Spatial Data Analysis

Week 11: Point Pattern Data II

Meredith Franklin

Department of Statistical Sciences and School of the Environment

November 14, 2025

Outline

- Cluster point patterns
- Fitting point process models
- Machine learning for cluster detection: k-means, DBSCAN and HDBSCAN

Review: CSR, HPP, IPP

Definitions & properties

- CSR $\equiv$ HPP on W: constant λ, independent counts.
- IPP: $\lambda(s)$ varies with s (covariates/trend); conditional independence.
- $N(A) \sim \mathrm{Pois}(\mu(A))$; $\mu(A) = \lambda|A|$ (HPP) or $\int_A \lambda(s)\,ds$ (IPP).
- Likelihood $\propto \exp\big(-\int_W \lambda\big) \prod_i \lambda(x_i)$.
- CSR: $K(r) = \pi r^2$, $g(r) = 1$; $K > \pi r^2$ cluster, $K < \pi r^2$ inhibition.

Estimation & diagnostics

- HPP: $\hat{\lambda} = n/|W|$.
- IPP: $\log \lambda(s) = \beta_0 + \beta^\top z(s)$.
- Tests: quadrat; envelopes for K/L, G, H, etc.
- Inhomog.: $K_{\text{inhom}}, G_{\text{inhom}}$; residual maps.

Cluster and Inhibition Processes

- Building block: Poisson process.
- Add-on: dependence between points.
- **Cox Process**: clustering arises because the intensity function is random; reflects a continuous latent field (doubly stochastic) and is more general.
- **Poisson Cluster Process (PCP)**: parent–offspring mechanism.
- **Inhibition (SIP)**: repulsion via interaction; regular spacing/competition.
- Example CSR baseline: $K(r) = \pi r^2$, $L(r) = r$
- Example Clustering: $K(r) > \pi r^2$ (or $L(r) > r$)
- Example Inhibition: $K(r) < \pi r^2$ $(L(r) < r)$.

Cox Cluster Process

- A Cox process is a Poisson process driven by a random intensity surface $\Lambda(s)$.
- If two nearby locations s and $s+r$ tend to have simultaneously high intensity on that surface, then points are more likely to occur at both locations together—this is clustering.
- The "pair correlation" measures how often pairs at distance r occur relative to a Poisson baseline.
- **Cox process:** Poisson with *stochastic intensity* $\Lambda(s) \geq 0$. Conditional on Λ, points are IPP; clustering reflects

$$g(r) = 1 + \frac{\mathrm{Cov}\{\Lambda(s), \Lambda(s+r)\}}{\lambda^2}, \quad \lambda = \mathbb{E}[\Lambda(s)].$$

- $\lambda = \mathbb{E}[\Lambda(s)]$ is the (mean) intensity.
- $g(r)$ is the pair correlation function (PCF).
- For stationary processes, $g(r) = \frac{\rho^{(2)}(r)}{\lambda^2}$, where $\rho^{(2)}(r)$ is the second-order product density at separation r. Note $\lambda(s)$ is first order intensity (could be written $\rho^{(1)}(s)$). Second order is joint intensity $\rho^{(2)}(s,t)$ expected number of ordered pairs with one point near s and one near t (for $s \neq t$).

Cox Cluster Process

- Why $g(r) = 1 + \mathrm{Cov}\{\Lambda(s), \Lambda(s+r)\}/\lambda^2$ for Cox?
- Draw a random intensity field $\Lambda(s) \geq 0$; given Λ, points are IPP.
- Definitions: $\lambda = \mathbb{E}[\Lambda(s)]$; for stationary processes $g(r) = \rho^{(2)}(r)/\lambda^2$.
- Derivation:

$$\rho^{(2)}(r) = \mathbb{E}\big[\rho^{(2)}(s, s{+}r \mid \Lambda)\big] = \mathbb{E}\big[\Lambda(s)\Lambda(s{+}r)\big] = \lambda^2 + \mathrm{Cov}\{\Lambda(s), \Lambda(s{+}r)\}.$$

$$\Rightarrow \quad g(r) = \frac{\rho^{(2)}(r)}{\lambda^2} = 1 + \frac{\mathrm{Cov}\{\Lambda(s), \Lambda(s{+}r)\}}{\lambda^2}.$$

- Interpretation: positive covariance at small $r \Rightarrow g(r) > 1$ (clustering); as covariance $\to 0$, $g(r) \to 1$ (Poisson-like).

Clustered Point Processes: Two Routes

- **Cox process:** Poisson with *stochastic intensity* $\Lambda(s) \geq 0$. Conditional on Λ, points are IPP; clustering reflects

$$g(r) = 1 + \frac{\mathrm{Cov}\{\Lambda(s), \Lambda(s+r)\}}{\lambda^2}, \quad \lambda = \mathbb{E}[\Lambda(s)].$$

- **PCP (a special case of Cox):**

$$\Lambda(s) = \mu \sum_i h\big(s - s_i^{(P)}\big),$$

where parents P and their locations $\{s_i^{(P)}\}$ are Poisson with rate $\rho(s)$, N_i offspring/parent with $\mathbb{E}[N_i] = \mu$, and h is a kernel.

- **LGCP (another Cox subtype):** $\Lambda(s) = \exp(Z(s))$, Z Gaussian.

PCP: Definition

- A Poisson Cluster Process (PCP) generates events in $D \subset \mathbb{R}^2$ as **clusters** around **parent** points.
- Parent points form a Poisson process with intensity $\rho(s)$ (homogeneous: ρ constant).
- Each parent at location $s_i^{(P)}$ produces a random number of offspring N_i (iid; mean $\mathbb{E}[N_i] = \mu$).
- Offspring locations are $s_{ij} = s_i^{(P)} + u_{ij}$, where displacements u_{ij} are iid with pdf $h(u)$.
- Typical applications: seed/seedling patterns, insect larvae, galaxy clustering.

PCP: Mechanism

- **Parents:** HPP/IPP with rate ρ (or $\rho(s)$). Each parent i has iid offspring count N_i with $\mathbb{E}[N_i] = \mu$.
- **Displacements:** offspring offsets u_{ij} iid with pdf $h(\cdot)$; locations $s_{ij} = s_i^{(P)} + u_{ij}$.
- **Intensity field:**

$$\Lambda(s) = \mu \sum_i h\big(s - s_i^{(P)}\big), \qquad \text{so } \lambda(s) = \mathbb{E}[\Lambda(s)] = \rho(s)\,\mu \text{ (homog.: } \lambda = \rho\mu\text{)}.$$

- **PCP examples: Thomas** (Gaussian h), **Matérn cluster** (uniform-in-disk h). Typical for parent–offspring mechanisms.
- `spatstat`: simulate `rThomas`, `rMatClust`.

PCP: Notation and First-Order Intensity

- Parent intensity: $\rho(s)$ on D.
- Cluster size: N_i with mean μ (e.g., Poisson, geometric, fixed).
- Displacement kernel: $h(u)$, a bivariate pdf $(\int_{\mathbb{R}^2} h(u)\,du = 1)$.
- **Offspring intensity:** $\lambda(s) = \mu\,\rho(s)$ (homogeneous case: $\lambda = \mu\rho$).
- Isotropic kernels: $h(u) = \tilde{h}(\|u\|)$; e.g., Gaussian with variance σ^2.

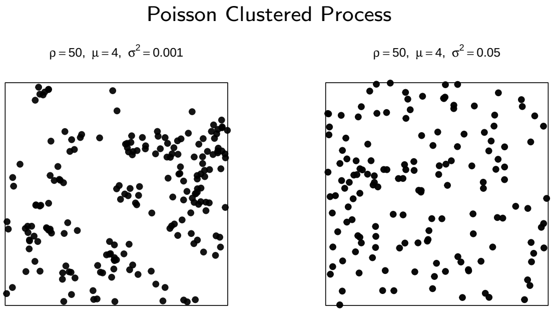
PCP: Displacement Kernels $h(u)$

- **Gaussian (Thomas process):**

$$h(u) = \frac{1}{2\pi\sigma^2} \exp\Big(- \frac{\|u\|^2}{2\sigma^2} \Big).$$

- **Uniform-in-disk (Matérn cluster):** $h(u) = \frac{1}{\pi R^2}$ for $\|u\| \leq R$, else 0.
- Interpretation: σ^2 or R sets cluster spread; ρ controls number of clusters; μ controls cluster size.

PCP Example: Thomas Process



- D: unit square. Parents: $\rho = 50$. Cluster size: $E[S] = 4$.
- Left: smaller spread (e.g., Gaussian h with $\sigma^2 = 0.001$) ⇒ tight clusters.
- Right: larger spread (e.g., $\sigma^2 = 0.05$) ⇒ diffuse clusters.

PCP: Second-Order (Stationary/Isotropic)

Assume parent HPP with constant ρ; isotropic kernel h; cluster size N with $\mathbb{E}[N] = \mu$. Let $\lambda = \mu\rho$. For separation vector v with $r = \|v\|$:

$$\lambda_2(s, s{+}v) = \lambda^2 + \rho\,\mathbb{E}[N(N{-}1)]\;(h * h)(v), \quad (h * h)(v) = \int h(u)\,h(v{-}u)\,du.$$

Pair correlation: $g(r) = 1 + A\,(h * h)(r),\; A = \dfrac{\mathbb{E}[N(N{-}1)]}{\mu^2\,\rho}$.

$$K(r) = \pi r^2 + A\int_0^r 2\pi t\,(h * h)(t)\,dt.$$

Recall Poisson (CSR) baseline: $g(r) = 1$, $K(r) = \pi r^2$.

Nonstationary note: if parents have intensity $\rho(s)$, then $\lambda(s) = \mu \int h(s{-}u)\rho(u)\,du$ and use $K_{\text{inhom}}, g_{\text{inhom}}$.

Neyman–Scott: PCP Special Case

- Neyman Scott assumptions of **homogeneous** Poisson cluster process:
  - Parent events are realizations of a Poisson process with intensity ρ
  - Each parent i produces a random number of offspring N_i and the N_i are i.i.d.
  - The positions of offspring wrt the parent are i.i.d with bivariate pdf.
- **Neyman Scott (with Thomas Gaussian kernel, Poisson N):** $A = 1/\rho$, and

$$(h*h)(r) = \frac{1}{4\pi\sigma^2}\exp\Big(-\frac{r^2}{4\sigma^2}\Big), \quad g(r) = 1 + \frac{1}{\rho}\cdot\frac{1}{4\pi\sigma^2}\exp\Big(-\frac{r^2}{4\sigma^2}\Big).$$

- **Matérn (uniform disk radius R, Poisson N)):** $A = 1/\rho$; $(h*h)(r)$ is the area-overlap kernel of two radius-R disks.
- Larger σ or $R \Rightarrow$ broader clusters; larger $\rho \Rightarrow$ more clusters but weaker $g(r)-1$.

spatstat Pointers (PCP)

- **Simulate (PCP):** `rThomas(kappa=`ρ`, mu=`μ`, sigma=`σ`)`, `rMatClust(kappa, mu, r)`.
- **Fit (PCP):** `kppm(X ~ 1, clusters="Thomas")` or `"MatClust"`.
- **Predict & simulate from fit:** `lam` $\leftarrow$ `predict(fit, type="intensity")`, `simulate(fit, nsim=199)`.
- **Diagnostics:** global envelopes for K, L, g; intensity-weighted versions if trend suspected:
  - CSR envelopes: `envelope(X, Kest, nsim=199, global=TRUE)`.
  - Inhomogeneous: `Kinhom(X, lambda=lam)`, `pcfinhom(X, lambda=lam)`.
  - Residual-style checks (from fitted Cox/cluster model): simulate from `fit` and compare summaries.

Cox Processes & Log-Gaussian Cox Process

- **Cox process:** Poisson with *stochastic intensity* $\Lambda(s) \geq 0$. Conditional on Λ, points are IPP; clustering reflects $\mathrm{Cov}\{\Lambda(s), \Lambda(s+r)\}$ with

$$g(r) = 1 + \frac{\mathrm{Cov}\{\Lambda(s), \Lambda(s+r)\}}{\lambda^2}, \quad \lambda = \mathbb{E}[\Lambda(s)].$$

- **LGCP (a Cox subtype):** $\Lambda(s) = \exp(Z(s))$ with Gaussian field Z. If Z is stationary with mean μ_Z, variance σ_Z^2, correlation $\rho_Z(r)$:

$$\mathbb{E}[\Lambda] = \exp\big(\mu_Z + \tfrac{1}{2}\sigma_Z^2\big), \qquad g(r) = \exp\big(\sigma_Z^2 \rho_Z(r)\big).$$

- **Examples (LGCP):** species occurrences (latent habitat), disease cases (latent risk), crime hotspots, rainfall/lightning fields.

spatstat Pointers (LGCP)

- **Simulate (LGCP):** `rLGCP()` with a covariance model (e.g., Matérn/exponential):
  - X ← `rLGCP(model="matern", mu=`η`, var=`σ^2`, scale=`α`, nu=`ν`, win=W)`.
  - η is the log-mean intensity (so $E[\Lambda] = \exp(\eta + \sigma^2/2)$).
- **Fit (LGCP):** either
  - `lgcp(X ~ trend, covmodel="matern", ...)` (direct LGCP fit), *or*
  - `kppm(X ~ trend, clusters="LGCP")` (version-dependent shortcut).
- **Predict intensity**: `lam` ← `predict(fit, type="intensity")` ⇒ image (`im`) for mapping and diagnostics.
- **Diagnostics (inhomogeneous):**
  - Pair correlation: `pcfinhom(X, lambda=lam)`; Poisson baseline is $g(r) = 1$.
  - K-function: `Kinhom(X, lambda=lam)`; compare against simulations from the fitted LGCP:
  - `envelope(X, Kinhom, nsim=199, simulate=expression(simulate(fit)), global=TRUE)`.
- **Trend + covariates:** include covariates in the trend (e.g., `X ~ z1 + z2`); supply rasters via `as.im()`.
- **Practical tips:** start with a simple trend, choose Matérn covariance (flexible ν); check that estimated $g(r)$ shape matches model-implied $g(r) = \exp(\sigma_Z^2 \rho_Z(r))$.

LGCP vs PCP

- LGCP: $\Lambda(s) = \exp(Z(s))$, with Gaussian field Z.
- If Z stationary: mean μ_Z, var σ_Z^2, corr $\rho_Z(r)$:

$$\mathbb{E}[\Lambda(s)] = \exp\bigl(\mu_Z + \tfrac{1}{2}\sigma_Z^2\bigr), \qquad g(r) = \exp\bigl(\sigma_Z^2 \rho_Z(r)\bigr).$$

- PCP induces $g(r) = 1 + A(h * h)(r)$ (additive bump); LGCP induces multiplicative $g(r) = \exp(\cdot)$.

Fitting Point Process Models: Overview

- Data: observed points $\{x_1, \ldots, x_n\}$ in region D.
- Choose a model class:
  - **IPP (trend only)**: $\lambda(s)$ varies with covariates/trend.
  - **Clustered Cox**:
    - **PCP / Neyman–Scott (Thomas, Matérn)**: parent→offspring mechanism.
    - **LGCP**: smooth latent field $\Lambda(s) = \exp(Z(s))$.
- Estimation routes:
  - **IPP**: log-linear likelihood via `ppm`.
  - **PCP/LGCP**: minimum contrast (least-squares in K or g) via `kppm`.
  - **Gibbs/interaction (e.g., Strauss)**: *pseudolikelihood* via `ppm`.
- Checking: $K/L/g$ envelopes (CSR or inhomogeneous), intensity-reweighted versions if trend present; simulate from fitted model.

Fit Inhomogeneous Poisson Process (IPP) via `ppm`

- Log-linear trend: $\log\lambda(s) = \beta_0 + \beta^\top z(s)$ (coordinates and/or covariates).
- spatstat: `ppm(X ~ z1 + z2 + x + y)`.
- Computation: maximizes the **log likelihood**

$$L(\theta \mid x) = \exp\Big(-\int_D \lambda(u;\theta)\,du\Big)\prod_{i=1}^{n}\lambda(x_i;\theta).$$

$$\ell(\theta|x) = \sum_{i=1}^{n}\log\lambda(x_i;\theta) - \int_D \lambda(u;\theta)\,du,$$

- Diagnostics: `Kinhom`, `pcfinhom`, residual maps; envelopes with `simulate=expression(rpoispp(^lambda))`.

PCP (Thomas, Matérn) via `kppm` (Minimum Contrast)

- Model: parents Poisson with rate ρ (or $\rho(s)$); each parent i has iid N_i with $\mathbb{E}[N_i] = \mu$; offsets $u_{ij} \sim h(\cdot)$.
- Homogeneous case: $\lambda = \mu\rho$; Thomas: Gaussian h (spread σ); Matérn: uniform-in-disk (R).
- **Fit in** `spatstat`:
  - `fitT <- kppm(X ~ 1, clusters="Thomas", method="mincontrast")`
  - `fitM <- kppm(X ~ 1, clusters="MatClust", method="mincontrast")`
- Statistic matched: by default $K(r)$ (or choose `statistic="pcf"`).
- Typical parameters returned: $\hat{\rho}$ (parent rate), $\hat{\mu}$ (mean cluster size), $\hat{\sigma}$ or $\hat{R}$ (spread).
- Check: `simulate(fitT)`, `envelope(X, Kest, simulate=expression(simulate(fitT)), global=TRUE)`; also compare `pcf`.

LGCP via `kppm` (Minimum Contrast)

- LGCP: $\Lambda(s) = \exp(Z(s))$; Z Gaussian with mean μ_Z, variance σ_Z^2, correlation (often Matérn) with range/scale α and smoothness ν.
- **Fit in** `spatstat` (minimum contrast to K or g):
  - `fitL <- kppm(X ~ trend, clusters="LGCP", method="mincontrast")`
- Interpreting parameters: $\mathbb{E}[\Lambda] = \exp(\mu_Z + \frac{1}{2}\sigma_Z^2)$, $g(r) = \exp(\sigma_Z^2 \rho_Z(r))$ (scale/smoothness shape g).
- Diagnostics: `pcf` or `pcfinhom` (if trend), envelopes via `simulate(fitL)`; intensity map: `predict(fitL, type="intensity")`.

Minimum Contrast: What `kppm` Optimizes

- Match summary function to theory by weighted least squares:

$$D(\theta) = \int_0^{r_{\max}} w(r) \left\{ \hat{S}(r)^c - S(r;\theta)^c \right\}^2 dr,$$

  where $S \in \{K, g\}$, c is a power transform (variance stabiliser), $w(r)$ is a weight.
- Choices:
  - $S = K$: cumulative departures (broad clustering).
  - $S = g$: scale-specific departures (local range).
  - $c \approx 0.5$ (near-CSR), $c \approx 0.25$ (strong clustering); try a few to assess sensitivity.
  - $r_{\max}$: typically $\sim$ 1/3–1/2 of the shorter window side.
- Example (Thomas): $K(r;\rho,\sigma) = \pi r^2 + \frac{1}{\rho}\left(1 - e^{-r^2/(4\sigma^2)}\right)$.
- Reference: Guan & Sherman (2007, JRSS B) for asymptotics; Baddeley et al. (2015) for practice.

Example: Fit Thomas PCP by K-contrast

Model: Thomas PCP with parameters $\theta = (\rho, \mu, \sigma)$ (ρ=parent intensity; μ=mean offspring/parent; σ=spread).

Intensity: $\lambda = \mu \rho$ (homogeneous).

Theoretical K (Thomas):

$$K_{\mathsf{Th}}(r; \rho, \sigma) = \pi r^2 + \frac{1}{\rho}\left(1 - e^{-r^2/(4\sigma^2)}\right) \quad \text{(independent of } \mu\text{).}$$

Criterion (minimum contrast):

$$D(\rho, \sigma) = \int_0^{r_{\max}} \left\{\hat{K}(r)^c - K_{\mathsf{Th}}(r; \rho, \sigma)^c\right\}^2 dr,$$

choose c (e.g., 0.5; use 0.25 for strong clustering) and $r_{\max} \approx 1/3$–$1/2$ of the shorter window side.

Estimation: minimise $D(\rho, \sigma)$; then $\widehat{\mu} = \widehat{\lambda}/\widehat{\rho}$ with $\widehat{\lambda} = n/|D|$.

Note, we use ρ in the slides but it is ‘kappa‘ in spatstat.

Testing & Diagnostics: What to Plot

- **CSR null**: `envelope(X, Kest)` and `envelope(X, pcf)` (global envelopes recommended).
- **Beyond trend**: fit IPP $\hat{\lambda}$, then `Kinhom(X, lambda=ˆ` λ`)`, `pcfinhom(X, lambda=ˆ` λ`)`; envelopes with `simulate=expression(rpoispp(ˆ` λ`))`.
- **Model-based**: compare to `simulate(fit)` for PCP/LGCP; plot `pcf` vs theoretical from `pcfmodel(fit)`.
- **Bandwidth for** g: use `bw.pcf(X)` to avoid over/under-smoothing.
- **Edge correction**: use `correction="iso"` (or translation) and restrict r to safe range.

Code Cheatsheet (spatstat)

- **IPP (trend)**: `fitI <- ppm(X ~ z1 + z2); lam <- predict(fitI, type="trend")`.
- **PCP (Thomas/Matérn)**: `fitT <- kppm(X ~ 1, clusters="Thomas"); fitM <- kppm(X ~ 1, clusters="MatClust")`.
- **LGCP**: `fitL <- kppm(X ~ trend, clusters="LGCP")`.
- **Envelopes**: `envelope(X, Kest, nsim=199, global=TRUE); envelope(X, pcfinhom, simulate=expression(rpoispp(lam))); envelope(X, Kest, simulate=expression(simulate(fitT)))`.
- **Pair correlation**: `pcf(X)` (homog), `pcfinhom(X, lambda=lam)` (inhom); theoretical: `pcfmodel(fitT)`.
- **Residuals/diagnostics**: `diagnose(fitI)` or `diagnose(fitT)`.

From Point Processes to Machine Learning

Spatial Clustering

- **Point process view:** a generative model for events in space (e.g., PCP, LGCP); clustering is inferred from $K(r), g(r)$, parameters (ρ, μ, σ), and model comparison.
- **ML clustering view:** an algorithmic partition of observed points into groups $+$ noise, without specifying a generative process.
- **Bridge:** where $g(r)>1$ at short ranges (excess close pairs), ML methods search for *locally dense* groups consistent with that signal.
- **Goal shift:** PP $\Rightarrow$ estimate mechanisms & parameters; ML $\Rightarrow$ discover structure, summarize shapes, detect anomalies.

Core Ingredients in ML Spatial Clustering

- **Inputs:** coordinates $s = (x, y)$ and optionally features $z(s)$.
- **Similarity:** a distance/affinity on points (Euclidean on projected coords; or geodesic for lon/lat).
- **Scale:** a neighborhood bandwidth (explicit or implicit) controlling what "dense" means.
- **Outputs:** cluster labels and possibly noise/outliers; sometimes soft membership or stability.
- **Validation:** internal indices (silhouette), stability under resampling, and spatial diagnostics (e.g., Moran's I within clusters).
- **Good practice:** project to a planar CRS for meters, standardize heterogeneous features, set sensible spatial extents to reduce edge effects.

Method Families for Spatial Clustering

- **Centroid-based:** k-means
- **Density-based:** *DBSCAN* (radius ε, MinPts) finds high-density regions and labels noise; *HDBSCAN* builds a density hierarchy (no global ε) and selects stable clusters.
- **Hierarchical (agglomerative):** linkage on distances; dendrogram cut chooses granularity.
- **Model-based:** Gaussian Mixture Models (elliptical clusters; EM; choose k by AIC/BIC).
- **Graph/spectral:** build a kNN or radius graph and partition via eigenvectors; handles nonconvex shapes (parameter: neighborhood size).
- **Spatially constrained:** enforce adjacency/contiguity (e.g., region merging, spatial Ward, SKATER) when clusters must be connected areas.

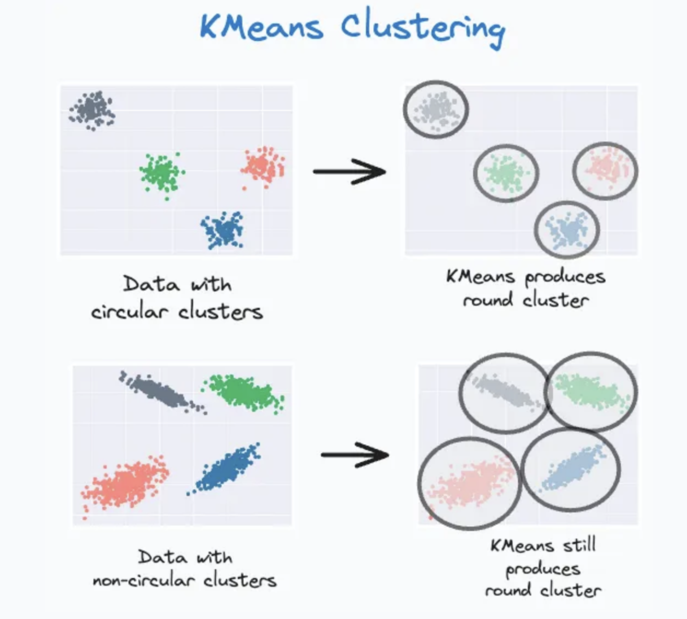
From PP Clues to ML Choices

- **Use PP diagnostics to set scale:** if $g(r)$ peaks near r^*, start DBSCAN with $\varepsilon \approx r^*$; for varying density, prefer HDBSCAN.
- **Trend vs clustering:** if intensity varies strongly with s, de-trend or include features $z(s)$ before clustering; otherwise dense regions may reflect sampling effort, not structure.
- **Noise handling:** density-based methods provide an explicit noise label—useful when PP analysis suggested inhibition or sparsity between groups.
- **Report both:** ML clusters (maps, sizes, stability) *and* PP summaries $(K/L/g)$ to justify the chosen scale and to compare alternative clusterings.

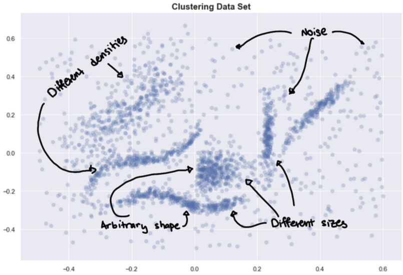
Density-Based Clustering (DBSCAN/HDBSCAN)

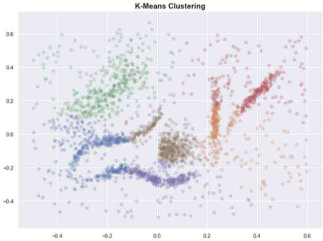
- **Idea:** clusters are regions of high point density separated by low-density gaps.
- **DBSCAN** (Ester et al., 1996): requires radius ε and `MinPts`; discovers arbitrary shapes and flags noise.
- **HDBSCAN**: hierarchical extension—*no global* ε; builds a density tree and extracts stable clusters.
- Contrast to k-means: no centroids, no k chosen in advance, robust to outliers; shapes are non-convex.

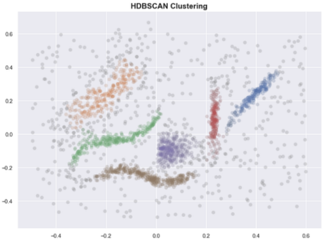
k-means Spatial Clustering


KMeans Clustering
Data with circular clusters
KMeans produces round cluster
Data with non-circular clusters
KMeans still produces round cluster

k-means vs DBSCAN


Clustering Data Set


K-Means Clustering


HDBSCAN Clustering

https://pberba.github.io/stats/2020/07/08/intro-hdbscan/

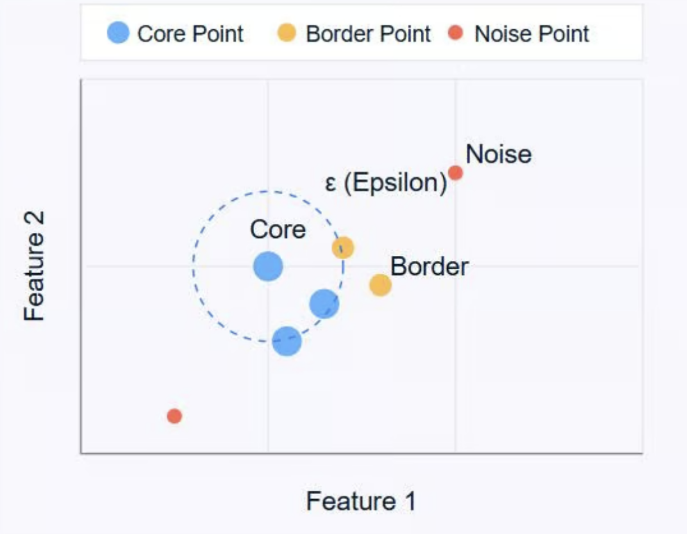
DBSCAN: Core Definitions

Given distance $d(\cdot,\cdot)$, radius $\varepsilon > 0$, and $\texttt{MinPts}$:

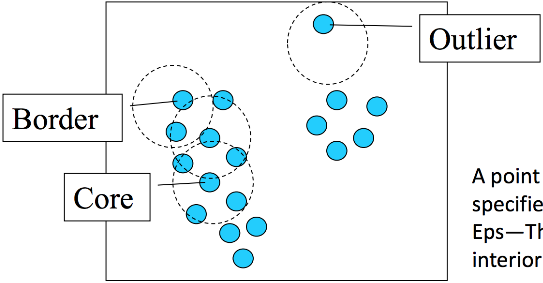
- **ε-neighborhood**: $N_\varepsilon(x) = \{y : d(x,y) \leq \varepsilon\}$. The count of neighbors within radius ε.
- **Core point:** $|N_\varepsilon(x)| \geq \texttt{MinPts}$. This is how a core point is determined. $|N_\varepsilon(x)| < \texttt{MinPts}$ says x does not have enough neighbors to be a core point (it will be noise or a border of a different core point).
- **Border point:** not core, but within ε of a core.
- **Noise**: neither core nor border.
- **Directly density-reachable**: $y \in N_\varepsilon(x)$ with x core.
- **Density-reachable**: connected by a chain of directly density-reachable steps.
- **Cluster**: maximal set of points density-connected to each other.

Density Based Clustering

The ε defines the neighborhood, where points within ε radius from another point are considered part of a cluster as long as it fulfills that there are a certain number of MinPts.



Density Based Clustering



ε = 1unit, MinPts = 5

Given ε and *MinPts*, categorize the objects into three exclusive groups.

A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

DBSCAN: Algorithm and Parameters

Algorithm (sketch):

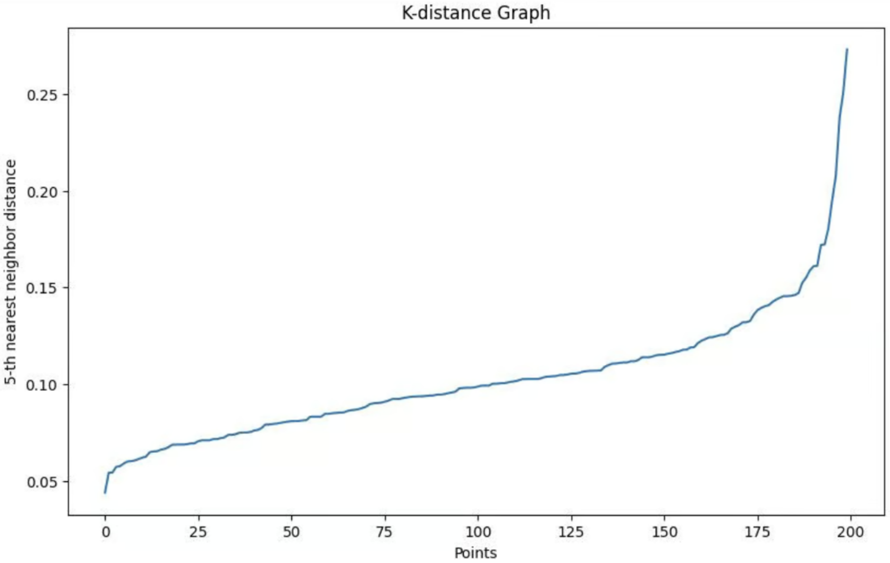
1. Visit an unassigned point x; if $|N_\varepsilon(x)| < \texttt{MinPts}$, mark as noise (may be upgraded to border).
2. If core, start a new cluster and iteratively absorb all points density-reachable from x.

Parameters:

- `MinPts`: rule-of-thumb $\geq d+1$ in d dimensions (e.g., 4–5 in 2D), larger in noisy data.
- ε: pick via the k-**distance plot** (with $k = \texttt{MinPts}$); choose the "elbow."

Complexity: $O(n \log n)$ with spatial index (kd/ball tree); $O(n^2)$ naive.

DBSCAN k-distance

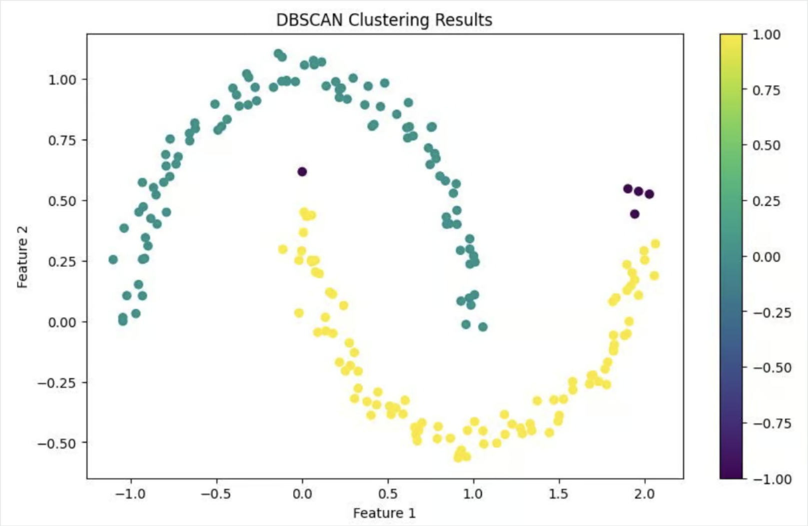


pick $\varepsilon = 0.15$ based on this plot

DBSCAN: Practical Considerations

- **Distance:** choose a metric appropriate to the data; for latitude/longitude either project (e.g., UTM) or use geodesic distances.
- **Scaling:** standardize heterogeneous features before Euclidean distance.
- **Strengths:** arbitrary shapes, automatic noise detection, no k.
- **Limitations:** single global $(\varepsilon, \texttt{MinPts})$ struggles with *varying density*; degraded in high dimension.

DBSCAN output



2 clusters and 5 noise points

DBSCAN and HDBSCAN

- Here is a good visualization of DBSCAN results: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/
- A better version in Hierarchical DBSCAN (HDBSCAN), where we do not need to set the ε neighborhood. With HDBSCAN we do set the minimum number of points that we wish to have in a cluster (2 to n). A larger MinPts means larger clusters.
  - HDBSCAN uses the concept of mutual reachability, where we look at distances that connect all of the points
  - A tree (hierarchy) is formed based on these distances

HDBSCAN: Why and How

Motivation: avoid a global ε; handle clusters of *varying density*.

Key constructs:

- **Core distance** (order k): $c_k(x) =$ distance to the k-th nearest neighbor ($k = \texttt{MinPts}$).
- **Mutual reachability distance:**

$$m_k(x,y) = \max\{c_k(x), c_k(y), d(x,y)\}.$$

- Build an MST (minimum spanning tree) on the graph weighted by $m_k(\cdot,\cdot)$; varying the density level induces a **cluster hierarchy**.

Minimum Spanning Tree (MST)

Definition. For a connected, undirected, weighted graph $G=(V,E,w)$, a minimum spanning tree is a subset of edges $T \subseteq E$ that (i) connects all vertices (spans), (ii) has no cycles (is a tree), and (iii) has minimum total weight $\sum_{e\in T} w(e)$ among all spanning trees.

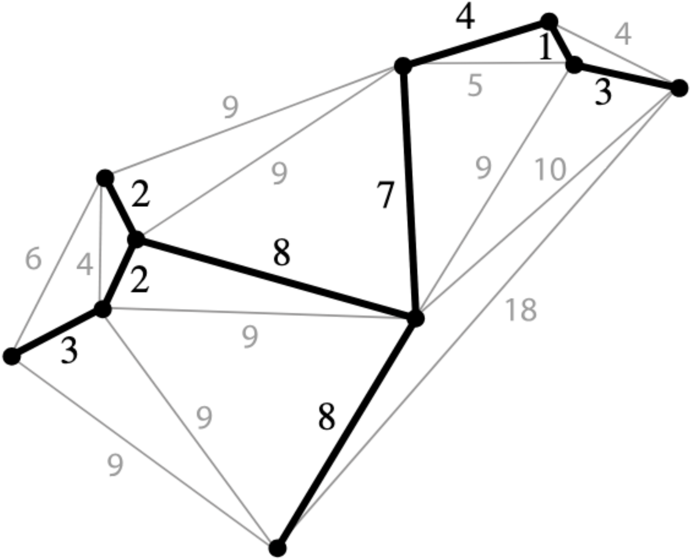
Key facts

- $|T| = |V| - 1$ edges; acyclic; connects all nodes.
- **Uniqueness:** if all edge weights are distinct, the MST is unique.
- **Cut property:** for any cut of V, the minimum-weight edge crossing the cut belongs to *some* MST.
- **Cycle property:** in any cycle, the maximum-weight edge cannot belong to *any* MST.
- **Algorithms:** Kruskal's ($O(E\log E)$), Prim's ($O(E\log V)$), Borůvka's (parallel-friendly).

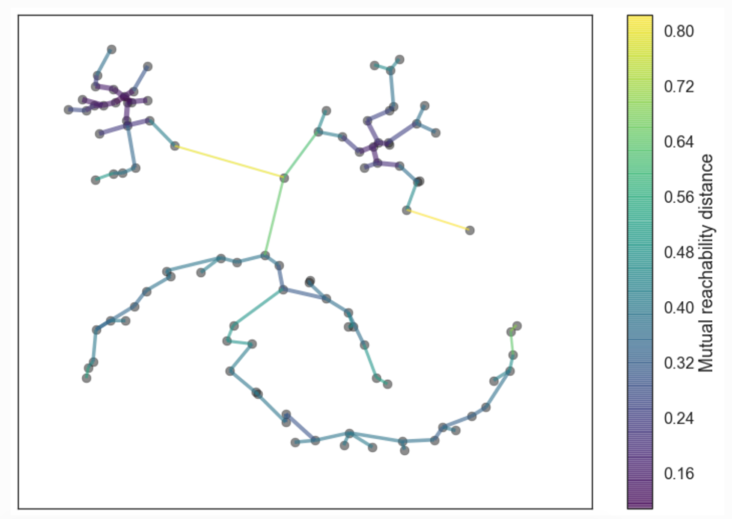
Why for clustering (DBSCAN/HDBSCAN)

- Single-linkage clustering = building an MST, then cutting long edges.
- HDBSCAN builds the MST of mutual reachability distances, then extracts stable clusters from the hierarchy.
- For large spatial datasets, sparsify with a kNN graph before MST.

Minimum Spanning Tree



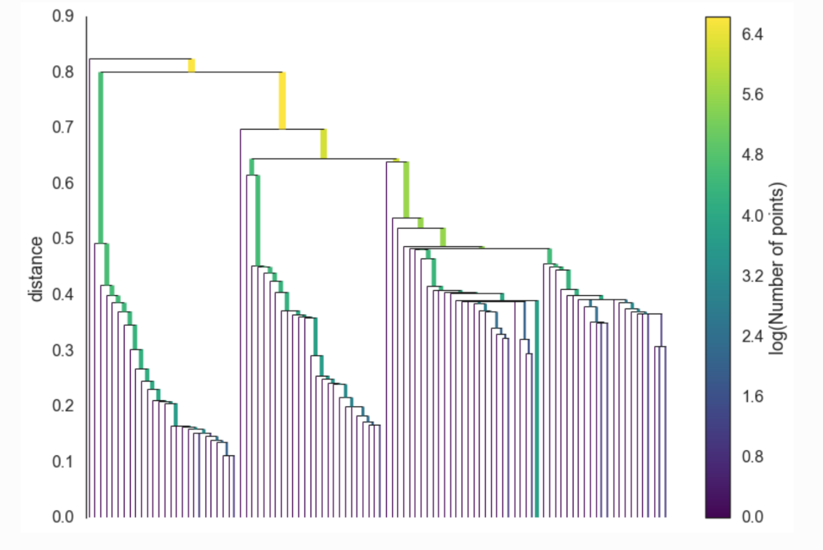
HDBSCAN Minimum Spanning Tree



HDBSCAN: From MST to Hierarchy

- **Input:** MST of *mutual reachability* distances $m_k(u,v)$ on the points.
- **Goal:** build a hierarchy (single-linkage dendrogram) of connected components.
- **Procedure (sweep on edge weights):**
  - Sort MST edges by distance (ascending).
  - Initialize each point as its own cluster (use a union–find/DSU structure).
  - For each edge (u,v,w) in order:
    - Find current clusters $C(u)$ and $C(v)$.
    - If $C(u) \neq C(v)$, **merge** them; record a dendrogram merge at level w.
  - Continue until all points are connected (one component).
- **Output:** a dendrogram where branch heights correspond to mutual reachability distance.
- **Notes:**
  - Cutting the dendrogram at a fixed level (fixed ε) reproduces DBSCAN.
  - HDBSCAN then **condenses** the tree using `min_cluster_size` and selects **stable** clusters.

HDBSCAN tree

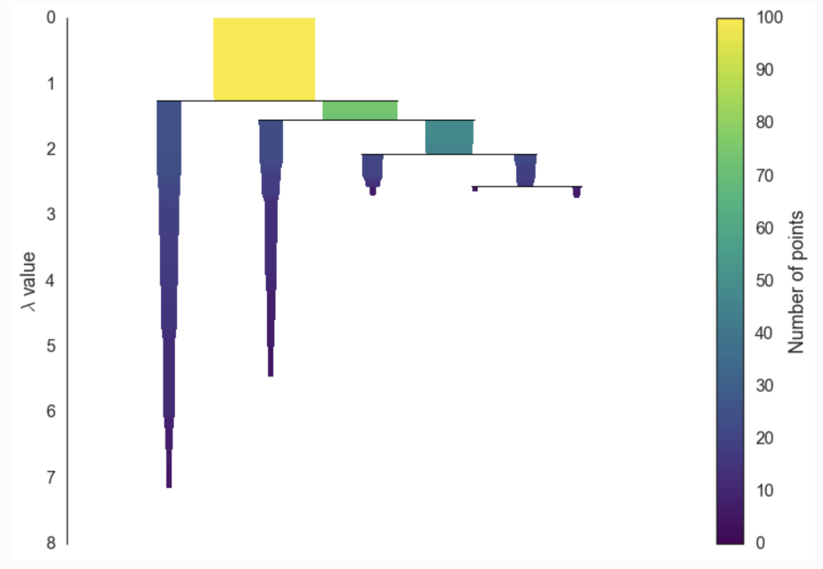


If we cut the tree at a given distance (ε) that is giving us the same estimate as DBSCAN

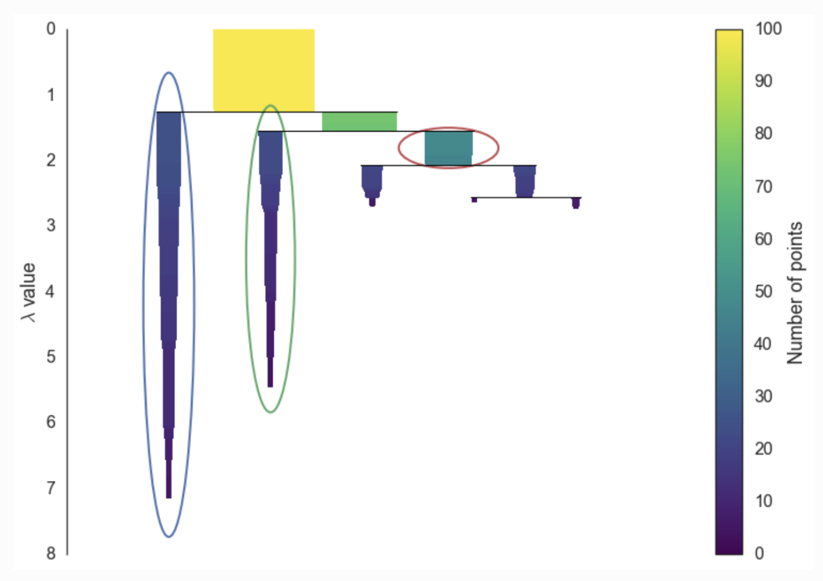
HDBSCAN: Condensed Tree and Cluster Selection

- Convert the hierarchy to a **condensed tree** using a `min_cluster_size` (often $\geq$ `MinPts`).
- At each split, discard branches with fewer than `min_cluster_size` points.
- **Select clusters** by maximizing **stability** (excess-of-mass): favor branches that *persist* over wide density ranges.
- Output: hard labels (clusters + noise), optional soft memberships and cluster *persistence* scores.

HDBSCAN condensed tree



HDBSCAN condensed tree



HDBSCAN: Cluster Stability

- Use density level $\lambda \equiv 1/(\text{mutual reachability distance})$.
- For each cluster C in the *condensed* tree:
  - **Birth** $\lambda_{\text{birth}}(C)$: first λ where $|C| \geq \texttt{min\_cluster\_size}$.
  - **Death** $\lambda_{\text{death}}(C)$: λ where C splits or disappears.
  - For each point $p \in C$, let λ_p be the λ when p leaves C (to a child or to noise); if it never leaves before C dies, set $\lambda_p = \lambda_{\text{death}}(C)$.
- **Stability** (a.k.a. excess of mass):

$$\mathrm{Stability}(C) = \sum_{p \in C} \big(\lambda_p - \lambda_{\text{birth}}(C)\big).$$

- **Selection rule** (dynamic programming on the tree): choose children if their total stability exceeds the parent's; otherwise keep the parent. This yields a non-overlapping set of *stable* clusters.

HDBSCAN: Membership Probability

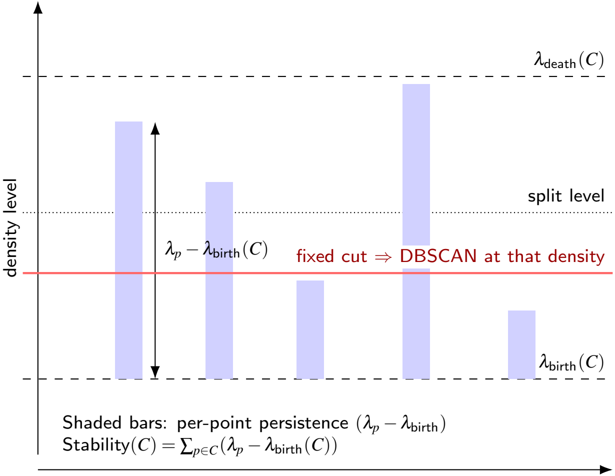
- After selecting clusters, assign a **probability** to each point p for its cluster C based on how long p *persists* in C above birth:

$$\Pr(p \in C) = \frac{\lambda_p - \lambda_{\text{birth}}(C)}{\lambda_{\text{death}}(C) - \lambda_{\text{birth}}(C)} \in [0,1],$$

  clipped to $[0,1]$. (Intuition: 1 if p stays with C for its whole lifetime; smaller if p peels off early.)
- Implementation details:
  - The Python `hdbscan` package computes equivalent *persistence-based* scores; formulas are normalized so the most persistent points get probability near 1.
  - Points never in any selected cluster get probability 0 (noise).
- Practical readout: report **cluster stability** (one score per cluster) and **membership probabilities** (one per point) to summarize confidence.

HDBSCAN: Cluster Stability Schematic

$\lambda = 1/(\text{mutual reachability})$


$\lambda_{\text{death}}(C)$
split level
density level
$\lambda_p - \lambda_{\text{birth}}(C)$
fixed cut ⇒ DBSCAN at that density
$\lambda_{\text{birth}}(C)$

Shaded bars: per-point persistence $(\lambda_p - \lambda_{\text{birth}})$
Stability$(C) = \sum_{p \in C}(\lambda_p - \lambda_{\text{birth}}(C))$

DBSCAN vs HDBSCAN

	DBSCAN	**HDBSCAN**
Parameters	ε, `MinPts`	`MinPts`, `min_cluster_size` (no global ε)
Density	Single global threshold	Varying density via hierarchy
Noise	Explicit label	Explicit label; soft membership
Shapes	Arbitrary	Arbitrary
When	Uniform-density clusters	Mixed-density clusters; parameter-robust

R Workflow: DBSCAN and HDBSCAN

- **Data prep** (project & extract coords):

```
library(sf); library(dbscan)
pts <- st_transform(pts_sf, 32610)  # project (ex: UTM zone 10N)
X   <- st_coordinates(pts)          # n x 2 matrix
```

- **Pick ε via k-NN distance plot** (use $k = \texttt{MinPts}$):

```
k <- 5
kNNdistplot(X, k = k); abline(h = 120, lty = 2)  # knee near 120 m -> eps
```

- **Run DBSCAN**:

```
mod_db <- dbscan(X, eps = 120, minPts = k)
table(mod_db$cluster)   # 0 = noise
```

- **Run HDBSCAN** (no eps):

```
mod_hd <- hdbscan(X, minPts = 5, minClusterSize = 12)
mod_hd$cluster_scores   # stability/persistence
```

- **Plot**: color by cluster; mark noise separately.
  Tip: scale non-spatial features before combining with coordinates.