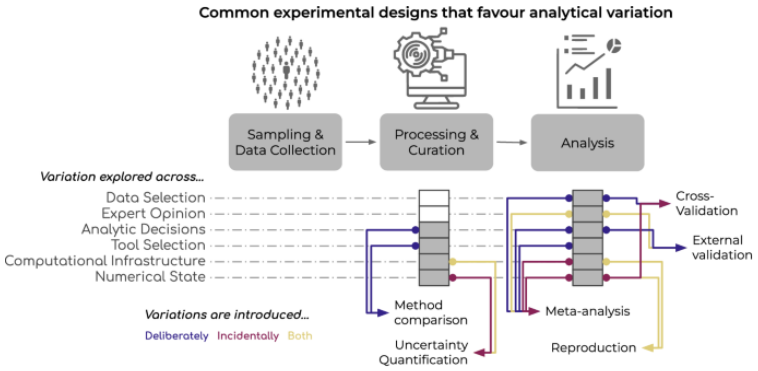


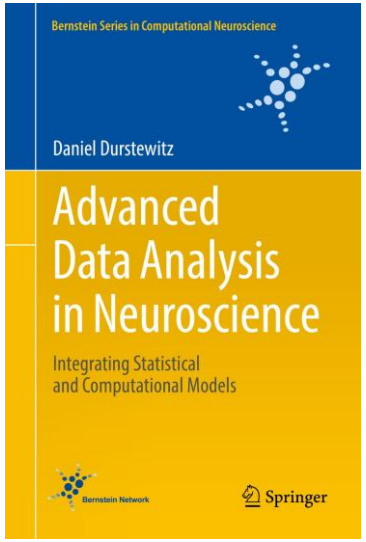
Processing and interpretation of neuroscience data:

Module 1 – Data overview and single cell sequencing

Dr. Kaja Moczulska & Dr. Łukasz Piszczek

Univ.-Prof. Dr. Wulf Haubensak
Department of Neuronal Cell Biology

Center For Brain Research (Zentrum für Hirnforschung)



Overview

Resources needed:

- Laptop
- Internet access
- Materials:
  - VM Machine
  - https://github.com/HugoMalagon/NeuroData_860.053-MUW

21.10
- Introduction to R, basics
- Visual analytics

28.10
- Dimensional reduction: PCA, UMAP
- Normalization/scaling
- Clustering: k-means, knn

4.11
- Intro to Seurat
- Single cell RNA seq
- Dataset merging and preprocessing

11.11
- Clustering in Seurat
- DEG interpretation

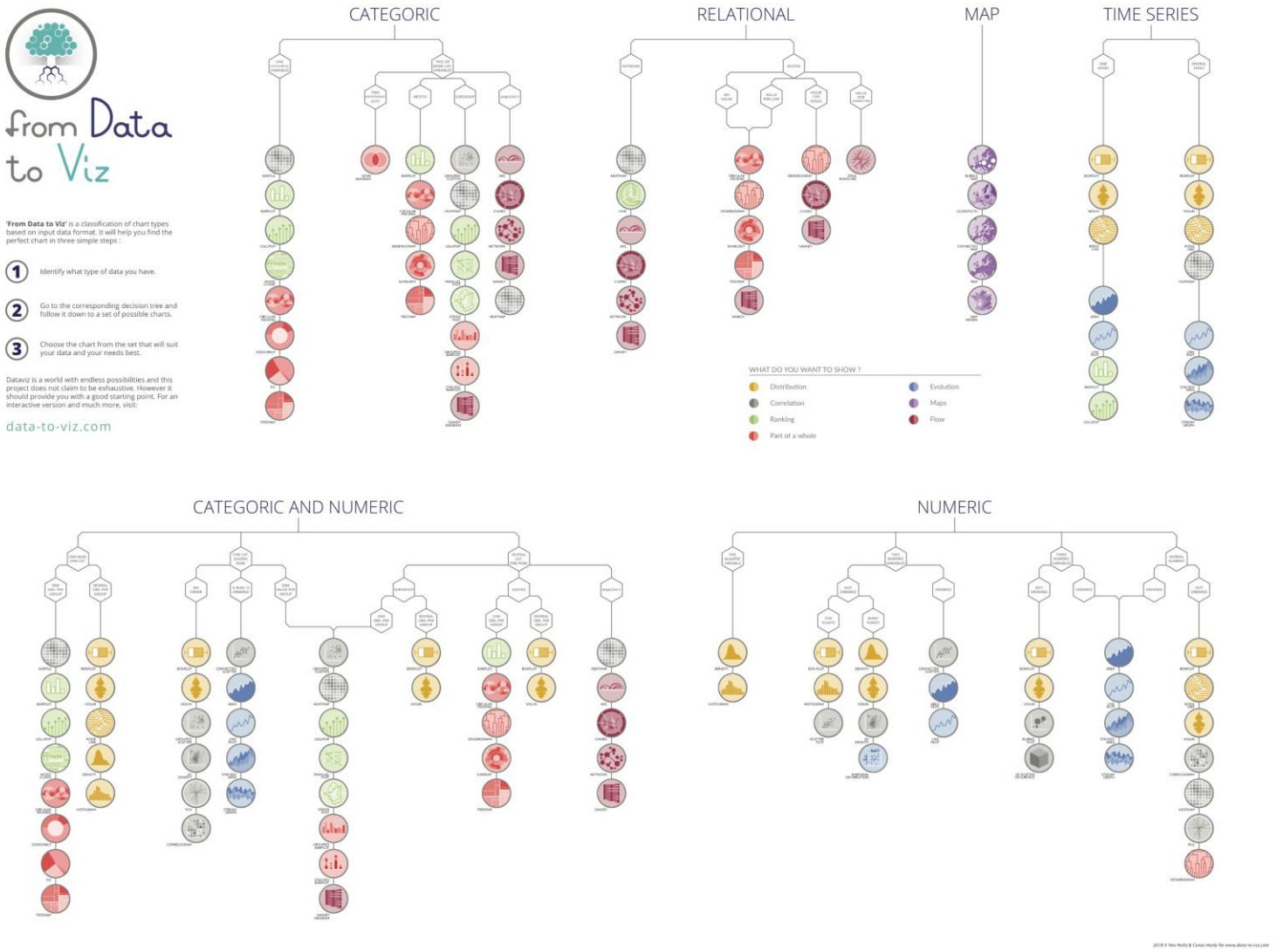
„Exam“: Assessment via email after each class

Day 2

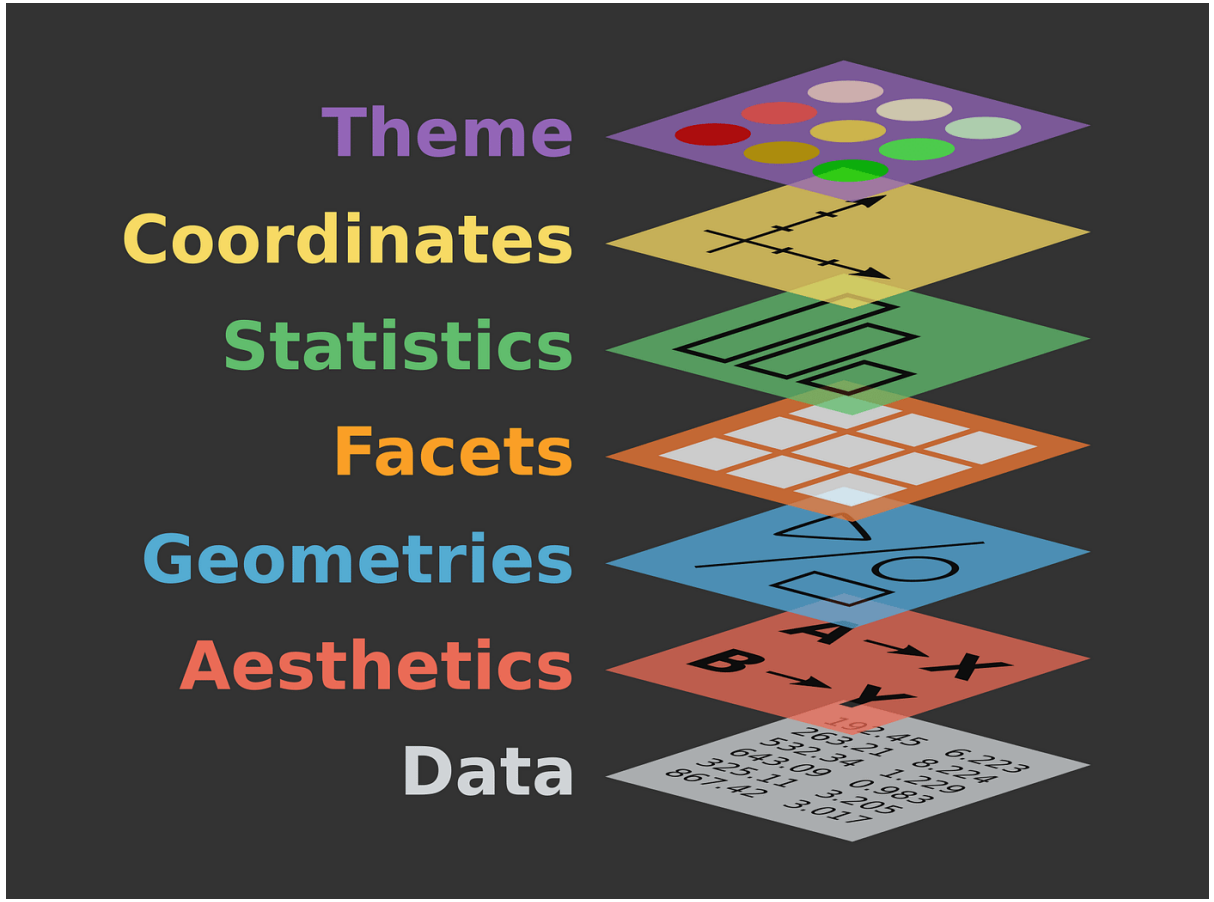
Dimensional reduction techniques: PCA, UMAP

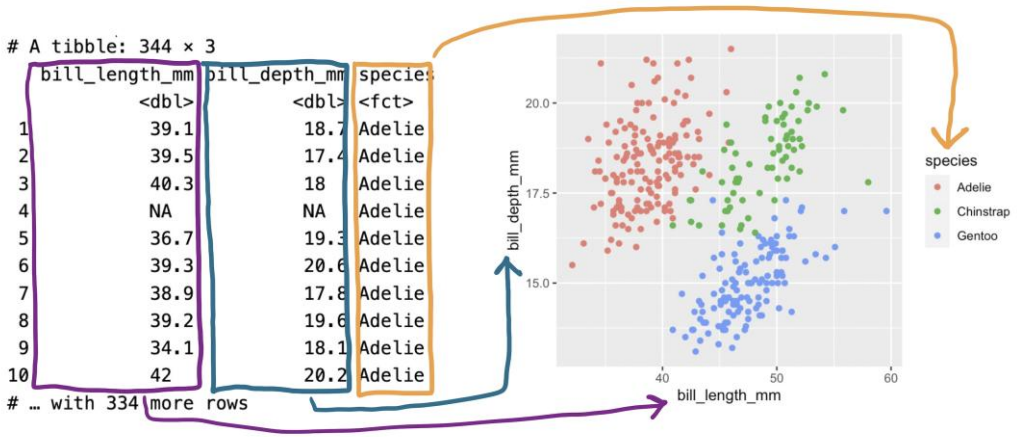
Clustering methods: k-means, k-NN

Different types of plots



Grammar of Graphics





https://www.stat20.org/2-summarizing-data/03-a-grammar-of-graphics/notes

GGplot

Data	• Variables from the dataset
Aesthetics	• x-axis, y-axis, color, fill, size, labels, line width, line type, alpha, shape
Geometries	• Point, line, histogram, bar, boxplot
Facets	• Columns and rows
Statistics	• Binning, smoothing, descriptive stats, inferential stats
Coordinates	• Cartesian, fixed, polar, limits
Themes	• Describes the design elements. Non-data Ink.

Introduction Day 2

- Why scale data/Normalize
- Purpose of dimensional reduction
- Clustering methods – why and how?

- https://cran.r-project.org/web/packages/NeuroDataSets/index.html
- https://cran.r-project.org/web/packages/DataExplorer/vignettes/dataexplorer-intro.html#qq-plot

Load data

```r
# Read normalized gene expression table. Drop unneeded columns; keep gene names.
df <- read_delim(file = "./Input_data/E-GEOD-36980-A-AFFY-141-normalized-expressions.tsv") %>%
  select(-`Gene ID`, -DesignElementAccession) %>%      # Remove Gene ID columns as not needed
  rename(Gene = `Gene Name`) %>%                       # Rename for clarity
  pivot_longer(cols = starts_with("GSM"),              # Reshape: aim is to have one row per patient
    names_to = "Assay", values_to = "Gene_expression") %>%
  pivot_wider(names_from = Gene, values_from = Gene_expression, values_fn = mean) %>%
  dplyr::select(Assay, any_of(genes))                  # Keep only selected genes


df_meta <- read_delim(file = "./Input_data/E-GEOD-36980-experiment-design.tsv") %>%
  dplyr::select(Assay, `Sample Characteristic[disease]`, `Sample Characteristic[sex]`, `Sample Characteristic[organism part]`) %>%
  rename(Class = `Sample Characteristic[disease]`, # Rename for readability
         sex = `Sample Characteristic[sex]`,
         Brain_Area = `Sample Characteristic[organism part]`) %>%
  mutate(Class = factor(Class), # Convert to categorical
         sex = factor(sex),
         Brain_Area = factor(Brain_Area))

# Merge gene expression and metadata on assay ID to create full analysis table.
df <- df %>% left_join(df_meta, ., by = "Assay") #%>% filter(Brain_Area == "hippocampus")
```

```r
## Ivestigate the data structure

Lets investigate the data set

```{r structure}
# Show structure of the main data frame (column types, dime
str(df)
```

And use the *dlookr* library to do summaries
```{r diagnose, echo=FALSE}
# Generate summary statistics for the first 20 columns.
summary_df <- diagnose(df[,1:20])
head(summary_df)

# Numeric summaries (mean, sd, missing)
summary_numeric_df <- diagnose_numeric(df[,1:20])
head(summary_numeric_df)

# Categorical summaries (counts, frequencies)
summary_cat_df <- diagnose_category(df[,1:20])
head(summary_cat_df)
```
```
```
```

Looking at data

```{r structure}
# Show structure of the main data frame (column types, dimensions).
str(df)
```

And use the *dlookr* library to do summaries

```{r diagnose, echo=FALSE}
# Generate summary statistics for the first 20 columns.
summary_df <- diagnose(df[,1:20])
head(summary_df)

# Numeric summaries (mean, sd, missing)
summary_numeric_df <- diagnose_numeric(df[,1:20])
head(summary_numeric_df)

# Categorical summaries (counts, frequencies)
summary_cat_df <- diagnose_category(df[,1:20])
head(summary_cat_df)
```

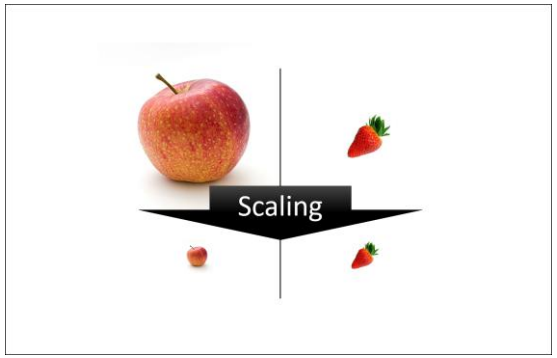
tbl_df 6 x 6 | tbl_df 6 x 10 | tbl_df 6 x 6

A tibble: 6 × 10

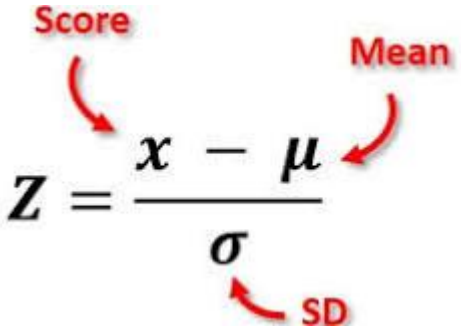
variables <chr>	min <dbl>	Q1 <dbl>	mean <dbl>	median <dbl>	Q3 <dbl>	max <dbl>	zero <int>	minus <int>	outlier <int>
MET	6.026954	7.895806	8.326343	8.290617	8.890246	9.659133	0	0	2
PCSK1	6.301945	8.867314	9.318842	9.430548	9.892827	10.872164	0	0	3
RGS4	8.318429	11.736494	11.742186	11.907373	12.078598	12.470060	0	0	9
HS3ST2	7.941592	8.790481	8.989536	8.980663	9.149315	10.834097	0	0	12
NPTX2	8.353973	8.811007	9.335945	9.190865	9.846022	11.125804	0	0	0
NEUROD6	6.595357	8.023035	8.343437	8.381035	8.704242	9.595137	0	0	3

6 rows

Scaling the data



Data scaling is the process of transforming numerical features in a dataset to a common scale or range, which is essential for many machine learning algorithms that rely on distance calculations.

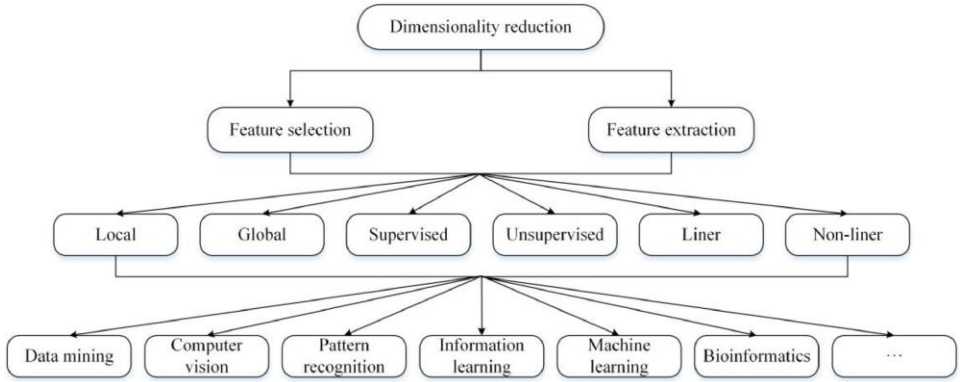


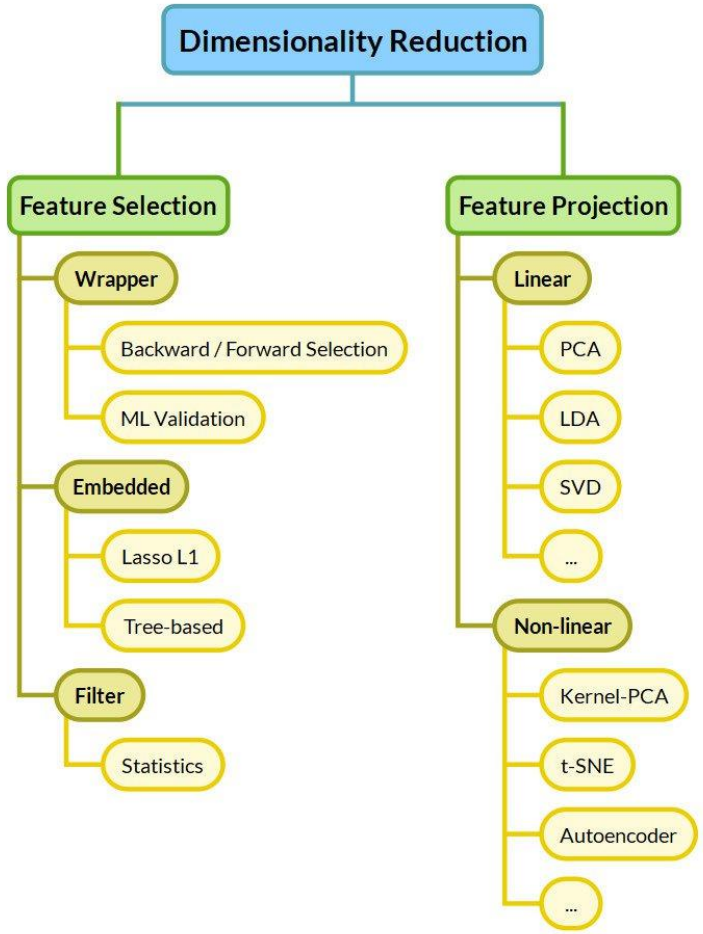

```
# Select and Standardize the Data
Here we will select our variables, taking only numeric columns.
PCA works best on scaled data. We will center and scale numeric columns:
```{r PCA_prep, echo=FALSE}
# Select only numeric columns (gene expression values).
df_numeric <- df %>% select(where(is.numeric))

# Standardize each numeric feature: mean 0, unit SD. Essential for PCA or clustering.
df_scaled <- df_numeric %>% mutate(across(everything(), ~ as.numeric(scale(.x))))

```
```

Dimensional reduction





https://www.sciencedirect.com/science/article/pii/S0925231218309469#fig0002

Dimensionality reduction: introduction

10.000 genes

Gene ID	ell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
TP53	80	50	63	36	60	0		99
EGFR	90	160	100	32	18	0		95
VEGFA	109	29	95	29	36	1		92
APOE	95	123	84	28	48	5		89
IL6	67	95	76	23	67	10		100
TGFB1	82	92	78	25	36	10		112
AKT1	71	124	81	26	37	0		113
MTHFR	70	97	90	24	91	0		100
...								
BRCA1	75	90	78	21	142	0		82

UMAP

2 "UMAP coordinates" or embeddings

UMAP	Cell_1	Cell_2	Cell_3	Cell_4	...	Cell_50000
UMAP_1	3	-8	9	-2		3
UMAP_2	4	-1	3	-1		4

t-SNE

2 "t-SNE coordinates" or embeddings

tSNE	Cell_1	Cell_2	Cell_3	Cell_4	...	Cell_50000
t-SNE_1	3	-8	9	-2		3
t-SNE_2	4	-1	3	-1		4

PCA

50 principal components

PC	Cell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
PC_1	3	-8	9	-2	5	2		3
PC_2	4	-1	3	-1	5	-4		4
PC_3	3	2	4	2	5	2		2
PC_4	-4	3	2	4	4	-1		4
PC_5	3	4	-4	5	3	2		5
PC_6	3	5	5	2	-4	3		2
PC_7	-6	2	5	-4	3	4		-4
PC_8	3	-4	-6	5	4	5		-6
...								
PC_50	5	2	-4	2	-4	3		2

Play with settings:
https://pair-code.github.io/understanding-umap/

https://biostatsquid.com/pca-umap-tsne-comparison/

PCA

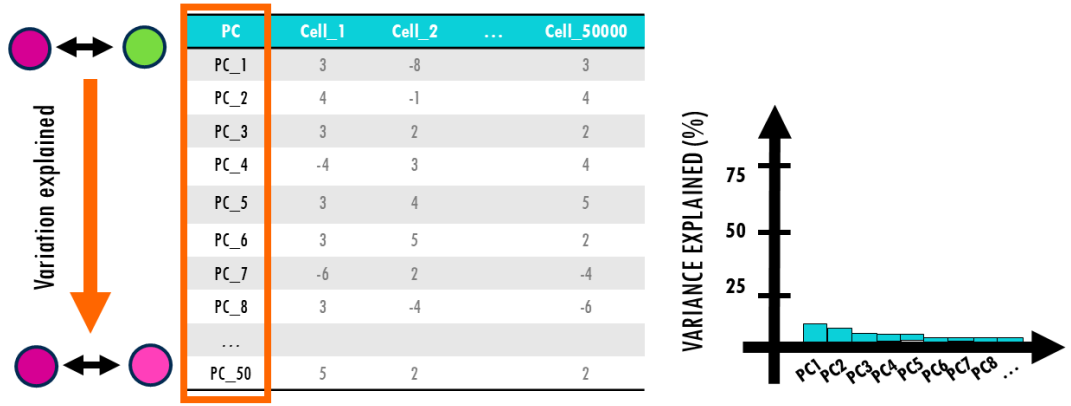
PCA

10.000 genes

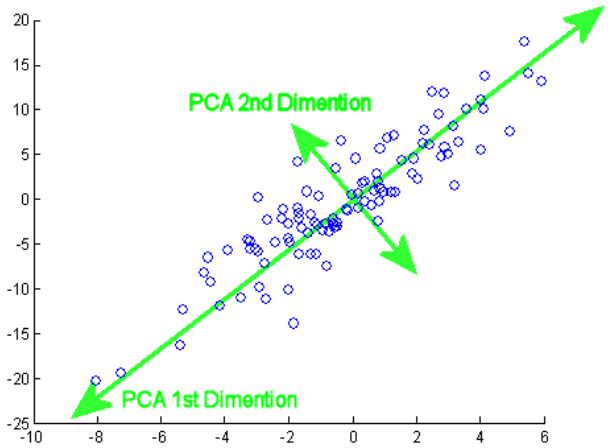
Gene ID	Cell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
TP53	80	50	63	36	60	0		99
EGFR	90	160	100	32	18	0		95
VEGFA	109	29	95	29	36	1		92
APOE	95	123	84	28	48	5		89
IL6	67	95	76	23	67	10		100
TGFBI	82	92	78	25	36	10		112
AKT1	71	124	81	26	37	0		113
MTHFR	70	97	90	24	91	0		100
...								
BRCA1	75	90	78	21	142	0		82

50 principal components

PC	Cell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
PC_1	3	-8	9	-2	5	2		3
PC_2	4	-1	3	-1	5	-4		4
PC_3	3	2	4	2	5	2		2
PC_4	-4	3	2	4	4	-1		4
PC_5	3	4	-4	5	3	2		5
PC_6	3	5	5	2	-4	3		2
PC_7	-6	2	5	-4	3	4		-4
PC_8	3	-4	-6	5	4	5		-6
...								
PC_50	5	2	-4	2	-4	3		2



COMPLEX DATASETS NEED MANY PRINCIPAL COMPONENTS TO EXPLAIN ENOUGH VARIABILITY



Principal Components are **ranked and orthogonal** (independent from each other): the first principal component (PC1) captures the largest possible variation across all cells. PC2 captures the second largest variation across all cells, that was not captured by PC1…, and so on.

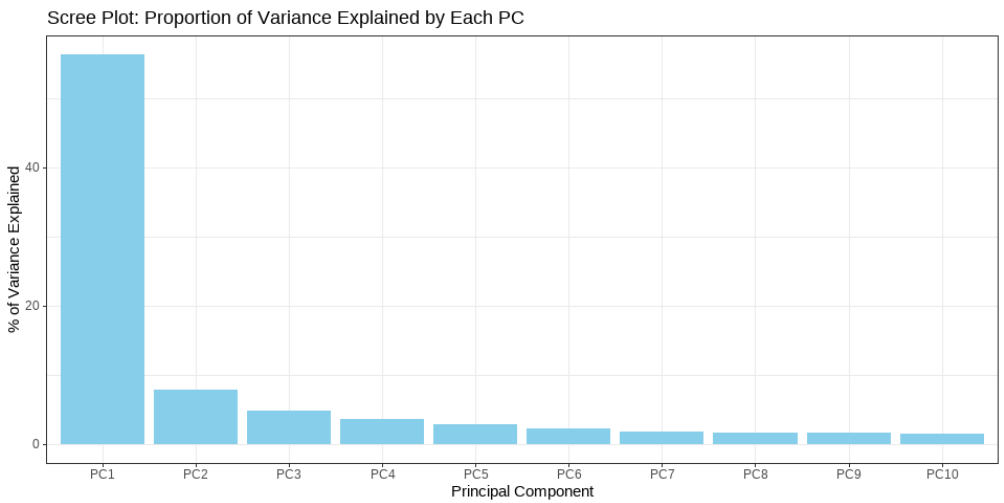
https://biostatsquid.com/pca-umap-tsne-comparison/

PCA

```{r PCA, echo=FALSE}
# Run principal component analysis on scaled data.
pca_result <- prcomp(df_scaled, center = FALSE, scale. = FALSE)
summary(pca_result) # Print variance explained by each principal component.

```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6    PC7     PC8     PC9
Standard deviation     9.1829 3.4445 2.67174 2.33756 2.07085 1.84366 1.6523 1.58523 1.53867
Proportion of Variance 0.5622 0.0791 0.04759 0.03643 0.02859 0.02266 0.0182 0.01675 0.01578
Cumulative Proportion  0.5622 0.6413 0.68886 0.72529 0.75388 0.77654 0.7947 0.81149 0.82727
```

```{r PCA, echo=FALSE, fig.width=10, fig.height=5}
# Plot variance explained by each principal component (base plot).
plot(pca_result)

max_PCA_plot <- 5# ITerate here max 10?

# More informative: Use ggplot to show individual and cumulative explained variance for first X PCs.
explained_var <- pca_result$sdev^2 / sum(pca_result$sdev^2) * 100
explained_df <- tibble(PC = paste0("PC", 1:length(explained_var)),
                       variance = explained_var) %>%
  mutate(PC = factor(PC, levels = PC),
         Cummulative_var = cumsum(variance)) # calculate cummulative sum

ggplot(explained_df %>% slice(1:max_PCA_plot),
       aes(x = PC, y = variance)) +
  geom_col(fill = "skyblue") +
  ylab("% of Variance Explained") +
  xlab("Principal Component") +
  ggtitle("Scree Plot: Proportion of Variance Explained by Each PC") +
  theme_bw()

ggplot(explained_df %>% slice(1:max_PCA_plot),
       aes(x = PC, y = Cummulative_var)) +
  geom_col(fill = "skyblue") +
  ylab("Cummulative Variance Explained") +
  xlab("Principal Component") +
  ggtitle("Scree Plot: Cummulative Variance Explained") +
  theme_bw()

```



PCA

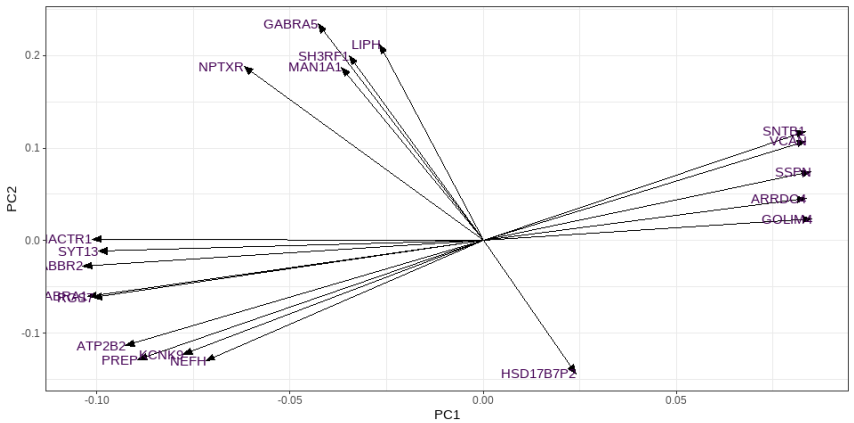
```{r PCA_plot, echo=FALSE, fig.width=10, fig.height=5}
# Extract PCA rotation matrix, which shows how features (genes) load on PCs.
pca_rotations_df <- pca_result %>%
  tidy(matrix = "rotation") %>%
  pivot_wider(names_from = "PC", names_prefix = "PC", values_from = "value") %>% # make the table longer, as wee need PC as columns
  select(column, PC1, PC2)

# Find top 5 positive/negative contributors for PC1/PC2.
pca_rotations_filtered_df <- pca_rotations_df %>%
  slice_max(order_by = PC1, n = 5, with_ties = FALSE) %>%
  bind_rows(
    pca_rotations_df %>%
      slice_min(order_by = PC1, n = 5, with_ties = FALSE),
    pca_rotations_df %>%
      slice_max(order_by = PC2, n = 5, with_ties = FALSE),
    pca_rotations_df %>%
      slice_min(order_by = PC2, n = 5, with_ties = FALSE),
  ) %>%
  ungroup() %>% unique()

arrow_style <- arrow(
  angle = 20, ends = "first", type = "closed", length = grid::unit(8, "pt")
)

# Plot gene loadings as arrows for PC1/PC2, names highlighted.
pca_rotations_filtered_df %>%
  ggplot(aes(PC1, PC2)) +
  geom_segment(xend = 0, yend = 0, arrow = arrow_style) +
  geom_text(
    aes(label = column),
    hjust = 1, nudge_x = 0,
    color = "#440154"
  ) + theme_bw()

```

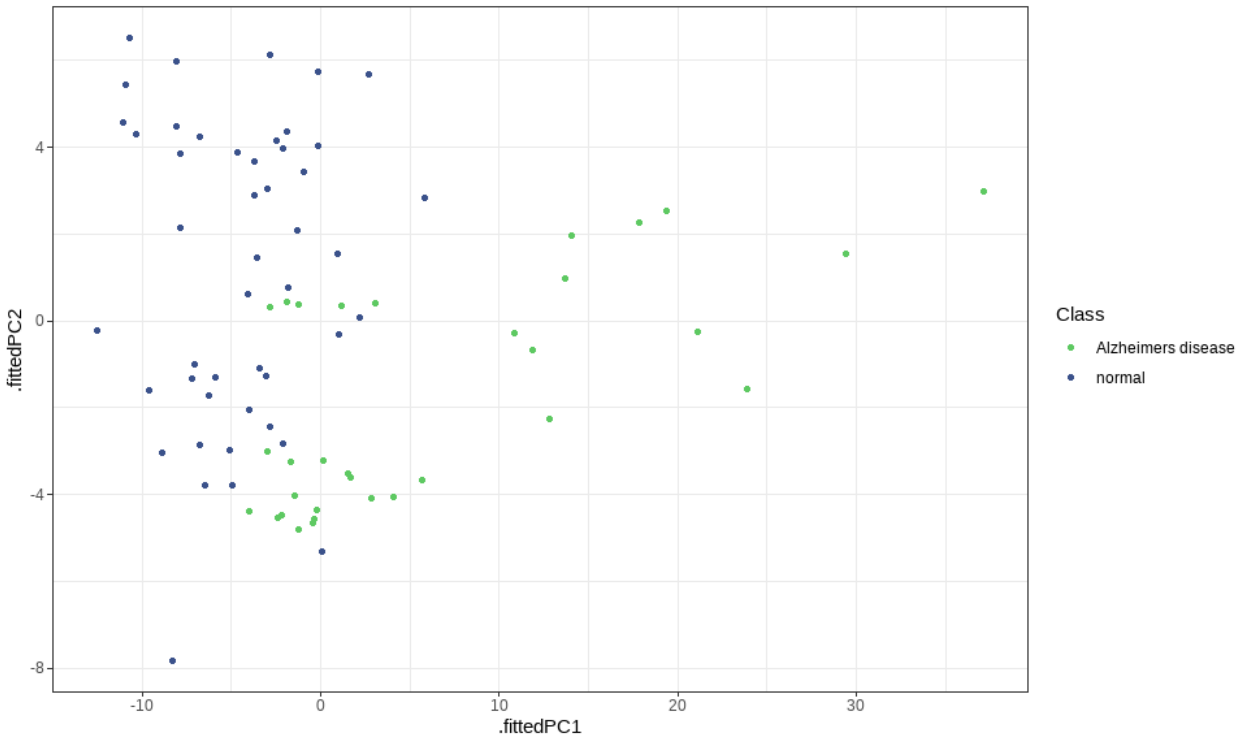


PCA

```
Purpose: Visualize how patients cluster by principal components. Color points by diagnosis.
```{r PCA_plot, echo=FALSE, fig.width=10, fig.height=6}
# basic plot
pca_with_df <- pca_result %>%
  augment(df) # Add original data to PCA result

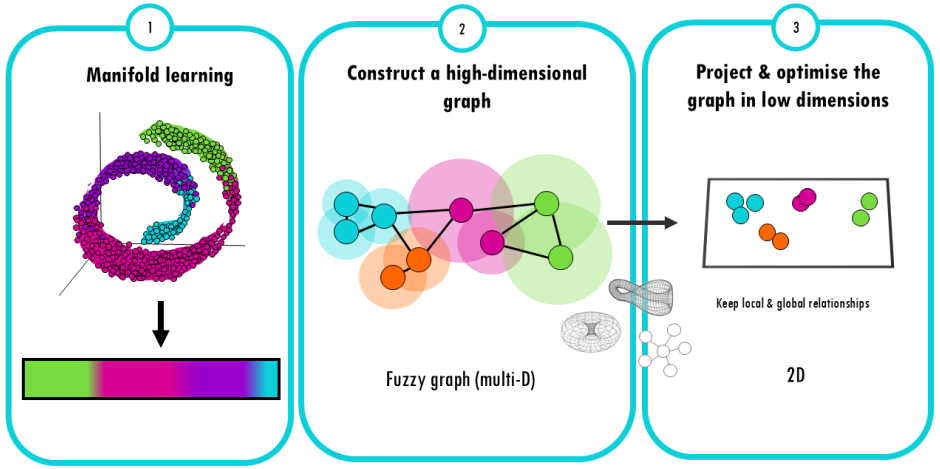
# Scatterplot: patients colored by disease status in PC1 vs PC2 space.
ggplot(pca_with_df, aes(.fittedPC1, .fittedPC2, color = Class)) +
  geom_point(size = 1.5) +
  scale_color_manual(
    values = c(`Alzheimers disease` = "#5ec962", normal = "#3b528b")
  ) + theme_bw()

```
```



UMAP

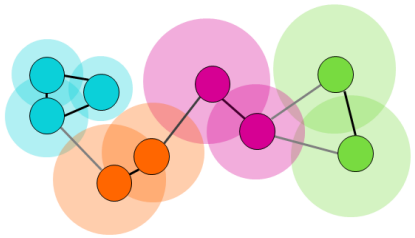
How does UMAP summarise many dimensions into 2?



How does UMAP construct a high-dimensional fuzzy graph?

1. Position the points (cells) in multi-dimensional space

n_neighbours = 2



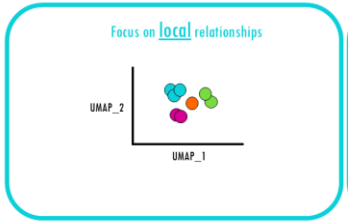
n_neighbours = 4

The hyperparameter n_neighbours is used to control the size of the radii

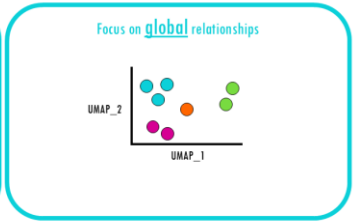
n_neighbours is a hyperparameter which controls how much of the local vs global structure is preserved

min_dist is the minimum distance between points in the low-dimensional space.

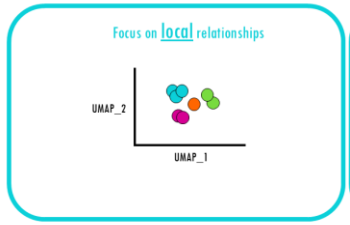
Low n_neighbours



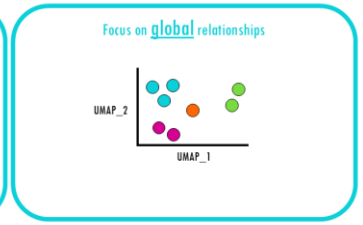
High n_neighbours



Low min_dist



High min_dist



Dimensionality reduction: introduction

PCA is deterministic, t-SNE and UMAP are stochastic

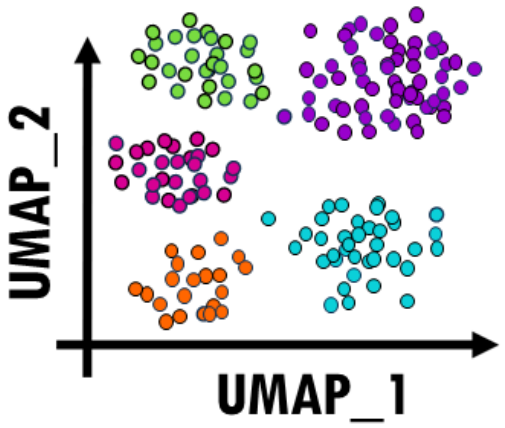
Same input

Gene ID	ell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
TP53	80	50	63	36	60	0		99
EGFR	90	160	100	32	18	0		95
VEGFA	109	29	95	29	36	1		92
APOE	95	123	84	28	48	5		89
IL6	67	95	76	23	67	10		100
TGFBI	82	92	78	25	36	10		112
AKT1	71	124	81	26	37	0		113
MTHFR	70	97	90	24	91	0		100
...								
BRCA1	75	90	78	21	142	0		82

UMAP

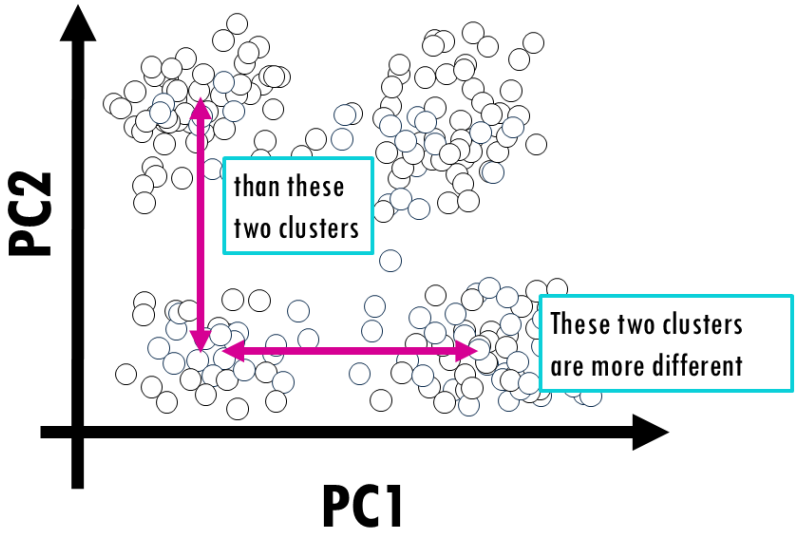
min.dist = 5,
n_neighbours = 10

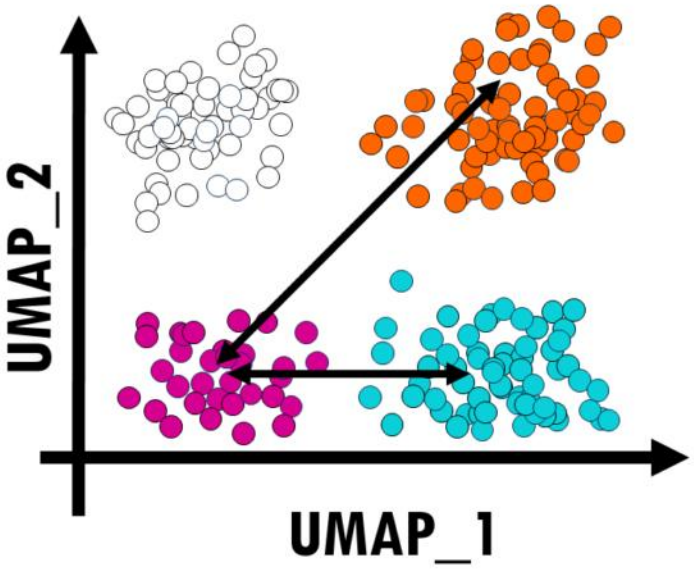
Slightly different outputs



https://biostatsquid.com/pca-umap-tsne-comparison/

Dimensionality reduction: introduction





The distances between cluster do not mean anything

https://biostatsquid.com/pca-umap-tsne-comparison/

Dimensionality reduction: PCA vs t-SNE/UMAP

	PCA	t-SNE	UMAP
Type of Method	Linear method (uses linear transformations)	Non-linear method (focuses on local structure)	Non-linear method (focuses on both local and global structure)
Focus	Maximizing variance (global structure)	Preserving local structure (neighborhoods)	Preserving both local and global structure
Preserves	Variance (overall spread of the data)	Local relationships (similarity between neighbors)	Local and global relationships (overall shape and clusters)
Output	Linear transformation of the data into principal components	2D or 3D representation that reflects local similarities	2D or 3D representation with more global structure
Scalability	Highly scalable (works well with large datasets)	Computationally expensive on large datasets	Scalable, faster than t-SNE, works well with large datasets
Speed	Fast	Slow, especially for large datasets	Faster than t-SNE, more scalable
Reproducibility	Very stable and deterministic	Results can vary with different runs	More stable than t-SNE, but less deterministic than PCA
Interpretability	Results are easy to interpret (principal components)	Results are harder to interpret (abstract relationships)	Results are more interpretable than t-SNE, but not as clear as PCA
Best for	When you want to preserve overall variance and reduce dimensionality linearly (e.g., for feature extraction, noise reduction)	When you want to explore local structure and identify clusters in the data	When you want to preserve both local and global structure, especially in complex, large datasets

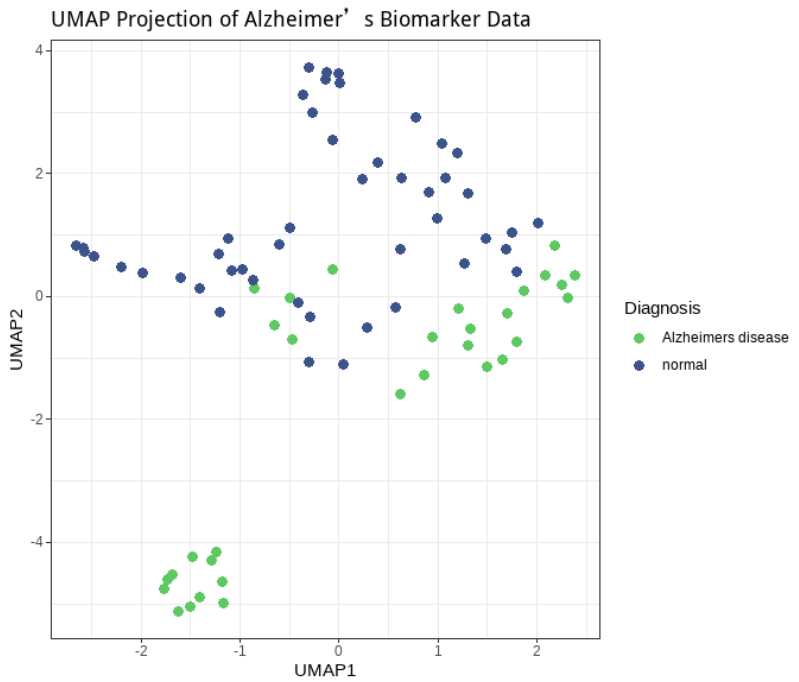
https://biostatsquid.com/pca-umap-tsne-comparison/

https://www.youtube.com/watch?v=aBUuNHt3YsA

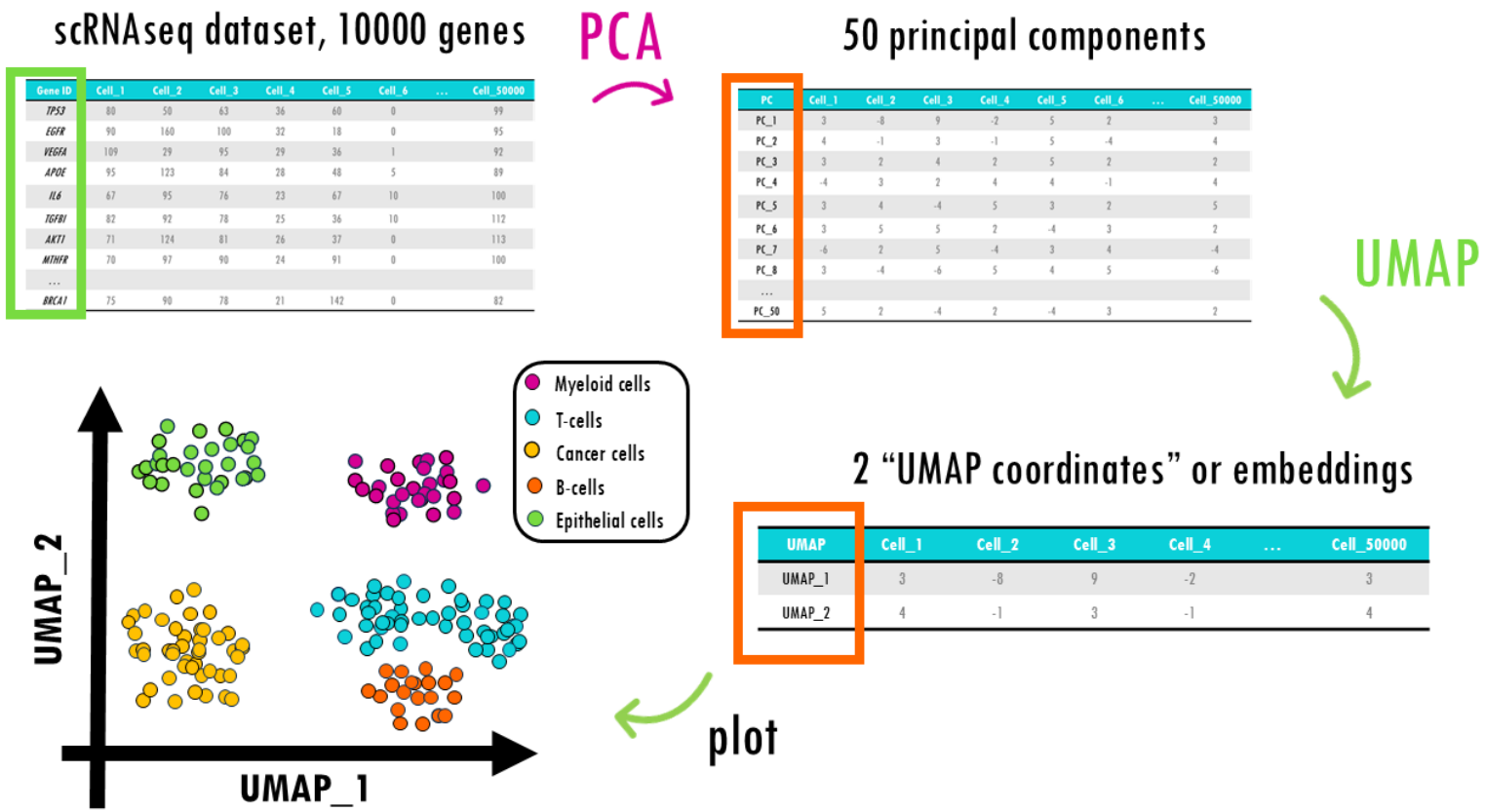
UMAP

```{r UMAP_run, echo=FALSE, fig.width=10, fig.height=5}
# Run UMAP
set.seed(42)
umap_res <- umap(df_scaled ,
                 n_components = 2,
                 n_neighbors = 8, min_dist = 0.1, metric = "euclidean")

# Make table for plotting, add diagnosis label for each sample.
umap_df <- as_tibble(umap_res, .name_repair = "minimal") %>%
  setNames(c("UMAP1", "UMAP2")) %>%
  mutate(clinical_diagnosis = df$Class)

```

```{r UMAP_run, echo=FALSE, fig.width=7, fig.height=6}
# Plot UMAP results colored by diagnosis
plot1 <- ggplot(umap_df, aes(UMAP1, UMAP2, color = clinical_diagnosis)) +
  labs(title = "UMAP Projection of Alzheimer's Biomarker Data",
       color = "Diagnosis")  +
  geom_point(size = 3) +
  scale_color_manual(
     values = c(`Alzheimers disease` = "#5ec962", normal = "#3b528b")
  ) + theme_bw()
plot1
```



Dimensionality reduction: PCA + t-SNE/UMAP



Gene ID	Cell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
TP53	80	50	63	36	60	0		99
EGFR	90	160	100	32	18	0		95
VEGFA	109	29	95	29	36	1		92
APOE	95	123	84	28	48	5		89
IL6	67	95	76	23	67	10		100
TGFBI	82	92	78	25	36	10		112
AKT1	71	124	81	26	37	0		113
MTHFR	70	97	90	24	91	0		100
...								
BRCA1	75	90	78	21	142	0		82

PC	Cell_1	Cell_2	Cell_3	Cell_4	Cell_5	Cell_6	...	Cell_50000
PC_1	3	-8	9	-2	5	2		3
PC_2	4	-1	3	-1	5	-4		4
PC_3	3	2	4	2	5	2		2
PC_4	-4	3	2	4	4	-1		4
PC_5	3	4	-4	5	3	2		5
PC_6	3	5	5	2	-4	3		2
PC_7	-6	2	5	-4	3	4		-4
PC_8	3	-4	-6	5	4	5		-6
...								
PC_50	5	2	-4	2	-4	3		2

UMAP	Cell_1	Cell_2	Cell_3	Cell_4	...	Cell_50000
UMAP_1	3	-8	9	-2		3
UMAP_2	4	-1	3	-1		4

https://biostatsquid.com/pca-umap-tsne-comparison/

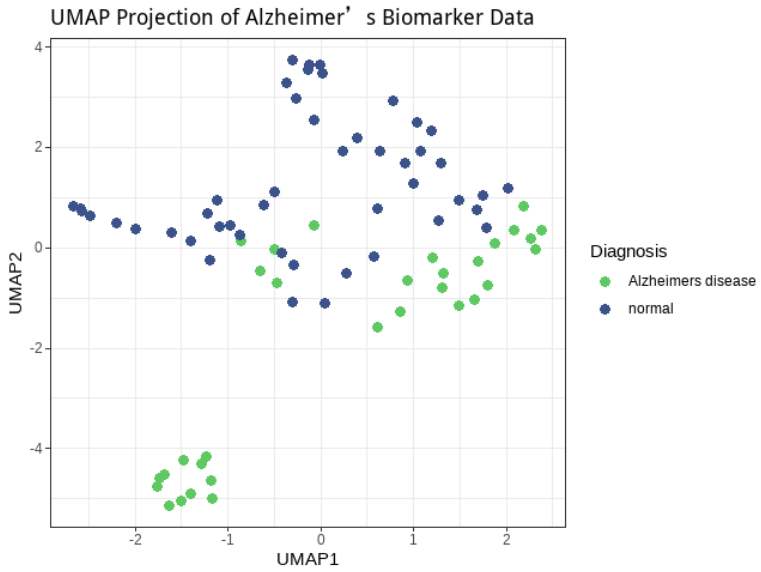
https://www.youtube.com/watch?v=aBUuNHt3YsA

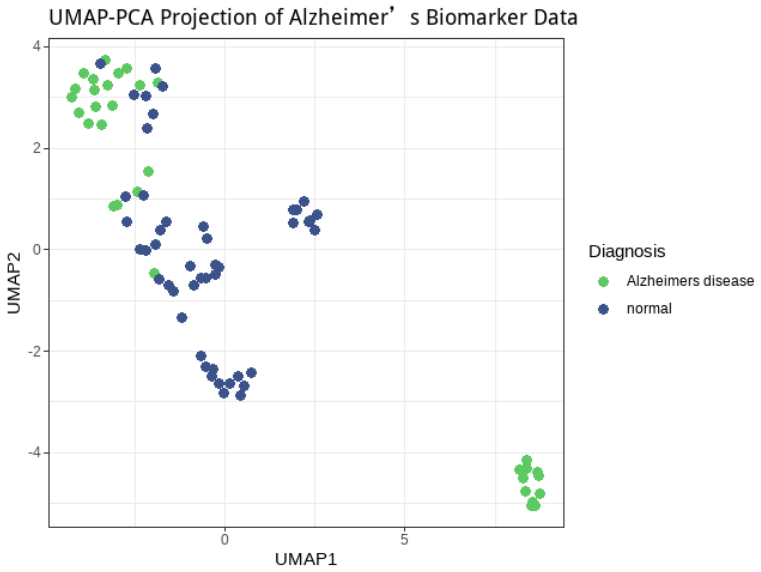
PCA -> UMAP

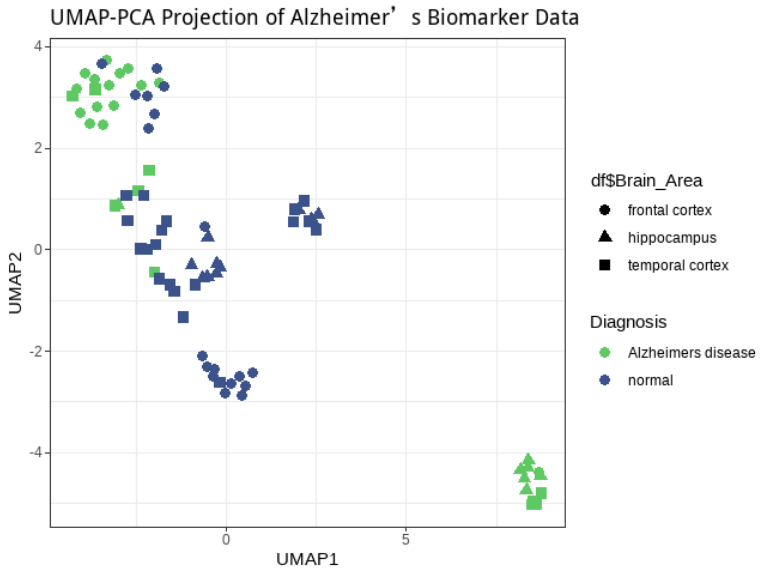
Purpose: Apply UMAP after reducing to 4 principal components. May improve

```{r UMAP_run, echo=FALSE, fig.width=10, fig.height=5}
# Run UMAP
set.seed(42)
umap_PCA_res <- umap(pca_with_df %>% select(starts_with(".fitted")),
                     n_components = 2, pca = 4,
                     n_neighbors = 8, min_dist = 0.1, metric = "euclidean") #

# Combine UMAP results with diagnosis labels
umap_PCA_df <- as_tibble(umap_PCA_res, .name_repair = "minimal") %>%
  setNames(c("UMAP1", "UMAP2")) %>%
  mutate(clinical_diagnosis = df$class)

```

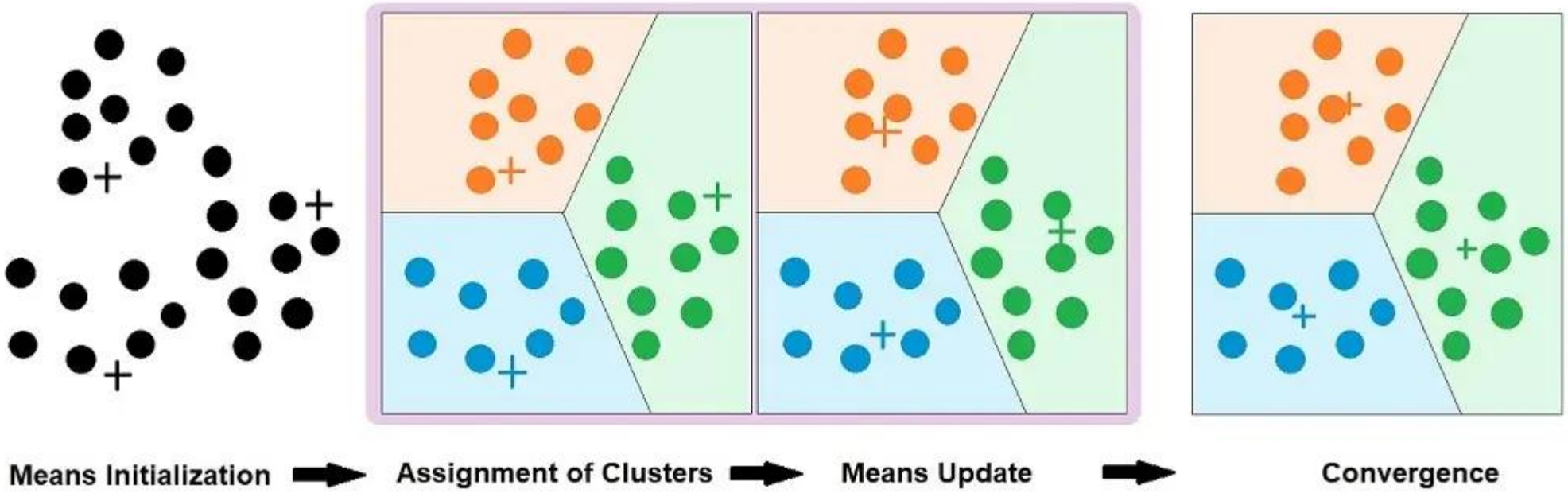
```{r UMAP_run, echo=FALSE, fig.width=20, fig.height=5}
# Plot UMAP results colored by diagnosis
plot2 <- ggplot(umap_PCA_df, aes(UMAP1, UMAP2, color = clinical_diagnosis)) +
  labs(title = "UMAP-PCA Projection of Alzheimer's Biomarker Data",
       color = "Diagnosis")  +
  geom_point(size = 3) +
  scale_color_manual(
    values = c(`Alzheimers disease` = "#5ec962", normal = "#3b528b")
  ) + theme_bw()

plot2A <- ggplot(umap_PCA_df, aes(UMAP1, UMAP2, color = clinical_diagnosis, shape = df$Brain_Area)) +
  labs(title = "UMAP-PCA Projection of Alzheimer's Biomarker Data",
       color = "Diagnosis")  +
  geom_point(size = 3) +
  scale_color_manual(
    values = c(`Alzheimers disease` = "#5ec962", normal = "#3b528b")
  ) + theme_bw()

grid.arrange(plot1, plot2, plot2A, ncol=3)
```





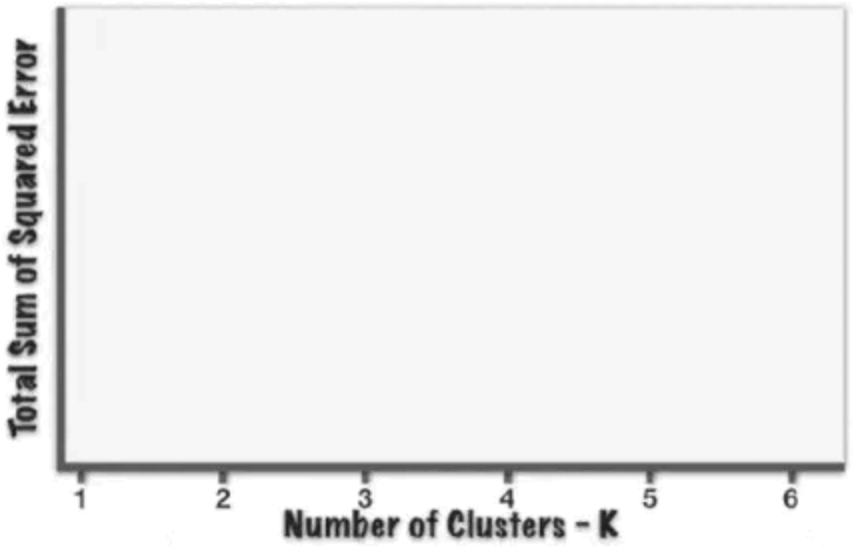


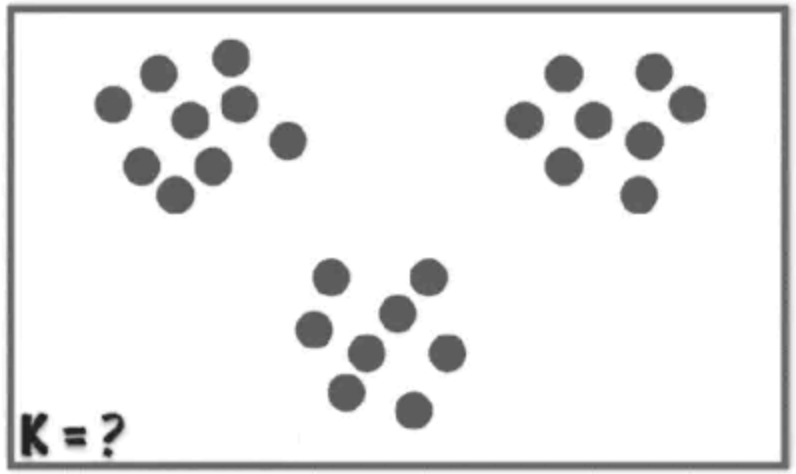
Clustering: k-means



https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/k-means-clustering/

Clustering: k-means

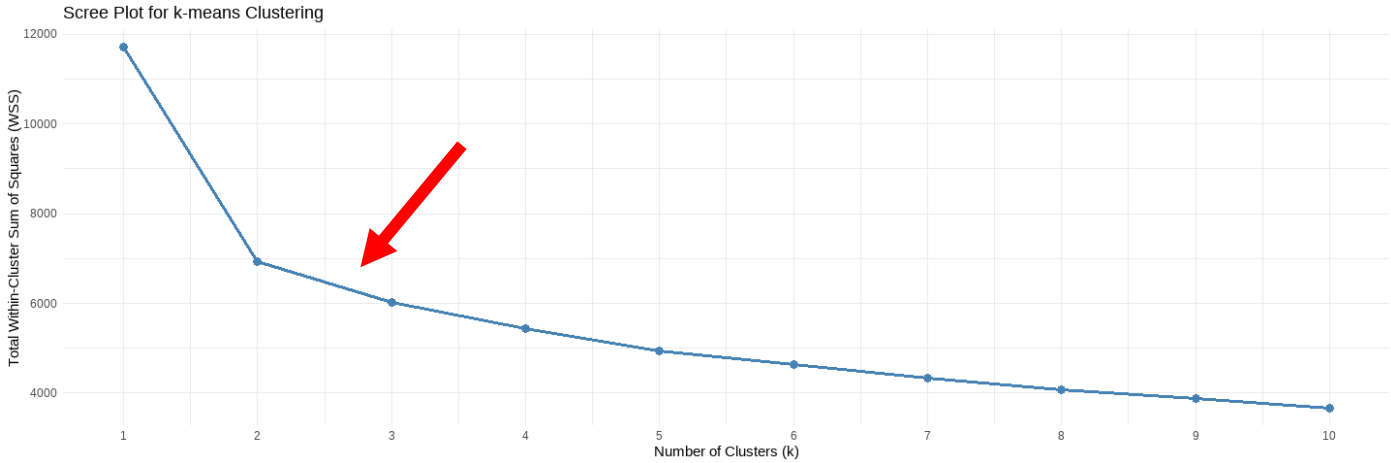




https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/k-means-clustering/

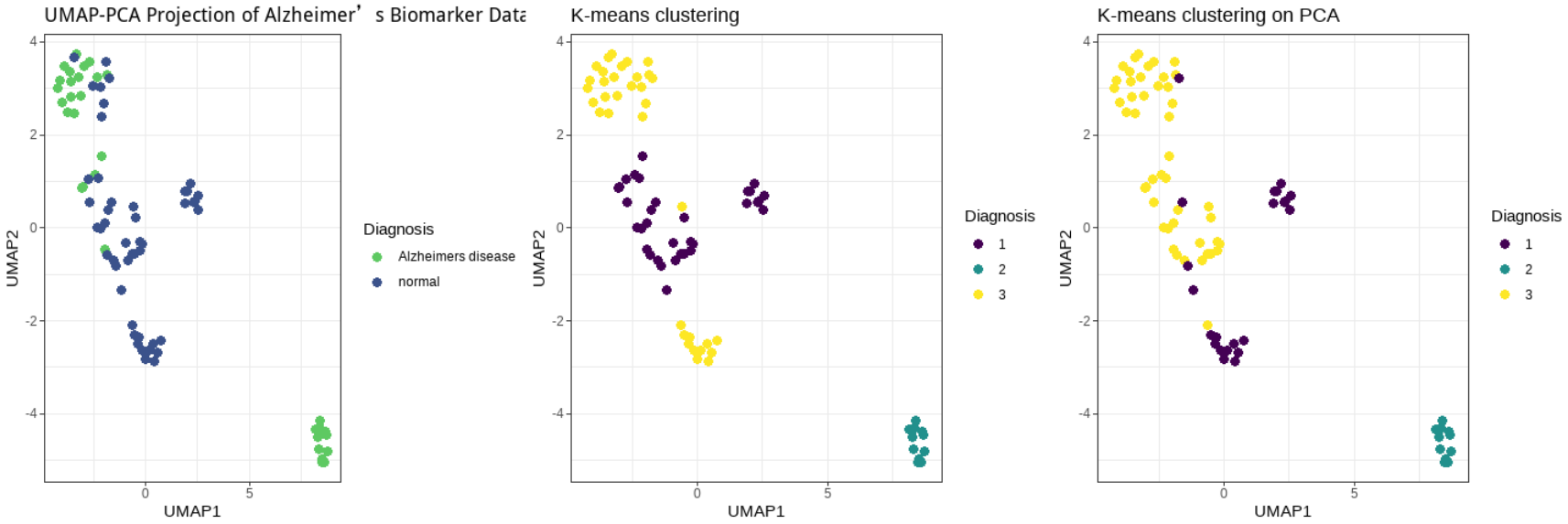
K-means

```{r K-means_K, echo=FALSE, fig.width=15, fig.height=5}
# Range of k values to evaluate
k_values <- 1:10

# Calculate total within-cluster sum of squares (WSS) for each k
wss <- map_dbl(k_values, function(k) {
  km <- kmeans(df_scaled, centers = k, nstart = 25)
  km$tot.withinss
})

# Create scree (elbow) plot
wss_df <- tibble(k = k_values, WSS = wss)

ggplot(wss_df, aes(x = k, y = WSS)) +
  geom_line(color = "steelblue", linewidth = 1.2) +
  geom_point(color = "steelblue", size = 3) +
  scale_x_continuous(breaks = k_values) +
  labs(title = "Scree Plot for k-means Clustering",
       x = "Number of Clusters (k)",
       y = "Total Within-Cluster Sum of Squares (WSS)") +
  theme_minimal()

```

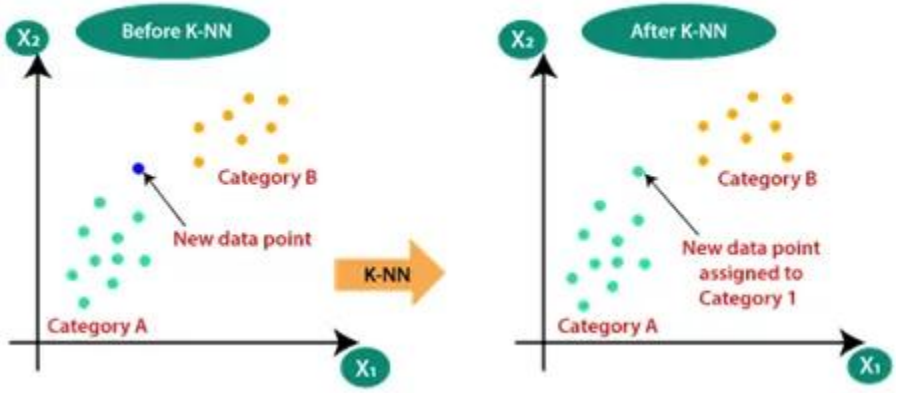


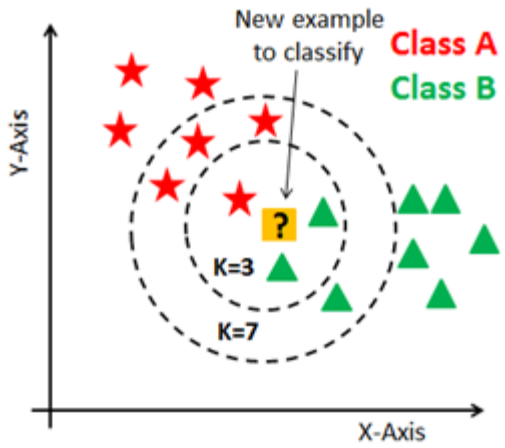
K-means

```{r K-means, echo=FALSE, fig.width=15, fig.height=5}
set.seed(42)
k <- 2 # Iterate here
kmeans_res <- kmeans(df_scaled, centers = k) # you run the k-means on the original data not UMAP projection
df_kmeans <- umap_PCA_df %>% mutate(cluster = factor(kmeans_res$cluster))

# Run k-means clustering on the first 6 PCA components
n_PCA <- 3# Iterate here
kmeans_res_PCA <- kmeans(pca_with_df %>% select(one_of(paste0(".fittedPC", 1:n_PCA))), centers = k) # you run the k-means on the original data not UMAP projection
df_kmeans_PCA <- umap_PCA_df %>% mutate(cluster = factor(kmeans_res_PCA$cluster))


plot3 <- ggplot(df_kmeans, aes(UMAP1, UMAP2, color = cluster)) +
  labs(title = "K-means clustering",
       color = "Diagnosis")  +
  geom_point(size = 3) +
  scale_color_viridis_d() + theme_bw()

# Plot clusters on UMAP-PCA visualization
plot4 <- ggplot(df_kmeans_PCA, aes(UMAP1, UMAP2, color = cluster)) +
  labs(title = "K-means clustering on PCA",
       color = "Diagnosis")  +
  geom_point(size = 3) +
  scale_color_viridis_d() + theme_bw()

grid.arrange(plot2, plot3, plot4, ncol=3)


```



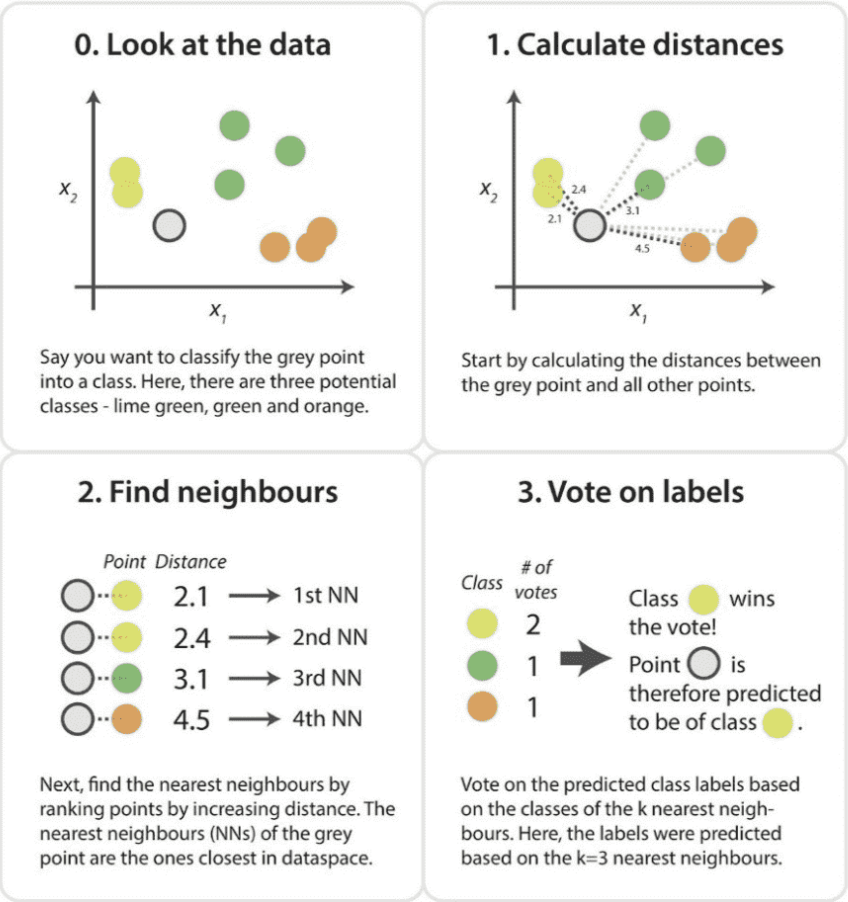
Clustering – shared nearest neighbours





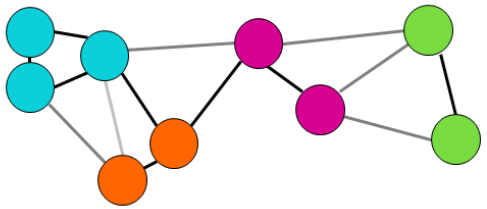
https://medium.com/@sravanthi.dande/k-nearest-neighbor-knn-algorithm-its-metrics-42c3f196fdda

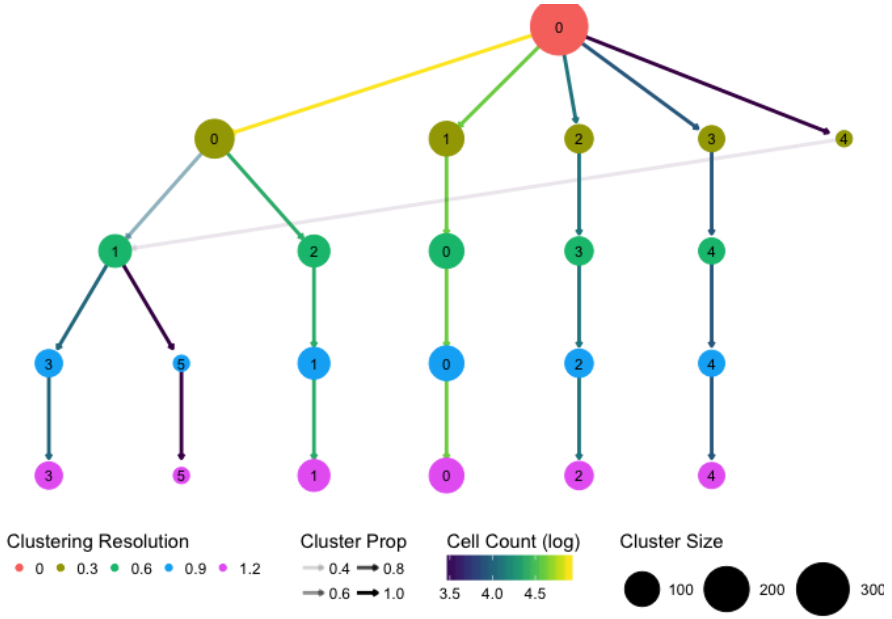
Clustering – shared nearest neighbours



Cell ID	Cell_1	Cell_2	Cell_3	...
Cell_1	1	1	0	0
Cell_2	1	1	1	0
Cell_3	0	1	1	0
...	0	0	0	0
Cell_50000	1	1	0	0

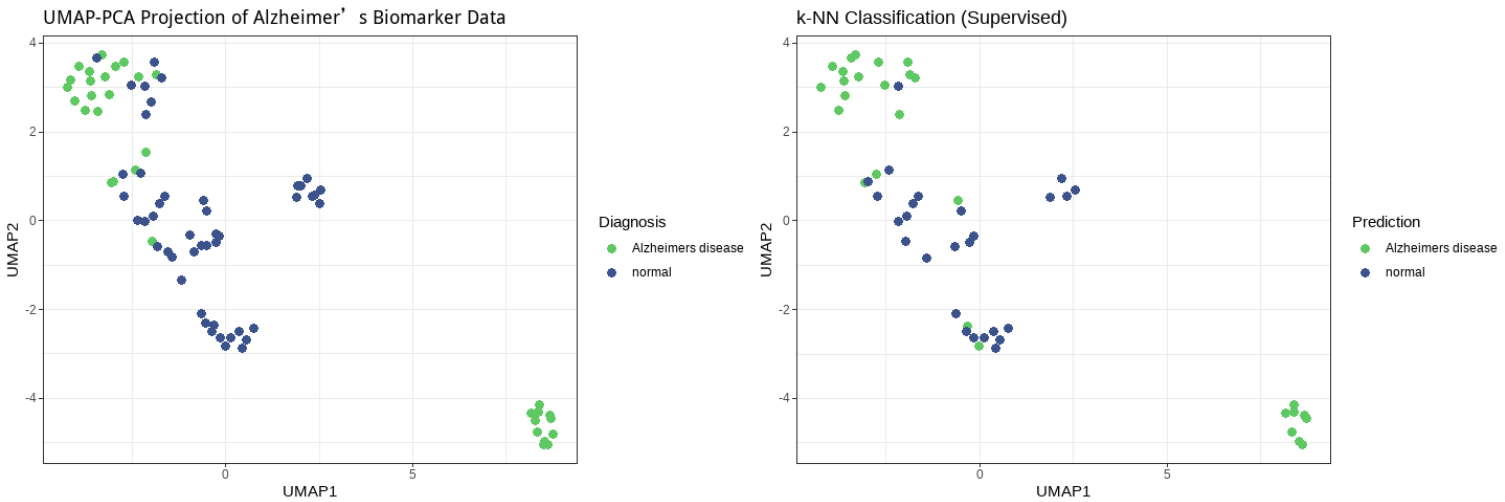
1 = connected
0 = not connected





K-NN

```{r K-NN, echo=FALSE, fig.width=15, fig.height=5}
set.seed(42)

train_index <- createDataPartition(df$Class, p = 0.3, list = FALSE)
train <- df_scaled[train_index, ]
test <- df_scaled[-train_index, ]
train_labels <- df$Class[train_index]
test_labels <- df$Class[-train_index]

k_nn <- 3
knn_pred <- knn(train, test, train_labels, k = k_nn)

# Visualize k-NN predictions in PCA space
test_pca <- umap_PCA_df[-train_index, ]
test_pca <- test_pca %>% mutate(pred = knn_pred, true = test_labels)

plot5<- ggplot(test_pca, aes(UMAP1, UMAP2, color = pred)) +
  labs(title = "k-NN Classification (Supervised)",
       color = "Prediction")  +
  geom_point(size = 1.5) +
  scale_color_manual(
    values = c(`Alzheimers disease` = "#5ec962", normal = "#3b528b")
  ) + theme_bw()


grid.arrange(plot2, plot5, ncol=2)

```



Thank you for the attention