Lab 6 – Comparing Datasets: Venn/Euler Analysis and Dimensionality Reduction

[Software needed: web access and R]

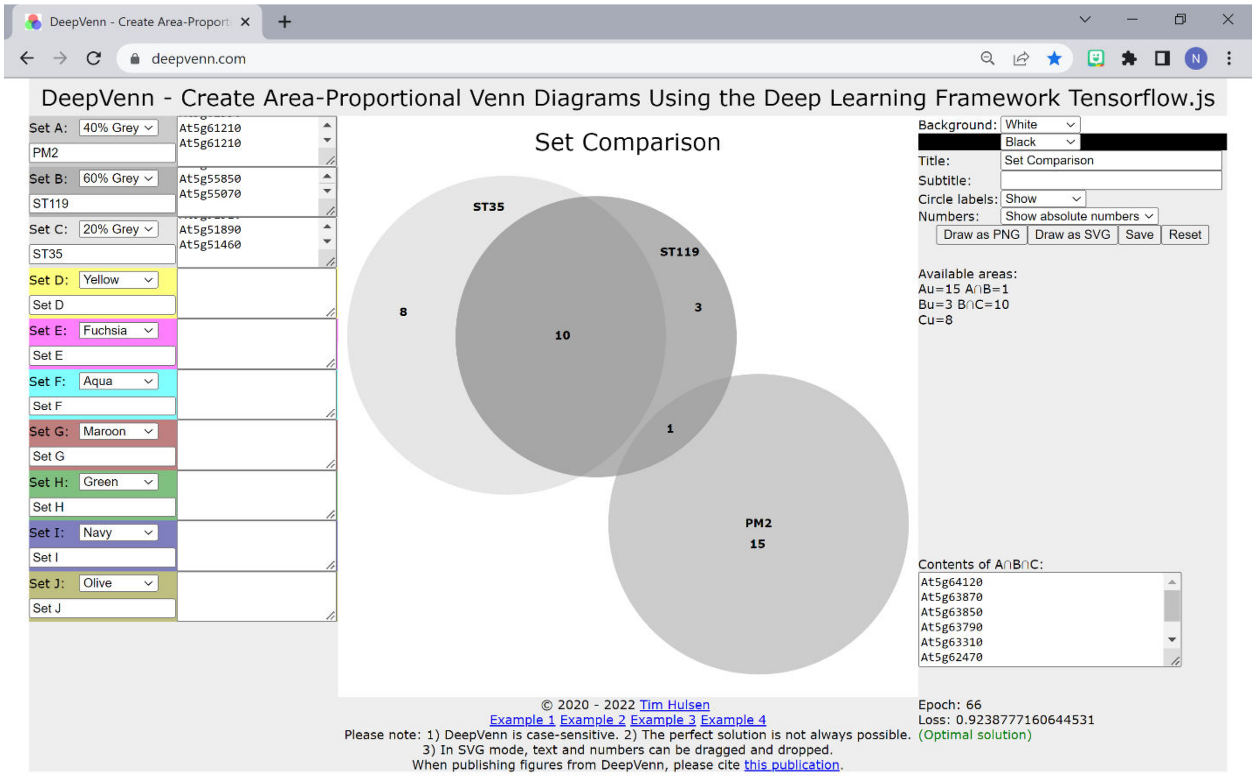
As we heard in the mini-lecture, the dramatic decrease in sequencing costs and concomitant increase in sequencing throughput have driven an expansion in the datasets available for biological analysis. Often, we would like to compare datasets across treatments or with other publicly available datasets: how many genes are in common in Treatment A versus Treatment B? How many are unique to each treatment? We're all familiar with Venn diagrams, which show all possible combinations between 3 sets, depicted as 3 partially overlapping circles. When comparing sets, it is often nicer to use area-proportional Euler diagrams, which use circles to depict only observed combinations that exist in the data. Scaling the area of the circles by the number of elements in the set can make small or large sets readily apparent, and using a Euler diagram makes non-overlapping sets stand out. It is important to note that Euler diagrams are typically useful for comparison of 3 or 4 sets and although there are wacky, generalizable ways of displaying more comparisons, it is probably advisable to use the elegant UpsetR package to concisely show the number of elements in common or unique to larger numbers of datasets.

A second way of comparing data involves dimensionality reduction to group samples by similarity of e.g. expression values across all genes, using principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), uniform manifold approximation and projection (UMAP) and the like. With all these methods, complex multidimensional data, such as expression profiles for 20,000 genes from 300k single cells, can be compared and grouped with one another in 2-dimensional space. In the last exercise of the lab we'll perform a UMAP analysis with some transcriptomic data so see how similar one sample is to the others. If a sample of unknown provenance is included in such an analysis, we can make hypotheses about its provenance by visualizing how closely it associates with one of known provenance.

DeepVenn – an online app for generating area-proportional Euler diagrams

Let's visualize how similar 3 sets of gene identifiers are, in terms of identifier membership. These sets could be genes that pass some significant differential expression cutoff, as compared to an untreat control. Go to https://www.deepvenn.com/, a web app that is based on BioVenn at www.biovenn.nl by Tim Hulsen and colleagues (T. Hulsen, J. de Vlieg and W. Alkema [2008] BioVenn - a web application for the comparison and visualization of biological lists using area-proportional Venn diagrams, BMC Genomics 9: 488; https://doi.org/10.1186/1471-2164-9-488). The DeepVenn app has been updated using deep learning to improve the circle layout.

In the interface, paste the 3 lists of gene IDs in the "Coursera_DataViz_Lab06_BioVenn_ test_data.xlsx" file, PM2, ST119, and ST35 into the Set A, Set B, and Set C boxes, respectively, as shown in **Figure 1**. Just paste the identifiers and not the descriptions in the first two rows. The names in the first row can be entered into the label boxes to the left of the boxes where you pasted the identifiers. Click on "Draw as SVG" to generate an area-proportional Euler diagram. In the SVG mode, it is possible to move the text so that it is better situated. It is also possible to have the interface return the identifiers that are in common between different sets, such as the 10 identifiers common to all three sets, A∩B∩C, by clicking on these in the “Available areas” panel on the right.


DeepVenn - Create Area-Proport
deepvenn.com
DeepVenn - Create Area-Proportional Venn Diagrams Using the Deep Learning Framework Tensorflow.js
Set Comparison
ST35
ST119
PM2
8
10
3
1
15
Draw as PNG
Draw as SVG
Save
Reset
Example 1 Example 2 Example 3 Example 4

Set A: 40% Grey
PM2
At5g61210
At5g61210

Set B: 60% Grey
ST119
At5g55850
At5g55070

Set C: 20% Grey
ST35
At5g51890
At5g51460

Set D: Yellow
Set D

Set E: Fuchsia
Set E

Set F: Aqua
Set F

Set G: Maroon
Set G

Set H: Green
Set H

Set I: Navy
Set I

Set J: Olive
Set J

Background: White
Black
Title: Set Comparison
Subtitle:
Circle labels: Show
Numbers: Show absolute numbers

Available areas:
Au=15 A∩B=1
Bu=3 B∩C=10
Cu=8

Contents of A∩B∩C:
At5g64120
At5g63870
At5g63850
At5g63790
At5g63310
At5g62470

Epoch: 66
Loss: 0.9238777160644531
(Optimal solution)

© 2020 - 2022 Tim Hulsen
Please note: 1) DeepVenn is case-sensitive. 2) The perfect solution is not always possible. 3) In SVG mode, text and numbers can be dragged and dropped.
When publishing figures from DeepVenn, please cite this publication.

Figure 1: DeepVenn area-proportional Euler diagram of the results of a set membership analysis for the 3 datasets provided in the example.

With such an approach it is immediately apparent that there is very little overlap between the identifiers in our PM2 sample and those in the other two datasets. We can see too that the three sets are approximately equal in size.

Comment on the “Class” labels in the example Excel file. Is something sus?

Lab Quiz Question 1

Such an approach is useful for the comparison of 3 or 4 different sets. With larger numbers of sets, things can get overwhelming very quickly. Let’s check out an R package called UpSetR that provides a scalable solution for comparing a greater number of sets.

UpSetR – scalable set comparisons

UpSetR was published in 2017 by Conway, Lex and Gehlenborg (Bioinformatics 33: 2938-2940, https://doi.org/10.1093/bioinformatics/btx364) and is available on Github or CRAN. There is also a Shiny web app for it at https://gehlenborglab.shinyapps.io/upsetr/, but you’ll see that the R code is very straightforward.

1. Upload the "mutations.csv" file to your Jupyter notebook as per Step 3 of the R part of Lab 1.

2. As for other labs, we'll load a package that will help us with our goal of creating an UpSetR plot this week.

```
# Install UpSetR pkg onto JupyterHub – part of base build
library(UpSetR)
```

   You will receive a few messages back about the package having been attached successfully. Ignore warnings.

3. Load the dataset from your working directory, which should be `'/home/jovyan/work'` if you're using Coursera's Jupyter Hub.

```
# load in data
mutations <- read.csv( system.file("extdata",
"mutations.csv", package = "UpSetR"), header=T, sep = ",")
str(mutations)
```

```
'data.frame': 284 obs. of 101 variables:
$ Identifier: Factor w/ 284 levels "02-0003" "02-0033" "02-0047"…
$ TTN : int 0 0 0 1 0 0 0 0 1 1 …
$ PTEN : int 0 0 0 1 1 0 0 0 0 0 …
$ TP53 : int 1 1 0 1 0 1 1 0 0 0 …
$ EGFR : int 1 0 0 0 0 0 1 0 0 0 …
$ MUC16 : int 0 0 0 0 0 0 1 0 0 0 …
$ FLG : int 0 0 0 0 0 0 1 0 0 0 …
$ RYR2 : int 0 0 1 0 1 0 0 0 0 1 …
…
```

   This data set consists of a matrix of values of 1s and 0s indicating whether or not a mutation (of 101 possible oncogenic mutations) occurs in any of 284 cancer genomes sequenced by the International Cancer Genomes Consortium (this is a subset of genomes from a much larger project). The question is, how often do each of these mutations appear in different cancer genome samples?

4. Let's answer this question for a subset of common cancer driver mutations, as displayed in a short commentary piece by Lex & Gehlenborg in Nature Methods (http://www.nature.com/nmeth/journal/v11/n8/abs/nmeth.3033.html). Keep in mind that for *n* sets, there are 2^n possible intersections, so even the UpSetR display has limitations! The authors recommend their method for more than 3 and less than 30 sets. See **Figure 2** for the output.

```
upset(mutations, sets = c("PTEN", "TP53", "EGFR", "PIK3R1",
"RB1"), sets.bar.color = "#56B4E9", order.by = "freq",
empty.intersections = "on")
```

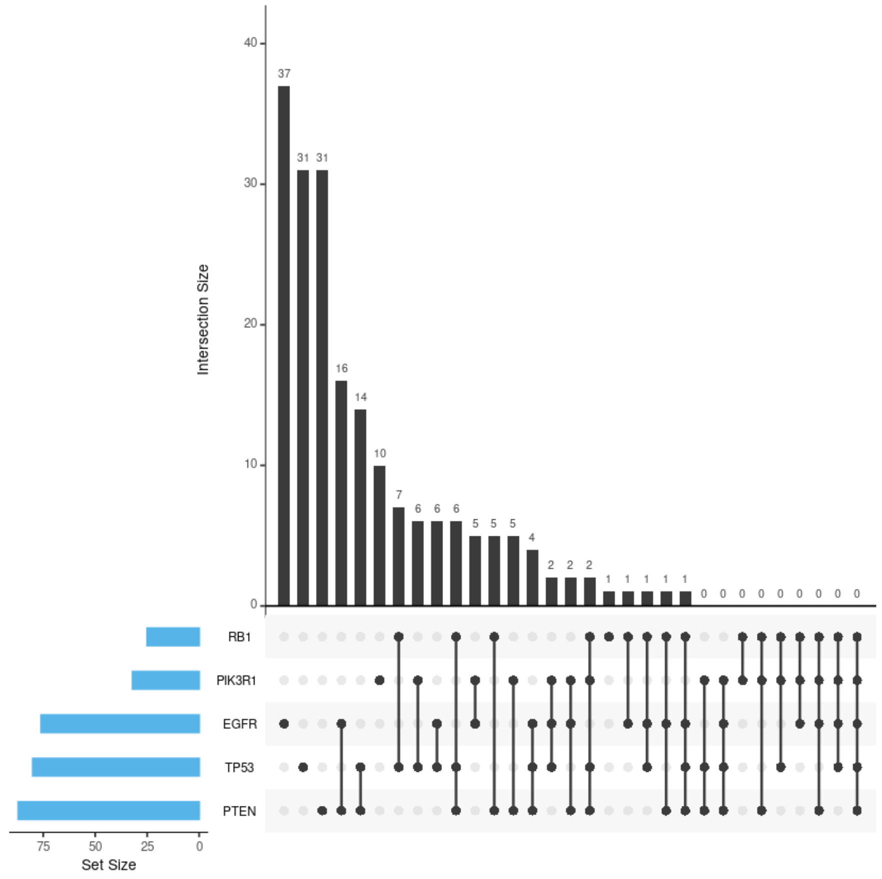


Figure 2: UpSetR output for an analysis of how often 5 common cancer driver genes occur across 284 cancer genomes sequence by the International Cancer Genomes Consortium. See text for interpretation.

In the UpSetR plot looking at how often there are mutations in our subset of 5 oncogenic genes across 284 cancer genomes, the single dots represent the number of times that gene uniquely appears in those genomes, in the context of the comparison sets: for example, there are 76 cancer genome samples that have a mutation in the EGFR gene, and 37 of those have only an EGFR mutation and not any of the other mutations listed. Vertical lines connecting dots denote other comparisons: for example, there are 2 cancer genomes that have mutations in RB1, PIK3R1, TP53, and PTEN. *How many cancer genomes have mutations in TP53 and RB1?* To find out which set members are in each intersection subset, try the UpSet2 app!

Lab Quiz Question 2

UMAP

Uniform Manifold Approximation and Projection is one of several methods commonly used for dimensionality reduction. We'll ask the question how similar expression datasets from many COVID-19 patients are. Desai et al. (2020; doi: https://doi.org/10.1101/2020.07.30.20165241) published this work as a preprint, and the data were download from the GEO entry that accompanied it, at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE150316 (the file was GSE150316_DeseqNormCounts_final.txt.gz, this dataset has normalized expression counts from RNA-seq data). There are expression datasets from lung tissue but also from other organs. UMAP plots better preserve the global structure of the data, and have a different theoretical foundation to balance local and global structure. They are faster to run than e.g. t-SNE plots, too.

1. As in previous weeks, we'll load in some packages that will help us chart our data. You'll get a few notes that the packages were installed successfully.

```
# Packages to help tidy our data
library(tidyverse)
library(readxl)

# Packages for the graphical analysis section
library(RColorBrewer)

# Data projection packages
library(umap)
```

2. Now we can load in the fairly large RNA-seq datasets and the corresponding metadata.

```
# Read in our RNAseq data (.gz file is recognized as text)
tissue_data.df <- read.table(file =
"GSE150316_DeseqNormCounts_final.txt.gz",
header = TRUE,
row.names = 1)
# Take a quick look at it
head(tissue_data.df)
dim(tissue_data.df)
#Read in some additional patient data
patient_data.df <- read_excel("2020.07.30.20165241-
1_supp_table3.xlsx", sheet=1)
# Take a quick look at it
head(patient_data.df)
dim(patient_data.df)
```

A data.frame: 6 × 88

	case1.lung1	case1.lung2	case1.lung3	case1.lung4	case1.heart1	case2.lung1	case2.lung2	case2.jejunum1	case2.lung3	case2.heart1	⋯	cas
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	⋯	
5S_rRNA	7.96319457	4.5713330	7.6071080	39.461459	225.083082	3.7434610	7.1978551	1.7137780	2.195275	11.33237	⋯	
5_8S_rRNA	1.01657803	0.0000000	1.0867297	10.961516	83.364104	1.4037979	1.6936130	0.0000000	2.927033	12.20409	⋯	
7SK	0.42357418	0.0000000	0.0000000	2.192303	2.778803	0.0000000	0.4234032	0.0000000	0.000000	0.00000	⋯	
A1BG	0.16942967	0.0000000	0.5433649	0.000000	2.778803	0.0000000	0.8468065	0.0000000	0.000000	0.00000	⋯	
A1BG-AS1	0.16942967	0.9142666	0.5433649	8.769213	0.000000	0.4679326	0.8468065	0.0000000	1.829396	0.00000	⋯	
A1CF	0.08471484	0.0000000	0.5433649	0.000000	36.124445	0.0000000	0.0000000	0.4284445	0.000000	0.00000	⋯	

59090 · 88

A tibble: 6 × 18

Case No	Viral high vs. viral low*	Viral load	ISH hyaline membrane reactivity	RNA seq coverage	qRT-PCR**	Keratin cells / mm2	Napsin A cells / mm2	CD1 63 cells/ mm2	CD 3 Cells/ mm2	CD 4 cells/ mm2	CD8 cells/ mm2	CD20 cells/ mm2	CD123 cells/ mm2	CD18 cells/ mm2	CD56 cells/ mm2	IDO1 cells/ mm2	PD-L1 cells/ mm2
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	High	81.2	Present	99.96	Positive	192	174	216	174	98	24	17	6	3	59	51	50
2	Low	0.5	Absent	9.4	Positive	335	429	1026	420	225	78	21	63	17	31	68	13
3	Low	2	Absent	1.27	Positive	663	474	855	956	354	212	51	14	3	25	52	1
4	Low	<0.01	Absent	0.15	Negative	1045	680	723	290	152	39	23	31	7	54	27	1
5	High	18.5	Present	24.27	Positive	478	183	1010	386	203	65	12	4	2	20	28	10
6	Low	0.02	Absent	0.17	Negative	198	294	317	167	50	10	15	3	0.4	24	11	0.2

24 · 18

2. Next, we'll reformat our patient data and store that information in a data frame.

```
# Reformat our patient data, store in patient_viral_load.df
patient_viral_load.df <-
patient_data.df %>%
rename_with(str_replace_all, pattern="\\r\\n|\\s",
replacement = "_") %>%
select(1:3)
```

3. As we saw in when used the **`dim(tissue_data.df)`** command, the RNA-seq data sets are fairly large. Genes where transcript levels are very low, or which don't show a lot of variance, are not that informative for our UMAP analysis, and will just slow it down. Let's filter those out…this code block will take a long time to run!

```
# Trim the tissue data down...this takes a while!
tissue_data_filtered.df <-
tissue_data.df %>%
# Convert the row names to a column
rownames_to_column(var="gene") %>%
# Set up the table to perform row-wise operations
rowwise() %>%
# Calculate the mean expression of each gene across all
# tissue samples
```

```
mutate(mean = mean(c_across(where(is.numeric)))) %>%
# Filter for samples with low expression
filter(mean > 0.5) %>%
# Calculate overall variance in case we need to make our
dataset smaller
mutate(variance = var(c_across(where(is.numeric)))) %>%
# Arrange samples by descending variance
arrange(desc(variance)) %>%
# Remove the grouping specification
ungroup()
```

4. Did our filtering help? Let's take a look…

```
# Take a look at the final results
head(tissue_data_filtered.df)
# how big is our filtered data frame?
dim(tissue_data_filtered.df)
```

```
A tibble: 6 × 91

   gene  case1.lung1  case1.lung2 case1.lung3 case1.lung4 case1.heart1 case2.lung1 case2.lung2 case2.jejunum1 case2.lung3 ⋯ caseE.lung.NYC cas
  <chr>        <dbl>        <dbl>       <dbl>       <dbl>        <dbl>       <dbl>       <dbl>          <dbl>       <dbl> ⋯          <dbl>
MT-ND5     148.16625    223.53818    328.1924    541.4989     4854.570    6289.014    3615.440       6206.875    3324.012 ⋯       5588.286
    MT-
   RNR2   1819.92882   1450.02682   3469.9280   4106.1840    21257.847    5766.334    6830.764       3656.345    5538.313 ⋯      22644.770
MT-ND6      56.58951     79.99833    146.1651    190.7304     1889.586    2698.099    1279.525       2287.894    1129.835 ⋯       2432.395
MT-ND4     271.17219    335.99297    484.6815    506.4221     7808.438    7505.639    3720.868       6911.667    3504.391 ⋯       6336.858
MT-CO1     201.96017    378.04924    431.4317    530.5374     7135.967    4354.581    2830.027       4528.658    2828.612 ⋯       4053.063
MALAT1   13603.08476  21286.41203  22529.5371  30872.0147    22994.599   19772.025   21814.158      13768.064   20359.346 ⋯      21388.172

29220 · 91
```

5. Next, we'll transpose the data and merge it with some additional information

```
# We need to transpose the data.
# We can do it with dplyr to keep it as a data frame and to
add some info
tissue_RNAseq.df <-
tissue_data_filtered.df %>%
select(1:89) %>% # trim down the columns
pivot_longer(cols=c(2:89), names_to = "sample", values_to =
"norm_counts") %>%
pivot_wider(names_from = gene, values_from = norm_counts)
```

6. Now we'll add some additional information.

```
# We want to add some additional sample information before
# assessing the data
```

```r
tissue_RNAseq.df <-
tissue_RNAseq.df %>%
# Grab just the sample names
select(sample) %>%
# Grab information from it like case number, tissue, and
# tissue number
str_match_all(., pattern=c("case([\\w]+)\\.([a-
z]+)([\\d|\\.NYC]*)|(NegControl\\d)")) %>%
# Bind it all together
do.call(rbind, .) %>%
# Convert results to a data frame,
# DO NOT convert strings as factors
as.data.frame(stringsAsFactors = FALSE) %>%
# Rename the columns based on the capture groups
rename(., sample = V1, case_num = V2, tissue = V3,
tissue_num = V4, neg_num = V5) %>%
# Coalesce some of the info due to negative control samples
# and clean up a column
mutate(case_num = coalesce(case_num, neg_num),
tissue_num = str_replace_all(.$tissue_num, pattern = "\\.",
replace = ""), tissue = replace_na(tissue, "None")
# Previous line: replace negative control tissues to "None"
) %>%
# Drop the neg_num column
select(1:4) %>%
# Join this result to the RNA-seq info
full_join(., y=tissue_RNAseq.df, by=c("sample" =
"sample")) %>%
# Join that result to grab viral load information
right_join(patient_viral_load.df, y=., by=c("Case_No" =
"case_num")) %>%
# Fix some column names
rename(case_num = Case_No,
viral_load = `Viral_high_vs._viral_low*`,
viral_load_percent = `Viral_load`)
head(tissue_RNAseq.df)
```

A tibble: 6 × 29226

case_num	viral_load	viral_load_percent	sample	tissue	tissue_num	MT-ND5	MT-RNR2	MT-ND6	MT-ND4	⋯	ZSCAN32	SUGT1P2	FW83563
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	⋯	<dbl>	<dbl>	<
1	High	81.2	case1.lung1	lung	1	148.1662	1819.929	56.58951	271.1722	⋯	0.1694297	0.4235742	0.254
1	High	81.2	case1.lung2	lung	2	223.5382	1450.027	79.99833	335.9930	⋯	0.4571333	0.0000000	0.457
1	High	81.2	case1.lung3	lung	3	328.1924	3469.928	146.16515	484.6815	⋯	1.0867297	0.5433649	0.543
1	High	81.2	case1.lung4	lung	4	541.4989	4106.184	190.73039	506.4221	⋯	0.0000000	2.1923033	0.000
1	High	81.2	case1.heart1	heart	1	4854.5697	21257.847	1889.58637	7808.4378	⋯	0.0000000	2.7788035	0.000
2	Low	0.5	case2.lung1	lung	1	6289.0144	5766.334	2698.09948	7505.6392	⋯	0.0000000	0.9358652	0.467

7. Let's see how many different tissue types we have.

```
# How many tissue types do we have?
table(tissue_RNAseq.df$tissue)
```

```
bowel      fat    heart  jejunum   kidney    liver     lung   marrow 
    4        1        7        1        3        6       52        1 
 None placenta     skin 
    5        7        1 
```

8. For efficiency's sake, we'll convert our data frame into a matrix. This will save a lot of memory space.

```
# Generate a matrix version of our data but drop the sample
# information!
tissue_RNAseq.mx <- as.matrix(tissue_RNAseq.df[,c(-1:-6)])
```

9. Let's run the UMAP analysis!

```
# Set our seed
set.seed(1981)
# Generate our projection
tissue_umap <- umap(tissue_RNAseq.mx)
```

10. What does the structure of the tissue_umap look like?

```
# load in data
str(tissue_umap)
```

```
List of 4
 $ layout: num [1:88, 1:2] -1.55 1.25 1.56 1.65 1.29 ...
 $ data  : num [1:88, 1:29220] 148 224 328 541 4855 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:29220] "MT-ND5" "MT-RNR2" "MT-ND6" "MT-ND4" ...
 $ knn   :List of 2
  ..$ indexes  : int [1:88, 1:15] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ distances: num [1:88, 1:15] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "class")= chr "umap.knn"
 $ config:List of 24
  ..$ n_neighbors         : int 15
  ..$ n_components        : int 2
  ..$ metric              : chr "euclidean"
  ..$ n_epochs            : int 200
  ..$ input               : chr "data"
  ..$ init                : chr "spectral"
  ..$ min_dist            : num 0.1
  ..$ set_op_mix_ratio    : num 1
```

11. We'll do a little rejigging of our data labels.

```
# Re-map our projection points with our tissue data
tissue_umap.df <- data.frame(x.coord =
tissue_umap$layout[,1],
y.coord = tissue_umap$layout[,2])
tissue_umap.df <- cbind(tissue_RNAseq.df[,1:6],
tissue_umap.df)
tissue_umap.df <-
tissue_umap.df %>%
mutate(viral_load = replace_na(viral_load, replace =
"DNW"))
```

12. Last, let's generate a UMAP plot!

```
# Adjust our plot window size according to the expected
# output
options(repr.plot.width=20, repr.plot.height=20)
combo.colours = c(brewer.pal(12, "Paired"), brewer.pal(12,
"Set3"), brewer.pal(8, "Set1"))
# try combining some other palettes, see
# https://www.datanovia.com/en/blog/top-r-color-palettes-
to-know-for-great-data-visualization/#rcolorbrewer-palettes
# combo.colours = c(brewer.pal(8, "Dark2"),brewer.pal(8,
"Paired"))
# 1. Data
ggplot(data = tissue_umap.df) +
# 2. Aesthetics
aes(x = x.coord, y = y.coord, colour = tissue, shape =
viral_load, ) +
# Themes
theme_bw() +
theme(text = element_text(size=20)) +
# 3. Scaling
scale_colour_manual(values = combo.colours) +
# 4. Geoms
geom_text(aes(label = case_num), size = 10)
# make the data point markers bold:
# geom_text(aes(label = case_num), size = 10,
fontface="bold")
```

You will notice that the first colour options we use (a mix of 3 Colour Brewer palettes) provide terrible contrast for the "skin" marker colour, namely yellow-on-white, which provides a contrast of just 1.04:1 – ideally this should be above 3 or, better yet, 4.5 for normal size text. Check out https://webaim.org/resources/contrastchecker/ (the colour of the yellow marker is #FFFFAC, and the white background is #FFFFFF). If you uncomment the two bolded comment lines in the code block above, and comment out the ones they replace, you will greatly improve the contrast and visibility of the markers, by changing the colour palettes and bolding the markers.

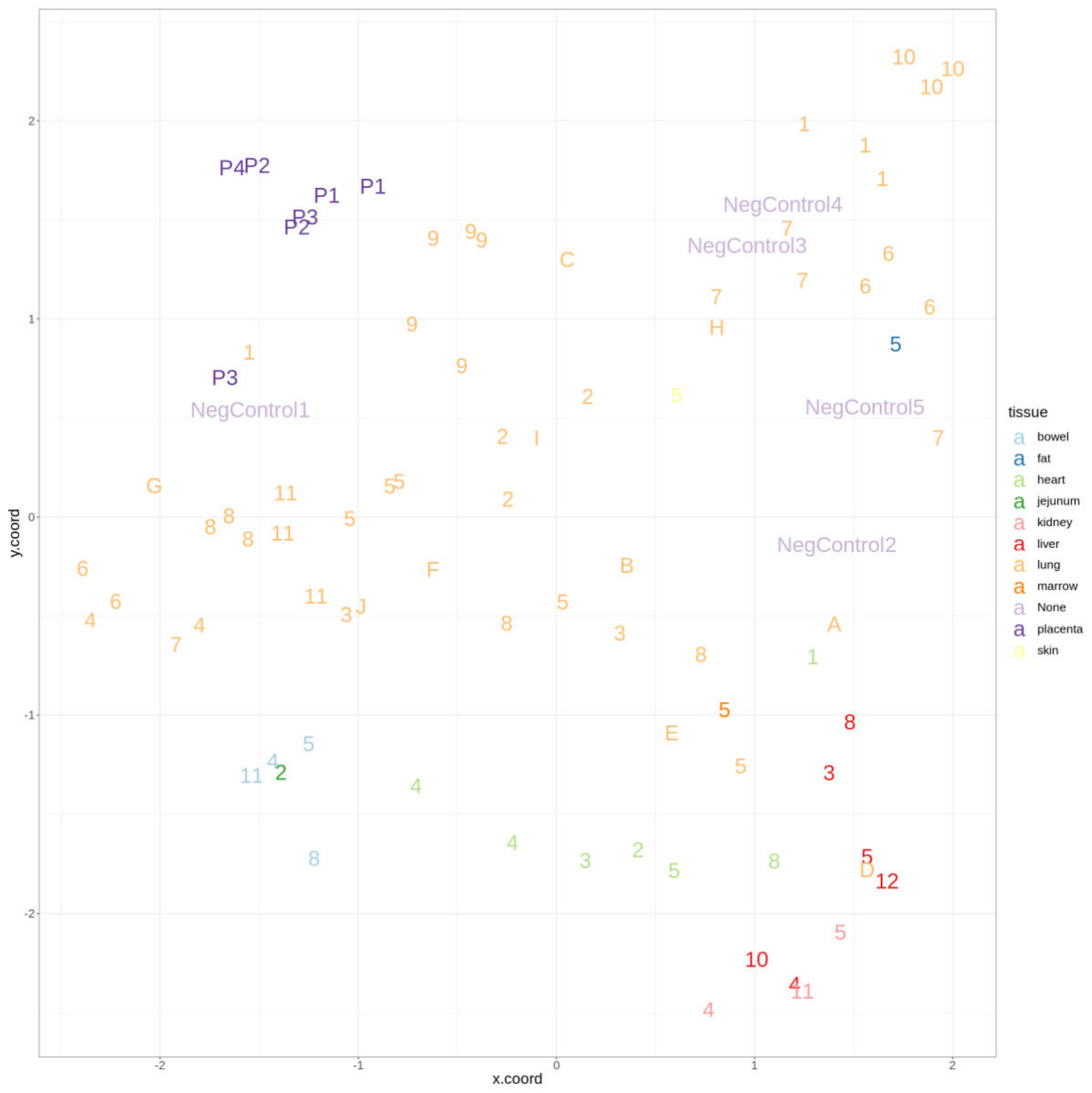


Figure 3. UMAP output for the COVID-19 RNA-seq datasets from Desai et al. (2020). We see clustering of similar tissue samples based on their RNA-seq expression profiles, such as liver and placenta. The COVID-19 lung samples are more broadly dispersed, indicating a phenomenon perhaps worthy of further exploration. This output uses the first `combo.colours` and `geom_texts` options, which don't provide great contrast, especially for the "skin" sample! See the Lab Discussion video for an improved output.

In this lab, we've seen a couple of ways of comparing sets of data and for analyzing to what extent the datasets overlap. We've also explored reducing dimensionality using the UMAP approach in order to see if certain groups of samples are related one to another. Together with clustering approaches covered in a previous lab, these methods are powerful both for data exploration and hypothesis generation.

End of Lab!

Lab Quiz Question 3

Lab 6 Objectives

By the end of Lab 6 (comprising the labs including their boxes, and the lectures), you should:

- understand how improvements in sequencing technologies have led to an explosion of data;
- be able to assess how similar 3 or more datasets are in terms of their members (identifiers) using DeepVenn or UpSetR;
- be able to use the UMAP method for clustering RNA-seq samples;

Do not hesitate to use the Coursera discussion forums if you do not understand any of the above after reading the relevant material.

Further Reading

Desai N, et al. (2020). Temporal and Spatial Heterogeneity of Host Response to SARS-CoV-2 Pulmonary Infection. MedRXiv, https://doi.org/10.1101/2020.07.30.20165241

Hulsen T, de Vlieg J and Alkema W (2008). BioVenn - a web application for the comparison and visualization of biological lists using area-proportional Venn diagrams, BMC Genomics 9: 488; https://doi.org/10.1186/1471-2164-9-488)

Jake R Conway JR, Lex A, Gehlenborg N (2017) UpSetR: an R package for the visualization of intersecting sets and their properties. Bioinformatics 33: 2938–2940, https://doi.org/10.1093/bioinformatics/btx364