Clustering

(k-Means and Hierarchical)

What is clustering?

- Unsupervised learning
- Group similar observations together
- Observations in a cluster are more similar to one another than to observations in another cluster

What is clustering?

- Minimize the intracluster variance (within cluster variance)
- WSS = “within sums of squares”
- We will focus on k-Means and hierarchical

Before we go further...

k-Means Clustering

k-Means

- Also known as Linde-Buzo-Gray (LBG) algorithm, or as the generalized Lloyd algorithm
- We have n observations, p variables, and k clusters

k-Means Algorithm

- Step 1: Randomly select *k* observations, called the “cluster centroids”
  - Also known as centres, vector quantifiers (VQs), codewords, or codebook vectors

k-Means Algorithm

- Step 2: Compute the Euclidean distance from every observation to each of the *k* cluster centroids
- Step 3: For each of the *n* observations, assign the observation to its closest cluster

k-Means Algorithm

- Step 4: Update the cluster centroid for each of the k clusters
  - Compute the mean of all observations in that cluster (results in a p-dimensional vector)

k-Means Algorithm

- Step 5: Iterate until either:
  - Tolerance is reached (cluster centroids no longer “move”); or,
  - Maximum number of iterations reached

Syntax in R for k-Means

- Example using `iris` dataset (5th column is "`Species`", which is non-numeric -- must remove it to perform clustering)

```
> iris_kmeans <- kmeans(x = iris[, -5],
                        centers = 3,
                        iter.max = 10)
```

IMPORTANT!

Reproducibility of k-Means

- You MUST set a seed before running `kmeans()` if you want it to be reproducible

```
> set.seed(2021)
> iris_kmeans <- kmeans(x = iris[, -5],
                        centers = 3,
                        iter.max = 10)
```

Outputs of the `kmeans()` function in R

- Type `?kmeans` to see the full list of outputs
- The `cluster` element gives the cluster labels for each of the n observations

```
> iris_kmeans$cluster

  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
  [33] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1
  [65] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  [97] 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1
  [129] 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2 2 1
```

Outputs of the `kmeans()` function in R

- `withinss` gives the WSS for each of the *k* clusters

```
> iris_kmeans$withinss

  [1] 39.82097 23.87947 15.15100
```

- `tot.withinss` gives the total WSS

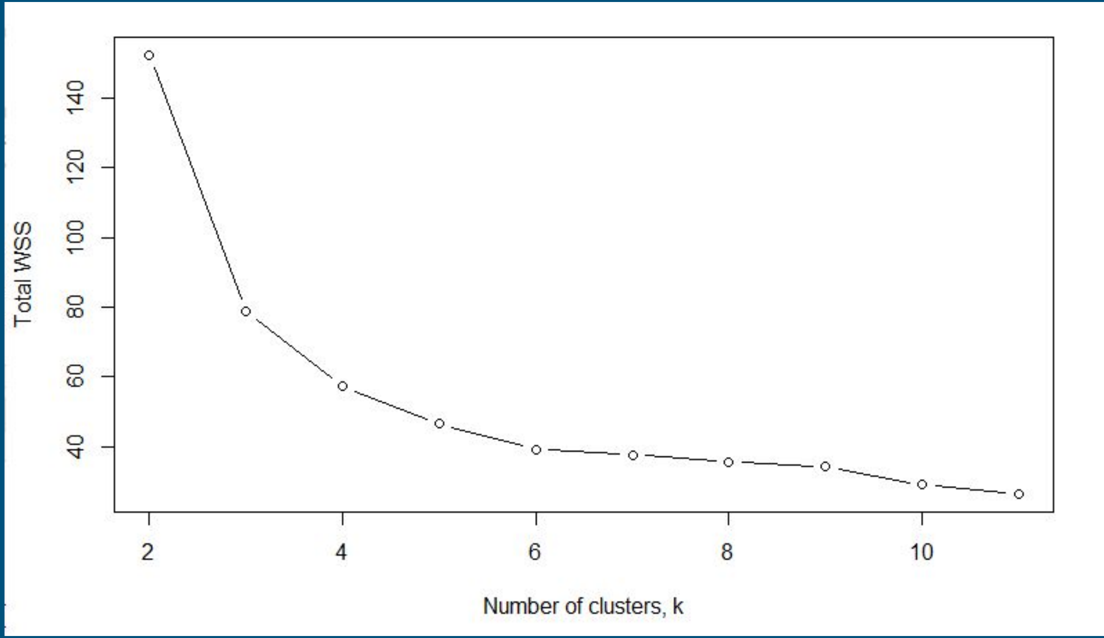
```
> iris_kmeans$tot.withinss

[1] 78.85144
```

So, how do we pick *k*?

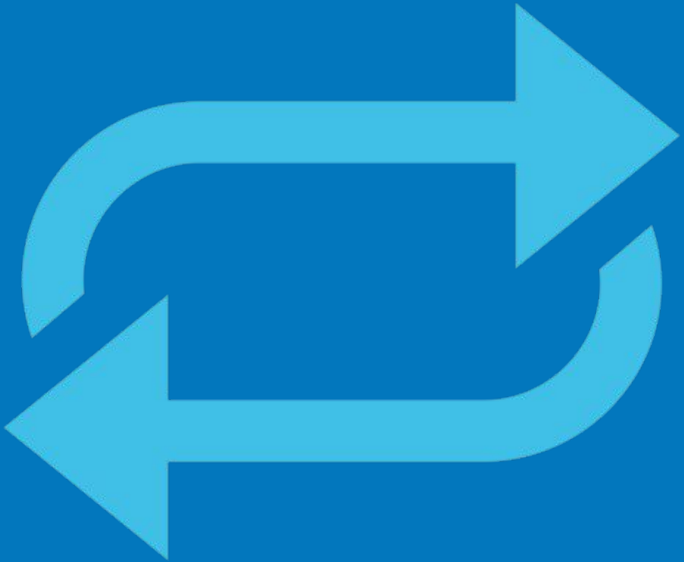
Check total WSS for varying values of *k*

- Find the value of *k* for which there is no longer a meaningful decrease in the Total WSS



Use Davies-Bouldin Index

- Want our clusters to be meaningful
- Can use the Davies-Bouldin index to prevent overfitting
  - Lower values are preferred



Hierarchical Clustering

Hierarchical Clustering

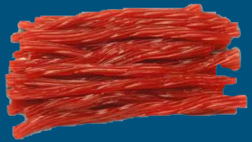
- Two types:
  - Divisive (“top-down”)
  - Agglomerative (“bottom-up”)
- We will focus on agglomerative, which is what’s implemented in the `hclust()` function in R

Agglomerative Hierarchical Clustering

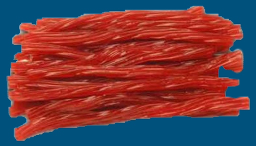
- Step 1: Compute the distance matrix
- Step 2: Make every observation its own cluster, for a total of *n* clusters
- Step 3: Combine the two most similar clusters into one
- Step 4: Update the distance matrix (compute pairwise distances between each pair of clusters)
  - `?hclust` to see all possible linkage methods
- Step 5: Iterate until there is only one cluster

Example of the algorithm

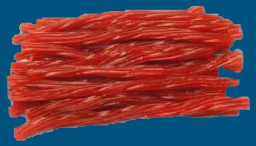
- Let’s do an example where we draw what the dendrogram (output of hierarchical clustering) might look like for 5 different candies



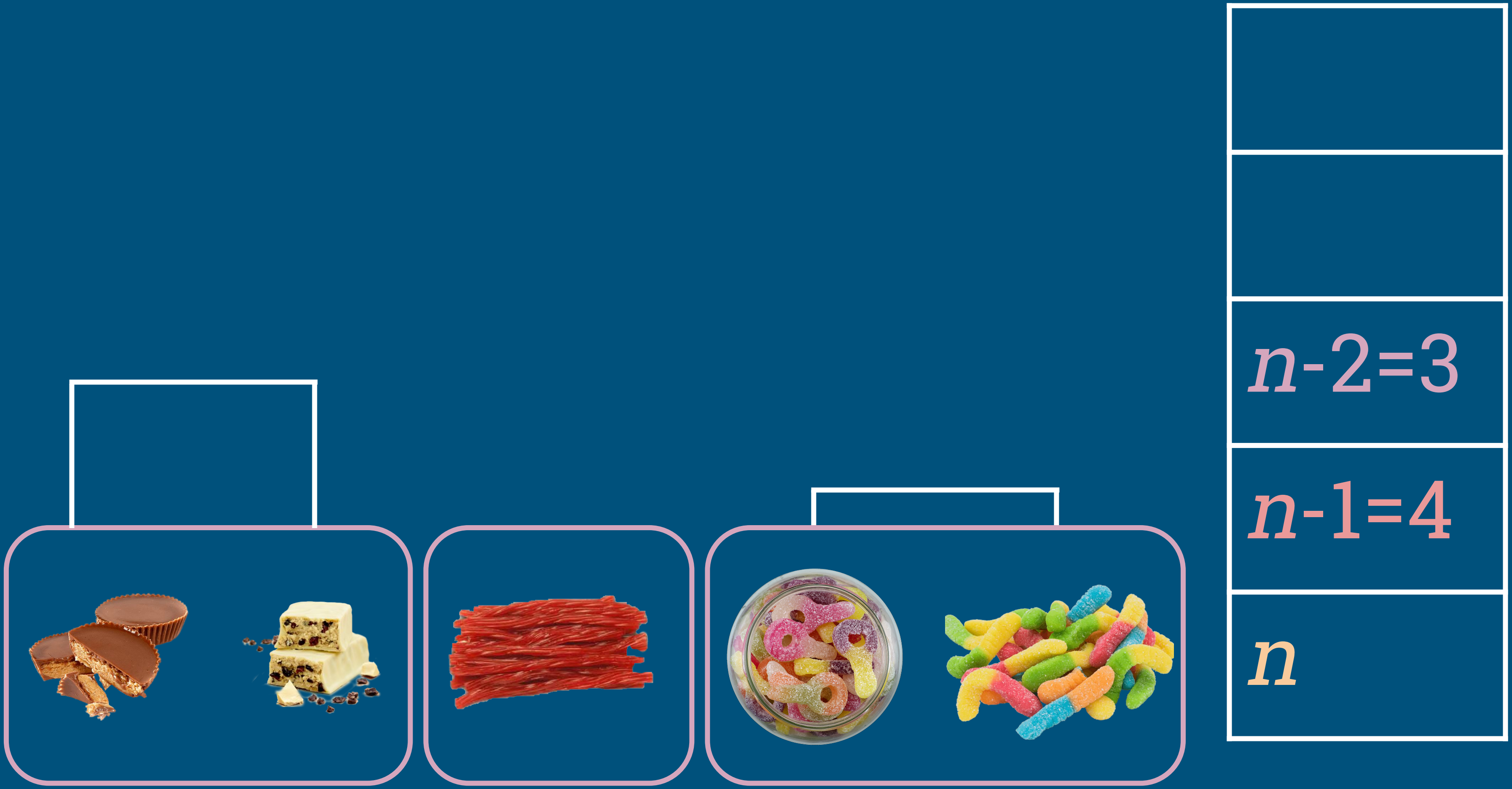


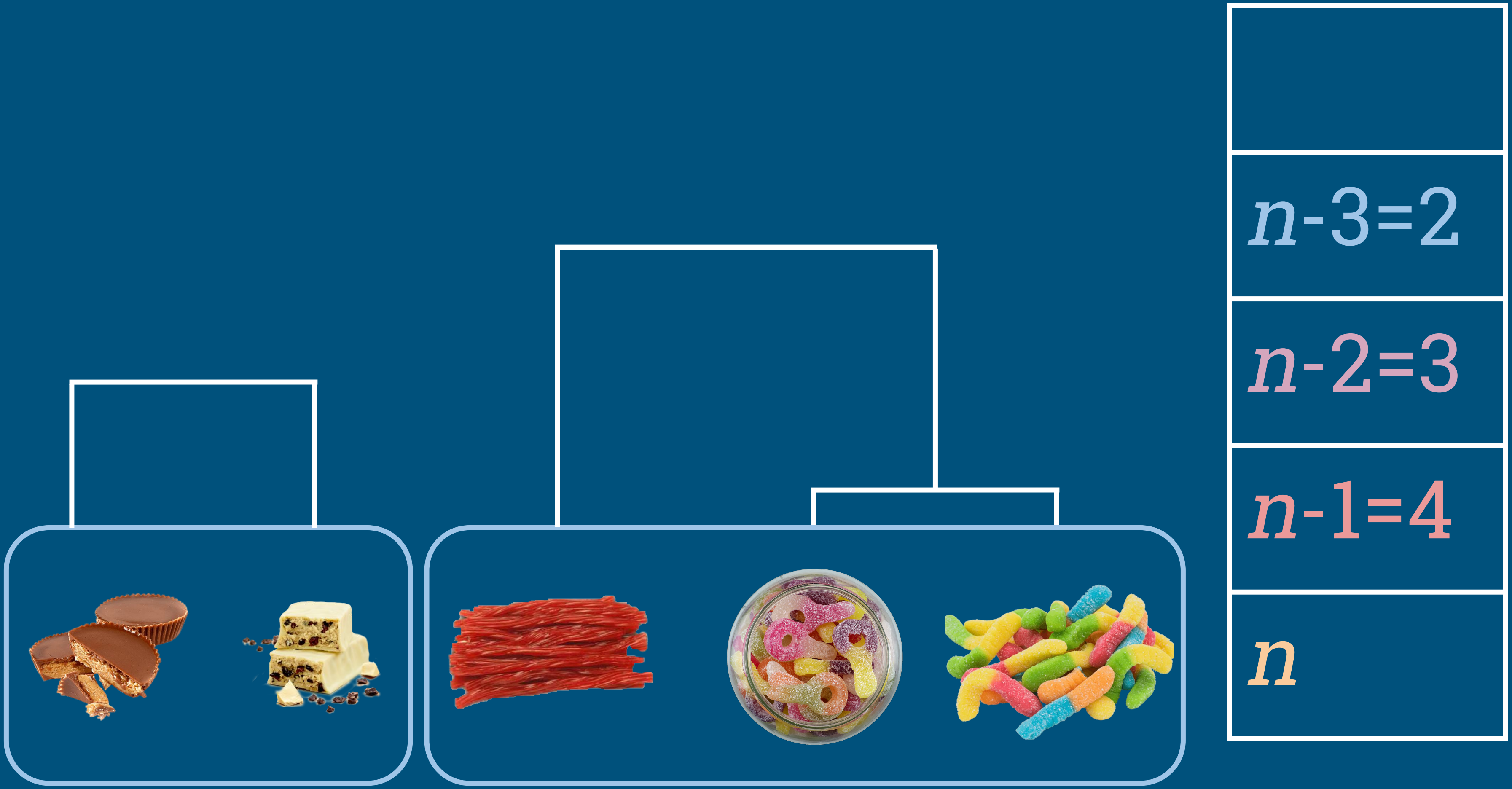
n



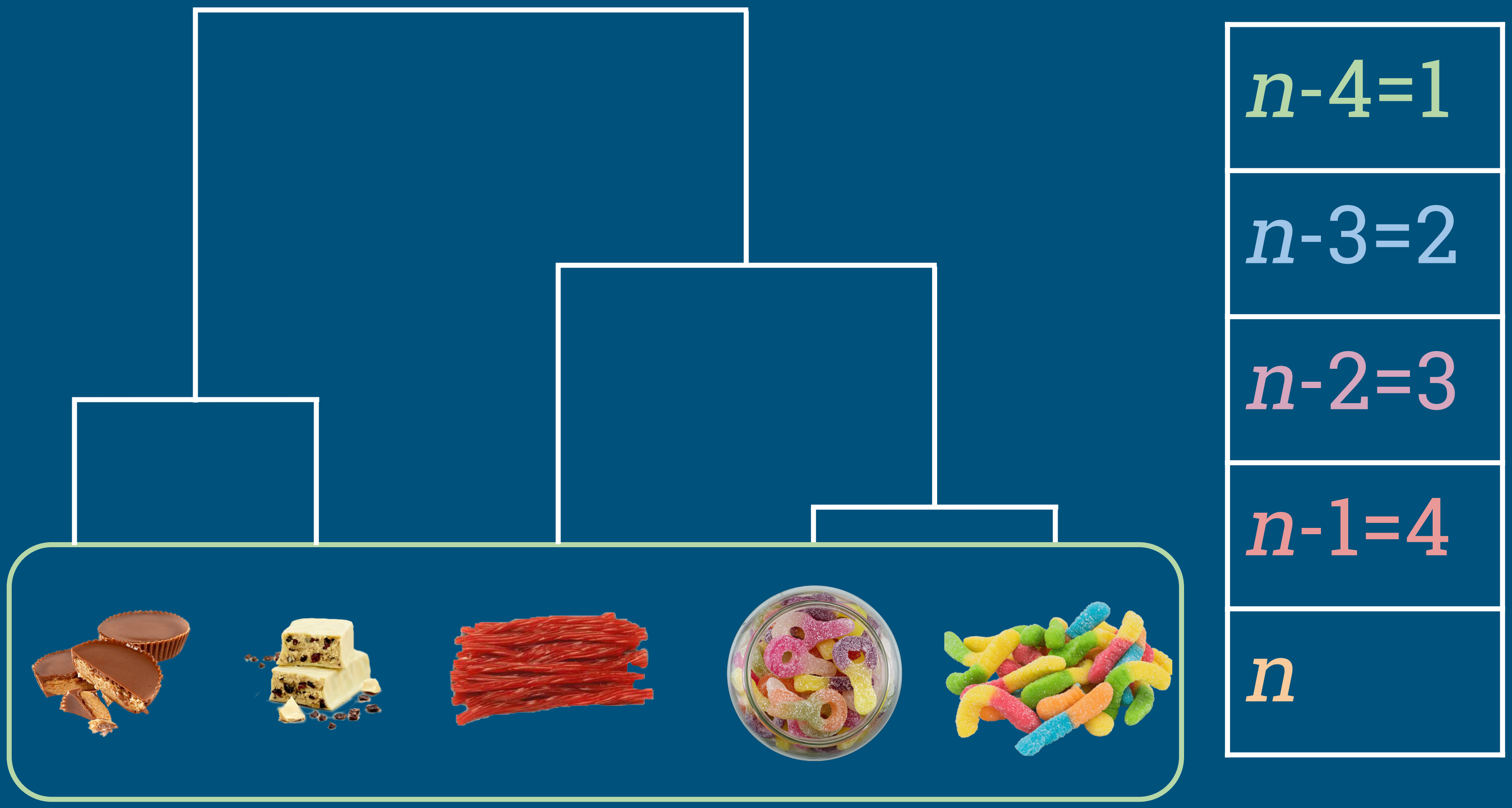
n-1=4

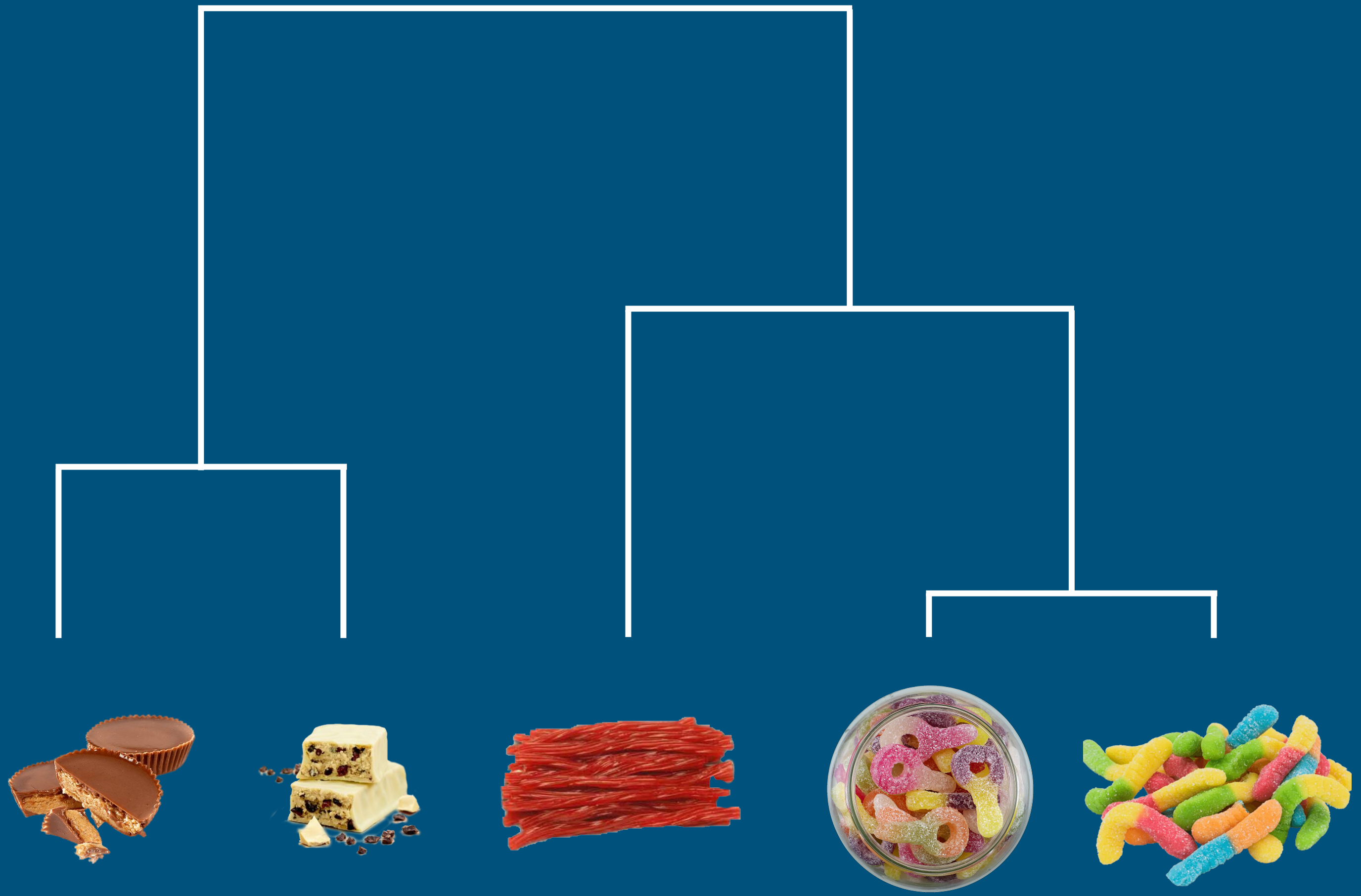
n


n-2=3
n-1=4
n

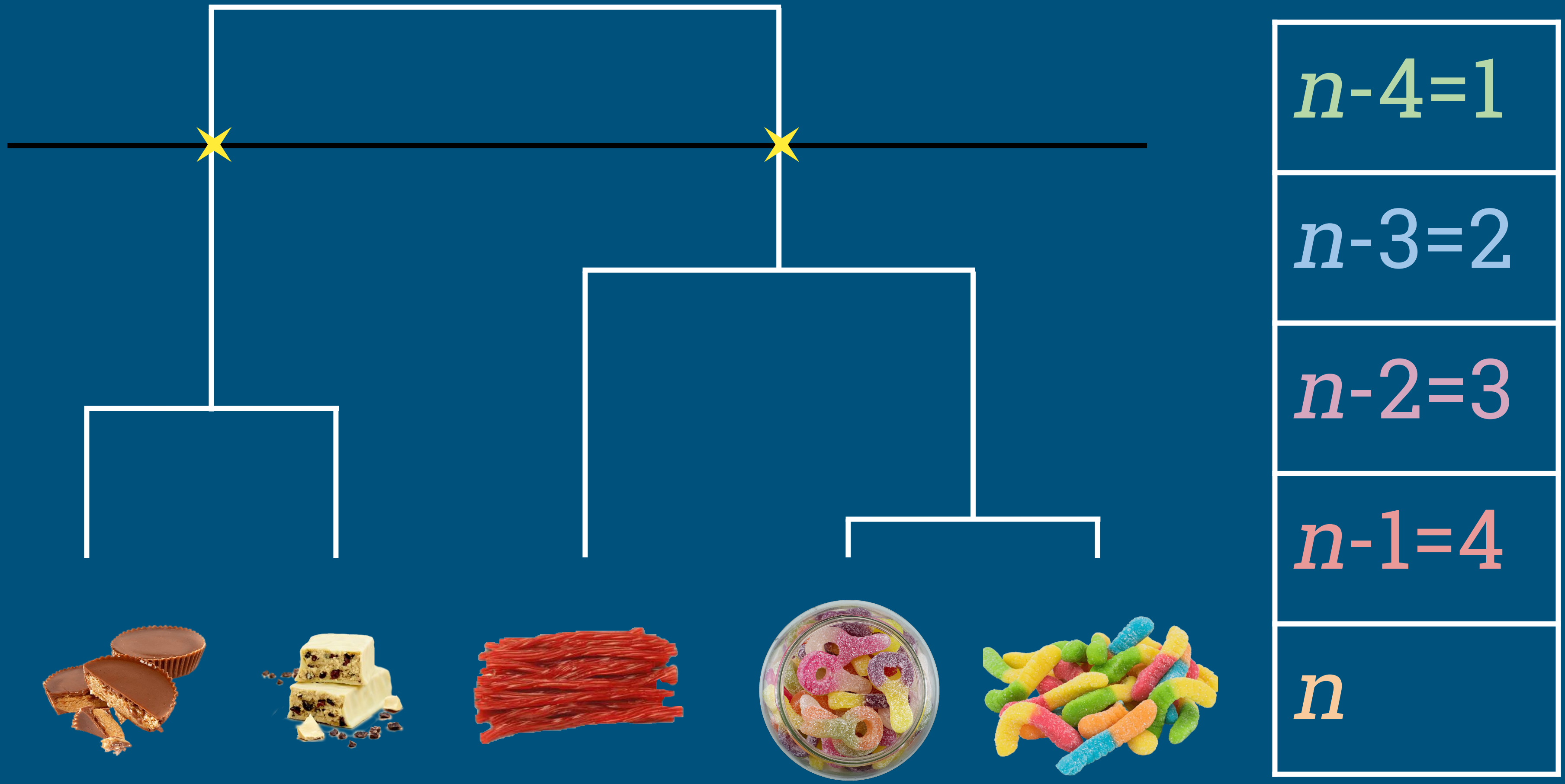


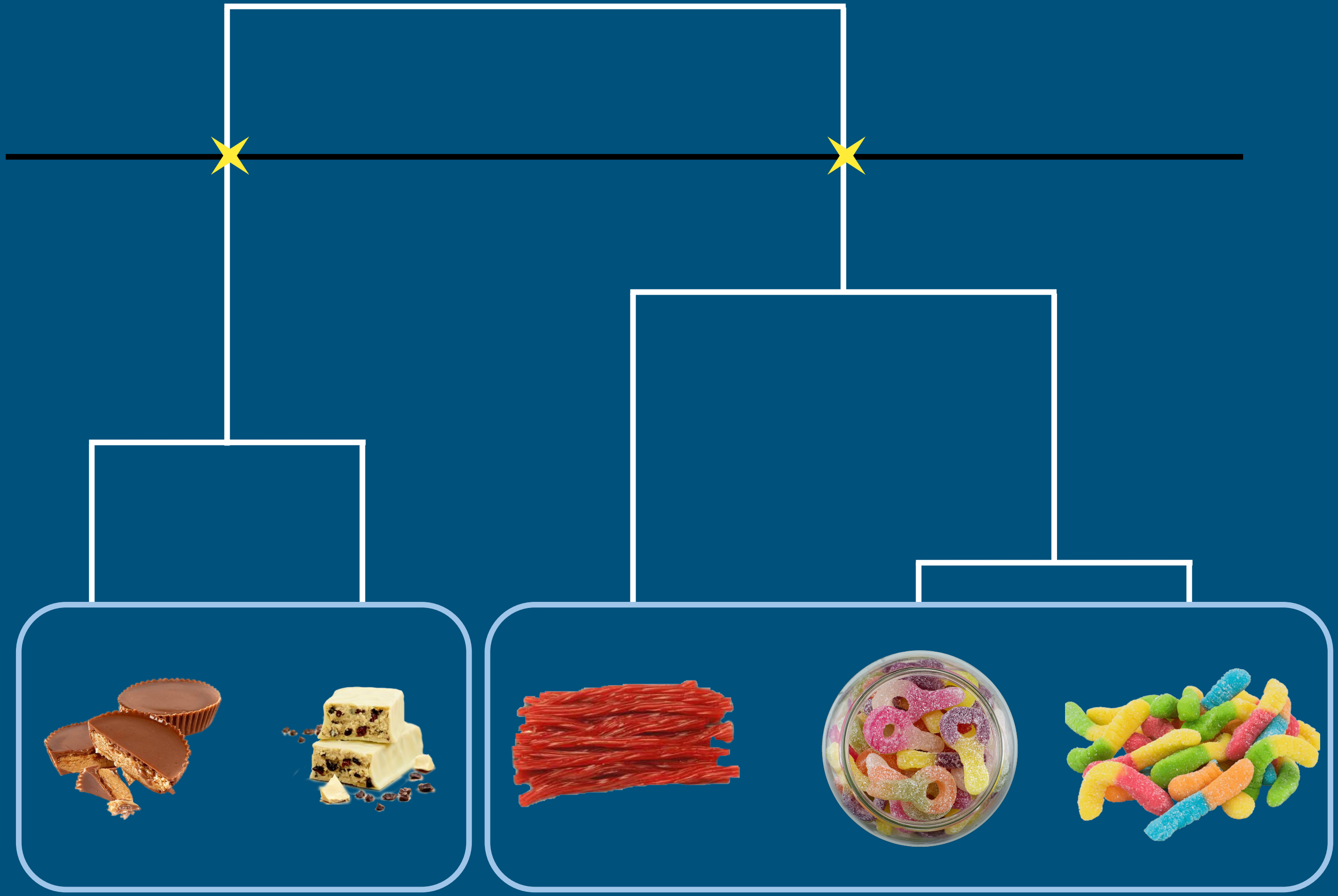
n-3=2
n-2=3
n-1=4
n


n-4=1
n-3=2
n-2=3
n-1=4
n

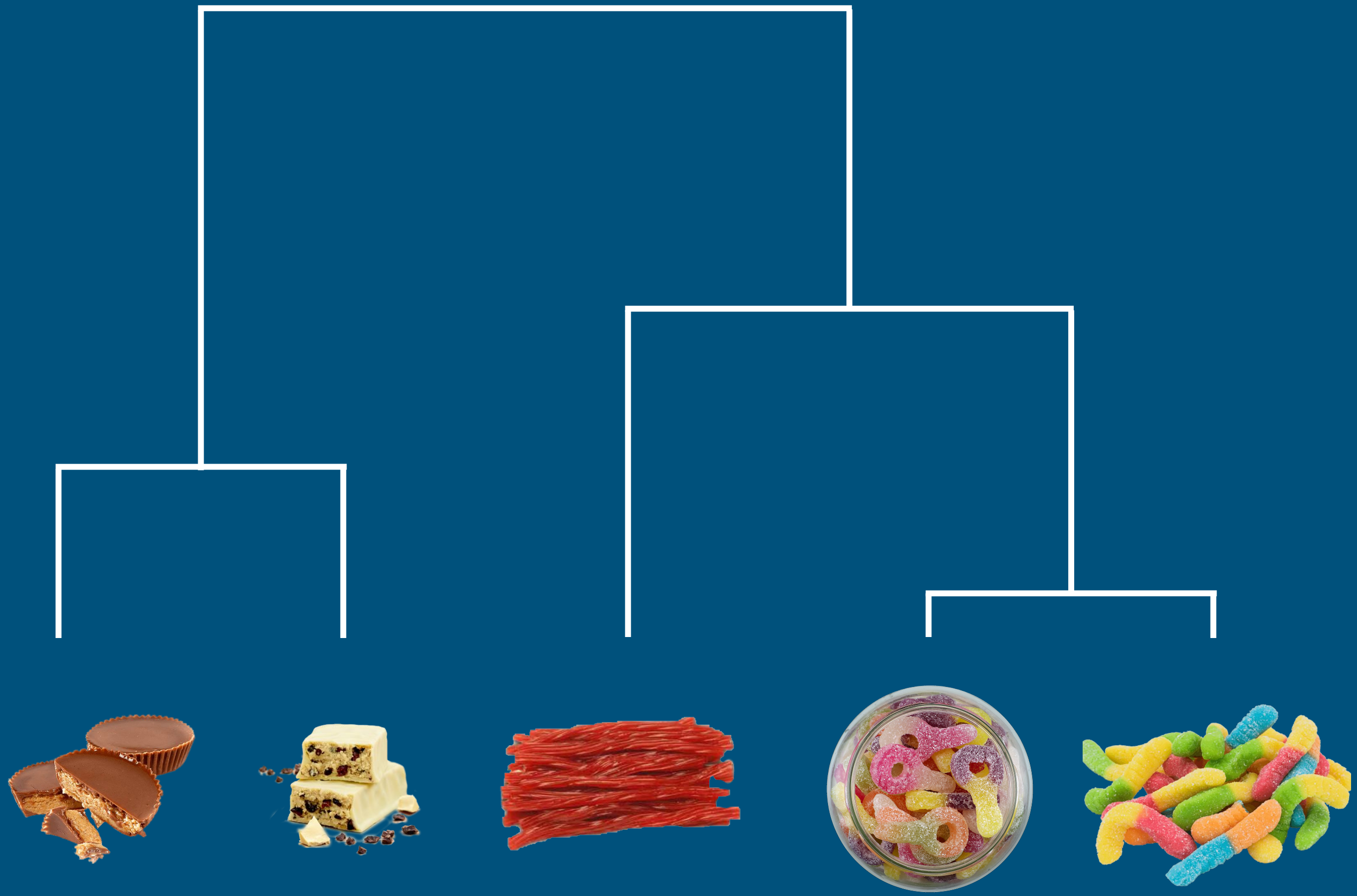


n-4=1
n-3=2
n-2=3
n-1=4
n

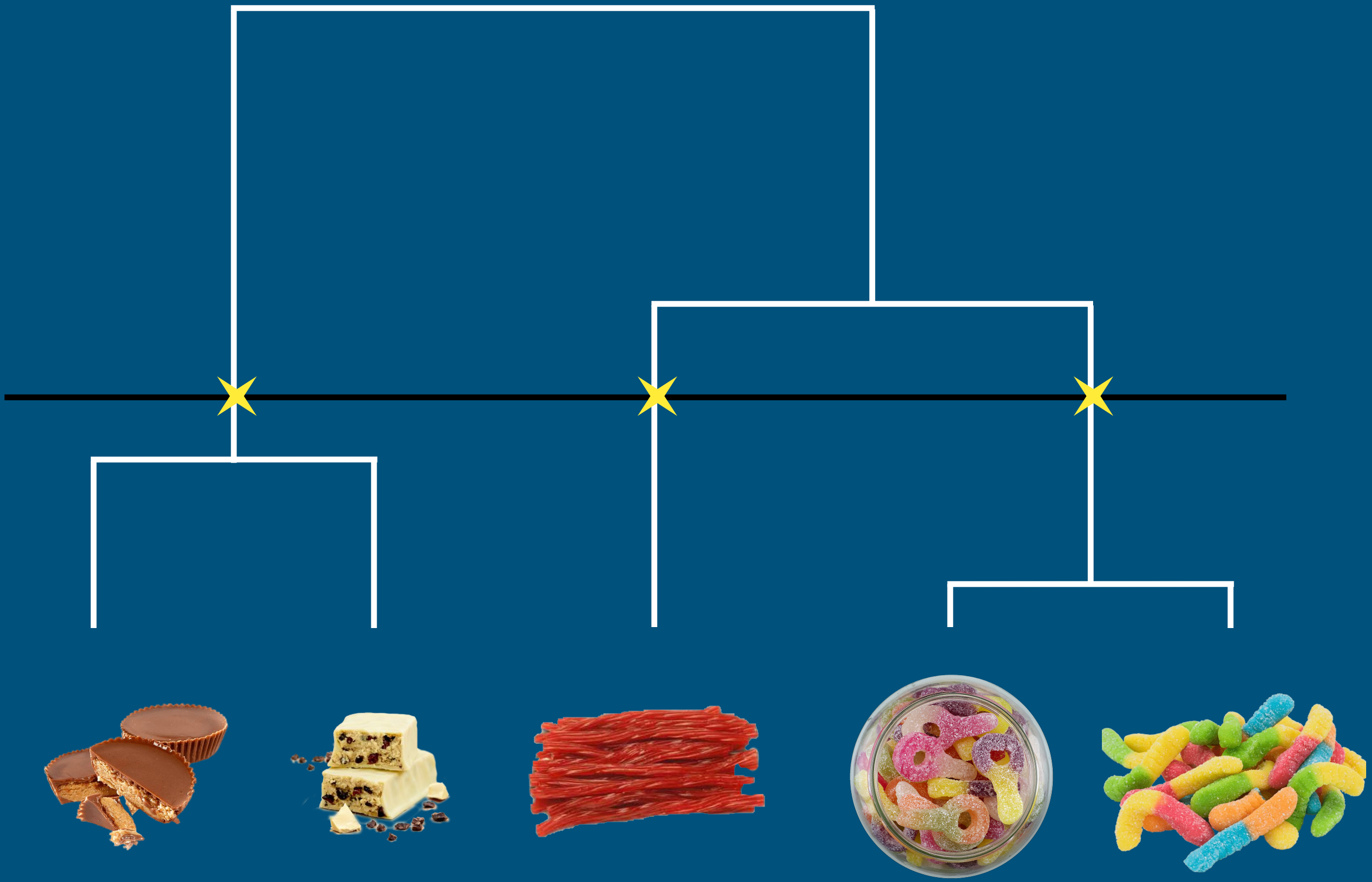

n-4=1
n-3=2
n-2=3
n-1=4
n



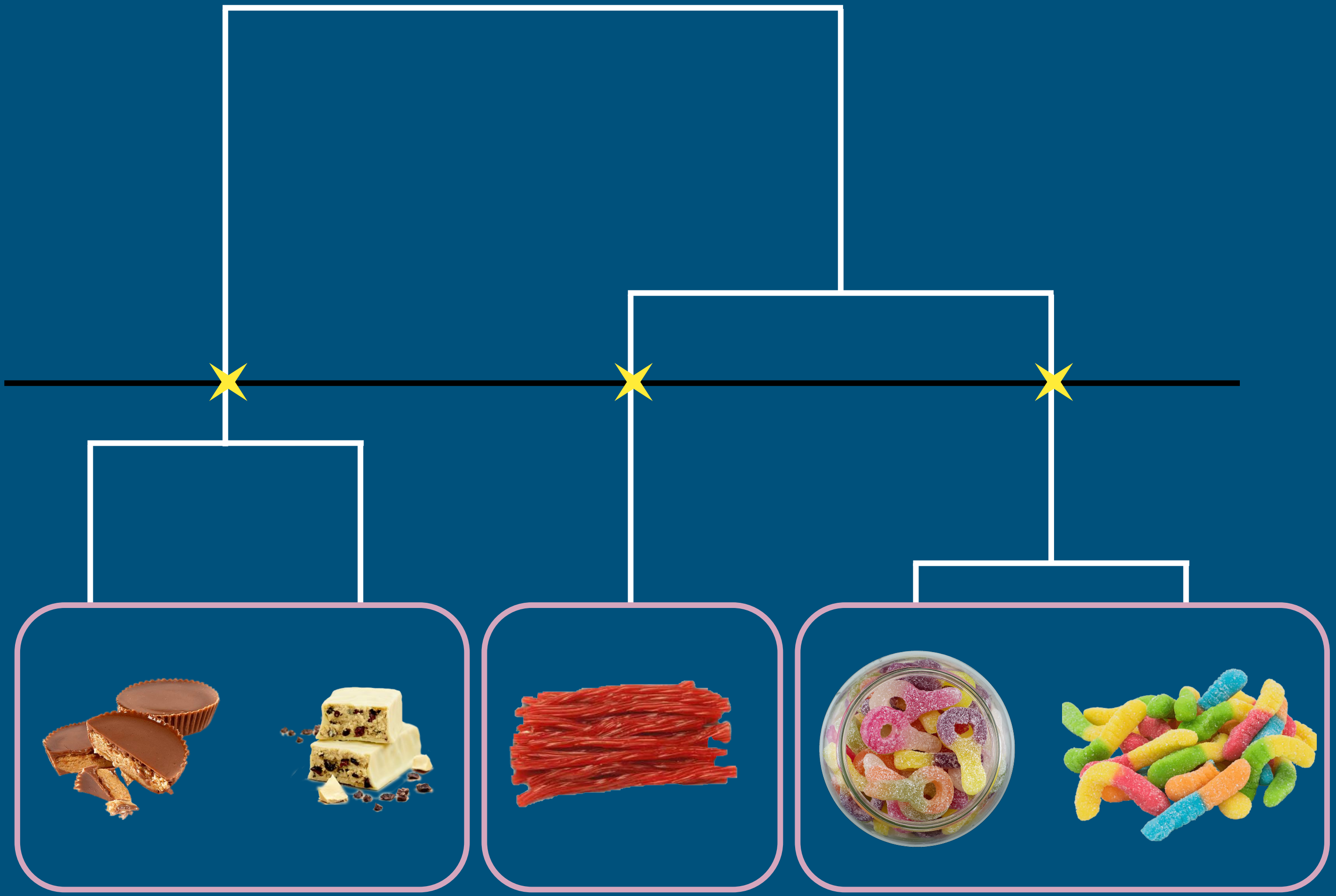
n-4=1
n-3=2
n-2=3
n-1=4
n



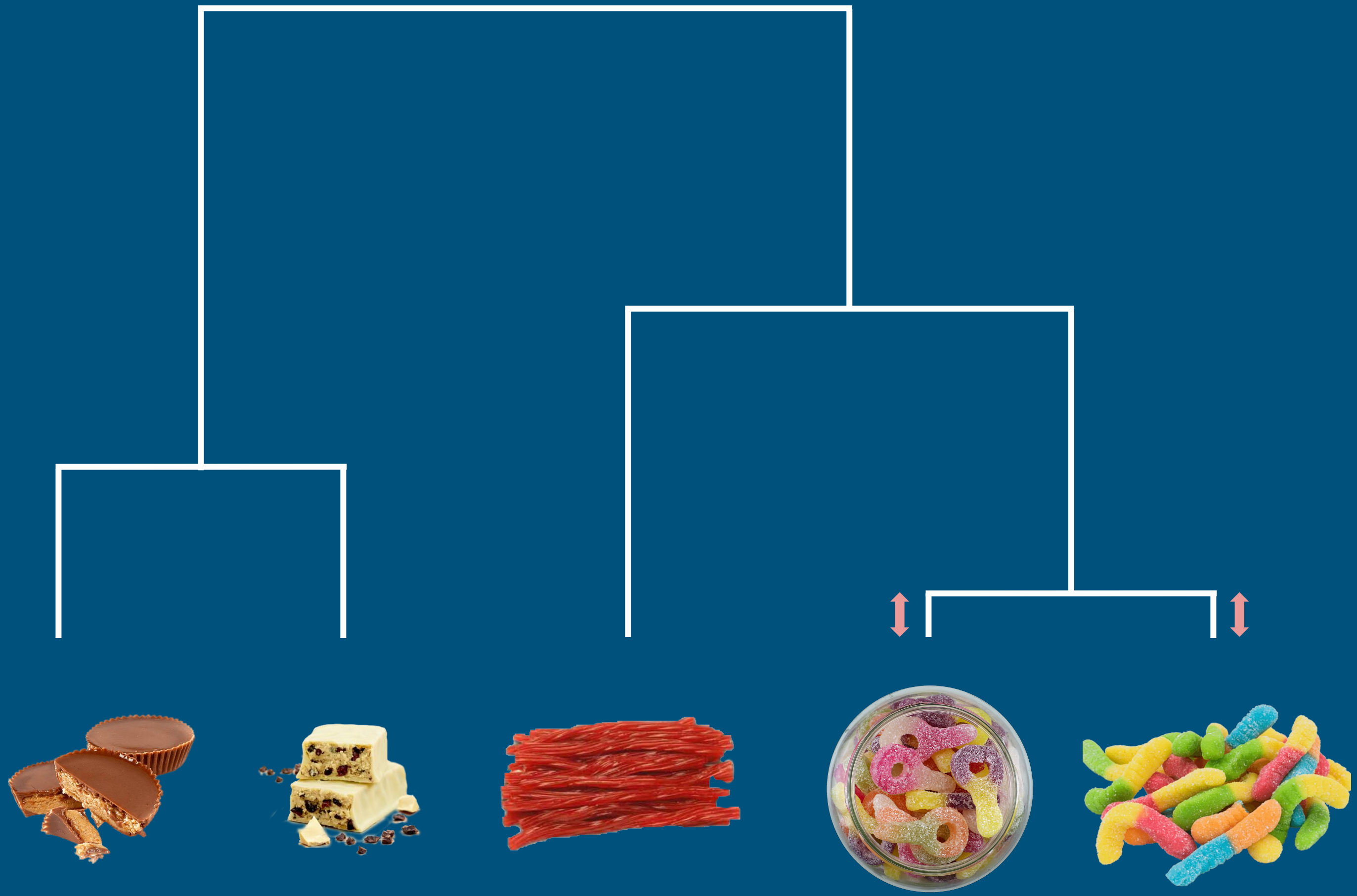
$n-4=1$
$n-3=2$
$n-2=3$
$n-1=4$
n



n-4=1
n-3=2
n-2=3
n-1=4
n



n-4=1
n-3=2
n-2=3
n-1=4
n



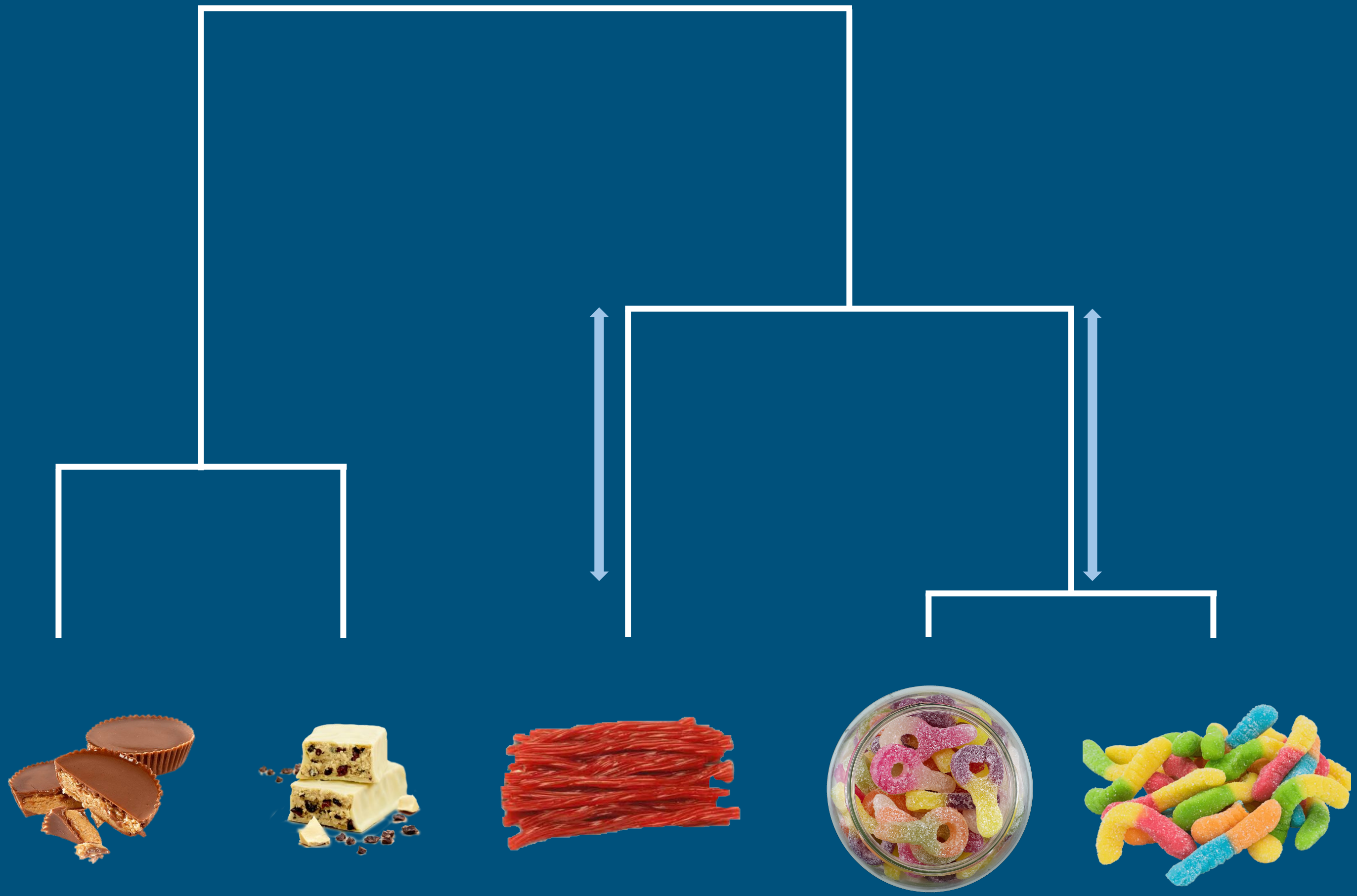
n-4=1

n-3=2

n-2=3

n-1=4

n



n-4=1

n-3=2

n-2=3

n-1=4

n

Using the `factoextra` package in R

Common Questions

QUESTION: Why pick one over the other?

ANSWER: With hierarchical clustering, you don't need to know *k* ahead of time (computationally expensive, though), and have more freedom in how you define "similarity".

The k-Means algorithm is much faster and thus the better choice for very large datasets. Sensitive to initialization, though (important to repeat).