Downstream Analysis in R

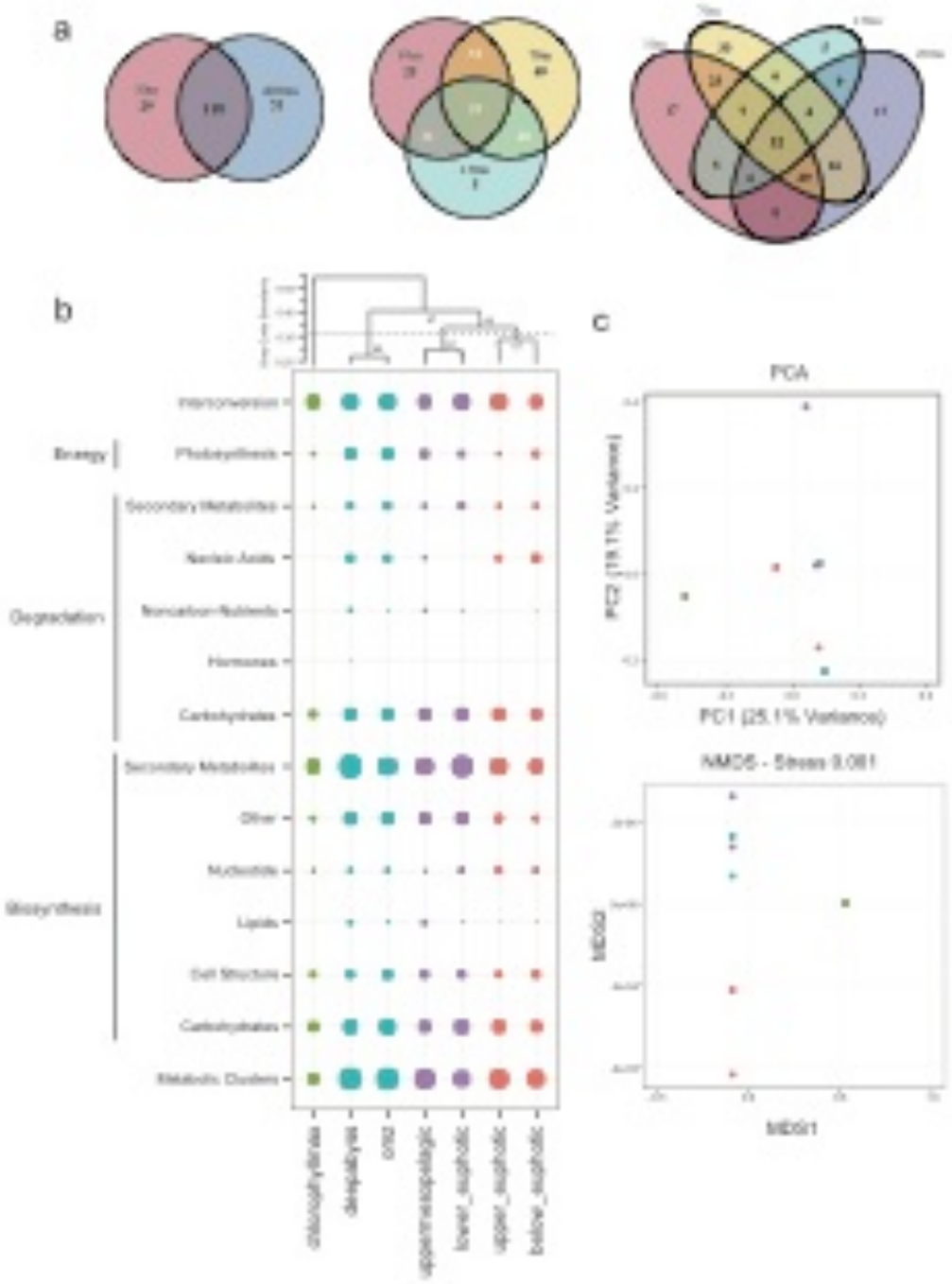
Niels W. Hanson
Ph.D. Candidate Bioinformatics
Wednesday, February 12 2014

Hallam Laboratory
Hydrocarbon MetaPathways Workshop
The University of British Columbia, Vancouver

https://github.com/nielshanson/mp_tutorial/wiki/Introduction-to-Downstream-Analysis-in-R

Goals of Tutorial

- Understand general problems when trying to load tabular formats in R.
- Re-ordering data & Subsetting Data Frames
- Understanding the 'wide' and 'long' table formats
- Using Visualization Frameworks: ggplot2 and lattice


a
b
c
PCA
PC2 (19.1% Variance)
PC1 (25.1% Variance)
NMDS - Stress 0.081
MDS2
MDS1
Energy
Degradation
Biosynthesis
Photosynthesis
Secondary Metabolites
Nucleic Acids
Hormones
Carbohydrates
Other
Nucleotides
Lipids
Cell Structure
Metabolic Clusters
chlorophyllmax
deepchlmax
mix
uppermesopelagic
lower_euphotic
upper_euphotic
below_euphotic

1. Loading Data

- Data should be tab-delimited text

```
pathways_wide <- read.table("../files/HOT_Sanger_pwy.wide.txt", sep="\t",
header=T, row.names=1)
hot_metadata <- read.table("../files/HOT_Sanger_ex_var.csv.txt",
sep="\t", header=T)
```

- Want to check `head()`, `tail()`, and `dim()` for size in case of bad parsing
- Avoid these guys: `;'"\``
- slicing and dicing with brackets [] and logic

```
# slicing and dicing
pathways_wide[1,] # first row
pathways_wide[,1] # first column
pathways_wide$X1_upper_euphotic # first factor
```

- logic operators generates Boolean vector

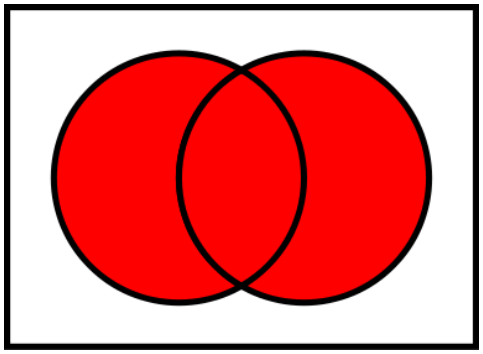
```
pathways_wide[,1] > 1
  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE F
 [14]  TRUE FALSE
```

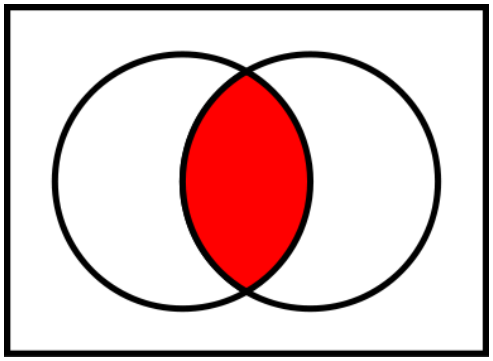
`sum( pathways_wide[,1] > 1 )` gives you the number of TRUE fields

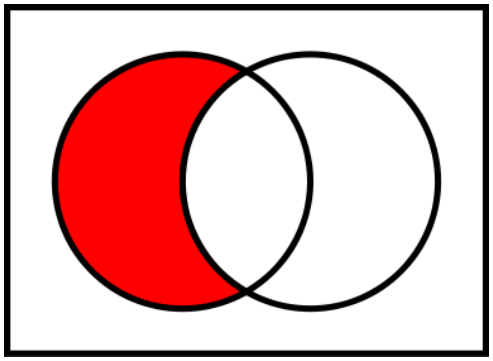
Application: Venn Diagrams

- **venn_diagram2**(), **venn_diagram3**(), **venn_diagram4**()
- created to do all the set-logic between sets and produce a Venn Diagram

e.g., **union**, **intersect**, **setdiff**





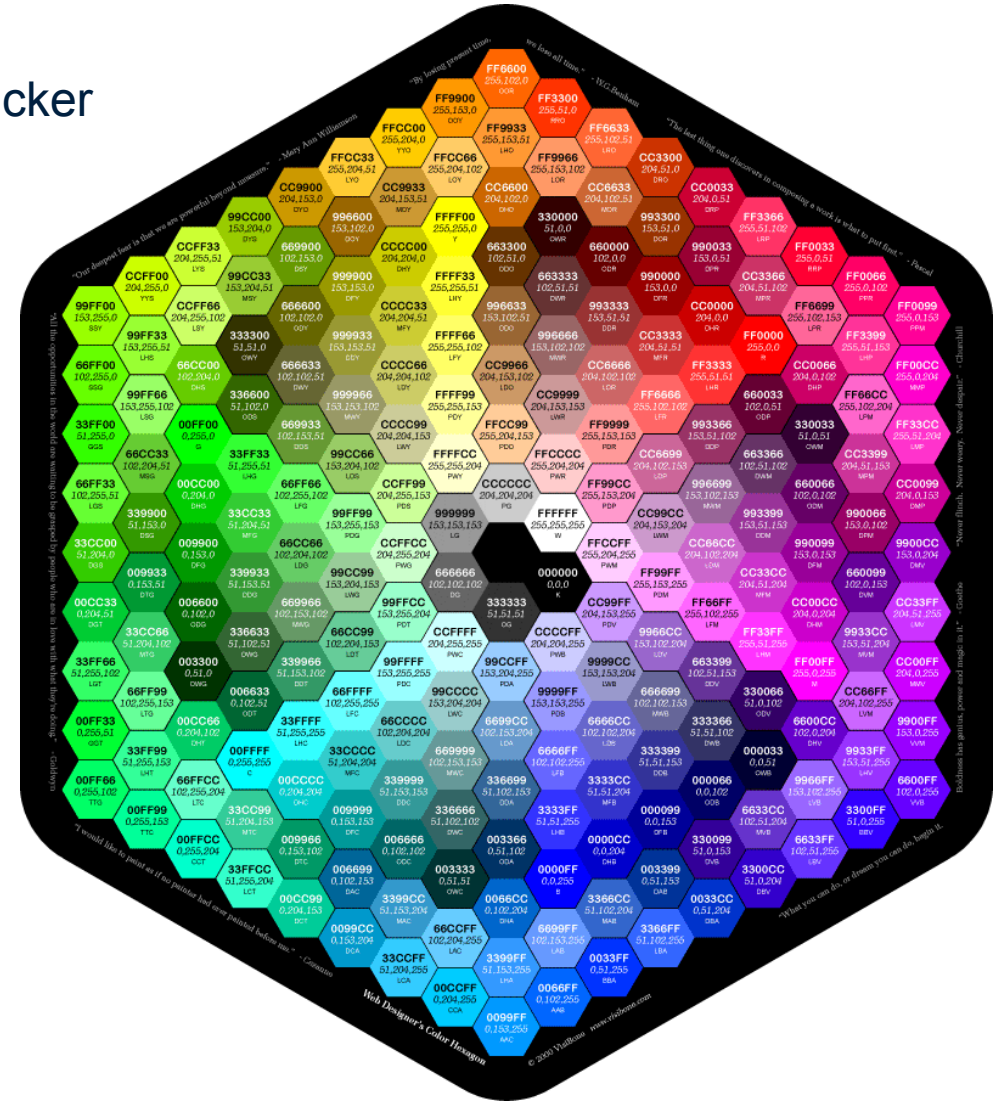


```
venn_diagram3(setA, setB, setC, “name1”, “name2”, “name3”,
                        colors = c("#B80830","#EACC33","#46E2D9")
                       )
```

- combine `c()` function probably one of the most common in R, used to make a list of arguments. Used everywhere.

Application: Venn Diagrams

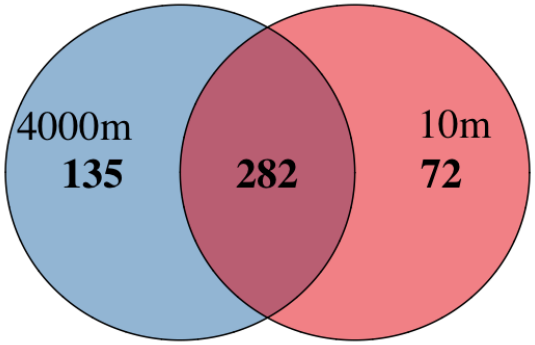
- precise colors specified by six hex digits 0-9, A,B,C,D,F #FFFFFF for amounts of red green and blue. Don't need to memorize, just use for precision.
- Get a color picker

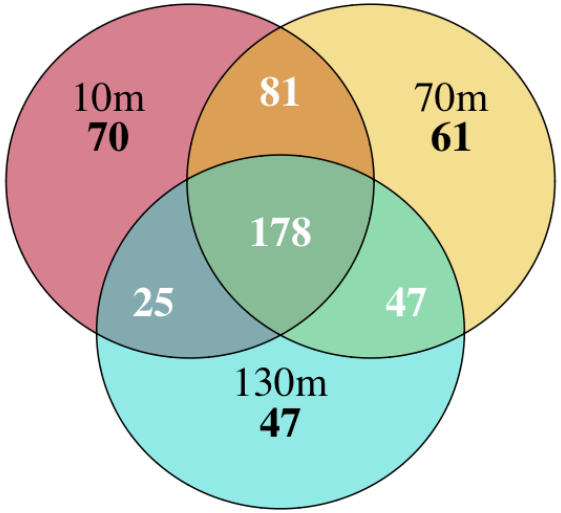


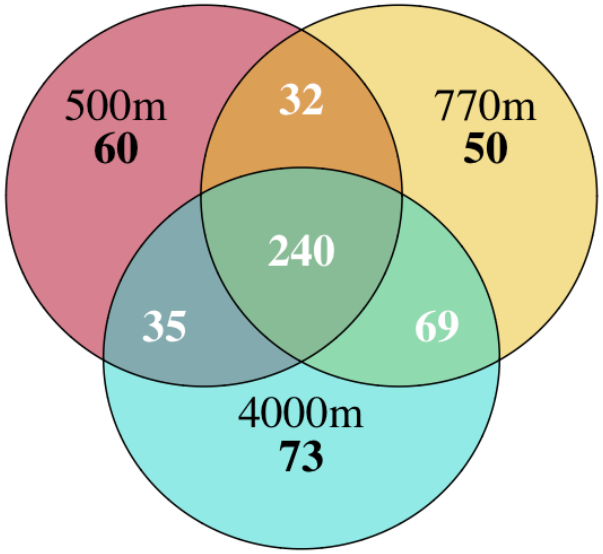
Application: Venn Diagrams

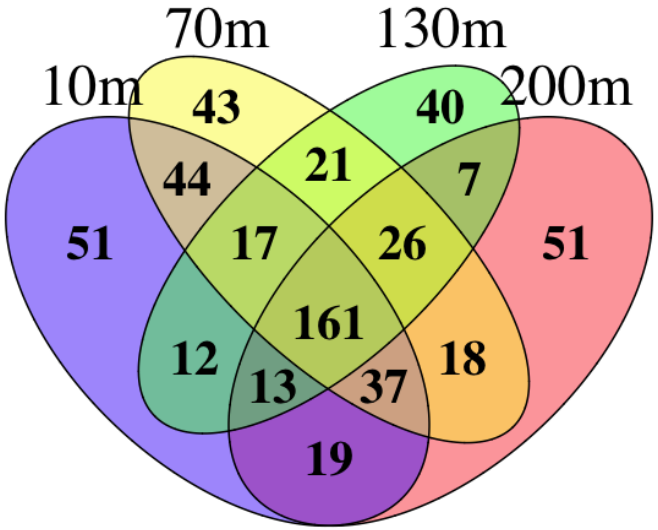
```
# use our newly found skills to identify which pathways had a signal in each
# sample
pwys_10m <- rownames(pathways_wide)[pathways_wide[,"X1_upper_euphotic"] > 0]
pwys_70m <- rownames(pathways_wide)[pathways_wide[,"X6_upper_euphotic"] > 0]
pwys_130m <- rownames(pathways_wide)[pathways_wide[,"X2_chlorophyllmax"] > 0]
pwys_200m <- rownames(pathways_wide)[pathways_wide[,"X3_below_euphotic"] > 0]
pwys_500m <- rownames(pathways_wide)[pathways_wide[,"X5_uppermesopelagic"] > 0]
pwys_770m <- rownames(pathways_wide)[pathways_wide[,"X7_omz"] > 0]
pwys_4000m <- rownames(pathways_wide)[pathways_wide[,"X4_deepabyss"] > 0]


# We can now use these as inputs to the our venn_diagram scripts
venn_10m_and_4000m <- venn_diagram2(pwys_10m, pwys_4000m,
                                    "10m", "4000m")
venn_10m_70m_130m <- venn_diagram3(pwys_10m, pwys_70m, pwys_130m,
                                   "10m", "70m", "130m")
venn_500m_770m_4000m <- venn_diagram3(pwys_500m, pwys_770m, pwys_4000m,
                                      "500m", "770m", "4000m")
venn_10m_70m_130m_200m <- venn_diagram4(pwys_10m, pwys_70m, pwys_130m, pwys_200m,
                                        "10m", "70m", "130m", "200m")
venn_200m_500m_770m_4000m <- venn_diagram4(pwys_200m, pwys_500m, pwys_770m,
pwys_4000m,
                                           "200m", "500m", "770m", "4000m")
```

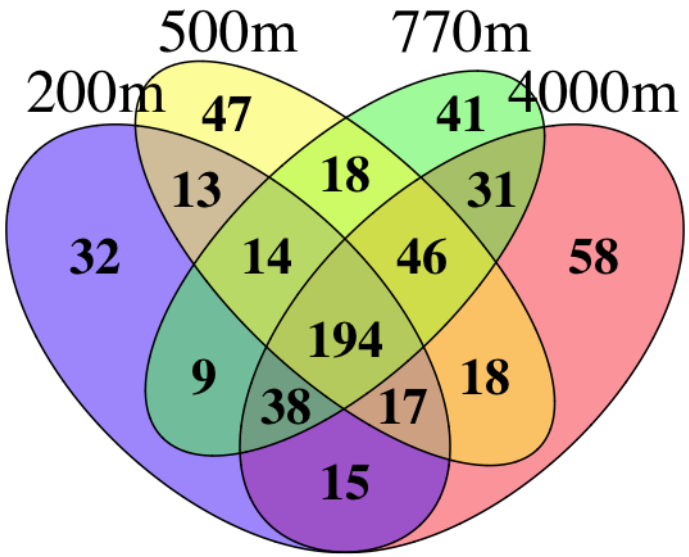
Application: Venn Diagrams







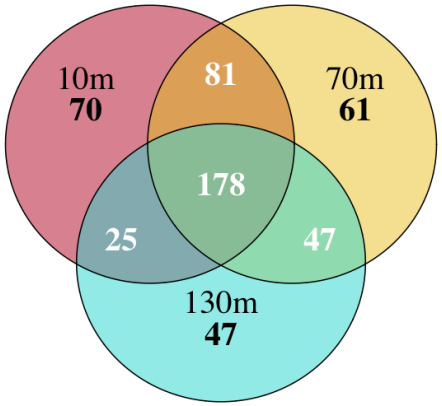


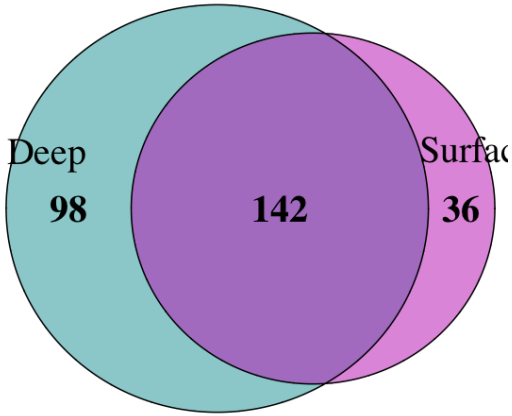


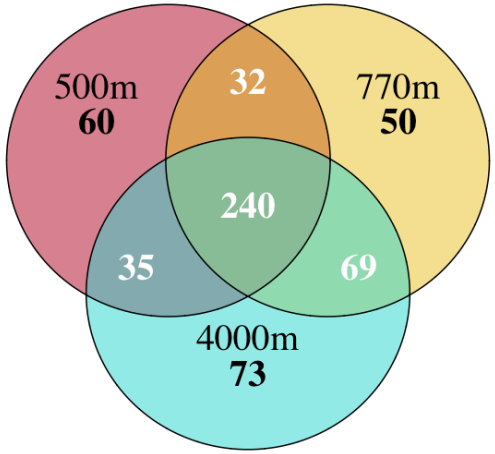
Application: Venn Diagrams

- venn_diagram scripts return a list of all sets defined by combining names of sets:
  - name1_name2: means the intersection of name1 and name2
  - name1_only: means only pathways contained in set name1
- this can be used to compare the cores of our two triple sets

```
my_colors = c("#96B4D4", "#ED808B") # custom colors
compare_cores <- venn_diagram2(venn_500m_770m_4000m
$"500m_770m_4000m", venn_10m_70m_130m$"10m_70m_130m",
                               "Deep", "Surface", colors=my_colors, euler=TRUE)
```







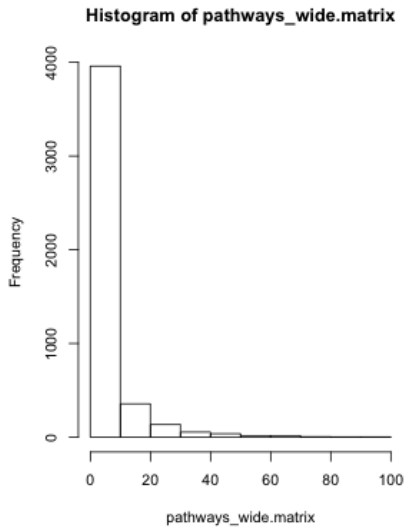
Data distributions

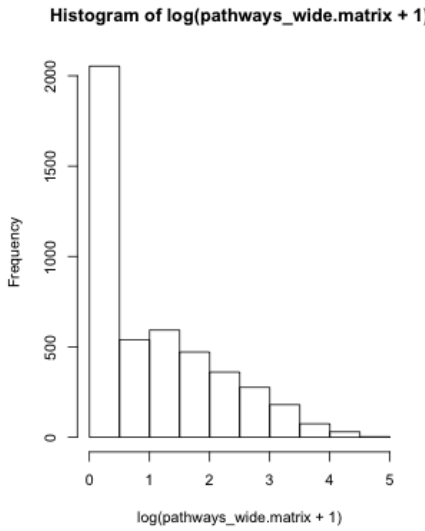
- use hist and density to check distribution of data

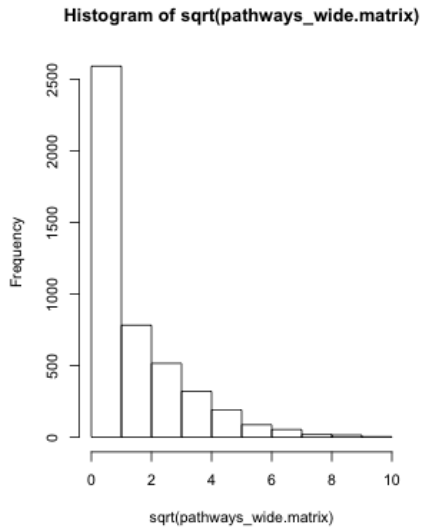
```
par(mfrow=c(2,3)) # ready to six images 2x3
pathways_wide.matrix <- as.matrix(pathways_wide)
hist(pathways_wide.matrix)
hist(log(pathways_wide.matrix+1))
hist(sqrt(pathways_wide.matrix))
plot(density(pathways_wide.matrix))
plot(density(log(pathways_wide.matrix+1)))
plot(density(sqrt(pathways_wide.matrix)))
par(mfrow=c(1,1)) # need to set back to 1x1 at end
```

Data distributions

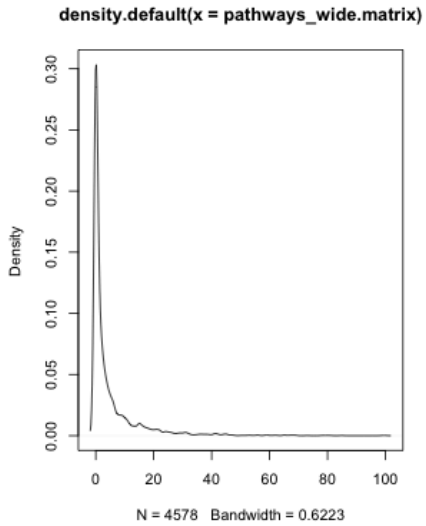
hist()

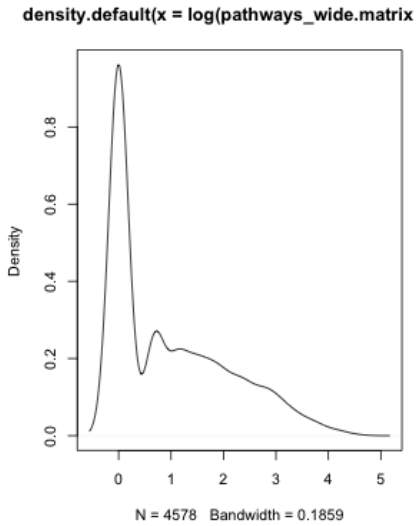


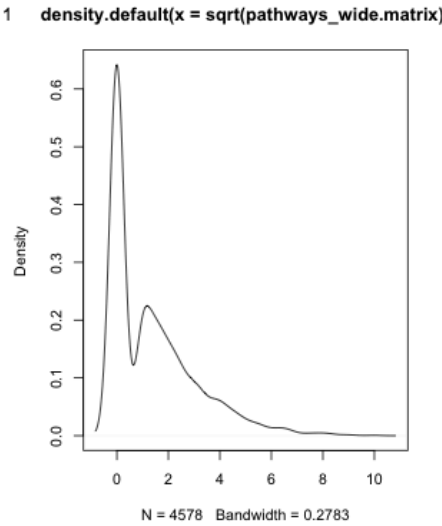




density()







ggplot2

"Wide" Table Format

	sample1	sample2	sample3
taxa1	2	5	4
taxa2	4	6	9
taxa3	2	5	7

"Long" Table Format

taxa	sample	value
taxa1	sample1	2
taxa1	sample1	4
taxa1	sample1	2
taxa2	sample2	5
taxa2	sample2	6
taxa2	sample2	5
taxa3	sample3	4
taxa3	sample3	9
taxa3	sample3	7

- Good for clustering with hclust(), pvclust()
- Also valuable to know the transpose `t()` fxn
- Good to have taxa in some kind of acceptable order

- Good for plotting in visualization libraries `ggplot()` and `lattice()` which need to be classified as a *measured* or *categorical* variable

wide to long

- To prepare to use *ggplot2* and *lattice* packages. We are going to transform our wide table to a long table format using **melt()** from the *reshape2* package

```
# add pathways from rowsnames to matrix
pathways_wide$pwy = rownames(pathways_wide)

# go from wide to long table format
pathways_long <- melt(pathways_wide)
colnames(pathways_long)[2] = "samp"
pathways_long$samp <- sub("X","", pathways_long$samp)

# add long pathway names
meta_17 <- read.table("../files/meta_17.txt", sep="\t", header=T, row.names=1)
pathways_long$pwy_long = meta_17[pathways_long$pwy,1]

# download metacyc hierarchy & level the factors
meta_17_hier <- read.table("../files/meta_17_hierarchy.txt", sep="\t", header=F,
row.names=1)
meta_17_hier$V3 <- factor(meta_17_hier$V3, levels=unique(meta_17_hier$V3))
meta_17_hier$V4 <- factor(meta_17_hier$V4, levels=unique(meta_17_hier$V4))
meta_17_hier$V5 <- factor(meta_17_hier$V5, levels=unique(meta_17_hier$V5))
```

wide to long

```
# add pathways from rowsnames to matrix
pathways_wide$pwy = rownames(pathways_wide)
# go from wide to long table format
pathways_long <- melt(pathways_wide)
colnames(pathways_long)[2] = "samp”
pathways_long$samp <- sub("X","", pathways_long$samp)

# add long pathway names
meta_17 <- read.table("../files/meta_17.txt", sep="\t", header=T, row.names=1)
pathways_long$pwy_long = meta_17[pathways_long$pwy,1]

# download metacyc hierarchy & level the factors
meta_17_hier <- read.table("../files/meta_17_hierarchy.txt", sep="\t", header=F,
row.names=1)
meta_17_hier$V3 <- factor(meta_17_hier$V3, levels=unique(meta_17_hier$V3))
meta_17_hier$V4 <- factor(meta_17_hier$V4, levels=unique(meta_17_hier$V4))
meta_17_hier$V5 <- factor(meta_17_hier$V5, levels=unique(meta_17_hier$V5))
```

wide to long

```
# add additional pathway levels
pwy_level1 <- meta_17_hier[pathways_long$pwy,2]
pwy_level2 <- meta_17_hier[pathways_long$pwy,3]
pathways_long <- cbind(pathways_long, pwy_level1)
pathways_long <- cbind(pathways_long, pwy_level2)

# small correction for pathways not found in hierarchy
missing <- setdiff(unique(pathways_long$pwy), rownames(meta_17_hier))
pathways_long <- pathways_long[!(pathways_long$pwy %in% missing),]

# order pathways first by level V3 and then by V4
pwy_order <- intersect(rownames(meta_17_hier[order(meta_17_hier[,"V3"],
meta_17_hier[,"V4"]),]),
                       unique(pathways_long$pwy))

# factor pathways and then the samples
pathways_long$pwy <- factor(pathways_long$pwy, levels = pwy_order)
pathways_long$samp <- factor(pathways_long$samp, levels = hot_metadata$sample)

# whew, you got the longtable
```

ggplot2

- Anatomy of a ggplot call. First you call the main ggplot function with the data frame and an aesthetics function aes() to specify x= and y= variables

```
library(ggplot2)
g <- ggplot(DataFrame, aes(x=samp,y=pwy_level1))
```

- After this you add feature to the plot that can be driven by your data factors, e.g. **geom_point**

```
g <- g + geom_point(aes(size=sqrt(value), color=samp))
```

geom_abline, geom_bar, geom_boxplot, geom_density, geom_errorbar, geom_histogram, geom_rect, geom_smooth, geom_tile, geom_violin

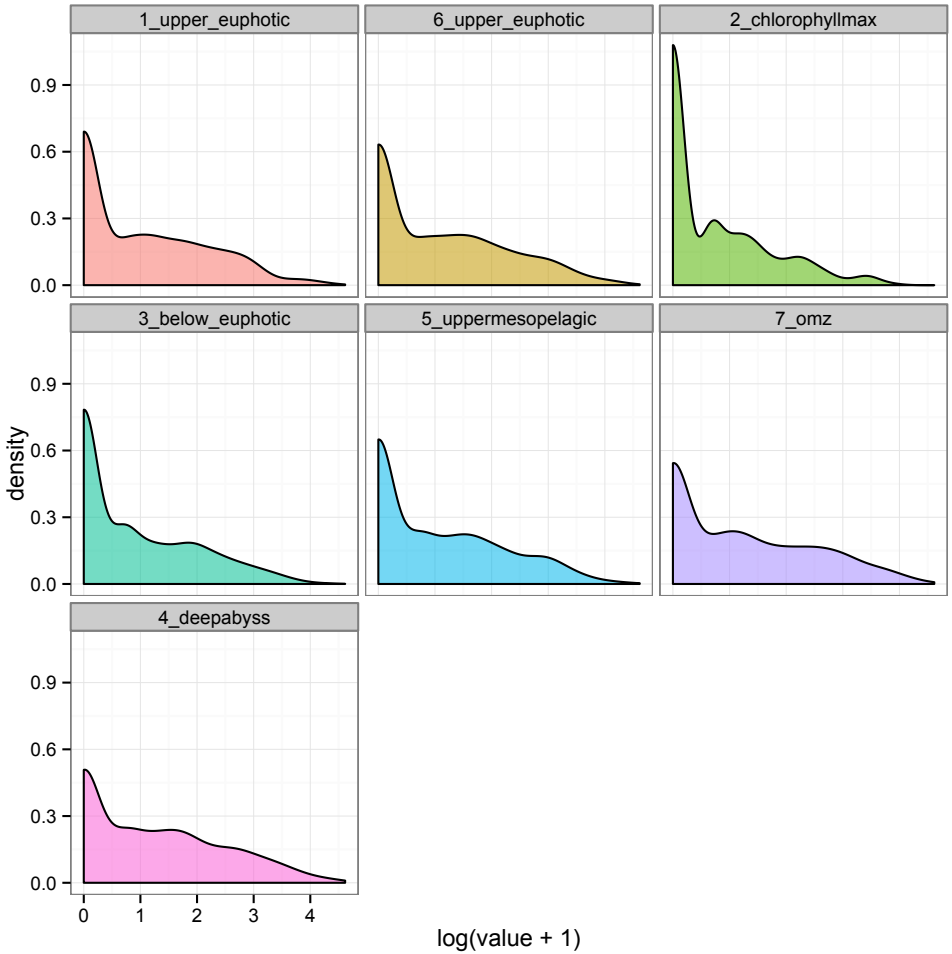
- Each point has its own aesthetics values to change, which gives you powerful set of options to create a variety of plots

http://docs.ggplot2.org/0.9.3.1/geom_bar.html

ggplot2

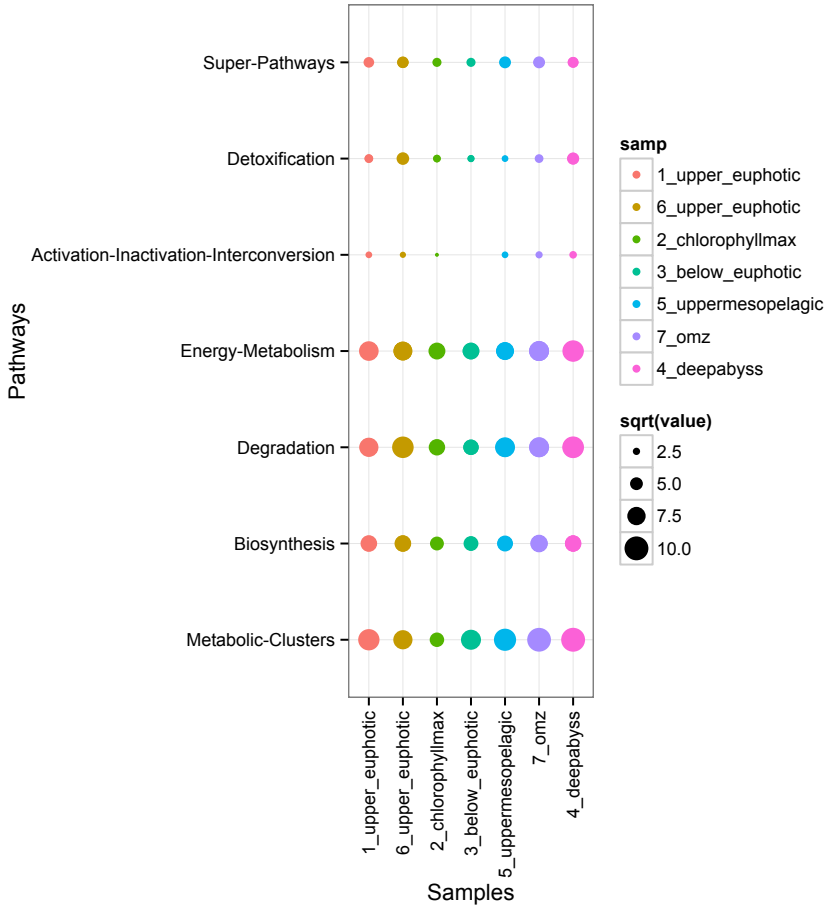
- The **facet_wrap** or **facet_grid** functions allow you to plot multiple subsets of your data on one plot based on some category or condition

```
facet_wrap(~ samp)
```



ggplot2 in action

```
g <- ggplot(subset(pathways_long, value >0),
     aes(x=samp, y=pwy_level1)) +
     geom_point(aes(size=sqrt(value), color=samp)) +
     theme_bw() +
     theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
     labs(x = "Samples", y = "Pathways")
g
```



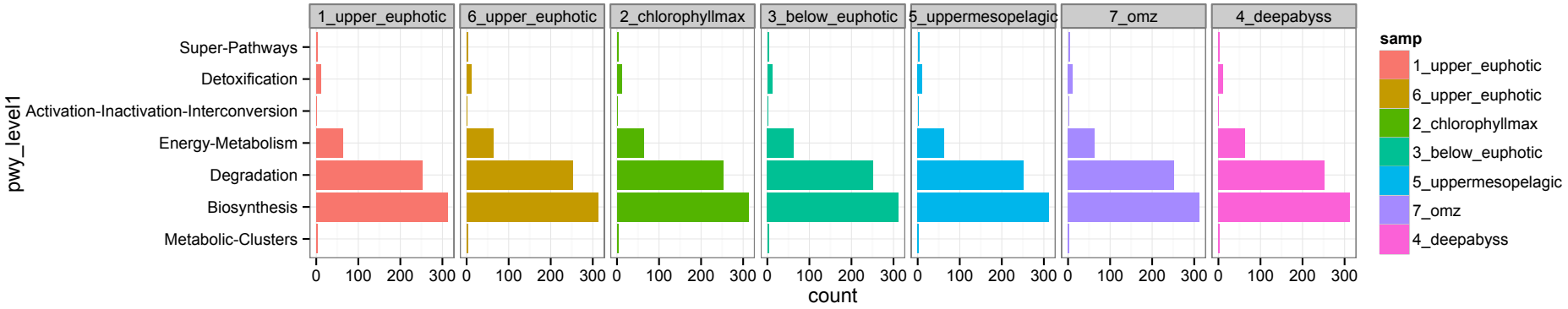
ggplot2 in action

```
g <- ggplot(subset(pathways_long, value >0), aes(x=samp,y=pwy_level2)) +
     geom_point(aes(size=sqrt(value), color=samp)) + theme_bw() +
     theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
     labs(x = "Samples", y = "Pathways")
```

ggplot2 in action

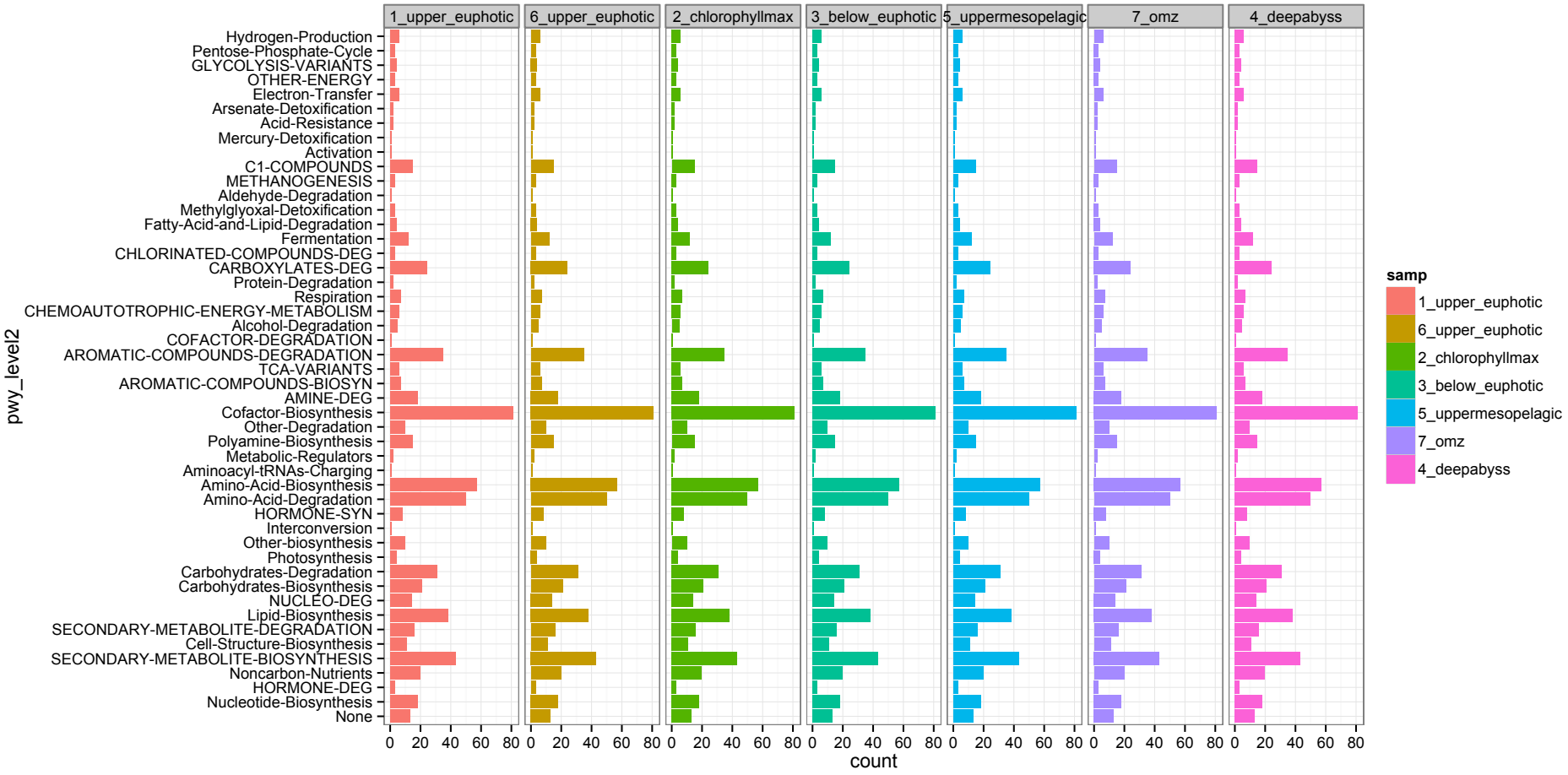
- If your first call to aes only has one factor it assumes you want to count it. So in this case it will count all pathways in the different pwy_level1 classes

```
g <- ggplot(pathways_long, aes(pwy_level1)) +
     geom_bar(aes(fill=samp)) +
     theme_bw() +
     coord_flip() +
     facet_wrap(~ samp, ncol = 7)
g
```



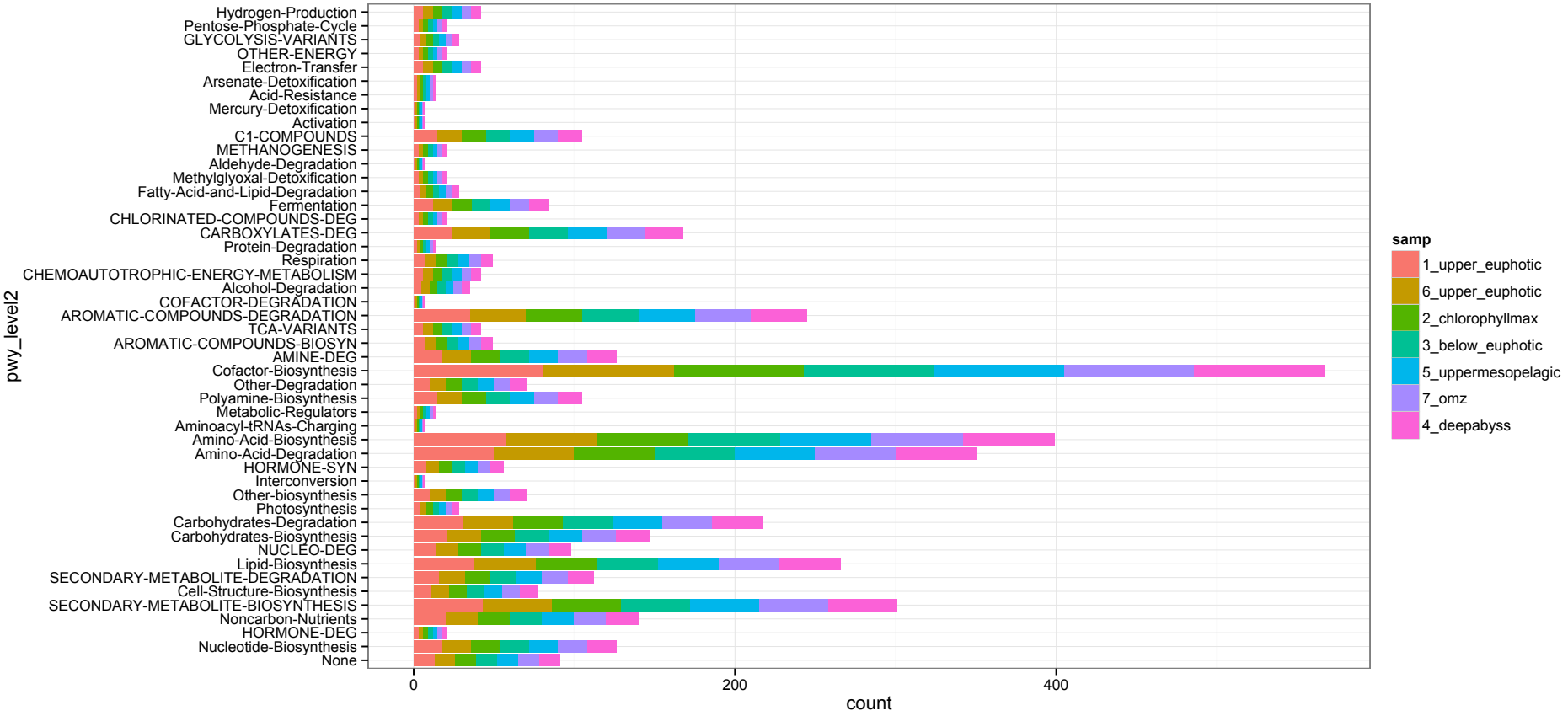
ggplot2 in action

```
g <- ggplot(pathways_long, aes(pwy_level2)) +
    geom_bar(aes(fill=samp)) + theme_bw() + coord_flip() +
    facet_wrap(~ samp, ncol = 7)
g
```



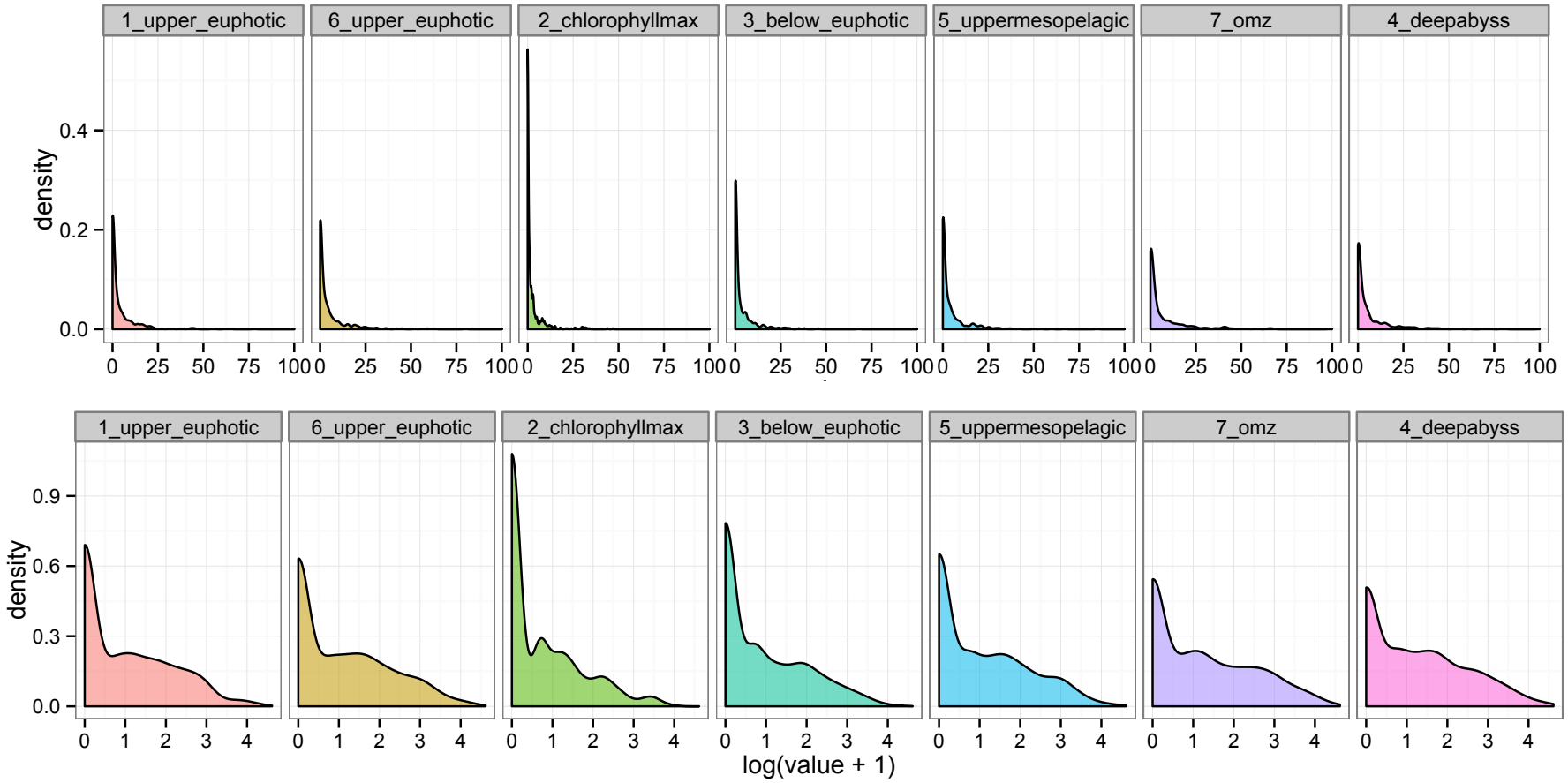
ggplot2 in action

```
g <- ggplot(pathways_long, aes(pwy_level2, fill=samp)) +
     geom_bar() + theme_bw() + coord_flip()
g
```



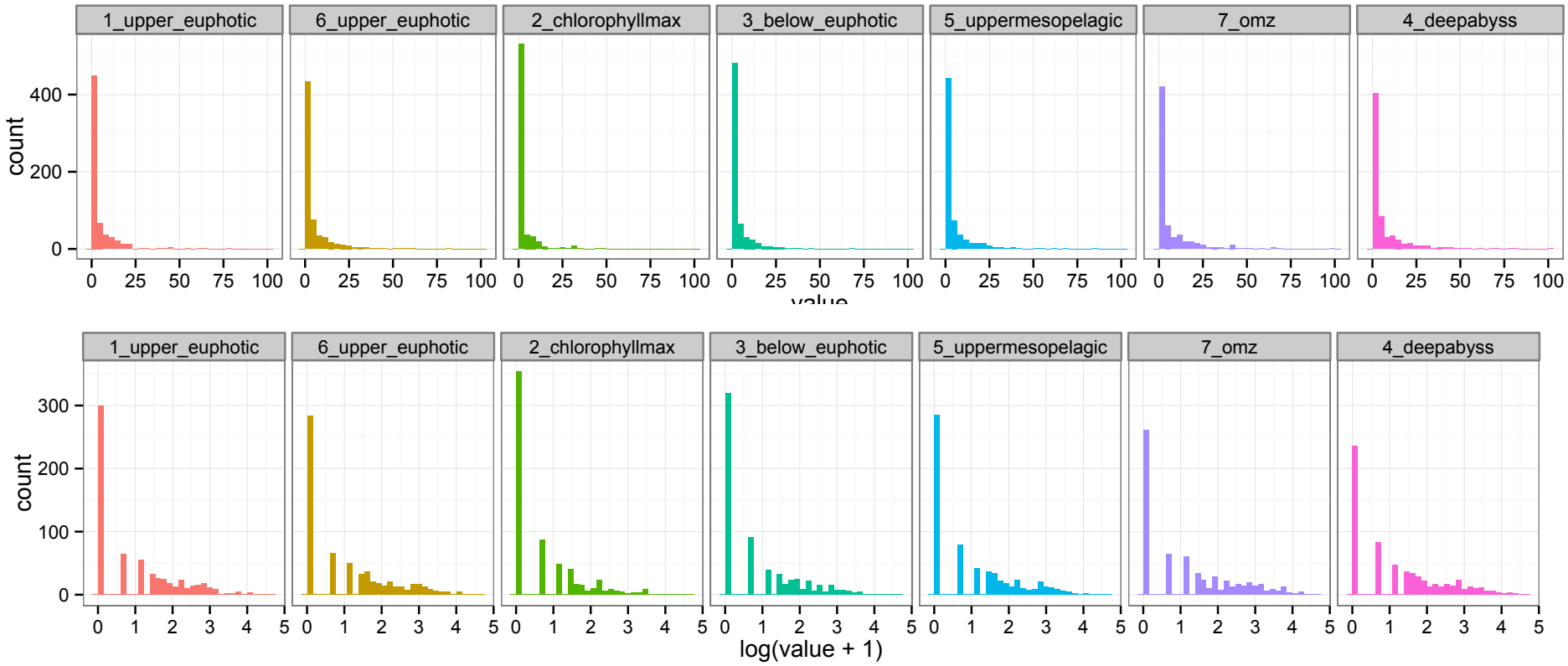
ggplot2 in action

```
g <- ggplot(pathways_long, aes(value)) +
     geom_density( aes(fill=samp, alpha=0.6)) + facet_wrap(~ samp, ncol=7) +
     theme_bw() + theme(legend.position="none")
g
```



ggplot2 in action

```
g <- ggplot(pathways_long, aes(value)) +
     geom_histogram( aes(fill=samp)) + facet_wrap(~ samp, ncol=7) +
     theme_bw() + theme(legend.position="none")
g
```



qplot

- “quick plot” a simpler version of ggplot2, but in generally harder to get full control
- Hadley’s Was Here (He is Loud!)
  Lecture UBC May 2013 -
  http://www.mathtube.org/lecture/video/visualising-data-ggplot2
- Hadley covers it in a lot more detail

learn more

- Google is your friend, i.e., how do I do "**X**" in ggplot2? or I'm getting error "**Y**" in ggplot2
- Really Struggling? http://www.r-tutor.com/r-introduction
- There are also a number of websites:
  ggplot2 doc - http://docs.ggplot2.org/0.9.3.1/index.html
  Cookbook for R - http://www.cookbook-r.com/
- Sit down and experiment to learn structure
- Quick R – Lots of examples of getting things done http://www.statmethods.net/
- Jeff Leek – http://blog.revolutionanalytics.com/2013/04/coursera-data-analysis-course-videos.html
- Getting better: Experimentation and experience make all the difference.