



BIOC | BioInformatics Competence Center

Bioinformatic Analysis of RNA-sequencing

R Introduction : Plots

Allison Burns, Maxime Jan, Linda Mhalla, Christian Iseli, Nicolas Gueux

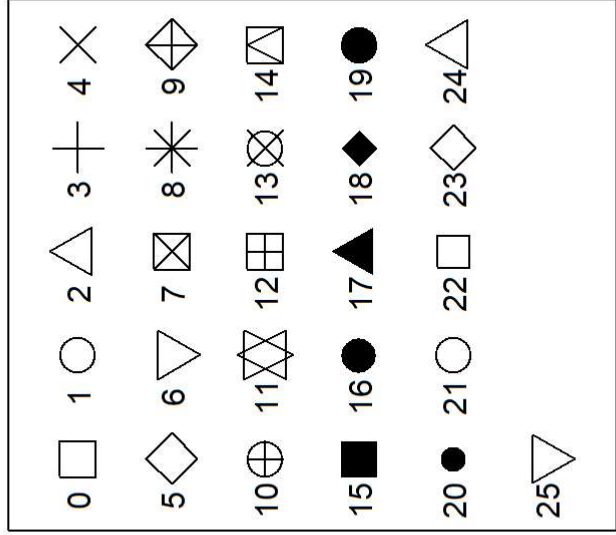
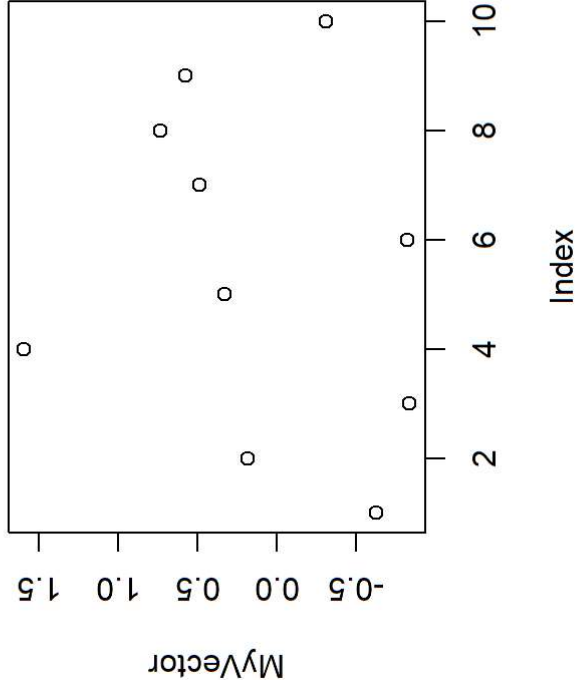
General plot function

Scatterplot

```
MyVector<-rnorm(10)  
plot(MyVector,main="ScatterPlot",pch=1)
```

You can customise the type of points you want using e.g. `pch=22`

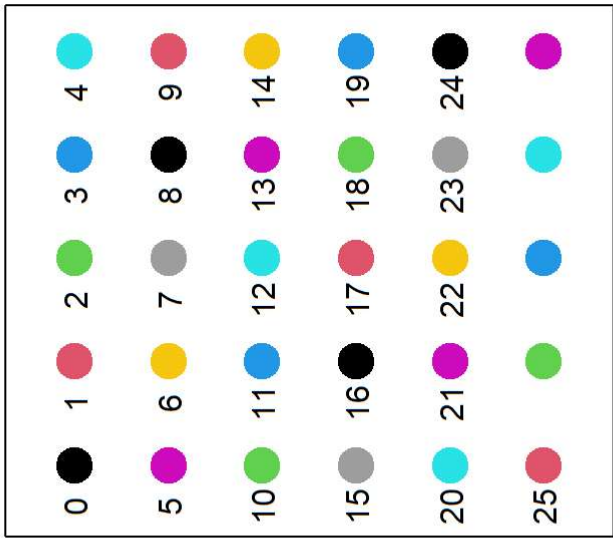
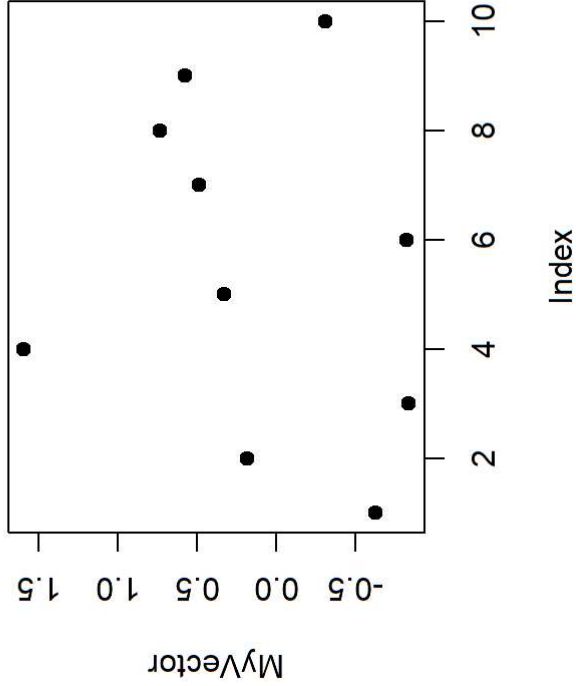
Scatterplot



You can customize also color

```
plot(MyVector,main="Scatterplot",pch=19,col=1)
```

Scatterplot



or use name like:

```
plot(MyVector,main="Scatterplot",pch=19,col="red")
```

color names

white	aliceblue	antiquewhite	antiquewhite1	antiquewhite2
antiquewhite3	antiquewhite4	aquamarine	aquamarine1	aquamarine2
aquamarine3	aquamarine4	azure	azure1	azure2
azure3	azure4	beige	bisque	bisque1
bisque2	bisque3	bisque4		blanchedalmond
blue	blue1	blue2	blue3	blue4
blueviolet	brown	brown1	brown2	brown3
brown4	burlywood	burlywood1	burlywood2	burlywood3
burlywood4	cadetblue	cadetblue1	cadetblue2	cadetblue3
cadetblue4	chartreuse	chartreuse1	chartreuse2	chartreuse3
chartreuse4	chocolate	chocolate1	chocolate2	chocolate3
chocolate4	coral	coral1	coral2	coral3
coral4	cornflowerblue	cornsilk	cornsilk1	cornsilk2
cornsilk3	cornsilk4	cyan	cyan1	cyan2
cyan3	cyan4	darkblue	darkcyan	darkgoldenrod
darkgoldenrod1	darkgoldenrod2	darkgoldenrod3	darkgoldenrod4	darkgray
darkgreen	darkgrey	darkkhaki	darkmagenta	darkolivegreen
darkolivegreen1	darkolivegreen2	darkolivegreen3	darkolivegreen4	darkorange
darkorange1	darkorange2	darkorange3	darkorange4	darkorchid
darkorchid1	darkorchid2	darkorchid3	darkorchid4	darkred
darksalmon	darkseagreen	darkseagreen1	darkseagreen2	darkseagreen3
darkseagreen4	darkslateblue	darkslategray	darkslategray1	darkslategray2
darkslategray3	darkslategray4	darkslategrey	darkturquoise	darkviolet
deeppink	deeppink1	deeppink2	deeppink3	deeppink4
deepskyblue	deepskyblue1	deepskyblue2	deepskyblue3	deepskyblue4

Use factor for color

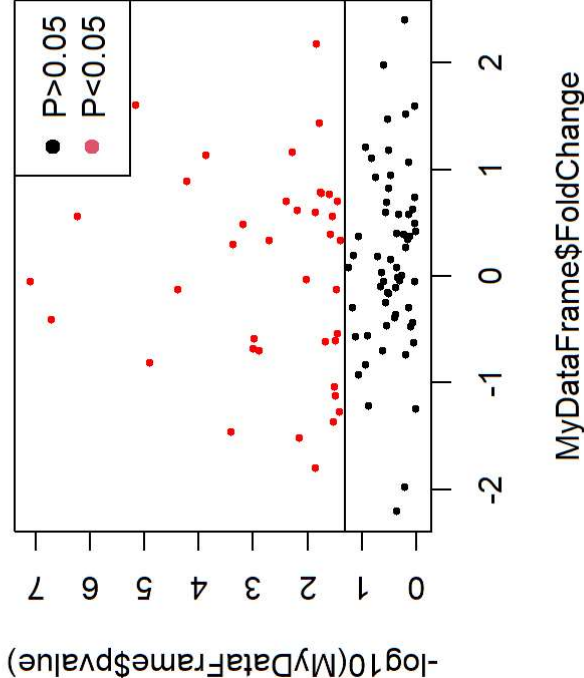
You can use factors to define your color

```
head(MyDataFrame)
```

```
##      Genes  FoldChange      pvalue
## 1 Gene_1 -0.6264538 8.960140e-01
## 2 Gene_2  0.1836433 1.934082e-01
## 3 Gene_3 -0.8356286 1.145058e-01
## 4 Gene_4  1.5952808 6.642649e-06
## 5 Gene_5  0.3295078 1.970055e-03
## 6 Gene_6 -0.8204684 1.249134e-05
```

```
pLot(MyDataFrame$FoldChange, -log10(MyDataFrame$pvalue),
      pch=19, cex=.5,
      col=c("black", "red"))[(MyDataFrame$pvalue < 0.05) +1])
abline(h=-log10(0.05))

legend("topright", legend = c("P>0.05", "P<0.05"),
      pch = 19, col = 1:2,)
```



Options for plots

before plot:

- Add e.g. `par(mfrow=c(2,1))` to have 2 plot (2 row, 1 column)
- Add e.g. `par(mar=c(5.1,4.1,4.1,2.1))` to change margin

plot option:

- `type="p"` or `type="l"` for points for lines
- `xlim=c()`, `ylim=c()` to set limits in the plots
- `pch` for the type of point you want
- `col` for color
- `lty` for linetype
- `lwd` for line width
- `main` for title
- `ylab` and `xlab` for axis name

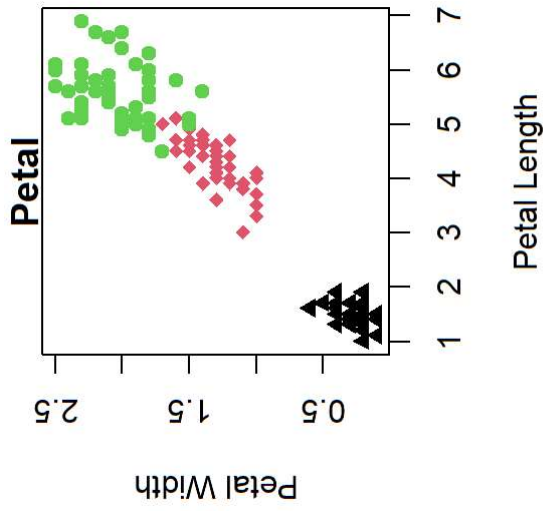
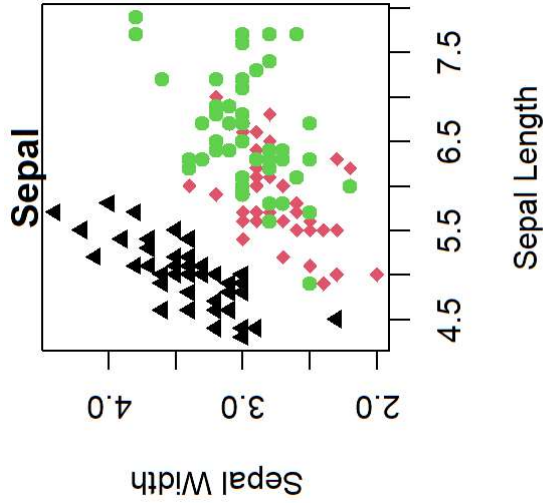
after plot:

- `abline(a,b)` to add some lines (using $y=ax+b$)
- `points()` to add some points
- `lines()` to add some lines

Example

```
data(iris)
par(mar=c(5,5,1,1))
par(mfrow=c(1,2))
plot(iris$Sepal.Length, iris$Sepal.Width, col=as.factor(iris$Species),
     pch=as.numeric(as.factor(iris$Species))+16, xlab="Sepal Length",
     ylab="Sepal Width", main="Sepal")

plot(iris$Petal.Length, iris$Petal.Width, col=as.factor(iris$Species),
     pch=as.numeric(as.factor(iris$Species))+16, xlab="Petal Length",
     ylab="Petal Width", main="Petal")
```



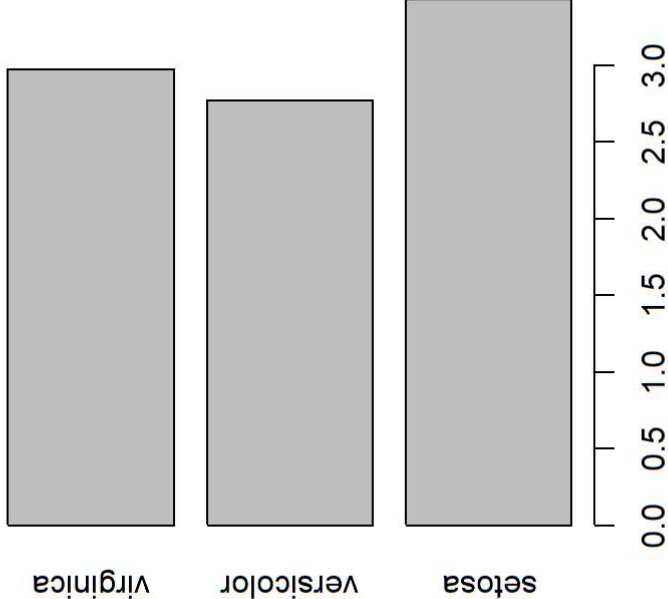
Barplot

```
iris_mean_sepal_width<-aggregate(  
  iris$Sepal.Width,list(iris$Species),  
  mean)
```

```
iris_mean_sepal_width
```

```
##      Group.1      x  
## 1   setosa 3.428  
## 2 versicolor 2.770  
## 3  virginica 2.974
```

```
barplot(iris_mean_sepal_width$x,  
        horiz=TRUE,  
        names.arg = iris_mean_sepal_width$Group.1)
```

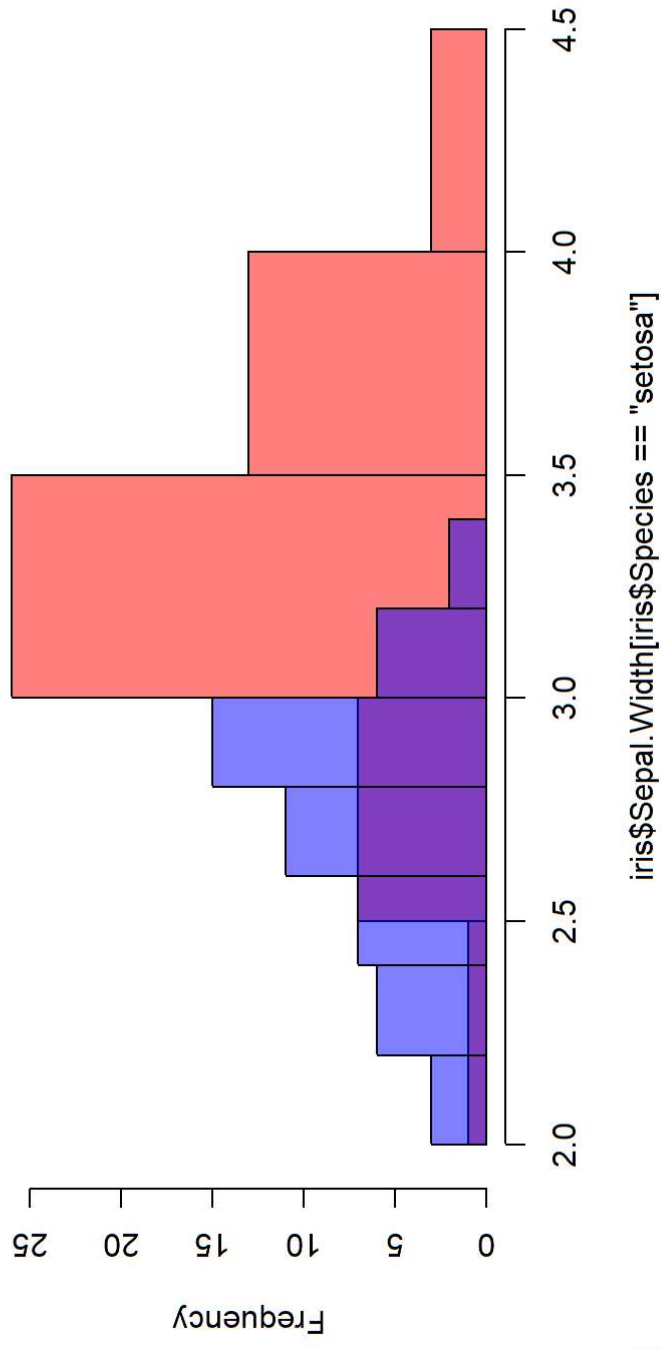


Histogram

you can use the library scales to add some transparency to color using *alpha()*

```
library(scales)
hist(iris$Sepal.Width[iris$Species == "setosa"], col=alpha("red",.5))
hist(iris$Sepal.Width[iris$Species == "versicolor"], add=T, col=alpha("blue",.5))
```

Histogram of iris\$Sepal.Width[iris\$Species == "setosa"]



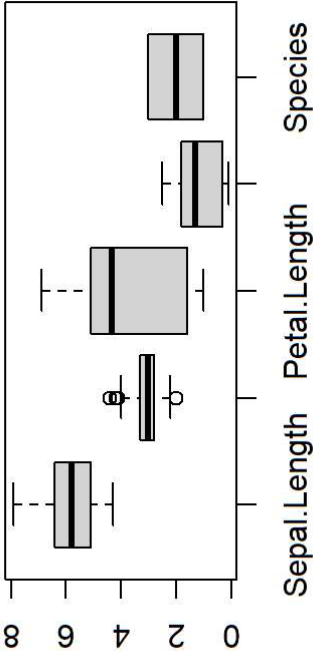
boxplot

You can simply create boxplot with the *boxplot()* function. It will create a boxplot per column in your data frame

```
head(iris,n=3)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa

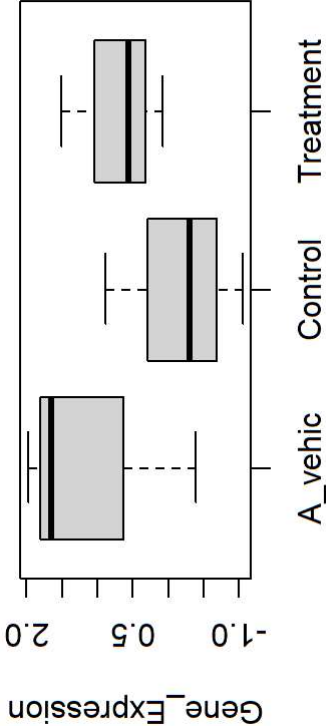
```
par(mar=c(3,4,1,1))
boxplot(iris)
```



Or you can use the linear model annotation

##	Gene_Expression	Treatment	Batch
## 1	-0.3058154	Control	B1
## 2	0.8936737	Control	B2
## 3	-1.0472981	Control	B3
## 4	1.9713374	A_vehic	B1

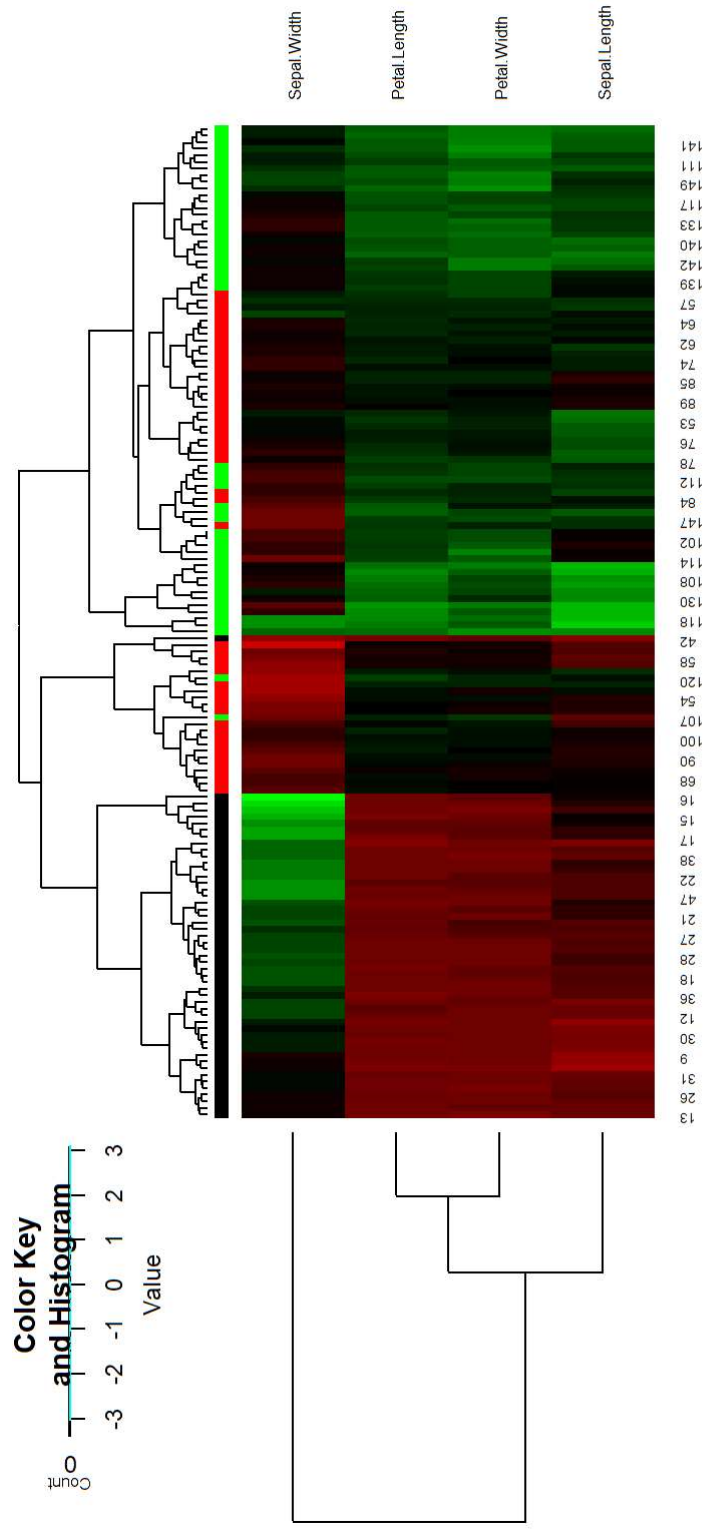
```
par(mar=c(3,4,1,1))
boxplot(Gene_Expression~Treatment,data=MyMetadata)
```



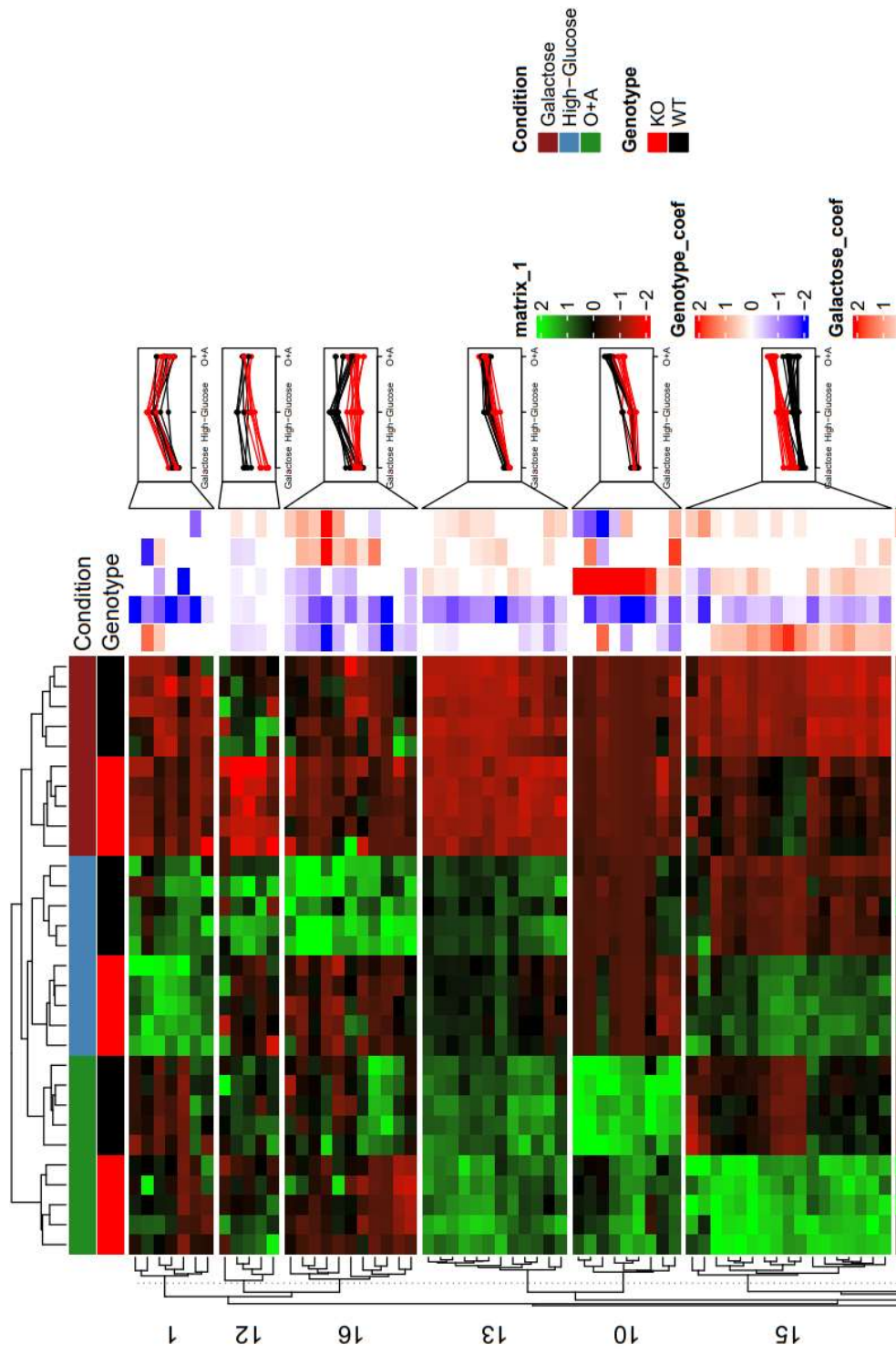
Heatmaps

Using gplots package, we can plot heatmap and row color

```
iris_scaled<-apply(iris[,1:4],2,scale)
heatmap.2(t(iris_scaled),col=redgreen(75),trace="none",
          ColSideColors = c("black", "red", "green"),[factor(iris$Species)],
          cexRow = .8,)
```



Even more complex with ComplexHeatmap

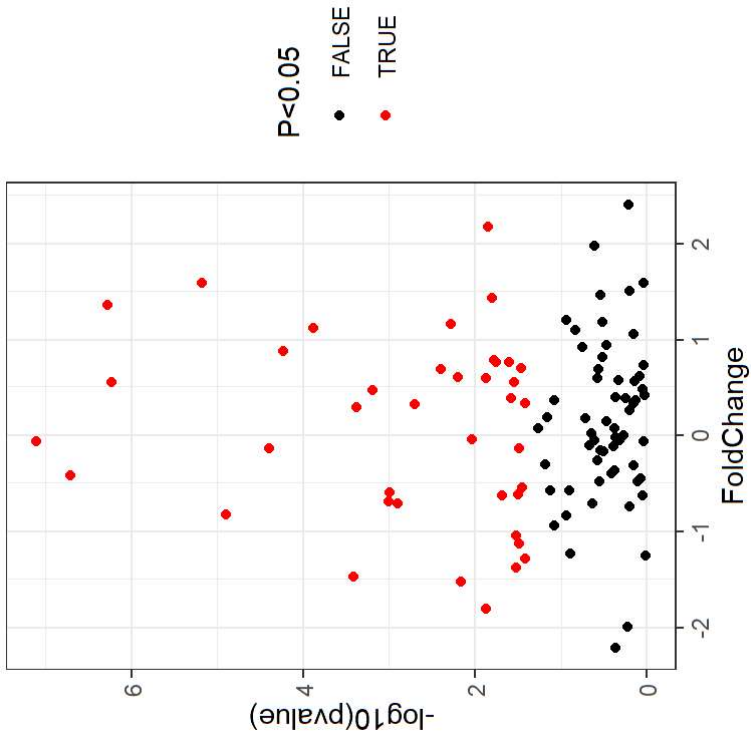


ggplot2

Another way to have nicer plot, is to use the ggplot2 package:

```
library(ggplot2)

ggplot(aes(FoldChange, -log10(pvalue), color=as.factor(pvalue<0.05)), data=MyDataFrame)+
  geom_point()+
  scale_color_manual("P<0.05", values=c("FALSE"="black", "TRUE"="red"))+
  theme_bw()
```



What you need to make ggplot2

- data.frame of your data

```
head(MyDataFrame)
```

```
##      Genes  FoldChange      pvalue
## 1 Gene_1 -0.6264538 8.960140e-01
## 2 Gene_2  0.1836433 1.934082e-01
## 3 Gene_3 -0.8356286 1.145058e-01
## 4 Gene_4  1.5952808 6.642649e-06
## 5 Gene_5  0.3295078 1.970055e-03
## 6 Gene_6 -0.8204684 1.249134e-05
```

- Specify easthetic mappings: (which columns of your data.frame will be used for what)
 - You can specify: x,y axes (position)
 - color
 - fill
 - groups
 - shape
 - size
 - transparency


```
MyPlot <- ggplot(aes(FoldChange, -log10(pvalue), color=as.factor(pvalue<0.05)), data=MyDataFrame)
```

- Then you add some layers (how data should be represented)
 - it can be point: `geom_point()`
 - lines: `geom_lines()`
 - histogram, polygon, boxplot, violin_plot etc...

```
MyPlot <- MyPlot + geom_point()
```

- Then you can rescale everything, here just for color:
 - `scale_color_manuals()`
 - `scale_color_continuous()`
 - `scale_color_gradientn()`, but see also `scale_color_gradient2()` and `scale_color_gradient()`
 - `scale_colour_brewer(palette = "Set1")`
 - `scale_color_viridis_c()`

Here we rescaled color

```
MyPlot <- MyPlot + scale_color_manual(values=c("FALSE"="black", "TRUE"="red"))
```

Finally we apply a theme:

```
MyPlot <- MyPlot + theme_bw()
```

Some tips:

- Combine plots using package: patchwork
- `+ylim()` and `+xlim()` sometimes will remove value, you can also use `coord_cartesian()`
- You can use `+facet_grid(~yourgroup)` to plot multiple groups
- for labels, you can use labs:
- Add some lines using `geom_hline`, `geom_vline`
- Add some text using ggrepel package for non-overlapping text
- With theme, you can change font size
- Use “annotate()” to add extra point or lines not present in your original data

```

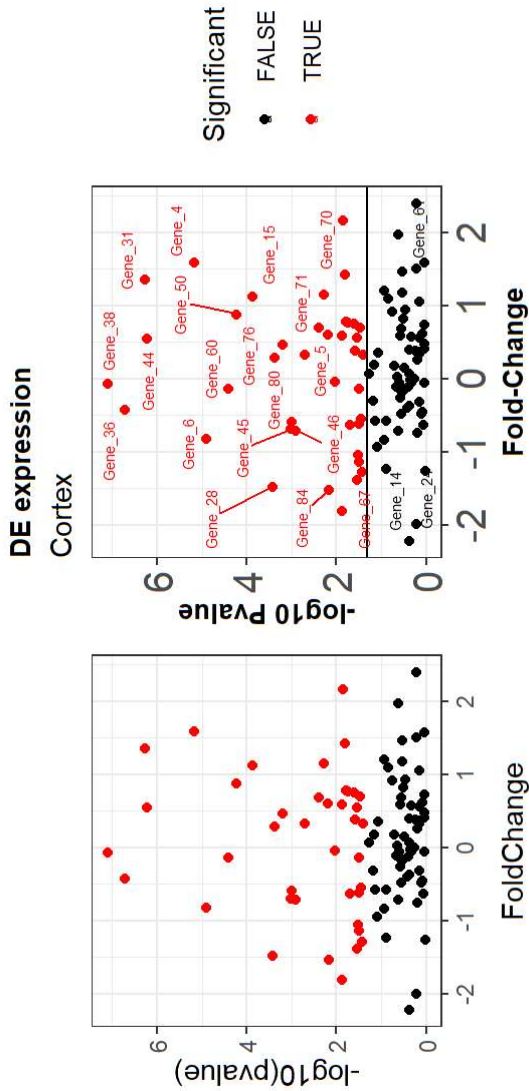
(MyPlot + theme(legend.position = "none")) | # / tell patchwork to plot side by side plot

MyPlot + labs(title = "DE expression", subtitle = "Cortex", x = "Fold-Change", y = "-log10 Pvalue", color = "Significant") +
  geom_hline(yintercept = -log10(0.05)) + geom_text_repel(aes(label=Genes), size=2) +
  theme( plot.title=element_text(size=10, face="bold"),
        axis.text=element_text(size=15),
        axis.title=element_text(size=10, face="bold"))

```

warning: ggrepel: 78 unlabeled data points (too many overlaps). Consider

increasing max.overlaps



Example of combined plots

