Venn/Euler Diagrams

Joann Mudge

January 5, 2022

1 Venn and Euler Diagrams

1.1 Set up

- Create and navigate to your working directory.
  - Call it "venn"
- Make a symbolic link to the data files
  - ln -s /home/jm/VisualizationData_venn/* .
- Don't forget to activate your environment
  - We needed an updated version of R so we'll use the venn2 environment
    - source activate visualization
- Open R.

1.2 Load library

There are lots of packages that do venn and euler diagrams in R. We'll use eulerr.

- Check out some of the eulerr vignettes at https://cran.r-project.org/web/packages/eulerr/index.html.
- There is also a web implementation of eulerr at http://jolars.co/

```r
library(eulerr)
```

1.3 Load data

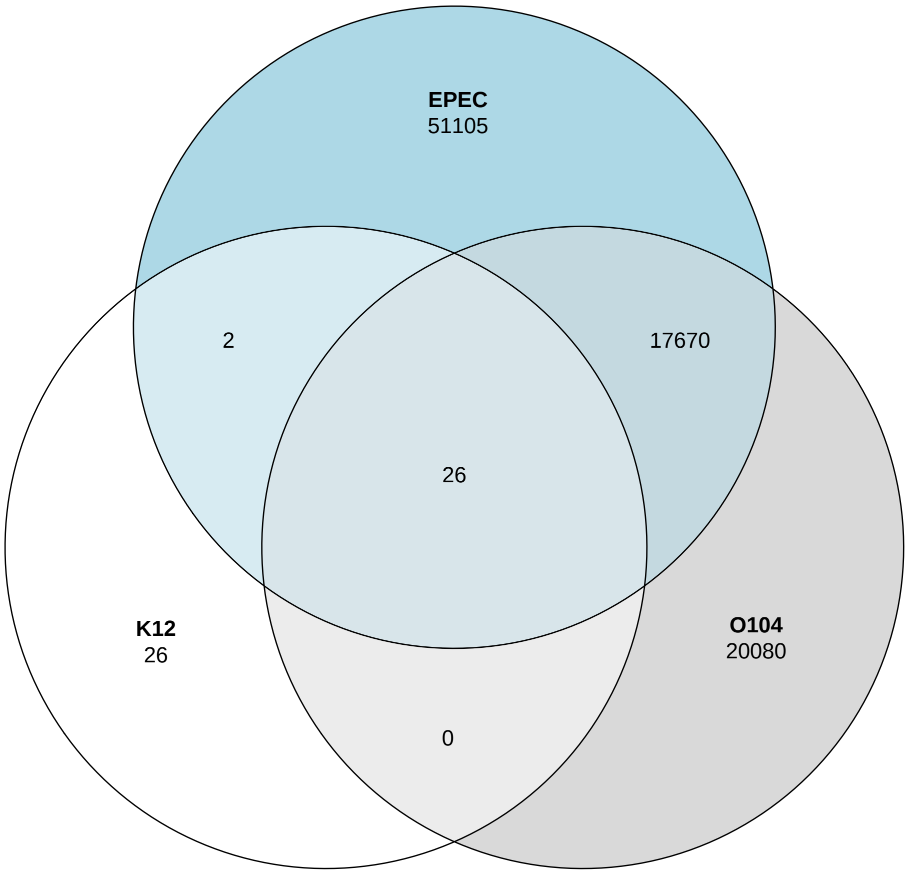
We'll use the variants we called on the three *E. coli* samples to see which differences from the reference (alternate alleles) are shared among strains. The data you linked to above is my data. Later, you'll use your own data.

```
gt3 = read.table("3samples.gt.txt",header=TRUE)

# Take a look
head(gt3)

##   K12 O104 EPEC
## 1   0    1    1
## 2   0    0    1
## 3   0    0    1
## 4   0    1    0
## 5   0    1    0
## 6   0    1    1
```

1.4 Make a venn diagram

A venn diagram is not area proportional and it requires that all set interactions (overlaps) are present even if the count is zero. By default, the venn function plots the quantities because you cannot infer them from the area porportions.

```
pdf("ecoli_venn.pdf")

myvenn = venn(gt3)
plot(myvenn)

dev.off()

## pdf
##   2
```

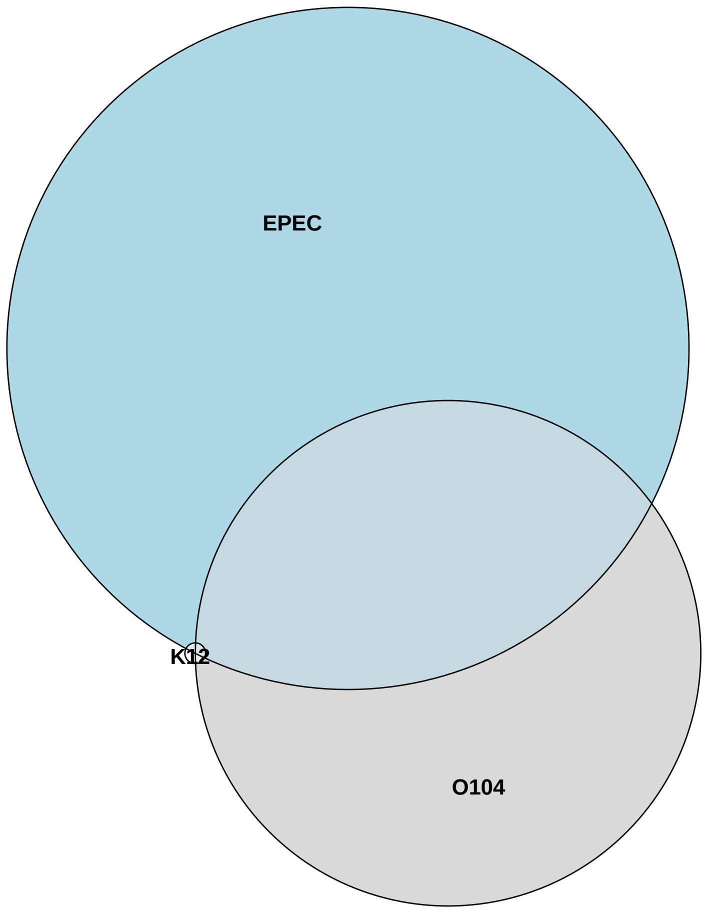


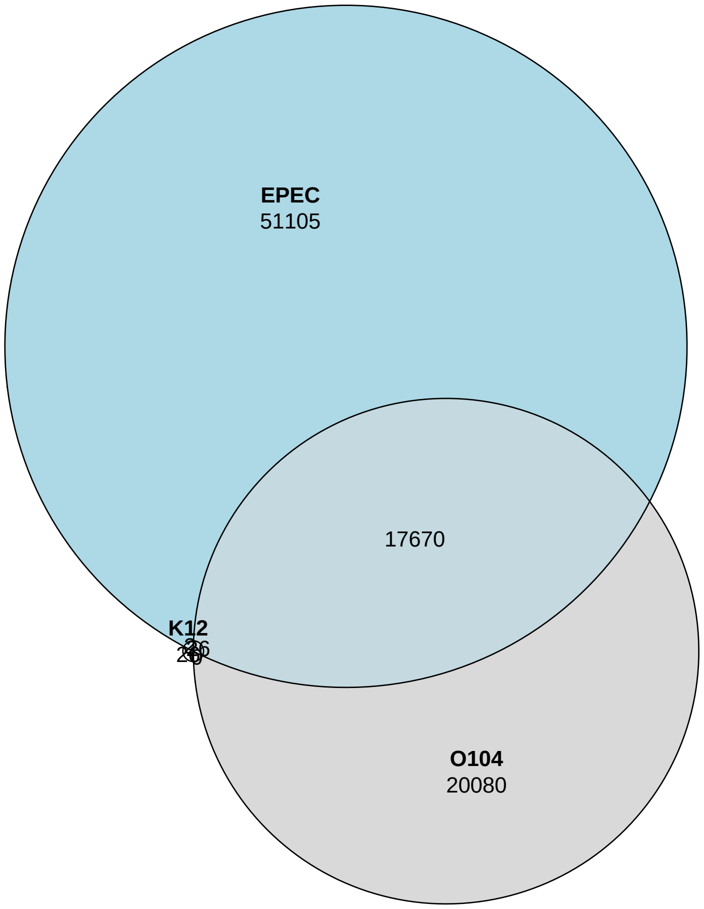
1.5 Make an euler diagram

An euler diagram is area proportional. By default the quantities are turned off but you can add them.

```
pdf("ecoli_euler.pdf")

# No quantities
myeuler = euler(gt3)
plot(myeuler)

# Quantities
plot(myeuler, quantities = TRUE)

dev.off()
```

```
## pdf
##   2
```





1.6 Goodness of fit

Euler diagrams try to represent the areas proportionally but there are limitations. Some datasets work well and others do not. You should check how good your fit is because if the fit is not very good, the areas will be off and the diagram will be misleading.

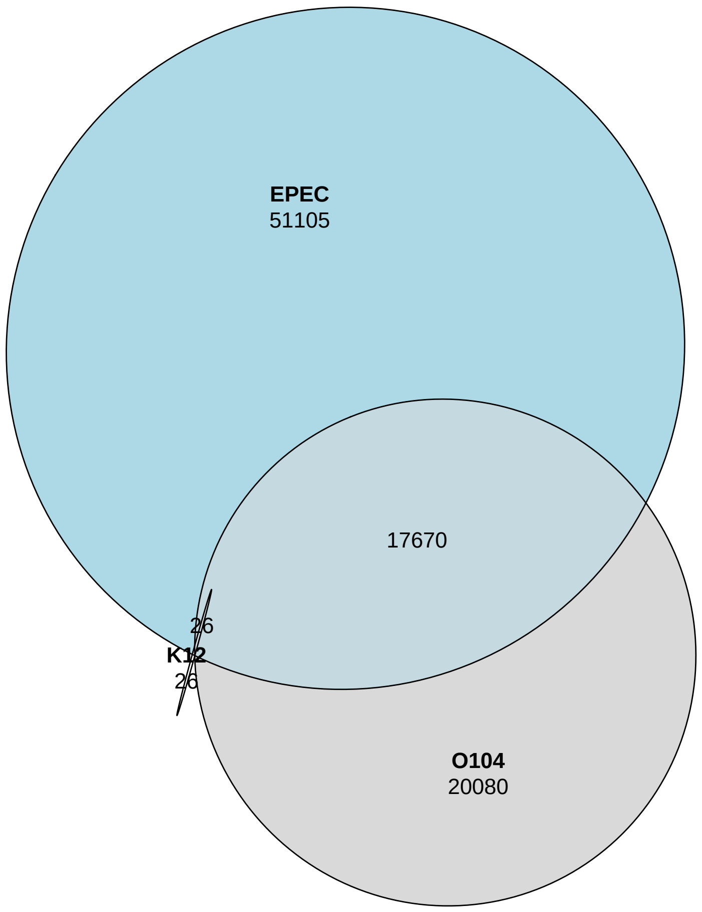
```
# When you create the euler object, it calculates fit

# Create the object
myeuler = euler(gt3)

# Look at the object
myeuler

##                original    fitted residuals regionError
## K12                  26    20.642     5.358           0
## O104              20080 20079.995     0.005           0
## EPEC              51105 51104.998     0.002           0
## K12&O104              0    10.561   -10.561           0
## K12&EPEC              2    11.732    -9.732           0
## O104&EPEC         17670 17670.008    -0.008           0
## K12&O104&EPEC        26    20.341     5.659           0
##
## diagError: 0
## stress:    0
```

By looking at the original area, the fitted area, and the residuals (the difference between the two), you can get a pretty good idea of how well it was able to do. In our euler diagram, it did really well for EPEC and O104 but any of the intersections that contain K12 are way off because the K12 circle is so small.

1.7 Ellipse

So, let's try using ellipses, instead of circles, which are less constrained and usually do a better fit.

```
pdf("ecoli_euler_ellipse.pdf")

myeuler2 = euler(gt3,shape="ellipse")
plot(myeuler2, quantities=TRUE)

# Look at the fit
myeuler2

##                 original    fitted residuals regionError
## K12                   26    25.679     0.321           0
## O104               20080 20080.021    -0.021           0
## EPEC               51105 51105.027    -0.027           0
## K12&O104               0     0.006    -0.006           0
## K12&EPEC               2     2.184    -0.184           0
## O104&EPEC          17670 17669.981     0.019           0
## K12&O104&EPEC         26    26.271    -0.271           0
##
## diagError: 0
## stress:    0

dev.off()

## pdf
##   2
```



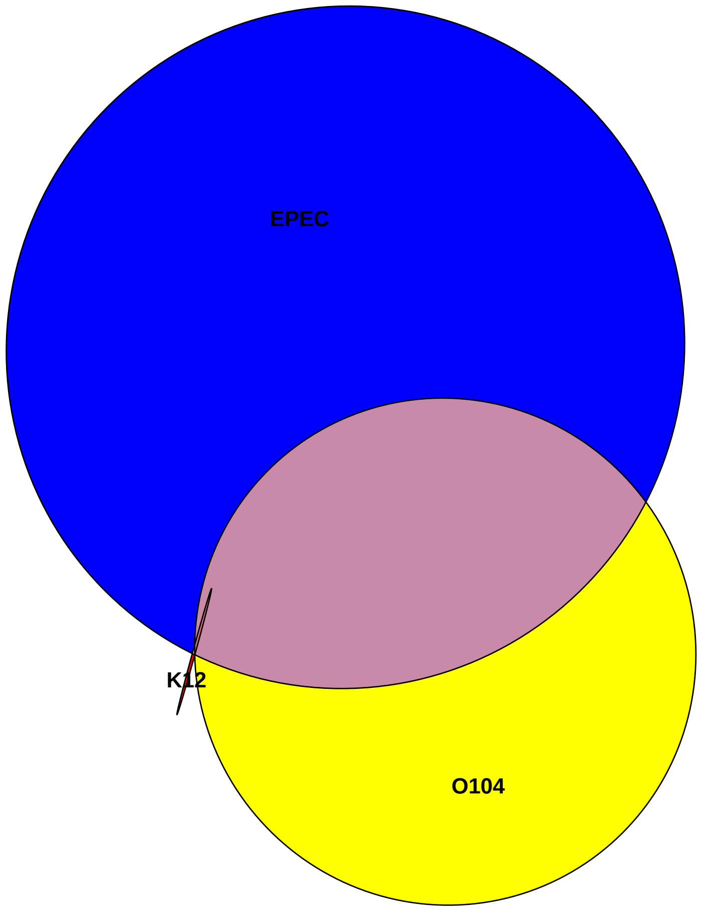
```
##                original    fitted residuals regionError
## K12                  26    25.679     0.321           0
## O104              20080 20080.021    -0.021           0
## EPEC              51105 51105.027    -0.027           0
## K12&O104              0     0.006    -0.006           0
## K12&EPEC              2     2.184    -0.184           0
## O104&EPEC         17670 17669.981     0.019           0
## K12&O104&EPEC        26    26.271    -0.271           0
##
## diagError: 0
## stress:    0
```

Much better fit!

1.8 Colors

It's easy to customize the the plots. We can change the colors.

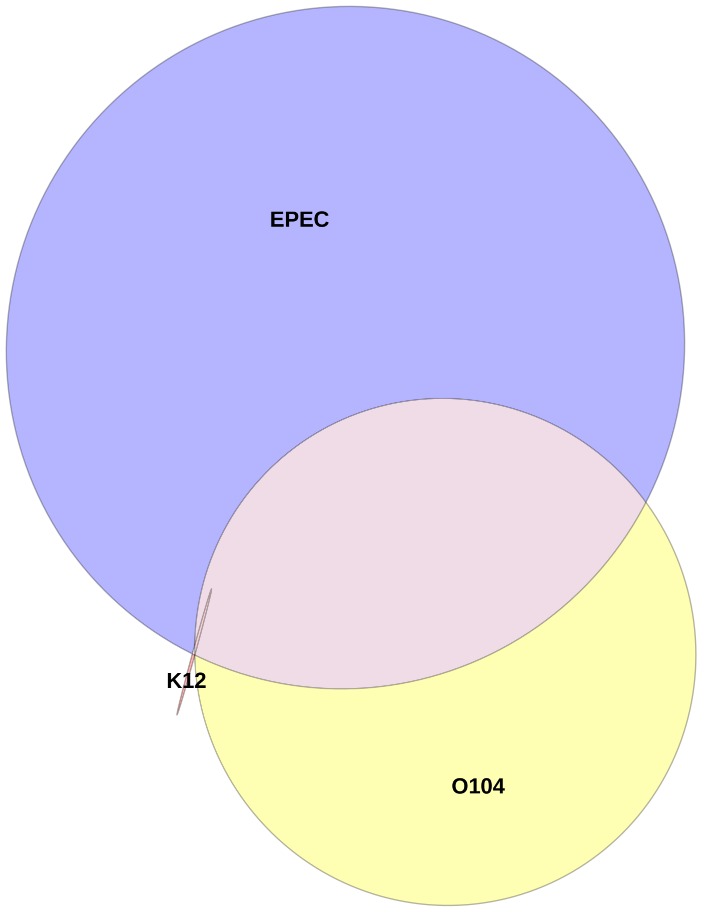
```
pdf("ecoli_euler_color.pdf")

plot(myeuler2,fills = c("red", "yellow", "blue"))

dev.off()

## pdf
##   2
```



A bit garrish! Let's increase the transparency.

```
pdf("ecoli_euler_color_transparent.pdf")

plot(myeuler2,fills = c("red", "yellow", "blue"),alpha=0.3)

dev.off()
```

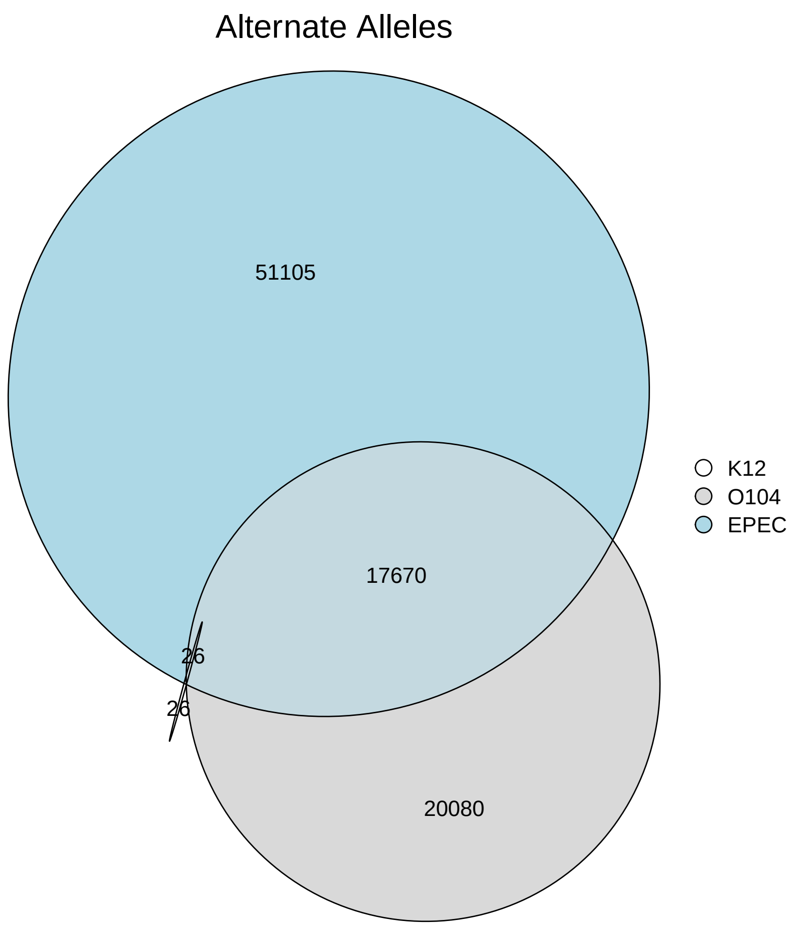
```
## pdf
##   2
```



1.9 Legends and Title

We can also add a legend and a title. I'm using default colors here.

```
pdf("ecoli_euler_ellipse_legend_title.pdf")

plot(myeuler2, quantities=TRUE,legend=list, main="Alternate Alleles")

dev.off()

## pdf
##   2
```



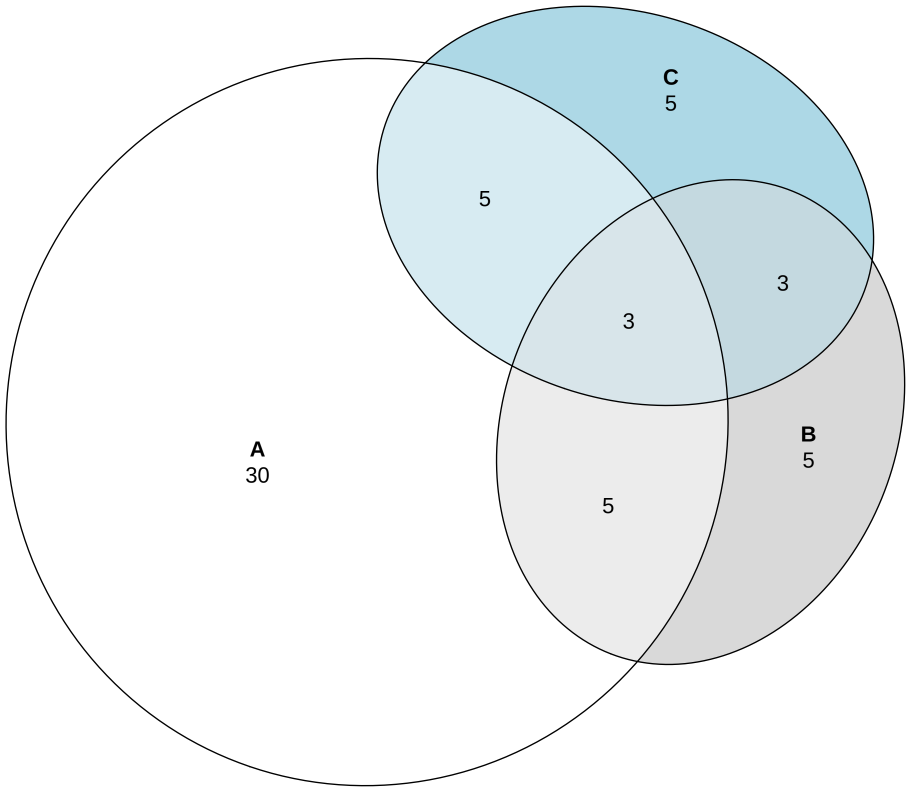
1.10 Summarized Data

You can also use data that has already been summarized rather than using a TRUE/FALSE or 1/0 matrix. You can simply tell the euler function what quantities you want for each of the sections of the diagram.

```
pdf("ecoli_euler_summ.pdf")

myeuler3 = euler(c("A" = 30, "B" = 5, "C" = 5, "A&B" = 5,
        "A&C" = 5, "B&C" = 3, "A&B&C" = 3),shape="ellipse")
plot(myeuler3,quantities=TRUE)

# Check the fit
myeuler3

##         original fitted residuals regionError
## A           30    30         0           0
## B            5     5         0           0
## C            5     5         0           0
## A&B          5     5         0           0
## A&C          5     5         0           0
## B&C          3     3         0           0
## A&B&C        3     3         0           0
##
## diagError: 0
## stress:    0
```

```
dev.off()
```

```
## pdf
##   2
```

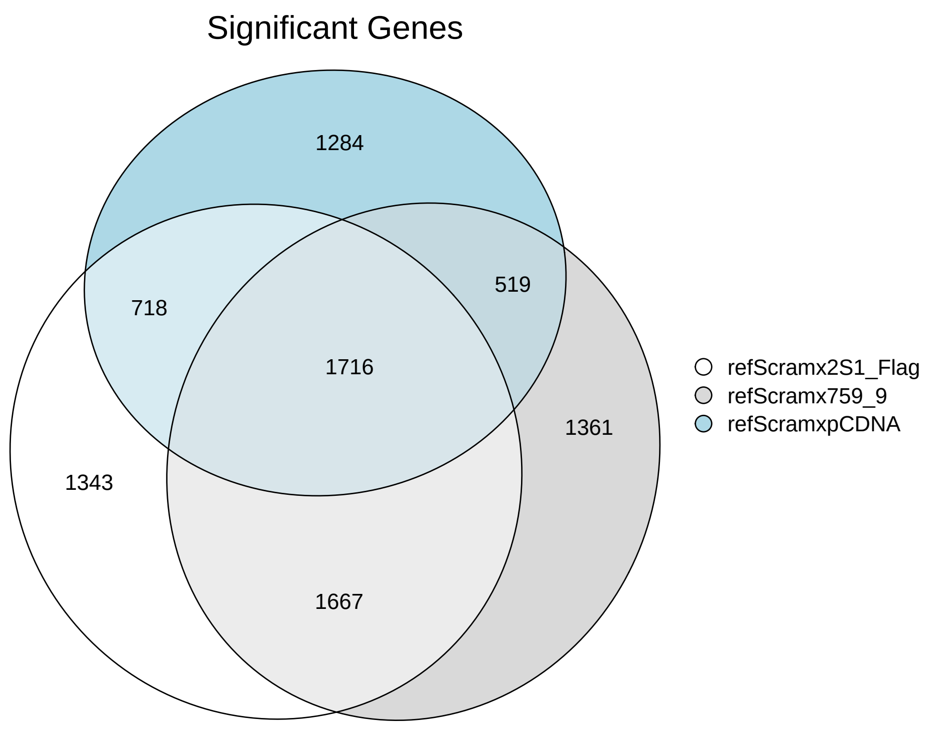


```
##        original fitted residuals regionError
## A            30     30         0           0
## B             5      5         0           0
## C             5      5         0           0
## A&B           5      5         0           0
## A&C           5      5         0           0
## B&C           3      3         0           0
## A&B&C         3      3         0           0
##
## diagError: 0
## stress:    0
```

1.11 Expression data

These are 3 samples that have genes that are significantly (p<=0.05) differentiallly expressed from a reference sample.

```
expr_data = read.table("significant_genes.txt",header=TRUE)

head(expr_data)

##   refScramx2S1_Flag refScramx759_9 refScramxpCDNA
## 1                 1              1              0
## 2                 1              0              1
## 3                 1              1              0
## 4                 1              1              0
## 5                 0              0              1
## 6                 1              1              1

pdf("expression_euler.pdf")

my_expr_euler = euler (expr_data,shape="ellipse")
plot(my_expr_euler, quantities=TRUE, legend=list, main="Significant Genes")

dev.off()

## pdf
##   2

# Check the fit
my_expr_euler

##                                                      original fitted residuals
## refScramx2S1_Flag                                        1343   1343         0
## refScramx759_9                                           1361   1361         0
## refScramxpCDNA                                           1284   1284         0
## refScramx2S1_Flag&refScramx759_9                1667   1667          0
## refScramx2S1_Flag&refScramxpCDNA                 718    718          0
## refScramx759_9&refScramxpCDNA                    519    519          0
## refScramx2S1_Flag&refScramx759_9&refScramxpCDNA  1716   1716          0
##                                                regionError
## refScramx2S1_Flag                                        0
## refScramx759_9                                           0
## refScramxpCDNA                                           0
## refScramx2S1_Flag&refScramx759_9                         0
## refScramx2S1_Flag&refScramxpCDNA                         0
## refScramx759_9&refScramxpCDNA                            0
## refScramx2S1_Flag&refScramx759_9&refScramxpCDNA          0
##
## diagError: 0
## stress:    0
```



```
##                                                  original fitted residuals
## refScramx2S1_Flag                                    1343   1343         0
## refScramx759_9                                       1361   1361         0
## refScramxpCDNA                                       1284   1284         0
## refScramx2S1_Flag&refScramx759_9                     1667   1667         0
## refScramx2S1_Flag&refScramxpCDNA                      718    718         0
## refScramx759_9&refScramxpCDNA                         519    519         0
## refScramx2S1_Flag&refScramx759_9&refScramxpCDNA      1716   1716         0
##                                                  regionError
## refScramx2S1_Flag                                          0
## refScramx759_9                                             0
## refScramxpCDNA                                             0
## refScramx2S1_Flag&refScramx759_9                           0
## refScramx2S1_Flag&refScramxpCDNA                           0
## refScramx759_9&refScramxpCDNA                              0
## refScramx2S1_Flag&refScramx759_9&refScramxpCDNA            0
##
## diagError: 0
## stress:    0
```

If we want to get the genes that are differentially expressed compared to the reference in each of the three samples (ie. the middle of the venn), we can do that using the dplyr library.

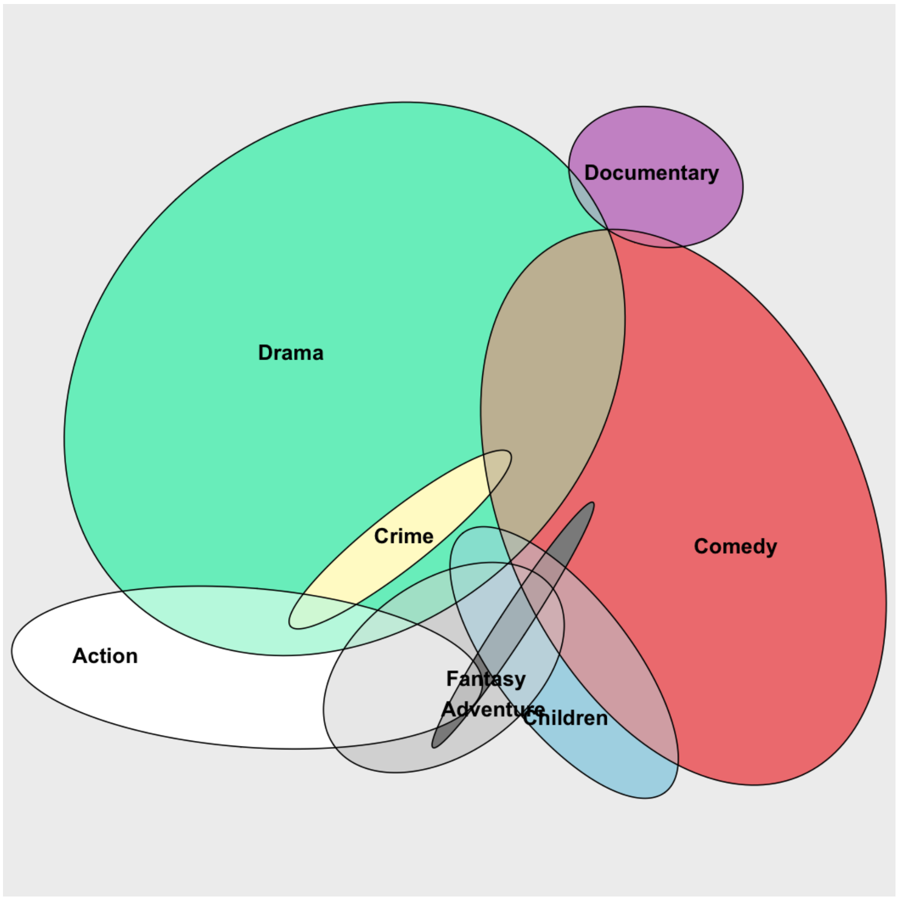
```
library(dplyr)

##
## Attaching package:  'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

diff_all = filter(expr_data, refScramx2S1_Flag==1 & refScramx759_9==1 & refScramxpCDNA==1)

sink("alldiff.txt")
diff_all
sink()
```

Try to get other subsets.

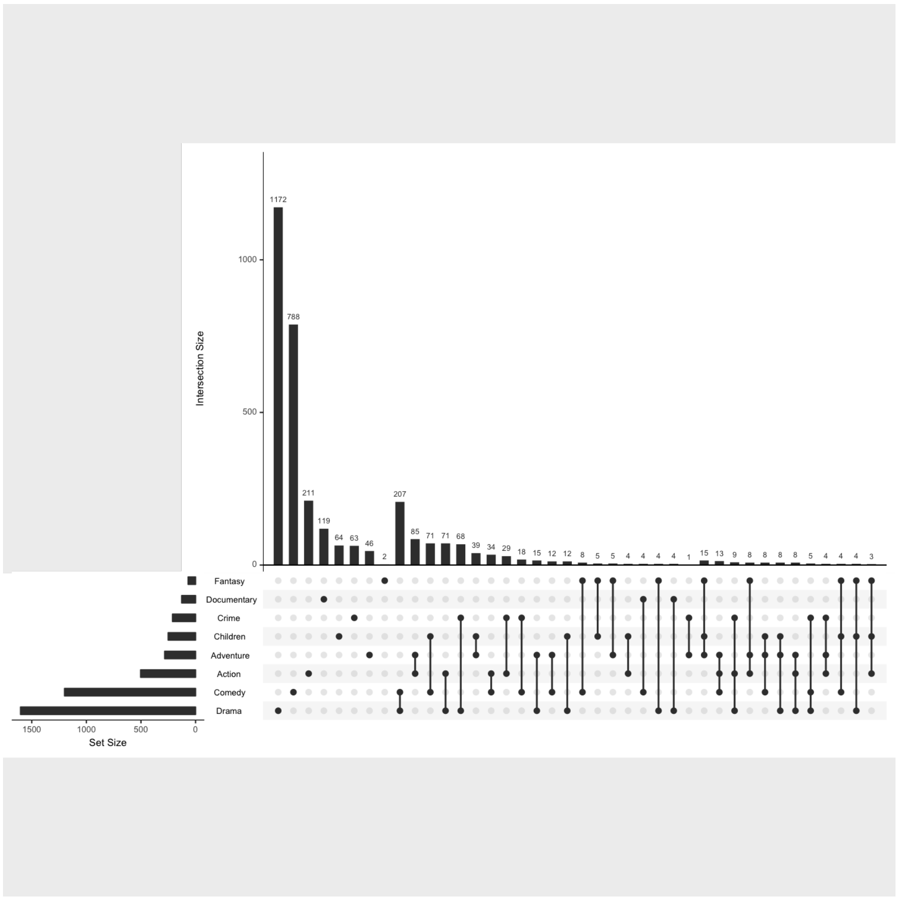
2 UpSet Plots for Complicated Data

You need to be realistic about whether it makes sense to make an euler or venn diagram from your data. When you can't find a good fit for your data, a venn or euler diagram will be misleading. Search for an alternative way to present your data. Here, we'll use an UpSet plot using the UpSetR R library.

For instance, this euler diagram shows the overlap of movie genres (ie. a single film can be classified in multiple genres). Given the number of categories in this diagram, it would be a miracle if you got a good fit.

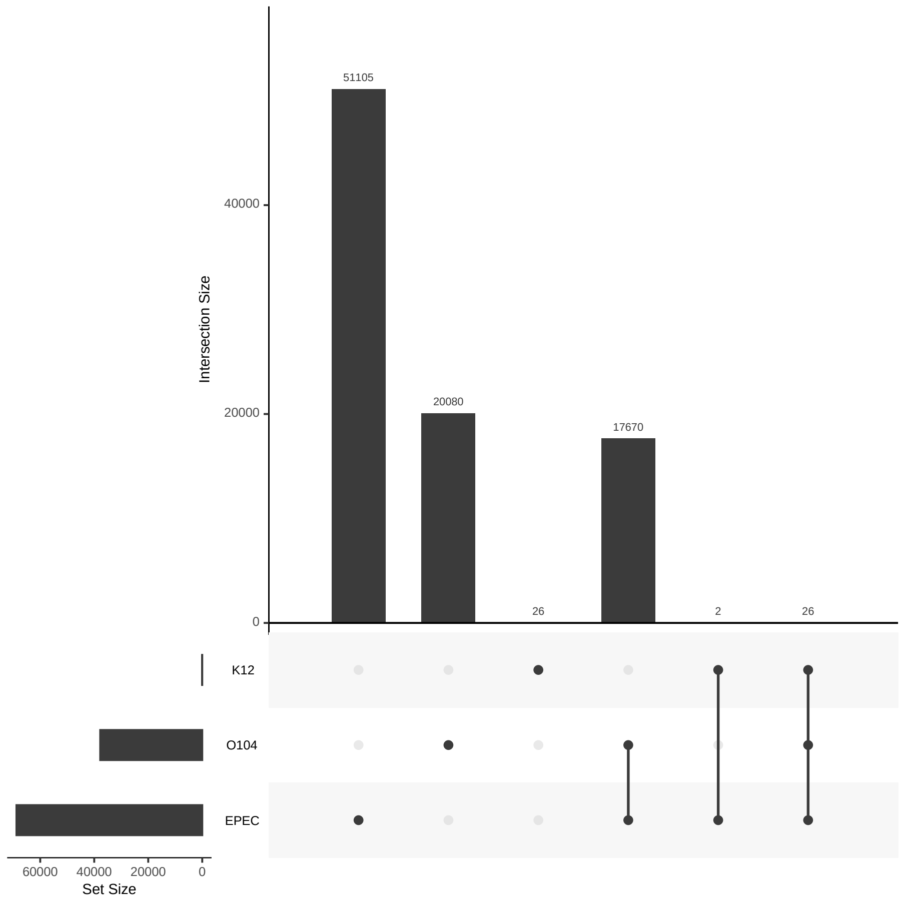


And, indeed, the fit isn't good. For instance, there were 34 films that were classified as both action and comedy (but in no other genres), but the diagram shows 0 films. Here is an alternate way of showing the data using an UpSet plot, which captures the overlap between action and comedy.

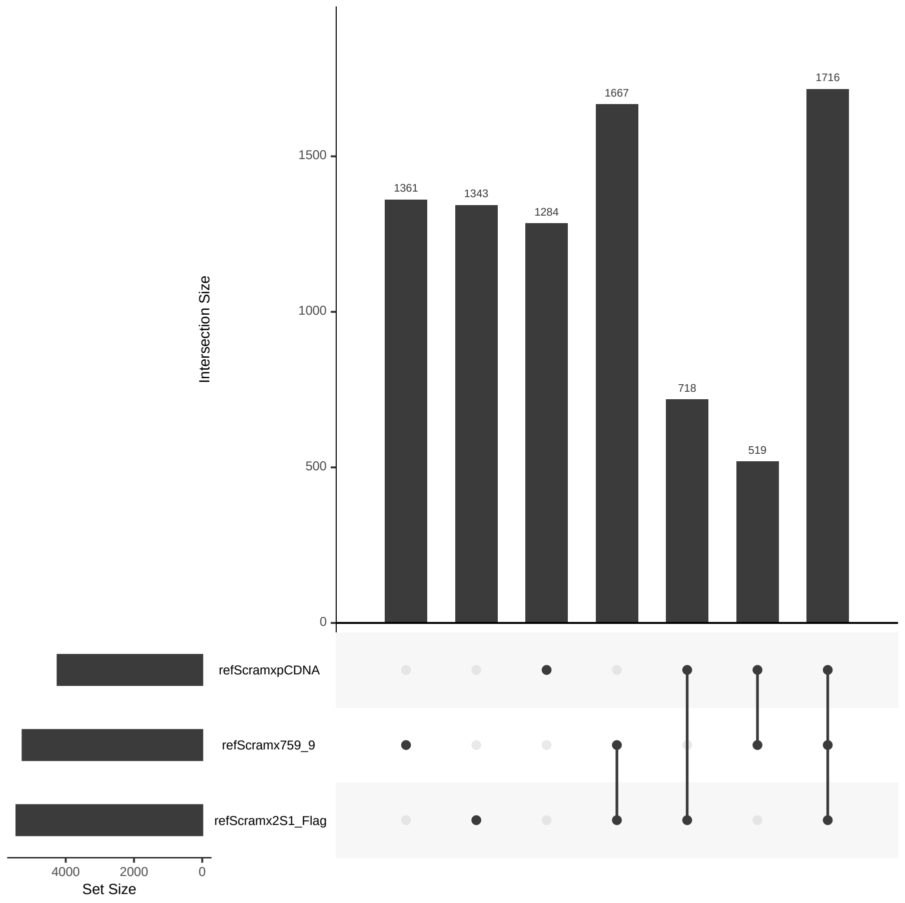


Let's try this with our data, both the variant and the expression data.

```
library(UpSetR)

# Make file
pdf("upsetr.pdf")

# Generate the plotw
upset(gt3)
upset(expr_data)

# Close the file
dev.off()

## pdf
##   2
```



51105
20080
26
17670
2
26
40000
20000
0
Intersection Size
K12
O104
EPEC
60000
40000
20000
0
Set Size



How does this compare to the euler diagram?