Lec8: Hierarchical Clustering & Heatmaps

Xin (Shayne) Xing

Lecture Outline

Why Hierarchical Clustering?

Limitations of K-Means

- Requires specifying K *a priori*
- Random initialization can yield different solutions
- Assumes roughly spherical, equal-sized clusters
- Provides no information about cluster relationships

Hierarchical Clustering Addresses These

- **No need to commit** to a single K
- Deterministic: same data $\Rightarrow$ same tree
- Produces a **dendrogram**: a full hierarchy of clusterings
- Any number of clusters can be read off by cutting

Two Flavors

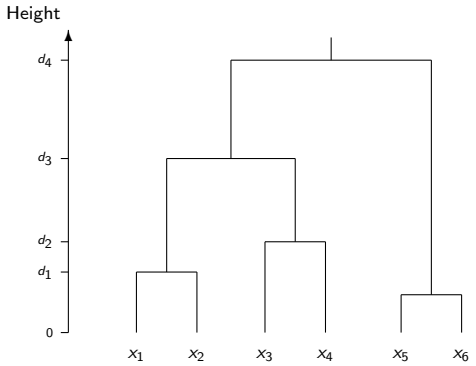
- **Agglomerative** (bottom-up):
  Start with n singletons; repeatedly merge the two most similar clusters.
  Most common.
- **Divisive** (top-down):
  Start with one cluster containing all points; recursively split. Less frequently used.

• This lecture focuses exclusively on agglomerative hierarchical clustering.

Anatomy of a Dendrogram

A dendrogram is an **upside-down tree** encoding a full hierarchy:

- **Leaves** (bottom): individual observations
- **Internal nodes**: cluster merge events
- **Root** (top): single cluster containing all n points
- **Height of a node**: the dissimilarity at which the two sub-clusters merged



$d_1 < d_2 < d_3 < d_4$ indicates successive merges at increasing dissimilarity.

Key Insight

The **vertical axis** (height) is the only meaningful spatial dimension. The horizontal ordering of leaves is *arbitrary* and conveys no similarity information.

Reading a Dendrogram: Similarity

Fundamental Rule

The similarity between two observations is determined by the **height at which their branches first fuse**, *not* by their left-right position.

- **Low fusion height** $\Rightarrow$ observations are very **similar**
- **High fusion height** $\Rightarrow$ observations are very **dissimilar**
- Two observations can appear adjacent on the page yet fuse very high (large dissimilarity)

Common Misconception

Do not judge similarity by horizontal proximity. Rotating any subtree around its root is valid and changes leaf order without changing the tree structure.

Example (9 observations)

From the dendrogram at right:

- Obs. 5 & 7 fuse at low height $\Rightarrow$ most similar pair
- Obs. 1 & 6 fuse next
- Obs. 9 merges with $\{2\}$ at moderate height
- Obs. 4 fuses last $\Rightarrow$ most different from all others

The scatter plot (right panel) confirms these relationships in 2D feature space.

Cutting the Dendrogram to Obtain Clusters

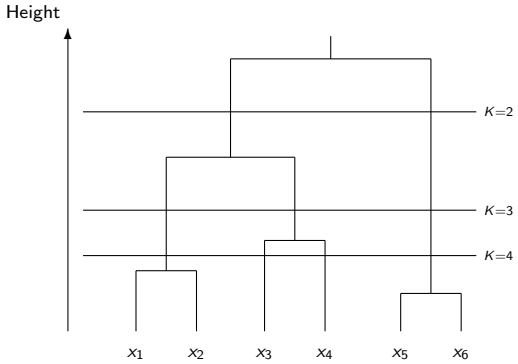
The Cut Operation

Drawing a horizontal line at height h partitions the leaves into **disjoint clusters**: one cluster per branch that crosses the cut line.

- A single dendrogram encodes **all possible** flat clusterings simultaneously
- **Higher cut** $\Rightarrow$ fewer, larger clusters
- **Lower cut** $\Rightarrow$ more, finer clusters

"Hierarchical" Property

Clusters obtained at height $h_1 < h_2$ are **nested**: every cluster at h_1 is a subset of some cluster at h_2. This is a consequence of the agglomerative construction and *cannot* be violated.



Orange cut $\Rightarrow K = 2$; **Blue** cut $\Rightarrow K = 3$; **Green** cut $\Rightarrow K = 4$

Step 1: Dissimilarity Between Observations

Before clustering, we must choose a **dissimilarity measure** between pairs of observations $x_i, x_j \in \mathbb{R}^p$.

Euclidean Distance

$$d_E(x_i, x_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$

- Sensitive to **scale** $\Rightarrow$ standardize variables when units differ
- Groups observations with similar **magnitudes**
- Standard choice for continuous numeric data

Correlation-Based Distance

$$d_\rho(x_i, x_j) = 1 - \text{cor}(x_i, x_j)$$

where cor is Pearson correlation across the p features.

- $d_\rho \in [0, 2]$; zero means perfectly correlated
- Groups observations with similar **profile shapes**
- Preferred in genomics where scale varies widely

Scale Matters

Euclidean distance is dominated by variables with large variance. Always consider **standardizing** (zero mean, unit variance) before computing d_E, unless raw scale differences are scientifically meaningful.

Euclidean vs. Correlation-Based Distance

Consider three observations measured on $p = 20$ variables:

- **Obs. 1 & 2**: different magnitudes but nearly identical *shape* (rise and fall together)
  - d_E is **large** (different scale)
  - $d_\rho \approx 0$ (highly correlated $\Rightarrow$ similar)
- **Obs. 2 & 3**: similar magnitudes but different shape
  - d_E is **small**
  - d_ρ is **large** (uncorrelated $\Rightarrow$ dissimilar)

Practical Guidance

- **Gene expression**: use d_ρ (genes co-regulated together regardless of absolute expression level)
- **Consumer profiling**: use d_E after standardization (absolute spending differences matter)

Schematic: Profiles Across 20 Variables

Obs	Profile shape (schematic)
1	**High values**: $\uparrow\downarrow\uparrow\downarrow\cdots$ (wavy, large magnitude)
2	**Low values**: same $\uparrow\downarrow$ pattern, small magnitude
3	**Low values**: $\downarrow\uparrow\downarrow\uparrow\cdots$ (opposite phase)

Pair	d_E	d_ρ
1 vs. 2	**Large**	≈ 0
2 vs. 3	Small	**Large**

Step 2: Dissimilarity Between Clusters (Linkage)

Let $\mathcal{A}$ and $\mathcal{B}$ be two clusters. For all pairs (i, j) with $i \in \mathcal{A}$, $j \in \mathcal{B}$, compute pairwise distances $d(i, j)$.

Linkage	Definition	Properties
Complete	$D(\mathcal{A}, \mathcal{B}) = \max_{i \in \mathcal{A}, j \in \mathcal{B}} d(i, j)$	Compact, balanced clusters; robust to outliers
Single	$D(\mathcal{A}, \mathcal{B}) = \min_{i \in \mathcal{A}, j \in \mathcal{B}} d(i, j)$	Can produce chaining; sensitive to outliers
Average	$D(\mathcal{A}, \mathcal{B}) = \frac{1}{\lvert\mathcal{A}\rvert \lvert\mathcal{B}\rvert} \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{B}} d(i, j)$	Compromise; most widely recommended
Centroid	$D(\mathcal{A}, \mathcal{B}) = d(\bar{x}_{\mathcal{A}}, \bar{x}_{\mathcal{B}})$	Intuitive but can cause *inversions*
Ward's	Minimize total within-cluster variance after merge	Often yields most interpretable results

Linkage Methods: Visual Comparison

Complete Linkage

- Uses the **farthest** pair of points
- Produces **compact**, roughly spherical clusters
- Sensitive to outliers on the boundary
- **Recommended** as a default when clusters are expected to be well-separated

Single Linkage

- Uses the **nearest** pair of points
- Can produce **chaining**: observations attach one-at-a-time to a growing cluster
- Useful for detecting elongated or non-convex clusters
- **Not** recommended when clusters are globular

Average Linkage

- Uses **all** pairwise distances (mean)
- Balances Complete and Single
- Less sensitive to outliers than Complete
- **Most commonly used** in practice; robust across many settings

Worked Example: 4 Observations

Dissimilarity matrix:

$$D = \begin{pmatrix} 0 & 0.3 & 0.4 & 0.7 \\ 0.3 & 0 & 0.5 & 0.8 \\ 0.4 & 0.5 & 0 & 0.45 \\ 0.7 & 0.8 & 0.45 & 0 \end{pmatrix}$$

Step-by-step (Complete Linkage):

1. $\min D = d(1,2) = 0.3 \Rightarrow$ merge $\{1,2\}$
2. Update: $d(\{1,2\},3) = \max(0.4, 0.5) = 0.5$; $d(\{1,2\},4) = \max(0.7, 0.8) = 0.8$
3. $\min = d(\{1,2\},3) = 0.5$, $d(3,4) = 0.45 \Rightarrow$ merge $\{3,4\}$
4. Update: $d(\{1,2\},\{3,4\}) = \max(0.4, 0.7, 0.5, 0.8) = 0.8$
5. Merge $\{1,2,3,4\}$ at height 0.8

Agglomerative Hierarchical Clustering Algorithm

Formal Algorithm

Input: n observations $x_1, \ldots, x_n \in \mathbb{R}^p$; a dissimilarity measure $d(\cdot, \cdot)$; a linkage $D(\cdot, \cdot)$

1. Initialize n clusters, one per observation: $\mathcal{C}_i = \{x_i\}$ for $i = 1, \ldots, n$.
   Compute all $\binom{n}{2} = \frac{n(n-1)}{2}$ pairwise dissimilarities.
2. **For** $t = n, n-1, \ldots, 2$:
   (a) Find the pair $(\mathcal{C}_a, \mathcal{C}_b)$ minimizing $D(\mathcal{C}_a, \mathcal{C}_b)$ over all current clusters.
   (b) **Merge** $\mathcal{C}_a$ and $\mathcal{C}_b$ into $\mathcal{C}_{\text{new}} = \mathcal{C}_a \cup \mathcal{C}_b$. Record $D(\mathcal{C}_a, \mathcal{C}_b)$ as the height of this node.
   (c) Update the dissimilarity matrix: compute $D(\mathcal{C}_{\text{new}}, \mathcal{C}_k)$ for all remaining clusters $\mathcal{C}_k$.
3. **Output:** The sequence of $n-1$ merges constitutes the **dendrogram**.

Complexity:

- Naïve: $\mathcal{O}(n^3)$ time, $\mathcal{O}(n^2)$ space
- Efficient (e.g. SLINK for single): $\mathcal{O}(n^2)$ time

Key property:

- Merges are **irreversible** — once two clusters join, they stay merged
- This is why the nested/hierarchical structure

Algorithm in Action: 9 Observations

Sequential merges (Complete Linkage, Euclidean):

Step	Merge	Clusters	Height
1	$\{5, 7\}$	8 remain	small
2	$\{1, 6\}$	7 remain	small
3	$\{5, 7, 8\}$	6 remain	medium
4	$\{1, 6, \ldots\}$	5 remain	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	All merged	1 cluster	large

- At each step, we pick the minimum inter-cluster dissimilarity
- Height in the dendrogram records the exact dissimilarity value
- Cutting at height h reveals structure at that scale

Practical Tips for Choosing a Cut

1. **Large gap heuristic**: look for a big jump in fusion height; cut just below it
2. **Domain knowledge**: use K suggested by scientific context
3. **Stability**: compare clusterings across linkages; prefer ones that agree
4. **Visualization**: use heatmaps (covered next) to visually validate clusters

Fitting the Hierarchical Cluster Model

Generate data with two true clusters

```
set.seed(2)
x <- matrix(rnorm(50 * 2), ncol = 2)
# Shift first 25 obs to create separation
x[1:25, 1] <- x[1:25, 1] + 3
x[1:25, 2] <- x[1:25, 2] - 4

# Compute Euclidean distance matrix
D <- dist(x)   # default: Euclidean

# Fit with three linkage methods
hc.complete <- hclust(D, method = "comple
hc.average  <- hclust(D, method = "averag
hc.single   <- hclust(D, method = "single
hc.ward     <- hclust(D, method = "ward.D
```

hclust Key Arguments

Argument	Description
`d`	dissimilarity matrix (from dist)
`method`	linkage: `"complete"`, `"average"`,`"single"`, `"ward.D2"`, ...

dist Key Arguments

Argument	Description
`x`	data matrix ($n \times p$)
`method`	`"euclidean"` (default), `"manhattan"`,

Drawing and Interpreting Dendrograms

```
par(mfrow = c(1, 3))
plot(hc.complete, main = "Complete Linkage", xlab = "", sub = "", cex = .9)
plot(hc.average, main = "Average Linkage", xlab = "", sub = "", cex = .9)
plot(hc.single, main = "Single Linkage", xlab = "", sub = "", cex = .9)
```

Complete:

- Two clear, balanced groups
- Fusion heights well-separated
- Easy to cut at $K = 2$

Average:

- Similar to complete here
- Gap between $K = 2$ groups visible
- Less extreme heights

Single:

- Chaining visible
- One large group, then singletons attach
- Hard to identify $K = 2$ cleanly

Extracting Cluster Labels: cutree

Cut by number of clusters K:

```
# Returns integer vector of length n
labels.k2 <- cutree(hc.complete, k = 2
labels.k3 <- cutree(hc.average,
k = 3)
table(labels.k2)
```

Cut by height h:

```
# All obs whose subtree root is below
labels.h2 <- cutree(hc.complete, h = 2
labels.h4 <- cutree(hc.average,
h = 4)
```

Output Format

cutree returns a named integer vector of **cluster labels** $1, 2, \ldots, K$ for each of the n observations.

Downstream Use

```
# Visualize cluster assignment
plot(x, col = cutree(hc.complete, k=
     pch = 19, xlab = "X1", ylab = "

# Compare with true labels
table(cutree(hc.complete, k=2),
      true.labels)
```

Comparing Euclidean vs. Correlation Distance

```
# Euclidean distance (default)
plot(hclust(dist(x), method = "complete"),
     main = "Euclidean Distance", xlab = "", sub = "")

# Correlation-based distance: 1 - cor(obs_i, obs_j)
# Note: cor(t(x)) gives obs-by-obs correlation
dd <- as.dist(1 - cor(t(x)))
plot(hclust(dd, method = "complete"),
     main = "Correlation-Based Distance", xlab = "", sub = "")
```

Important: `cor(t(x))` vs `cor(x)`

- `cor(x)` computes **feature-feature** correlations ($p \times p$ matrix)
- `cor(t(x))` computes **observation-observation** correlations ($n \times n$ matrix) — this is what we need for clustering observations

What is a Heatmap?

Definition

A **heatmap** encodes a matrix of numerical values as a **grid of colored cells**, with color intensity (or hue) representing the magnitude of each value.

Key components:

- **Color-coded matrix**: each cell (i, j) represents x_{ij}
- **Row dendrogram**: hierarchical ordering of observations
- **Column dendrogram**: hierarchical ordering of features
- **Color scale bar**: maps colors to values
- **Row/column labels**: observation and feature names
- **Sidebars (optional)**: annotation strips for group membership

Fields of Application

Field	Use Case
Genomics	Gene expression
Finance	Return correlation
Neuroscience	fMRI connectivity
Ecology	Species abundance
ML	Confusion matrices

Advantage

Hundreds of rows $\times$ hundreds of columns can be visualized **on a single screen** — impossible with scatter plots or tables.

Reference: Gehlenborg & Wong (2012), *Nature*

The heatmaply Package: Getting Started

Installation & Basic Usage

```
install.packages("heatmaply")
library("heatmaply")

# Basic heatmap of mtcars dataset
heatmaply(mtcars)
```

What heatmaply produces

- **Interactive** (Plotly-based) heatmap
- Hover to see exact row/column/value
- Zoom and pan capabilities
- Export as PNG or HTML

The `mtcars` Dataset

- 32 car models (rows)
- 11 features: `mpg`, `hp`, `wt`, `disp`, ...
- Without scaling: `disp` (50–470) dominates `vs` (0 or 1)

Key observation: the unscaled heatmap is dominated by displacement (`disp`) and horsepower (`hp`), masking patterns in other variables $\Rightarrow$ **must scale**.

Scaling the Data

```
# Z-score normalize each column
heatmaply(mtcars, scale = "column")

# Also possible: scale rows
heatmaply(mtcars, scale = "row")
```

Column scaling transforms each feature j:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

- All features now have mean 0 and standard deviation 1
- Color now encodes **relative position** within each feature
- Enables fair visual comparison across features with

When to Scale?

Scale	When
Columns	Variables on different scales / units
Rows	Observations vary widely in magnitude (e.g. gene expression)
None	Variables already on the same scale; absolute values matter

After column scaling, Cadillac Fleetwood appears clearly in its own high-weight, high-displacement cluster, while small economy cars cluster separately.

Customizing the Heatmap

Change linkage method:

```
heatmaply(mtcars,
  scale           = "column",
  hclust_method   = "average"
# or "ward.D2"
)
```

Change color palette:

```
# Sequential palette
heatmaply(mtcars, scale="column",
  col = viridis::viridis(200))

# Diverging palette (good for z-scores
heatmaply(mtcars, scale="column",
  col = colorRampPalette(
    c("navy","white","firebrick"))(200
```

Color Palette Guidelines

- **Sequential** (e.g. viridis):
  Use when values go from low to high
  (e.g. raw counts, distances)
- **Diverging** (e.g. blue-white-red):
  Use when values have a meaningful midpoint
  (e.g. z-scores centered at 0, log fold-change)
- **Qualitative**: never use for continuous data;
  reserve for categorical annotations

k_row / k_col

Colors the dendrogram branches by cutting at k groups — immediately reveals the main clustering structure without inspecting branch heights.

Correlation Heatmaps

```
heatmaply_cor(cor(mtcars),
  xlab     = "Features",
  ylab     = "Features",
  k_col    = 2,
  k_row    = 2)
```

```
r <- cor(mtcars)
# Compute pairwise p-values
cor.test.p <- function(x) {
  FUN <- function(x, y) cor.test(x,y)$p
  z <- outer(colnames(x), colnames(x),
             Vectorize(function(i,j) FU
  dimnames(z) <- list(colnames(x), colna
  z
}
p <- cor.test.p(mtcars)
```

Interpreting Correlation Heatmaps

- **Red** cells: strong positive correlation
- **Blue** cells: strong negative correlation
- Diagonal always = 1 (dark red)
- Clustering reveals **groups of correlated features**

Advanced: Dot Size = Significance

- Dot **color** = correlation value
- Dot **size** = $-\log_{10}(p\text{-value})$: larger dot $\Rightarrow$ more significant
- Conveys both *effect size* and *significance* simultaneously

Adding Annotations: Row Side Colors

```
library(tidyverse)
x  <- as.matrix(datasets::mtcars)

# Create annotation: Merc vs. others
is_merc <- unlist(
  lapply(str_split(rownames(mtcars), " "),
         function(v) v[1])) == "Merc"
rc <- ifelse(is_merc, "firebrick", "steel
```

```
heatmaply(x,
  scale         = "column",
  RowSideColors = rc,
  RowSideColors.size = 1.2)
```

Extending to Multiple Annotations

```
# Multiple sidebars as a data frame
```

Why Annotations Matter

- Annotations bridge the heatmap with **known metadata**
- Visually test whether the unsupervised clustering **recovers known groups**
- Mercedes models (**red**) should cluster together if brand is informative

Validation

If clustering groups **red** (Merc) and **blue** (others) cleanly, the features contain brand-specific signal. Mixing $\Rightarrow$ features not brand-discriminative.

The NCI60 Dataset

Dataset Description

- **Source**: National Cancer Institute
- **Rows**: 64 cancer cell lines
- **Columns**: 6,830 gene expression measurements
- **Cancer types**: 14 types including BREAST, CNS, COLON, LEUKEMIA, MELANOMA, OVARIAN, ...

Scientific question: Do gene expression profiles cluster by cancer type without using the type labels?

```
library(ISLR)
nci.labs <- NCI60$labs
# 64 cancer type labels
nci.data <- NCI60$data
# 64 x 6830 matrix
table(nci.labs)
```

Cancer Type	Count
BREAST	7
CNS	5
COLON	7
K562A-repro	1
K562B-repro	1
LEUKEMIA	6
MCF7A-repro	1
MCF7D-repro	1
MELANOMA	8
NSCLC	9
OVARIAN	6
PROSTATE	2
RENAL	9
UNKNOWN	1

NCI60: Heatmap with Cancer Type Annotations

```
# Select high-variance genes (diagonal of cov matrix > 5)
idx  <- which(diag(cov(nci.data)) > 5)

# Assign a unique color to each cancer type
cols <- colorspace::rainbow_hcl(length(unique(nci.labs)))
cols <- factor(nci.labs, labels = cols)

# Heatmap with cancer type sidebar
heatmaply(nci.data[, idx],
          RowSideColors = as.character(cols),
          main          = "NCI60 Gene Expression Heatmap",
          xlab          = "Genes (high variance subset)",
          ylab          = "Cell Lines")
```

t-SNE: Step 1 — Modeling Similarities in High-D Space

Given n points $x_1, \ldots, x_n \in \mathbb{R}^p$, t-SNE defines a **conditional probability** that x_j is a neighbor of x_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \qquad p_{ii} = 0$$

and symmetrizes to form a joint probability:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

The Bandwidth σ_i

Each point i has its *own* bandwidth, chosen automatically so that the **perplexity**

$$\mathrm{Perp}(P_i) = 2^{H(P_i)}, \quad H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

matches a user-specified target value. Intuitively, perplexity $\approx$ effective neighborhood size.

Gaussian Kernel Intuition

- Nearby x_j: $\|x_i - x_j\|$ small $\Rightarrow$ $p_{j|i}$ **large**
- Distant x_j: $\|x_i - x_j\|$ large $\Rightarrow$ $p_{j|i} \approx 0$
- σ_i shrinks in **dense** regions (fine resolution) and grows in **sparse** regions
- Symmetrization ensures $p_{ij} = p_{ji}$, giving a well-defined joint distribution

t-SNE: Step 2 — Embedding in Low-D & the Objective

Let $y_1, \ldots, y_n \in \mathbb{R}^2$ be the low-dimensional map points. Their similarities are modeled with a **Student-t distribution** (df $= 1$, i.e. a Cauchy):

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}, \qquad q_{ii} = 0$$

Objective: KL Divergence

t-SNE minimizes the **Kullback–Leibler divergence** between P and Q:

$$\mathcal{L} = \mathrm{KL}(P \| Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Minimized via **gradient descent** w.r.t. the map coordinates $\{y_i\}$.

Why Student-t in Low-D?

Aspect	Explanation
Heavy tails	Moderate distances in 2D can represent large distances in high-D without compression
Crowding fix	Gaussian q_{ij} would "crowd" medium-distance points near the origin.

t-SNE: Algorithm, Complexity & Key Hyperparameters

Algorithm Summary

1. Compute pairwise affinities p_{ij} in high-D (bandwidth set by perplexity via binary search)
2. Initialize map points $y_i^{(0)}$ randomly (or from PCA)
3. **For** $t = 1, \ldots, T$ iterations:
   (a) Compute q_{ij} from current $\{y_i\}$
   (b) Evaluate gradient $\partial \mathcal{L} / \partial y_i$
   (c) Update $y_i \leftarrow y_i - \eta \nabla_{y_i} \mathcal{L}$ with momentum
4. Return final map coordinates $\{y_i\}$

Complexity

- Naïve: $\mathcal{O}(n^2)$ per iteration (all pairs)
- Barnes-Hut approximation (`Rtsne` default): $\mathcal{O}(n \log n)$ per iteration — feasible up to $n \approx 10^5$

Key Hyperparameters

Parameter	Guidance
`perplexity`	Typical range 5–50; larger $\Rightarrow$ more global structure captured. Try 5, 15, 30.
`max_iter`	$\geq$1000 recommended; 300 may be too few for large n
`eta`	Learning rate; `Rtsne` default `"auto"` ($= n/12$) usually sufficient
`theta`	Barnes-Hut accuracy (0 = exact); default 0.5
`pca`	Pre-whiten to top-50 PCs before t-SNE

NCI60: t-SNE Dimensionality Reduction

```
library(Rtsne)
library(ggplot2)

# t-SNE embedding: 6830D -> 2D
nc60.tsne <- Rtsne(nci.data,
  dims       = 2,
  perplexity = 5,
  verbose    = TRUE,
  max_iter   = 300)

# Build data frame for plotting
embedding
<- as.data.frame(nc60.tsne$Y)
embedding$Class <- as.factor(nci.labs)
colnames(embedding)[1:2] <- c("Dim1", "

ggplot(embedding, aes(Dim1, Dim2, color
```

What is t-SNE?

t-Distributed Stochastic Neighbor Embedding is a nonlinear dimensionality reduction technique that:

- Maps high-dimensional data to 2D (or 3D)
- Preserves **local structure**: nearby points in high-D remain nearby in 2D
- Useful for visualizing cluster structure

t-SNE Caveats

- **Perplexity** (5–50) controls the neighborhood size; try multiple values
- Distances **between** clusters are not meaningful

Lecture Summary

Hierarchical Clustering

- Agglomerative algorithm: $\mathcal{O}(n^3)$ complexity; no K required
- Output is a **dendrogram**; cut at height h to obtain K clusters
- **Linkage** determines inter-cluster dissimilarity:
  - Complete: compact clusters
  - Single: can chain
  - Average / Ward's: recommended defaults
- **Dissimilarity measure** choice:
  - Euclidean: magnitude
  - Correlation: shape

Heatmaps

- Encode matrix data as **colored grid**; rows & columns ordered by clustering
- Always **scale** when features are on different units
- Use **diverging** palette for z-scores; **sequential** for raw counts
- **Row/col sidebars** annotate with metadata for validation