Alencar Xavier

Graphics in R for Genetics

Dataset

Data can be loaded into R from your computer, from the internet, from data bases (3), from packages, and some datasets are come with R. Statistical packages usually provide datasets that illustrate the functions available in the package. For this exercise, we are loading a wheat dataset which corresponds to an experimental multi-environmental trial from CYMMIT.

Loading data into R's global environment

The dataset can be loaded as follows:

```
data(wheat, package = 'BGLR')
```

If you don't have the package BGLR installed, do so:

```
install.packages('BGLR')
```

An additional dataset will be require for the spatial analysis using heatmap. This dataset can be loaded as:

```
data(met, package = 'NAM')
```

If you don't have the package NAM installed, do so:

```
install.packages('NAM')
```

BGLR dataset reference

- Crossa et al (2010). Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers. Genetics, 186(2), 713-724.
- Perez-Rodriguez et al (2012). Comparison between linear and non-parametric regression models for genome-enabled prediction in wheat. G3: Genes, Genomes, Genetics, 2(12), 1595-1605.
- Daetwyler et al (2013). Genomic prediction in animals and plants: simulation of data, validation, reporting, and benchmarking. Genetics, 193(2), 347-365.

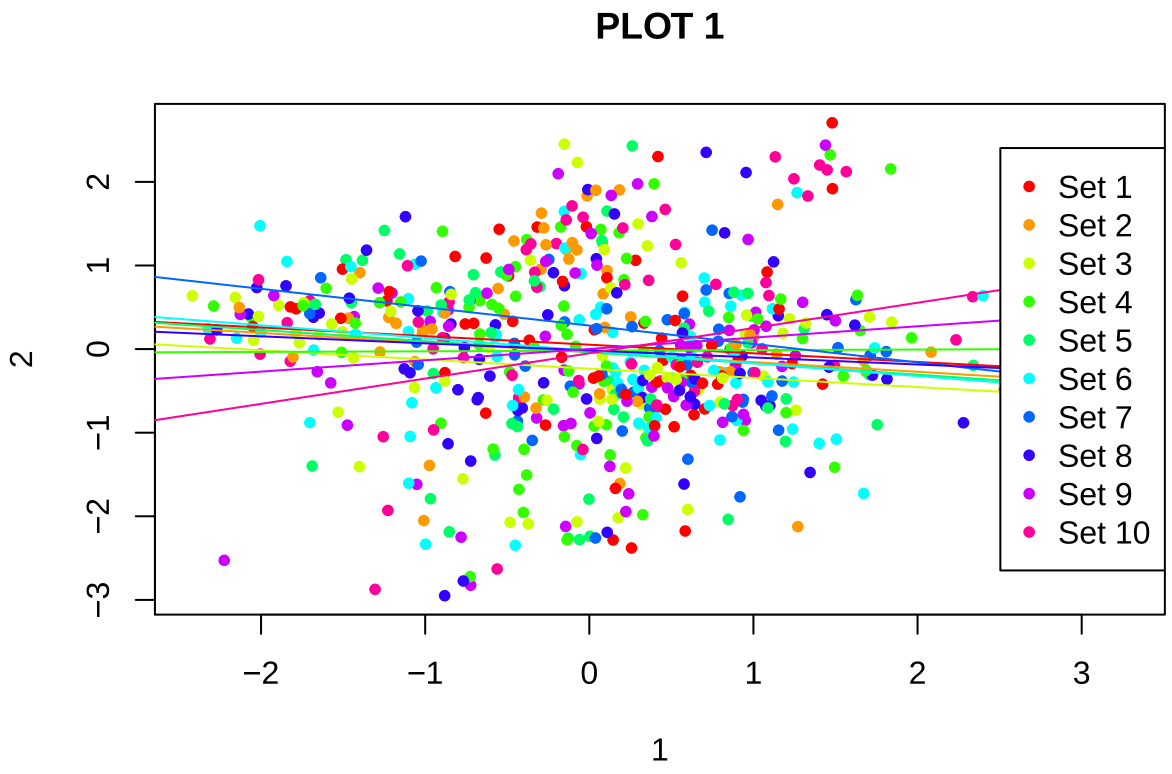
NAM dataset reference

- Xavieret al (2017). Genetic Architecture of Phenomic-Enabled Canopy Coverage in Glycine max. Genetics, 206(2), 1081-1089.

Plot 1 - Scatter plot

Below we are plotting yield from two locations from the matrix `wheat.Y` using closed dots (`pch=20`), titled plot 1 (`main="PLOT 1"`), and in 10 rainbow colors (`rainbow(10)`) where each color represent one of the 10 wheat sets. The next step will be to add a fitted line for each of the 10 subsets using a `for` loop with the function `abline` to add the fitted lines, which are computed through the linear model function (`lm`). The loop below can be read as "*for each component (i) in the set (1:10) do the following*". Then, we will add legend in the right side (`legend("right",...)`). Thus:

```
COLORS = rainbow(10)

plot(wheat.Y, col = rainbow(10)[wheat.sets], pch=20, main="PLOT 1")

for(i in 1:10) abline( lm(wheat.Y[,1]~wheat.Y[,2],subset=wheat.sets==i), col=rainbow(10)[i])

legend("right",legend = paste("Set",1:10), col=rainbow(10)[1:10], pch=20)
```



Plot 2 - Density plot

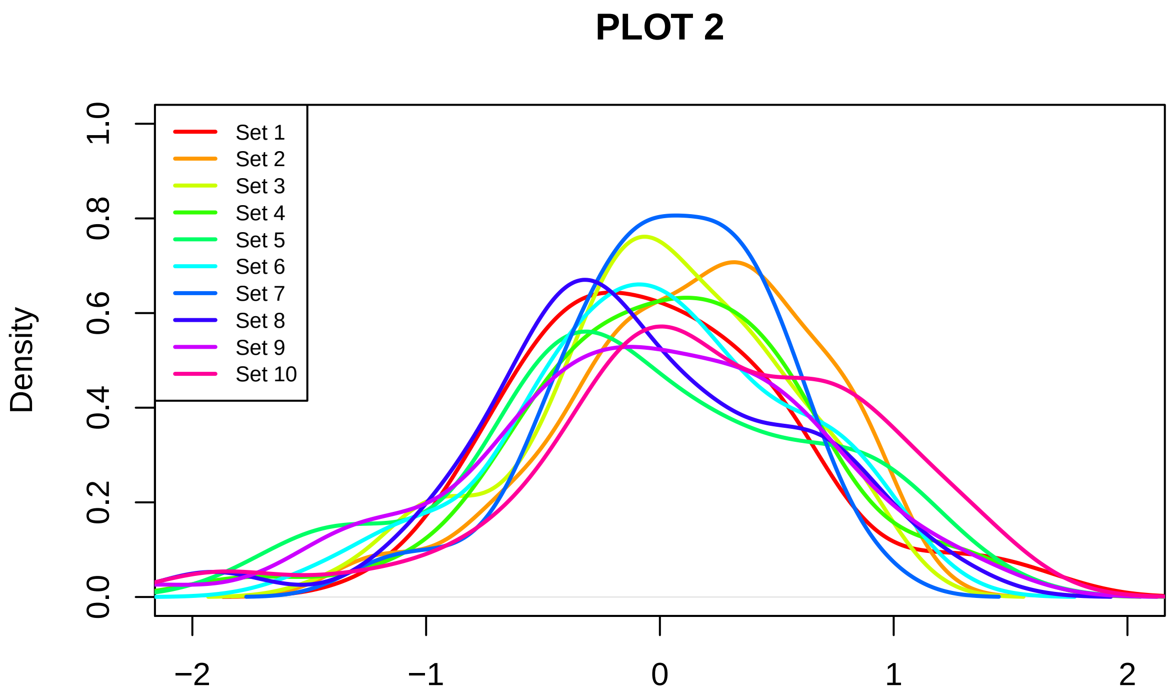
The dataset under investigation has 4 environments, but for simplicity we will average the performance of each genotype across environment using `rowMeans`. We are plotting this average yield estimate from each wheat subset with the same rainbow colors used before, on a plot titled plot 2 (`main="PLOT 1"`), and we are defining the limits of x and y axis of this plot beforehand (`xlim=c(A,B)` and `ylim=c(C,D)`). The way this overlay of plots will be done is by plotting the first group using `plot`, and the following 9 groups using a `for` loop with the function `lines` to draw the additional lines. For illustration purpose, the line width will be twice as thicker (`lwd=2`). The legends will be added, now on the left corner, with slightly smaller letters, just 70% the size of the legend from PLOT 1 (`cex=0.7`).

```
Y = rowMeans(wheat.Y)
```

```
plot(density(Y[wheat.sets==1]),xlim=c(-2,2),ylim=c(0,1),col=COLORS[1],main='PLOT 2', lwd=2)
```

```
for(i in 2:10){ lines(density(Y[wheat.sets==i]),col=COLORS[i], lwd=2) }
```

```
legend("topleft",legend = paste("Set",1:10), col=rainbow(10)[1:10], lwd=2, cex=0.7, lwd=2)
```

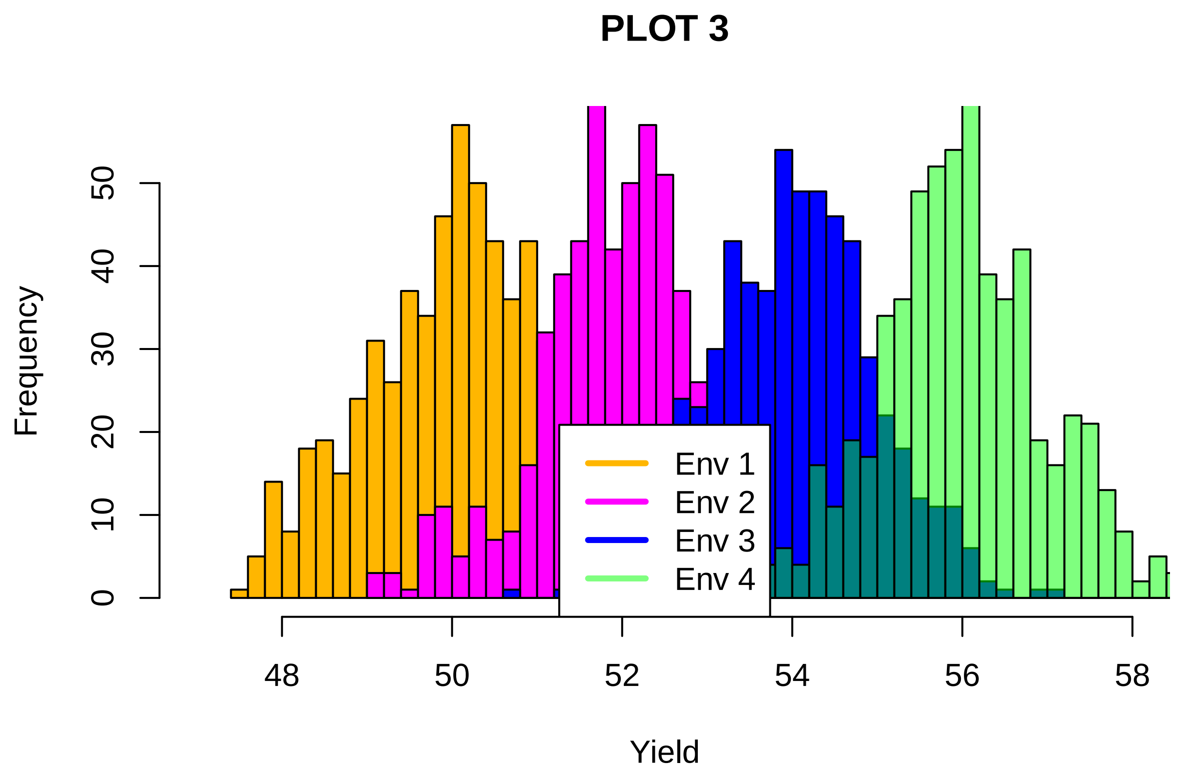


Plot 3 - Histogram

Histograms (`hist`) provide similar information as density plots, they are useful to assess the normality of the data and the existence of outliers. Here, we are plotting an overlay histogram of all four environment. Because the environments are zero-centered, we will be adding an intercept in each location as if the various environment yielded differently. In the function `hist`, the number of breaks define the number of bars to be used, and additional histograms can be overlay using `add=TRUE`. In addition, we are also demonstrating here that there are different ways of represent colors in R, that is: through the color name (eg. `blue`), color functions (eg. `rainbow(5)[3]`), the numeric code (from 1 to 8), and using the **rgb** description of the color, which that also allows us to define the level of transparency (eg. `rgb(red = 1,blue = 0.2,green = 0.3,alpha = 0.25)`).

Thus,

```
COLORS = c( heat.colors(10)[6], "magenta", 4, rgb(0,1,0,0.5) )

hist( 50 + wheat.Y[,1],breaks=30,col=COLORS[1],xlim=c(47,58),xlab='Yield',main='PLOT 3' )

hist( 52 + wheat.Y[,2],breaks=30,add=T,col=COLORS[2] )

hist( 54 + wheat.Y[,3],breaks=30,add=T,col=COLORS[3] )

hist( 56 + wheat.Y[,4],breaks=30,add=T,col=COLORS[4] )

legend( "bottom", legend = paste("Env",1:4), col=COLORS, lwd=3, bg = 'white' )
```



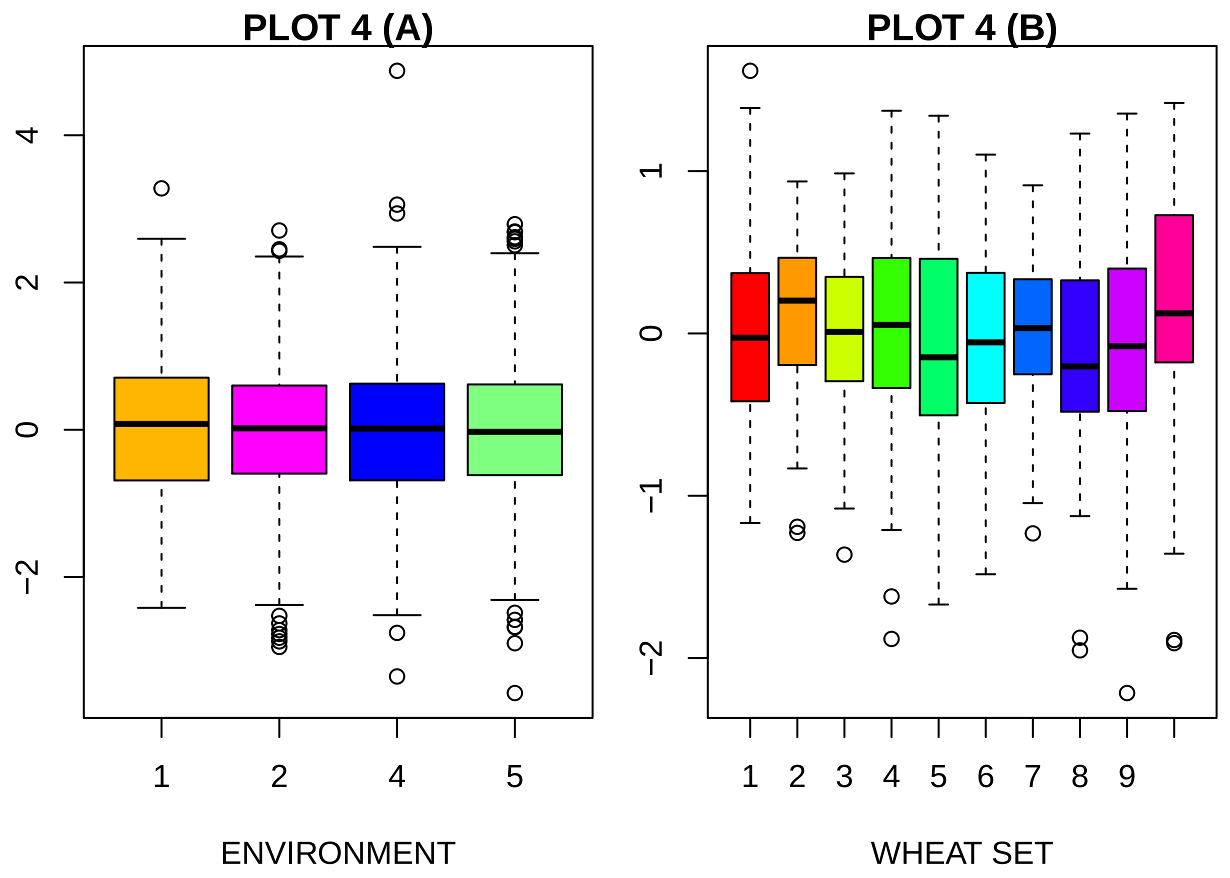
Plot 4 - Box plot

Box plots are used to describe the dispersion of the data through the *quantiles* across various levels. The *box* indicates where 50% of the data sits by showing the 25% percentile, median and 75% percentile. Data points outside the whiskers can be interpreted as outliers, where the whiskers roughly represent the 95% confidence intervals.

The function `boxplot` is flexible with its input, accepting either matrices containing the data or formulas described the terms you want to see (the phenotype and the categorical stratification). In addition, we are showing how to obtain multiple plots in the same figure through the plot parameterization function `par`, where we are asking our plots to fit into a matrix with 1 row and 2 columns (`mfrow = c(Rows,Columns)`), with margins smaller than what R makes by default (`mar=c(Dow,Left,Up,Right)`). We are also adding a label to the X-axis to help with the interpretability of the graphics (argument `xlab` inside the plot function).

Thus,

```
par(mfrow = c(1,2))

boxplot( wheat.Y, col=COLORS, main='PLOT 4 (A)',xlab='ENVIRONMENT')

boxplot( Y~wheat.sets, col=rainbow(10), main='PLOT 4 (B)', xlab='WHEAT SET')
```

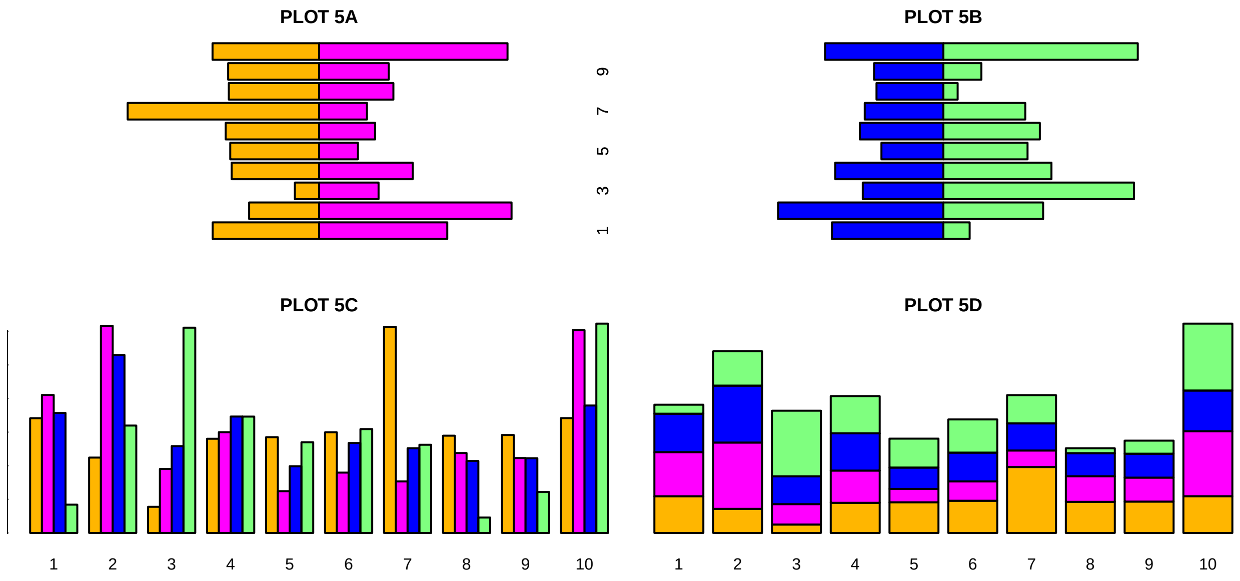


Plot 5 - Bar plots

Bar plots is a common approach to present and compare means. In R, the function `barplot` requires as input the means of the levels, which can be provided as vector or matrix. Matrices must be provided in the case we seek to present the mean in two levels. In the example that follows, we are going to use the simple bar plot to compare the wheat sets from two environments in a mirror-bar plot, and then we are going to compare the wheat sets in all four environment side-by-side. To obtain the average performance of each wheat set within each environment we are using the function `tapply`. The argument of this are function are: 1) the value or phenotype; 2) the index or vector indicating sets; and 3) the function to apply in each index, in this case we want to use `mean`.

To make the mirror or bar plot, we will make the value of environment 1 negative such that each environment will be presented towards one direction. Then, we will use `barplot` with the arguments `horiz=TRUE` to make horizontal bar plots, plotting environment 1 first followed by plotting environment 2 with the argument `add=TRUE` to overlay both graphics. The second type of plot (with all sets and environments) is plotted in straight forward manner, providing a matrix where each cell is the average performance of a set (columns) within a environment (row).

```
par(mfrow=c(2,2),mar=c(2,0,2,0),cex=0.5)

TABLE = t(apply(wheat.Y,2,function(x) tapply(x,wheat.sets,mean)))+0.3

barplot(-TABLE[1,],main='PLOT 5A',col=COLORS[1],xlim=c(-1,1),horiz=T, xaxt='n')

barplot(TABLE[2,],add=T, col=COLORS[2], horiz = T, xaxt='n')

barplot(-TABLE[3,],main='PLOT 5B',col=COLORS[3],xlim=c(-1,1),horiz=T, xaxt='n')

barplot(TABLE[4,],add=T, col=COLORS[4], horiz = T, xaxt='n')

barplot(TABLE,beside = T,col=COLORS, offset=-.3,main='PLOT 5C')

barplot(TABLE,beside = F,col=COLORS, offset=-.3,main='PLOT 5D')
```

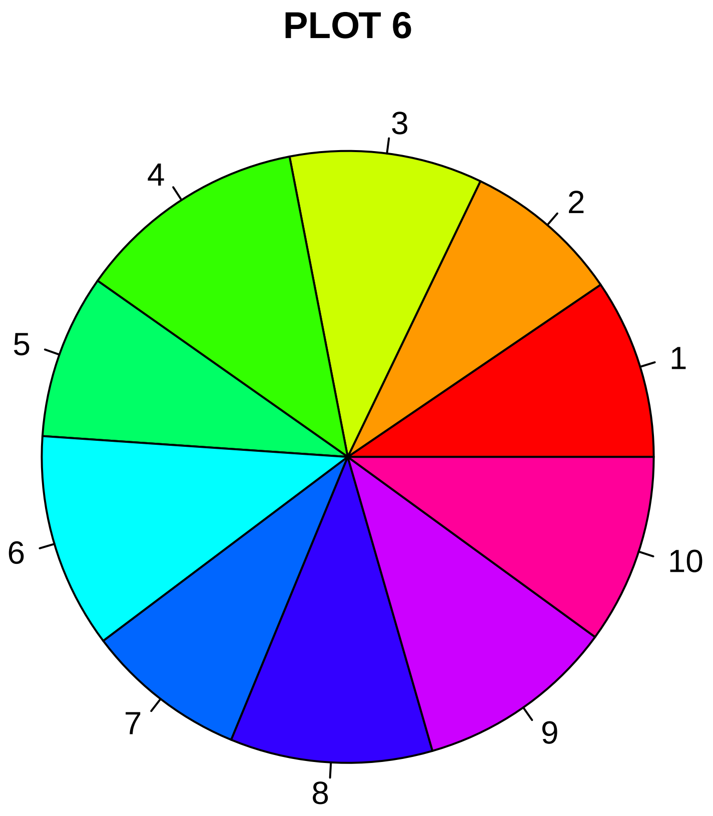


Plot 6 - Pie chart

Pie chart is a circular graphic divided into slices to illustrate proportions. In the example below, we are plotting the proportion of genotypes that come from each wheat set. The `pie` function is used to generate the charts, and the input is the count or proportion of each factor. In this exercise we used the function `table` to obtain the counts of each set. For this example, we are also reducing the margin of the image with `par` to reduce the empty spaces in the image.

```
par(mfrow=c(1,1),mar=c(0,0,1,0))
```

```
pie(table(wheat.sets),col=rainbow(10),main='PLOT 6')
```

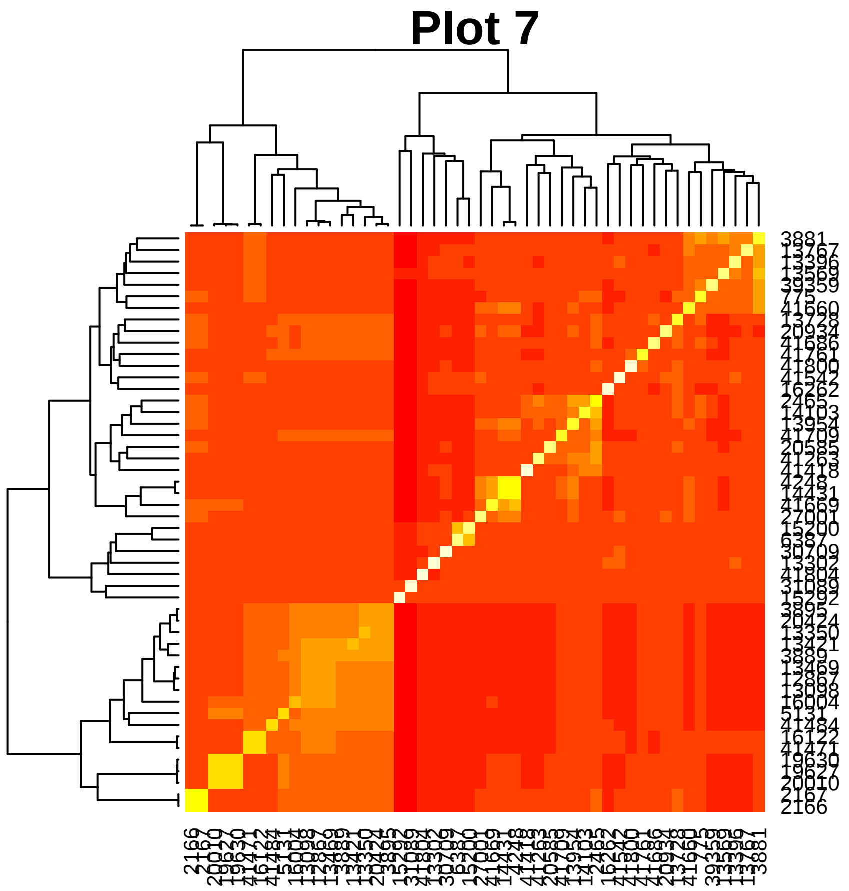


Plot 7 - Heatmap

Heatmaps are very informative visualization of data from 2 or 3 dimensions. Major applications of heatmaps include (1) the identification of pattern of geographical data such as yield in a crop field, and (2) color-coding relationship matrices such as correlations and relationship matrices.

Correlation matrices heatmap. For the first example we are going to use the first 50 individuals from the relationship matrix provided by the BGLR package: `wheat.A`. The `heatmap` function will generate a heatmap that find patterns and clusters the rows and columns of the data. This function implies in demanding computational capabilities when large matrices are involved.

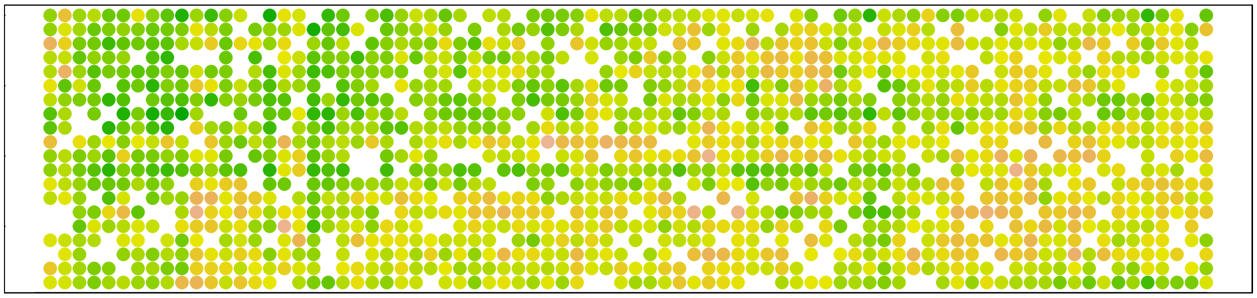
```
heatmap(wheat.A[1:50,1:50], main="Plot 7")
```



Field patterns. For the second example we are going to use the observations from Block 5, data frame `Obs` from the `met` data (NAM package). In the code below we are showing how to display a heatmap as a **scatter plot** and as **image**.

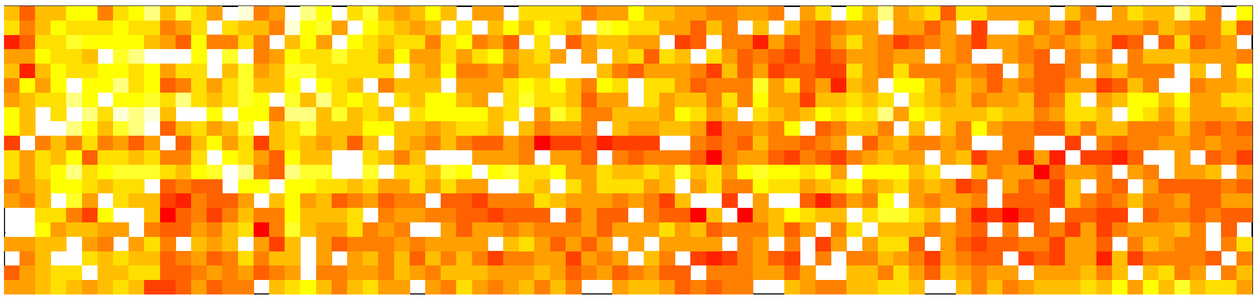
The scatter plot heatmap is fundamentally based on plotting the coordinates and color-coding each observation. Below, we are plotting *yield* as it was observed in the field on Block 5. Hence we are creating a dataset named `FD`, which is a subset of the data `Obs` containing only the observations from block 5 (`which(Obs$Block==5)`). The plot is made with the `plot` function. The colors are assigned in continuous scale using `terrain.colors` ranging from 1 to 100, where 1 means pale orange and 100 means green. Each datum is represented by a closed dot (`pch=20`).

```
FD = Obs[which(Obs$Block==5),]

COL = rev(terrain.colors(100))

plot( FD$Col, FD$Row, col = COL[FD$YLD], pch=20, lwd=2)
```



The image version of heatmap require us to generate a matrix that represents the field as it is, like a map, similar to the heatmap functionality available in MS Excel. For that, we are creating an empty matrix `M` with the field dimension (20x80). We are then filling each cell with the correspondent data point using a double `for` loop that uses the `ifelse` to check whether each data point is present in our dataset, filling with `NA` otherwise. At last, we use the function `image` upon this matrix to print the heatmap.

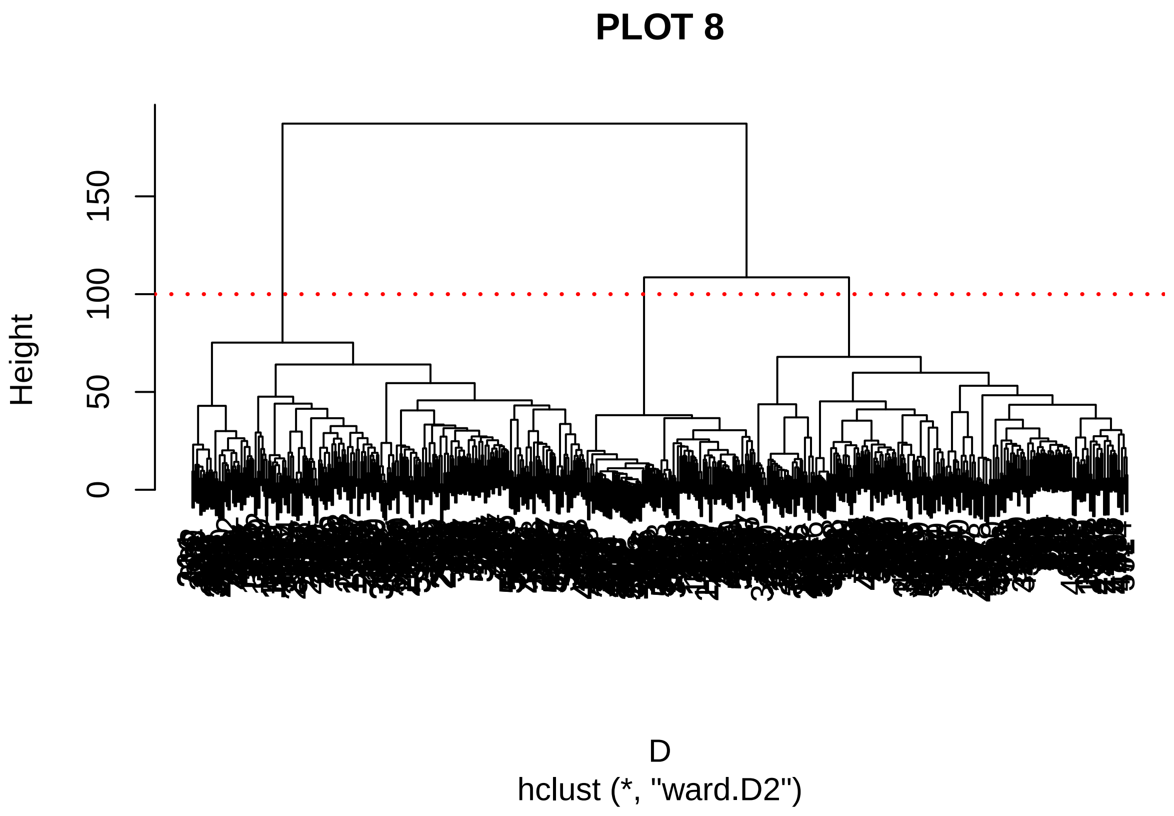
```
M = matrix(NA,20,80)

for(i in 1:20) for(j in 1:80)

M[i,j] = ifelse(any(FD$Row==i&FD$Col==j), yes=FD$YLD[which(FD$Row==i&FD$Col==j)], no=NA)

image(t(M))
```



Plot 8 - Phylogram

Phylograms, also known as dendrogram, cladograms or phylogenetic trees, are branching diagrams that illustrate levels of relatedness among individuals as estimated through a statistical procedure known as hierarchical clustering. Hierarchical clustering follows two steps: dissimilarity and agglomeration. Dissimilarity refers to computation of the distance among individual based on a set of parameters - which correspond to molecular markers in genetics, for that we are using the function `dist` to estimate the Euclidean distance (`method = "euclidean"`). Agglomeration refers to how we bring the individuals together based on the estimated distance. Different methods of agglomeration provide differences in topology, here we are using Ward's D (`method = "ward.D2"`), which is often used in genetics to obtain balanced groups and it also has been reported to provide groups similar to the software STRUCTURE. After plotting, we are adding a horizontal (`h=100`) red dotted lines (`col=2, lty=3`) using the function `abline`, indicating where the tree could be cut to split the germplasm into three clusters. We then cut the three into these 3 clusters (`cutree(HG, k=3)`).

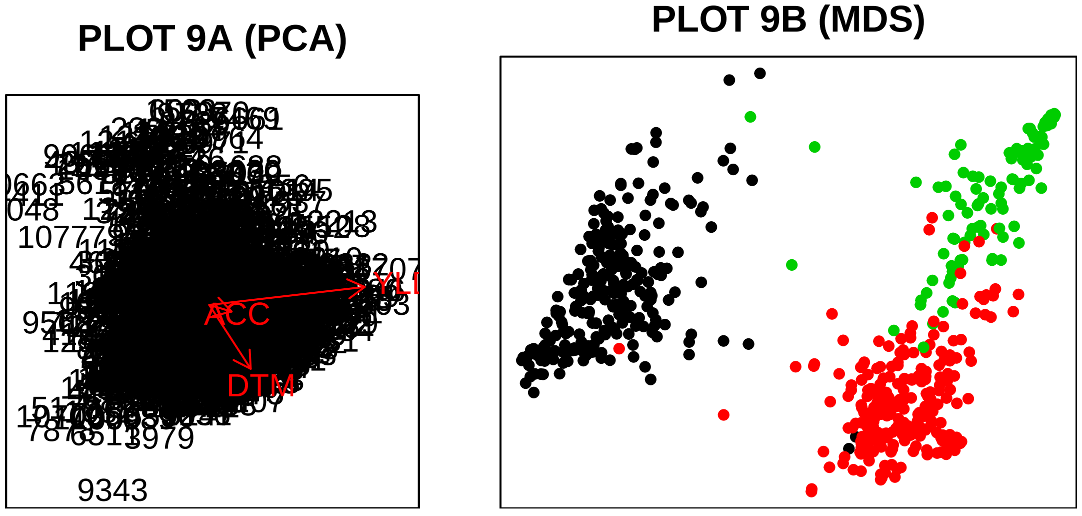
```
D = dist( wheat.X, method = "euclidean" )

HC = hclust( D, method = "ward.D2" )

plot( HC, main = "PLOT 8" )

abline(h = 100, col = 2, lty = 3, lwd = 2)

Wheat_Clusters = cutree( HC, k=3)
```



Plot 9 - Biplot

Although the term biplot can be used to describe any scatter plot, it has recently been reserved to plot Eigen vectors for principal component analysis (PCA) or multi-dimensional scaling (MDS). Below, we are plotting both PCA and MDS (`par(mfrow=c(1,2))`). To demostrate how to generate PCA biplot, we are using the function `princomp` upon the traits of the `FD` data (used for *heatmap*). We are also generating MDS using the `cmdscale` function, reducing the distance matrix (used for *phylogram*) to two dimension (`k=2`), and coloring according to the clusters estimated from cutting the phylogenic tree in the previous exercise.

```
par(mfrow=c(1,2))

PCs = princomp(FD[,6:8])

biplot( PCs, main = "PLOT 9A (PCA)")

MDS = cmdscale(D, k=2 )

plot( MDS, main="PLOT 9B (MDS)", pch=20, col=Wheat_Clusters )
```



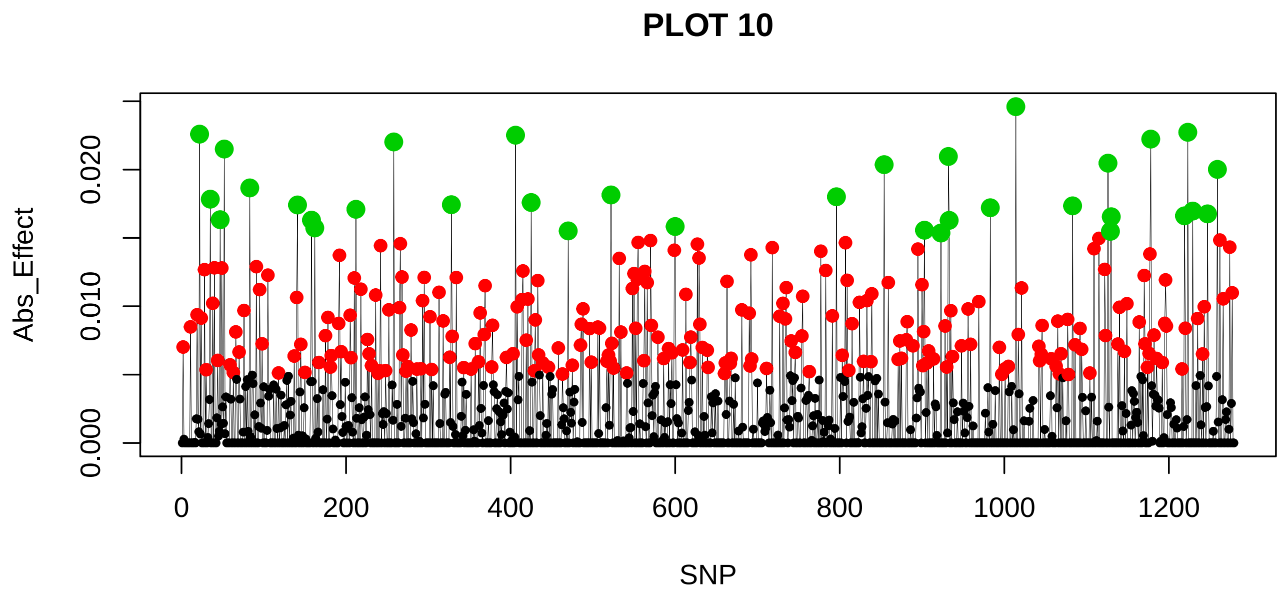
Plot 10 - Manhattan plot

Manhattan plot is a scatter plot designed to illustrate the association between molecular markers and phenotypes. There are various way of displaying this association, including: -log(*p-value*), SNP effect, variance components, posterior probability and variable importance. Each chromosome is usually plotted in a different color, and SNPs thought to be located in candidate genes often are presented in a different shape. In the example below, we are estimating the SNP effects through a elastic-net implemented in the function `emEN` in the package `NAM` (many other packages are specifically design for fitting highdimensional models, such as `glmnet`, `BGLR`, `ranger` and `bwGR`). Here we are also creating a vector to represent the color and size of each SNP, but we do not have the chromosome information to color-code chromosomes.

```
Abs_Effect = abs(NAM::emEN(Y,wheat.X)$b)
```

```
COLOR = round(Abs_Effect*100)
```

```
plot(Abs_Effect,xlab="SNP",pch=20,lwd=COLOR*4,col=COLOR+1,type='o',main='PLOT 10')
```



Other plots

Mosaic plot

```
example(mosaicplot)
```

Association plot

```
example(assocplot)
```