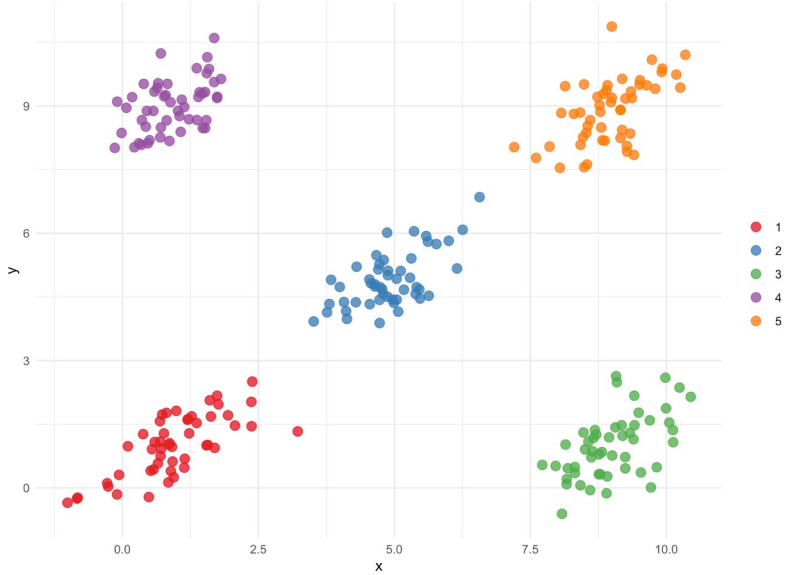
Clustering concepts and correlation

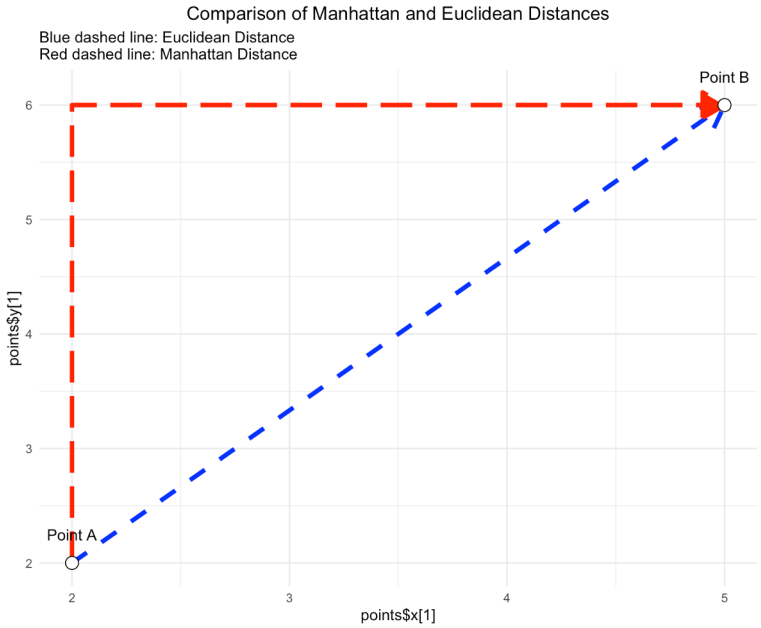
Clustering

- How do we group similar data points?
  - Patients
  - Cells
  - etc..
- Goal is a set of labels for each observation
  - Thinking back to dataframe, an observation is a row
- Clustering is an unsupervised approach
  - As opposed to classification or regression
  - Doesn’t require a set of targets
- Many clustering algorithms
  - K-means
  - Hierarchical Clustering
  - DBSCAN
  - Louvain/Leiden



Distance

- How do we decide if two points are similar?
  - **Euclidean** distance.
  - Shortest path (L2)
- Other distance metrics
  - **Manhattan** (L1)
    - Robust to outliers
    - Less sensitive to difference
  - Cosine distance
    - High dimension and sparse
    - Normalization (Angle)
  - String distance
    - Useful for sequences
    - Levenshtein
    - How many character changes to make two strings equivalent
- `dist` function
  - `dist(X,method="euclidean")`
    - X = matrix, dataframe, etc..
    - returns a dist object



`dist` in R

```
# Load the iris dataset
data(iris)
iris_numeric <- iris[, 1:4]  # Exclude the species column

# Euclidean distance
euclidean_distances <- dist(iris_numeric, method = "euclidean")

# Manhattan distance
manhattan_distances <- dist(iris_numeric, method = "manhattan")
```

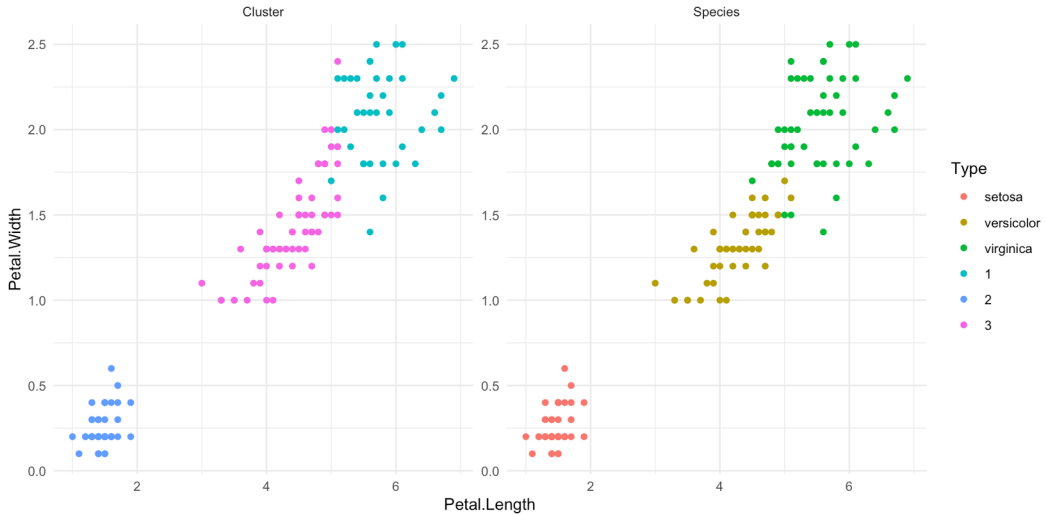
K-means Clustering

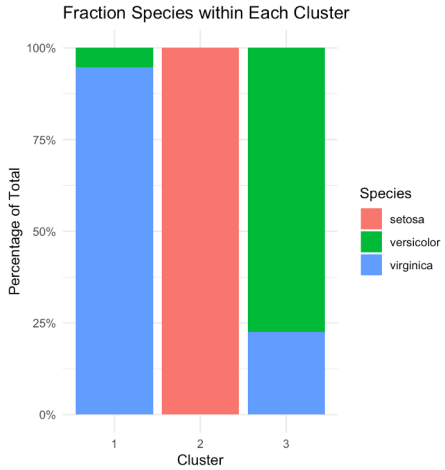
Algorithm:

- Pick a set of centers
- Assign all points to the closest center
- Update the centers to the mean of the points assigned
- Repeat until convergence

Using the `kmeans()` function from stats package.

- `kmeans(data,centers, iter.max,..)`
  - centers is the number of clusters (k)
  - iter.max is the number of times the algorithm loops
- Returns an object with:
  - cluster: vector
  - centers: matrix
- Plot our results





Hierarchical Clustering

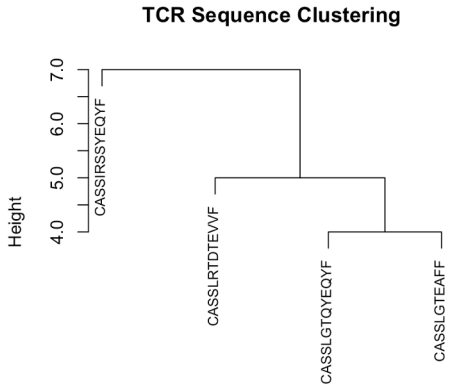
Algorithm:

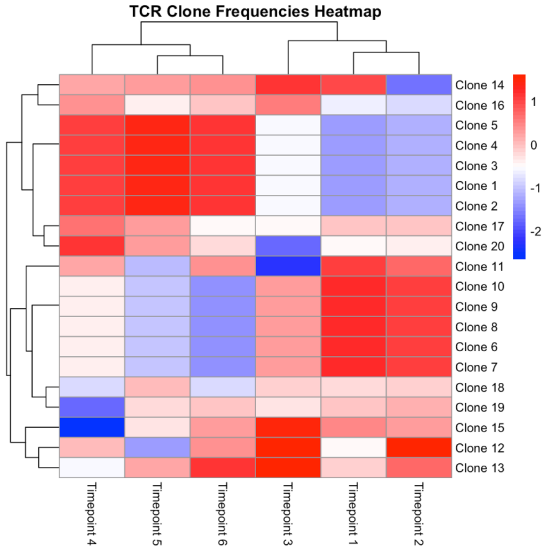
- Every data point is a cluster
- Compute distance between all clusters (using *dist*).
- At each step, join the two closest clusters.
- Recompute distances on n-1 clusters.
- End when only one cluster remains.

This is easily viewed as a **dendrogram**.

Using `hclust` function in R:

- `hclust(data, method="single")`
- data: output of dist
- method = linkage
  - single
  - complete
  - ...
- Tree cutting
  - `cutree`(obj, k=4)





Heatmaps with Hierarchical Clustering

- Organizing observations by similarity (or distance) allows easily visualization of trends.
- Can use the `pheatmap` function
  - Primary input: matrix (numeric)
  - Clustering options
  - Color schemes
  - Annotations
  - Scaling
  - Dendrogram

```
# Creating a heatmap with clustering and annotation
pheatmap(transposed_gene_expression,
         show_rownames = TRUE,
         show_colnames = FALSE,
         clustering_distance_rows = "euclidean",
         cluster_rows  = TRUE,
         cluster_cols  = FALSE,
         main = "Heatmap of Gene Expression with Clustering")
```

Network-based Clustering

Leiden

- Detecting clusters in network data.
- `leiden` package
- Doesn’t require a pre-specified number of clusters
  - Resolution parameter
- Use in single cell RNA-seq:
  - Construct a k-nearest neighbor (k-NN) graph or a similarity matrix based on the gene expression profiles of individual cells. Each cell is represented as a node in the graph, and edges connect cells that are similar to each other.

Dimensionality Reduction

What is dimensionality reduction?

- Reduce the number of variables by obtaining a set of representative features.

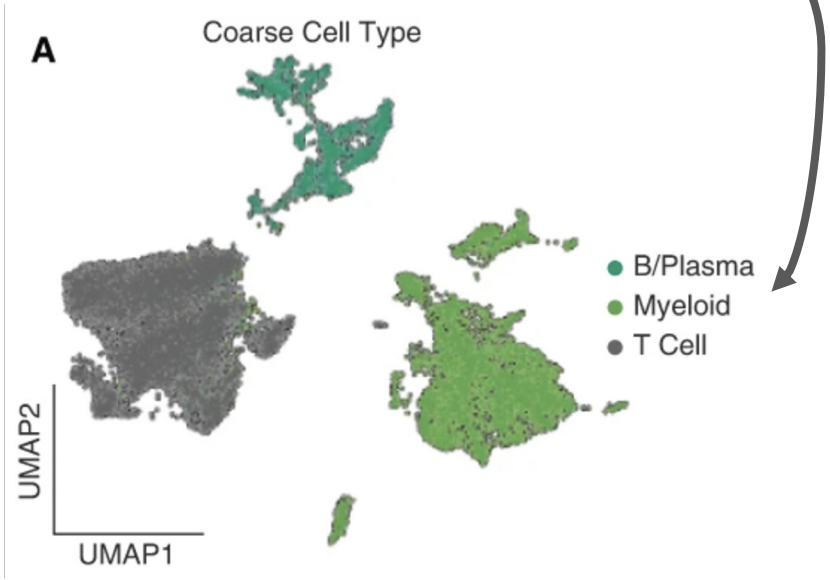
Why do we do this?

- Visualization of high dimensional data
- Removing noise
- Efficiency

Methods

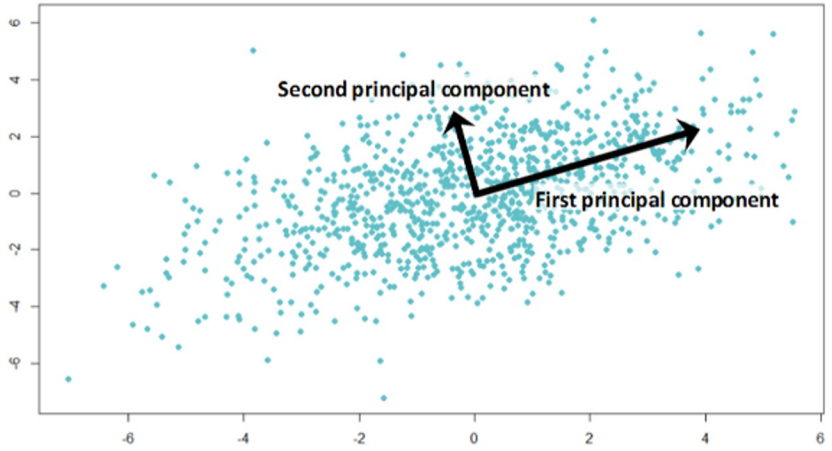
- Principal component analysis (PCA)
- Diffusion component analysis
- Autoencoders
- UMAP / tSNE

	Cell 1	Cell2	Cell3	Cell4	...
Gene 1	0	0	3	10	
Gene 2	24	0	41	12	
Gene 3	175	284	93	162	
Gene 4	0	0	0	0	
Gene 5	36	0	32	21	
...	...	...	...	...	



Principal Component Analysis (PCA)

- Common method for reduction to a set of explanatory features.
  - “Principal Components”
- Each component explains differences, or variation, in the dataset.
- Components are ordered by how much variance they explain.
  - The first principal component explains the most variance.



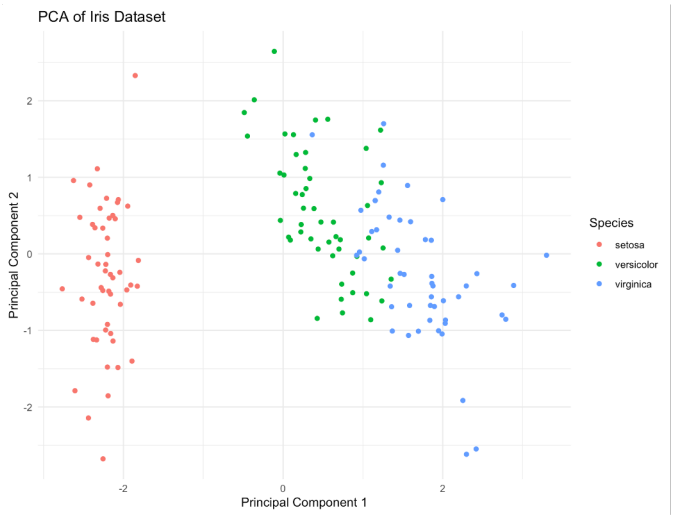
PCA in R

```
prcomp(data, center = TRUE, scale. = TRUE)
```

- data: numeric matrix or dataframe
- *center*
  - subtracts the mean per variable
- *scale*
  - standardize each column to have zero mean and unit variance
- *rank*
  - number of PCs to compute
- We can use summary to view components
- We can cluster PCs!

```
> print(summary(pca_results))
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

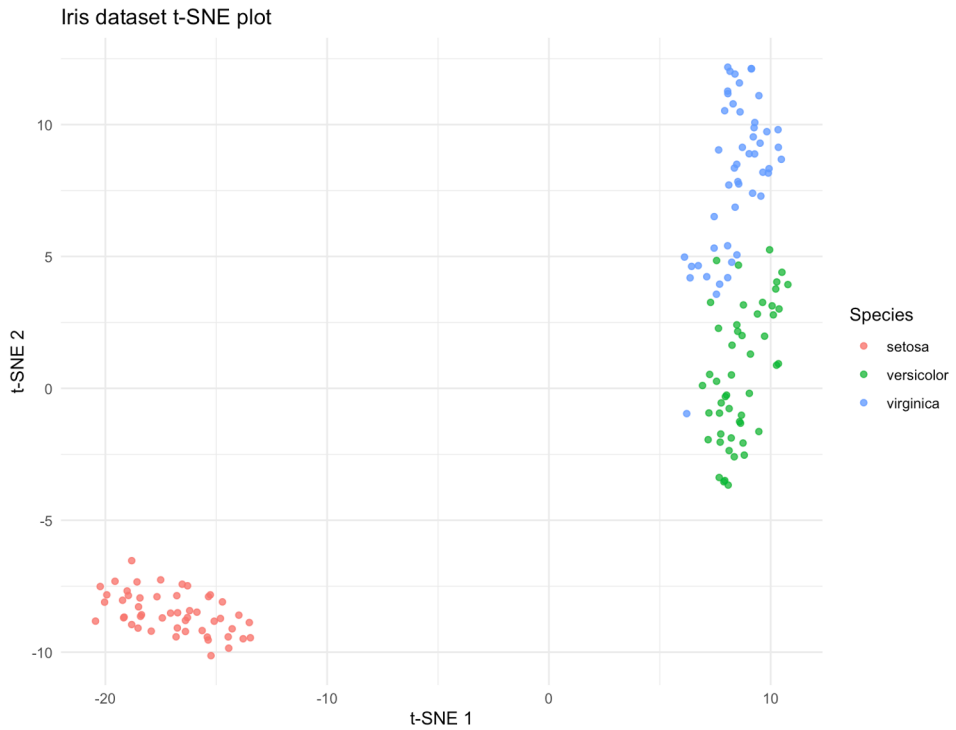
```r
# Load necessary packages
library(ggplot2)

# Load data
data(iris)
iris_data <- iris[, 1:4]

# PCA
pca_results <- prcomp(iris_data, center = TRUE, scale. = TRUE)
print(summary(pca_results))


# Scatter plot of the first two PCs
pc_df <- data.frame(PC1 = pca_results$x[,1], PC2 = pca_results$x[,2], Species = iris$Species)
ggplot(pc_df, aes(x = PC1, y = PC2, color = Species)) +
    geom_point() +
    labs(title = "PCA of Iris Dataset",
         x = "Principal Component 1",
         y = "Principal Component 2") +
    theme_minimal()
```



t-SNE

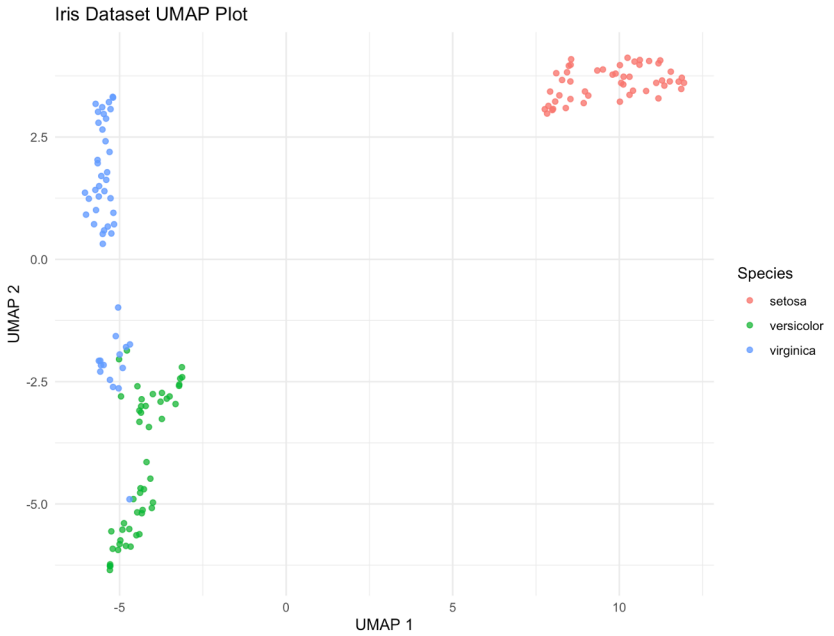
t-Distributed Stochastic Neighbor Embedding

- Attempts to preserve local structure between points in the high-dimensional state in the lower dimensional representation.
- Converts euclidean distances to probabilities in the high and low dimensions and minimizes the difference between distributions.
- Great for very complex data.
- At the expense of global structure.
- No linear mapping from low dimension to original data - unlike PCA.
- Can use Rtsne
  - `tsne_results <- Rtsne(data, dims = 2, perplexity = 30, verbose = TRUE)`
  - perplexity: hyperparameter that balances attention between local and global structure
  - dims: number of components



UMAP

- Uniform Manifold Approximation and Projection
- Similar to t-SNE, UMAP focuses on preserving the local structure of the data but also tries to retain more of the global structure.
- Does this by assuming a uniform distribution of data points.
- Faster than t-SNE
- Non-linear
- Can use `umap` library
  - `umap_results <- umap(data)`
  - n_neighbors: increasing preserves more global structure, computationally expensive
  - min_dist: controls the absolute min dist between points in embedding.
  - method: distance metric.. “euclidean”, etc.

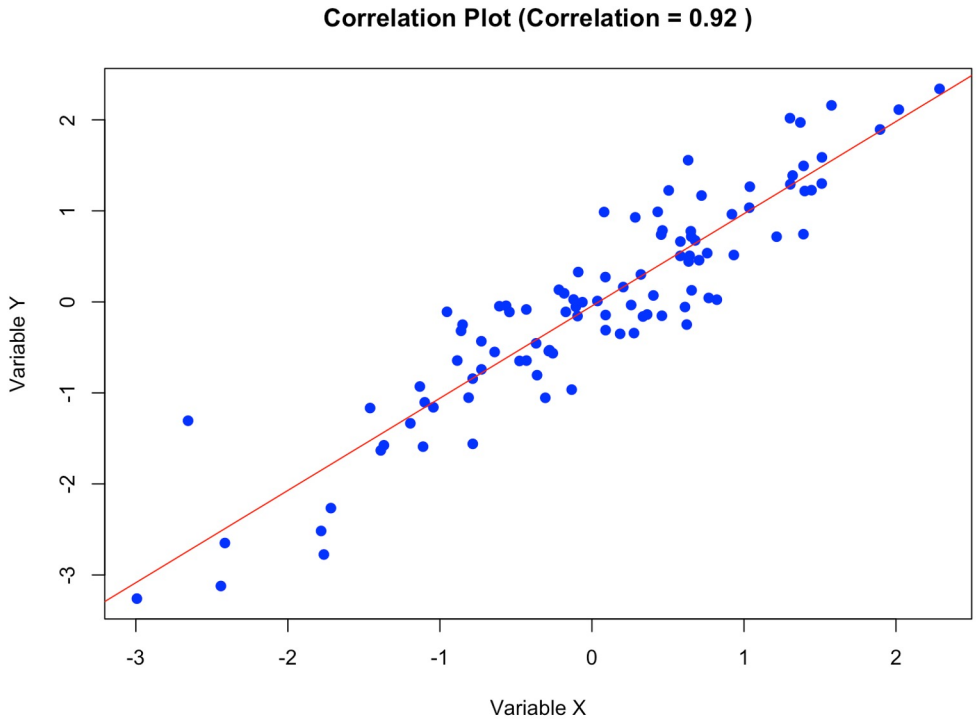


Other Dimensionality Reduction Methods

1. **Multidimensional Scaling (MDS)**:
   - MDS is a classical technique for dimensionality reduction that aims to preserve the pairwise distances between data points in the low-dimensional space as much as possible. It's available in R through functions like `cmdscale()`.
2. **Isomap**:
   - Isomap is a nonlinear dimensionality reduction method that focuses on preserving the geodesic distances (i.e., distances along the manifold) between data points. It's implemented in R through packages like `lle`.
3. **Locally Linear Embedding (LLE)**:
   - LLE is another nonlinear dimensionality reduction method that seeks to preserve local relationships between data points. It's useful for uncovering the underlying manifold structure of high-dimensional data. R implementations are available in packages like `lle`.
4. **Autoencoders**:
   - Autoencoders are a type of neural network architecture used for dimensionality reduction and feature learning. They consist of an encoder network that compresses the input data into a lower-dimensional representation and a decoder network that reconstructs the original data from the compressed representation. Various neural network libraries in R, such as `keras` and `torch`, can be used to implement autoencoders.
5. **Sparse Principal Component Analysis (Sparse PCA)**:
   - Sparse PCA is an extension of PCA that introduces sparsity constraints on the loadings matrix, resulting in a more interpretable representation. It's useful for identifying a small number of important features in high-dimensional data. R provides implementations of Sparse PCA in packages like `elasticnet`, `pcaMethods`, and `irlba`.
6. **Non-negative Matrix Factorization (NMF)**:
   - NMF is a dimensionality reduction technique that decomposes a non-negative data matrix into two lower-dimensional matrices, one of which contains only non-negative values. It's commonly used for feature extraction and topic modeling. R packages like `NMF` and `nmf` provide implementations of NMF algorithms.
7. **Independent Component Analysis (ICA)**:
   - ICA is a method for separating a multivariate signal into additive, independent components. It's useful for blind source separation and finding hidden factors in data. R implementations of ICA can be found in packages like `fastICA`.

Correlation

- Statistical measure that describes the strength and direction of the relationship between two variables.
- Quantifies how much changes in one variable correspond to changes in another.
- Correlation analysis is often used to explore relationships between gene expression levels across different samples or experimental conditions.



Correlation

Spearman vs. Pearson

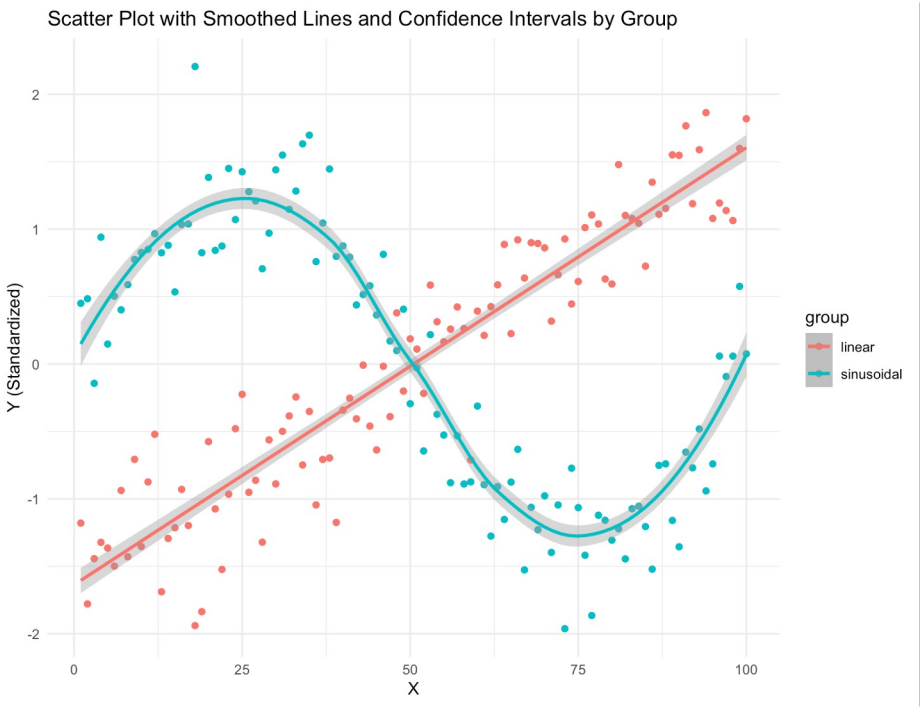
- Pearson Correlation: Measures the linear relationship between two variables. It assumes that the variables are normally distributed and have a linear relationship.
- Spearman Correlation: Measures the monotonic relationship between two variables. It does not assume linearity and is more robust to outliers and non-normal distributions.

We can use the `cor` function to compute both pearson and spearman.

`geom_smooth()` is a ggplot2 function that fits a model to a set of data points and plots a smooth lined.

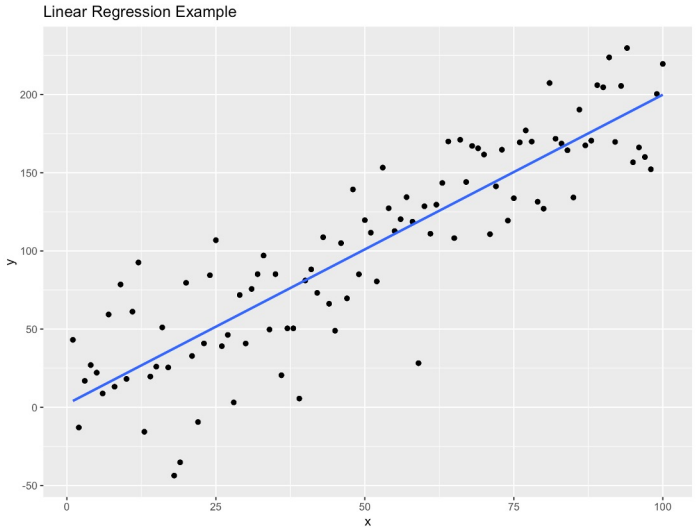
- method argument specifies the model (“lm”,”loess”)
  - lm: linear regression
  - loess: locally weighted smoothing
  - ...
- Add confidence intervals with *se*.

```r
# Generate example data
set.seed(42)
x <- rnorm(100)  # Generate 100 random numbers from a standard normal distribution
y <- x + rnorm(100, mean = 0, sd = 0.5)  # Create y as a noisy version of x

# Calculate Spearman correlation coefficient
spearman_correlation <- cor(x, y, method = "spearman")
print(paste("Spearman correlation coefficient:", round(spearman_correlation, 2)))
```



Regression

- `lm()` function
- Statistical model to estimate the linear relationship between a dependent variable and a set of independent variables.
- The goal is find the best fit line by fitting the observed data to a linear equation.
- `model <- lm(y ~ x, data = df)`
  - The ~ operator is used to specify the linear equation




```
> summary(model)

Call:
lm(formula = y ~ x, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-90.584 -19.855   2.427  19.581  66.791

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.06804    3.13910   0.659     0.51
x            1.97836    0.05397  36.659   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 31.16 on 398 degrees of freedom
Multiple R-squared:  0.7715,    Adjusted R-squared:  0.7709
F-statistic:  1344 on 1 and 398 DF,  p-value: < 2.2e-16
```

Concepts

- Clustering
  - Grouping similar data points together into meaningful clusters.
  - Similarity is measured by a distance function
- Dimensionality reduction
  - Reducing high dimensional data into compact set of explanatory features
  - Useful for visualization
- Correlation
  - Measuring the strength and direction of relationships and plotting the relationships using geom_smooth.