to as the *population correlation coefficient* or the *population Pearson's correlation coefficient*. However, when applied to a sample the Pearson's correlation coefficient is commonly represented by the letter r and is referred to as the *sample correlation coefficient* or the *sample Pearson's correlation coefficient*. The formula for ρ is:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

where *cov* is the covariance, σ_X is the standard deviation of X , $\overline{Y}$ is the mean of X , and E is the expected value. We obtain a formula for r by substituting estimates for the population covariances and variances based on a sample:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{(\sum_{i=1}^{n}(X_i - \overline{X})^2)}\sqrt{(\sum_{i=1}^{n}(Y_i - \overline{Y})^2)}}$$

It's important to note that the Pearson's correlation coefficient cannot capture the slope of a linear relationship, nor many aspects of a nonlinear relationship.

Now let's investigate the Pearson's correlation between the gene expression patterns of 3 patients while considering just the Top 50 most differentially expressed genes between the ALL and AML patients. We will use R code from the previous chapter and call these Top 50 genes `biomarkers50`:

```
> # Order rows by decreasing difference between the mean expression between ALL and AML
    ↪ patients
> golubFactor <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> meanALL <- apply(golub[,golubFactor=="ALL"], 1, mean)
> meanAML <- apply(golub[,golubFactor=="AML"], 1, mean)
> o <- order(abs(meanALL-meanAML), decreasing=TRUE)
> biomarkers50 = golub[o[1:50],]
> dim(biomarkers50)
[1] 50 38
> cor(biomarkers50[,33], biomarkers50[,37]) # compare 2 AML patients, method = Pearson's
[1] 0.8496906
> cor(biomarkers50[,1], biomarkers50[,33])  # campare an ALL to an AML patient, method =
    ↪ Pearson's
[1] -0.6094429
> cor(biomarkers50[,1], biomarkers50[,37])  # campare an ALL to another AML patient, method
    ↪  = Pearson's
[1] -0.5363629
```

From the three Pearson's correlations above, we can see that the Top 50 most differentially expressed genes for patient 33 and patient 37 (both of whom have AML) are highly positively correlated with an $r = .85$. In addition, we see that for the same 50 genes, the expression profiles of patient 1 (has ALL) and patient 33 (has AML) are very anti-correlated with an $r = -.61$. Lastly, since we already know that patients 33 and 37 are strongly positively correlated, it follows that patients 1 and 37 will also be very anti-correlated with an $r = -.54$. This example shows that correlation tends to be very transitive.

The Spearman's Correlation Coefficient (also called ρ or r) is defined as the Pearson's correlation coefficient between the ranked values of two datasets, and is a nonparametric measure of statistical dependence between two variables using the relative rankings of the values. In general, Spearman's Correlation Coefficient assesses how well the relationship between two variables can be described using a monotonic function. In particular, if X and Y are monotonically related then all datapoints with greater x-values than that of a given data-point will have greater y-values as well.

If there are no repeated data values, a perfect Spearman's correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other. Identical values (rank ties or value duplicates) are assigned a rank equal to the average of their positions in the ascending order of the values.

Example: If in an ordered ranking entries 4 and 5 have the same value, then both are given rankings of 4.5.

An important characteristic of the Spearman's Correlation Coefficient is that it is more resistant to outliers compared to Pearson's Correlation Coefficient.

Now let's investigate the Spearman's correlation between the gene expression patterns of the same 3 patients as before:

```
> cor(biomarkers50[,33], biomarkers50[,37], method="spearman")
[1] 0.7661254
> cor(biomarkers50[,1], biomarkers50[,33], method="spearman")
[1] -0.4882262
> cor(biomarkers50[,1], biomarkers50[,37], method="spearman")
[1] -0.3347384
```

We can see that the Spearman's correlations between the 3 patients give similar results as the Pearson's correlations. The fact that the magnitude of the Pearson's are slightly greater than the magnitude of the Spearman's tells us that the 3 gene expression profiles are very linearly related. If the magnitude of the Spearman's had been greater, then that would have told

us that there are either considerable outliers or non-linear correlations in the data for those 50 genes and 3 patients.

2.2 Data display

Through its many packages, R provides many visualization methods to observe trends and outliers within a dataset. In practice, these basic visualization methods are commonly used during the first pass in understanding the distribution and statistical characteristics of the data.

2.2.1 Frequency tables and pie charts

Discrete data occur when the values naturally fall into different categories. A counts or frequency table can be used to display the number of occurrences or observed frequencies for each category.

Example 1: Zyxin nucleotide content. A gene consists of a long sequence of DNA nucleotides from the set $\{A, C, G, T\}$. The number of occurrences of each nucleotide found in the gene can be displayed in a frequency table. This will be illustrated with the Zyxin gene - which plays an important role in cell adhesion (Golub et al., 1999). The accession number X94991.1 for a splice variant of Zyxin can be used to query the DNA sequence from an online database like NCBI (UniGene). The code below illustrates how to access the coding DNA sequence for the X94991.1 human isoform of the Zyxin gene from the GenBank database. Then we construct a pie chart from a frequency table of the nucleotides found in the coding sequence of the Zyxin gene:

```
> install.packages(c("ape"),repo="http://cran.r-project.org",dep=TRUE)
> library(ape)
> zyxin = read.GenBank(c("X94991.1"),as.character=TRUE)
> zyxinTable = table(zyxin)
> zyxinTable
zyxin
  a   c   g   t
410 789 573 394
> percentLabels = round(100*zyxinTable/sum(zyxinTable), 1) # 1 decimal place
> pieLabels = paste(percentLabels, "%", c(" A"," C"," G"," T"), sep="")
> pie(zyxinTable,           # frequency table
+     cex=1.5,              # make labels big
+     col=rainbow(4),       # 4 very distant colors from the color spectrum
+     labels=pieLabels)     # nucleotide labels
```

From the resulting frequencies in Table 2.1 we can see that the 4 nucleotides are not equally likely. A nice way to visualize a frequency table is by plotting a pie chart as seen in Figure 2.1.

Frequency table

	A	C	G	T
Count	410	789	573	394

Table 2.1: A nucleotide frequency table for the coding sequence of the Zyxin gene.

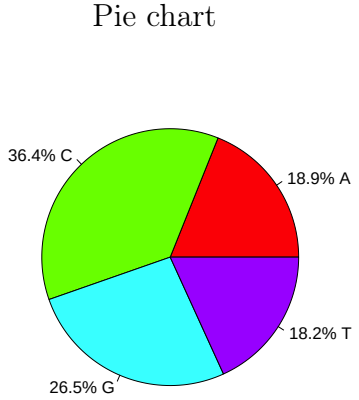

Figure 2.1: A nucleotide frequency pie chart for the coding sequence of the Zyxin gene.

2.2.2 Scatter plots and strip charts

Two basic methods to visualize data are the so-called scatter plot and strip chart. Scatter plots come in 2 flavors: 2-dimensional (2D) and 3-dimensional (3D). In a 2D scatter plot, the values of 2 variables are plotted against each other in a 2-dimensional cartesian plane. While in a 3D scatterplot, the values of 3 variables are plotted against each other in a 3-dimensional cube. In both 2D and 3D scatter plots, the data points are drawn as small symbols and most commonly as dots - which is why scatter plots are also sometimes referred to as "dot plots".

Strip plots are a series of 1-dimensional scatter plots with one strip plot for each value of a categorical (qualitative) variable. In R, often the categorical variable is a factor that distinguishes members from different experimental

conditions or patient/sample groups.

Example 1: CCND3 gene expression. We will be using the Golub et al. (1999) data to illustrate many of the most commonly used plots for visualizing data. For our first example we will concentrate on the expression values for the gene `CCND3 (Cyclin D3)`, which are collected in row 1042 of the data matrix `golub`. To plot the data values we can simply execute:

```
> data(golub, package = "multtest")
> plot(golub[ccnd3,], # values
+      pch=19,       # plot solid circles
+      cex.lab=1.5,  # make axis labels big
+      xlab="Patient number",
+      ylab="CCND3 (Cyclin D3) expression",
+      col="blue")
```

In the resulting scatter plot in Figure 2.2, the vertical axis gives the expression values and the horizontal axis the index of the patient. It can be observed that the CCND3 (Cyclin D3) expression values for AML patients (28 to 38) are somewhat lower, but, indeed, the picture is not very clear because the groups are not plotted separately.

To produce two adjacent strip charts, one for the ALL patients and one for the AML patients, we use the factor called `golubFactor` from the previous chapter and partition the dataset based on that factor:

```
> golubFactor <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> stripchart(golub[ccnd3,] ~ golubFactor,  # values
+            method="jitter",  # add random horizontal jitter
+            cex.lab=1.5,      # make axis labels big
+            vertical = TRUE,  # boxplots vertical
+            col=c("red", "darkgreen"),
+            xlab="Leukemia subtype",
+            ylab=NULL)
```

From the resulting Figure 2.3 we see that the CCND3 (Cyclin D3) expression values for all the ALL patients tend to have larger expression values than those for the AML patients:

2.2.3 Histogram

Another method to visualize data is attained by dividing the range of data values into a number of intervals and to plot the frequency per interval as a vertical bar. Such a plot is called a histogram.

Scatter Plot

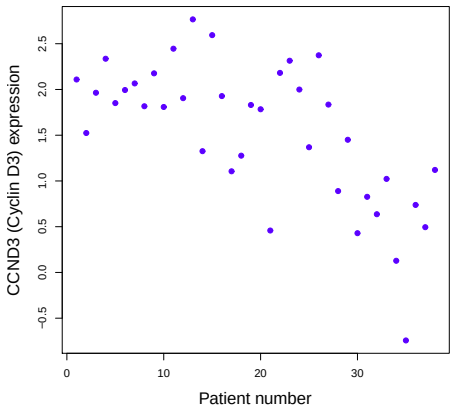

Figure 2.2: Scatter plot of gene expression values of CCND3 (Cyclin D3).

Strip Chart

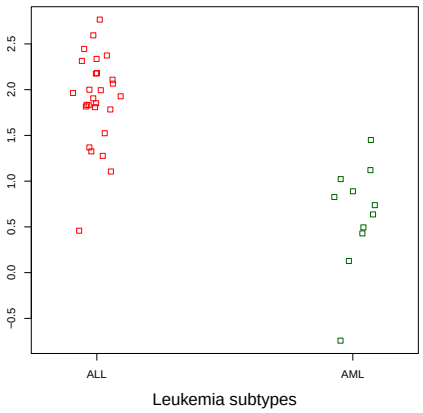

Figure 2.3: Strip chart of gene expression values of CCND3 (Cyclin D3) for ALL and AML patients.

Example 1: CCND3 gene expression. A histogram of the expression values of the gene `CCND3 (Cyclin D3)` for the acute lymphoblastic leukemia patients can be produced as follows:

```
> hist(golub[ccnd3, golubFactor=="ALL"],  # values
+      cex.lab=1.5,       # make axis labels big
+      main=NULL,         # no title
+      xlab="CCND3 (Cyclin D3) Expression",
+      col="cyan")
```

The function `hist()` divides the data into 5 intervals having width equal to 0.5, see Figure 2.4. From this histogram we can see that the datapoints are more or less symmetrically distributed around the mean. By default, the `hist()` function will try to compute the optimal number of bars or breaks from the data. If the data are distributed according to a bell-shaped curve, then this is often a good strategy. Otherwise, the number of bars can be specified by using the `breaks` optional argument to the `hist()` function. Optimal choices for the number of breaks (a.k.a. bars or bins) depending on the data at-hand are discussed in Venables and Ripley (2002).

2.2.4 Box plot

It is always possible to sort n data values to have increasing order $x_1 \leq x_2 \leq \cdots \leq x_n$, where x_1 is the smallest, x_2 is the next smallest, etc. Let $x_{0.25}$ be a number for which 25% of the data values $x_1, \cdots, x_n$ are smaller. That is, 25% of the data values lay on the left side of the number $x_{0.25}$ - which is why it is called the first *quartile* or the 25th *percentile*. The second quartile is the value $x_{0.50}$ where 50% of the data values are smaller. Similarly, the third quartile or 75th percentile is the value $x_{0.75}$ where 75% of the data is smaller. A popular method to display data is by drawing a box around the first and the third quartiles, a bold horizontal line segment through the median, and the smaller line segments (whiskers) for the smallest and the largest data values outside the box. Such a data display is known as a box-and-whisker plot, or simply a box plot.

Example 1: CCND3 gene expression.

A box plot view of the distribution of the expression values for the CCND3 (Cyclin D3) gene for either the ALL or the AML patients can be obtained by constructing two separate box plots adjacent to one another. To produce this box plot we partition the dataset by the factor `golubFactor` while calling the `boxplot()` function:

```
> boxplot(golub[ccnd3,] ~  golubFactor,  # values
+         cex.lab=1.5,      # make axis labels big
+         main=NULL,        # no title
+         xlab="Leukemia subtype",
+         ylab="CCND3 (Cyclin D3) Expression",
+         col=c("purple","green"))
```

From the position of the boxes in Figure 2.5 we observe that the CCND3 (Cyclin D3) gene expression values for the ALL samples are larger than those for AML. Furthermore, since the two sub-boxes around the medians are more or less equally vertically long, the data are quite symmetrically distributed around their respective medians.

To compute exact values for the quartiles, we need a sequence running from 0.00 to 1.00 with steps equal to 0.25. To construct such a sequence we can use the `seq()` function:

```
> pvec <- seq(0,1,0.25)
> quantile(golub[ccnd3, golubFactor=="ALL"],pvec)
   0%   25%   50%   75%  100%
0.458 1.796 1.928 2.179 2.766
```

Histogram

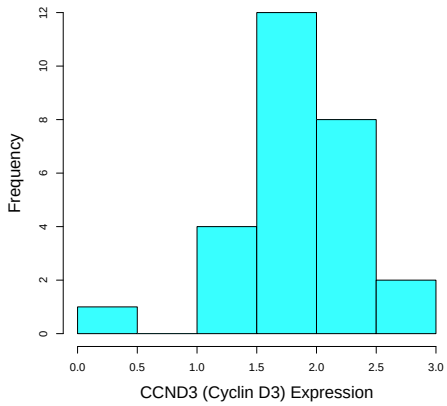

Figure 2.4: Histogram of ALL expression values of gene CCND3 (Cyclin D3).

Box plots

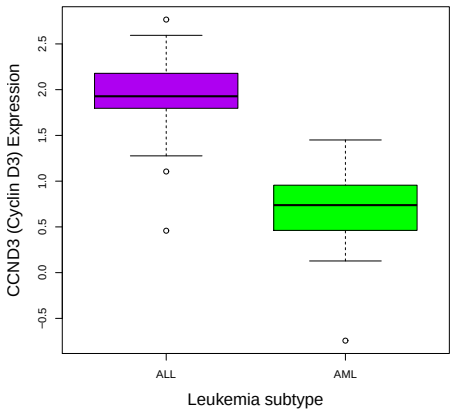

Figure 2.5: Box plot of ALL and AML expression values of gene CCND3 (Cyclin D3).

The first quartile $x_{0.25} = 1.796$, the second quartile $x_{0.50} = 1.928$, and the third third $x_{0.75} = 2.179$. The smallest observed expression value is $x_{0.00} = 0.458$ and the largest is $x_{1.00} = 2.77$. The latter values can be obtained by using the `min()`, `max()`, and `range()` functions:

```
> min(golub[ccnd3, golubFactor=="ALL"])
[1] 0.45827
> max(golub[ccnd3, golubFactor=="ALL"])
[1] 2.7661
> range(golub[ccnd3, golubFactor=="ALL"])
[1] 0.45827 2.76610
```

Outliers are data values laying far away from the pattern set by the majority of the data values. The implementation in R of the (modified) `boxplot()` function draws such *outlier* points separately as small circles. Specifically, a data point x is defined as an outlier point if

$$x < x_{0.25} - 1.5 \cdot (x_{0.75} - x_{0.25}) \quad \text{or} \quad x > x_{0.75} + 1.5 \cdot (x_{0.75} - x_{0.25}).$$

For example, in Figure 2.5 we see that there are outliers among the gene expression values for both ALL and AML patients. In the ALL gene expression values these outliers are the smallest values 0.45827 and 1.10546, and the largest value 2.76610. The AML gene expression values have one outlier with a value of -0.74333.

To define *extreme outliers*, the factor 1.5 is raised to 3.0. Note that this is a descriptive way of defining outliers instead of actually statistically testing for the existence of an outlier - which will be covered in a later chapter.

2.2.5 Heatmaps

When analyzing gene expression data (and other kinds of data) heatmaps are often used to simultaneously graphically display the differences in expression for a wide assortment of genes. We can use the `heatmap.2()` function to view the gene expression patterns in the `golub` dataset.

Let's generate a heatmap using just the Top 50 most differentially expressed genes between the ALL and AML patients. We will use R code from the previous chapter and call these Top 50 genes `biomarkers50`:

```
> library("gplots")
> library("RColorBrewer")
> heatmap.2(biomarkers50,      # values
+           Rowv = NA,         # no row clustering
+           Colv = NA,         # no column clustering
+           scale = "row",     # row-wise scaling
+           col=greenred(75),  # green to red heatmap
+           dendrogram="none", # no dendrogram
+           key=TRUE,          # add color key
+           symkey=FALSE,
+           density.info="none",
+           trace="none",
+           cexRow=0.5)       # make row labels small
```

Our heatmap in Figure 2.6 displays the color-coded levels of expression for the Top 50 most differentially expressed genes where each row represents a gene and each column a patient. Highly expressed genes are colored in red and lowly expressed genes in green. The expression levels corresponding to colors in the spectrum between bright red and bright green are given in the legend. In this heatmap, the gene expression values are row-wise normalized and the order of the patients has not been altered. In this figure, we can already see that there are significant differences in the expression patterns for these 50 genes between the ALL patients (1-27) and the AML patients (28-38).

We will now enhance our "Top 50 Genes" heatmap by performing heirarchical clustering on both the rows and columns. Heirarchical clustering of the rows is achieved by iteratively finding the most similarly expressed genes or clusters of genes and then placing them in the same group (cluster). This process is repeated until only one large cluster finally remains. Likewise,

column-wise clustering is achieved by iteratively finding the patients or clusters of patients with the most similar gene expression profiles. There are many different similarity metrics that can be used to perform the clustering, but they are all trying to achieve the same thing - cluster based on similarity which is akin to positive correlation. The most positively correlated genes will cluster together first:

```
> heatmap.2(biomarkers50,      # values
+           scale = "row",     # row-wise scaling
+           col=colorRampPalette(c("blue", "yellow"))(20),  # blue to yellow heatmap
+           dendrogram="both", # add row and column dendrogram
+           key=TRUE,
+           symkey=FALSE,
+           density.info="none",
+           trace="none",
+           cexRow=0.5)       # make row labels small
```

The heatmap in Figure 2.7 displays the color-coded levels of expression for the Top 50 most differentially expressed genes just like Figure 2.6. However, now the order of the patients and genes have been altered to reflect their respective clustering based on gene expression similarity (positive correlation). The graphical rendering of the heirarchical clustering on each axis if referred to as a "dendrogram". In this figure, we can now see that the clustering of the expression profiles has generated 3 distinct sets (clusters) of patients and 2 distinct sets of genes. One set of patients contains all the ALL patients, while two other sets contain mostly AML patients. Likewise, one set of genes is generally expressed higher in the AML samples, while the other set is generally expressed higher in the ALL samples. Notice that the clustering is certainly not perfect. The AML patients (28-38) are not completely clustered together and 3 ALL patients (12, 25, and 2) are misclassified with the AML patients.

In Figures 2.6 and 2.7, we used the difference between the *mean* of the AML and ALL expression values to find the Top 50 genes with the greatest differential expression. However, we learned in section 2.1.2 that the *mean* is not robust to outliers in the data. We might get better clustering of the patients if we use the *median* instead to define a *robust* set of the 50 most differentially expressed genes between the ALL and AML patients. We will call this new list of Top 50 genes `robustBiomarkers50`:

```
> medianALL <- apply(golub[,golubFactor=="ALL"], 1, median)
> medianAML <- apply(golub[,golubFactor=="AML"], 1, median)
> o <- order(abs(medianALL - medianAML), decreasing=TRUE)
> robustBiomarkers50 = golub[o[1:50],] # Create a set of biomarkers out of the Top 50
>
> # Row scaling and row and column dendrograms (clustering)
```

```
> heatmap.2(robustBiomarkers50,      # values
+           scale = "row",     # row-wise scaling
+           col=bluered(75),   # blue to red heatmap
+           dendrogram="both", # add row and column dendrogram
+           key=TRUE,
+           symkey=FALSE,
+           density.info="none",
+           trace="none",
+           cexRow=0.5)       # make row labels small
```

In the resultant Figure 2.8, we see that the clustering of the patients is indeed better when using the *median* as opposed to the *mean* to define our set of the Top 50 Biomarkers of the most differentially expressed genes that differentiate ALL from AML patients. In Figure 2.8, the AML patients (28-38) on the right-hand side are completely clustered together and only one ALL patient (patient number 2) is misclassified with the AML patients.

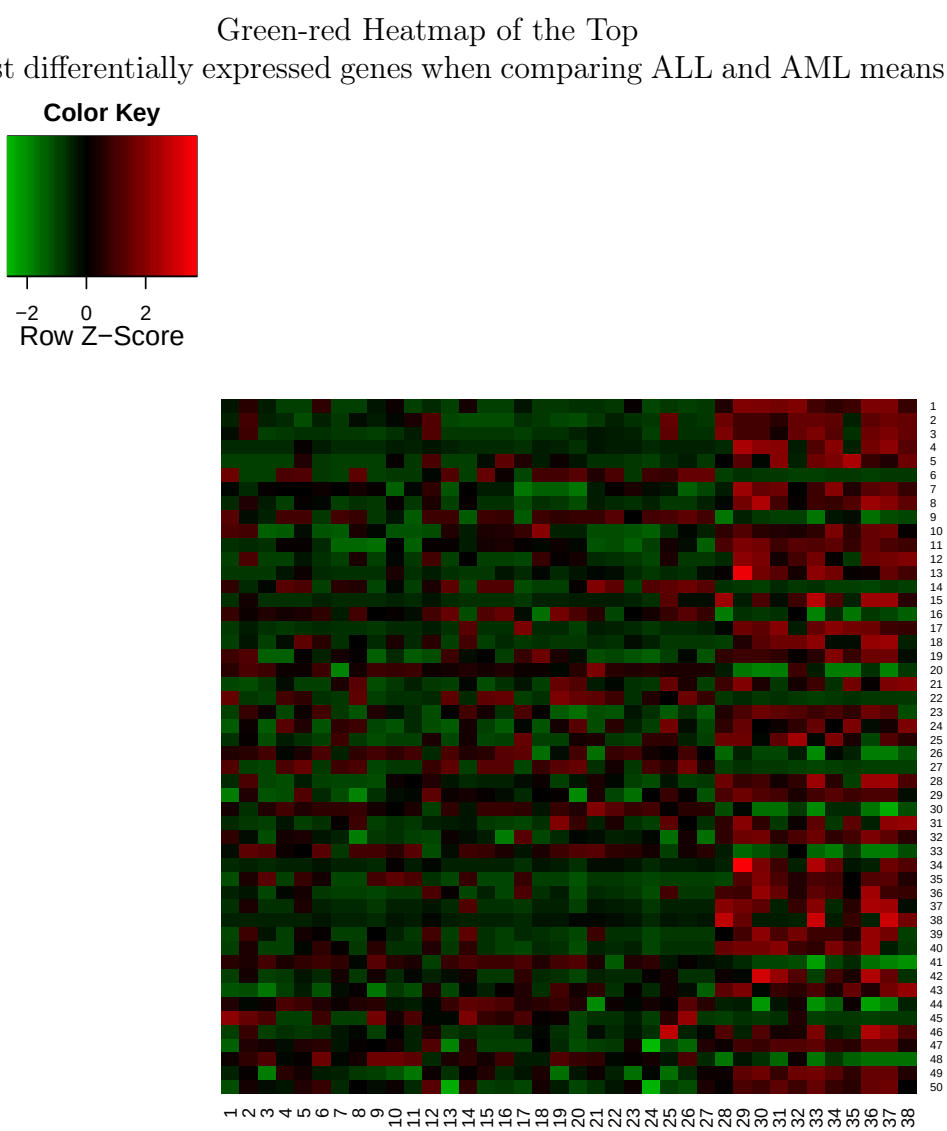

Figure 2.6: Green-red heatmap of the gene expression values for the Top 50 most differentially expressed genes between the ALL and AML patients using their mean expression values. Each row represents a gene and each column a patient. The gene expression values are row-wise normalized and the orders of the genes and patients are preserved relative to the `golub` matrix. There are significant differences in the expression patterns for these 50 genes between the ALL patients (rows 1-27) and the AML patients (rows 28-38).

Blue-yellow Heatmap with dendrograms of the Top
50 most differentially expressed genes when comparing ALL and AML means

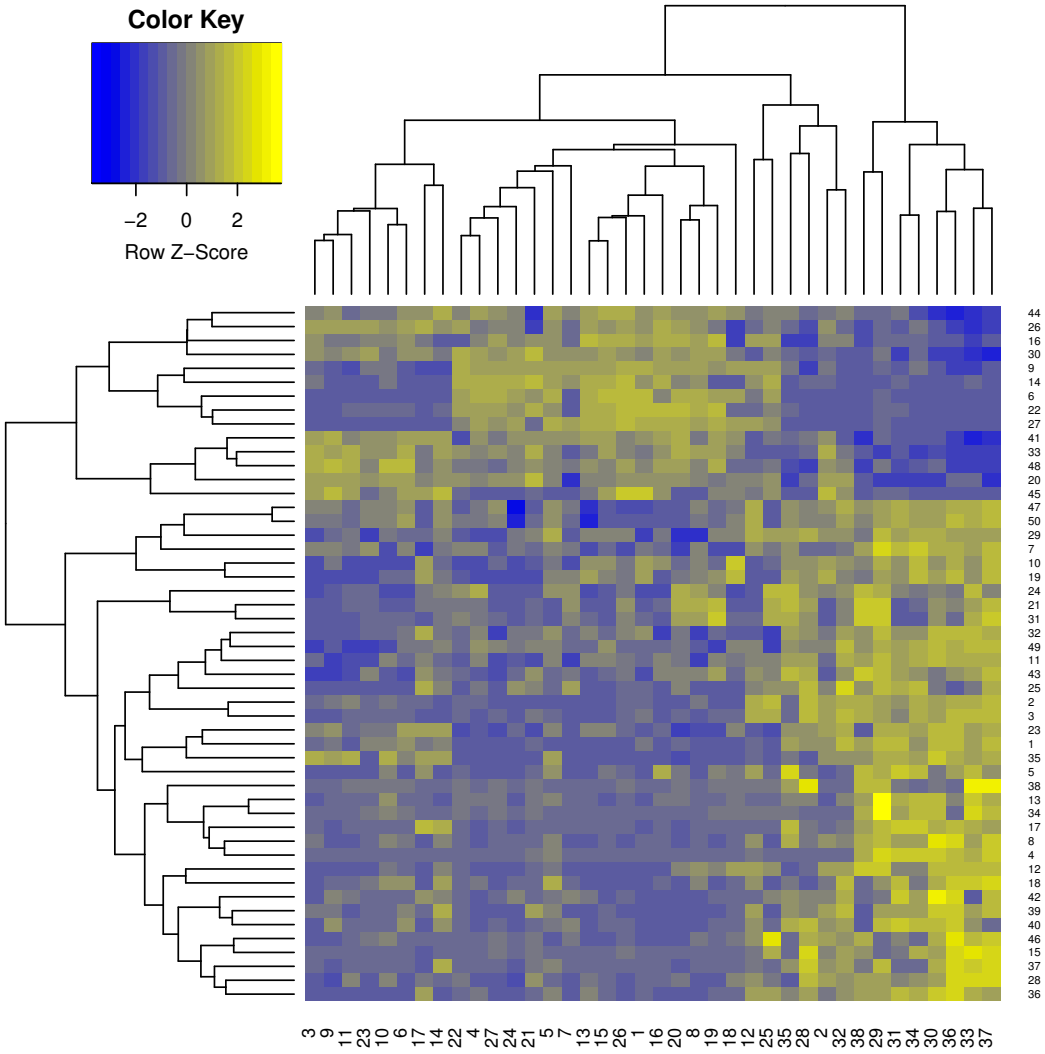

Figure 2.7: Blue-yellow heatmap of the gene expression values for the Top 50 most differentially expressed genes between the ALL and AML patients (using their mean expression values) - where both the patients and genes are clustered based on the similarity of their expression profiles. Each row represents a gene and each column a patient. The AML patients (28-38) on the right-hand side are not completely clustered together and 3 ALL patients (12, 25, and 2) are misclassified with the AML patients.

Blue-red Heatmap with dendrograms of the Top 50 most differentially expressed genes when comparing ALL and AML medians

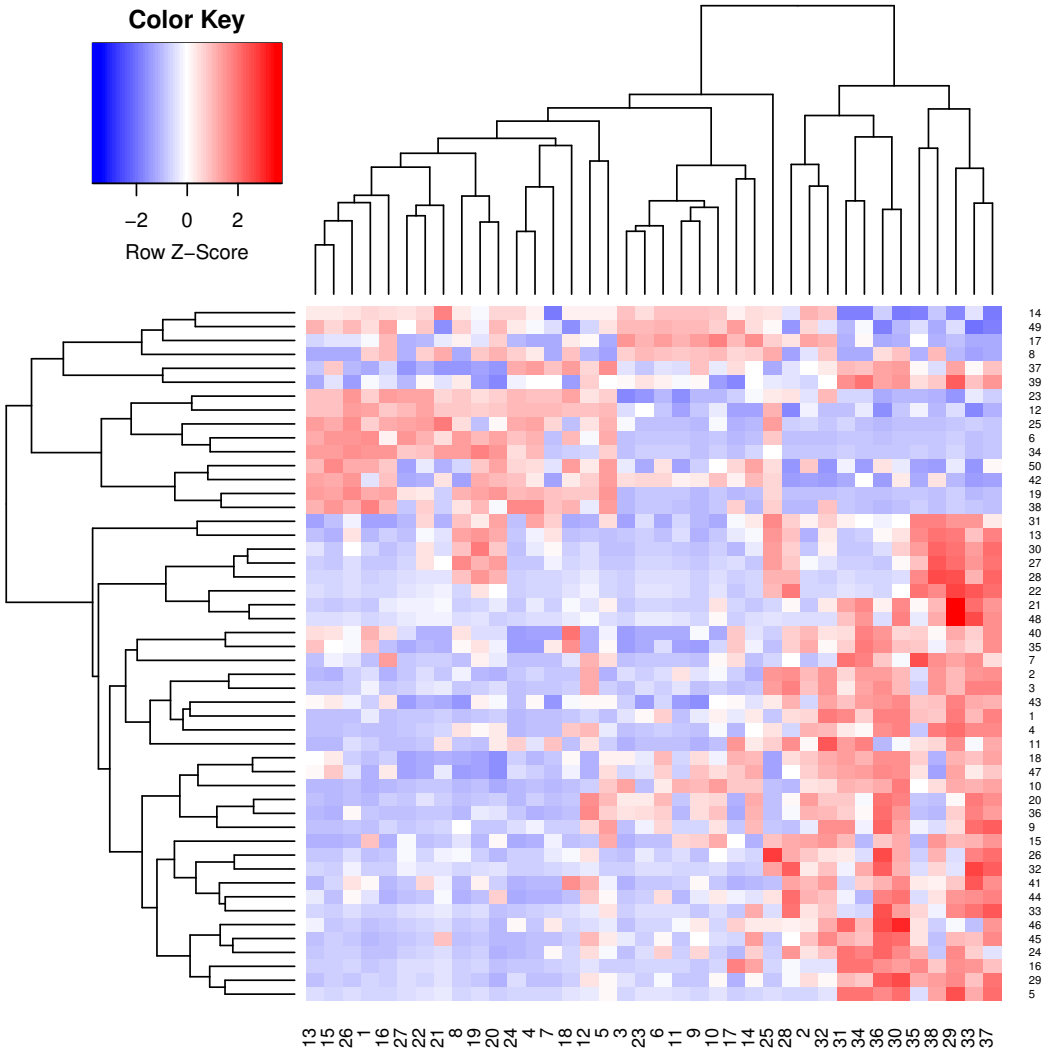

Figure 2.8: Blue-red heatmap of the gene expression values for the Top 50 most differentially expressed genes between the ALL and AML patients (using their median expression values) - where both the patients and genes are clustered based on the similarity of their expression profiles. Each row represents a gene and each column a patient. When using the difference between AML and ALL medians we get better clustering than in Figure 2.7 - with the AML patients (28-38) on the right-hand side completely clustered together and only one ALL patient (patient number 2) misclassified with the AML patients.

2.2.6 Quantile-Quantile plot

Another method to visualize the distribution of gene expression values is by the so-called quantile-quantile (Q-Q) plot. In a Q-Q plot, the quantiles of the gene expression values are displayed against the corresponding quantiles of the normal (bell-shaped) distribution. A straight line is often added to represent points which correspond exactly to the quantiles of the normal distribution. By observing the extent in which the points fall on the line, it can be evaluated to what degree the data are normally distributed. That is, the closer the gene expression values follow the line, the more likely it is that the data are normally distributed.

Example 1: CCND3 gene expression. To produce a Q-Q plot of the ALL gene expression values for the gene CCND3 (Cyclin D3), we can use the `qqnorm()` and `qqline()` functions:

```
> qqnorm(golub[ccnd3, golubFactor=="ALL"],  #values
+        pch=19,       # plot solid circles
+        cex.lab=1.5,  # make axis labels big
+        col="red",
+        ylab="CCND3 (Cyclin D3) Sample Quantiles",
+        main=NULL);   # no title
> qqline(golub[ccnd3, golubFactor=="ALL"], col="blue")
```

From the resulting Figure 2.9 we see that most of the data points are on or near the straight line in the center of the distribution, while the data points increasingly deviate from the line as the points approach either tail of the distribution. This deviation from the diagonal line illustrates a moderate degree of non-normality where both the positive and negative tails are "skinny" compared to the tails of a normal distribution - which is indeed the the case when we look at Figure 2.10. In general, at either tail if the sample quantiles are more horizontal than the normal then the sample has a "fat" tail, and if the sample quantiles are more vertical at a tail then the sample has a "skinny" tail compared to the normal. In the exercises below, the reader will gain more experience with measuring the degree in which gene expression values are normally distributed.

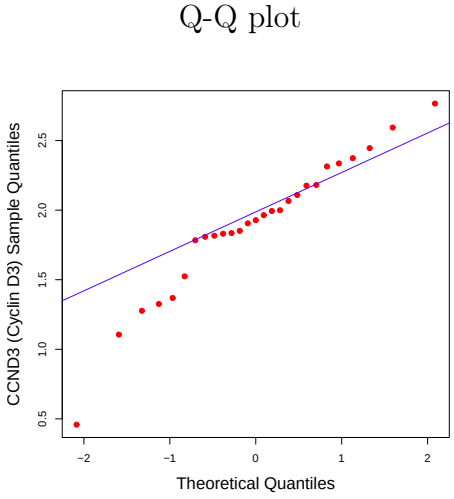

Figure 2.9: Q-Q plot of ALL gene expression values of CCND3 (Cyclin D3).

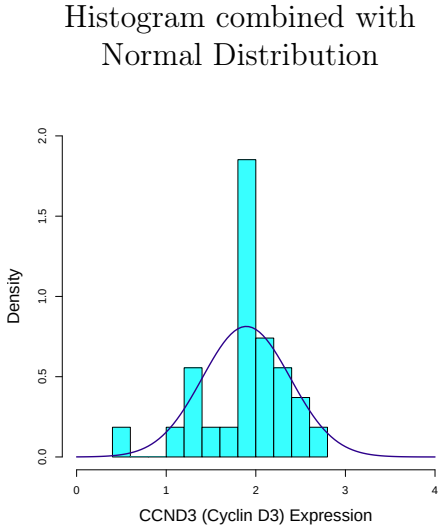

Figure 2.10: Histogram of ALL expression values of gene CCND3 (Cyclin D3) combined with the fitted normal distribution.

2.2.7 Combined plots

It is often the case that we wish to combine multiple plots into one figure. For example, in Figure 2.10 we have superimposed two plots on top of each other to see by eye how well a histogram of the gene expression data fits a normal distribution. We achieve this by adding the normal curve to the existing histogram:

```
> hist(golub[ccnd3, golubFactor=="ALL"],  # values
+      breaks=9,          # number of breaks (bins-1)
+      prob=TRUE,         # scale probability density function
+      xlim=c(0,4),       # x axis range
+      ylim=c(0,2),       # y axis range
+      cex.lab=1.5,       # make axis labels big
+      main=NULL,         # no title
+      xlab="CCND3 (Cyclin D3) Expression",
+      col="cyan")
> m<-mean(golub[ccnd3, golubFactor=="ALL"])
> std<-sqrt(var(golub[ccnd3, golubFactor=="ALL"]))
> curve(dnorm(x, mean=m, sd=std), # fit a normal curve
+       col="darkblue",
+       lwd=2,                    # width 2
```

```
+       add=TRUE,                # add to existing plot
+       yaxt="n")                # don't add to existing y axis
```

When calling the `curve()` function it is necessary to include the `add=TRUE` parameter in order for the curve to be plotted on top of the existing plot.

A combined plot commonly used to check the quality of microarray and next-generation sequencing data is the so-called MA plot - which typically plots the average log_2 of two samples (A) versus their log_2 fold-change (M):

$$M = log_2(sample1/sample2) = log_2(sample1) - log_2(sample2)$$
$$A = \frac{1}{2} log_2(sample1 \times sample2) = \frac{1}{2}(log_2(sample1) + log_2(sample2))$$

However, since the golub dataset has negative relative gene expression values, we will not be plotting in log_2 space. In addition, in an MA plot a LOESS (or LOWESS) local regression curve is added to the plot to graphically show the spatial trend of the datapoints. We will use the `maPlot()` function from the `edgeR` package:

```
> maPlot(meanALL,
+        meanAML,
+        normalize=FALSE,
+        lowess=TRUE,  # plot Loess curve
+        pch=19,       # plot solid circles
+        cex=0.1,      # make solid circles very small
+        cex.lab=1.5,  # make axis labels big
+        logAbundance=.5*(meanAML+meanALL),  # X-values
+        logFC=meanAML/meanALL,              # Y-values
+        xlab="1/2 (mean(AML) + mean(ALL))",
+        ylab="mean(AML) / mean(ALL)")
> abline(h=1, lwd=2, lty=2, col="blue")    # add horizontal line
> abline(v=0, lwd=2, lty=2, col="darkgreen")   # add vertical line
```

Note that the `abline()` function automatically adds a line to the existing plot and does not require an `add=TRUE` argument. We will look at the MA plots for both ALL/AML and AML/ALL fold-changes to see if we can spot any biases in the data. Here is the same code now looking at the ALL/AML fold-change in expression:

```
> maPlot(meanAML,
+        meanALL,
+        normalize=FALSE,
+        lowess=TRUE,  # plot Loess curve
+        pch=19,       # plot solid circles
+        cex=0.1,      # make solid circles very small
+        cex.lab=1.5,  # make axis labels big
+        logAbundance=.5*(meanAML+meanALL),  # X-values
+        logFC=meanALL/meanAML,              # Y-values
```

```
+         xlab="1/2 (mean(AML) + mean(ALL))",
+         ylab="mean(ALL) / mean(AML)")
> abline(h=1, lwd=2, lty=2, col="blue")    # add horizontal line
> abline(v=0, lwd=2, lty=2, col="darkgreen")   # add vertical line
```

Modified MA Plot 1

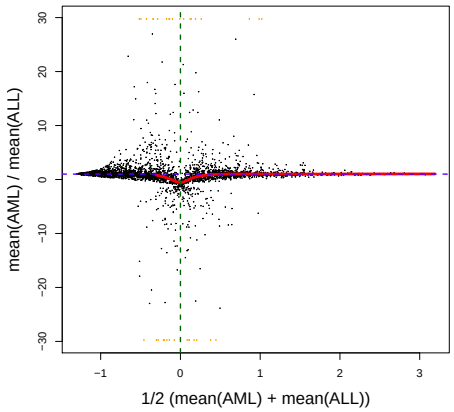

Figure 2.11: A modified MA plot that plots the AML/ALL fold-change in expression versus the amount of expression.

Modified MA Plot 2

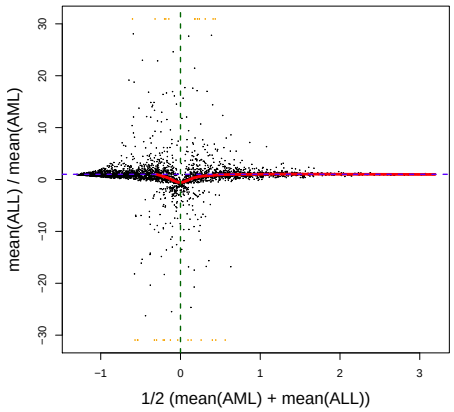

Figure 2.12: A modified MA plot that plots the ALL/AML fold-change in expression versus the amount of expression.

The resultant MA plots in Figures 2.11 and 2.12 show us two important characteristics of the data. First, most genes do not exhibit significant difference in expression between the ALL and AML patients. Second, as the mean expression values approach zero from both sides, the deviation of the fold-changes in expression increases dramatically. This fold-change versus abundance relationship is typical of microarray and next-gen sequencing data. This relationship informs us that the noise of the data increases as expression levels decrease - or equivalently, the signal-to-noise ratio decreases as the level of expression decreases. Therefore, we should be hesitant to trust the measurement accuracy of lowly expressed genes.

We will use combined plots to answer another important, related question: "How well does the difference between the AML and ALL gene expressions correlate with the ALL vs. AML diagnosis?" To answer this question we will plot the difference between the mean ALL and AML expression versus the Pearson's correlation with the ALL versus AML diagnosis. Then we will superimpose a LOESS curve and some straight lines as visual aids. The

LOESS curve is a smoothly piecewise "LOcal regrESSion" curve that best fits the data and summarizes the general trends in the data:

```
> x = meanAML-meanALL
> y = apply(golub, 1, function(x) {cor(x, golub.cl)} )
> plot(x,            # X-values
+      y,            # Y-values
+      pch=19,       # plot solid circles
+      cex=0.1,      # make solid circles very small
+      cex.lab=1.5,  # make axis labels big
+      xlab="mean(AML) - mean(ALL)",
+      ylab="Pearson's Correlation with Diagnosis")
> lines(lowess(x,y), col="orange", lwd=4) # add lowess line (x,y)
> abline(h=0, lwd=2, lty=2, col="blue")   # add horizontal line
> abline(v=0, lwd=2, lty=3, col="red")    # add vertical line
```

Below is a similar plot that looks at the relationship between the absolute values of: (1) the differences in gene expression and (2) the correlation with diagnosis:

```
> x = abs(meanAML-meanALL)
> y = apply(golub, 1, function(x) {abs(cor(x, golub.cl))} )
> plot(x,            # X-values
+      y,            # Y-values
+      pch=19,       # plot solid circles
+      cex=0.1,      # make solid circles very small
+      cex.lab=1.5,  # make axis labels big
+      xlab="abs(mean(AML) - mean(ALL))",
+      ylab="Absolute Value of Pearson's Correlation with Diagnosis")
> lines(lowess(x,y), col="orange", lwd=4) # add lowess curve (x,y)
> abline(h=0, lwd=2, lty=2, col="blue")   # add horizontal line
> abline(v=0, lwd=2, lty=2, col="red")    # add vertical line
```

Indeed, the resultant Figures 2.13 and 2.14 show us that the correlation with diagnosis decreases dramatically as the difference in mean gene expression approaches zero. However, even for genes that have a large difference in mean expression between the ALL and AML patients, the correlation with diagnosis can vary dramatically.

Figures 2.11, 2.12, and 2.14 show us two very important characteristics of this gene expression data (and expression data in general). First, we cannot trust expression values close to zero because those signals are dominated by noise. Second, gene expressions with small differences between conditions, in general, do not correlate well with the different conditions. Thus, good gene expression biomarkers should have at least one measurement far from zero and also exhibit considerable difference in expression between the conditions.

Combined Plot 1

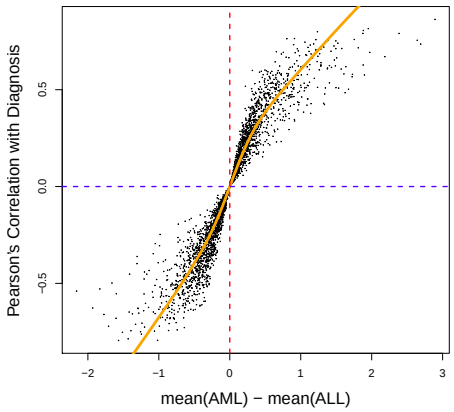

Figure 2.13: A combined plot that contains a scatter plot, LOESS regression curve, vertical line, and horizontal line. The difference in mean ALL vs. AML gene expression is plotted versus the Pearson's correlation with the diagnosis.

Combined Plot 2

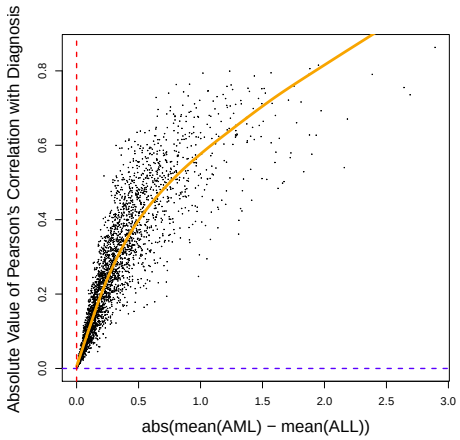

Figure 2.14: Another combined plot that also contains a scatter plot, LOESS regression curve, vertical line, and horizontal line. The magnitude of the difference in mean ALL vs. AML gene expression is plotted versus the magnitude of the Pearson's correlation with the diagnosis.

2.3 Overview and concluding remarks

Data can be stored as a vector, matrix, or data.frame and many useful statistical and visual functions are defined to better understand the data and ascertain important characteristics. In particular, it is easy to produce pie, histogram, scatter, strip, box, Q-Q, heatmap, MA, and combined plots to graphically interrogate the data. These plots typically give useful first impressions that provide the direction for further detailed analysis.

2.4 Exercises

Since the majority of the exercises are based on the Golub et al. (1999) data, it is essential to make these available and to learn to work with it. To stimulate self-study the answers are given at the end of the book.

1. **Illustration of mean and standard deviation.**

(a) Compute the mean and the standard deviation for $1, 1.5, 2, 2.5, 3$.

(b) Compute the mean and the standard deviation for $1, 1.5, 2, 2.5, 30$.

(c) Comment on the differences.

2. **Testing normality of gene expression.** Consider the gene expression values in row 790 and 66 of the Golub et al. (1999) data.

   (a) Produce a box plot for the expression values of the ALL patients and comment on the differences. Are there outliers?

   (b) Produce a QQ-plot and formulate a hypothesis about the normality of the genes.

   (c) Compute the mean and the median for the expression values of the ALL patients and compare these. Do this for both genes.

3. **Effect size.** An important statistic to measure is the effect size which is defined for a sample as $\overline{x}/s$. It measures the mean relative to the standard deviation, so that its value is large when the mean is large and the standard deviation small.

   (a) Determine the five genes with the largest effect size of the ALL patients from the Golub et al. (1999) data. Comment on their size.

   (b) Invent a robust variant of the effect size and use it to answer the previous question.

4. **Plotting gene expressions for** `CCND3`. Use the gene expressions from `CCND3 (Cyclin D3)` of Golub et al. (1999) collected in row 1042 of the matrix `golub` from the `multtest` library. Use `grep()` to get the correct row for the `CCND3 (Cyclin D3)` gene expression values. After using the function `plot()`, you will produce an object on which you can program.

   (a) Produce a so-called stripchart for the gene expressions separately for the ALL as well as for the AML patients. Hint: Use `factor()` to separate the data between the two categories.

   (b) Rotate the plot to a vertical position and keep it that way for the questions to come.

(c) Color the ALL expressions red and AML blue. Hint: Use the `col` parameter.

(d) Add a title to the plot. Hint: Use the `title()` function.

(e) Change the boxes into stars. Hint 1: Use the `pch` parameter. Hint 2: Using your favorite text editor, save the final script for later use.

5. **Box-and-Whiskers plot of `CCND3` expression.** Use the gene expressions for `CCND3 (Cyclin D3)` of Golub et al. (1999) from row 1042 of the matrix `golub` for the ALL patients. Use `grep()` to get the correct row for the `CCND3 (Cyclin D3)` gene expression values.

   (a) Construct the box plot in Figure 2.15.

   (b) Add text to the plot to explain the meaning of the upper and lower part of the box.

   (c) Do the same for the whiskers.

   (d) Export your plot to `eps` format.

   Hint 1: Use `boxplot.stats()` to find coordinates of the positions in the plot.
   Hint 2: Use parameter `xlim` when calling `boxplot()` to make the plot somewhat wider.
   Hint 3: Use `arrows()` to add an arrow.
   Hint 4: Use `lwd()` to make line widths wider.
   Hint 5: Use `text()` to add information at a certain position.

6. **Box-and-whiskers plot of patients.**

   (a) Use `boxplot(data.frame(golub))` to produce a box-and-whiskers plot for each column (patient). Make a screen shot to save it in a word processor. Describe what you see. Are the medians of similar size? Is the inter quartile range more or less equal? Are there outliers?

   (b) Compute the mean and medians of the patients. What do you observe?

(c) Compute the range (minimal and maximum value) of the standard deviations, the IQR and MAD of the patients. Comment on what you observe.

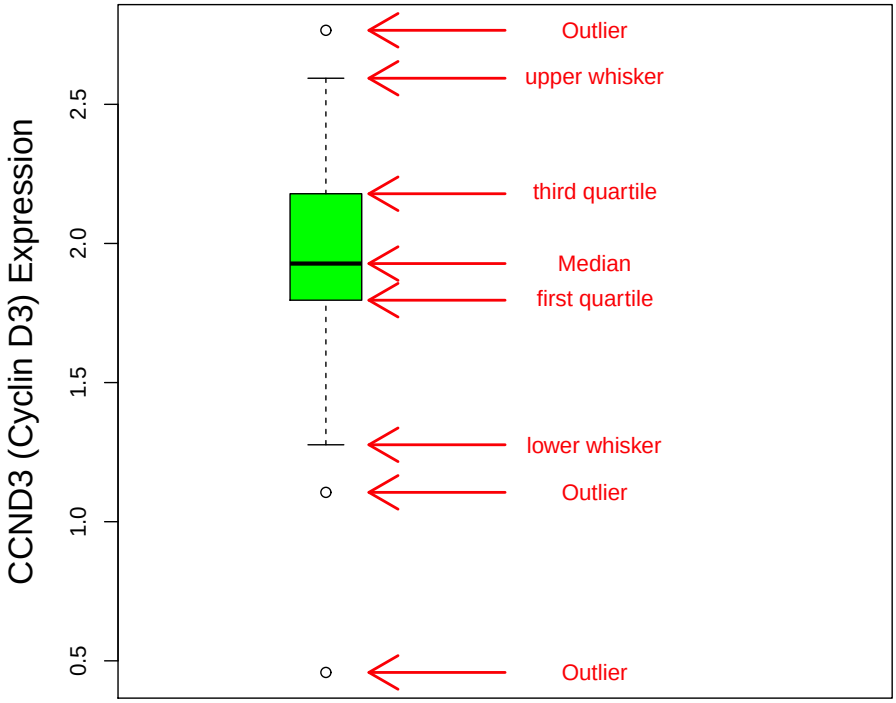

Figure 2.15: Box plot with arrows and explaining text.

7. **Oncogenes in the Golub et al. (1999) data.**

(a) Select the oncogenes with the `grep("oncogene")` function and produce a box-and-whiskers plot of the gene expressions of the ALL patients. Be sure to perform a case-insensitive `grep()` search.

(b) Do the same for the AML patients and use `par(mfrow=c(2,1))` to combine the two plots such that the second is beneath the first. The `par(mfrow=c(2,1))` command splits the plotting canvas into 2 rows and 1 column. After the two `boxplot()` calls then you can go back to the default 1 plot per window behavior with the `par(mfrow=c(1,1))` command. Are there genes with clear differences between the groups?

8. **Descriptive statistics for the ALL gene expression values.**

   (a) Compute the mean and median for gene expression values of the ALL patients, report their range and comment on it.

   (b) Compute the SD, IQR, and MAD for gene expression values of the ALL patients, report their range and comment on it.